Министерство образования Республики Башкортостан
ГАОУ СПО Стерлитамакский колледж строительства, экономики и права

Учебно-методический комплекс

по дисциплине

ОП. 05.«Основы программирования»

для групп специальности 230115
«Программирование в компьютерных системах»

Разработала преподаватель: ***В.Ф. Аришина***

2013

Одобрена на заседании предметно-цикловой комиссии специальности 230115 «Программирование в компьютерных системах»

Протокол №____ «___»_____2013г.

Председатель ПЦК_____О.А. Комиссарова

УТВЕРЖДАЮ

Зав. Методическим кабинетом ГАОУ СПО СКСЭиП

_____Н.Б. Дубанова

«____»_____2013г.

Учебно-методический комплекс по дисциплине «Основы программирования» по специальности 230115 «Программирование в компьютерных системах»

Составила	В.Ф. Аришина	Преподаватель математики и информатики ГАОУ СПО Стерлитамакский колледж строительства, экономики и права
Рецензенты:	Э.Р. Гиззатова	Кандидат физико-математических наук доцент кафедры «Математического моделирования» Стерлитамакского филиала Баш ГУ
	О.А. Комиссарова	Председатель предметно-цикловой комиссии специальности 230115 «Программирование в компьютерных системах» ГАОУ СПО Стерлитамакский колледж строительства, экономики и права

Оглавление

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Целью изучения дисциплины «Основы программирования» является приобретение студентами знаний о языке программирования Паскаль; умений писать программы для решения поставленной задачи, в том числе прикладной, привитие навыков сознательного и рационального использования ЭВМ в своей учебной и профессиональной деятельности. Полученные знания и умения помогут усвоить материал профессиональных модулей ПМ 01 «Разработка программных модулей программного обеспечения для компьютерных систем » и ПМ 02 «Разработка и администрирование баз данных» и сформировать общие компетенции и профессиональные компетенции ПК 1.1 - Выполнять разработку спецификаций отдельных компонент, ПК 1.2 - Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля, ПК 1.3 - Выполнять отладку программных модулей с использованием специализированных программных средств, ПК 1.4 - Выполнять тестирование программных модулей, ПК 1.5 - Осуществлять оптимизацию программного кода модуля, ПК 3.1 - Анализировать проектную и техническую документацию на уровне взаимодействия компонент программного обеспечения.

Изучение дисциплины способствует повышению уровня логического мышления и развивает способность принимать оптимальные решения и решать поставленную задачу рациональным способом. Полученные навыки и заложенные основы профессиональных компетенций помогут студентам в освоении профессиональных модулей ПМ 03 «Участие в интеграции программных модулей» и ПМ 04 «Выполнение работ по одной или нескольким профессиям рабочих, должностям служащих».

Освоение курса предполагает решение широкого круга задач и выполнение большого объема самостоятельной работы.

Изучение дисциплины «Основы программирования» включает: теоретическую базу в области основ написания программного кода, овладения навыками решения поставленной задачи с использованием компьютера. В ходе освоения дисциплины обучающиеся приобретают опыт анализа поставленной задачи, построения ее математической и компьютерной модели и составление оптимального алгоритма ее решения. В частности, студенты должны научиться читать, редактировать, составлять и отлаживать программы.

Применение учебно-методического комплекса позволит:

- Сформировать у студентов практические навыки по использованию вычислительной техники и программного обеспечения;
- Научить студентов основам программирования, включая постановку задачи, выбор метода решения задачи, создание или выбор алгоритма, реализацию алгоритма на языке программирования, отладку и тестирование программы;
- Приобрести опыт самостоятельной работы по освоению тонкостей программирования прикладных задач.

Учебно-методический комплекс курса "Основы программирования” представляет собой совокупность учебных, учебно-методических, контрольно-измерительных материалов, обеспечивающих организованную и содержательную целостность системы обучения по дисциплине, способствующих эффективному усвоению обучающимися учебной программы.

Учебно-методический комплекс по дисциплине «Основы программирования» предназначен для преподавателей общепрофессиональных дисциплин и профессиональных модулей по образовательным программам группы специальностей 230000 «Информатика и вычислительная техника».

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

ОП.05 Основы программирования

2011г.

Рабочая программа учебной дисциплины разработана на основе Федерального государственного образовательного стандарта (далее – ФГОС) по специальности среднего профессионального образования (далее СПО) **230115 Программирование в компьютерных сетях**, входящей в укрупненную группу специальностей 230000 Информатика и вычислительная техника.

Организация-разработчик: ГАОУ СПО Стерлитамакский колледж строительства, экономики и права

Разработчик: Аришина Вера Федоровна преподаватель первой квалификационной категории

Рекомендована _____

Заключение №_____ от «____»_____20__ г.

номер

СОДЕРЖАНИЕ

1. ПАСПОРТ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

Основы программирования

1.1. Область применения программы

Рабочая программа учебной дисциплины является частью основной профессиональной образовательной программы в соответствии с ФГОС по специальности **230115 Программирование в компьютерных системах**, входящей в состав укрупненной группы специальностей СПО **230000 Информатика и вычислительная техника** и может быть использована в дополнительном профессиональном образовании в рамках реализации программ переподготовки кадров в учреждениях СПО.

1.2. Место дисциплины в структуре основной профессиональной образовательной программы: дисциплина «Основы программирования» является общепрофессиональной дисциплиной профессионального цикла.

1.3. Цели и задачи дисциплины – требования к результатам освоения дисциплины:

В результате освоения дисциплины обучающийся должен **уметь**:

- Работать в среде программирования
- Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования;

В результате освоения дисциплины обучающийся должен **знать**:

- Этапы решения задачи на компьютере
- Типы данных
- Базовые конструкции изучаемых языков программирования
- Принципы структурного и модульного программирования
- Принципы объектно-ориентированного программирования

1.4. Рекомендуемое количество часов на освоение программы дисциплины:

максимальной учебной нагрузки обучающегося **216** часов, в том числе:

обязательной аудиторной учебной нагрузки обучающегося **144** часа;

самостоятельной работы обучающегося **72** часа.

2. СТРУКТУРА И ПРИМЕРНОЕ СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

2.1. Объем учебной дисциплины и виды учебной работы (дисциплина изучается в течение двух семестров)

Вид учебной работы	Объем часов
Максимальная учебная нагрузка (всего)	*216*
Обязательная аудиторная учебная нагрузка (всего)	*144*
в том числе:	
лабораторные занятия	*70*
контрольные работы	*10*
Самостоятельная работа обучающегося (всего)	*72*
в том числе:	
- поиск информации и составление опорного конспекта по работе изучаемой среды программирования, по теме: «отладки и тестирование»	*6*
- разработка программного кода по заданию в группе	*10*
- выполнение упражнений на анализ структуры готовой программы.	*2*
- написание программы по выданной задаче с четким выделением структурных элементов.	*6*
- написание процедур и функций пользователя по индивидуальному заданию	*6*
- проведение анализа учебной литературы и информации сети интернет и составление опорного конспекта «встроенные модули среды программирования	*6*
- ознакомление с нормативными документами по написанию программ и подготовка доклада.	*4*
- составление теста по теме «основные приемы работы в среде программирования», по принципам ООП	*8*
- составление конспекта ответов на контрольные вопросы по изученному материалу (что такое программирование, перечислите и охарактеризуйте методы программирования и пр.)	*4*
- создание презентации по теме «встроенные модули среды программирования», «особенности работы со встроенными модулями среды программирования»,	*8*
- составление сравнительной таблицы «модульное или структурное программирование», «виды программирования и их характеристика»	*8*
- подготовка к устным ответам по контрольным вопросам и анализу решенных задач.	*4*
Итоговая аттестация в форме	***экзамена***

2.2. Примерный тематический план и содержание учебной дисциплины «Основы программирования»

Наименование разделов и тем	Содержание учебного материала, лабораторные и практические работы, самостоятельная работа обучающихся	Объем часов	Уровень освоения
1	2	3	4
Раздел 1. Изучение конкретного языка программирования		***180***	
Тема 1.1. Интерфейс системы программирования	Содержание учебного материала.	*18*	
	1. Основы работы в среде программирования. (ОК 1)		*2*
	2. Методы программирования : структурное и модульное		
	3. Принципы разработки ПО.		
	4. Структура программы..		
	5 Алгоритм: блок-схема алгоритма, анализ алгоритма.		*2*
	6. Среда программирования . Основные приемы работы в среде программирования.		
	7. Функции среды программирования.		
	8. Операции с программным кодом: запуск и редактирование.		
	Контрольная работа № 1 «Основы работы в среде программирования»	*2*	
	Лабораторные занятия	*4*	
	1 Выполнение операций с кодом программы в среде программирования: открытие, отладка, сохранение. (ОК 2.1.1)		
	2 Выполнение операций с кодом программы в среде программирования: редактирование, сохранение.		
	Самостоятельная работа обучающихся	*11*	
	Составление теста по теме: «Основные приемы работы в среде программирования» Составление конспекта ответов на контрольные вопросы по изученному материалу (Что такое программирование, перечислите и охарактеризуйте методы программирования и пр.) Поиск дополнительной информации и составление опорного конспекта по работе изучаемой среды программирования (ОК 9)		
Тема 1.2. Современный язык программирования	Содержание учебного материала	*32*	
	1. Алфавит языка программирования. Типы данных.		*1*
	2. Операторы ввода-вывода. Простейшие операторы.		*1*
	3. Типы данных.		*2*
	4. Структурированные типы данных: строки		
	5. Реализация линейной и разветвляющейся алгоритмических конструкций в изучаемом языке программирования.		
	6. Реализация циклических алгоритмических конструкций в изучаемом языке программирования		
	7. Реализация основных алгоритмических конструкций. Составление программного кода по блок-схеме.		*2*
	8. Структурированные типы данных: массивы.		
	9. Методы сортировки массива.		
	10. Обработка символьного массива.		
	11. Обработка двумерного массива		*2*
	12. Тип данных записи.		
	13. Понятие подпрограммы. Процедуры.		*2*
	14. Подпрограммы: Функции.		*2*
	15. Программирование подпрограмм		
	16. Стандартные встроенные модули.		*1*
	Лабораторные занятия	*60*	
	1. Решение поставленной задачи в изучаемой системе программирования с использованием структурного программирования.		
	2. Реализация построенного алгоритма в виде программы . Элементы отладки программного кода.		
	3. Редактирование программного кода в соответствии со стандартами кодирования (ОК 3.1.1)		
	4. Реализация программ разветвляющейся структуры		
	5. Реализация программ циклической структуры		
	6. Реализация цикла с параметром.		

Наименование разделов и тем	Содержание учебного материала, лабораторные и практические работы, самостоятельная работа обучающихся	Объем часов	Уровень освоения
	7. Составление алгоритма задачи на использование цикла с предусловием		
	8. Составление алгоритма задачи на использование цикла с постусловием.		
	9. Составление и отладка программы по блок-схеме.		
	10 Реализация программ усложненной структуры.		
	11 Реализация способов обработки одномерных массивов		
	12 Поиск минимального(максимального) элемента в массиве.		
	13 Составление программы на использование строковых одномерных массивов		
	14 Решение задач на обработку строк.		
	15 Выполнение сортировки массивов		
	16 Решение поставленной задачи с использованием сортировки массива. (ОК 2.1.2)		
	17 Решение задач с использованием двумерных массивов		
	18 Решение задач на обработку двумерных массивов.		
	19 Поиск максимального (минимального) элемента в двумерном массиве		
	20 Использование стандартных функций для работы со строками и массивами		
	21 Сортировка двумерного массива		
	22 Решение задач с использованием записей		
	23 Решение задач с использованием записей		
	24 Организация процедур при решении задач		
	25 Организация функций при решении задач		
	26 Использование подпрограмм при решении задач		
	27 Составление программ для решения конкретной задачи с использованием процедур и функций		
	28 Использование процедур и функций при работе с массивами и строками.		
	29 Работа со встроенными модулями при решении конкретной задачи		
	30 Решение прикладной задачи с использованием структурного программирования. (ОК 3.2.2)		
	Контрольная работа №2 «Основные алгоритмические конструкции» Контрольная работа №3 «Массивы» Контрольная работа №4 «Подпрограммы»	*6*	
	Самостоятельная работа обучающихся	*49*	
	Выполнение упражнений на анализ структуры готовой программы. Написание программы по выданной задаче с четким выделением структурных элементов. Написание процедур и функций пользователя по индивидуальному заданию. Проведение анализа учебной литературы и информации сети Интернет и составление опорного конспекта «Встроенные модули» Создание презентации по теме «Встроенные модули среды программирования», «Особенности работы со встроенными модулями среды программирования» Составление опорного конспекта по теме: «Отладки и тестирование». (ОК 4.2.2) Выполнение работы с конспектом лекций: ответы на контрольные вопросы Подготовка к устным ответам по контрольным вопросам и анализу решенных задач. Составление сравнительной таблицы «Модульное или структурное программирование»		
Раздел 2. Объектно-ориентированное программирование		***36***	
Тема 2.1. Реализация ООП	Содержание учебного материала	*16*	
	1. Понятие ООП Основные принципы ООП.		*1*
	2. Событийно-управляемая модель программирования.		
	3. Виды классов ООП. Методы и события классов ООП		*2*
	4. Перегрузка методов. Создание наследованного класса.		
	5. Структура программы на основе ООП		
	6. Интегрированная среда разработчика. Интерфейс среды.		*2*
	7. Этапы разработки приложения		*1*

Наименование разделов и тем	Содержание учебного материала, лабораторные и практические работы, самостоятельная работа обучающихся	Объем часов	Уровень освоения
	8. Написание программы, следуя принципам ООП (ОК 6.1.1)		
	Лабораторные занятия	*6*	
	1. Разработка программы конкретной задачи на основе ООП.		
	2. Описание объектов в программе. Описание и реализация методов объектов		
	3. Моделирование работы физического объекта		
	Контрольная работа №5 «Основные понятия ООП»	*2*	
	Самостоятельная работа обучающегося	*12*	
	Выполнение упражнений на программное описание объекта выданного на лабораторной работе. Работа над контрольными вопросам из задания лабораторной работы. Составление опорного конспекта по теме: «Область применения ООП» (ОК 5.) Разработка программного кода по заданию в группе по теме ООП. Ознакомление с нормативными документами по написанию программ и подготовка доклада. Разработка теста по принципам ООП. Подготовка к экзамену.		
	Всего:	***144***	

3. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ ДИСЦИПЛИНЫ

3.1. Требования к минимальному материально-техническому обеспечению

Реализация программы дисциплины предполагает наличие кабинета информатики и полигона вычислительной техники.

Оборудование учебного кабинета и рабочих мест кабинета:

- рабочие места по количеству обучающихся с установленным лицензионным программным обеспечением и выходом в глобальную сеть Internet;
- рабочее место преподавателя;
- комплект плакатов по дисциплине;
- комплект учебно-методической документации.

Технические средства обучения:

- точки электропитания;
- сетевое оборудование, обеспечивающее работу локальной сети;
- мультимедийное оборудование;
- источники бесперебойного питания;
- интерактивная доска;

3.2. Информационное обеспечение обучения

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

Основные источники:

1. Гост 19.701-90 (ИСО 5807-85) схемы алгоритмов, программ, данных и систем.
2. Канцедал С.А. Алгоритмизация и программирование: учебное пособие. – М.: ИД «Форум»: ИНФРА-М, 2008 – 352 с.
3. Программирование на языке Паскаль: задачник/ под редакцией Усковой О.Ф. – СПб.: Питер, 2005. – 336 с.
4. Установочный диск с развернутой системой помощи языка программирования Visual Basic, 2010
5. Культин, Н.Б. Turbo Pascal в задачах и примерах: учебное пособие. – БХВ., 2007 – 256 с.
6. Павловская, Т.А. Паскаль. Программирование на языке высокого уровня. - СПб: Питер, 2007 – 393 с.

Дополнительные источники:

1. В.В. Фаронов Турбо Паскаль. Начальный курс. Учебное пособие-М.: «Нолидж», 1996.-613 с.
2. С.А. Абрамов и др. Задачи по программированию. — М.: Наука, 1988. – 210 с.

Интернет-ресурсы

1. Методы программирования
   http://www.tstu.ru/education/elib/pdf/2006/kulakov.pdf
2. Структурное программирование
   http://digital.sibsutis.ru/Progr/StrProgr.htm
3. Модульное программирование
   http://digital.sibsutis.ru/Progr/ManyMod.htm

4. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Контроль и оценка **результатов освоения дисциплины осуществляется преподавателем в процессе проведения практических занятий и лабораторных работ, тестирования, а также выполнения обучающимися индивидуальных заданий, проектов, исследований.**

Результаты обучения (освоенные умения, усвоенные знания)	**Формы и методы контроля и оценки результатов обучения**
В результате освоения дисциплины обучающийся должен **уметь**:	
работать в среде программирования	Оценка продукта учебной деятельности (написанного программного кода в среде программирования, выполненных операций по сохранению и отладке программы) по критериям (использование соответствующих правил работы в указанной среде программирования) на лабораторном занятии
реализовывать простейшие алгоритмы в виде программ на конкретном языке программирования	Оценка продукта учебной деятельности (программный код) по критериям (использование соответствующих алгоритмических конструкций, отсутствие синтаксических ошибок, использование соответствующих процедур и функций, соответствие поставленной задаче) на экзамене (2 семестр)
В результате освоения дисциплины обучающийся должен **знать**:	
этапы решения задачи на компьютере	Оценка результатов стандартизованного тестирования сопоставлением с эталоном (ключом, модельным ответом) дифференцированном зачете (1 семестр) и экзамене (2 семестр)
типы данных	
базовые конструкции изучаемых языков программирования	
принципы структурного и модульного программирования	
Знать принципы объектно-ориентированного программирования	

Формы и методы контроля и оценки результатов обучения должны позволять проверять у обучающихся не только сформированность умений и знаний, но и развитие общих компетенций.

Результаты обучения (освоенные общие компетенции)		**Основные показатели оценки результата**	**Формы и методы контроля и оценки результатов обучения**
Код	**Формулировка**		
ОК 1	Понимать сущность и социальную значимость своей профессии, проявлять к ней устойчивый интерес	Приведены произвольные примеры социальной значимости своей профессии Дано объяснение сущности профессии	Оценка результатов стандартизованного тестирования на итоговом испытании
ОК 2	Организовывать собственную деятельность,	Поставленная цель разбита на задачи.	Оценка продукта деятельности обучающегося (ре-

	выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.	Набор ресурсов определен в соответствии с условиями предложенного задания Выбранный метод и способ решения профессиональной задачи соответствует типовому (известному) алгоритму решения Оценка эффективности и качества метода и способа решения задачи соответствует заданной методике оценивания	шение практико-ориентированного задания) по критериям на итоговом испытании
ОК 3	Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.	Анализ проблемы проведен в соответствии с условиями заданной стандартной / нестандартной ситуации Принятое решение соответствует результатам проведенного анализа проблемы и позволяет ее решить	Оценка результатов формализованного наблюдения за деятельностью обучающегося по критериям на учебной практике
ОК 4	Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.	Обращается к информационным системам, базам и банкам компьютерных данных по интересующему вопросу в ходе решения поставленной профессиональной задачи Общается со специалистами по интересующему вопросу	Оценка результатов формализованного наблюдения за учебной / профессиональной деятельностью обучающегося по критериям на учебной / профессиональной практике
ОК 5	Использовать информационно - коммуникационные технологии в профессиональной деятельности.	Использование информационно - коммуникационных технологий в профессиональной деятельности соответствует условиям поставленной профессиональной задачи и характеристикам имеющегося аппаратного обеспечения.	Оценка продукта учебной / профессиональной деятельности обучающегося по критериям на учебной/производственной практике
ОК 6	Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.	Предложена идея для достижения цели групповой работы Озвученная идея аргументирована/разъяснена всему коллективу	Оценка результатов формализованного наблюдения за деятельностью обучающегося по критериям во время выполнения группового проекта на практических занятиях
ОК 7	Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.	Общая цель задания четко сформулирована и разъяснена всем членам коллектива/команды	Экспертная оценка продукта учебной деятельности (коллективного проекта) по критериям на защите
ОК 8	Самостоятельно определять задачи профессионального и личностного	Задачи профессионального развития определены в соответствии с условиями прак-	Оценка результатов стандартизованного тестирования на итоговом испы-

	развития, заниматься самообразованием, осознанно планировать повышение квалификации.	тико-ориентированного задания Запланированное повышение квалификации соответствует запросам потенциальных работодателей, предъявляемым к специалисту «техник-программист»	тании
ОК 9	Ориентироваться в условиях частой смены технологий в профессиональной деятельности.	Перечисляет современные программные и аппаратные ресурсы, соответствующие условиям поставленной профессиональной задачи Формулирует основные характеристики имеющихся программных и аппаратных ресурсов Осваивает предложенную технологию по имеющейся сопроводительной документации	Оценка результатов стандартизованного тестирования на итоговом испытании
ОК 10	Исполнять воинскую обязанность, в том числе с применением полученных профессиональных знаний (для юношей).	Приводит произвольные примеры применения полученных знаний при исполнении воинской обязанности	Экспертная оценка результатов стандартизованного тестирования на итоговом испытании

УЧЕБНАЯ ДИСЦИПЛИНА

Основы программирования

(дисциплина изучается в течение двух семестров)

Результаты обучения (освоенные умения, усвоенные знания)	**Основные показатели результатов подготовки**	**Формы и методы контроля и оценки результатов обучения**
В результате освоения дисциплины обучающийся должен **уметь**:		
работать в среде программирования	Операции с текстом программы, настройки среды программирования, сохранение, загрузка, запуск на исполнение и отладку программы, выполнены в соответствии с правилами работы в указанной среде программирования.	Оценка продукта учебной деятельности (написанного программного кода в среде программирования, выполненных операций по сохранению и отладке программы) по критериям (использование соответствующих правил работы в указанной среде программирования) на лабораторном занятии
реализовывать простейшие алгоритмы в виде программ на конкретном языке программирования	Программа, построенная на основе простейших алгоритмов, написана в соответствии с правилами программирования и соответствует поставленной задаче	Оценка продукта учебной деятельности (программный код) по критериям (использование соответствующих алгоритмических конструкций, отсутствие синтаксических ошибок, использование соответствующих процедур и функций, соответствие поставленной задаче) на экзамене (2 семестр)
В результате освоения дисциплины обучающийся должен **знать**:		
этапы решения задачи на компьютере	Перечисляет этапы решения задачи на компьютере	Оценка результатов стандартизованного тестирования сопоставлением с эталоном (ключом, модельным ответом) экзамене (2 семестр)
типы данных	Формулирует типы данных	
базовые конструкции изучаемых языков программирования	Воспроизводит базовые конструкции изучаемых языков программирования	
принципы структурного и модульного программирования	Формулирует принципы структурного программирования	
	Формулирует принципы модульного программирования	
Знать принципы объектно-ориентированного программирования	Формулирует принципы объектно-ориентирован-ного программирования	

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
	Знакомство с видами контроля по учебной дисциплине**.**	2		1	Теор.	Доска, мел	Составление конспекта	изучение конспекта	
	Раздел 1. Изучение конкретного языка программирования								
1.1.	Методы программирования: структурное и модульное	2		2	Теор	Проектор, экран, презентация по теме	Составление конспекта Поиск доп информации по теме	[2], гл. 1, стр.13	
1.2.	Структура программы. Принципы разработки ПО.	2		3	Теор	Проектор, экран, презентация по теме	Составление конспекта	[2], гл. 1, стр. 20	
1.3.	Алгоритм: блок-схема алгоритма, анализ алгоритма.	2		4	Теор	Доска, мел	Выполнение задания	[1], Составление сравнительной таблицы «Модульное или структурное программирование»	
1.4.	Среда программирования. Основные приемы работы, функции, запуск и редактирование программы	2		5	Теор	Доска, мел, Проектор, экран, презентация по теме	Составление конспекта Работа с эл. учебником	[2], гл. 1, стр. 36 Подготовка к к.р	
1.5.	Контрольная работа № 1 «Основы работы в среде программирования»	2		6	Теор	Дидактические материалы, компьютер	Выполнение контрольных заданий		
1.6.	Алфавит языка программирования. Типы данных: целый.	2		7	Теор	Доска, мел	Составление конспекта	[6], гл. 1, стр. 8	
1.7.	Типы данных: порядковые, вещественные.	2		8	Теор	Доска, мел	Составление конспекта Выполнение задания	[6], гл. 1, стр. 11	
1.8.	Операторы ввода-вывода. Простейшие операторы.	2		9	Теор	Доска, мел, дид. материалы	Составление конспекта	[6], гл. 1, стр. 13	
1.9.	Строковый тип: операции и функции.	2		10	Теор	Доска, мел, дид. материалы	Составление конспекта	Подготовка ответов на контр.вопросы	
1.10.	Тип данных множество.	2		11	Теор	Доска, мел, дид. материалы	Составление конспекта	Работа со литературой	
1.11.	Реализация линейной и разветвляющейся алгоритмических конструкций. Оператор Case of.	2		12	Теор	Доска, мел	Составление конспекта Выполнение задания	[2], гл. 2, стр 28	
1.12.	Выполнение операций с кодом про-	2		1	Лаб	Доска, маркер, ди-	Выполнение задания	Изучение конспек-	

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
	граммы в среде программирования: открытие, отладка, сохранение.					дактические материалы, компьютер		та	
1.13.	Применение простейших операторов при решении задач линейной структуры.	2		2	Лаб	Доска, маркер, дидакт.материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.14.	Решение задач с использованием структурного программирования.	2		3	Лаб	Доска, маркер, дид. материалы, компьютер	Выполнение задания Анализ структуры программы	Выполнение задания в конспекте	
1.15.	Реализация построенного алгоритма в виде программы. Элементы отладки программного кода.	2		4	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Изучение конспекта	
1.16.	Реализация циклических алгоритмических конструкций	2		13	Теор	Доска, мел	Составление конспекта Выполнение задания	[2], гл. 4, стр 30	
1.17.	Реализация основных алгоритмических конструкций. Составление программного кода по блок-схеме.	2		14	Теор	Доска, мел	Выполнение задания	[2], гл. 4, стр 38	
1.18.	Редактирование программного кода в соответствии со стандартами кодирования	2		5	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.19.	Реализация программ разветвляющейся структуры	2		6	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.20.	Реализация программ циклической структуры	2		7	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.21.	Реализация цикла с параметром.	2		8	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Изучение конспекта	
1.22.	Составление алгоритма задачи на использование цикла с предусловием.	2		9	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Изучение конспекта	
1.23.	Составление алгоритма задачи на использование цикла с постусловием.	2		10	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Изучение конспекта	
1.24.	Составление и отладка программы	2		11	Лаб	Доска, маркер, ди-	Выполнение задания	Изучение кон-	

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
	по блок-схеме.					дактические материалы, компьютер	Анализ структуры программы	спекта	
1.25.	Реализация программ усложненной структуры.	2		12	Лаб	Доска, маркер, дидактические материалы, компьютер	Выполнение задания	Подготовка к к.р.	
1.26.	Контрольная работа №2 «Основные алгоритмические конструкции»	2		15	Теор	Дидактические материалы, компьютер	Выполнение контрольных задан.		
1.27.	Структурированные типы данных: массивы.	2		16	Теор	Доска, маркер, электрон лекция, компьютер	Составление конспекта	[2], гл. 4, стр 38	
1.28.	Методы сортировки массива.	2		17	Теор	Презентация по теме, компьютер	Составление конспекта Выполнение задания	[6], гл. 2, стр. 98	
1.29.	Обработка символьного массива	2		18	Теор	дидактические материалы, дидактические материалы, компьютер	Составление конспекта Выполнение задания	[6], гл. 2, стр. 8	
1.30.	Реализация способов обработки одномерных массивов	2		13	Лаб	Дидактические материалы, компьютер	Выполнение задания	Контрольные вопросы	
1.31.	Поиск минимального(максимального) элемента в массиве.	2		14	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.32.	Составление программы на использование строковых одномерных массивов	2		15	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.33.	Решение задач на обработку строк.	2		16	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.34.	Выполнение сортировки массивов	2		17	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.35.	Решение поставленной задачи с использованием сортировки массива.	2		18	Лаб	Дидактические материалы, компьютер	Выполнение задания Анализ структуры программы	Выполнение задания в конспекте	

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
1.36.	Обработка двумерного массива	2		19	Теор	Доска, мел, дид. материалы	Составление конспекта	[2], гл. 2, стр. 109	
1.37.	Решение задач с использованием двумерных массивов	2		19	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.38.	Решение задач на обработку двумерных массивов.	2		20	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.39.	Поиск максимального (минимального) элемента в двумерном массиве	2		21	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.40.	Использование стандартных функций для работы со строками и массивами	2		22	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.41.	Сортировка двумерного массива	2		23	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.42.	Контрольная работа №3 «Массивы»	2		20	Теор	Дидактические материалы, компьютер	Выполнение контрольных задан.		
1.43.	Тип данных записи.	2		21	Теор	Проектор, экран, презентация по теме	Составление конспекта Анализ учебной литературы по теме	[2], гл. 8, стр 69	
1.44.	Решение задач с использованием записей.	2		24	Лаб	Дидактические материалы, компьютер	Выполнение задания	Выполнение задания в конспекте	
1.45.	Массив записей.	2		25	Теор	Дидактические материалы, компьютер	Выполнение задания	Подготовка к проверочной работе	
1.46.	Решение задач с использованием записей	2		22	Лаб	Дидактические материалы, компьютер	Выполнение заданий		
1.47.	Понятие подпрограммы. Процедуры.	2		23	Теор	ПК, раздаточный материал	Составление конспекта	[2], гл. 8, стр 80	
1.48.	Подпрограммы: Функции.	2		24	Теор	ПК, электрон лек-	Составление конспекта	[2], гл. 8, стр 82	

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
						ция	выполнение заданий		
1.49.	Программирование подпрограмм	2		25	Теор	ПК, электрон лекция	Составление конспекта выполнение заданий	[2], гл. 8, стр 82	
1.50.	Организация процедур при решении задач	2		26	Лаб	Доска, мел, дид. материалы	выполнение заданий	[3], гл. 4, стр 98	
1.51.	Организация функций при решении задач	2		27	Лаб	ПК, раздаточный материал	выполнение заданий	[2], гл. 8, стр 82	
1.52.	Использование подпрограмм при решении задач	2		28	Лаб	ПК, раздаточный материал	выполнение заданий	[2], гл.11, стр 117	
1.53.	Составление программ для решения конкретной задачи с использованием процедур и функций	2		29	Лаб	ПК, раздаточный материал	выполнение заданий	[2], гл.11, стр 117	
1.54.	Использование процедур и функций при работе с массивами и строками.	2		30	Лаб	Проектор, экран, презентация по теме	Тестирование	Изучение конспекта	
1.55.	Стандартные встроенные модули.	2		26	Теор	ПК, электрон лекция	Выполнение заданий Создание презентации по теме «Встроенные модули среды программирования»	[2], гл.13, стр 142	
1.56.	Работа со встроенными модулями при решении задачи	2		31	Лаб	ПК, раздаточный материал	Выполнение заданий Создание презентации по теме «Особенности работы со встроенными модулями среды программирования»	Задание в конспекте	
1.57.	Работа со встроенными модулями системы программирования	2		32	Лаб	ПК, раздаточный материал	Выполнение заданий	Задание в конспекте	
1.58.	Работа с графическим модулем	2		33	Лаб	ПК, раздаточный материал	Выполнение заданий	Задание в конспекте	
1.59.	Работа с графическим модулем	2		34	Лаб	ПК, раздаточный материал	Выполнение заданий	Задание в конспекте	
1.60.	Решение прикладной задачи с использованием структурного программирования.	2		35	Лаб	Доска, мел, дид. Материалы		Подготовка к к.р.	
1.61.	Контрольная работа №4 «Подпро-	2		27	Теор	Дидактические	Выполнение контроль-		

№	Наименование разделов, тем	Кол час.	Дата провед	№ занятия	Вид занятия	Оборудование занятия	Самостоятельная работа студентов	Дом. задание	Прим
	граммы»					материалы, компьютер	ных заданий		
1.62.	**Раздел 2. Объектно-ориентированное программирование**								
	Понятие ООП Основные принципы ООП.	2		28	Теор	Доска, мел, дид. материалы	Составление конспекта	Изучение конспекта	
2.1.	Событийно-управляемая модель программирования.	2		29	Теор	ПК, электрон лекция	Составление конспекта	Эл учебник	
2.2.	Виды классов ООП. Методы и события классов ООП. Перегрузка методов.	2		30	Теор	ПК, электрон лекция	Составление конспекта Составление тестовых вопросов по ООП	Изучение конспекта	
2.3.	Структура программы на основе ООП	2		31	Теор	ПК, электрон лекция	Составление конспекта Доклад по нормативным документами по написанию программ	Эл учебник	
2.4.	Контрольная работа №5 «Основные понятия ООП»	2		32	Теор	Дидактические материалы, компьютер	Выполнение контрольных задан.	Подготовка к экзамену	
2.5.	**Итого:**	**134ч.**	**Из них: Теоретических 64 ч. Лабораторных70 ч.**						

РАЗДЕЛ 2 КОНСПЕКТЫ ТЕОРЕТИЧЕСКИХ ЗАНЯТИЙ

Методы программирования: структурное и модульное

Методы программирования.

В основе того или иного языка программирования лежит некоторая руководящая идея, оказывающая существенное влияние на стиль соответствующих программ.

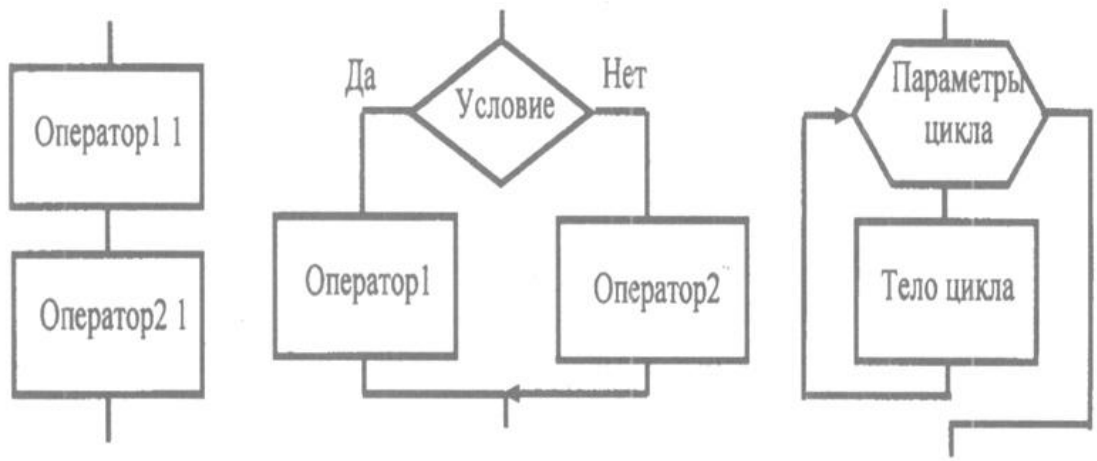
Структурное программирование

Структурное программирование - методология программирования, базирующаяся на системном подходе к анализу, проектированию и реализации программного обеспечения. Эта методология родилась в начале 70-х годов и оказалась настолько жизнеспособной, что и до сих пор является основной в большом количестве проектов.

Основу этой технологии составляют следующие положения:

- Сложная задача разбивается на более мелкие, функционально лучше управляемые задачи. Каждая задача имеет один вход и один выход. В этом случае управляющий поток программы состоит из совокупности элементарных подзадач с ясным функциональным назначением.
- Простота управляющих структур, используемых в задаче. Это положение означает, что логически задача должна состоять из минимальной, функционально полной совокупности достаточно простых управляющих структур. В качестве примера такой системы можно привести алгебру логики, в которой каждая функция может быть выражена через функционально полную систему: дизъюнкцию, конъюнкцию и отрицание.

Разработка программы должна вестись поэтапно. На каждом этапе должно решаться ограниченное число четко поставленных задач с ясным пониманием их значения и роли в контексте всей задачи. Если такое понимание не достигается, это говорит о том, что данный этап слишком велик, и его нужно разделить на более элементарные шаги. Согласно требованиям структурного программирования, наиболее часто детально проработанные алгоритмы изображаются в виде блок-схемы. При их разработке используются условные обозначения согласно ГОСТу («следование», «ветвление», «цикл») (рис. 1).



Концепция модульного программирования

Так же как и для структурной технологии программирования, концепцию модульного программирования можно сформулировать в виде нескольких понятий и положений:

- **Функциональная декомпозиция задачи** - разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач - модулей. Модули связаны между собой только по входным и выходным данным.
- **Модуль** - основа концепции модульного программирования. Каждый модуль в функциональной декомпозиции представляет собой "черный ящик" с одним входом и одним выходом. Модульный подход позволяет безболезненно производить модернизацию программы в процессе ее эксплуатации и облегчает ее сопровождение. Дополнительно модульный подход позволяет разрабатывать части программ одного проекта на разных языках программирования, после чего с помощью компоновочных средств объединять их в единый загрузочный модуль.
- Реализуемые решения должны быть простыми и ясными. Если назначение модуля непонятно, то это говорит о том, что декомпозиция начальной или промежуточной задачи была проведена недостаточно качественно. В этом случае необходимо еще раз проанализировать задачу и, возможно, провести дополнительное разбиение на подзадачи. При наличии сложных мест в проекте их нужно подробнее документировать с помощью продуманной системы комментариев. Этот процесс нужно продолжать до тех пор, пока действительно не удастся добиться ясного понимания назначения всех модулей задачи и их оптимального сочетания.
- Назначение всех переменных модуля должно быть описано с помощью комментариев по мере их определения.

С момента своего возникновения программирование боролось за возможность решать все более сложные задачи, создавать все более сложные программные системы и делать это как можно быстрее. В процессе разработки программ модульное программирование явилось воплощением общих методов борьбы со сложностью, обеспечения независимости компонент системы и использования иерархических структур.

Метод разработки программ по частям называют **модульным программированием.**

Программный модуль - это любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в других описаниях процесса.

Критерии приемлемости выделенного модуля **(по Хольту):**

- хороший модуль снаружи проще, чем внутри;
- хороший модуль проще использовать, чем построить.

Критерии приемлемости выделенного модуля **(по Майерсу):**

- размер модуля,
- прочность модуля,
- сцепление с другими модулями,
- рутинность модуля (независимость от предыстории обращений к нему).

Размер модуля измеряется числом содержащихся в нем операторов или строк. Обычно рекомендуются программные модули размером от нескольких десятков до нескольких сотен операторов.

Прочность модуля - это мера его внутренних связей. Чем выше прочность модуля, тем больше связей он может спрятать от внешней по отношению к нему части программы и, следовательно, тем больший вклад в упрощение программы он может внести. Для оценки степени прочности модуля **Майерс** предлагает упорядоченный по степени прочности набор из семи классов модулей. Для использования рекомендуются только два высших по прочности класса модулей. Функционально прочный модуль - это модуль, реализующий одну какую-либо определенную функцию. При реализации этой функции такой модуль может использовать и другие модули. Информационно прочный модуль -

это модуль, выполняющий (реализующий) несколько операций (функций) над одной и той же структурой данных (информационным объектом), которая считается неизвестной вне этого модуля. Для каждой из этих операций в таком модуле имеется свой вход со своей формой обращения к нему.

Сцепление модуля - это мера его зависимости по данным от других модулей. Характеризуется способом передачи данных. Единственным видом сцепления модулей, который рекомендуется для использования современной технологией программирования, является параметрическое сцепление (сцепление по данным по **Майерсу**) - данные передаются модулю либо при обращении к нему как значения его параметров, либо как результат его обращения к другому модулю для вычисления некоторой функции. Такой вид сцепления модулей реализуется на языках программирования при использовании обращений к процедурам (функциям).

Рутинность модуля - это его независимость от предыстории обращений к нему. Модуль будем называть рутинным, если результат (эффект) обращения к нему зависит только от значений его параметров (и не зависит от предыстории обращений к нему).

Модуль будем называть зависящим от предыстории, если результат (эффект) обращения к нему зависит от внутреннего состояния этого модуля, изменяемого в результате предыдущих обращений к нему.

Приемлема следующая рекомендация:

- всегда следует использовать рутинный модуль, если это не приводит к плохим (не рекомендуемым) сцеплениям модулей;
- зависящие от предыстории модули следует использовать только в случае, когда это необходимо для обеспечения параметрического сцепления;
- в спецификации зависящего от предыстории модуля должна быть четко сформулирована эта зависимость таким образом, чтобы было возможно прогнозировать поведение (эффект выполнения) данного модуля при разных последующих обращениях к нему.

Стандартные модули

Модули библиотек исполняющей системы программирования загружаются в память вместе с системой программирования; всегда их можно использовать.

Создание ваших собственных модулей

Если вы хотите написать модуль, содержащий некоторые полезные подпрограммы, и использовать эти подпрограммы в своих программах, напишите модули и сохраните его под именем, заданным в заголовке модуля.

Контрольные вопросы

1. Структурное программирование.
2. Запишите и расскажите как изображаются детально проработанные алгоритмы согласно ГОСТ.
3. Идея нисходящего структурного программирования «сверху-вниз».
4. Модульное программирование Требования к программным модулям
5. Модульное программирование Типы модулей:

Общие принципы разработки программного обеспечения

Программы различаются по назначению, выполняемым функциям, формам реализации. Однако можно полагать, что существуют некоторые общие принципы, которые следует использовать при разработке программ.

Частотный принцип

Принцип основан на выделении в алгоритмах и данных особых групп по частоте использования. Для действий, наиболее часто встречающихся при работе программ, создаются условия их быстрого выполнения. К часто используемым данным обеспечивается наиболее быстрый доступ. «Частые» операции стараются делать более короткими. Следует отметить, что лишь не более 5 % операторов программы оказывают ощутимое влияние на скорость выполнения программы. Этот факт позволяет значительную часть операторов программы кодировать без учета скорости вычислений, обращая основное внимание при этом на «красоту» и наглядность текстов.

Принцип модульности

Под модулем в данном контексте понимают функциональный элемент рассматриваемой системы, имеющий оформление, законченное и выполненное в пределах требований системы, и средства сопряжения с подобными элементами или элементами более высокого уровня данной или другой системы. Способы обособления составных частей программ в отдельные модули могут различаться существенно. В значительной степени разделение системы на модули определяется используемым методом проектирования программ.

Принцип функциональной избирательности

Этот принцип является логическим продолжением частотного и модульного принципов и используется при проектировании программ. В программах выделяется некоторая часть важных модулей, которые постоянно должны быть в состоянии готовности для эффективной организации вычислительного процесса. Эту часть в программах называют ядром или монитором. При формировании состава монитора требуется учесть два противоречивых требования. В состав монитора, помимо чисто управляющих модулей, должны войти наиболее часто используемые модули. Количество модулей должно быть таким, чтобы объем памяти, занимаемой монитором, был не слишком большим. Программы, входящие в состав монитора, постоянно хранятся в оперативной памяти. Остальные части программ постоянно хранятся во внешних запоминающих устройствах и загружаются в оперативную память только при необходимости, перекрывая друг друга также при необходимости.

Принцип генерируемости

Основное положение этого принципа определяет такой способ исходного представления программы, который бы позволял осуществлять настройку на конкретную конфигурацию технических средств, круг решаемых проблем, условия работы пользователя.

Принцип функциональной избыточности

Этот принцип учитывает возможность проведения одной и той же работы различными средствами. Особенно важен учет этого принципа при разработке пользовательского интерфейса для выдачи одних и тех же данных разными способами вызова из-за психологических различий в восприятии информации.

Принцип «по умолчанию»

Применяется для облегчения организации связей с системой как на стадии генерации, так и при работе с уже готовыми программами. Принцип основан на хранении в системе некоторых базовых описаний структур, модулей, конфигураций оборудования и данных, определяющих условия работы с программой. Эту информацию программа использует в качестве заданной по умолчанию, если пользователь забудет или сознательно не конкретизирует ее.

Качество ПО

Каждое ПО должно выполнять определенные функции, т.е. делать то, что задумано. Хорошее ПО должно обладать еще целым рядом свойств, позволяющим успешно его использовать в течении длительного периода, т.е. обладать определенным качеством. *Качество* ПО - это совокупность его черт и характеристик, которые влияют на его способность удовлетворять заданные потребности пользователей. Это не означает, что разные ПО должны обладать одной и той же совокупностью таких свойств в их высшей возможной степени. Этому препятствует тот факт, что повышение качества ПО по одному из таких свойств часто может быть достигнуто лишь ценой изменения стоимости, сроков завершения разработки и снижения качества этого ПО по другим его свойствам. Качество ПО является удовлетво-

рительным, когда оно обладает указанными свойствами в такой степени, чтобы гарантировать успешное его использование.

Совокупность свойств ПО, которая образует удовлетворительное для пользователя качество ПО, зависит от условий и характера эксплуатации этого ПО, т.е. от позиции, с которой должно рассматриваться качество этого ПО. Поэтому при описании качества ПО должны быть прежде всего фиксированы *критерии* отбора требуемых свойств ПО. В настоящее время критериями качества ПО принято считать:

- функциональность,
- надежность,
- легкость применения,
- эффективность,
- сопровождаемость,
- мобильность.

Функциональность - это способность ПО выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей. Набор указанных функций определяется во внешнем описании ПО.

Надежность - это его способность с достаточно большой вероятностью безотказно выполнять определенные функции при заданных условиях и в течение заданного периода времени.

Легкость применения - это характеристики ПО, которые позволяют минимизировать усилия пользователя по подготовке исходных данных, применению ПО и оценке полученных результатов, а также вызывать положительные эмоции определенного или подразумеваемого пользователя.

Эффективность - это отношение уровня услуг, предоставляемых ПО пользователю при заданных условиях, к объему используемых ресурсов.

Сопровождаемость - это характеристики ПО, которые позволяют минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации в соответствии с изменяющимися потребностями пользователей.

Мобильность - это способность ПО быть перенесенным из одной среды (окружения) в другую, в частности, с одной ЭВМ на другую.

Функциональность и надежность являются обязательными критериями качества ПО, причем обеспечение надежности будет красной нитью проходить по всем этапам и процессам разработки ПО. Остальные критерии используются в зависимости от потребностей пользователей в соответствии с требованиями к ПО.

Контрольные вопросы

1. Какими принципами следует руководствоваться при разработке ПО?
2. Раскрыть каждый принцип.
3. Что включает в себя понятие качество ПО?
4. Раскрыть каждый элемент, характеризующий качество ПО.
5. Приведите пример, демонстрирующий необязательный характер свойств мобильность, легкость применения.

Структура программы на языке Паскаль.

Язык программирования строиться на совокупности трех составляющих: алфавита, синтаксиса (жестких правил написания объектов языка) и семантики (правил их использования).

Pascal – это язык, который учит аккуратности и четкости (разделы программы нельзя менять местами, необходимо четко представлять работу программы и т.д.). Вот почему необходимо четко знать и понимать структуру программы на языке Pascal.

Общая структура программы на Паскаль.

PROGRAM имя программы;
(английскими буквами, одно слово)
USES подключаемые библиотеки (модули);
(дополнительные возможности, их можно подключать к программе в этой строке)
LABEL список меток;
Метки позволяют менять естественный ход выполнения программы. Ссылка на метку осуществляется оператором GOTO . Если в программе меток нет, то раздел LABEL отсутствует. В теле программы (в разделе операторов) метка ставится перед требуемым оператором и отделяется от него двоеточием.
CONST раздел описания констант;
(постоянные величины, их нельзя изменять)
CONST
<идентификатор 1=""> = <Значение 1="">;
<идентификатор 2=""> = <Значение 2="">;
<идентификатор n=""> = < Значение N>;

TYPE описание типов переменных; (тайп)
Стандартные типы данных (REAL, INTEGER, BOOLEAN, CHAR) не требуют описаний в этом разделе. Описания требуют только типы, образованные пользователем.
Концепция типов — одно из основных понятий в языке. С каждым данным связывается один и только один определенный тип.
Тип — это множество значений + множество операций, которые можно выполнять над этими значениями, то есть правила манипулирования данными. Использование типов позволяет выявлять многочисленные ошибки, связанные с некорректным использованием значений или операций еще на этапе трансляции без выполнения программ.
О Паскале говорят, что он строго типизирован, то есть программист должен описать все объекты, указывая их типы, и использовать в соответствии с объявленными типами. Программы становятся более надежными и качественными. При компиляции информация используется для уточнения вида операции. Так знаком + для данных типа REAL и INTEGER обозначается операция сложения, а для множеств (тип SET) — объединение. Структура раздела описания типов имеет вид:
TYPE
<имя 1=""> = <значение 1="">;
<имя 2=""> = <значение 2="">;
…
<имя l=""> = <значение l="">;
Имя типа представляет собой идентификатор, который может употребляться в других типах, описанных вслед за данным типом. Раздел TYPE не является обязательным, так как тип можно описать и в разделе переменных VAR. Примеры описания пользовательских типов:

VAR определение глобальных переменных;
(описание всех переменных величин, которые в программе могут изменяться)
VAR
<список 1="">:<тип 1="">;
<список 2="">:<тип 2="">;
…
<список n="">:<тип n="">;
ОПРЕДЕЛЕНИЕ ПРОЦЕДУР;
ОПРЕДЕЛЕНИЕ ФУНКЦИЙ;
BEGIN
основной блок программы
END.
Комментарий
Это пояснительный текст, который можно записать в любом месте программы, где разрешен пробел. Текст комментария ограничен: слева - '{', справа - '}', и может содержать любые символы. Комментарий игнорируется транслятором, и на программу влияния не оказывает.
Пример использования комментария:

PROGRAM PR;
<разделы>
BEGIN
<оператор 1="">
<оператор 2="">
{< Оператор 3; >
…
<оператор n="">}
END.
Средства комментария часто используются для отладки. Так в приведенном выше примере, операторы — 3,... N, заключенные в фигурные скобки, временно не выполняются.

Правила пунктуации.

Основным средством пунктуации является символ точка с запятой – ';'.

1. Точка с запятой не ставится после слов LABEL, TYPE, CONST, VAR, а ставится после каждого описания этих разделов.
2. Точка с запятой не ставится после BEGIN и перед END, так как эти слова – операторные скобки.
3. Точка с запятой разделяет операторы,
4. Точка с запятой не ставится после операторов WHILE, REPEAT, DO и перед UNTIL.
5. В условных операторах ';' не ставится после THEN и перед ELSE.

Разумеется, что не все приведенные выше блоки обязательны для использования. Приведу пример минимума, который необходимо использовать.

```
Program primer;
Begin
End.
```

Данная программа ни чего не делает, так как в ней нет ни одного оператора.

Пример программы

```
Program urok;
Begin
  writeln('Поздравляю! Ты написал первую программу!');
End.
```

Контрольные вопросы.

1. Что такое программа?
2. Раскройте основные составляющие языка программирования.
3. Перечислите блоки программы на языке Паскаль.
4. Расскажите какие из них обязательны, а какие могут быть опущены.
5. Какие есть правила пунктуации при написании операторов и строк программы в Паскаль?

АЛГОРИТМ: блок-схема алгоритма, анализ алгоритма.

АЛГОРИТМ – система правил, сформулированная на понятном исполнителю языке, которая определяет процесс перехода от допустимых исходных данных к некоторому результату и обладает свойствами массовости, конечности, определенности, детерминированности.

Алгоритм обладает следующими свойствами:

1. Дискретность. Это свойство состоит в том, что алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов. При этом для выполнения каждого шага алгоритма требуется конечный отрезок времени, т.е. преобразование исходных данных в результат осуществляется во времени дискретно.

2. Определенность. Каждое правило алгоритма должно быть четким, однозначным.

3. Результативность. Алгоритм должен приводить к решению за конечное число шагов.

4. Массовость. Алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными.

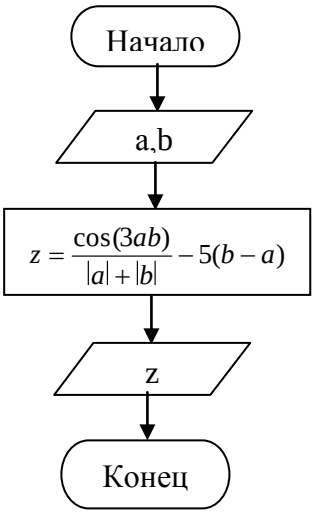
5. Правильность. Алгоритм правильный, если его выполнение дает правильные результаты решения поставленной задачи.

Написать алгоритм вычисления значения выражения $z = \dfrac{\cos(3ab)}{|a|+|b|} - 5(b-a)$

Решение:

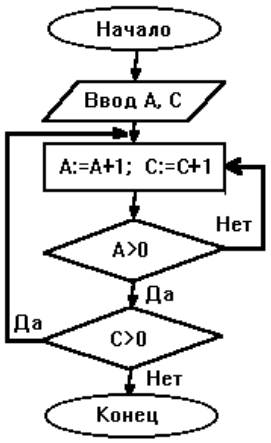
1. начало алгоритма
2. ввод исходных данных
3. вычисление $z = \dfrac{\cos(3ab)}{|a|+|b|} - 5(b-a)$
4. вывод результата на экран
5. конец алгоритма

Запишем приведенный алгоритм в виде блок-схемы

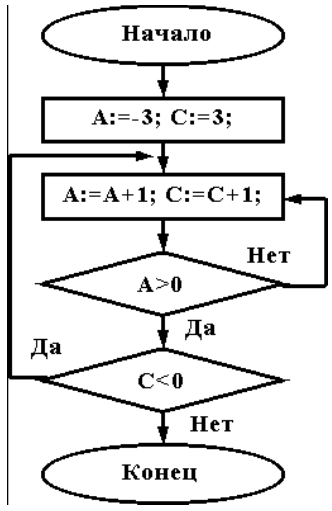


Анализ алгоритма

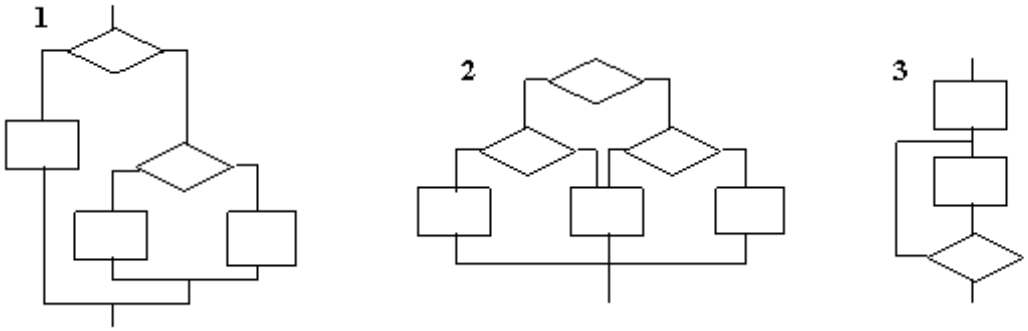
При каких начальных значениях переменных алгоритм закончит работу

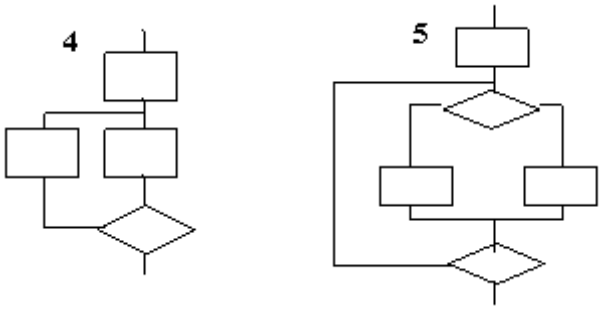
вариант ответа	схема алгоритма
1) A=-2; C=-3 2) A=-3; C=-2 3) A=-3, C=-3 4) A=-2; C=-1 5) A=-4, C=-3	

Определить выходные значения переменных A и C после выполнения алгоритма

Выходные значения A и C	Блок-схема
1) 1, 7 2) 0, -4 3) 1, 3 4) 0, -5 5) зацикливание	

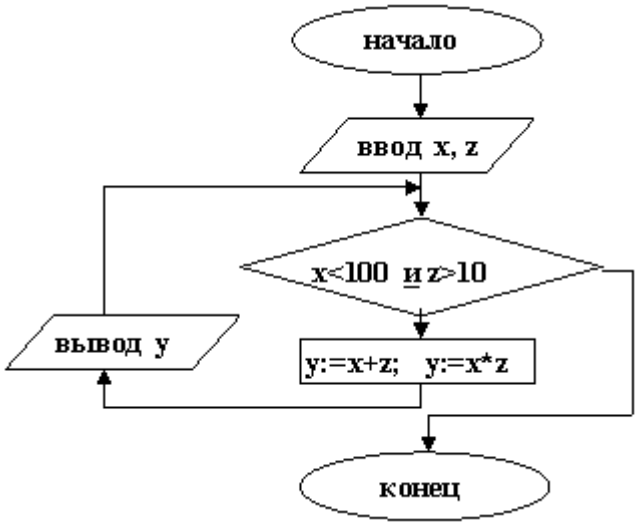
Укажите, какие из приведенных схем алгоритмов могут быть отнесены к основным (типовым) структурным схемам:





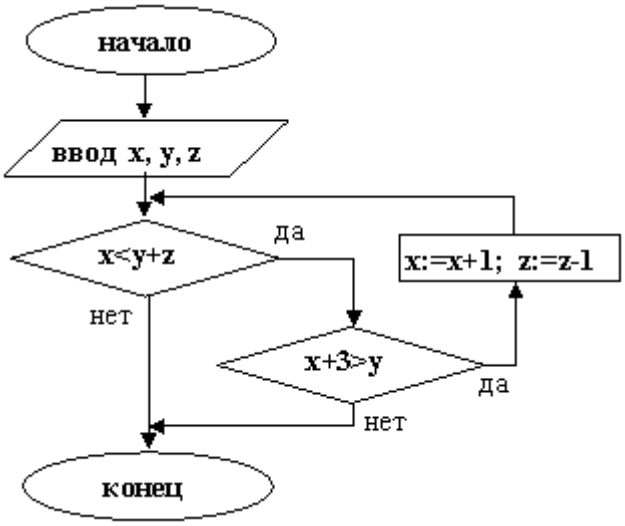
1) 1, 3, 4, 5; 2) 1, 3, 5; 3) 2, 3, 5; 4) 3, 4, 5; 5) 1, 2, 3.

Дана схема алгоритма. Укажите, какой из приведенных ниже программ она описывается.



1) начвещ x,y,z ввод x,z нц если (x<100) и (z>10) то y:=x*z иначе y:=x+z все вывод y кц кон	2) начвещ x,y,z ввод x,z нц пока (x<100) и (z>10) y:=x*z y:=x+z вывод y кц кон	3) начвещ x,y,z ввод x,z нц если (x<100) и (z>10) то y:=x+z иначе y:=x*z все вывод y кц кон
4) начвещ x,y,z ввод x,z выбор при x<100 y:=x*z при z>10 y:=x+z все вывод y кон	5) начвещ x,y,z ввод x,z выбор при x<100 y:=x+z при z>10 y:=x*z все вывод y кон	

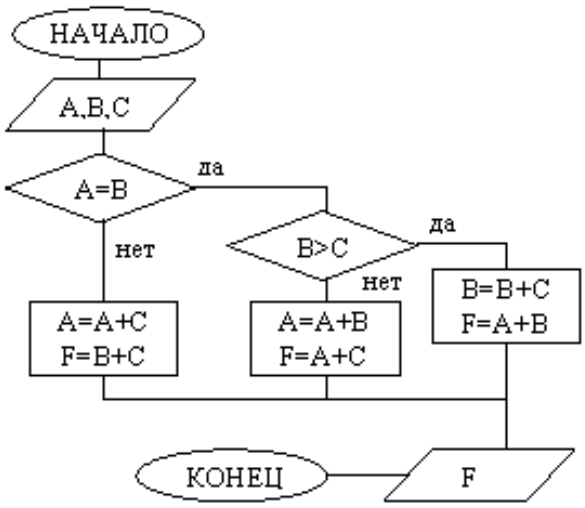
При каких значениях x, y, z выделенная команда выполнится 2 раза?



1) x=1 y=4 z=0 2) x=3 y=3 z=1 3) x=4 y=3 z=2 4) x=2 y=4 z=1 5) x=4 y=3 z=1

Вычисленное по блок-схеме значение переменной F для входных данных **1,1,3** равно

1) 7 2) 3 3) 4 4) 5 5) 6



Контрольные вопросы.

1. Дать понятие алгоритма и перечислить его свойства.
2. Раскрыть все свойства алгоритма.
3. Перечислите и запишите основные блоки блок-схемы , используемые для записи алгоритмов.
4. Нарисуйте блок-схемы основных алгоритмических конструкций.

Среда программирования. Основные приемы работы, функции, запуск и редактирование программы

Система программирования Турбо-Паскаль представляет собой не только сам язык, но и некоторую среду, с помощью которой создаются и компилируются (преобразуются в машинные коды) исходные тексты программ, а также запускаются на выполнение и отлаживаются готовые программы. Сама система (а это тоже программа) находится в файле turbo.exe в каталоге tp.
На скриншоте обозначены основные элементы интерфейса Pascal

Обратите внимание на верхнюю строку экрана, она представляет собой меню системы Турбо-Паскаль. С помощью меню Вы можете выполнять все действия, которые только возможны в данной системе. Нажмите F10 и выбирайте, что бы Вы хотели сейчас сделать. Например, Вы можете создать новый файл, содержащий текст программы на Паскале. Для этого нужно: выбрать пункт меню File (поместите на него курсор и нажмите Enter или просто щелкните мышью); в появившемся выпадающем меню выбрать пункт New.

В центре самой первой строки окна расположен его заголовок – имя файла, содержимое которого отображено в окне. В нашем случае файл называется NONAME00.PAS. Такое имя автоматически присваивается вновь создаваемому файлу.

На верхней строке слева расположена кнопка закрытия окна [¦]. Попробуйте щелкнуть на ней мышкой – окно исчезнет (не забудьте потом снова открыть его). Можно закрыть окно и другим способом – нажатием клавиши Alt+F3.

При работе с программой Турбо-Паскаль Вы будете использовать множество различных окон. И размеры окон могут быть самыми разными. Если в верхней строке окна справа имеется кнопка [^] или [¦], то, щелкнув мышкой на этой кнопке, можно увеличить окно до максимального размера или вернуть прежний размер, соответственно.

Обратите внимание на число рядом с кнопкой – это номер окна. Следует отметить, что для перехода из одного окна в другое нужно, нажать Alt+*номер окна*.

Теперь перейдем к самой нижней строке окна.

Слева отображаются координаты курсора – номер строки и столбца, в которых он находится. Для перехода на новую строку необходимо нажать Enter. Если Вы нажмете Enter, то сможете перемещаться уже по двум строкам и т.д.

Большую часть самой нижней строки окна самый правый столбец занимают ***полосы скроллинга***. Курсор на каждой из них показывает, текущее положение текста в окне относительно всего текста. В этом Вы убедитесь, когда напишите программу побольше – не помещающуюся на экране целиком.

Теперь уделим внимание самой нижней строке экрана. Эта строка постоянно напоминает Вам о том, как можно выполнить самые важные действия с помощью функциональных клавиш я для управления средой Турбо Паскаля. Они обозначаются F1, F2, ..., F12 и располагаются в самом верхнем ряду

клавиатуры. С каждой из этих клавиш связывается некоторая команда меню. Действие почти всех функциональных клавиш можно модифицировать тремя особыми клавишами: *Alt* (от *ALTernative - дополнительный*), *Ctrl* (*ConTRoL -управляющий*) и *Shift* (*SHIFT- сдвиговый*). Эти клавиши используются подобно клавиши временной смены регистра на пишущей машинке: нужно нажать на одну из них и затем, не отпуская ее, нажать функциональную клавишу. В дальнейшем такое совместное нажатие двух клавиш будем обозначать чертой. Например, *Alt-F3* означает, что вместе с клавишей *Alt* необходимо нажать клавишу *F3*, *Ctrl-F9* - вместе с *Ctrl* нажимается F9 и т.д.

Ниже приводятся команды, которые передаются среде Турбо Паскаля функциональными клавишами и некоторыми их комбинациями с клавишами *Ctrl* и *Alt*:

Fl - обратиться за справкой к встроенной справочной службе (*Help*-помощь);

F2 - записать редактируемый текст в дисковый файл;

F3 - прочитать текст из дискового файла в окно редактора;

F4 - используется в отладочном режиме: начать или продолжить исполнение программы и остановиться перед исполнением той ее строки, на которой стоит курсор;

F5 - распахнуть активное окно на весь экран;

F6 - сделать активным следующее окно;

F7 - используется в отладочном режиме: выполнить следующую строку программы; если в строке есть обращение к процедуре (функции), войти в эту процедуру и остановиться перед исполнением первого ее оператора;

F8- используется в отладочном режиме: выполнить следующую строку программы; если в строке есть обращение к процедуре (функции), исполнить ее и не прослеживать ее работу;

F9 - компилировать программу, но не выполнять ее;

F10 - перейти к диалоговому выбору режима работы с помощью главного меню;

Ctrl-F9 - выполнить прогон программы: компилировать программу, находящуюся в редакторе, загрузить ее в оперативную память и выполнить, после чего вернуться в среду Турбо Паскаля.

Alt-F5 - сменить окно редактора на окно вывода результатов работы (прогона) программы.

Команда Ctrl-F9 для проверки работы программы и *Alt-X* - для выхода из Турбо Паскаля. Командой *Alt-F5* в любой момент можно просмотреть данные, выданные на экран в результате прогона программы.

ТЕКСТОВЫЙ РЕДАКТОР

Текстовый редактор среды Турбо Паскаля предоставляет пользователю удобные средства создания и редактирования текстов программ. Рассмотрим основные приемы работы с текстовым редактором.

Для создания текста программы нужно ввести этот текст с помощью клавиатуры ПК подобно тому, как это делается при печатании текста на пишущей машинке. После заполнения очередной строки следует нажать на клавишу *Enter*, чтобы перевести курсор на следующую строку (курсор всегда показывает то место на экране, куда будет помещен очередной вводимый символ программы).

Окно редактора имитирует длинный и достаточно широкий лист бумаги, фрагмент которого виден в окне. Размеры листа по горизонтали и вертикали ограничиваются только общим числом символов в файле, которых не должно быть больше 64535, однако компилятор Турбо Паскаля воспринимает строки программы длиной не более 126 символов.

Окно можно смещать относительно листа с помощью следующих клавиш:

Page Up -на страницу вверх;

Page Down - на страницу вниз;

Home - в начало текущей строки;

End - в конец текущей строки;

Ctrl-Page Up -в начало текста;

Ctrl-Page Down - в конец текста.

Нормальный режим работы редактора - режим вставки, в котором каждый вновь вводимый символ как бы «раздвигает» текст на экране, смещая вправо остаток строки. Следует учитывать, что разрезание текста и последующая вставка пропущенных строк возможны только в этом режиме. Редактор может также работать в режиме наложения новых символов на существующий старый текст: в этом режиме новый символ заменяет собой тот символ, на который указывает курсор, а остаток строки не смещается вправо. Для перехода к режиму наложения нужно нажать клавишу *Insert*, а если нажать эту клавишу еще раз, вновь устанавливается режим вставки.

И еще об одной возможности редактора. Обычно редактор работает в режиме автоотступа. В этом режиме каждая новая строка начинается в той же позиции на экране, что и предыдущая. Режим автоотступа поддерживает хороший стиль оформления текста программы: отступы от левого края выделяют тело условного или составного оператора и делают программу более наглядной. Отказаться от автоотступа можно командой *Ctrl-O I* (при нажатой *Ctrl* нажимается сначала клавиша с латинской буквой *O*, а затем *O* отпускается и нажимается *I*), повторная команда *Ctrl-O I* восстановит режим автоотступа.

Команды редактирования

Backspace - стирает символ слева от курсора;

Delete - стирает символ, на который показывает курсор;

Enter - вставляет новую строку, разрезает старую;

Работа с блоком программы

Ctrl - Y - уничтожает выделенный блок;

Ctrl- C - копирует блок (*Ctrl - Insert*);

Ctrl - V - перемещает блок на новое место(*Shift - Insert*);

Ctrl- W – вставляет блок в файл;

ОСНОВНЫЕ ПРИЕМЫ РАБОТЫ В СРЕДЕ ТУРБО ПАСКАЛЯ

Работа с файлами

Основной формой хранения текстов программ вне среды являются файлы. После завершения работы с Турбо Паскалем можно сохранить текст новой программы в дисковом файле с тем, чтобы использовать его в следующий раз.Для сохранения текста программы в файле нужно нажать F2. В этот момент среда проверит имя программы и, если это стандартное имя *NONAME,* спросит, нужно ли его изменять: на экране появится небольшое окно запроса с надписью

```
Save File as
(Сохранить в файле с именем...)
```

Если в имени опущено расширение, среда присвоит файлу стандартное расширение *PAS*. Если Вы захотите завершить работу с Турбо Паскалем, а в редакторе остался не сохраненный в файле текст, на экране появится окно с запросом

```
NONAMEOO.PAS has been modified.Save?
(Файл NONAME00.PAS был изменен. Сохранить?)
```

В ответ следует нажать Y(*Yes* - да), если необходимо сохранить текст в файле, или N(*No* - нет), если сохранять текст не нужно.

Прогон и отладка программы

После подготовки текста программы можно попытаться исполнить ее, т.е. откомпилировать программу, связать ее (если это необходимо) с библиотекой стандартных процедур и функций, загру-

зить в оперативную память и передать ей управление. Вся эта последовательность действий называется прогоном программы и реализуется командой *Ctrl-F9.*

Если на каком-либо этапе среда обнаружит ошибку, она прекращает дальнейшие действия, восстанавливает окно редактора и помещает курсор на ту строку программы, при компиляции или исполнении которой обнаружена ошибка. При этом в верхней строке редактора появляется диагностическое сообщение о причине ошибки. Все это позволяет очень быстро отладить программу, т.е. устранить в ней синтаксические ошибки и убедиться в правильности ее работы. Иногда прибегают к пошаговому исполнению программы с помощью команд, связанных с клавишами F4, F7 и F8. Пока еще не накоплен достаточный опыт отладки, можно воспользоваться одной клавишей F7, после нажатия на которую среда осуществит компиляцию, компоновку (связь с библиотекой стандартных процедур и функций) и загрузку программы, а затем остановит прогон перед исполнением первого оператора. Строка программы, содержащая этот оператор, будет выделена на экране указателем (цветом). Теперь каждое новое нажатие F7 будет вызывать исполнение всех операций, запрограммированных в текущей строке, и смещение указателя к следующей строке программы. В подозрительном месте программы можно просмотреть текущее значение переменной или выражения. Для этого нужно установить курсор в то место строки, где находится имя интересующей Вас переменной, и нажать *Ctrl-F4*. На экране появится диалоговое окно, состоящее из трех полей (в верхнем поле будет стоять имя переменной, два других поля будут пустыми). Нажмите *Enter*, чтобы в среднем поле получить текущее значение переменной. Если перед нажатием *Ctrl-F4* курсор стоял на пустом участке строки или указывал на имя другой переменной, верхнее поле диалогового окна также будет пустым или содержать имя этой другой переменной. В этом случае следует ввести с помощью клавиатуры имя нужной переменной и нажать *Enter*.

Справочная служба Турбо Паскаля

Неотъемлемой составной частью среды Турбо Паскаля является встроенная справочная служба. Эта справка зависит от текущего состояния среды, поэтому справочную службу называют контекстно-чувствительной. Например, если нажать F1 в момент, когда среда обнаружила ошибку в программе, в справке будут сообщены дополнительные сведения о причинах ошибки и даны рекомендации по ее устранению.

Существуют четыре способа обращения к. справочной службе непосредственно из окна редактора:

- *F1* - получение контекстно-зависимой справки;
- *Shift-F1* - выбор справки из списка доступных справочных сообщений;
- *Ctrl-F1* - получение справки о нужной стандартной процедуре, функции, о стандартной константе или переменной;
- *Alt-F1* - получение предыдущей справку.

По команде *Shift-F1* на экране появится окно, содержащее упорядоченный по алфавиту список стандартных процедур, функций, типов, констант и переменных, для которых можно получить справочную информацию.

Или напечатайте на экране имя процедуры (функции, типа и т.д.) или подведите курсор к имеющемуся в тексте стандартному имени и нажмите *Ctrl-F1*. Среда проанализирует ближайшее окружение курсора, выделит имя и даст нужную справку.

Основные команды и горячие клавиши

Ниже приведены основные команды среды разработчика Турбо Паскаль и соответствующие им горячие клавиши.

- **Ctrl+F9** - запуск программы
- **Alt+F5** - просмотр пользовательского экрана
- **F2** - сохранение программы
- **F3** - открытие сохраненной программы
- **Alt+F3** - закрытие активного окна
- **Alt+X** - выход из Турбо Паскаль
- **F1** - контекстная помощь
- **Ctrl+F1** - справка об операторе, на котором установлен курсор
- **Alt+Backspace** - отмена последнео изменения

- **Ctrl+Y** - удаление строки
- **Shift+стрелки** - выделение блока текста
- **Ctrl+Insert** - копирование выделенного блока в буфер
- **Shift+Insert** - вставка из буфера

Контрольные вопросы.

1. Раскройте назначение всех элементов окна среды программирования Паскаль.
2. Как можно запустить программу на исполнение? (Ctrl+F9, F8, F7)
3. Максимальная длина строки программы? (126 символов)
4. Как включить/выключить режим замены символов при наборе и редактировании программного кода? (Insert)
5. Опишите, как выделить, скопировать и вставить блок программы в заданное место программного кода. (Shift – клавиши управления курсором ,Ctrl - Insert, Shift - Insert)
6. Как осуществить просмотр результатов работы программы на экране? (Alt+F5)

Алфавит языка программирования. Типы данных.

Алфавит языка

Текст Pascal-программы представляет собой последовательность ***строк*** состоящих из символов, образующих ***алфавит*** языка. Строки программы завершаются специальными управляющими символами, не входящими в алфавит. Максимальная длина строки составляет 126 символов.

Алфавит языка состоит из следующих символов:

- Заглавные и строчные латинские буквы и символ подчеркивания:

  `A,B,C.. .,X,Y,Z,a,b,c, .. .,x,y,z.`

  Обратите внимание, что в языке Turbo Pascal символ подчеркивания считается буквой.Буквы используются для формирования идентификаторов и служебных слов.

- Десять арабских цифр от 0 до 9:

  `0,1,2,3,4,5,6,7,8,9`

  Цифры используются для записи чисел и идентификаторов.

- Двадцать два специальных символа:

  `+ -*/-><.  ,  ;  : ( )[ ]{ }#$`

  Специальные символы используются для конструирования знаков операций, выражений, комментариев, а также как синтаксические разделители.

Лексическая структура языка.

Символы из алфавита языка используются для построения базовых элементов Pascal-программ - лексем.

Лексема - минимальная единица языка, имеющая самостоятельный смысл.

В Turbo Pascal имеются следующие классы лексем:

1. ***Служебные (зарезервированные) слова.***

Это ограниченная группа слов, построенных из букв. Каждое служебное слово представляет собой неделимое образование, смысл которого фиксирован в языке. Служебные слова НЕЛЬЗЯ ис-

пользовать в качестве имен, вводимых программистом (т.е. в качестве идентификаторов переменных, констант и т.д.).

Все 55 служебных слов языка представлены ниже.

absolute	array	and	asm	assembler
begin	case	const	constructor	destructor
div	downto	else	end	external
file	for	forward	function	goto
if	implementation	in	inline	interface
interrupt	label	mod	nil	not
object	of	or	packed	private
procedure	program	record	repeat	set
shl	shr	string	then	to
type	unit	until	uses	var
virtual	while	with	xor	

Заметим, что синтаксис языка Turbo Pascal на самом деле допускает использование некоторых служебных слов в качестве идентификаторов (к числу таких слов относятся assembler, external, forward, interrupt, private, virtual). Строго говоря, эти слова называются в языке *директивами*. Однако в целях большей ясности программ использование директив в качестве идентификаторов не рекомендуется.

2. ***Идентификаторы (имена)***. Идентификаторы вводятся для обозначения в программе переменных, констант, типов, меток, процедур и функций и формируются из букв и цифр, но может начинаться только с буквы.

Длина идентификатора может быть произвольной, однако компилятор воспринимает только ПЕРВЫЕ 63 его символа.

Важно помнить, что в языке Turbo Pascal соответствующие заглавные и строчные буквы в идентификаторах и служебных словах НЕ РАЗЛИЧАЮТСЯ. Таким образом, следующие три идентификатора обозначают одну и ту же переменную:

index
INDEX
Index

3. ***Изображения***. Эта группа лексем обозначает числа, символьные строки и некоторые другие значения.

4. ***Знаки операций*** формируются из одного или нескольких специальных символов и предназначены для задания действий по преобразованию данных и вычислению значений.

5. ***Разделители*** также формируются из специальных символов и в основном используются для повышения наглядности текстов программ. Примерами разделителей могут служить следующие конструкции:

; : = (.

Необходимо выделить еще одну группу лексических конструкций языка, обычно называемых *директивами компилятора* или прагматами. Эти конструкции так же, как комментарии, заключаются в фигурные скобки, но они не носят характер пояснений к программе, а являются инструкциями Pascal-компилятору, предписывающими тот или иной режим обработки Pascal-программы.

Директивы компилятора должны содержать сразу же после открывающей фигурной скобки (без пробелов и символов табуляции) символ '$' (доллар), а за ним - одиночную букву, определяющую конкретный режим компиляции. После буквы может присутствовать знак '+' (плюс) или '-' (минус), задающий, соответственно, установку или отмену заданного режима, например:

{$N-}, {SN+}

Важную роль в тексте Pascal-программы играет символ, не имеющий специального графического изображения, - *пробел*. Символы пробелов используются для отделения лексем друг от друга в тех случаях, когда слитное написание двух или более лексем может исказить смысл программы. В частности, если идентификаторы и служебные слова записываются друг за другом, то они обязательно должны быть отделены одним или несколькими пробелами (или расположены в различных строках), так как их слитное написание будет воспринято как один идентификатор.

В остальных случаях использование пробелов необязательно и служит целям наглядности, способствуя лучшему внешнему виду программ. Можно сказать, что пробелы, так же, как и комментарии, могут (в любом количестве) находиться между любыми двумя лексемами программы.

Контрольные вопросы.

1. Назовите алфавит языка прогаммирования.
2. Раскройте лексическую структуру языка программирования.

Данные. Понятие типа данных.

Данные – это любая информация, представленная в формализованном виде и пригодная для обработки алгоритмом.

Данные, известные перед выполнением алгоритма называются исходными данными.

Результат решения задачи – выходные данные.

Обрабатываемые в программе данные подразделяются на переменные, константы и литералы:

Константы представляют собой данные, значения которых установлены в разделе объявления констант и не изменяются в процессе выполнения программы.

Переменные объявляются в разделе объявления переменных, но в отличие от констант получают свои значения уже в процессе выполнения программы, причем допускается изменение этих значений. К константам и переменным можно обращаться по именам.

Каждая переменная и константа должна иметь свое уникальное имя. Имена переменных и констант задаются идентификаторами. Идентификатор представляет собой последовательность букв и цифр, начинающаяся с буквы.

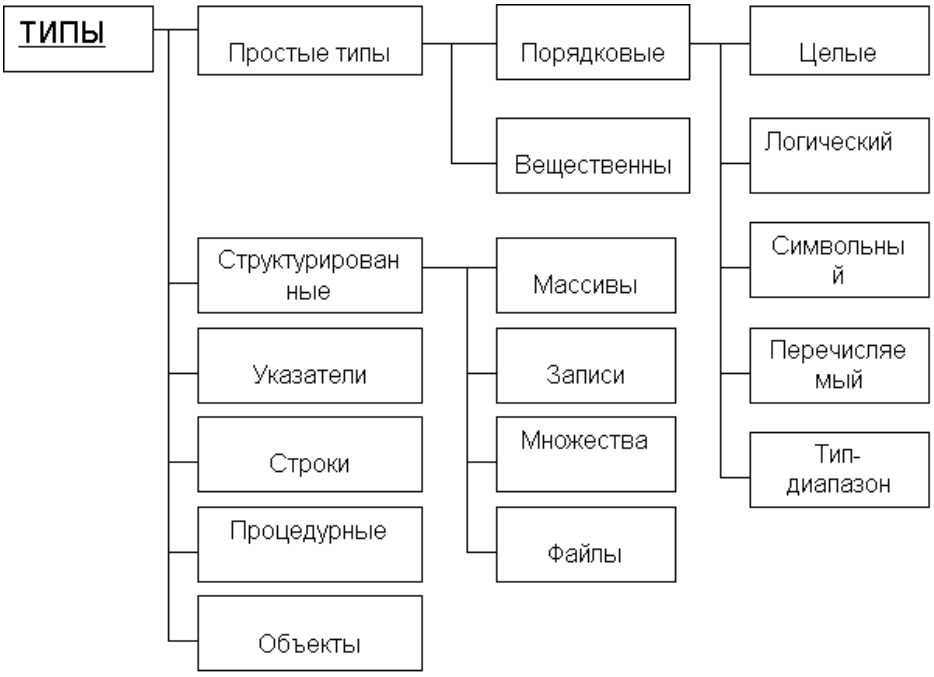
Литерал не имеет идентификатора и представляется в тексте программы непосредственно значением.

Любые данные характеризуются своими типами. Тип определяет множество допустимых значений, которые может иметь тот или иной объект, а также множество допустимых операций, которые применимы к нему. Кроме того, тип определяет также и формат внутреннего представления данных в памяти ПК.

Вообще язык Pascal характеризуется разветвленной структурой типов данных. В языке предусмотрен механизм создания новых типов, благодаря чему общее количество используемых в программе типов может быть сколь угодно большим.

Тип определяет множество значений, которые могут принимать элементы данных, и совокупность допустимых над ними операций.

Структура типов данных в языке Pascal.



ПРОСТЫЕ ТИПЫ

К простым типам относятся порядковые и вещественные типы.

Порядковые типы отличаются тем, что каждый из них имеет конечное число возможных значений. Эти значения можно определённым образом упорядочить (отсюда – название типов) и, следовательно, с каждым из них можно сопоставить некоторое целое число – порядковый номер значения.

Вещественные типы тоже имеют конечное число значений, которое определяется форматом внутреннего представления вещественного числа. Однако количество возможных значений вещественных типов настолько велико, что сопоставить с каждым из них целое число (его номер) не представляется возможным.

ПОРЯДКОВЫЕ ТИПЫ

К порядковым типам относятся целые, логический, символьный, перечисляемый и тип-диапазон.

Целые типы. Диапазон возможных значений целых типов зависит от их внутреннего представления, которое может занимать один, два или четыре байта

Таблица 1 Целые типы

Название	**Длина, байт**	**Диапазон значений**
Byte	1	0…255
ShortInt	1	-128…+127
Word	2	0…65535
Integer	2	-32768…+32767
LongInt	4	-2 147 483 648…+2 147 483 647

При использовании процедур и функций с целочисленными параметрами следует руководствоваться «вложенностью» типов, т.е. везде, где может использоваться Word, допускается использование Byte (но не наоборот), в LongInt “входит” Integer, который, в свою очередь, включает в себя ShortInt.

При действии с целыми числами тип результата будет соответствовать типу операндов, если операнды относятся к различным целым типам,- типу того операнда, который имеет максимальную мощность (максимальный диапазон значений). Возможное переполнение никак не контролируется, что может привести к недоразумениям.

Над целыми операндами можно выполнять следующие арифметические операции: сложение, вычитание, умножение, деление, получение остатка от деления. Знаки этих операций:

+ - * div mod

Результат арифметической операции над целыми операндами есть величина целого типа. Результат выполнения операции деления целых величин есть целая часть частного. Результат выполнения операции получения остатка от деления - остаток от деления целых. Например:

17 div 2 = 8, 3 div 5 = 0.

17 mod 2 = 1, 3 mod 5 = 3.

Операции отношения, примененные к целым операндам, дают результат логического типа TRUE или FALSE (истина или ложь).

В языке ПАСКАЛЬ имеются следующие операции отношения: равенство =, неравенство <>, больше или равно >=, меньше или равно <=,больше >, меньше < .

К аргументам целого типа применимы следующие стандартные (встроенные) функции, результат выполнения которых имеет целый тип:

Abs(X), Sqr(X), Succ(X), Pred(X),

и которые определяют соответственно абсолютное значение X, X в квадрате, X+1, X-1.

Следующая группа стандартных функций для аргумента целого типа дает действительный результат:

Sin(X), Cos(X), ArcTan(X), Ln(X), Exp(X), Sqrt(X).

Эти функции вычисляют синус, косинус и арктангенс угла, заданного в радианах, логарифм натуральный, экспоненту и корень квадратный соответственно.

Результат выполнения функции проверки целой величины на нечетность **Odd(X)** имеет значение истина, если аргумент нечетный, и значение ложь, если аргумент четный:

X=5 Odd(X)=TRUE , X=4 Odd(X)=FALSE.

Для быстрой работы с целыми числами определены процедуры:

Inc(X) X:=X+1
Inc(X,N) X:=X+N
Dec(X) X:=X-1
Dec(X,N) X:=X-N

Логический тип

Таблица 2 Логические типы

Название	**Длина, Байт**	**OS**	**Значение**
BOOLEAN	1	Linux, DOS	False, True
BYTEBOOL	1	Совместим с C	False, True
WORDBOOL	2	Win	False, True
LONGBOOL	4	Win	False, True

Значениями логического типа может быть одна из предварительно объявленных констант FALSE (ложь) или TRUE (истина).

Логические операции

Логические операции применяются к величинам логического типа, результат операции — тоже логического типа. Имеется одна унарная логическая операция not (отрицание) и три бинарные операции and (и), or (или), xor (исключающее или).

Логическая операция not

Ставиться перед логической переменной (выражением). Инвертирует (меняет на противоположное) значение логической переменной или логического выражения.

Логическая операция and

Объединяет две логические переменные (логические выражения). Результат получившегося выражения будет истинным, если истинны обе переменные (оба выражения) составляющие данное выражение. В противном случае выражение ложно.

Логическая операция or

Объединяет две логические переменные (логические выражения). Результат получившегося выражения будет истинным, если истинной является хотя бы одна из переменных (выражений) составляющие данное выражение. В противном случае выражение ложно.

Логическая операция xor

Объединяет две логические переменные (логические выражения). Результат получившегося выражения будет истинным, если значения этих переменных (выражений) различны. В противном случае выражение ложно.

Для описания всех логических операций используют так называемые таблицы истинности. В этих таблицах X и Y — это логические переменные (выражения), составляющие результирующее выражение.

Таблица истинности not

X	not X
False	True
True	False

Таблица истинности операций and, or ,xor

X	Y	X and Y	X or Y	X xor Y
False	False	False	False	False
False	True	False	True	True
True	False	False	True	True
True	True	True	True	False

Примеры использования логических операций

a:=False; d34:=True;

Done := not a; {Done = True}

Done := a and d34; {Done = False}

Done := a or d34; {Done = True}

Done := a xor d34; {Done = True}

Существует ряд операций и функций, результатом которых являются величины логического типа.

Операции отношения

Операции отношения предназначены для сравнения двух величин. Результат сравнения имеет логический (Boolean) тип.

= равно;

<> не равно;

< меньше;

> больше;

<= меньше или равно;

>= больше или равно;

Примеры использования операций отношения

Done:=(5<0); {Done = False}

Done:=(c<=2); {Done = True при c<=2}

Done:=(c<=10) and (c>=0); {Done = True при 0>=c>=10}

Логическая функция ODD (x)

Определяет четность целого числа x. Возвращает значение True, если x нечетно и False в противоположном случае.

Упорядоченность логических переменных

К логическим переменным могут применяться операции отношения. Упорядочены логические (булевские) переменные следующим образом:

False < True

При составлении сложных арифметических или логических выражений необходимо помнить о порядке (очередности выполнения) операций того или иного типа. Порядок выполнения операций (вычисления выражений) часто называют приоритетом действий.

Порядок вычисления выражений — приоритеты действий:

1. Вычисления в круглых скобках

2. Вычисление значений функций {sin(x), cos(x), sqrt(x) и т. д.}

3. Унарные операции { not }

4. Мультипликативные операции { *, /, div, mod, and }

5. Аддитивные операции { +, -, or, xor }

6. Операции отношения { =, <>, <, >, <=, >= }

Символьный тип

CHAR – занимает 1 байт. Значением символьного типа является множество всех символов ПК. Каждому символу присваивается целое число в диапазоне 0…255. Это число служит кодом внутреннего представления символа.

Для кодировки испоьзуется код ASCII (American Standart Code for Information Interchange – американский стандартный код для обмена информацией). Это 7-битный код, т.е. с его помощью можн озакодировать лишь 128 символов в диапозоне от 0 до 127. В то же время в 8-битном байте, отведенном для хранения символа в Паскаль, можно закодировать в два раза больше символов в дапозоне от 0 до 255. Первая половина символов ПК с кодами 0…127 соответствует стандарту ASCII. Вторая половина символов с кодами 128...255 не ограничена жёсткими рамками стандарта и может менятся на ПК разных типов.

Символы с кодами 0…31 относятся к слжебным кодам. Если эти коды использовать в символьном тексте программы, они считаются пробелами.

Функция *Ord(x),* преобразовывает букву в ее числовой код.

Функция *Chr(x)* противоположна функции Ord(x). Эта функция будет преобразовывать числовой код символа в сам символ.

Перечисляемый тип

Перечисляемый тип представляет собой ограниченную упорядоченную последовательность скалярных констант, составляющих данный тип. Каждое значение именуется некоторым идентификатором и рапологается в списке, обрамлённом круглыми скобками, например:

Type

Colors = (red, white, blue);

Применение перечисляемых типов делает программы нагляднее.

Соответствие между значениями перечисляемого типа и порядковыми номерами этих значений устанавливается порядком перечисления: первое значение списке получает порядковый номер 0, второе – 1 и т.д. максимальная мощность перечисляемого типа составляет 65536 значений, поэтому фак-

тически перечисляемый тип задаёт некоторое подмножество целого типа WORD и может рассматриваться как компактное объявление сразу группы целочисленных констант со значениями 0,1 и т.д.

Переменные перечисляемого типа могут быть описаны в разделе описания переменных, например:

```
var Section: (RED, YELLOW, GREEN);
```

К переменным перечисляемого типа может быть применим оператор ***присваивания***:

```
Section:= YELLOW;
```

К перечисляемым переменным и константам могут быть применены операции отношения и стандартные функции ***Pred, Succ, Ord.***

Использование перечисляемых типов повышает надёжность программы, благодаря возможности контроля тех значений, которые получают соответствующие переменные.

Интервальный тип данных (Тип-диапазон)

Тип-диапазон есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона.

Тип-диапазон задаётся границами своих значений внутри базового типа:

<мин.знач.>..<макс.знач.>

Здесь <мин.знач.> - минимальное значение типа-диапазона.

<макс.знач.> - максимальное его значение.

```
Type
    Digit = '0' ..'9';
    Dig2 = 48 .. 57;
```

При определении типа-диапазона нужно руководствоваться следующими правилами:

1. два символа «..» рассматриваются как один символ, поэтому между ними недопустимы пробелы.

2. левая граница диапазона не должна превышать его правую границу.

Тип-диапазон наследует все свойства базового итпа, но с ограничениями, связанными с его меньшей мощностью. Над переменными, относящимися к интервальному типу, могут выполняться все операции и применяться все стандартные функции, которые допустимы для соответствующего базового типа.

Таблица 3 - Стандартные процедуры и функции, применимые к целым типам

Обращение	Тип результата	Действие
abs (x)	x	Возвращает модуль x
chr(b)	Char	Возвращает символ по его коду
dec (vx [, i])	-	Уменьшает значение vx на i, а при отсутствии i - на 1
inc(vx[,i])	-	Увеличивает значение vx на i, а при отсутствии i -на 1
Hi(w)	Byte	Возвращает старший бант аргумента
Hi(I)	То же	Возвращает третий по счету байт
Lo(i)	"	Возвращает младший байт аргумента
Lo(w)	"	То же
odd(l)	Boolean	Возвращает True, если аргумент-нечетное число
Random(w)	Как у параметра	Возвращает псевдослучайное число, равномерно распределенное в диапазоне 0...(w-l) Чтобы активировать генератор случайных чисел, вызовите процедуру Randomize.
sqr(x)	X	Возвращает квадрат аргумента
swap(i)	Integer	Меняет местами байты в слове
swap (w)	Word	Тоже

Пример:

```
Var x : word;
begin
x := swap(1234); { 3412 }
end.
```

Пример генератора случайных чисел от 2 до 5

```
Var x:integer;
begin
randomize;
x :=random (4)+2;
end
```

.

ВЕЩЕСТВЕННЫЕ ТИПЫ

Определение: ***Вещественный тип*** *– это простой стандартный тип, предназначенный для хранения подмножества вещественных чисел и выполнения операций над ними.*

В отличие от порядковых типов, значения которых всегда сопоставляются с рядом целых чисел и, следовательно, представляется в ПК абсолютно точно, значения вещественных типов определяют произвольное число лишь с некоторой конечной точностью, зависящей от внутреннего формата вещественного числа.

Вещественные числа могут быть представлены в виде с фиксированной точкой и с плавающей точкой. Первый вид вам хорошо известен - это обычная запись чисел: 0.1, 3.14 и т.д. Числа с плавающей точкой имеют формат mE+p, где 'm'- целое или дробное число, а E+p означает 10 в степени 'p'.

*Пример: 10E-03 = 10 * 10-3 = 0.01*
*3.14E00 = 3.14 * 100 = 3.14*

Таблица 4 Вещественные типы

Длина, байт	Название	Количество значащих цифр	Диапазон десятичного порядка
6	Real	11…12	-39…+38
4	Single	7…8	-45…+38
8	Double	15…16	-324…+308
10	Extended	19…20	-4951…+4932
8	Comp	19…20	$-2*10^{63}+1…+2*10^{63}-1$

Особое положение в Паскаль занимает тип COMP, который трактуется как вещественное число без экспоненциальной и дробной частей. Фактически, COMP - это «большое» целое число со знаком, сохраняющее 19...20 значащих десятичных цифр (во внутреннем представлении COMP занимает 8 смежных байт). В то же время в выражениях COMP полностью совместим с любыми другими вещественными типами: над ним определены все вещественные операции, он может использоваться как аргумент математических функций и т.д. Наиболее подходящей областью применения типа COMP являются бухгалтерские расчеты: денежные суммы выражаются в копейках или центах и действия над ними сводятся к операциям с достаточно длинными целыми числами

Над действительными числовыми типами данных можно выполнять следующие операции:

1.**Операции отношения (>,<,<>,>=,<=).**
Желательно избегать равенства, т.к. значения приблизительные (с разной точностью).
Можно использовать в операторах сравнения if, в операторах цикла while и repeat.
2.**Операции присваивания.**
Примеры присвоения значения:

```
c := -125.6;
e := 10;
d := 1.e2;     {1*10^2}
d1 := 1.2e-257; { 1.2*10^-257 }
b := 32*a+Sin(x);
```

3.**Арифметические операции (+, -,*,/)**

При записи формул (выражений) в программе следует учитывать такую особенность: компьютер будет в первую очередь выполнять вычисления в скобках, затем операции умножения и деления, и в последнюю очередь- сложение и вычитание:

1. ( )
2. *, /
3. +, -

Операции с равным приоритетом (+ , -), (*, /) выполняются слева направо в том порядке, как записаны в выражении. Если у программиста нет уверенности в том, что вычисления будут выполняться в нуж-

ном порядке, то лучше расставить лишние скобки, помня о том, что вычисления в скобках выполняются в первую очередь. Например, в выражении A+B*C/D-E, согласно правилам языка Паскаль, сначала будет выполнено B*C, затем результат делится на D, и только после этого наступит очередь сложения и вычитания: $A + \frac{B \cdot C}{D} - E$

Если сложение и вычитание требуется выполнить в первую очередь, то следует соответствующим образом расставить скобки: (A+B)*C/(D-E), что в форме алгебраической записи будет выглядеть так: $\frac{A + B}{D - E} \cdot C$

4.Стандартные функции

Для работы с вещественными данными могут использоваться встроенные математические функции,

Таблица 5 Стандартные математические функции Паскаль

Обращение	Тип параметра	Тип результата	Примечание
abs (x)	Real	Real	Модуль аргумента
ArcTan (x)	Real	Real	Арктангенс (значение в радианах)
cos (x)	То же	То же	Косинус, угол в радианах
exp (x)	"	"	Экспонента
frac (x)	"	"	Дробная часть числа
int(x)	"	"	Целая часть числа
ln(x)	"	"	Логарифм натуральный
Pi	-	"	л = 3.141592653...
Random	-	"	Псевдослучайное число, равномерно распределенное в диапазоне 0...[1]
Randomize	-	-	Инициация генератора псевдослучайных чисел
sin(x)	Real	Real	Синус, угол в радианах
sqr (x)	То же	То же	Квадрат аргумента
sqrt (x)	"	"	Корень квадратный

Таблицы преобразования данных.

Таблица 6. Работа с переменными целого и вещественного типа.

Имя и параметры	Процедура или функция	Типы параметров	Тип возвращаемого значения	Действие
Abs(x)	функция	x - integer, real, complex	совпадает с типом параметра	возвращает абсолютное значение (модуль) x
Sqr(x)	функция	x - integer, real, complex	совпадает с типом параметра	возвращает квадрат x
Sqrt(x)	функция	x - real, complex	совпадает с типом параметра	возвращает квадратный корень из x
Round(x)	функция	x - real	integer	возвращает результат округления x до ближайшего целого
Trunc(x)	функция	x - real	integer	возвращает целую часть x
Int(x)	функция	x - real	real	возвращает целую часть x
Frac(x)	функция	x - real	real	возвращает дробную часть x

В Паскаль отсутствует функция возведения числа в степень, а также нет тригонометрических функций tg(), ctg(), arcsin() и arccos(), но они могут быть реализованы сочетанием стандартных функций:

X^n = EXP(N*LN(X));
tg(X) = SIN(X)/COS(X);

ctg(X) = COS(X)/SIN(X);
arcsin(X) = ARCTAN(X/SQRT(1-SQR(X)));
arccos(X) = ARCTAN(SQRT(1-SQR(X))/X).

ПРИОРИТЕТ ОПЕРАЦИЙ

Порядок вычисления выражения определяется старшинством (приоритетом) содержащихся в нем операций. В языке Паскаль принят следующий приоритет операций:

- унарная операция not, унарный минус -, взятие адреса @
- операции типа умножения:* / div mod and
- операции типа сложения: + - or xor
- операции отношения: = <> < > <= >= in

Порядок выполнения операций переопределить можно с помощью скобок.

Например 2*5+10 равно 20, но 2*(5+10) равно 30.

Контрольные вопросы.

1. Дать понятие данных при составлении алгоритма решения задачи.
2. Перечислите и раскройте понятие видов данных (константы, переменные, литералы)
3. Что определяет тип данных?
4. Перечислите основные типы данных (простые, структурированные, указатели, строки, процедурные, объекты)
5. Расскажите принцип действия операций div и mod.
6. Какой результат дают операции отношения? (true или false)
7. Раскрыть результат выполнения стандартных функций Abs(X), Sqr(X), Succ(X), Pred(X), Exp(X), Sqrt(X), Odd(X).
8. Раскрыть результат выполнения стандартных процедур Inc(X), Inc(X,N), Dec(X), Dec(X,N).
9. Какие значения может принимать переменная логического типа?
10. Какие значения может принимать переменная символьного типа? Раскрыть работу функций *Ord(x)* и *Chr(x)*?
11. Что представляет собой перечисляемый тип? Приведите пример переменной перечисляемого типа.
12. Какие стандартные функции могут быть применены переменным перечисляемого типа? (Pred, Succ, Ord)
13. Раскройте понятие и приведите пример ин6тервального типа данных.
14. Раскрыть результат функций Random(w), указать типы аргумента и результата.
15. Раскрыть результат функций Round(x), Trunc(x), Int(x), Frac(x), указать типы аргумента и результата.
16. Раскрыть порядок выполнения операций в выражении.

Простейшие операторы. Операторы ввода-вывода.

Перейдем теперь к изучению операторов — специальных конструкций языка Pascal.

Если говорить строго, то оператором называется (минимальная) структурно законченная единица программы.

Все операторы языка Pascal должны заканчиваться знаком «;» (точка с запятой), и ни один оператор не может разрываться этим знаком. Единственная возможность не ставить после оператора «;» появляется в том случае, когда сразу за этим оператором следует ключевое слово **end** или **else**.

К простейшим операторам языка Pascal относятся:

1. a := b; — присваивание переменной a значения переменной b. В правой части присваивания может находиться переменная, константа, арифметическое выражение или вызов функции.
2. ; — пустой оператор, который можно вставлять куда угодно, а также вычёркивать откуда угодно, поскольку на целостность программы это никак не влияет.
3. Операторные скобки, превращающие несколько операторов в один:

   Begin

   несколько операторов

   end;

Метки и безусловный переход

Метка помечает какое-либо место в тексте программы. Метками могут быть числа от 0 до 9999 или идентификаторы, которые в этом случае уже нельзя использовать для каких-либо иных нужд. Все метки должны быть описаны в специальном разделе label:

label список_всех_меток_через_запятую;

Меткой может быть помечен любой оператор программы

метка: оператор;

Любая метка может встретиться в тексте программы только один раз. Используются метки только операторами безусловного перехода **goto**:

```
goto метка;
```

Это означает, что сразу после оператора **goto** будет выполнен не следующий за ним оператор (как это происходит в обычном случае), а тот оператор, который помечен соответствующей меткой.

Вообще же использование безусловных переходов в структурном и надёжном программировании считается «дурным тоном».

Операторы ввода-вывода

Как мы уже говорили, любой алгоритм должен быть результативным. В общем случае это означает, что он должен сообщать результат своей работы потребителю: пользователю–человеку или другой программе.

Ввод данных

Для того чтобы получить данные, вводимые пользователем вручную, применяются команды

Read(<список_ввода>) и **ReadLn**(<список_ввода>).

Первая из этих команд считывает все предложенные ей данные, оставляя курсор в конце последней строки ввода, а вторая — сразу после окончания ввода переводит курсор на начало следующей строки.

Список ввода — это последовательность имён переменных, разделённых запятыми. Например, при помощи команды

ReadLn(k, x, c, s); {k : Byte; x : Real; c : Char; s : String}

программа может получить с клавиатуры данные сразу для четырёх переменных, относящихся к различным типам данных.

Вводимые значения необходимо разделять пробелами, а завершать ввод — нажатием клавиши Enter. Ввод данных заканчивается в тот момент, когда последняя переменная из списка ввода получила своё значение. Следовательно, вводя данные при помощи приведённой выше команды, вы можете нажать Enter четыре раза — после каждой из вводимых переменных, — либо же только один раз, предварительно введя все четыре переменные в одну строчку (обязательно нужно разделить их пробелами).

Типы вводимых значений должны совпадать с типами указанных переменных, иначе возникает ошибка. Поэтому нужно внимательно следить за правильностью вводимых данных.

Вообще, вводить с клавиатуры можно только данные базовых типов (за исключением логического). Если же программе всё–таки необходимо получить с консоли значение для **Boolean**–величины, придётся действовать более хитро: вводить оговоренный символ, а уже на его основе присваивать логической переменной соответствующее значение. Например:

```
WriteLn('Согласны ли Вы с этим утверждением? y - да, n - нет');
 ReadLn(c); {c : Char}
 case c of
   'y': b := True;
   'n': b := False;
    else WriteLn('Ошибка!');
 end;
```

Второе исключение: строки, хотя они и не являются базовым типом, вводить тоже разрешается. Признаком окончания ввода строки является нажатие клавиши Enter, поэтому все следующие за нею переменные необходимо вводить с новой строчки.

Вывод данных на монитор

Сделаем одно важное замечание: ожидая от человека ввода с клавиатуры, не нужно полагать, что он окажется ясновидящим и просто по мерцанию курсора на чёрном экране догадается, какого типа переменная нужна ожидающей программе. Перед тем как считывать что–либо с клавиатуры, необходимо сообщить пользователю, что именно он должен ввести: смысл вводимой информации, тип данных, максимальное и минимальное допустимые значения и т.п.

Для того, чтобы вывести на экран какое-либо сообщение, воспользуйтесь процедурой **Write**(<список_вывода>) или **WriteLn**(<список_вывода>).

Первая из них, напечатав на экране всё, о чем ее просили, оставит курсор в конце выведенной строки, а вторая переведёт его в начало следующей строчки.

Список вывода может состоять из нескольких переменных, записанных через запятую; все эти переменные должны иметь тип либо базовый, либо строчный. Например, **WriteLn**(a, b, c);

Форматированный вывод

Чтобы выводимые данные были легко читаемыми нужно либо позаботиться о пробелах между выводимыми переменными:

```
WriteLn(a, ' ', b, ' ', c);
```

либо задать для всех (или хотя бы для некоторых) переменных формат вывода:

```
WriteLn(a : 5, b, c : 20 : 5);
```

Первое число после знака «:» обозначает количество позиций, выделяемых под всю переменную, а второе — под дробную часть числа. Десятичная точка тоже считается отдельным символом.

Если число длиннее, чем отведённое под него пространство, количество позиций будет автоматически увеличено. Если же выводимое число короче заданного формата, то спереди к нему припишутся несколько пробелов. Таким образом можно производить вывод красивыми ровными столбиками, а также следить за тем, чтобы переменные не сливались.

Например, если a = 25, b = 'x', a c = 10.5, то после выполнения команды WriteLn(a : 5, ' ', b, c : 10 : 5) на экране или в файле будет записано следующее (подчерки в данном случае служат лишь для визуализации пробелов):

_ _ _25_x_ _10.50000

Особенно важен формат при выводе вещественных переменных. К примеру, если не указать формат, то число 10.5 будет выведено как 1.0500000000E+0001. Такой формат называется записью с плавающей точкой.

Пример простейшей программы на языке Pascal

```
program nachalo;
var s : String;
begin
 Write('Введи своё имя: ');
 ReadLn(s);
 WriteLn(s, ' ты молодец!');
end.
```

Контрольные вопросы.

1. Дать понятие оператора.
2. Перечислите и запишите простейшие операторы языка Паскаль.
3. Запишите и расскажите функции и отличия операторов ввода данных с клавиатуры.
4. Запишите и расскажите функции и отличия операторов вывода данных на монитор.
5. Раскрыть правила форматированного вывода данных.

СИМВОЛЫ И СТРОКИ В ПАСКАЛЬ

В большинстве применений компьютера алфавитно-цифровая информация используется наряду с числовой информацией. Прежде чем мы сможем написать программу, которая манипулирует алфавитно-цифровыми знаками (литерами), нам потребуется тип данных для их представления. для этих целей в языке Паскаль предусмотрен тип данных **char**. Переменная типа **char** может хранить **один символ.**

Задача. Написать программу, которая считывает две литеры и печатает больше, равна или меньше первая литера второй.

```
Program primer_char;
Var  First, Second : char;
Begin
 write ('Введите две литеры через пробел ');
 readln (First, Second);
 write ('Первая литера ');
 if First > Second  then  write ('больше второй. ')
                  else  if First = Second  then  write ('равна второй. ')
                                           else  write ('меньше второй. ');
End.
```

Так как char - порядковый тип, то к его значениям применимы следующие функции.

Succ - возвращает следующий символ литерного множества;

Pred - возвращает предыдущий символ литерного множества;

Ord - возвращает значение кода литеры;

Chr - возвращает значение литеры, является обратной по отношению к функции Ord.

Например,

Succ('0')='1' - символ, следующий за символом 0, равен символу 1.

Pred('3')='2' - символ, предшествующий символу 3, равен 2;

Chr(65)='A' - символ, соответствующий коду 65, равен A;

Ord('A')=65 - код символа A равен 65

Строки

Для обработки строковой информации в Турбо Паскаль введен строковый тип данных. Строкой в Паскале называется последовательность из определенного количества символов. Количество символов последовательности называется длиной строки. Синтаксис:

```
var s: string[n];
var s: string;
```

n - максимально возможная длина строки - целое число в диапазоне 1..255. Если этот параметр опущен, то по умолчанию он принимается равным 255.

Строковые константы записываются как последовательности символов, ограниченные апострофами. Допускается формирование строк с использованием записи символов по десятичному коду (в виде комбинации # и кода символа) и управляющих символов (комбинации ^ и некоторых заглавных латинских букв).

Пример:

```
'Текстовая строка'
#54#32#61
'abcde'^A^M
```

Пустой символ обозначается двумя подряд стоящими апострофами. Если апостроф входит в строку как литера, то при записи он удваивается.

Переменные, описанные как строковые с разными максимальными длинами, можно присваивать друг другу, хотя при попытке присвоить короткой переменной длинную лишние символы будут отброшены.

Выражения типа **char** можно присваивать любым строковым переменным.

В Турбо Паскаль имеется простой доступ к отдельным символам строковой переменной: i-й символ переменной st записывается как st[i]. Например, если st - это 'Строка', то st[1] - это 'С', st[2] - это 'т', st[3] - 'р' и так далее.

Строка имеет две разновидности длины:

- Общая длина **строки**, которая характеризует размер памяти, выделяемый **строке** при описании;
- Текущая длина **строки** (всегда меньше или равна общей длине), которая показывает количество смысловых символов **строки** в каждый конкретный момент времени.

Каждый символ строковой величины занимает 1 байт памяти и имеет числовой код в соответствии с таблицей кодов ASCII.

Код ASCII (American Code for Information Interchange – Американский стандартный код для обмена информацией) имеет основной стандарт и его расширение. Основной стандарт использует шестнадцатеричные коды 00-7F, расширение стандарта – 80-FF. Основной стандарт является международным и используется для кодирования управляющих символов, цифр и букв латинского алфавита; в расширении стандарта используются символы псевдографики и буквы национальных алфавитов.

32 пробел	48 0	64 @	80 P	96 `	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 #	51 3	67 C	83 S	99 c	115 s
36 $	52 4	68 D	84 T	100 d	116 t
37 %	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 ‘	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45 -	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	127

Действия со строками в Паскале

Операция слияния (сцепления, конкатенации) применяется для соединения нескольких строк в одну, обозначается знаком «+». Операция слияния применима для любых строковых выражений, как констант, так и переменных.

Операции отношения позволяют сравнивать строки на отношение равенства (=), неравенства (<>), больше (>), меньше (<), больше или равно (>=), меньше или равно (<=). В результате сравнения двух строк получается логическое значение (true или false). Сравнение строк производится слева направо посимвольно до первого несовпадающего символа, большей считается та строка, в которой первый несовпадающий символ имеет больший код в таблице кодировки. Если строки имеют различную длину, но в общей части символы совпадают, считается, что короткая строка меньше. Строки равны, если они имеют равную длину и соответствующие символы совпадают.

Пример действий со строками в Паскале:

‘строка’<>’строки’ (верно, т.к. не совпадают последние символы);

‘Abc’<’abc’ (отношение истинно, т.к. код символа ‘A’ равен 65 в десятичной системе счисления, а код символа ‘a’ – 97);

‘год’>’век’ (отношение верно, т.к. буква ‘г’ в алфавите стоит после буквы ‘в’, а, следовательно, имеет больший код).

Стандартные функции для работы со строками в Паскале

Copy (S, poz, n) выделяет из строки S, начиная с позиции poz, подстроку из n символов. Здесь S – любое строковое выражение, poz, n – целочисленные выражения.

Значение S	Выражение	Результат
‘строка символов’	Copy(S,3,3)	рок

Concat (s1, s2,...,sn) выполняет слияние строк s1, s2,...,sn в одну строку.

Выражение	Результат
Concat(‘язык’, ‘’, ‘Pascal’)	‘язык Pascal’

Length(S) определяет текущую длину строкового выражения S. Результат – значение целого типа.

Значение S	Выражение	Результат
‘(a+в)*c’	Length(s)	7

Pos(subS, S) определяет позицию первого вхождения подстроки subS в строку S. Результат – целое число, равное номеру позиции, где находится первый символ искомой подстроки. Если вхождение подстроки не обнаружено, то результат функции будет равен 0.

Значение S	Выражение	Результат
‘предложение’	Pos(‘e’, S)	3
‘предложение’	Pos(‘a’, S)	0

Стандартные процедуры для работы со строками в Паскале

Delete (S, poz, n) удаляет из строки S, начиная с позиции poz, подстроку из n символов. Здесь S – строковая переменная (в данном случае нельзя записать никакое другое строковое выражение, кроме имени строковой переменной, т.к. только с именем переменной связана область памяти, куда будет помещен результат выполнения процедуры); poz, n – любые целочисленные выражения.

Исходное значение S	Оператор процедуры	Конечное зн-е S
‘abcdefg’	Delete(s, 2, 3)	‘aefg’

Insert(subS, S, poz) вставляет в строку S, начиная с позиции poz, подстроку subS. Здесь subS – любое строковое выражение, S – строковая переменная (именно ей будет присвоен результат выполнения процедуры), **poz** – целочисленное выражение.

Исходное значение S	Оператор процедуры	Конечное зн-е S
‘рис. 2’	Insert(‘№’, S, 6)	‘рис. №2’

Процедуры преобразования типов в Паскале

Str(x, S) преобразует число x в строковый формат. Здесь x – любое числовое выражение, S – строковая переменная. В процедуре есть возможность задавать формат числа x. Например, str(x: 8: 3, S), где 8 – общее число знаков в числе x, а 3 – число знаков после запятой.

Оператор процедуры	Значение S
Str (sin(1):6:4, S)	‘0.0175’
Str (3456, S)	‘3456’

Val(S, x, kod) преобразует строку символов S в число x. Здесь S – строковое выражение, x – числовая переменная *(именно туда будет помещен результат)*, kod – целочисленная переменная *(типа integer)*, которая равна номеру позиции в строке S, начиная с которой произошла ошибка преобразования, если преобразование прошло без ошибок, то переменная kod равна 0.

Тип X	Оператор процедуры	Значение X	Значение kod
Real	Val(’12.34’, x, kod)	12.34	0
Integer	Val(’12.34’, x, kod)	12	3

В дополнение приведем некоторые функции, связанные с типом char, но которые тем не менее часто используются при работе со строками.

Chr(n: byte): char

Функция возвращает символ по коду, равному значению выражения n. Если n можно представить как числовую константу, то можно также пользоваться записью #n.

Ord(ch: char): byte;

В данном случае функция возвращает код символа ch.

UpCase(c: char): char;

Если с - строчная латинская буква, то функция возвращает соответствующую прописную латинскую букву, в противном случае символ с возвращается без изменения.

Контрольные вопросы.

1. Назовите и раскройте отличия типов данных работающих с символьной информацией.
2. Перечислите функции обработки переменных типа char.
3. Как можно объявить строковую переменную в Паскаль (указав и не указывая максимальную длину переменной)
4. Как в тексте программы строковая и символьная величины обозначаются? (в апострофах)
5. Раскройте какие действия можно выполнить над переменными строкового типа? (слияния, отношения)
6. Раскройте действия функций работы со строками. (Copy, Concat, Length, Pos)
7. Раскройте действия процедур работы со строками. (Delete, Insert)
8. Раскройте действия процедур преобразования типов.

Множества в языке Pascal

Множество — это структурированный тип данных, представляющий собой набор взаимосвязанных по какому-либо признаку или группе признаков объектов, которые можно рассматривать как единое целое. Каждый объект в множестве называется элементом множества.

Все элементы множества должны принадлежать одному из порядковых типов, содержащему не более 256 значений. Этот тип называется базовым типом множества. Базовый тип задается диапазоном или перечислением.

Область значений типа множество — набор всевозможных подмножеств, составленных из элементов базового типа. В выражениях на языке Паскаль значения элементов множества указываются в квадратных скобках: [1,2,3,4], ['a',‘b','c'], ['a'..'z'].

Если множество не имеет элементов, оно называется пустым и обозначается как []. Количество элементов множества называется его мощностью.

Множество может принимать все значения базового типа. Базовый тип не должен превышать 256 возможных значений. Поэтому базовым типом множества могут быть byte, char, boolean и производные от них типы.

Множество в памяти хранится как массив битов, в котором каждый бит указывает является ли элемент принадлежащим объявленному множеству или нет. Максимальное число элементов множества 256, а данные типа множество могут занимать не более 32 байт.

Число байтов, выделяемых для данных типа множество, вычисляется по формуле:

ByteSize = (max div 8) - (min div 8) + 1,

где max и min — верхняя и нижняя границы базового типа данного множества.

Номер байта для конкретного элемента E вычисляется по формуле:

ByteNumber = (E div 8) - (min div 8),

номер бита внутри этого байта по формуле:

BitNumber = E mod 8

Не имеет значения порядок записи элементов множества внутри конструктора.

Например, [1, 2, 3] и [3, 2, 1] — это эквивалентные множества.

Каждый элемент в множестве учитывается только один раз. Поэтому множество [1, 2, 3, 4, 2, 3, 4, 5] эквивалентно [1..5].

Переменные множественного типа описываются так:

Var <идентификатор> : set of <базовый тип>;

Например:

```
Var A, D : Set Of Byte;
    B : Set Of 'a'..'z';
    C : Set Of Boolean;
```

Нельзя вводить значения во множественную переменную процедурой ввода и выводить процедурой вывода.

Множественная переменная может получить конкретное значение только в результате выполнения оператора присваивания:

<множественная переменная> := <множественное выражение>;

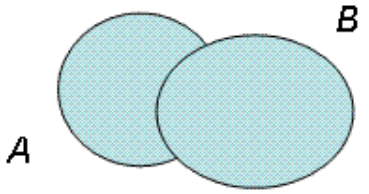
Например:

```
A : = [50, 100, 150, 200]; B : = ['m', 'n', 'k'];
C : =[True, False]; D : = A;
```

Кроме того, выражения могут включать в себя операции над множествами.

Операции над множествами

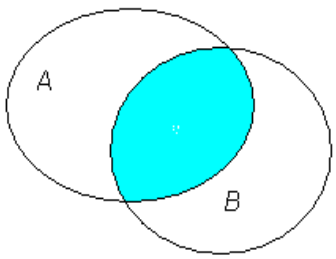
Объединением двух множеств A и B называется множество, состоящее из элементов, входящих хотя бы в одно из множеств A или B. **Знак операции объединения в Паскале «+».**



Примеры:

```
1) [1, 2, 3, 4] + [3, 4, 5, 6] => [1, 2, 3, 4, 5, 6]
2) []+['a'..'z']+['A'..'E', 'k']  =>
['A'..'E', 'a'..'z']
3) [5<4, true and false] + [true]  => [false, true]
```

Пересечением двух множеств A и B называется множество, состоящее из элементов, одновременно входящих во множество A и во множество B.

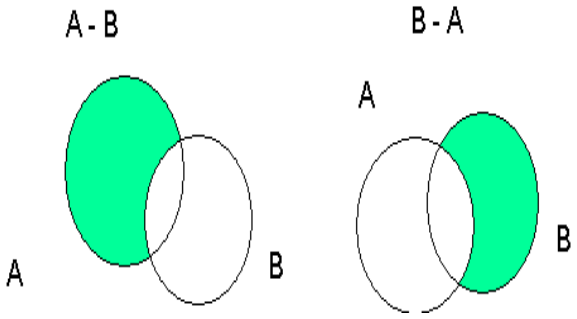


Знак операции пересечения в Паскале «*»

Примеры:

```
1) [1, 2, 3, 4] * [3, 4, 5, 6] => [3, 4]
2) ['a'..'z']*['A'..'E', 'k']  => ['k']
3) [5<4, true and false] * [true]  => []
```

Разностью двух множеств A и B называется множество, состоящее из элементов множества A, не входящих во множество B.



Примеры:

```
1a)  [1,  2,  3,  4] - [3,  4,  5,  6] =>
[1, 2]

1b)  [3,  4,  5,  6] - [1,  2,  3,  4]  =>
[5, 6]

2a)['a'..'z']-['A'..'E',     'k']     =>
['a'..'j', 'i'..'z']

2b)['A'..'E',   'k']   -   ['a'..'z']   =>
 ['A'..'E']

3a)    [5<4,    true    and    false]    -
[true] => [false]

3b)    [true]    -    [5<4,    true    and
false] => [true]
```

Операция вхождения. Это операция, устанавливающая связь между множеством и скалярной величиной, тип которой совпадает с базовым типом множества. Если x — такая скалярная величина, а M — множество, то операция вхождения записывается так: **x in M.**

Результат — логическая величина true, если значение x входит в множество M, и false — в противном случае.

Например, 4 in [3, 4, 7, 9] — true, 5 in [3, 4, 7, 9] — false.

Используя данную операцию, можно не только работать с элементами множества, но и, даже если в решении задачи явно не используются множества, некоторые логические выражения можно записать более лаконично.

1) Натуральное число n является двухзначным. Вместо выражения **(n >= 10) and (n <=99)** можно записать **n in [10..99]**.

2) Символ c является русской буквой. Вместо выражения **(c >= ‘А’) and (c <= ‘Я’) or (c>=‘а’) and (c<=‘п’) or (c>=‘р’) and (c<=‘я’)** пишем **c in [‘А’.. ‘Я’, ‘а’.. ‘п’, ‘р’.. ‘я’]** и т.д.

Добавить новый элемент в множество можно с использованием операции объединения. Например, a:= a+[5] Для этих же целей в Turbo Pascal предназначена процедура Include: **include (M, A)** M – множество, A – переменная того же типа, что и элементы множества M. Тот же пример можно записать так: Include (a, 5)

Исключить элемент из множества можно с помощью операции «разность множеств». Например, a:= a-[5] Для этих же целей в Turbo Pascal 7.0 предназначена процедура Exclude: **exclude (M, A)** M – множество, A – переменная того же типа, что и элементы множества M. Тот же пример можно записать так: Exclude (a, 5)

Контрольные вопросы и задания

1. Что такое множество?
2. Почему множество является структурированным типом данных?
3. Как хранится множество в памяти ЭВМ? Какой максимальный объем оперативной памяти может быть отведен под хранение одного множества?
4. Какие операции можно выполнять над множествами?
5. Как добавить элемент в множество?
6. Как исключить элемент из множества?
7. Как вывести элементы множества? Как подсчистать количество элементов в множестве?

8. Как может быть использована операция вхождения?

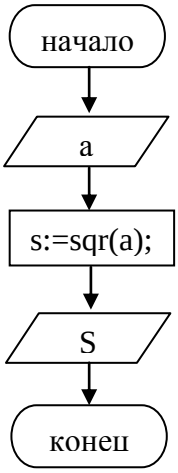
Реализация линейной и разветвляющейся алгоритмических конструкций в Паскаль.

Линейная структура алгоритма обеспечивает последовательное, обязательное, однократное выполнение каждого записанного оператора (действия) в коде программы. Нпример,

Вычислить площадь квадрата со стороной a.

```
program pl_kv;
 var s,a:real;
 begin
 write('vvedite storony a'); {строка – пояснение ко вводу данных для пользователя}
 readln(a);
 s:=sqr(a); {s:=a*a}
 write('s=',s:7:3);
 readln; {задержка экрана}
 end.
```

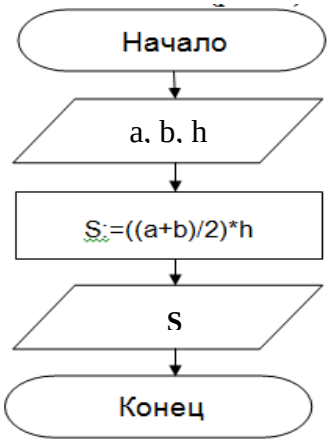
Составим блок-схему этой программы



Теперь решим у доски следующие задачи:

Задача 1. Составить программу вычисления $z = \dfrac{\cos(4x) - \sqrt{|x|}}{|2y| - 3} + \sqrt[3]{x \cdot y}$ для произвольных x, y.

Задача 2. По предложенной блок-схеме составить программу и определить какую задачу решает предложенный алгоритм



Задача 3. По предложенной программе составить блок-схему

```
var x, y, b: integer;
begin
   write ('Input first number: ');
   readln (x);
   write ('Input second number: ');
   readln (y);
   writeln ('x = ', x, ', y = ', y);
   b := x;       {или обмен значений можно произвести без буферной переменной x:=x+y }
   x := y;       {y:=x-y}
   y := b;       {x:=x-y}
 writeln ('x = ', x, ', y = ', y);
readln
end.
```

Конструкция ветвления (условный оператор).

1) Конструкция ветвления

Конструкция ветвления – это способ организации действий, когда в зависимости от выполнения или невыполнения некоторого условия выполняется только одно из двух действий. **Условие** – это логическое выражение, которое принимает значение ИСТИНА (TRUE, выполняется) или ЛОЖЬ (FALSE, не выполняется).

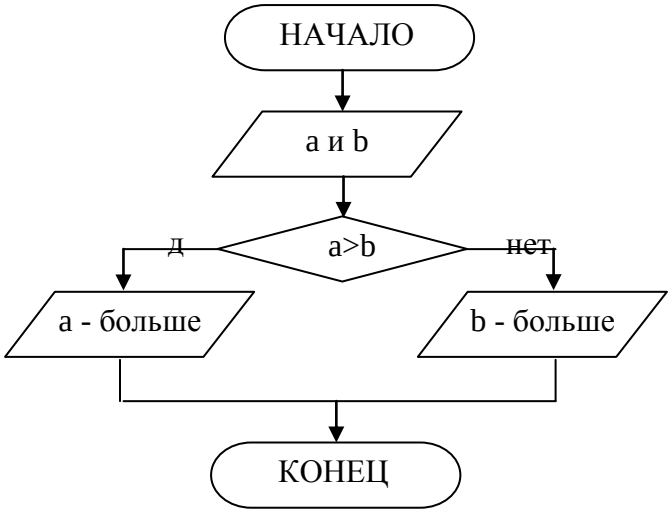
Она используется для выбора одного из двух направлений дальнейшего хода программы. Выбор последующих действий осуществляется во время выполнения программы в зависимости от выполнения условия. В программировании используется два вида ветвления. Рассмотрим оба на примерах:

Полное ветвление

Задача 1. Найти наименьшее из двух неравных чисел a и b.

Решение:

1) Введем два неравных числа a и b.
2) Если a>b, то выведем ‘a – наибольшее', иначе выведем ‘b – наибольшее’.



Механизм работы полного ветвления:

1) Определяем значение условия (истина или ложь).
2) Если условие истинно, то выполняем Действие 1 и идем к пункту 4.
3) Если условие ложно, то выполняем Действие 2.
4) Переходим к следующему действию алгоритма.

Таким образом, конструкция полного ветвления на языке псевдокода имеет вид:

Если <условие>
то <Действие 1>
иначе <Действие 2>

конец если

На языке Pascal она будет записываться следующим образом:

If *<условие>* ***Then*** *<Действие 1>*
Else *<Действие 2>;*

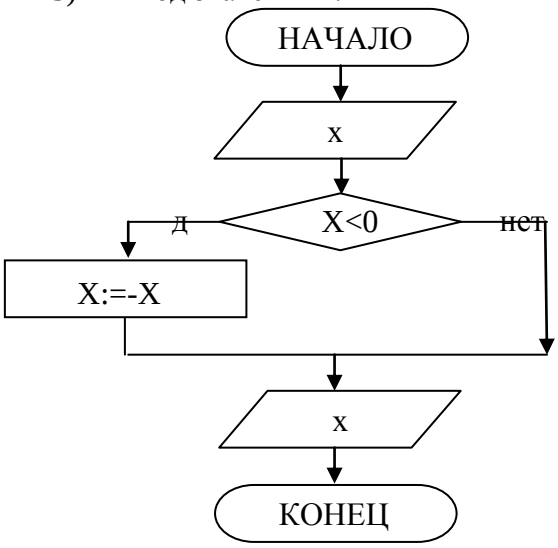
Необходимо заметить, что перед Else (после Действия 1) точку с запятой ставить не надо.

Неполное ветвление

Задача 2. Найти модуль числа x.

Решение:

1) Введем число x.
2) Если x<0, то поменяем его знак.
3) Вывод значения x.



Механизм работы неполного ветвления:

1) Определяем значение условия (истина или ложь).
2) Если условие истинно, то выполняем Действие 1 (если условие ложно, то ничего не делаем).
3) Переходим к следующему действию алгоритма.

Таким образом, конструкция неполного ветвления на языке псевдокода имеет вид:

Если <условие>
то <Действие 1>
конец если

На языке Pascal она будет записываться следующим образом:

If *<условие>* ***Then*** *<Действие 1>;*

Необходимо заметить, что ветвь Else отсутствует, поэтому после Действия 1 сразу ставим точку с запятой.

Программа для задачи 1 имеет вид:

```
Program Ex1;
Var a, b: Integer;

Begin
    Writeln('Введите два целых неравных числа');
    Readln(a, b);
    If a>b Then Writeln('a – наибольшее')
        Else Writeln('b – наибольшее');
    Readln;
End.
```

Программа для задачи 2 имеет вид:

```
Program Ex2;
Var x: Integer;

Begin
    Writeln('Введите целое число');
    Readln(x);
    If x<0 Then x:=-x;
    Writeln('x=', x);
    Readln;
End.
```

Итак, **полное ветвление используется тогда, когда мы должны выполнять какие-либо действия в обоих случаях, а если необходимо выполнять действие только в одном случае, то используется неполное ветвление**. В каждое ветви можно использовать другое ветвление.

Задача 3. Даны три неравных числа a, b и c. Вывести на экран большее из них.

Решение:

1) Вводим три неравных числа: a, b, c.
2) Если a>b То
   Если a>c То Max:=a

      Иначе Max:=c

   Конец если

   Иначе

      Если b>c То Max:=b

         Иначе Max:=c

      Конец если

   Конец если

3) Выводим Max

Программа для задачи 3 имеет вид:

```
Program Ex3;

Var a, b, c, max: Integer;

Begin

    Writeln ('Введите три целых числа');

    Readln (a, b, c);

    If a>b Then

        If a>c Then max:=a

            Else max:=c

    Else

    If b>c Then max:=b

        Else max:=c;

Writeln ('max=', max);
```

Readln; End.

2) Условия

Условия бывают:

- Простые – в языке программирования Pascal они обычно записываются в виде неравенств с использованием знаков: < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), = (равно) и < > (не равно)
- Сложные - составляются из нескольких простых условий с использование логических операций And (союз «и»), Or (союз «или»), Not (отрицание «не»).

Примеры сложных условий:

1) Например, в математике часто используются интервалы, промежутки и отрезки:

- Переменная x принимает значения из интервала от –6 до 9 – на языке математики это будет записано так: x∈(-6; 9). Это условие можно записать двойным неравенством -6<x<9 или системой двух простых неравенств $\begin{cases} x > -6 \\ x < 9 \end{cases}$. На языке программирования Pascal это условие будет записано следующим образом: (x>-6) **And** (x<9).
- Переменная x принимает значение их промежутка от 10 до 23, включая 23 – на языке математики это будет записано так: x∈(10; 23]. Это условие можно записать двойным неравенством 10<x≤23 или системой двух простых неравенств $\begin{cases} x > 10 \\ x \le 23 \end{cases}$. На языке программирования Pascal это условие будет записано следующим образом: (x>10) **And** (x<=23).
- Условие a∈(-2; 4)∪[7; 12) на языке программирования будет записано так: ((a>-2) **And** (a<4)) **Or** ((a>=7) **And** (a<12)).
- Условие |a|>2 можно записать так: (a>2) **Or** (a<-2)

2) Также часто используется кратность чисел. Напомним, что число a кратно числу b, если число a делится на число b без остатка

- Число a кратно 3 или 7 – это условие соответствует записи: (a **Mod** 3=0) **Or** (a **Mod** 7=0).
- Число a кратно 3 и 7 – это условие соответствует записи: (a **Mod** 3=0) **And** (a **Mod** 7=0).
- Число a кратно 3 и не кратно 7 – это условие соответствует записям: (a **Mod** 3=0) **And Not**(a **Mod** 7=0) или (a **Mod** 3=0) **And** (a **Mod** 7< >0).

Задача. Определить, является ли число x трехзначным.

Решение:

1) Введем число x.
2) Если 100≤x≤999, то выводим ‘число трехзначное’, иначе выводим ‘число не трехзначное’.

```
...
Write ('введите число x=');
Readln (x);
If (X<=999) And (x>=100) Then Writeln ('число трехзначное')
        Else Writeln ('число не трехзначное'); ...
```

Теперь решим у доски следующие задачи:

Задача 1. Составить программу определения максимального (минимального) из двух чисел a, b.

Задача 2. Составить программу определения максимального (минимального) из трех чисел a, b, c.

Задача 3. Составить программу нахождения корней квадратного уравнения $ax^2 + bx + c = 0$ (коэффициенты a,b,c вводятся с клавиатуры)

Контрольные вопросы.

1. Возможна ли ситуация что в блок-схеме пять блоков, а в программе строк гораздо больше? Почему?

2. Раскрыть способы обмена значений двух переменных.
3. Записать опрераторы реализации полного и неполного ветвления в Паскаль.
4. Записать логические операторы, позволяющие реализовать сложные условия.
5. В каких случаях следует применять операторные скобки при реализации ветвления?

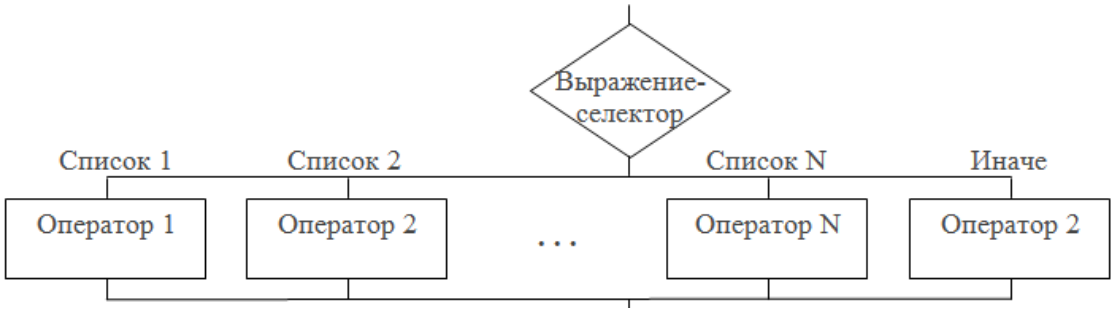
Оператор CASE OF

Оператор CASE является обобщением оператора IF и позволяет сделать выбор из произвольного числа имеющихся вариантов.

Формат оператора варианта:

```
CASE <Выражение-селектор> OF
        < Список 1>: <Оператор 1 >;
        <Список 2>: <Оператор 2 >;
        …
        <Список N>: <Оператор N >
        [ ELSE <Оператор N+1> ];
END;
```

В этой конструкции <Выражение селектор> может быть любого перечисляемого типа: стандартного (INTEGER, BOOLEAN, CHAR и т.д.) или пользовательского. <Список i> представляет собой подмножество значений, разделенных через запятую, выражения-селектора при которых следует выполнить <оператор i>. Оператор ELSE может отсутствовать. Если выражение-селектор принимает значение, которое не входит ни в один из списков 1,2,N, то в этом случае выполняется оператор N+1, стоящий за ELSE. Когда оператор ELSE отсутствует, вместо оператора N+1 выполняется пустой оператор. Далее приведен алгоритм оператора CASE ... OF:



Вариант, помеченный словом 'Иначе', соответствует ветке ELSE оператора CASE. Если ELSE отсутствует, то в блок-схеме нет этой ветви. Оператор CASE ... OF работает следующим образом:

- Вычисляется значение выражения селектора – G.
- Значение G сравнивается с множеством значений, представленных в списке 1. Если в списке есть такое значение, то выполняется оператор 1.
- Если в списке 1 нет значения G, то проверяется список 2. Если найдено в списке 2 значение G, то выполняется оператор 2 и т.д.
- Если значение G не найдено ни в одном из списков с номерами 1, 2, 3,..., N, то выполняется оператор N + 1.
- Если в операторе CASE нет ветки ELSE и значение G не найдено ни в одном из списков 1, 2, 3,..., N, то выполняется пустой оператор.

Пример 1. Составить программу для вычисления площади одной из пяти геометрических фигур (прямоугольника, треугольника, трапеции, круга и сектора).

```
var s, a, в, н, r, fi: real; n: integer;
begin
writeln('l - прямоугольник');
writeln('2 — треугольник');
```

```
writeln('3 - трапеция');
writeln('4 - круг');
writeln('5 - сектор');
readln(n);
s:=0;
case n of
1: begin
     writeln('введите стороны - a, в');
     read(a, в);
     s:= a*b;
   end;
2: begin
     writeln('введите основание и высоту - a, н');
     read(a, н);
     s:=a*h/2;
   end;
3: begin
     writeln('введите основания и высоту - a, в, н');
     read(a, в, н);
     s:= (a + в)*н/2;
   end;
4: begin
     writeln('введите радиус – r');
     read(r);
     s:= pi*r*r;
   end;
5: begin
     writeln('введите радиус и угол, град - r, fi');
     read(r, fi);
     s:=pi*r*r*fi/360;
   end;
end; {case}
writeln(фигуры = ', s:8:2)
end.
```

В этой программе оператор CASE используется для организации простейшего меню для выбора фигуры по предлагаемому номеру из списка номеров. Оператор указывает номер фигуры, и управление передается соответствующей ветви вычислительного процесса.

Пример 2. Для заданного целого положительного K и значения вещественного числа X вычислить Y = F(X) по формуле:

$$Y=\begin{cases} X^K+X+1, & K=2 \quad или \quad K=3; \\ \dfrac{1}{|X+1|}, & K=4 \quad или \quad K=5 \quad или \quad K=6; \\ \sqrt{|X+K|}+\sqrt{|X-K|}, & K>7 \quad или \quad K<2. \end{cases}$$

```
var    x, y: real; к: integer;
gin   writeln('введите x и к');
readln(x, k);
к of
: y := exp(k *ln(x)) + x + 1;    { список 1}
6: y:= 1/abs(x+1);    {список 2}
y:= sqrt(abs(x +k))+sqrt(abs(x-k)) { в противном случае}
end; { case}
writeln('y = ',y:5:1)
end.
```

be-

case

2, 3

4 ..

else

Реализация циклических алгоритмических конструкций в Паскаль

В большинстве задач, встречающихся на практике, необходимо производить многократное выполнение некоторого действия. Такой многократно повторяющийся участок вычислительного процесса называется ***циклом***.

Цикл с параметром

```
for переменная := значение 1 to значение 2 do оператор
```

или

```
for переменная := значение 1 downto значение 2 do оператор
```

Оператор for вызывает *оператор*, находящийся после слова do, по одному разу для каждого значения в диапазоне от *значения 1* до *значения 2*.

Переменная цикла, начальное и конечное значения должны иметь порядковый тип. Со словом to, значение переменной цикла **увеличивается** на 1 при каждой итерации цикла. Со словом downto, значение переменной цикла **уменьшается** на 1 при каждой итерации цикла. Не следует самостоятельно изменять значение управляющей переменной внутри цикла.

Следует помнить, что синтаксис языка допускает запись только одного оператора после ключевого слова do, поэтому, если вы хотите в цикле выполнить группу операторов, обязательно надо объединить их в составной оператор (операторные скобки begin ... end).

Пример 1. Квадраты чисел от 2-х до 10-и.

```
for x:=2 to 10 do WriteLn(x*x);
```

Пример 2. Латинский алфавит.

```
for ch:='A' to 'Z' do Writeln(ch);
```

Пример 3. Использование цикла с downto.

```
for i:=10 downto 1 do WriteLn(i);
```

Пример 4. Использование составного оператора.

```
for x:=1 to 10 do begin
     y:=2*x+3;
     WriteLn('f(',x,')=',y);
   end;
```

Операторы завершения цикла

Для всех операторов цикла выход из цикла осуществляется как вследствие естественного окончания оператора цикла, так и с помощью операторов перехода и выхода.

В версии Паскаль определены стандартные процедуры:

```
Break

Continue
```

Процедура Break выполняет безусловный выход из цикла. Процедура Continue обеспечивает переход к началу новой итерации цикла.

Заметим, что хотя и существует возможность выхода из цикла с помощью оператора безусловного перехода goto, делать этого не желательно. Во всех случаях можно воспользоваться специально предназначенными для этого процедурами Break и Continue.

ЗАДАНИЕ

Дописать недостающую строку в программе и составить блок-схему.

Найти все простые делители числа К.

Решение:

```
Program deliteli;
Var f:  boolean; i, j: byte; K: real;
Begin
Writeln('введите число'); Read (K);
F:=false;
For  i:=1  to  K  do
    Begin
        If  k mod I = 0  then      {проверяем является ли I делителем К}
            Begin
                for  j:=2  to  int(i/2)   do   if  I mod j = 0   then  f:=true; { проверяем является ли I простым числом }
                ??????????????????????????????????????
            End;
    End;
Readln;
End.
```

Циклическая структура WHILE

Оператор цикла *while* аналогичен оператору *repeat*, но проверка условия выполнения тела цикла производится в самом начале оператора.

Инструкция WHILE

WHILE условие **DO**
begin
тело цикла
end ;

Проверяется значение выражения *условие* (выражение логического типа), если оно равно *True* (условие выполняется), то выполняются инструкции, находящиеся между *begin* и *end* (инструкции цикла). Затем снова проверяется значение выражения *условие*, и так продолжается до тех пор, пока значение выражения *условие* не станет равным *False*. Таким образом, после слова *while* записывается условие выполнения инструкций цикла.

Примечание: если между *begin* и *end* находится только одна инструкция, то слова *begin* и *end* можно не писать.

Не следует забывать, что:

- число повторений инструкций цикла *while* определяется ходом выполнения программы;
- инструкции цикла *while* выполняются до тех пор, пока условие, записанное после слова *while*, истинно;
- после слова *while* надо записывать условие выполнения инструкций цикла;
- для завершения цикла *while* в теле цикла обязательно должны присутствовать инструкции, влияющие на условие выполнения инструкций цикла;
- цикл *while* - это цикл с предусловием, т.е. инструкции тела цикла вообще могут быть не выполнены;
- цикл *while*, как правило, используется для организации приближённых вычислений, задач поиска и обработки данных, вводимых с клавиатуры или из файла.

Рассмотрим примеры, иллюстрирующие использование данного оператора.

Пример

```
Вычисление числа "Пи"
Program PI ;
var p : real ; {вычисляемое значение Пи}t : real ; {точность вычисления}n : integer ; {номер члена ряда}elem
: real ; {значение члена ряда}
begin
  p : = 0 ; n : = 1 ;   elem : = 1 ;
  write ('Задайте точность вычисления ПИ ->') ;
  readln (t) ;
  while elem >= t do
     begin
       elem : = 1/ (2*n-1) ;
       if (n MOD 2) = 0   then p : = p – elem  else p : = p + elem ;
       n : = n + 1) ;
    end;
  p : = p*4 ;
  writeln ('Значение Пи с точноттью', t : 9 : 6, 'равно', p : 9 : 6) ;
  writeln ('Просуммировано', n , 'членов ряда.') ;  readln ;
end.
```

Repeat Until-цикл с постусловием.

Состоит из заголовка **REPEAT** , тела и условия окончания цикла **UNTIL**. **Repeat Until** формально переводится как **"повторять,пока"** (то биш повторять выполнение тела цикла,пока оно верно) . В цикле типа repeat ... until тело цикла помещается между зарезервированными словами языка (лексемами) repeat и until, операторные скобки не требуются, в названии цикла его тело условно обозначается тремя точками.

Особенности:

- выполняется всегда хотя-бы один раз
- возможно зацикливание,если условие-всегда ложно
- условием завершения цикла является истина условия (противоположен циклу **WHILE**) после ключевого слова **UNTIL**

Пример:

```
…
Repeat
Readln(n);
Until N >= 100;
…
```

Данный цикл выполняется пока вводимое значение **N** меньше 100.

Задание. Запишите решение задачи «Вычисление числа "Пи"» с использованием цикла с постусловием.

Контрольные вопросы.

1. Раскройте принцип работы каждого вида цикла.
2. Как осуществляется изменение параметра цикла в цикле с параметром?
3. Как обозначается тело цикла в каждом из циклов?
4. Как можно досрочно выйти из цикла?
5. Нарисовать блок-схемы реализующие каждый вид цикла.

Реализация основных алгоритмических конструкций. Составление программного кода по блок-схеме

Задание1.

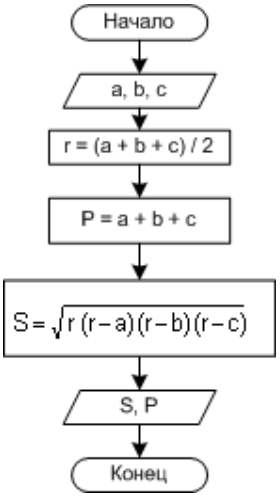
Зная длины трех сторон треугольника, вычислить площадь и периметр треугольника.

Пусть **a, b, c** - длины сторон треугольника. Необходимо найти **S** - площадь треугольника, **P** - периметр.

Для нахождения площади можно воспользоваться формулой Герона: $S = \sqrt{r(r-a)(r-b)(r-c)}$, где **r** - полупериметр.

Входные данные: **a, b, c**. *Выходные данные:* **S, P**.

Блок-схема алгоритма



Задание 2.

Заданы коэффициенты **a, b** и **c** биквадратного уравнения $\mathbf{ax^4 + bx^2 + c = 0}$. Решить уравнение.

Для решения биквадратного уравнения необходимо заменой $\mathbf{y = x^2}$ привести его к квадратному и решить это уравнение.

Входные данные: **a, b, c**. Выходные данные: $\mathbf{x_1, x_2, x_3, x_4}$.

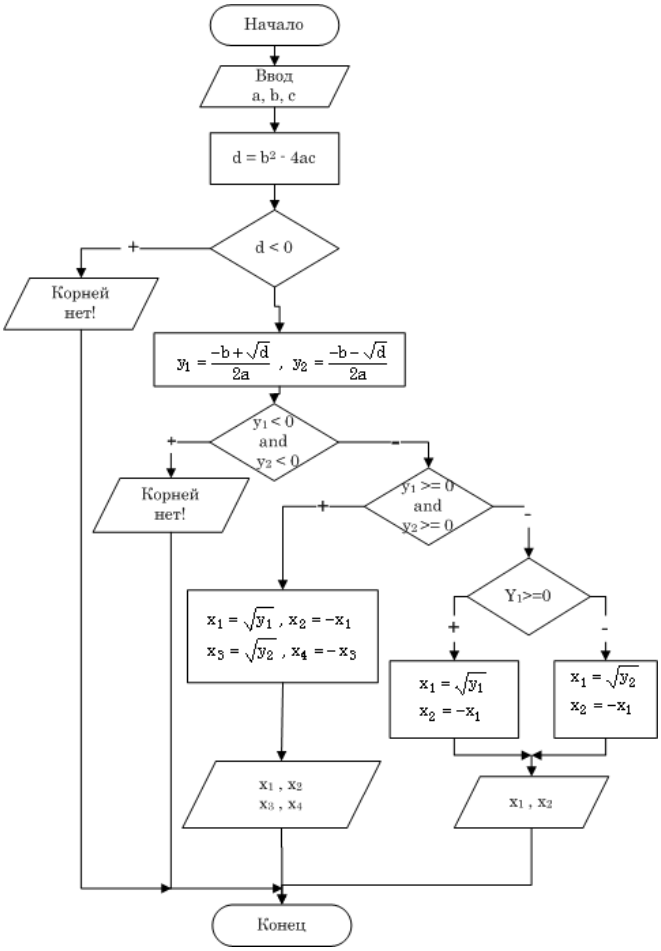
Блок-схема представлена на рис. 1.13. Алгоритм состоит из следующих этапов:

1. Вычисление дискриминанта уравнения **d.**
2. Если d >= 0, определяются $\mathbf{y_1}$ и $\mathbf{y_2}$, а иначе корней нет.
3. Если $\mathbf{y_1}$, $\mathbf{y_2}$ < 0 , то корней нет.
4. Если $\mathbf{y_1}$, $\mathbf{y_2}$ >= 0 , то вычисляются четыре корня по формулам и выводятся значения корней.

$$\pm\sqrt{y_1},\ \pm\sqrt{y_2} \qquad (1)$$

5. Если условия 3) и 4) не выполняются, то необходимо проверить знак $\mathbf{y_1}$. Если $\mathbf{y_1}$ >= 0, то вычисляются два корня по формуле 1.4. Если же $\mathbf{y_2}$ >= 0, то вычисляются два корня по формуле 2. Вычисленные значения корней выводятся.

$$\pm\sqrt{y_1} \qquad (1.4) \qquad\qquad \pm\sqrt{y_2} \qquad (2)$$



Задание 3.

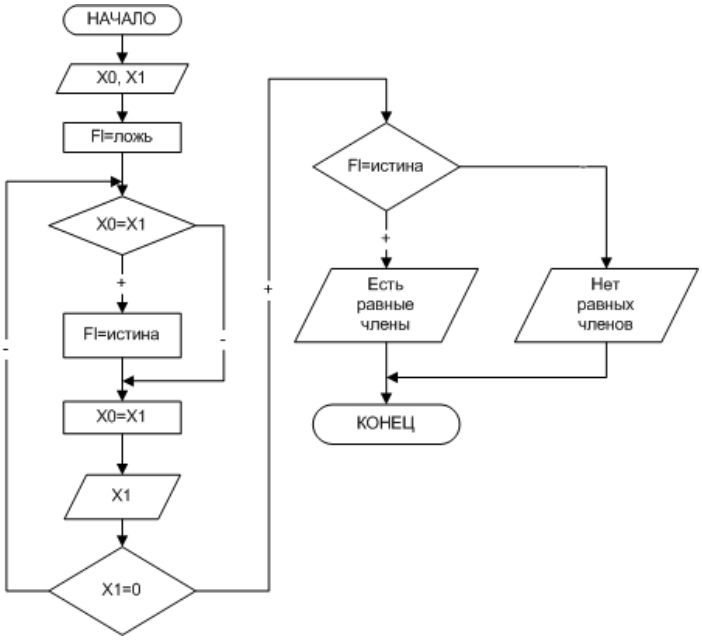
Вводится последовательность чисел, 0 - конец последовательности. Определить, содержит ли последовательность хотя бы два равных соседних числа.

Входные данные: $\mathbf{x_0}$ - текущий член последовательности, $\mathbf{x_1}$ - следующий член последовательности.

Выходные данные: сообщение о наличии в последовательности двух равных соседних элементов.

Вспомогательные переменные: $\mathbf{F_l}$ - логическая переменная, сохраняет значение "истина", если в последовательности есть равные рядом стоящие члены и "ложь" - иначе.

Блок-схема решения задачи. Применение здесь цикла с постусловием обосновано тем, что необходимо вначале сравнить два элемента последовательности, а затем принять решение об окончании цикла.

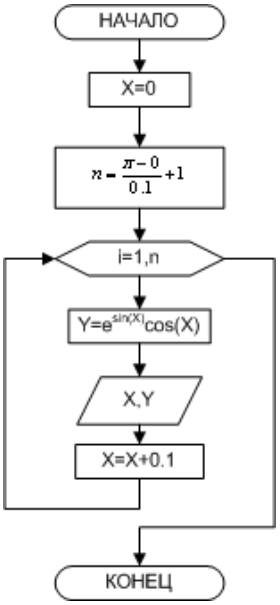


Задание 4.

Составить таблицу значений функции $\mathbf{y = e^{sin(x)}cos(x)}$ на отрезке $[0;\boldsymbol{\pi}]$ с шагом 0.1.

Входные данные: начальное значение аргумента - 0, конечное значение аргумента - $\boldsymbol{\pi}$, шаг изменения аргумента - 0.1.

Выходные данные: множество значений аргумента **X** и соответствующее им множество значений функции **Y**.



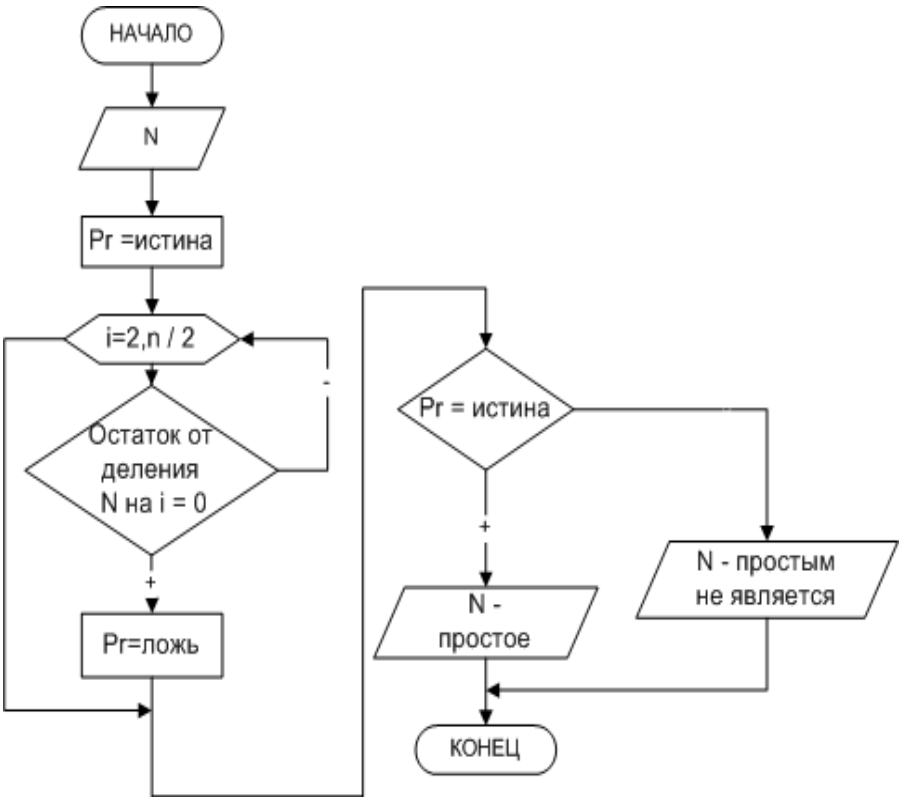
Задание 5.

Дано натуральное число **N**. Определить, является ли оно простым. Натуральное число **N** называется **простым**, если оно делится нацело без остатка только на единицу и **N**. Число 13 - простое, так как делится только на 1 и 13, **N = 12** не является простым, так как делится на 1, 2, 3, 4, 6 и 12.

Входные данные: **N** - целое число.

Выходные данные: сообщение.

Промежуточные данные: **i** - параметр цикла, возможные делители числа **N**.



Алгоритм решения этой задачи заключается в том, что число **N** делится на параметр цикла **i**, изменяющийся в диапазоне от 2 до **N/2**. Если среди значений параметра не найдется ни одного числа, на которое заданное число **N** делится нацело, то **N** - простое число, иначе оно таковым не является. Обратите внимание на то, что в алгоритме предусмотрено два выхода из цикла. Первый - естественный, при исчерпании всех значений параметра, а второй - досрочный. Нет смысла продолжать цикл, если будет найден хотя бы один делитель из указанной области изменения параметра.

Структурированные типы данных: массивы

Одномерные массивы.

Массив - это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его *компонентой* (или *элементом* массива). (**Массив** – структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип. Элементы, образующие массив, упорядочены таким образом, что каждому элементу соответствует номер (индекс), определяющий его местонахождение в общей последовательности.)

Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент.

Переменные, представляющие компоненты массивов, называются переменными с индексами в отличие от простых переменных, представляющих в программе элементарные данные. Индекс в обозначении компонент массивов может быть константой, переменной или выражением порядкового типа.

Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется *линейным*. Вообще количество индексов элементов массива определяет *размерность* массива. По этом признаку массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д.

Формат записи массива через раздел описания типов имеет вид:

*Type <имя типа>=**array** [тип индекса] **of** <тип компонента>;*

Var <идентификатор,…>: <имя типа>;

Формат записи массива через раздал описания переменных:

*Var <идентификатор,…>: **array** [тип индекса] **of** <тип компонента>;*

Массивы различают по количеству индексов: одномерные (1 индекс), двумерные и N-мерные (N индексов).

Пример: Type Vector = array [1..25] of real; {одномерный массив}

Matrix = array [1..20, 1..30] of byte; {двумерный массив}

Пример: числовая последовательность четных натуральных чисел 2, 4, 6, ..., *N* представляет собой линейный массив, элементы которого можно обозначить A[1]=2, A[2]=4, A[3]=6, ..., A[K]=2*(K+1), где *K* — номер элемента, а 2, 4, 6, ..., *N* — значения. Индекс (порядковый номер элемента) записывается в квадратных скобках после имени массива.

Например, A[7] — седьмой элемент массива A; D[6] — шестой элемент массива D.

Для размещения массива в памяти ЭВМ отводится поле памяти, размер которого определяется типом, длиной и количеством компонент массива.

тип идентификатор[количество строк];

Например,

int B[5]; char R[34];

— описывается массив *B*, состоящий из 5 элементов и символьный массив *R*, состоящий из 34 элементов. Для массива *B* будет выделено 5*6=30 байт памяти, для массива *R* — 1*34=34 байта памяти.

Рассмотрим работу с массивами на примере следующей задачи: Даны два массива целых чисел. Вычислить произведение максимального элемента первого массива на минимальный элемент второго массива. Удалить максимальный элемент из первого массива и добавить его во второй массив после минимального.

```
program example_9;
type massiv=array [1..40] of integer;
var a, b: massiv;
   i, n, k, p, j, min, Imin, max, Imax: integer;
begin
                                                {ввод массива с клавиатуры}
write('Введите размерность массива A n=');
readln(n);
writeln('Введите элементы массива A');
for i:=1 to n do begin
      write('a[',i,']=');
      readln(a[i]);
      end;
                                                {ввод массива случайным образом}
write('Введите размерность массива B k=');
readln(k);
randomize;                          {подключение генератора случайный чисел}
for i:=1 to k do
    b[i]:=random(16);       {заполнение массива случайными числами от 0 до 15}
                                        {Вывод массива на экран}
Writeln('Массив B');
for i:=1 to k do write(b[i],'   ');

writeln;                                {пустой оператор вывода}
                                        {Поиск максимального элемента}
max:=a[1];
Imax:=1;
```

```
for i:=2 to n do
   if a[i]>max then begin
                max:=a[i];                    {максимальный элемент}
                Imax:=i;                      {индекс максимального элемента}
                end;
                                              {Поиск минимального элемента}
min:=b[1];
Imin:=1;
for i:=2 to n do
   if b[i]<min then begin
                min:=b[i];                    {минимальный элемент}
                Imin:=i;                      {индекс минимального элемента}
                end;
                        {вычисление произведения и его вывод на экран}
p:=min*max;
writeln('Произведение max и min равно ',p);

                        {Удаление элемента из массива  с позиции Imax}
for i:=Imax to n-1 do
   a[i]:=a[i+1];
n:=n-1;                 {Уменьшение количества элементов массива на 1}

                        {Вставка элемента в массив после элемента равного min}
i:=1;
while i<k do
     begin
     if b[i]=min then   begin
       for j:=k+1 downto i+1 do b[j]:=b[j-1];        {смещение элементов на один вправо, начиная с последнего}
       k:=k+1;                        {Увеличение количества элементов на один}
       b[i+1]:=max;                 {Вставка элемента на позицию i+1}
       end;
     i:=i+1;                        {Увеличение счетчика итераций}
     end;

                                    {Вывод массивов A и B на экран в строку}
writeln('Массив A:    ');
for i:=1 to n do write(a[i],'   ');
```

```
writeln;
writeln(‘Массив B:    ‘);
for i:=1 to k do write(b[i],'   ');
end.
```

Заполнить массив можно следующим образом:

Первый способ с помощью оператора присваивания. Этот способ заполнения элементов массива особенно удобен, когда между элементами существует какая-либо зависимость, например, арифметическая или геометрическая прогрессии, или элементы связаны между собой реккурентным соотношением.

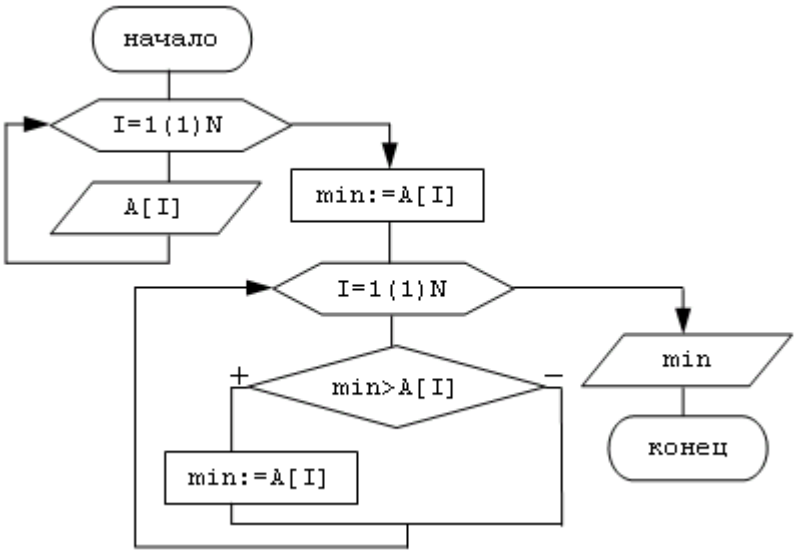
Второй способ с помощью генератора случайных чисел

```
randomize;                          {подключение генератора случайный чисел}
for i:=1 to k do
    b[i]:=random(X);        {заполнение массива случайными числами от 0 до X}
```

Третий способ с клавиатуры

```
for i:=1 to k do
    read(b[i]);
```

Задание: найти минимальный элемент в массиве.




```
program min;
const n=10;
var i: byte; a: array [1..n] of real; min: real;
begin
  for i:=1 to n do read (A[i]);
  min:=A[1];
  for i:=1 to n do
    if min>A[i] then min:=A[i];
  write (‘min= ’,min);
end.
```

Контрольные вопросы

1. Дать понятие массива.
2. Записать правило объявления одномерного массива.

3. Как можно заполнить массив?
4. Раскройте правило отыскание минимального (максимального) элемента массива.
5. Заполнение, вывод и обработка всех элементов массива осуществляется с использованием какой алгоритмической структуры?

Методы сортировки массива

Сортировкой или ***упорядочением*** массива называется расположение его элементов по возрастанию (или убыванию). Если не все элементы различны, то надо говорить о *неубывающем* (или *невозрастающем*) порядке.

Вообще говоря, это большая и сложная тема, в которой известно много различных алгоритмов. Критерии оценки эффективности этих алгоритмов могут включать следующие параметры:

- количество шагов алгоритма, необходимых для упорядочения;
- количество сравнений элементов;
- количество перестановок, выполняемых при сортировке.

Метод "пузырька"

По-видимому, самым простым методом сортировки является так называемый *метод **"пузырька"***. Чтобы уяснить его идею, представьте , что массив (таблица) расположен вертикально. Элементы с *большим* значением всплывают вверх наподобие больших пузырьков. При первом проходе вдоль массива, начиная проход "снизу", берется первый элемент и *поочередно* сравнивается с последующими. При этом:

- если встречается более "легкий" (с меньшим значением) элемент, то они меняются местами;
- при встрече с более "тяжелым" элементом, последний становится "*эталоном*" для сравнения, и все следующие сравниваются с ним .

В результате наибольший элемент оказывается в самом верху массива.

Во время второго прохода вдоль массива находится второй по величине элемент, который помещается под элементом, найденным при первом проходе, т.е на вторую сверху позицию, и т.д.

Заметим, что при втором и последующих проходах, нет необходимости рассматривать ранее "всплывшие" элементы, т.к. они заведомо больше оставшихся. Другими словами, во время **j**-го прохода не проверяются элементы, стоящие на позициях выше **j.**

Теперь можно привести текст программы упорядочения массива **M[1..N]**:

```
программа пузырьковая  сортировка.
program bs;
var i,j,x,n:integer;
   a:array[0..50] of integer;
begin
 writeln('введи  длину  массива');
 read(n);
 writeln('введи  массив');
 for i:=1 to n do read(a[i]);
 for i:=2 to n do  for j:=n downto i do if a[j-1]>a[j] then begin
  x:=a[j-1];
  a[j-1]:=a[j];
  a[j]:=x
 end;
 writeln('результат:');
 for i:=1 to n do write(a[i],' ')
```

end.

Заметим, что если массив M — глобальный, то процедура могла бы содержать только аргументы (а не результаты). Кроме того, учитывая специфику ее применения в данном алгоритме, можно свести число парметров к одному (*какому?*), а не двум.

Сортировка вставками

Второй метод называется *метод **вставок***., т.к. на **j**-ом этапе мы "вставляем" **j**-ый элемент **M[j]** в нужную позицию среди элементов **M[1]**, **M[2]**,. . ., **M[j-1]**, которые уже упорядочены. После этой вставки первые **j** элементов массива **M** будут упорядочены.
Сказанное можно записать следующим образом:

нц для j от 2 до N
переместить **M[j]** *на позицию* **i <= j** *такую, что*
M[j] < M[k] *для* **i<= k < j** *и*
либо **M[j] >= M[i-1]**, *либо* **i=1**
кц

Чтобы сделать процесс перемещения элемента **M[j]**, более простым, полезно воспользоваться *барьером*: ввести "фиктивный" элемент **M[0]**, чье значение будет заведомо меньше значения любого из "реальных"элементов массива (*как это можно сделать?*). Мы обозначим это значение через —**oo.**

Если барьер не использовать, то перед вставкой **M[j]**, в позицию **i-1** надо проверить, не будет ли **i=1**. Если нет, тогда сравнить **M[j]** (который в этот момент будет находиться в позиции **i**) с элементом **M[i-1]**.

Описанный алгоритм имеет следующий вид:

```
begin
  M[0] := -oo;
  for j:=2 to N do
   begin
     i := j;
       while M[i] < M[i-1] do
         begin
            swap(M[i],M[i-1]);
            i := i-1
         end
   end
end;
```

Сортировка посредством выбора

Идея сортировки с помощью выбора не сложнее двух предыдущих. На **j**-ом этапе выбирается элемент наименьший среди **M[j]**, **M[j+1]**,. . ., **M[N]** и меняется местами с элементом **M[j]**. В результате после **j**-го этапа все элементы **M[j]**, **M[j+1]**,. . ., **M[N]**будут упорядочены.

Сказанное можно описать следующим образом:

нц для j от 1 до N-1
выбрать среди **M[j]**,. . ., **M[N]** *наименьший элемент и*
поменять его местами с **M[j]**
кц

```
программа сортирвка с помощью прямого выбора.
program ss;
var i,j,r,x,n:integer;
   a:array[0..50] of integer;
begin
 writeln('введи  длину  массива');
 read(n);
 writeln('введи  массив');
```

```
 for i:=1 to n do read(a[i]);
 for i:=1 to n-1 do begin
  r:=i;
  x:=a[i];
  for j:=i+1 to n do if a[j]<x then begin
   r:=j;
   x:=a[r]
  end;
  a[r]:=a[i];
  a[i]:=x
 end;
 writeln('результат:');
 for i:=1 to n do write(a[i],' ')
end.
```

Метод Шелла

Недостаток метода сортировки "пузырьком" - работа с рядом стоящими элементами. В результате, элементы, которым надо "добраться" с одного конца массива до другого, делают это крайне долго. Метод Шелла аналогичен методу сортировки "пузырьком", однако работает с далеко отстоящими друг от друга элементами массива. На этот раз приведу сразу текст программы:

```
  const N=10; {Количество элементов массива}
  var a:       array[1..N] of integer; {массив}
     p:       boolean; {флаг наличия перестановки}
     l,d,i,j: integer; {служебные переменные}
  .....
d:=N div 2; {Расстояние между обрабатываемыми элементами массива,  на каждом этапе алгоритма
уменьшается в 2 раза вплоть до 0,  когда происходит останов алгоритма}
while d>0 do
begin
  for j:=1 to N-d do {Перебор всех пар элементов массива,  расстояние между которыми равно d}
  begin
    l:=j                  {запоминаем индекс текущего элемента}
    repeat
      p:=false;                    {пока элементы не переставлялись}
      if a[l]<a[l+d] then begin {если порядок нарушен, то}
        c:=a[l];a[l]:=a[l+d];a[l+d]:=c; {меняем местами элементы массива}
        l:=l-d;                    {перейдём к той паре, в которую входит меньший из переставленных элемен-
тов}
        p:=true;                          {запомним, что была перестановка}
      end;
    until (l<=1) and p;          {продолжаем, пока идут перестановки и не произошёл выход за пределы мас-
сива}
  end;
  d:=d div 2;                    {Уменьшим расстояние между сравниваемыми элементами в 2 раза}
end.
```

Контрольные вопросы.

1. Что значит отсортировать массив?
2. Перечислите и охарактеризуйте способы сортировки массива.

Обработка символьного массива

В языке программирования Паскаль переменная типа **array of char** может рассматриваться как строка постоянной длины. Переменные такого типа могут свободно использоваться в любых строковых

выражениях. При этом компилятор автоматически преобразует такой массив в строку, длина которой равна количеству элементов массива. Массивы символов представляют собой символьную строку определенной длины. Элементами символьного массива м.б. любой символ (типа CHR) как основного так и дополнительного набора символов кода ASCII. Основной набор ASCII символы с кодами 0..27, дополнительный (содержит символы национальных алфавитов и др. символы) символы с кодами 128..255. Символы можно записывать указывая их код #C где C принадлежит 0..255.

Массивы типа **char** можно сравнивать друг с другом и обращаться с ними почти так же, как с переменными типа **string**. Можно в операторе присваивания в левой части указывать имя такого массива, а в правой – строковую константу с длиной, равной количеству элементов в массиве. Кроме того, в Pascal массивам типа **array of char** разрешено присваивать строковые константы, длина которых меньше количества элементов массива; в оставшиеся при этом незаполненные элементы заносится символ #0. Однако нельзя переменной типа **array of char** присвоить значение строковой переменной или строкового выражения (кроме выражений над строковыми константами с результирующей длиной, равной размерности массива). Массивы типа **array of char** могут использоваться в процедуре **val** и функциях **concat, copy** и **length.**

Упакованные константы со строковым типом (символьные **массивы**) могут быть определены и как одиночные символы, и как строки. Например, такое определение:

Пример массива строкового типа

Const Digits : **Array** [0..9] Of Char = ('0', '1', '2', '3', '4', '5','6', '7', '8', '9');

может быть выражено более коротко:

Пример короткого массива строкового типа

Const Digits : **Array** [0..9] Of Char = '0123456789';

Нуль-основанные символьные массивы:

Нуль-основанный символьный массив - это такой массив, в котором индекс первого элемента равен нулю, а последнего - положительному ненулевому целому числу.

Пример нуль-основанного символьного массива

Array [0..X] Of Char;

Если вы включаете расширенный синтаксис (с помощью директивы компилятора {$X+}), то нуль-основанный символьный массив может быть инициализирован строкой, длина которой меньше, чем объявленная длина **массива.**

Пример нуль-основанного символьного массива с ограниченной длиной

Const FileName = **Array** [0..79] Of Char = 'TEST.PAS';

Если строка короче, чем длина **массива**, то оставшиеся символы устанавливаются равными NULL (0), и массив будет содержать строку с нулевым окончанием.

Пример.

1. Описать символьный массив длиной 80 символов.
2. Ввести предложение, состоящее из слов, разделенных пробелами.
3 Слово заканчивается заданной буквой.
4. Массив просмотреть до точки, если она есть, или до последнего введеного символа. Выдать слова с указанием их длины, слова, удовлетворяющие заданию, пометить примечанием, например, "Начинается на заданную букву".
5. В случае исключительных ситуаций выдать сообщение

```
var     a:array[1..80] of string; str,sl,by,by1:string; l:array[1..80] of integer;   {оддномерный массив}
          maxl,i:integer; space,tochka:char; p,q,st,sch:byte;
begin
maxl:=0; p:=0; space:=' '; tochka:='.'; sch:=0;
writeln ('Введите предложение:') ;
```

```
readln(str);
p:=pos(tochka,str);
if p>0 then str:=copy(str,1,p-1);
repeat
st:=length(str);
by1:=copy(str,st,st);
if by1=space then str:=copy(str,1,st-1)
else i:=1;
until i=1;
i:=0;
repeat
p:=pos(space,str);
  if p=1 then
       Delete(str,p,1)
 else begin
      i:=i+1;
      a[i]:=copy(str,1,p-1);
      Delete(str,1,p);
      l[i]:=length(a[i]);
      if maxl<l[i] then maxl:=l[i];
     end;
until  p=0;
if l[1]=0 then write('Введенная строка пуста')
else begin
repeat
   writeln('Будем проводить проверку слов? 1 - да, 0 - нет');
   readln(q);
until (q=1) or (q=0);
   if q=0
   then begin
      writeln('Проверка не была произведена');
      for p:=1 to i do begin
      writeln('слово №',p:2,':',a[p]:maxl,' (Длина слова =',l[p]:3,') Примечание: отсутсвует');
      end;
      end
   else begin
      writeln('Введите 1 символ для сравнения окончания слова');
      readln(by);
      st:=length(by);
      p:=pos(space,by);
      if (st=0) or (st>=2) or (p=1) then begin
                              writeln('Проверка не была произведена');
                              writeln('Строка пуста или было введено более 1 символа для проверки');
                              sl:='отсутствует'
                              end
      else begin
          for p:=1 to i do begin
          st:=length(a[p]);
          by1:=copy(a[p],st,st);
          if by=by1 then begin
                    sl:='Слово оканчивается на заданную букву';
                    inc(sch);
                    end
```

```
            else sl:='Слово не оканчивается на заданную букву';
            writeln ('слово №',p:2,':',a[p]:maxl,' (Длина слова =',l[p]:3,') Примечание: ',sl);
            end;
            writeln ('Количество слов удовлетворяющих условию: ',sch);
            end;
end;
end;
readln;
end.
```

Контрольные вопросы

1. Дать понятие символьного массива.
2. Какие процедуры и функции доступны для работы с элементами символьного массива?
3. Что такое Нуль-основанный символьный массив?
4. Можно ли переменной типа array of char присвоить значение строковой переменной или строкового выражения?

Обработка двумерного массива

Представьте себе таблицу, состоящую из нескольких строк. Каждая строка состоит из нескольких ячеек. Тогда для точного определения положения ячейки нам потребуется знать не одно число (как в случае таблицы линейной), а два: номер строки и номер столбца. Структура данных в языке Паскаль для хранения такой таблицы называется двумерным массивом. Описать такой массив можно двумя способами:

I. *Var A : Array [1..20] Of Array [1..30] Of Integer;*

II. *Var A : Array [1..20,1..30] Of Integer;*

В обоих случаях описан двумерный массив, соответствующий таблице, состоящей из 20 строк и 30 столбцов. Приведенные описания совершенно равноправны.

Отдельный элемент двумерного массива адресуется, естественно, двумя индексами. Например, ячейка, находящаяся в 5-й строке и 6-м столбце будет называться A[5,6].

Для иллюстрации способов работы с двумерными массивами решим задачу: "Задать и распечатать массив 10X10, состоящий из целых случайных чисел в интервале [1,100]. Найти сумму элементов, лежащих выше главной диагонали."

При отсчете, начиная с левого верхнего угла таблицы, главной будем считать диагональ из левого верхнего угла таблицы в правый нижний. При этом получается, что элементы, лежащие на главной диагонали будут иметь одинаковые индексы, а для элементов выше главной диагонали номер столбца будет всегда превышать номер строки. Договоримся также сначала указывать номер строки, а затем - номер столбца.

```
Var A : Array[1..10,1..10] Of Integer; I, K : Byte; S : Integer;
Begin
 S:=0;
 For I:=1 To 10 Do
   Begin
     For K:=1 To 10 Do
       Begin
         A[I,K]:=Trunc(Random*100)+1;
         Write(A[I,K]:6);
         If K>I Then S:=S+A[I,K]
       End;
     Writeln
   End;
Writeln('Сумма элементов выше гл. диагонали равна',S)
End.
```

Если модель данных в какой-либо задаче не может свестись к линейной или плоской таблице, то могут использоваться массивы произвольной размерности. N-мерный массив характеризуется N индексами. Формат описания такого типа данных:

Type

<Имя типа>=Array[<диапазон индекса1>,<диапазон индекса2>,...
<диапазон индекса N>] Of <тип компонент>;

Отдельный элемент именуется так:
<Имя массива>[<Индекс 1>,<Индекс 2>,...,<Индекс N>]

Заполнение элементов двумерного массива

1 Способ (заполнение с клавиатуры)

```
Var  M:array[1..7,1..5] of integer; I,J: byte;
Begin
  For I:=1 To 7 Do {цикл по строкам}
  For J:=1 To 5 Do {цикл по столбцам}
     ReadLn(M[I,J]);
End.
```

2 Способ (с использованием генератора случайных чисел)

```
Var  M: array[1..5,1..4] of integer; I,J: byte;
Begin
  For I:=1 To 5 Do
     Begin
        For J:=1 To 4 Do
           Begin
              M[I,J]:=Random(X);
              Write(M[I,J]:4);
            End;
         WriteLn;
       End;
End.
```

Вывод элементов двумерного массива в виде таблицы

```
  For I:=1 To n Do
     Begin
        For J:=1 To m Do
              M[I,J]:=Random(X);
              Write(M[I,J]:4);
        WriteLn;
      End;
```

Примеры решения задач

1. В течение недели измерялась температура три раза в день: утром, в обед и вечером. Показания записали в таблицу размерностью 7x3. Определить среднюю температуру за каждый день.

```
var
t:array[0..7,0..3] of integer; i,j:byte; s:integer; st:real; a:string[12];
begin
{ заполнение таблицы }
for i:=1 to 7 do begin
writeln(' введите показания за ',i,' день');
for j:=1 to 3 do begin
case j of
```

```
1: a:=' утром - ';
2: a:=' в обед - ';
3: a:=' вечером - ';
end;
write(a); readln(t[i,j]);
end;
end; writeln;
{ подсчет средней температуры за каждый день }
for i:=1 to 7 do begin
s:=0;
for j:=1 to 3 do
s:=s+t[i,j];
st:=s/3;
writeln('средняя температура за ',i,' день = ',st:4:1);
end;
end.
```

2. Поменять местами значения К-ого и Р-ого столбцов матрицы В, размерностью МхМ, заполненную случайным образом.

```
label m1;
const m=8;
var a:array[0..m,0..m] of integer; i,j,k,p:byte; pp:integer;
begin
{ заполнение двумерного массива случайным образом и вывод в виде прямоугольной матрицы }
writeln(' значения двумерного массива');
for i:=1 to m do begin
for j:=1 to m do begin
a[i,j]:=random(23); write(a[i,j]:3);
end; writeln;
end; writeln;
writeln(' введите номера столбцов, которые необходимо поменять');
writeln(' их значения не должны превышать ',m);
m1:readln(k,p);
if (k>m) or (p>m) then goto m1;
{ перестановка значений введенных столбцов }
for i:=1 to m do begin
pp:=a[i,k]; a[i,k]:=a[i,p]; a[i,p]:=pp;
end;
writeln(' измененный двумерный массив');
for i:=1 to m do begin
for j:=1 to m do begin
write(a[i,j]:3);
end; writeln;
end;
end.
```

Контрольные вопросы

1. Как объявить двумерный массив?
2. Сколько индексов у элемента двумерного массива? Какой за что отвечает?
3. Запишите как заполнить двумерный массив.
4. Запишите как вывести двумерный массив на экран в виде таблицы.

5. Запишите как обратиться к элементам главной диагонали двумерного массива.

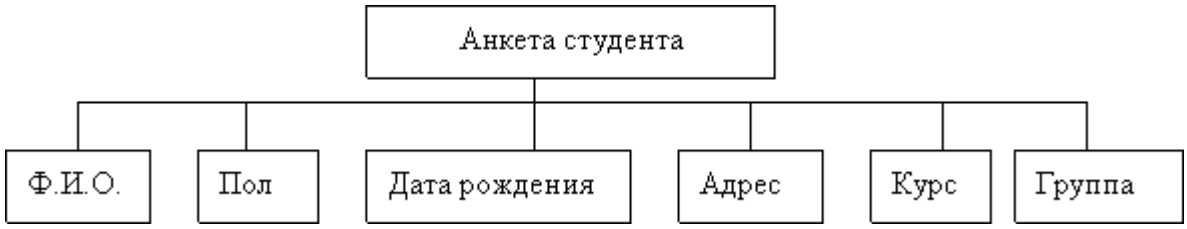
ЗАПИСИ в Паскаль

Многообразие информации нельзя свести только к какому-то одному типу данных. Например, указывая положение точки в пространстве, мы можем воспользоваться одним и тем же типом для указания ее координат, но, описывая человека, мы должны указать его имя, рост, цвет глаз и волос, то есть в одном описании объединим разнородную информацию. Точно так же, описывая автомобиль, мы укажем не только его марку, но и год выпуска, модификацию, да и цвет кузова может нас заинтересовать. Составляя автоматизированный каталог книгохранилища, мы для каждой книги должны указать ее название, имя автора, область знания, количество страниц, год издания, а также, возможно, признак нахождения на руках или в хранилище.

Данные такого рода, описывающие существенные стороны того или иного объекта путем включения в описание нескольких, часто разнотипных, элементов, называют *записью* (*record*). В языке Паскаль запись определяется путем указания служебного слова record и перечисления входящих в запись элементов с указанием типов этих элементов.

Запись Паскаля – структурированный комбинированный тип данных, состоящий из фиксированного числа компонент (полей) разного типа.

Например, анкетные данные о студенте вуза могут быть представлены в виде информационной структуры



Такая структура называется двухуровневым деревом. В Паскале эта информация может храниться в одной переменной типа record (запись). Задать тип можно следующим образом:

```
type < имя _ типа >=record
  <имя_поля1>: тип;
  <имя_поля2>: тип;
  ……..
  <имя_поля K >: тип
end ;
```

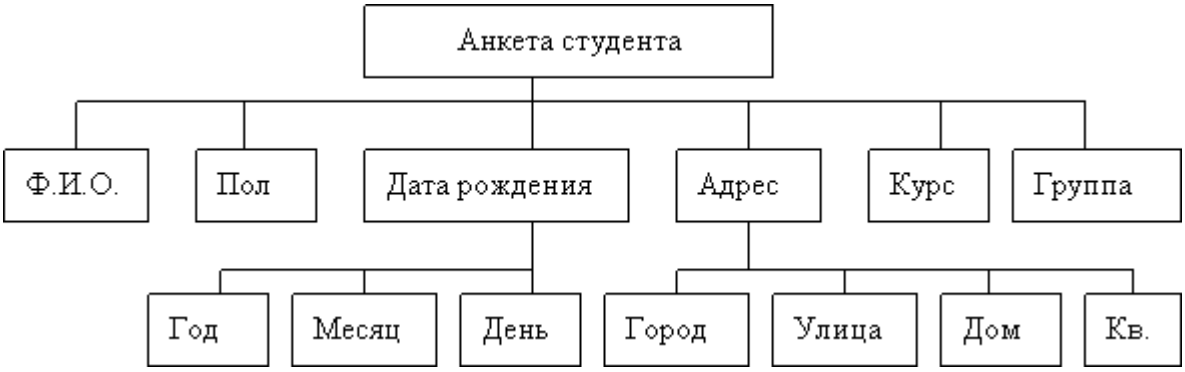
где record – служебное слово, а <имя_типа> и <имя_поля> - правильные идентификаторы языка Паскаль.

Описание анкеты студента в Паскале будет выглядеть так:

Пример фрагмента программы описания записи Паскаля

```
Type anketa=record
  fio: string[45];
  pol: char;
  dat_r: string[8];
  adres: string[50];
  curs: 1..5;
  grupp: string[3];
end;
```

Поля записи Паскаля могут иметь любой тип, в частности сами могут быть записями. Такая возможность используется в том случае, когда требуется представить многоуровневое дерево (более 2 уровней). Например, те же сведения о студентах можно отобразить трехуровневым деревом.



Такая организация данных позволит, например, делать выборки по году рождения или по городу, где живут студенты. В этом случае описание соответствующей записи в Паскале будет выглядеть так:

Пример фрагмента программы описания записи Паскаля

```
Type anketa1=record
  fio: string[45];
  pol: char;
  dat_r: record;
  god: integer;
  mes: string[10];
  den: 1..31;
end;
adres: record
  gorod: string[25];
  ulica: string [20];
  dom, kv: integer;
end;
curs: 1..5;
  grupp: string[3];
end;
```

Поля

После того, как определен тип записи Паскаля, можно определять переменную этого типа. Переменная определяется путем задания ее идентификатора и указания типа.

```
var
student: anketa;
student 1: anketa 1;
```

Элементы записи называются *полями*, а обращение к ним производится через использование их имен – *идентификаторов* полей. Практически, поля записи обрабатываются точно так же, как и любые другие переменные. Но в отличие от обычной переменной имена полей должны предваряться ссылкой на идентификатор записи Паскаля и отделяться от него точкой. Такая запись называется уточняющий идентификатор:

<имя_записи>.<имя_поля>

Например, чтобы обратиться к полю curs переменной student , необходимо указать следующее составное имя:

```
student.curs :=3;
```

Для того чтобы обратиться к полю god в записи student 1, необходимо записать уточняющий идентификатор, состоящий из трех имен:

```
student1.dat_r.god:=1982;
```

Использование полей записи Паскаля в выражениях и условиях идентично использованию обычных переменных.

Операции над записями Паскаля

Единственная операция, которую можно произвести над однотипными записями Паскаля – это присваивание.

Все другие операции производятся над отдельными полями записи.

Пример решения задачи с использованием записей Паскаля

Рассмотрим для начала простейший пример заполнения записи Паскаля и вывода ее на экран.

Пусть нам необходимо заполнить сведения о студенте (Ф.И.О., дата рождения, адрес, курс и группа), а затем вывести эти сведения на экран.

Пример программы с записью Паскаля

```
program primer1;
type anketa=record
  fio: string[45];
  dat_r: string[8];
  adres: string[50];
  curs: 1..5;
  grupp: string[3]
end;
var student: anketa;
begin
  writeln (‘введите сведения о студенте’);
  {обратите внимание, ввод каждого поля осуществляется отдельно}
  writeln (‘введите фамилию, имя и отчество’);
  readln (student.fio);
  writeln (‘введите дату рождения’);
  readln (student.dat_r);
  writeln (‘введите адрес’);
  readln(student.adres);
  writeln (‘введите курс’);
  readln(student.curs);
  writeln (‘введите группу’);
  readln (student.grupp);
  writeln (‘ввод закончен’);
  writeln ;
  {обратите внимание, что вывод записи осуществляется по полям}
  writeln (‘фамилия студента: ’, student . fio );
  writeln(‘ дата рождения : ’, student.dat_r);
  writeln(‘ адрес : ’, student.adres);
  writeln(‘ курс : ’, student.curs);
  writeln(‘ группа : ’, student.grupp);
end.
```

А теперь слегка усложним задачу. Пусть нам необходимо иметь сведения о многих студентах, например, нашего факультета. Следовательно, необходимо организовать массив записей Паскаля. А затем из общего списка вывести фамилии студентов 2-го курса.

Пример программы с записью Паскаля

```
program primer 2 ;
type anketa=record
  fio: string[45];
  dat_r: string[8];
  adres: string[50];
  curs: 1..5;
  grupp: string[3]
end;
var student: array [1..100] of anketa;
```

```
  I: integer;
begin
  {последовательно вводим каждую запись}
  for I:=1 to 100 do
  begin
    writeln (‘введите сведения о’, I , ‘-м студенте’);
    writeln (‘введите фамилию, имя и отчество’);
    readln (student[I].fio);
    writeln (‘введите дату рождения’);
    readln (student[I].dat_r);
    writeln (‘введите адрес’);
    readln(student[I].adres);
    writeln (‘введите курс’);
    readln(student[I].curs);
    writeln (‘введите группу’);
    readln (student[I].grupp);
  end;
  writeln (‘ввод закончен’);
  writeln ;
  {просматриваем массив записей и выбираем только студентов 2-го курса }
  for I:=1 to 100 do
    if student[I].curs=2 then
      writeln(‘ фамилия студента : ’, student[I].fio);
end.
```

Оператор присоединения или как избавиться от префикса

Префикс – обязательная предшествующая часть составного идентификатора для имен полей в структуре типа запись Паскаля. Очень часто у программиста возникает желание не указывать префикс в имени полей, например, когда идет постоянное использование одних и тех же записей. В языке Паскаль предусмотрена такая возможность, реализуемая при помощи оператора присоединения, который в общем виде выглядит так:

with <имя_записи> do <действие с полем записи>;

Следует обратить внимание на то, что после служебного слова *do* может стоять только один оператор, но он может быть составным (любая последовательность операторов, заключенная в операторные скобки begin end).

Например, фрагмент из предыдущей программы с использованием оператора присоединения будет выглядеть так:

Пример фрагмента программы с записью и префиксом Паскаля

```
for I:=1 to 100 do
  with student[I] do
  begin
    writeln (‘введите сведения о’, I , ‘-м студенте’);
    writeln (‘введите фамилию, имя и отчество’);
    readln (fio);
    writeln (‘введите дату рождения’);
    readln (dat_r);
    writeln (‘введите адрес’);
    readln(adres);
    writeln (‘введите курс’);
    readln(curs);
    writeln (‘введите группу’);
    readln (grupp);
```

end;

Контрольные вопросы

1. Чем отличается тип "запись" от других структурированных типов?
2. Могут ли поля записи быть одного и того же типа?
3. Как обратиться к отдельному полю записи?
4. Что такое "оператор присоединения"? В каких целях он используется?
5. Как заполнить массив записей?

Массив записей

Задание 1

Известны данные о сотрудниках фирмы: фамилия и отношение к во-инской службе (военнообязанный или нет). Напечатать фамилии всех военно-обязанных сотрудников.

Решение:

```
type tab = record
  fam:string;
  v:boolean;
end;
const n=5;
var   tabl :array [1..n] of tab;   i    :integer;   Otvet:integer;
begin  {заполнение массива с данными о сотрудниках}
  for i:=1 to n do
  begin
    writeln('Введите фамилию N',i,':');
    readln(tabl[i].fam);
    write('военнообязанный(если да введите = 1/если нет введите = 0)');
    readln(Otvet);
    if Otvet=0 then tabl[i].v:=false { не военнообязанный }
    else tabl[i].v:= true; { военнообязанный }
  end;
  {проверка признака и выдача подходящих записей}
  WriteLn;
  WriteLn('Список военнообязанных: ');
  for i:=1 to n do
  begin
   if tabl[i].v=true
       then writeln (tabl[i].Fam)
  end;
end.
```

Задание 2

В записной книжке указаны фамилии и номера телефонов 7 человек. Составить программу, которая определяет:

а) есть ли в записной книжке телефон запрашиваемого человека, если есть, то сообщить телефон;

б) есть ли в записной книжке информация о человеке с заданным носером телефона и если есть, то сообщить его фамилию.

Понятие подпрограммы. Процедуры и функции

В языке Паскаль, как и в большинстве языков программирования, предусмотрены средства, позволяющие оформлять вспомогательный алгоритм как подпрограмму. Это бывает необходимо тогда, когда какой-либо подалгоритм неоднократно повторяется в программе или имеется возможность использовать некоторые фрагменты уже разработанных ранее алгоритмов. Кроме того, подпрограммы применяются для разбиения крупных программ на отдельные смысловые части в соответствии с модульным принципом в программировании.

Для использования подалгоритма в качестве подпрограммы ему необходимо присвоить имя и описать алгоритм по правилам языка Паскаль. В дальнейшем, при необходимости вызвать его в программе, делают вызов подпрограммы упоминанием в нужном месте имени соответствующего подалгоритма со списком входных и выходных данных. Такое упоминание приводит к выполнению входящих в подпрограмму операторов, работающих с указанными данными. После выполнения подпрограммы работа продолжается с той команды, которая непосредственно следует за вызовом подпрограммы.

В языке Паскаль имеется два вида подпрограмм - **процедуры** и **функции.**

Процедуры и функции помещаются в раздел описаний программы. Для обмена информацией между процедурами и функциями и другими блоками программы существует механизм **входных** и **выходных параметров**. Входными параметрами называют величины, передающиеся из вызывающего блока в подпрограмму (исходные данные для подпрограммы), а выходными - передающиеся из подрограммы в вызывающий блок (результаты работы подпрограммы).

Одна и та же подпрограмма может вызываться неоднократно, выполняя одни и те же действия с разными наборами входных данных. Параметры, использующиеся при записи текста подпрограммы в разделе описаний, называют **формальными**, а те, что используются при ее вызове - **фактическими.**

Описание и вызов процедур и функций

Структура описания процедур и функций до некоторой степени похожа на структуру Паскаль-программы: у них также имеются заголовок, раздел описаний и исполняемая часть. Раздел описаний содержит те же подразделы, что и раздел описаний программы: описания констант, типов, меток, процедур, функций, перменных. Исполняемая часть содержит собственно операторы процедур.

Формат описания процедуры имеет вид:

procedure *имя процедуры* (*формальные параметры*);
 раздел описаний процедуры
begin

 исполняемая часть процедуры

end;

Формат описания функции:

function *имя функции* (*формальные параметры*):*тип результата*;
 раздел описаний функции
begin

 исполняемая часть функции

end;

Формальные параметры в заголовке процедур и функций записываются в виде:

```
var имя праметра: имя типа
```

и отделяются друг от друга точкой с запятой. Ключевое слово var может отсутствовать (об этом далее). Если параметры однотипны, то их имена можно перечислять через запятую, указывая общее для них имя типа. При описании параметров можно использовать только стандартные имена типов, либо имена типов, определенные с помощью команды type.Список формальных параметров может отсутствовать.

Вызов процедуры производится оператором, имеющим следующий формат:

```
имя  процедуры(список фактических параметров);
```

Список фактических параметров - это их перечисление через запятую. При вызове фактические параметры как бы подставляются вместо формальных, стоящих на тех же местах в заголовке. Таким образом происходит передача входных параметров, затем выполняются операторы исполняемой части процедуры, после чего происходит возврат в вызывающий блок. Передача выходных параметров происходит непосредственно во время работы исполняемой части.

Вызов функции в Турбо Паскаль может производиться аналогичным способом, кроме того имеется возможность осуществить вызов внутри какого-либо выражения. В частности имя функции может стоять в правой части оператора присваивания, в разделе условий оператора if и т.д.

Для передачи в вызывающий блок выходного значения функции в исполняемой части функции перед возвратом в вызывающий блок необходимо поместить следующую команду:

```
имя функции := результат;
```

При вызове процедур и функций необходимо соблюдать следущие правила:

- количество фактических параметров должно совпадать с количеством формальных;
- соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Заметим, что имена формальных и фактических параметров могут совпадать. Это не приводит к проблемам, так как соответствующие им переменные все равно будут различны из-за того, что хранятся в разных областях памяти. Кроме того, все формальные параметры являются временными переменными - они создаются в момент вызова подпрограммы и уничтожаются в момент выхода из нее.

ПРИМЕРЫ.

Рассмотрим использование процедуры на примере программы поиска максимума из двух целых чисел.

```
var x,y,m,n: integer;

procedure MaxNumber(a,b: integer; var max: integer);
begin
  if a>b then max:=a else max:=b;
end;

begin
  write('Введите x,y ');
  readln(x,y);
  MaxNumber(x,y,m);
  MaxNumber(2,x+y,n);
  writeln('m=',m,'n=',n);
end.
```

Аналогичную задачу, но уже с использованием функций, можно решить так:

```
var x,y,m,n: integer;

function MaxNumber(a,b: integer): integer;
  var max: integer;
begin
  if a>b then max:=a else max:=b;
  MaxNumber := max;
end;

begin
  write('Введите x,y ');
  readln(x,y);
  m := MaxNumber(x,y);
  n := MaxNumber(2,x+y);
  writeln('m=',m,'n=',n);
end.
```

Еще один классический пример. Задача: "Расположить в порядке неубывания три целых числа".

```
Program Pr;
Var
S1,S2,S3 :Integer;
Procedure Swap(Var A,B: Integer);{Процедура Swap с параметрами-переменными}
Var C : Integer; {C - независимая локальная переменная}
Begin
C:=A; A:=B; B:=C {Меняем местами содержимое A и B}
End;
Begin
Writeln('Введите три числа');
Readln(S1,S2,S3);
If S1>S2 Then Swap(S1,S2);
If S2>S3 Then Swap(S2,S3);
If S1>S2 Then Swap(S1,S2);
Writeln('Числа в порядке неубывания:V',S1,S2,S3)
End.
```

Рекурсия

Рекурсия — это такой способ организации вспомогательного алгоритма (подпрограммы), при котором эта подпрограмма (процедура или функция) в ходе выполнения ее операторов обращается *сама к себе*. Вообще, **рекурсивным** называется любой объект, который частично определяется через себя.

Например, приведенное ниже определение двоичного кода является рекурсивным:

```
<двоичный код> ::= <двоичная цифра> | <двоичный код><двоичная цифра>
<двоичная цифра> ::= 0 | 1
```

Здесь для описания понятия были использованы, так называемые, металингвистический формулы Бэкуса-Наура (язык БНФ); знак "::=" обозначает "по определению есть", знак "|" — "или".

Вообще, в рекурсивном определении должно присуствовать ограничение, *граничное условие*, при выходе на которое дальнейшая инициация рекурсивных обращений прекращается.

Приведём другие примеры рекурсивных определений.

Пример 1. Классический пример, без которого не обходятся ни в одном рассказе о рекурсии, — определение факториала. С одной стороны, факториал определяется так: $n!=1*2*3*...*n$. С другой стороны, $n! = \begin{cases} 1, & \text{если } n \le 1, \\ (n-1)! \times n, & \text{если } n > 1. \end{cases}$ Граничным условием в данном случае является $n<=1$.

Пример 2. Определим функцию *K(n)*, которая возвращает количество цифр в заданном натуральном числе *n*: $K(n) = \begin{cases} 1, & \text{если } n < 10, \\ K(n/10)+1, & \text{если } n >= 10. \end{cases}$

Задание. По аналогии определите функцию *S(n)*, вычисляющую сумму цифр заданного натурального числа.

Пример 3. Функция *C(m, n)*, где 0 <= *m* <= *n*, для вычисления биномиального коэффициента C_n^m по следующей формуле $C_n^0 = C_n^n = 1;\ C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при $0 < m < n$, является рекурсивной.

Обращение к рекурсивной подпрограмме ничем не отличается от вызова любой другой подпрограммы. При этом при каждом новом рекурсивном обращении в памяти создаётся новая копия подпрограммы со всеми локальными переменными. Такие копии будут порождаться до выхода на граничное условие. Очевидно, в случае отсутствия граничного условия, неограниченный рост числа таких копий приведёт к аварийному завершению программы за счёт переполнения стека.

Порождение все новых копий рекурсивной подпрограммы до выхода на граничное условие называется *рекурсивным спуском*. Максимальное количество копий рекурсивной подпрограммы, которое одновременно может находиться в памяти компьютера, называется *глубиной рекурсии*. Завершение работы рекурсивных подпрограмм, вплоть до самой первой, инициировавшей рекурсивные вызовы, называется *рекурсивным подъёмом*.

Выполнение действий в рекурсивной подпрограмме может быть организовано одним из вариантов:

```
Begin                    Begin                    Begin
  P;                       операторы;               операторы;
  операторы;               P                        P;
End;                     End;                       операторы
                                                    End;
```

рекурсивный подъём | рекурсивный спуск | и рекурсивный спуск, и рекурсивный подъём

Здесь **P** — рекурсивная подпрограмма. Как видно из рисунка, действия могут выполняться либо на одном из этапов рекурсивного обращения, либо на обоих сразу. Способ организации действий диктуется логикой разрабатываемого алгоритма.

Пример 4. *Вычислить сумму элементов линейного массива.*

При решении задачи используем следующее соображение: сумма равна нулю, если количество элементов равно нулю, и сумме всех предыдущих элементов плюс последний, если количество элементов не равно нулю.

```
Type LinMas = Array[1..100] Of Integer;
Var A : LinMas;
  I, N : Byte;
{Рекурсивная функция}
Function Summa(N : Byte; A: LinMas) : Integer;
Begin
  If N = 0 Then Summa := 0 Else Summa := A[N] + Summa(N - 1, A)
End;
{Основная программа}
Begin
  Write('Количество элементов массива? '); ReadLn(N); Randomize;
  For I := 1 To N Do
  Begin
    A[I] := -10 + Random(21); Write(A[I] : 4)
  End;
  WriteLn; WriteLn('Сумма: ', Summa(N, A))
End.
```

Задание. Определите, является ли заданное натуральное число палиндромом.

Использование рекурсии увеличивает время исполнения программы и зачастую требует значительного объёма памяти для хранения копий подпрограммы на рекурсивном спуске. Поэтому на практике разумно заменять рекурсивные алгоритмы на итеративные.

Контрольные вопросы

1. Какое определение называется рекурсивным? Приведите собственные примеры рекурсивных определений.
2. Какой вспомогательный алгоритм (подпрограмма) называются рекурсивными? Приведите собственные примеры содержательных задач, где для решения может быть использован рекурсивный вспомогательный алгоритм.
3. Что такое граничное условие и каково его назначение в рекурсивной подпрограмме?
4. Что такое рекурсивный спуск?
5. Что такое рекурсивный подъём?
6. Что такое глубина рекурсии? Чему равна глубина рекурсии в приведённых выше примерах?
7. На каком этапе выполнения рекурсивной подпрограммы могут выполняться её операторы?
8. Почему приведённый ниже алгоритм посимвольного формирования строки завершится аварийно?

```
Function Stroka : String;
Var C : Char;
Begin
   Write('Введите очередной символ: '); ReadLn(C);
   Stroka:=Stroka+C
End;
```

9. На каком этапе выполняются действия в этом алгоритме?

Программирование подпрограмм

При работе с подпрограммами важными являются понятия *формальных и фактических параметров. Формальные параметры* — это идентификаторы входных данных для подпрограммы. Если формальные параметры получают конкретные значения, то они называются *фактическими*. Формальные параметры могут получить конкретные значения только в той программе, где производится обращение к данному модулю-подпрограмме. **Тип** и **порядок** записи фактических параметров должны быть такими же, как и формальных параметров. В противном случае результат работы программы будет непредсказуемым. Из этого следует, что фактические параметры используются при обращении к подпрограмме из основной, а формальные параметры — только в самом модуле.

Пример 1. Используем алгоритм нахождения наибольшего общего делителя двух натуральных чисел в качестве вспомогательного при решении задачи: составить программу вычитания дробей (*a, b, c, d* — натуральные числа). Результат представить в виде обыкновенной несократимой дроби.

```
Program Sub;
Var A, B, C, D, G, E, F : Integer;
Procedure Nod(M, N : Integer; Var K : Integer);
Begin
   While M <> N Do
   If M > N Then M := M - N Else N := N - M;
   K := M
End;
Begin
   Write('Введите числители и знаменатели дробей:');
   ReadLn(A, B, C, D);
   E := A * D - B * C;
   F := B * D;
   If E = 0 Then WriteLn(E)
         Else
            Begin
              Nod(Abs(E), F, G);
              E := E Div G;
              F := F Div G;
```

```
            WriteLn('Ответ: ', E, '/', F)
          End
End.
```

По способу передачи фактических значений в подпрограмму в Pascal выделяют *параметры-переменные*, *параметры-значения*, *параметры-константы* и *массивы открытого типа*, *строки открытого типа*, *параметры-процедуры*, *параметры-функции*.

Функция (в отличие от процедуры) всегда возвращает единственное значение.

Пример 2. Дано натуральное число *n*. Переставить местами первую и последнюю цифры этого числа.

```
Program Integ;
  Var N : Integer;
  Begin
     Write('Введите натуральное число: ');
     ReadLn(N);
     If Impossible(N)
     Then WriteLn('Невозможно переставить цифры, возникнет переполнение')
     Else Begin
             Change(N);
             WriteLn('Ответ: ', N)
          End;
  End.
```

Виды параметров подпрограмм

Список параметров, то есть величин, передаваемых в подпрограмму и обратно, содержится в ее заголовке. Для каждого параметра обычно задается его имя, тип и способ передачи. Либо тип, либо способ передачи могут не указываться.

Важно запомнить, что в заголовке подпрограммы нельзя вводить описание нового типа, там должны использоваться либо имена стандартных типов, либо имена типов, описанных программистом ранее в разделе type.

В Паскале четыре вида параметров:

значения;

переменные;

константы;

нетипизированные параметры.

Параметры-значения

Параметр-значение описывается в заголовке подпрограммы следующим образом:

имя : тип;

Например, передача в процедуру P величины целого типа записывается так:

procedure P(x : integer);

Имя параметра может быть произвольным. Параметр x можно представить себе как локальную переменную, которая получает свое значение из главной программы при вызове подпрограммы. В подпрограмму передается копия значения аргумента.

Механизм передачи следующий: из ячейки памяти, в которой хранится переменная, передаваемая в подпрограмму, берется ее значение и копируется в область сегмента стека, называемую областью параметров. Подпрограмма работает с этой копией, следовательно, доступа к ячейке, где хранится сама переменная, не имеет. По завершении работы подпрограммы стек освобождается. Такой способ называется передачей по значению.

При вызове подпрограммы на месте параметра, передаваемого по значению, может находиться выражение. Тип выражения должен быть совместим по присваиванию с типом параметра.

Например, если в вызывающей программе описаны переменные

var x : integer;

c : byte;

y : longint;

то следующие вызовы подпрограммы P, заголовок которой описан выше, будут синтаксически правильными:

P(x); P(c); P(y); P(200); P(x div 4 + 1);

Недостатками передачи по значению являются затраты времени на копирование параметра, затраты памяти в стеке и опасность его переполнения, когда речь идет о параметрах, занимающих много места - например, массивах большого размера. Поэтому более правильно использовать для передачи в подпрограмму исходных данных параметры-константы, о которых речь пойдет чуть дальше.

Параметры-переменные

Признаком параметра-переменной является ключевое слово var перед описанием параметра:

var имя : тип;

Например, передача в процедуру P параметра-переменной целого типа записывается так:

procedure P(var x : integer);

При вызове подпрограммы в область параметров копируется не значение переменной, а ее адрес, и подпрограмма через него имеет доступ к ячейке, в которой хранится переменная. Этот способ передачи параметров называется передачей по адресу. Подпрограмма работает непосредственно с переменной из вызывающей программы и, следовательно, может ее изменить.

ВНИМАНИЕ При вызове подпрограммы на месте параметра-переменной может находиться только ссылка на переменную точно того же типа.

Параметры-константы

Параметр-константу можно узнать по ключевому слову const перед описанием параметра:

const имя : тип;

Это ключевое слово говорит о том, что в пределах подпрограммы данный параметр изменить невозможно. При вызове подпрограммы на месте параметра может быть записано выражение, тип которого совместим по присваиванию с типом параметра. Фактически параметры-константы передаются по адресу, но доступ к ним обеспечивается только для чтения.

Например, передача в процедуру P параметра-константы целого типа записывается так:

procedure P(const x : integer);

Подведем итоги. Если данные передаются в подпрограмму по значению, их можно изменять, но эти изменения затронут только копию в области параметров и не отразятся на значении переменной в вызывающей программе. Если данные передаются как параметры-константы, изменять их в подпрограмме нельзя. Следовательно, эти два способа передачи должны использоваться для передачи в подпрограмму исходных данных.

Параметры составных типов (массивы, записи, строки) предпочтительнее передавать как константы, потому что при этом не расходуется время на копирование и место в стеке.

Результаты работы процедуры следует передавать через параметры-переменные, результат функции - через ее имя.

Нетипизированные параметры

Как можно догадаться из названия, при описании нетипизированных параметров не указывается тип. Передаются они всегда по адресу - либо как константы, либо как переменные, например:

procedure P(const a, b; var y);

Есть одно важное ограничение: передать-то их можно, а вот делать с ними в подпрограмме что-либо до тех пор, пока они не приведены к какому-то определенному типу, все равно нельзя!

Контрольные вопросы

1.Какое количество вспомогательных алгоритмов может присутствовать в основном алгоритме?

2.Какие параметры называют формальными? фактическими?

3.Какое соответствие должно соблюдаться между формальными и фактическими параметрами?

4.Может ли фактических параметров процедуры (функции) быть больше, чем формальных? А меньше?

5.Существуют ли подпрограммы без параметров?

6.Какие виды формальных параметров существуют? Чем они отличаются друг от друга?

7.В чём состоит отличие процедур и функций?

8.В каких случаях целесообразно использовать функции?

9.Почему, если в функции используются параметры-переменные, необходимо преобразовать её в процедуру?

10. Какого типа может быть значение функции?

Стандартные встроенные модули

В Паскале имеется ряд стандартных модулей, в которых описано большое количество встроенных констант, типов, переменных и подпрограмм. Каждый модуль содержит связанные между собой ресурсы.

Модуль `System`

Модуль содержит базовые средства языка, которые поддерживают ввод-вывод, работу со строками, операции с плавающей точкой и динамическое распределение памяти. Этот модуль автоматически используется во всех программах, и его не требуется указывать в операторе `uses`. Он содержит все стандартные и встроенные процедуры, функции, константы и переменные Паскаля.

Модуль `Crt`

Модуль предназначен для организации эффективной работы с экраном, клавиатурой и встроенным динамиком. При подключении модуля `Crt` выводимая информация посылается в базовую систему ввода-вывода (BIOS) или непосредственно в видеопамять.

В текстовом режиме экран представляется как совокупность строк и столбцов. Каждый символ располагается на так называемом *знакоместе* на пересечении строки и столбца. Символы хранятся в специальной части оперативной памяти, называемой *видеопамятью*. Ее содержимое отображается на экране.

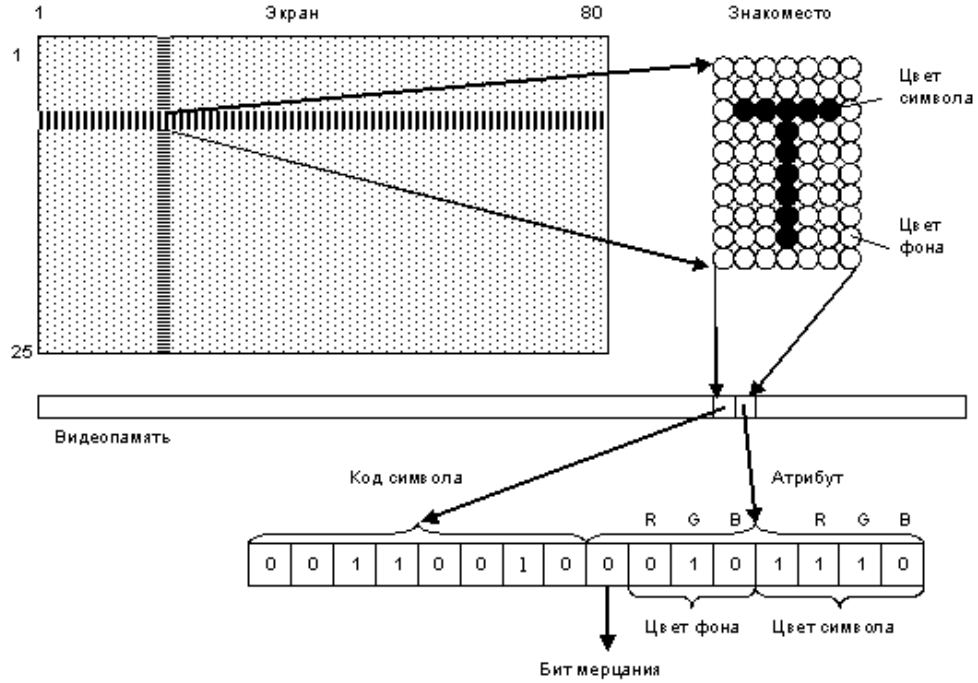


Рис. 1

Под каждый символ отводится два байта: один байт занимает ASCII-код символа, другой байт хранит атрибуты символа: его цвет, цвет фона и признак мерцания (рис. 1). Можно получить восемь различных цветов фона и 16 цветов символов.

Модуль `Crt` позволяет:

- выполнять вывод в заданное место экрана заданным цветом символа и фона;
- открывать на экране окна прямоугольной формы и выполнять вывод в пределах этих окон;
- очищать экран, окно, строку и ее часть;
- обрабатывать ввод с клавиатуры;
- управлять встроенным динамиком.

Работа с экраном

Текущие цвета символа и фона задаются с помощью процедур `TextColor` и `TextBackGround` и действуют на следующие за ними процедуры вывода. Вывод выполняется в текущую позицию курсора. Для ее изменения служит процедура `GotoXY`.

Окно определяется с помощью процедуры `Window`. Оно задается координатами левого верхнего и правого нижнего угла.

Очистка текущего окна выполняется с помощью процедуры `ClrScr`, которая заполняет его пробелами с текущим цветом фона и устанавливает курсор в левый верхний угол.

Пример 1

Программа "угадай число".

```
program luck;
uses crt;
const max = 10;
var
   i, k, n : integer;
begin
   clrscr;                        { очистить экран }
   randomize; i := random(max);       { загадать число }
   window(20, 5, 60, 20);             { определить окно }
   TextBackGround(Blue);              { цвет фона - синий }
   clrscr;                        { залить окно фоном }
   TextColor(LightGray);              { цвет символов - серый }
   k := -1;                       { счетчик попыток }
   GotoXY(12, 5); writeln(' Введите число : ');
   repeat                         { цикл ввода ответа }
     GotoXY(20, 9);                   { установить курсор }
     readln(n);                     { ввести число }
     inc(k);
   until i = n;
   window(20, 22, 60, 24);            { определить окно результата }
   TextAttr := 2 shl 4 + 14;          { желтые символы за зеленом фоне }
   clrscr;                        { залить окно фоном }
   GotoXY(6, 2);                      { установить курсор }
   writeln(' Коэффициент невезучести : ', k / max :5:1);
   readkey;                       { ждать нажатия любой клавиши }
   TextAttr := 15;                    { белые символы на черном фоне }
   clrscr;                        { очистить после себя экран }
end.
```

Работа с клавиатурой

Модуль `Crt` позволяет работать с управляющими клавишами и комбинациями клавиш. Нажатие каждой клавиши преобразуется либо в ее ASCII-код, либо в так называемый расширенный код (scan-код) и записывается в буфер клавиатуры, из которого затем и выбирается процедурами ввода. Под каждый код отводится два байта. Если нажатие клавиш соответствует символу из набора ASCII, в первый байт заносится код символа. Если нажата, к примеру, клавиша курсора, функциональная клавиша или комбинация

клавиш с CTRL или ALT, то первый байт равен нулю, а во втором находится расширенный код, соответствующий этой комбинации.

Для работы с клавиатурой модуль Crt содержит функции ReadKey и KeyPressed.

Модули Dos и WinDos

Модули Dos и WinDos содержат подпрограммы, реализующие возможности операционной системы MS-DOS - например, переименование, поиск и удаление файлов, получение и установку системного времени, выполнение программных прерываний и так далее. Эти подпрограммы в стандартном Паскале не определены. Для поддержки подпрограмм в модулях определены константы и типы данных.

Модуль Dos использует строки Паскаля, а WinDos - строки с завершающим нулем.

Модуль Graph

Модуль обеспечивает работу с экраном в графическом режиме.

Экран в графическом режиме представляется в виде совокупности точек - *пикселов*. Цвет каждого пиксела можно задавать отдельно. Начало координат находится в левом верхнем углу экрана и имеет координаты (0, 0). Количество точек по горизонтали и вертикали (*разрешение экрана*) и количество доступных цветов зависят от графического режима. Графический режим устанавливается с помощью служебной программы - *графического драйвера*.

В состав оболочки входит несколько драйверов, каждый из может работать в нескольких режимах. Режим устанавливается при инициализации графики либо автоматически, либо программистом.

Модуль Graph обеспечивает:

- вывод линий и геометрических фигур заданным цветом и стилем;
- закрашивание областей заданным цветом и шаблоном;
- вывод текста различным шрифтом, заданного размера и направления;
- определение окон и отсечение по их границе;
- использование графических спрайтов и работу с графическими страницами.

В отличие от текстового режима, в графическом курсор невидим.

Перед выводом изображения необходимо определить его *стиль*, то есть задать цвет фона, цвет линий и контуров, тип линий, шаблон заполнения, вид и размер шрифта, и так далее.

Эти параметры устанавливаются с помощью соответствующих процедур. Возможные значения параметров определены в модуле Graph в виде многочисленных констант.

Структура графической программы

Программа, использующая графический режим, должна содержать следующие действия:

- подключение модуля Graph;
- перевод экрана в графический режим;
- установку параметров изображения;
- вывод изображения;
- возврат в текстовый режим.

Модуль Strings

Модуль предназначен для работы со строками, заканчивающимися нуль-символом, то есть символом с кодом 0 (их часто называют ASCIIZ-строки). Модуль содержит функции копирования, сравнения, слияния строк, преобразования их в строки типа string, поиска подстрок и символов.

В модуле System определен тип pChar, представляющий собой указатель на символ (^Char). Этот тип можно использовать для работы со строками, заканчивающимися #0. Эти строки располагаются в динамической памяти, и программист должен сам заниматься ее распределением.

Кроме того, для хранения ASCIIZ-строк используются массивы символов с нулевой базой, например:

```
var  str : array[0 .. 4000] of char;
     p   : pChar;
```

Контрольные вопросы

1. Раскрыть функции стандартных модулей в Паскаль.
2. Как формируется изображение на экране?
3. Раскрыть структуру графической программы.

Объектно-ориентированное программирование pascal

Объектно-ориентированное программирование - это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. ООП использует в качестве базовых элементов объекты, а не алгоритмы

Объекты и классы

Базовыми блоками объектно-ориентированной программы являются объекты и классы. Содержательно объект можно представить как что-то ощущаемое или воображаемое и имеющее хорошо определенное поведение. Объект – осязаемая сущность, которая четко проявляет свое поведение.

Объект — это часть окружающей нас реальности, т. е. он существует во времени и в пространстве.

Класс — это множество объектов, имеющих общую структуру и общее поведение. Класс — описание (абстракция), которое показывает, как построить существующую во времени и пространстве переменную этого класса, называемую объектом. Смысл предложений «описание переменных класса» и «описание объектов класса» один и тот же.

Объект имеет состояние, поведение и паспорт (средство для его однозначной идентификации); структура и поведение объектов описаны в классах, переменными которых они являются .

Определим теперь понятия состояния, поведения и идентификации объекта.

Состояние объекта объединяет все его поля данных (статический компонент, т.е. неизменный) и текущие значения каждого из этих полей (динамический компонент, т.е. обычно изменяющийся).

Поведение выражает динамику изменения состояний объекта и его реакцию на поступающие сообщения, т.е. как объект изменяет свои состояния и взаимодействует с другими объектами.

Идентификация (распознавание) объекта — это свойство, которое позволяет отличить объект от других объектов того же или других классов. Осуществляется идентификация посредством уникального имени (паспорта), которым наделяется объект в программе, впрочем как и любая другая переменная.

Объектно-ориентированный подход представляет программы в виде набора объектов, взаимодействующих между собой. Взаимодействие объектов осуществляется через сообщения. Предположим, что нашим объектом является окружность. Тогда сообщение, посланное этому объекту, может быть следующим: «нарисуй себя». Когда мы говорим, что объекту передается сообщение, то на самом деле мы вызываем некоторую функцию этого объекта (компонент-функцию).

Пакетирование (инкапсуляция) предполагает соединение в одном объекте данных и функций, которые манипулируют этими данными. Доступ к некоторым данным внутри пакета может быть либо запрещен, либо ограничен.

Объект характеризуется как совокупностью всех своих свойств и их текущих значений, так и совокупностью допустимых для этого объекта действий. Указанное объединение в едином объекте как «материальных» составных частей, так и действий, манипулирующих этими частями называется инкапсуляцией.

В рамках ООП данные называются полями объекта, а алгоритмы – объектными методами.

Инкапсуляция позволяет в максимальной степени изолировать объект от внешнего окружения. Она существенно повышает надежность разрабатываемых программ, т.к. локализованные в объекте алгоритмы обмениваются с программой сравнительно небольшими объемами данных, причем количество и тип этих данных обычно тщательно контролируется. В результате замена или модификация алгоритмов и данных, инкапсулированных в объект, как правило, не влечет за собой плохо прослеживаемых последствий для программы в целом. Другим немаловажным следствием инкапсуляции является легкость обмена объектами, переноса их из одной программы в другую.

Наследование

И структурная, и объектно-ориентированная методологии преследуют цель построения иерархического дерева взаимосвязей между объектами (подзадачами). Но если структурная иерархия строится по простому принципу разделения целого

на составные части, то при создании объектно-ориентированной иерархии принимается другой взгляд на тот же исходный объект. В объектно-ориентированной иерархии непременно отражается наследование свойств родительских (вышележащих) типов объектов дочерним (нижележащим) типам объектов.

Наследование – это такое отношение между объектами, когда один объект повторяет структуру и поведение другого.

Принцип наследования действует в жизни повсеместно и повседневно. Млекопитающие и птицы наследуют признаки живых организмов, в отличие от растений, орел и ворон наследуют общее свойство для птиц – умение летать. С другой стороны, львы, тигры, леопарды наследуют «структуру» и поведение, характерное для представителей отряда кошачьих и т.д.

Типы верхних уровней объектно-ориентированной иерархии, как правило, не имеют конкретных экземпляров объектов. Не существует, например, конкретного живого организма, который бы сам по себе назывался «млекопитающее» или «птица». Такие типы называют абстрактными. Конкретные экземпляры объектов имеют, как правило, типы самых нижних уровней.

Наследование позволяет использовать библиотеки классов и развивать их (совершенствовать и модифицировать библиотечные классы) в конкретной программе. Наследование позволяет создавать новые объекты, изменяя или дополняя свойства прежних. Объект-наследник получает все поля и методы предка, но может добавить собственные поля, добавить собственные методы или перекрыть своими методами одноименные унаследованные методы.

Принцип наследования решает проблему модификации свойств объекта и придает ООП в целом исключительную гибкость. При работе с объектами программист обычно подбирает объект, наиболее близкий по своим свойствам для решения конкретной задачи, и создает одного или нескольких потомков от него, которые «умеют» делать то, что не реализовано в родителе.

Последовательное проведение в жизнь принципа «наследуй и изменяй» хорошо согласуется с поэтапным подходом к разработке крупных программных проектов и во многом стимулирует такой подход.

Когда вы строите новый класс, наследуя его из существующего класса, можно:

добавить в новый класс новые компоненты-данные;

добавить в новый класс новые компоненты-функции;

заменить в новом классе наследуемые из старого класса компоненты-функции.

Полиморфизм

позволяет использовать одни и те же функции для решения разных задач. Полиморфизм выражается в том, что под одним именем скрываются различные действия, содержание которых зависит от типа объекта.

Полиморфизм – это свойство родственных объектов (т.е. объектов, имеющих одного общего родителя) решать схожие по смыслу проблемы разными способами. Например, действие «бежать» свойственно большинству животных. Однако каждое из них (лев, слон, крокодил, черепаха) выполняет это действие различным образом.

В рамках ООП поведенческие свойства объекта определяются набором входящих в него методов, программист только указывает, какому объекту какое из присущих ему действий требуется выполнить. Изменяя алгоритм того или иного метода в потомках объекта, программист может придавать этим потомкам отсутствующие у родителя специфические свойства. Для изменения метода необходимо *перекрыть* его в потомке, т.е. объявить в потомке одноименный метод и реализовать в нем нужные действия. В результате в объекте-родителе и объекте-потомке будут действовать два одноименных метода, имеющих разную алгоритмическую основу и, следовательно, придающие объектам разные свойства. Это и называется полиморфизмом объектов.

Описание объектного типа

Класс или объект – это структура данных, которая содержит поля и методы. Как всякая структура данных она начинается зарезервированным словом и закрывается оператором end. Формальный синтаксис не сложен: описание объектного типа получается, если в описании записи заменить слово record на слово object или class и добавить объявление функций и процедур над полями.

```
Type <имя типа объекта>= object
  <поле>;
  <поле>;
  ….
  <метод>;
  <метод>;
end ;
```

В ObjectPascal существует специальное зарезервированное слово class для описания объектов

```
Type <имя типа объекта>= class
  <поле>;
   ….
  <метод>;
  <метод>;
end ;
```

ObjectPascal поддерживает обе модели описания объектов.

Компонент объекта – либо поле, либо метод. Поле содержит имя и тип данных. Метод – это процедура или функция, объявленная внутри декларации объектного типа, в том числе и особые процедуры, создающие и уничтожающие объекты (конструкторы и деструкторы). Объявление метода внутри описания объектного типа состоит только из заголовка. Это разновидность предварительного описания подпрограммы. Тело метода приводится вслед за объявлением объектного типа.

Пример . Вводится объектный тип «предок», который имеет поле данных Name (имя) и может выполнять два действия:

провозглашать: «Я – предок!»;

сообщать свое имя.

```
Type tPredoc = object Name : string ; {поле данных объекта}
  Procedure Declaration ; {объявление методов объекта}
  Procedure MyName ;
End ;
```

Тексты подпрограмм, реализующих методы объекта, должны приводиться в разделе описания процедур и функций. Заголовки при описании реализации метода повторяют заголовки, приведенные в описании типа, но дополняются именем объекта, которое отделяется от имени процедуры точкой. В нашем примере:

```
Procedure tPredoc.Declaration ; {реализация метода объекта}
begin
  writeln (‘Я – предок!’);
end ;
Procedure tPredoc.MyName ; {реализация метода объекта}
begin
  writeln(‘Я –’, Name);
end;
```

Внутри описания методов на поля и методы данного типа ссылаются просто по имени. Так метод MyName использует поле Name без явного указания его принадлежности объекту так, если бы выполнялся неявный оператор with <переменная_типа_объект> do .

Под объектами понимают и переменные объектного типа – их называют экземплярами. Как всякая переменная, экземпляр имеет имя и тип: их надо объявить.

```
{объявление объектного типа и описание его методов}
var v 1: tPredoc ; {объявление экземпляра объекта}
begin
  v1. Name := ‘Петров Николай Иванович’;
  v1.Declaration;
  v1.MyName
```

end.

Использование поля данных объекта v1 не отличается по своему синтаксису от использования полей записей. Вызов методов экземпляра объекта означает, что указанный метод вызывается с данными объекта v 1. В результате на экран будут выведены строчки

Я – предок!

Я – Петров Николай Иванович

Аналогично записям, к полям переменных объектного типа разрешается обращаться как с помощью уточненных идентификаторов, так и с помощью оператора with .

Например, в тексте программы вместо операторов

v1.Name := ‘Петров Николай Иванович’;

v1.Declaration ;

v1.MyName

возможно использование оператора with такого вида

```
with v1 do
begin
  Name:= ‘Петров Николай Иванович’;
  Declaration ;
  MyName
End ;
```

Более того, применение оператора with с объектными типами, также как и для записей не только возможно, но и рекомендуется.

Иерархия типов (наследование)

Типы можно выстроить в иерархию. Объект может наследовать компоненты из другого объектного типа. Наследующий объект — это потомок. Объект, которому наследуют — предок. Подчеркнем, что наследование относится только к типам, но не к экземплярам объектов.

Если введен объектный тип (предок, родительский), а его надо дополнить полями или методами, то вводится новый тип, объявляется наследником (потомком, дочерним типом) первого и описываются только новые поля и методы. Потомок содержит все поля типа предка. Заметим, что поля и методы предка доступны потомку без специальных указаний. Если в описании потомка повторяются имена полей или методов предка, то новые описания переопределяют поля и методы предка.

ООП всегда начинается с базового класса. Это шаблон для базового объекта. Следующим этапом является определение нового класса, который называется производным и является расширением базового.

Производный класс может включать дополнительные методы, которые не существуют в базовом классе. Он может переопределять (redefined) методы (или даже удалять их целиком).

В производном классе не должны переопределяться все методы базового класса. Каждый новый объект наследует свойства базового класса, необходимо лишь определить те методы, которые являются новыми или были изменены. Все другие методы базового класса считаются частью и производного. Это удобно, т.к. когда метод изменяется в базовом классе, он автоматически изменяется во всех производных классах.

Процесс наследования может быть продолжен. Класс, который произведен от базового, может сам стать базовым для других производных классов. Таким образом, ОО программы создают иерархию классов.

Наиболее часто структура иерархии классов описывается в виде дерева. Вершины дерева соответствуют классам, а корню соответствует класс, который описывает что-то общее (самое общее) для всех других классов.

Наследование дочерними типами информационных полей и методов их родительских типов выполняется по следующим правилам.

Правило 1. Информационные поля и методы родительского типа наследуются всеми его дочерними типами независимо от числа промежуточных уровней иерархии.

Правило 2. Доступ к полям и методам родительских типов в рамках описания любых дочерних типов выполняется так, как будто-бы они описаны в самом дочернем типе.

Правило 3. Ни в одном дочернем типе не могут быть использованы идентификаторы полей родительских типов.

Правило 4. Дочерний тип может доопределить произвольное число собственных методов и информационных полей.

Правило 5. Любое изменение текста в родительском методе автоматически оказывает влияние на все методы порожденных дочерних типов, которые его вызывают.

Правило 6. В противоположность информационным полям идентификаторы методов в дочерних типах могут совпадать с именами методов в родительских типах. В этом случае говорят, что дочерний метод перекрывает (подавляет) одноименный родительский метод. В рамках дочернего типа, при указании имени такого метода, будет вызываться именно дочерний метод, а не родительский.

Продолжим рассмотрение нашего примера. В дополнение к введенному нами типу предка tPredoc можно ввести типы потомков:

```
type tSon= object(tPredoc) {Тип, наследующий tPredoc }
  procedure Declaration; {перекрытие методов предка}
  procedure My Name(Predoc : tPredoc);
end ;
type tGrandSon=object(tSon) {Тип, наследующий tSon}
  procedure Declaration ; {перекрытие методов предка}
end ;
```

Имя типа предка приводится в скобках после слова object. Мы породили наследственную иерархию из трех типов: tSon («сын») наследник типу tPredoc , а тип tGrandSon (“внук”) - типу tSon. Тип tSon переопределяет методы Declaration и My N a m e, но наследует поле Name . Тип tGrandSon переопределяет только метод Declaration и наследует от общего предка поле Name , а от своего непосредственного предка (типа tSon) переопределенный метод Declaration .

Разберемся, что именно мы хотим изменить в родительских методах. Дело в том, что «сын» должен провозглашать несколько иначе, чем его предок, а именно сообщить ‘Я – отец!’

```
procedure tSon.Declaration ; {реализация методов объектов — потомков}
 begin
  writeln (" Я — отец !");
 end;
```

А называя свое имя, “сын” должен сообщить следующие сведения:

Я <фамилия имя отчество >

Я – сын <фамилия имя отчество своего предка>

```
 procedure tSon .My Name ( predoc : tPredoc );
begin
  inherited My Name ; {вызов метода непосредственного предка}
  writeln ("Я — сын ", predoc.Name, ‘ a ’ );
end;
```

В примере потомок tSon из метода My Name вызывает одноименный метод непосредственного предка типа tPredoc . Такой вызов обеспечивается директивой inherited , после которой указан вызываемый метод непосредственного предка. Если возникает необходимость вызвать метод отдаленного предка в каком-нибудь дочернем типе на любом уровне иерархии, то это можно сделать с помощью уточненного идентификатора, т.е. указать явно имя типа родительского объекта и через точку – имя его метода:

TPredoc.MyName;

Разберемся с «внуком». Метод, в котором «внук» называет свое имя, в точности такой же, как и у его непосредственного предка (типа tSon), поэтому нет необходимости этот метод переопределять, этот метод лучше автоматически наследовать и пользоваться им как своим собственным. А вот в методе Declaration нужно провозгласить ‘Я – внук!’, поэтому метод придется переопределить.

```
 procedure tGrandSon.Declaration;
begin
```

```
writeln (" Я — внук !");
 end;
```

Рассмотрим пример программы, в которой определим экземпляр типа tPredoc , назовем его «дед», экземпляр типа tSon – «отец», и экземпляр типа tGrandSon – «внук». Потребуем от них, чтобы они представились.

Пример программы с испльзованием ООП

```
{заголовок программы}
…….
{раздел описания типов, в том числе и объектных типов tPredoc , tSon , tGrandSon }
{Обратите внимание! Экземпляры объектных типов можно описать как типизированные константы, что мы для примера и сделали ниже}
const ded : tPredoc = ( Name : "Петров Николай Иванович");
  otec : tSon = ( Name : "Петров Сергей Николаевич");
  vnuk : tGrandSon = ( Name : "Петров Олег Сергеевич");
{раздел описания процедур и функций, где обязательно должны быть написаны все объявленные в объектных типах методы}
begin
  ded.Declaration ; {вызов методов общего предка}
  ded.My Name;
  writeln;
  otec.Declaration;
  otec.MyName(ded); { вызов методов объекта otec типа tSon}
  writeln;
  vnuk.Declaration; { вызов методов объекта vnuk типа tGrandSon}
  vnuk.MyName ( otec );
end .
```

Программа выведет на экран:

Пример вывода на экран результата

```
Я —предок!
Я —Петров Николай Иванович
Я —отец!
Я —Петров Сергей Николаевич
Я —сын Петров Николай Ивановича
Я —внук!
Я —Петров Олег Сергеевич
Я —сын Петров Сергей Николаевича
```

Обратите внимание, что в заголовке процедуры tSon . MyName в качестве параметра приведен тип данных tPredoc , а при использовании этой процедуры ей передаются переменные как типа tPredoc , так и типа tSon . Это возможно, так как предок совместим по типу со своими потомками. Обратное несправедливо. Если мы заменим в заголовке процедуры tSon . MyName при описании параметров тип tPredoc на tSon , компилятор укажет на несовместимость типов при использовании переменной ded в строке otec . MyName (ded).

Полиморфизм и виртуальные методы

Полиморфизм – это свойство родственных объектов (т.е. объектов, имеющих одного родителя) решать схожие по смыслу проблемы разными способами.

Два или более класса, которые являются производными одного и того же базового класса, называются полиморфными. Это означает, что они могут иметь общие характеристики, но так же обладать собственными свойствами.

В рамках ООП поведенческие свойства объекта определяются набором входящих в него методов. Изменяя алгоритм того или иного метода в потомках объекта, программист может придавать этим потомкам отсутствующие у родителя специфические свойства. Для изменения метода необходимо перекрыть его в потомке, т.е. объявить в потомке одноименный метод и реализовать в нем нужные действия. В результате чего в объекте-родителе и объекте-потомке

будут действовать два одноименных метода, имеющих разную алгоритмическую основу и, следовательно, придающие объектам разные свойства. Это и называется полиморфизмом объектов.

В рассмотренном выше примере во всех трех объектных типах tPredoc , tSon и tGrandSon действуют одноименные методы Declaration и MyName . Но в объектном типе tSon метод MyName выполняется несколько иначе, чем у его предка. А все три одноименных метода Declaration для каждого объекта выполняются по-своему.

Методы объектов бывают абстрактными, статическими, виртуальными и динамическими.

Абстрактными называются методы, которые определены в классе, но не содержат никаких действий, никогда не вызываются и обязательно должны быть переопределены в потомках класса. Абстрактными могут быть только виртуальные и динамические методы. В Object Pascal такие методы объявляются с помощью одноименной директивы abstract.

Статические методы

включаются в код программы при компиляции. Это означает, что до использования программы определено, какая процедура будет вызвана в данной точке. Компилятор определяет, какого типа объект используется при данном вызове, и подставляет метод этого объекта. Объекты разных типов могут иметь одноименные статические методы. В этом случае нужный метод определяется по типу экземпляра объекта.

Это удобно, так как одинаковые по смыслу методы разных типов объектов можно и назвать одинаково, а это упрощает понимание и задачи и программы. Статическое перекрытие – первый шаг полиморфизма. Одинаковые имена – вопрос удобства программирования, а не принцип.

Виртуальные методы

В отличие от статических, подключаются к основному коду на этапе выполнения программы. Виртуальные методы дают возможность определить тип и конкретизировать экземпляр объекта в процессе исполнения, а затем вызвать методы этого объекта. Этот принципиально новый механизм, называемый поздним связыванием, обеспечивает полиморфизм, т.е. разный способ поведения для разных, но однородных (в смысле наследования) объектов. Описание виртуального метода отличается от описания обычного метода добавлением после заголовка метода служебного слова virtual .

procedure Method (список параметров); virtual;

Использование виртуальных методов в иерархии типов объектов имеет определенные ограничения:

если метод объявлен как виртуальный, то в типе потомка его нельзя перекрыть статическим методом;

объекты, имеющие виртуальные методы, инициализируются специальными процедурами, которые, в сущности, также являются виртуальными и носят название constructor ;

списки переменных, типы функций в заголовках перекрывающих друг друга виртуальных процедур и функций должны совпадать полностью;

Динамические методы.

Динамические методы вызываются медленнее, но позволяют более экономно расходовать память. Каждому динамическому методу системой присваивается уникальный индекс. В таблице динамических методов класса хранятся индексы и адреса только тех динамических методов, которые описаны в данном классе. При вызове динамического метода происходит поиск в этой таблице; в случае неудачи просматриваются таблицы DMT всех классов-предков в порядке иерархии и, наконец, класс TObject, где имеется стандартный обработчик вызова динамических методов. Экономия памяти налицо.

Для перекрытия и виртуальных, и динамических методов служит директива override, с помощью которой (и только с ней!) можно переопределять оба этих типа методов. Приведем пример:

Попытка перекрытия с директивой не override, а virtual или dynamic приведет на самом деле к созданию нового одноименного метода.

Конструктор

Обычно на конструктор возлагается работа по инициализации экземпляра объекта: присвоение полям исходных значений, первоначальный вывод на экран и т.п. Помимо действий, заложенных в него программистом, конструктор выполняет подготовку механизма позднего связывания виртуальных методов. Это означает, что еще до вызова любого виртуального метода должен быть выполнен какой-нибудь конструктор.

Конструктор – это специальный метод, который инициализирует объект, содержащий виртуальные методы. Заголовок конструктора выглядит так:

constructor Method (список параметров);

Зарезервированное слово constructor заменяет слова procedure и virtual .

Основное и особенное назначение конструктора – установление связей с таблицей виртуальных методов (VMT) – структурой, содержащей ссылки на виртуальные методы. Таким образом, конструктор инициализирует объект установкой связи между объектом и VMT с адресами кодов виртуальных методов. При инициализации и происходит позднее связывание.

У каждого объекта своя таблица виртуальных методов VMT . Именно это и позволяет одноименному методу вызывать различные процедуры.

Следует сказать и о деструкторе. Его роль противоположна: выполнить действия, завершающие работу с объектом, закрыть все файлы, очистить динамическую память, очистить экран и т.д.

Заголовок деструктора выглядит таким образом:

destructor Done ;

Основное назначение деструкторов – уничтожение VMT данного объекта. Часто деструктор не выполняет других действий и представляет собой пустую процедуру.

destructor Done ;

begin end

Элементы класса можно объявить как public (общедоступные), private (закрытые) или protected (защищенные).

К общедоступным свойствам и методам можно получать доступ из любого контекста.

К закрытому свойству и методу можно получить доступ только из того класса, в котором они объявлены. Даже подклассы данного класса не имеют доступа к таким свойствам и методам.

К защищенным свойствам и методам можно получить доступ либо из содержащего их класса, либо из его подкласса. Никакому внешнему коду такой доступ не предоставляется.

Чем это может быть нам полезно? Ключевые слова, определяющие область видимости, позволяют показывать только те аспекты класса, которые требуются клиенту. Это позволяет создать ясный и понятный интерфейс для объекта.

Секция private

В некоторых случаях у вас могут иметься части описаний объектов, которые экспортировать нежелательно. Например, вы можете предусмотреть объекты для других программистов, которые они могут использовать, но не могут непосредственно манипулировать с данными объекта. Чтобы облегчить это, Borland Pascal позволяет задавать внутри объектов приватные (закрытые) поля и методы.

Приватные поля и методы доступны только внутри того модуля, в котором описан объект. Приватные поля и методы описываются непосредственно после обычных полей и методов, вслед за зарезервированным словом private. Таким образом, полный синтаксис описания объекта будет следующим:

```
type
  NewObject = object(родитель)
  поля;                { общедоступные }
  методы;                { общедоступные }
private
  поля;                { приватные }
  методы;                { приватные }
end;
```

Контрольные вопросы

1. Составить таблицу терминов ООП

№ п/п	Термин	Описание
1.		
2.		
3.		
4.		
5.		

6.		
7.		

Раскрыть механизм описания объекта.

2. Каким правилам подчиняется наследование типов.
3. Какие бывают методы объектов? Опишите их.
4. Как можно объявит элементы класса? Опишите эти способы.

РАЗДЕЛ 3 ЛАБОРАТОРНЫЕ ЗАНЯТИЯ

Умение решать задачи – такое же практическое искусство, как умение плавать или бегать на лыжах. Ему можно научиться только путем подражания или упражнения.

Д. Пойа

Лабораторная работа

по теме «Выполнение работы с кодом программы»

Цель работы: отработать приемы работы с системой программирования и кодом программы.

ЗАДАНИЕ

Часть 1. С использованием компьютера выполнить задания

Задание 1. Дописать недостающие части программы и проверить её для следующих входящих данных K=15, 123

Дано действительное число К. Определить его целую и дробную части. Вывести на экран целую, дробную части и остаток от деления целой части на 2.

```
begin
writeln('vvedite k');
readln(k);
d:=frac(k);
writeln('t=',t,'d=',d:4:4);
readln;
end.
```

Задание 2. Изменить предложенную последовательность операторов чтобы решить задачу: Дано число N. Вычислить среднее арифметическое его и числа составленного из его цифр в обратном порядке.

```
var N,i,j,k:integer;  y:real;
begin
writeln('rezultat ',y:2:3);
readln(N);
j:=N div 100   ;
y:=(N+k*100+i*10+j)/2  ;
i:=(N div 10) mod 10   ;
end.
write('vvedite chislo N ');
k:=N mod 10   ;
readln;
```

Задание 3. Определить результат выполнения программы. Не набирая её на компьютере. X:=5, Y:=24.6, Z:=11 (2 и 0.05)

```
var x,z,g,i:integer;  y:real;
begin
write('vvedite chisla x,y,z ');
readln(x,y,z);
g:=(z div 10) mod 10   ;
i:=g;
g:=x;
x:=i;
```

```
y:=y/(x+z);
writeln('rezultat',frac(y));
writeln(round(y));
readln;
end.
```

Задание 4. Решить задачу

Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1<R2).

Часть 2. Самостоятельно выполнить задания и записать результат в тетрадь

Задание 1 оценивается в 2 балла, решение задачи - в 3 балла

Задание 1. Дописать недостающие части программы. Изменить предложенную последовательность операторов чтобы решить задачу: Скорость первого велосипедиста V1 м/ч, второго — V2 м/ч. Велосипедисты начинают движение одновременно из одной точки. И движутся в одном направлении. Определить расстояние между ними через T часов.**Опрелить результат выполнения программы. V1=780, V2=645.6, T=3**

```
var  v1,v2,t,s:real;

readln;

readln(T);

write('vvedite T v chasax ');

S:=…(V1*t-V2*t)   ;

write('vvedite V1, V2 ');

writeln('rezultat ',S:2:3);

end.
```

Задание 2. Решить задачу

1. Даны координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь

2. Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга.

3. Даны $\vec{a}(x1, y1)$ и $\vec{b}(x2, y2)$. Найти скалярное произведение векторов и косинус угла между ними.

Лабораторная работа
по теме «Применение простейших операторов при решении задач линейной структуры»

Цель работы: применение простейших операторов при решении задач .

ЗАДАНИЕ

Часть 1. С использованием компьютера выполнить задания

Задание 1. Определить результат выполнения программы. Не набирая её на компьютере при a=3, b=4, c=5. Какую задачу решает предложенная программа?

```
var
   a, b, c, p, ha, hb, hc, t : real;
 begin
   writeln ('Введите стороны тр-ка a,b,c');
   readln (a,b,c);
   p:=(a+b+c)/2.;
   t:=2*sqrt (p*(p-a)*(p-b)*(p-c));
   ha:=t/a; hb:=t/b; hc:=t/c;
   writeln ('Искомые величины: ha=',ha:8:2,' hb=', hb:8:2,'  hc=',hc:8:2);
end.
```

Задание 2. Решить задачу

Для участия в конкурсе по профессиональной дисциплине необходимо из группы делегировать 2 юношей 1 девушку. Напишите программу, рассчитывающую сколькими способами можно составить требуемую тройку ребят из твоей группы. В программе рассчитать вероятность того, что ты попадешь в эту тройку.

$$n = C^1_{\hat{e}\hat{i}\ddot{e}\ .\ddot{a}\mathring{a}\hat{a}\acute{o}\o\mathring{a}\hat{e}} \cdot \tilde{N}^2_{\hat{e}\hat{i}\ddot{e}\ .\dot{p}\acute{i}\o\mathring{a}\acute{e}} \quad , \ p = \frac{C^1_{\hat{e}\hat{i}\ddot{e}\ .\ddot{a}\mathring{a}\hat{a}\acute{o}\o\mathring{a}\hat{e}} \cdot \tilde{N}^1_{\hat{e}\hat{i}\ddot{e}\ .\dot{p}\ddot{i}\o\mathring{a}\acute{e}}}{C^3_{cnel/\ uheggs}}$$

На зачет по дисциплине вынесены 35 теоретических вопросов и 15 задач. Все они пронумерованы. Студент называет сам три числа, чтобы выбрать задания для зачета. Подсчитать вероятность того, что задание будет содержать 2 задачи и 1 вопрос

Часть 2. Самостоятельно выполнить задания и записать результат в тетрадь

Решение задачи оценивается в 3 балла

Задача 1 <<Чет или нечет?>>

Условия игры: Компьютер генерирует случайное целое число, а человек пытается угадать, четное оно или нечетное. Результат сравнения выводится на экран

Задача 2 <<Отгадай число>>

Условия игры: Компьютер генерирует случайное целое число, не превосходящее 100. Играющий пытается угадать это число, делая несколько попыток, после каждой из которых компьютер сообщает, какое из чисел больше

Задача 3 <<Игральные кости>>

Условия игры: компьютер и человек <<бросают>> некоторое число n игральных костей. Победителем считается набравший большую сумму очков

Лабораторная работа

по теме «Решение задач с использованием структурного программирования»

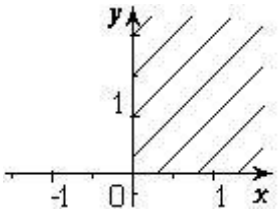
Цель работы: применение структурного программирования при решении задач.

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

1. Клоун предложил каждому из публики задумать число. Потом он сказал: «Прибавьте к задуманному числу 5. Теперь из результата вычтите 2. А теперь к результату прибавьте 7». Потом клоун спросил у желающих, какое число у каждого из них получилось. Услышав ответ, он немедленно объявлял каждому, какое число тот задумывал. Составьте программу, которая повторяла бы фокус клоуна.
2. Если хотя бы одно из двух введенных пользователем чисел четно, вывести слово ‘да’, в противном случае – вывести ‘нет’.
3. Определить, удовлетворяет ли условию $|x|\geq 8$ число, введенное с клавиатуры.
4. Определить, состоит ли двузначное число, введенное клавиатуры, из одинаковых цифр.
5. Умножить трехзначное число, введенное с клавиатуры, на 2, если оно содержит в своей записи хотя бы одну 1.

1*. Точка А задана координатами X,Y. Разработать схему алгоритма, который устанавливает значение флага F=1, если точка принадлежит заштрихованной области (см. рисунок) и значение флага F=0 в противном случае. Вывести значение F. Протестировать алгоритм для точек (1.5,2), (0,0), (-1.5, 1), (1,-1.2), (-2,-1).



2*. Прием на работу идет на конкурсной основе. Условия приема требуют 20 лет рабочего стажа и возраста не более 42 лет. Определите, будет ли человек принят на работу.

3*. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% - если сумма больше 1000 руб.

4*. Написать программу, которая «задумывает» число в диапазоне от A до B и предлагает угадать число с некоторого числа попыток.

Методические рекомендации педагогу

Первые пять задач предназначены для решения со всей подгруппой с некоторыми пояснениями и возможно у доски.

Пять задач со * предлагаются к решению более сильными студентами.

Лабораторная работа

по теме «Реализация построенного алгоритма в виде программы. Элементы отладки программного кода»

Цель работы: систематизация навыков решения задач с использованием линейной стктуры и применение оператора Case of.

ЗАДАНИЕ

Решение задачи оценивается в 2 балла

Решить задачу с номером, соответствующим номеру компьютера.

1.Дан номер месяца (1 — январь, 2 — февраль, ...). Вывести название соответствующего времени года ("зима", "весна" и т.д.).

2. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия и два числа A и B (B не равно нулю). Выполнить над числами указанное действие и вывести результат.

3.Единицы длины пронумерованы следующим образом: 1 — дециметр, 2 — километр, 3 — метр, 4 — миллиметр, 5 — сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах.

4.Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы и масса тела M в этих единицах (вещественное число). Вывести массу данного тела в килограммах.

5.. Робот может перемещаться в четырех направлениях ("С" — север, "З" — запад, "Ю" — юг, "В" — восток) и принимать три цифровые команды: 0 — продолжать движение, 1 — поворот налево, –1 — поворот направо. Дан символ C — исходное направление робота и число N — посланная ему команда. Вывести направление робота после выполнения полученной команды.

6.Локатор ориентирован на одну из сторон света ("С" — север, "З" — запад, "Ю" — юг, "В" — восток) и может принимать три цифровые команды: 1 — поворот налево, –1 — поворот направо, 2 — поворот на 180 градусов. Дан символ C — исходная ориентация локатора и числа N1 и N2 — две посланные ему команды. Вывести ориентацию локатора после выполнения данных команд.

7.Элементы окружности пронумерованы следующим образом: 1 — радиус (R), 2 — диаметр (D), 3 — длина (L), 4 — площадь круга (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке). В качестве значения Pi использовать 3.14.

8.Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 — катет (a), 2 — гипотенуза (c), 3 — высота, опущенная на гипотенузу (h), 4 — площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

9.Элементы равностороннего треугольника пронумерованы следующим образом: 1 — сторона (a), 2 — радиус вписанной njpsfmnqrh (R1), 3 — радиус описанной окружности (R2), 4 — площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

10. Дано целое число в диапазоне 20 – 69, определяющее возраст (в годах). Вывести строку — словесное описание указанного возраста, обеспечив правильное согласование числа со словом "год", например: 20 — "двадцать лет", 32 — "тридцать два года", 41 — "сорок один год"

11. Дано целое число в диапазоне 100 – 999. Вывести строку — словесное описание данного числа, например: 256 — "двести пятьдесят шесть", 814 — "восемьсот четырнадцать".

12. В восточном календаре принят 60-летний цикл, состоящий из 12- летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В каждом подцикле годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года вывести его название, если 1984 год был началом цикла — годом зеленой крысы.

13. Арифметические действия над числами пронумерованы следующим образом: 1 — сложение, 2 — вычитание, 3 — умножение, 4 — деление. Дан номер действия N (целое число в диапазоне 1–4) и вещественные числа

A и B (B не равно 0). Выполнить над числами указанное действие и вывести результат. Использовать оператор case.

14. Написать программу, выдающую по введенному возрасту пользователя сообщен к какому возрастному периоду он относится (подростковый, ранняя юность, юношество, взрослый период, зрелость).

15. Написать программу, которая по введенному цвету светофора выводит на экран что нужно делать пешеходу.

Выберите из предложенного списка вторую и третью задачу для решения.

1. Написать программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.
2. Написать программу, которая по номеру месяца выдает название следующего за ним месяца (при m=1 получаем февраль, при m=4 — май).
3. Написать программу, которая по вводимому числу от 1 до 5 (номеру курса) выдает соответствующее сообщение «Привет, k-курсник». Например, если k=1, «Привет, первокурсник»; при k=4: «Привет, четверокурсник».
4. Написать программу, которая по данному натуральному числу от 1 до 12 (номеру месяца) выдает все приходящиеся на этот месяц праздничные дни (например, если введено число 1, то: 1 января — Новый год, 7 января — Рождество).
5. Составить программу, позволяющую по последней цифре данного числа определить последнюю цифру куба этого числа.
6. Составить программу, позволяющую вывести полное название по сокращению: б. – бульвар, у. - улица, пр.- проспект, пер. – переулок.
7. Составить программу, которая для любого натурального числа печатает количество цифр в записи этого числа. Предполагается, что исходное число имеет не больше восьми цифр в записи.
8. Составить программу для определения подходящего возраста кандидатуры для вступления в брак, используя следующее соображение: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется соответственно как удвоенный возраст девушки минус 14.
9. Составить программу, которая бы по введенному названию страны выдавала название ее континента. (Задействовать не менее 8 стран)
10. Составить программу, позволяющую по обозначению учебной группы сообщить полное название специальности
11. Составить программу, позволяющую вывести название региона России по кодовым цифрам в авто номере: 102 – Башкортостан.
12. Составить программу, позволяющую по номеру в обозначении учебной группы сообщить курс обучения.
13. В зависимости от того введена ли открытая скобка или закрытая, напечатать "открытая круглая скобка" или "закрытая фигурная скобка". (Учитывать круглые, квадратные, фигурные скобки).

Лабораторная работа

по теме «Редактирование программного кода в соответствии со стандартами кодирования»

Цель работы: применение стандартов кодирования при решении простейших задач.

ЗАДАНИЕ

Решение задачи оценивается в 2,5 балла

Написать программу с дружественным интерфейсом.

Во всех заданиях требуется вывести логическое значение True, если приведенное высказывание для предложенных исходных данных является истинным, и значение False в противном случае. Все числа, для которых указано количество цифр (двузначное число, трехзначное число и т.д.), считаются целыми.

Задача № 1.Проверить истинность высказывания: "Данные числа x, y являются координатами точки, лежащей во второй координатной четверти".

Задача № 2.Проверить истинность высказывания: "Данные числа x, y являются координатами точки, лежащей в первой или третьей координатной четверти".

Задача № 3.Проверить истинность высказывания: "Данное целое число является четным двузначным числом".

Задача № 4.Проверить истинность высказывания: "Данное целое число является нечетным трехзначным числом".

Задача № 5.Проверить истинность высказывания: "Среди трех данных целых чисел есть хотя бы одна пара взаимно противоположных".

Задача № 6.Проверить истинность высказывания: "Сумма цифр данного трехзначного числа является четным числом".

Задача № 7.Проверить истинность высказывания: "Сумма двух первых цифр данного четырехзначного числа равна сумме двух его последних цифр".

Задача № 8.Проверить истинность высказывания: "Данное четырехзначное число читается одинаково слева направо и справа налево".

Задача № 9.Проверить истинность высказывания: "Все цифры данного трехзначного числа различны".

Задача № 10. Проверить истинность высказывания: "Среди трех данных целых чисел есть хотя бы одна пара совпадающих".

Задача № 11. Проверить истинность высказывания: "Точка с координатами (x, y) лежит внутри прямоугольника, левая верхняя вершина которого имеет координаты (x1, y1), правая нижняя — (x2, y2), а стороны параллельны координатным осям".

Задача № 12. Проверить истинность высказывания: "Цифры данного трехзначного числа образуют возрастающую последовательность".

Задача № 13. Проверить истинность высказывания: "Цифры данного трехзначного числа образуют возрастающую или убывающую последовательность".

Задача № 14. Проверить истинность высказывания: "Все цифры данного трехзначного числа различны".

Задача № 15. Проверить истинность высказывания: "Среди цифр данного трехзначного числа есть 1 или 0".

Лабораторная работа

по теме «Реализация программ разветвляющейся структуры»

Цель работы: систематизация знаний и умений решения задач разветвляющейся структуры.

ЗАДАНИЕ

Решение задачи оценивается в 2,5 балла

Задача № 1.Из. цифр трехзначного числа найти максимальное.

Задача № 2.Из цифр трехзначного числа найти минимальное.

Задача № 3.Для трех данных чисел вычислить сумму наибольшее и наименьшего.

Задача № 4.Из трех данных чисел сообщить не наименьшее и не наибольшее.

Задача № 5.Перераспределить значения переменных X и Y так, чтобы в X оказалось меньшее из этих значений, а в Y — большее.

Задача № 6.Значения переменных X, Y, Z поменять местами так, чтобы они оказались упорядоченными по возрастанию.

Задача № 7.Введенное с клавиатуры число уменьшить в 10 раз, если оно двузначное, увеличить в 10 раз, если это цифра и оставить без изменения, если оно трехзначное.

Задача № 8.Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной максимальное из этих значений, а если равны, то присвоить переменным нулевые значения.

Задача № 9.Даны вещественные координаты точки, не лежащей на координатных осях OX и OY. Вывести номер координатной четверти, в которой находится данная точка.

Задача № 10. Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Вывести порядковый номер этого числа.

Задача № 11. Для данного x вычислить значение следующей функции f, принимающей вещественные значения:

$$f(x)=\begin{cases}1, & \text{если } x <= 0,\\ x\cdot x, & \text{если } 0 < x < 2,\\ 4, & \text{если } x >= 2.\end{cases}$$

Задача № 12. Для четырех заданных чисел A, B, C, D определить что больше их сумма или произведение.

Задача № 13. Для данного x вычислить значение следующей функции f, принимающей значения целого типа:

$$f(x)=\begin{cases}0, & \text{если } x < 0,\\ 1, & \text{если x принадлежит } [0,1)\\ -1, & \text{если x принадлежит } [1,4)\end{cases}$$

Задача № 14. Для введенного с клавиатуры числа X сообщить делится ли оно на 2 , 3 или 5.

Задача № 15. Ввести с клавиатуры число сообщающее текущее время и сообщить в зависимости от времени: «1 пара», «2 пара», «Обед», «3 пара», «Домой».

Найдя первый гриб или сделав первое открытие, осмотритесь вокруг, - они родятся пучками.
Д. Пойа

Лабораторная работа

по теме «Реализация программ циклической структуры»

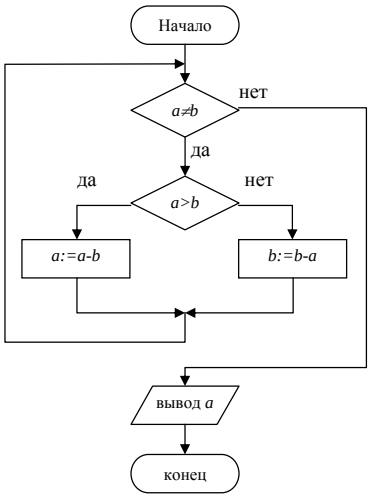
Цель работы: практическое применение знаний и умений решения задач циклической структуры.

ЗАДАНИЕ

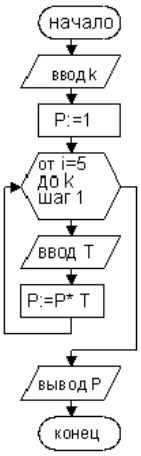
Решение задачи оценивается в 1,5 балла

По предложенной блок-схеме составить программу.

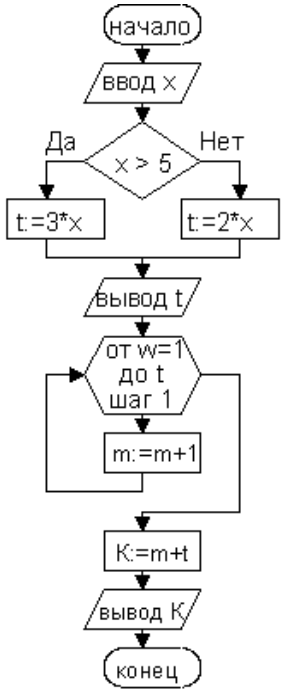
Блок-схема №1



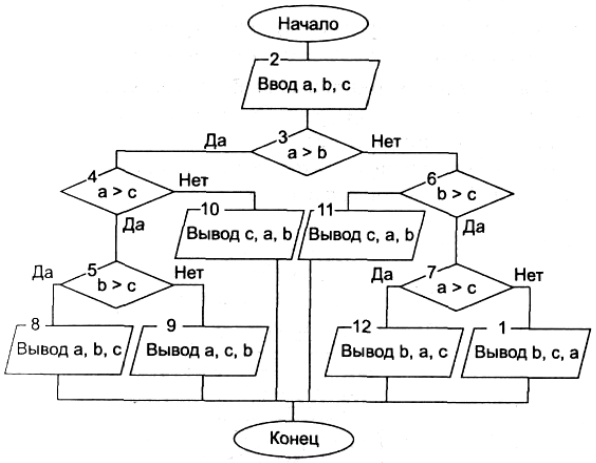
Блок-схема №2



Блок-схема №3



Блок-схема №4



Лабораторная работа

по теме «Реализация цикла с параметром»

Цель работы: практическое применение знаний и умений решения задач на цикл с параметром.

.

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

1. Составить программу вычисления суммы S=1^2+2^+3^2+4^2...+10^2
2. Маша очень любит пирожные. Она пришла в кондитерскую с суммой денег S. Ей очень понравилось пирожное «Бизе» и она купила его, заплатив Zрублей и тут же его съела. Не наелась, увидела пирожное «Заварное» и тоже купила его и попробовала. И так повторилось несколько раз, пока она не насытилась. Определить, сколько пирожных сумела съесть Маша и в какую сумму это ей обошлось.
3. Вывести на экран натуральные числа от 1 до 100, которые при делении на 6 дают в остатке 4, и их количество.
4. Определить суммарный объём в литрах 12 вложенных друг в друга шаров со стенками 5 мм. Внутренний диаметр внутреннего шара равен 10 см. Считать, что шары вкладываются друг в друга без зазоров.
5. Даны натуральные числа *n, k* (n, k<=9999). Из чисел от *n* до *k* выберете те, запись которых содержит три одинаковых цифры. Например, числа 6766, 5444, 0006, 0060 содержат ровно три одинаковых цифры.

1*. Найти первый отрицательный член последовательности sin(tg(n/2)) для n, изменяющегося следующим образом: n=1,2,3…

2*. Найти сумму цифр целого числа, больших 5.

3*. Дано натуральное число n. Верно ли, что сумма цифр этого числа является **нечётной.**

4*. Найти все делители натурального числа n.

Методические рекомендации педагогу

Первые пять задач предназначены для решения со всей подгруппой с некоторыми пояснениями и возможно у доски.

Пять задач со * предлагаются к решению более сильными студентамии.

Лабораторная работа

по теме «Составление алгоритма задачи на использование цикла с предусловием»

Цель работы: систематизация знаний и умений решения задач циклической структуры.

ЗАДАНИЕ

Написать программу для решения задачи и составить блок-схему.

Сколько слагаемых должно быть в сумме 1 + ½ + 1/3 + …+ 1/N, чтобы сумма оказалась больше 5.

Задачи для самостоятельного решения

Решение задачи оценивается в 1,5 балла

Написать программу для решения задачи и составить блок-схему.

1. С помощью цикла "пока" или цикла "до" написать программу возведения числа A в целую степень N.
2. В первый день пловец проплыл 3 км. В каждый следующий день он проплывал на 10% больше, чем в предыдущий.

а) В какой по счету день пловец начнет проплывать более 5 км?

б) К какому дню он суммарно проплывет более 30 км?

3. Урожай яблок в 1990 году составил 20 тонн. Далее каждые два года урожай уменьшался на 20%.

а) Начиная с какого года, будет собрано менее 5 тонн?

б) В каком году суммарный урожай яблок превысит 90 тонн?

4. Найти наименьший номер члена последовательности, для которого выполняется условие |an-an-1|<e, где an = arctg an-1+1, a1=0. Вывести на экран этот номер и все элементы ai (i = 1, 2, ..., n).

Лабораторная работа

по теме «Составление алгоритма задачи на использование цикла с постусловием»

Цель работы: систематизация знаний и умений решения задач циклической структуры.

ЗАДАНИЕ

Написать программу решения задачи

Решение задачи оценивается в 1,5 балла

1. Концентрация хлорной извести в бассейне V м3 составляет 10 г/л. Через одну трубу в бассейн вливают чистую воду со скоростью Q м3/час, через другую трубу с такой же скоростью вода выливается. При условии идеального перемешивания концентрация хлорной извести изменятся по закону C = C0e-Qt/V , где t – время, C0 – начальная концентрация.

а) Определить, через какое время концентрация хлорной извести в бассейне достигнет безопасной для человека величины. Задачу решить при Q = 150 м3/час, V = 10000 л, C0 = 10 г/л, t изменяется с шагом 0,5.

б) Напечатать таблицу изменения концентрации хлорной извести для интервала времени от 0 до 5 часов с шагом 0,5 часа.

2. С клавиатуры вводится произвольная последовательность положительных чисел, за которой следует 0. Определить:

а) Максимальное число в этой последовательности.

б) Количество четных чисел.

3. Найти значение минимального положительного члена числовой последовательности, заданной следующими соотношениями: Xn = Xn-1+ Xn-2+100 ; X1= X2= -99.

4. Вычислить значения функции y=ae –bx для x=0.1,0.2,0.3,..., прекратив вычисления тогда, когда значение e –bx станет меньше 0.001; a и b – действительные числа.

5. Сколько раз исполнится тело цикла во фрагменте программы?

   M := 123; While M <> 0 Do M := M Mod 10;

Для цикла с предусловием запишите его полный эквивалент с помощью цикла с постусловием.

6. Для цикла с постусловием запишите его полный эквивалент с помощью цикла с предусловием. Для цикла с параметром запишите его полный эквивалент с помощью циклов с пред- и постусловием.

7. Составьте алгоритм вычисления суммы всех натуральных чисел, не превышающих заданного натурального числа N.

Лабораторная работа

по теме «Составление и отладка программы по блок-схеме»

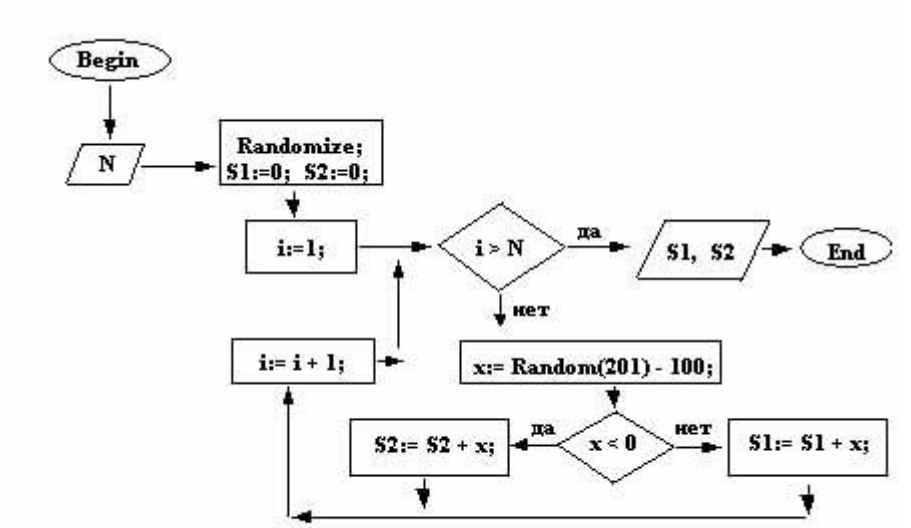
Цель работы: систематизация знаний и умений читать блок-схемы и выполнять отладку программы.

ЗАДАНИЕ

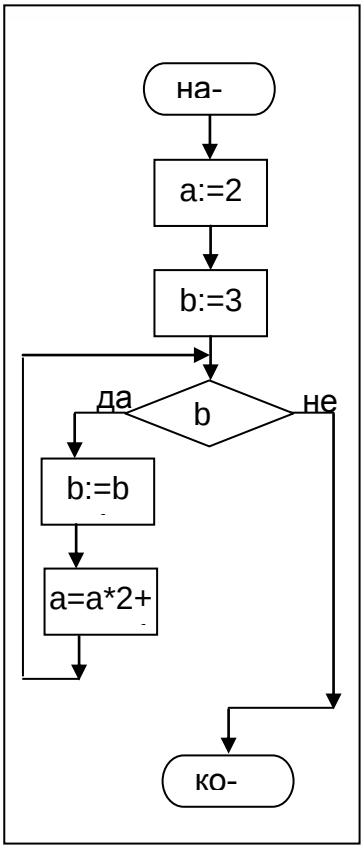
Написать программу по предложенной блок-схеме

Решение задачи оценивается в 1,5 балла

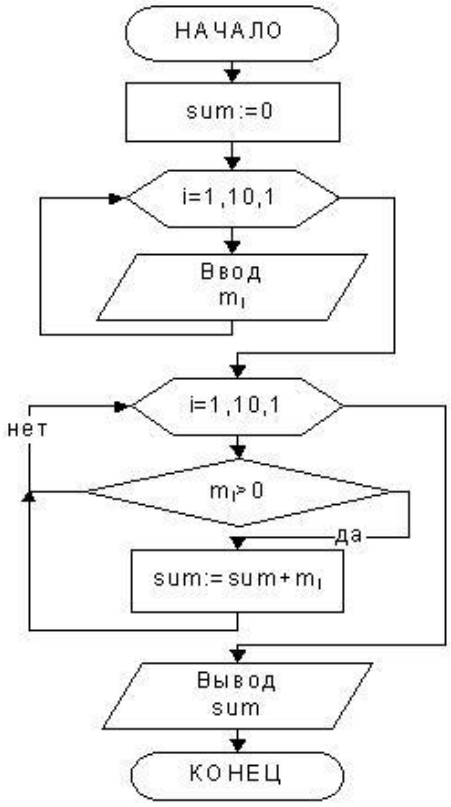
Блок-схема 1



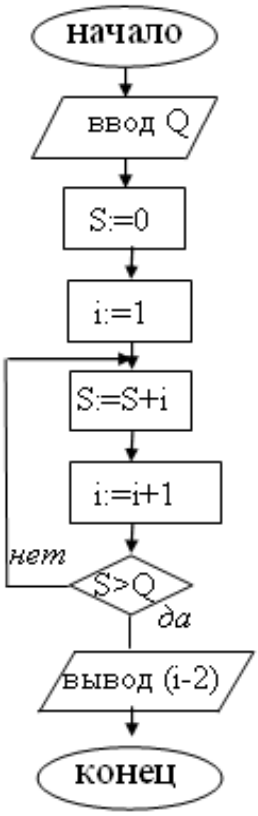
Блок-схема 2



Блок-схема 3



Блок-схема 4



Лабораторная работа

по теме «Реализация программ усложненной структуры»

Цель работы: отработка навыков и умений составления кода решения задач усложненной структуры.

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу для решения задачи.

1. Известны данные о количестве осадков, выпавших за каждый день месяца. Какого числа выпало самое большое количество осадков? Если таких дней несколько, то должна быть найдена дата последнего из них.
2. Даны натуральное число n и целые числа a1,a2,…,an. Выяснить, верно ли что сумма тех чисел ai, которые не больше m , превышает q.
3. Найти все целые числа из промежутка от1 до 300, у которых ровно пять делителей.
4. Даны натуральные числа m, n. Вычислить $1^n + 2^n + 3^n + \ldots + m^n$

Методические указания

При решении предложенных задач следует использовать вложенные циклы. Причем необходимо помнить что вложенный цикл проходит все свои значения для каждого значения переменной внешнего цикла. И при повторном проходе вложенного цикла входные параметры должны обнуляться или принимать начальные значения.

Лабораторная работа

по теме «Тип данных множество»

Цель работы: отработка навыков и умений составления кода решения задач с типом данных множество.

Задача 1. Дана строка. Сохранить в ней только первые вхождения символов, удалив все остальные.

```
program ex_set_3;
var m : set of char;
   s : string; i : byte;
begin
   write('Введите строку: ');
   readln(s);
   m :=[];
   i := 1;
   while i <= length(s) do
   if s[i] in m then delete(s, i, 1)
               else begin m:=m+[s[i]]; i := i + 1 end;
   writeln(s)
end.
```

Задача 2.

Дана строка символов. Удалить из нее все знаки препинания.

```
var   str, str1 : string;
       L, i : integer;
        m : set of char;
begin
        m:=['.', ',', '!', ':', ';', '?', '-'];
        writeln('Введите текст');
        readln(str);  L:=length(str); str1:='';
        for i:=1 to L do
             if not(str[i] in m) then str1:=str1+str[i];
        str:=str1;
        writeln('Преобразованный текст ', str);
end.
```

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Задача 3.

Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке гласных букв

Лабораторная работа

по теме «Реализация способов обработки одномерных массивов»

Цель работы: практическое применение операторов обрадотки одномерного массива

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу для решения задачи.

1. Задана последовательность из N вещественных чисел. Определить, сколько среди них чисел меньших К, равных К и больших К

2. Определить, сколько процентов от всего количества элементов последовательности целых чисел составляют нечетные элементы.

3. Дана последовательность натуральных чисел a1, a2,..., an. Создать массив из четных чисел этой последовательности. Если таких чисел нет, то вывести сообщение об этом факте

4. Дан массив действительных чисел, размерность которого N. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов

1*. В массиве целых чисел с количеством элементов п найти наиболее часто встречающееся число. Если таких чисел несколько, то определить наименьшее из них.
2*. Дана последовательность целых чисел a1, a2,..., an. Образовать новую последовательность, выбросив из исходной те члены, которые равны min(a1, a2,..., an,).
3*. У прилавка магазина выстроилась очередь из п покупателей. Время обслуживания i-го покупателя равно ti (i = 1, ..., п). Определить время Ci пребывания i-го покупателя в очереди.

Методические рекомендации педагогу

Первые пять задач предназначены для решения со всей подгруппой с некоторыми пояснениями и возможно у доски.

Пять задач со * предлагаются к решению более сильными студентамии.

Лабораторная работа

по теме «Поиск минимального(максимального) элемента в массиве.»

Цель работы: практическое применение приемов нахождения максимального и минимального элементов массива

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу для решения задачи.

1. Дан одномерный массив A[N]. Найти $\max(a_2,\ a_4,\ \ldots,\ a_{2k}) + \min(a_1,\ a_3,\ \ldots,\ a_{2k+1}).$

2. Даны действительные числа a1, a2,..., an. Найти $\max(a_1 + a_{2n},\ a_2 + a_{2n-1},\ \ldots,\ a_n + a_{n+1}).$

3. Дана последовательность целых чисел a1, a2,..., an. Образовать новую последовательность, выбросив из исходной те члены, которые равны min(a1, a2,..., an,).

4. Дан массив действительных чисел. Среди них есть равные. Найти его первый максимальный элемент и заменить его нулем.

Лабораторная работа

по теме «Составление программы на использование строковых одномерных массивов»

Цель работы: практическое применение приемов решения задач со строковыми массивами

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Алгоритм, отвечающий на вопрос: "Есть ли буква "А" в тексте?".

В булевской переменной OT будет содержатся ответ, есть или нет буква "А" в тексте. Перед началом сравнения будем считать, что нужной буквы нет и значением OT будет FALSE. Каждую букву слова будем сравнивать с буквой "А". Если сравнение произойдет, то переменной OT присвоим значение TRUE и закончим поиск. Если не найдем нужной буквы, то значение переменной OT останется без изменения.

После окончания цикла в переменной OT будет содержатся нужное значение.

```
program task;
  uses crt;
  var i,d:integer;
      t:string;
      ot:boolean;
  begin
   clrscr;
   write('Ведите текст ');readln(t);
   d:=length(t);
   i:=1; ot:=false;
   repeat
    if t[i]='A'
       then ot:=true;
       i:=i+1;
   untul (i>d) or ot;
    if ot
       then writeln('есть нужная буква')
       else writeln('нет нужной буквы');
   end.
```

Написать программу для решения задачи.

Задачи для самостоятельной работы:

Задача 1. Проверить, есть ли в введенном тексте пара одинаковых символов.

Задача 2. Напечатать заданный текст, удалив из него все пробелы.

Задача 3.Напечатать заданный текст, заменив все "+" на "-".

Задача 4. Напечатать заданный текст, удалив из него по одному символу, стоящему за буквой "а".

Лабораторная работа

по теме «Решение задач на обработку строк»

Цель работы: практическое применение операторов обработки строк.

Вопросы для проверки знаний:

1. Что такое длина слова?
2. Какие значения принимает логическая переменная?
3. Что мы называем логическим выражением?
4. Как можно обратится к конкретному символу литерной переменной?
5. Как определить длину литерной переменной?
6. Как выделить из литерной переменной части?

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу для решения задачи.

Задачи для самостоятельной работы:

Задача 1 Строка вводится посимвольно. Конец ввода ".". Указать номер первой запятой.

Задача 2 Напечатать из заданного текста все сим волы:

а) от начала до первой "*";

б) от первой "*" и до конца;

в) от первой "*" и до второй "*".

Задача 3В заданном тексте удвоить "=", а "-" удалить.

Задача 4 В тексте допущена ошибка. Вместо "мишки" везде напечатано слово "мышка". Исправить ошибку.

Лабораторная работа

по теме «Выполнение сортировки массивов»

Цель работы: практическое применение способов сортировки массива

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу и блок-схему решения задачи.

1. Дан одномерный массив. Упорядочить его по убыванию начиная с максимального элемента. Все элементы массива до максимального оставить на своих местах. (Сортировка вставкой)
2. Дан одномерный массив, заполненный случайным образом. Элементы между минимальным и максимальным упорядочить по возрастанию. (Сортировка поиском максимального)
3. Упорядочить по убыванию элементы расположенные на четных местах. (Сортировка «пузырьком»)

Лабораторная работа

по теме «Решение поставленной задачи с использованием сортировки массива»

Цель работы: практическое применение способов сортировки массива

ЗАДАНИЕ

Решение задачи оценивается в 2 балла

Написать программу решения задачи.

1. Даны два массива одного размера. Получить третий массив, каждый элемент которого равен:

а) сумме элементов с тем же номером; полученный массив упорядочить по возрастанию;

б) максимальному из элементов с тем же номером в заданных массивах; полученный массив упорядочить по убыванию.

2. Из элементов массива А сформировать массив В того же размера по правилу: неотрицательные элементы исходного массива уменьшить в три раза, ос тальные возвести в квадрат. Новый масив упорядочить по убыванию.

Лабораторная работа

по теме «Решение задач с использованием двумерных массивов»

Цель работы: практическое применение способов обработки двумерных массива

ЗАДАНИЕ

Решение задачи оценивается в 1 балл

Заполнить двумерный массив A[4×3] случайным образом.

а) Найти и вывести на экран индексы заданных элементов массива (если их нет выдать соответствующее сообщение):

- четных элементов каждой строки и нечетных элементов каждого столбца;
- симметричных чисел;
- первых k отрицательных элементов каждого столбца;
- последних k отрицательных элементов каждой строки;
- последних k отрицательных элементов каждого столбца;
- равных между собой элементов каждого столбца;
- элементов, являющимися действительными числами;
- элементов, являющихся трехзначными числами;
- элементов, не имеющих целой части;
- элементов, являющихся числами, сумма цифр которых равна заданному числу;
- элементов, являющихся числами, первая цифра которых равна заданной.

б) Найти сумму и количество элементов с заданным условием (хранить эти значения в массивах):

- элементы каждого столбца, кратные k1 или k2;
- элементы каждого столбца, попадающие в промежуток [A..B];
- элементы каждого столбца, которые являются простыми числами;
- элементы каждого столбца положительны и лежат выше главной диагонали;
- отрицательные элементы каждого столбца, меньшие заданного числа a;
- элементы каждого столбца, меньшие среднего арифметического элементов каждого столбца;
- элементы каждой строки, больших среднего арифметического элементов данной строки;
- максимальные элементы каждой строки;
- отрицательные элементы каждой строки;
- элементы каждой строки, равные сумме соседних с ним элементов;
- элементы каждой строки, равные элементу в том же столбце, но в предыдущей строке.

Лабораторная работа

по теме «Решение задач на обработку двумерных массивов»

Цель работы: систематизация способов обработки двумерных массива

ЗАДАНИЕ

Решение задачи оценивается в 1 балл

Заполнить двумерный массив A[3×5] случайным образом.

а) Используя функцию булева типа, определить:

- есть ли в данном массиве отрицательный элемент;
- есть ли два одинаковых элемента;
- есть ли данное число A среди элементов массива;
- является ли массив логическим квадратом, то есть суммы по всем горизонталям, вертикалям и двум диагоналям должны быть равны;
- добавить к предыдущему условию, что сумма должна быть равна данному числу A;
- состоящая только из положительных элементов;
- состоящая только из положительных или нулевых элементов;
- состоящая только из элементов, принадлежащих промежутку от A до B.

б) Измените исходный массив в соответствии с заданием:

1. в каждой строке сменить знак максимального по модулю элемента на противоположный;
2. отрицательный последний элемент каждого столбца заменить нулем;
3. положительные элементы умножить на первый элемент соответствующей строки, а отрицательные - на последний;
4. заменить все элементы строки с номером k и столбца с номером p на противоположные по знаку (элемент, стоящий на пересечении, не изменять);
5. к элементам столбца k1 прибавить элементы столбца k2;
6. переверните в массиве каждую третью строку;
7. поменяйте местами заданные элементы каждого столбца;
8. переверните в массиве каждую половину каждого столбца.

Лабораторная работа

по теме «Поиск максимального (минимального) элемента в двумерном массиве»

Цель работы: систематизация способов обработки двумерных массива

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу решения задачи

Заполнить двумерный массив A[3×5] случайным образом.

1. найти минимальный и максимальный элементы столбца и заменить их суммой последний элемент;
2. добавьте к массиву столбец, содержащий максимальный элемент соответствующей строки, и строку содержащую минимальный элемент соответствующего столбца;
3. найти максимальный элемент каждой строки и заменить им все минимальные элементы строки;
4. укажите номер строк, находящихся между первым минимальным и последним максимальным элементами текущего столбца. Если таких строк нет, то вывести сообщение об этом

Лабораторная работа

по теме «Использование стандартных функций для работы со строками и массивами»

Цель работы: систематизация способов обработки двумерных массива

Для решения задачи вставки строки необходимо:
- Первые k строк оставить без изменения.
- Все строки после k-ой сдвинуть на одну назад, это лучше сделать, начиная с последней строки и идти до (k+1)-ой.
- Элементам строки k+1 присвоить заданное значение.
- Увеличить количество строк.

Кроме того, необходимо изменить размерность массива. Так как мы вставляем строку, то число строк будет на одну больше.

ЗАДАНИЕ

Решение задачи оценивается в 2,5 балла

Задачи на вставку элементов:

1. Вставить первую строку после строки, в которой находится первый встреченный максимальный элемент и первый столбец перед всеми столбцами, в которых встретится заданное число. Если такого столбца или строки нет, то вывести сообщение об этом.

2. Вставить нулевую строку и нулевой столбец перед строками и столбцами, где находятся минимальные элементы.

3. Вставить перед всеми строками, в которых есть 0, первую строку, а после всех столбцов, в которых есть отрицательные элементы - первый столбец.

Задачи на удаление элементов:

1. Если в массиве есть равные строки, то удалите их. Если в получившемся после удаления строк массиве обнаружен столбец, каждый элемент которого больше на единицу соответствующего элемента в предыдущем столбце, то удалите его. Если такого столбца или строки нет, то вывести сообщение об этом.

2. Удалите строки, содержащие ноль, а затем столбцы, в которых только отрицательные элементы. Если такого столбца или строки нет, то вывести сообщение об этом.

3. Удалите строки, содержащие более одного максимального элемента, а затем столбцы, сумма элементов которых равна заданному числу. Если такого столбца или строки нет, то вывести сообщение об этом.

Лабораторная работа

по теме «Сортировка двумерного массива»

Цель работы: систематизация способов сортировки массива

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Заполнить двумерный массив A[5×5] случайным образом.

1. Выполнить сортировку по возрастранию строки , номер которой вводится с клавиатуры.

2. Выполнить сортировку по убыванию столбца , номер которой вводится с клавиатуры.
3. Выполнить сортировку по возрастранию главной диагонали матрицы.
4. Выполнить сортировку по убыванию побочной диагонали матрицы.

Лабораторная работа

по теме «Решение задач с использованием записей»

Цель работы: практическое применение операторов обработки данных сложной структуры.

ЗАДАНИЕ

Решение задачи оценивается в 2 балла

Написать программу решения задачи.

1. В файл записать информацию о сотрудниках некоторого предприятия: фамилия, домашний адрес, телефон, образование, оклад. Напечатать список сотрудников, имеющих высшее образование. В файл записать информацию о сотрудниках некоторого предприятия: фамилия, домашний адрес, телефон, образование, оклад. Напечатать список сотрудников, имеющих высшее образование.

2. Реализовать алгоритм обработки данных сложной структуры типа запись.

Информация службы занятости о вакансиях содержит следующие данные: должность, заработок, необходимые образование и стаж работы, количество вакантных мест.

Написать программу, в которой:

- обеспечить ввод данных с клавиатуры и сохранение их в типизированном файле;
- организовать выборку из типизированного файла информации о вакансии при заданном образовании с минимальным необходимым стажем, вывод данных о них на экран и сохранение в текстовом файле.

3. Создать не текстовый файл, который содержит сведения об игрушках (Наименование, цена, возрастные границы например, от 3-х до 10 лет).

Вывести в текстовый файл названия игрушек, цена которых не превышает 200 рублей и предназначенных для детей 5 лет.

Вывести на экран цену самого дорогого ‘Конструктора’.

Лабораторная работа

по теме «Решение задач с использованием записей»

Цель работы: практическое применение операторов обработки данных сложной структуры.

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

Написать программу решения задачи.

1. Известны следующие данные о N учениках класса: фамилия, имя, отчество, дата рождения (число, месяц и год). Вывести на экран в каждой строке фамилию и имя тех учеников, у кого сегодня день рождения (сегодняшнюю дату вводить с клавиатуры).

2. Даны сведения о К пассажирах авиарейса: фамилия, имя, отчество, место в самолете, количество вещей и вес вещей в килограммах. Во второй массив записать только тех пассажиров, которые имеют количество вещей превосходящее среднее число вещей.

3. Известны следующие данные о N учениках школы: фамилия, имя, отчество, адрес (улица, дом, квартира), класс. Записать все данные об учениках определенного класса во второй массив. Распечатать его, выделяя тех из них, кто живет на улице Ленина.

4. Известны следующие данные о расписании К поездов: номер поезда, направление (откуда - куда, Киров - Москва), время прибытия на станцию, время отправления (часы, минуты). Будем считать, что все поезда приходят каждый день. По данному времени определить, какие из поездов стоят сейчас на станции (время вводить с клавиатуры).

5. Известны следующие данные о N сотрудниках: фамилия, имя, отчество, пол (в виде буквы М или Ж), возраст, номер отдела. Вывести список сотрудников в порядке увеличения возраста. Вывести номер отдела, в котором самый большой средний возраст сотрудников.

6. Даны сведения о К пассажирах авиарейса: фамилия, имя, отчество, место в самолете, количество вещей и вес вещей в килограммах. Во второй массив записать только тех пассажиров с багажом, средний вес одной вещи в котором отличается не более чем на 0,3 кг от общего среднего веса одной вещи.

7. Известны следующие данные о N учениках класса: фамилия, имя, отчество, адрес (улица, дом, квартира) и домашний телефон (если есть). Вывести на экран в каждой строке фамилию, имя и адрес тех учеников, у которых нет домашнего телефона.

Методические рекомендации педагогу

Более сильным студентам можно предложить начать решение задач с последнй.

Лабораторная работа

по теме «Организация процедур при решении задач»

Цель работы: практическое применение приемов написания процедур при решении задач.

ЗАДАНИЕ

Решение задачи оценивается в 1балл

Написать программу решения задачи с использование подпрограммы - процедуры.

1. Даны три пары целых переменных. Поменять местами их значения (попарно).

2. Даны 6 переменных A, B, C, H, P, K. Найти наибольшее, используя две процедуры: нахождения наибольшего из двух значений и нахождения наибольшего из трех значений.

3. Даны две переменные x, y (вещественные). Найти наименьшее из

а) x,y

б) x+y, x*y, 0.5

в) 2x, |x-y|, 4.5, [(x+y)/3]

Процедура нахождения наибольшего из двух значений.

4. Используя процедуру для вычисления степени числа, найти значение выражения:

а) y= a4*x4 + a3*x3 + a2*x2 + a1*x1 + a0 коэффициенты a4, a3, a2, a1, a0 и x - вводятся с клавиатуры;

б) y= a*x10 + b*x7 + c*x5 + d*x3 коэффициенты a, b, c, d, и x - вводятся с клавиатуры.

5. Даны четыре числа. Для каждого числа найти все его делители и подсчитать их количество.

6. Даны четыре числа. Про каждое сказать является ли оно палиндромом. Процедуры - переворот числа.

Лабораторная работа

по теме «Организация функций при решении задач»

Цель работы: практическое применение приемов написания функций при решении задач.

ЗАДАНИЕ

Решение задачи оценивается в 1балл

1. Написать функцию, подсчитывающую количество цифр натурального числа. Используя ее определить, в каком из двух заданных чисел больше цифр.
2. Написать функцию, определяющую является ли число простым. Вводятся четыре числа. Про каждое сказать простое оно или нет. Найти сумму простых чисел.
3. Составить программу нахождения НОД пяти чисел, используя функцию нахождения НОД двух чисел с использованием подпрограммы – функции.
4. Составить программу, вычисляющую НОК четырех чисел с использованием подпрограммы – функции.
5. Даны четыре целых числа. Вывести на экран наибольшую из старших цифр заданных чисел. Нахождение цифры числа оформить как функцию.
6. Даны пять целых чисел. Вывести сумму старшей и младшей цифр числа. Нахождение цифры числа оформить как функцию.

Лабораторная работа

по теме «Использование подпрограмм при решении задач»

Цель работы: практическое применение приемов написания функций при решении задач.

ЗАДАНИЕ

Решение задачи оценивается в 1,5 балла

1. Написать подпрограмму-функцию степени a^x, где a, x – любые числа. Воспользуемся формулой: $a^x = e^{x \ln a}$
2. Написать программу, состоящую из трех подпрограмм и основной программы. Подпрограммы должны организовывать ввод чисел, вычисление их суммы и вывод результата.
3. Написать функцию, которая возвращает процент числа, полученного в качестве аргумента

4. Написать функцию, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней).

Лабораторная работа

по теме «Использование процедур и функций при работе с массивами и строками»

Цель работы: практическое применение использования подпрограмм при решении задач с массивами.

ЗАДАНИЕ

Решение задачи оценивается в 1балл

1. Найти средние арифметические пяти массивов, состоящих их десяти целых чисел.
2. Задача на процедуру: дана квадратная матрица действительных чисел размером m(m<=10). Преобразовать матрицу так, чтобы на главной диагонали стояли максимальные элементы соответсвенного столбца.
3. Даны два предложения. Найти общее количество букв “н” в них. (Определить функцию для расчета количества букв “н” в предложении.)
4. Напишите процедуру MinMax, возвращающую наименьшее и наибольшее значения в массиве вещественных чисел. Для того, чтобы процедура могла работать с массивами различной длины, используйте в качестве параметра открытый массив.
5. Методом выбора определите максимальную и минимальную суммы элементов матрицы m × n при условии, что при вычислении очередной суммы в каждой строке и каждом столбце может быть выбран только один элемент. При решении задачи применить обычные массивы.

Лабораторная работа

по теме «Работа с графическим модулем»

Цель работы: практическое применение использования встроенных модулей.

ЗАДАНИЕ

Решение задачи оценивается в 5 баллов

Построить изодражение на дисплее, содержащее все геометрические фигуры, обрамленные линиями разного цвета и залитые различными способами заливки.

РАЗДЕЛ 4. КОНТРОЛЬ ЗНАНИЙ

Текущий контроль знаний и умений

Текущий контроль предусматривает систематическую проверку качества знаний и умений студентов по всем изучаемым в данном семестре дисциплинам. Формы текущего контроля знаний и умений отражены в календарно-тематических планах преподавателей.

Контроль знаний и умений студентов - один из важнейших элементов учебного процесса. От его правильной организации во многом зависит эффективность управления учебно-воспитательным процессом и качество подготовки специалиста.

Интенсивность и регулярность работы студентов зависит, главным образом, от частоты и регулярности проведения контроля. От этого же зависит и длительность сохранения в памяти усвоенных знаний. При организации и проведении контроля следует учитывать определенные свойства памяти.

Организация проведения контроля включает следующие этапы:

- разработка целей контроля;
- разработка содержания контрольных заданий;
- выбор организационных форм контроля, адекватных целям и содержанию;
- разработка методов контроля;
- разработка критериев оценки, результатов выполнения контрольных заданий и требований к их анализу.

Назначение контроля и предъявляемые к нему требования

Текущий контроль – оценка знаний и умений обучаемых.

Наиболее важная цель контроля - мотивирование регулярной и целенаправленной работы студентов. Интенсивность и регулярность работы студентов зависит, главным образом, от частоты и регулярности проведения контроля. От этого же зависит и длительность сохранения в памяти усвоенных знаний.

Чем чаще проходит контроль, тем лучше студент адаптируется к контрольной процедуре: его нервное напряжение значительно снижается.

Другая цель контроля - дать возможность студенту сопоставить свою работу с требованиями преподавателя, выяснить недочеты, недоработки и внести, если нужно, необходимые коррективы в свою подготовку.

Контроль знаний и умений студентов выполняет в процессе обучения следующие функции: проверочную (диагностическую), обучающую, развивающую, воспитательную и методическую.

Проверочная (диагностическая)	Данные контроля констатируют не только результаты и оценку учебной деятельности отдельных студентов и преподавателей, но и состояние учебно-воспитательной работы всего учебного заведения, подсказывают меры, необходимые для его совершенствования.
Обучающая	В ходе проведения контрольных заданий происходят повторение и закрепление, совершенствование приобретенных ранее знаний путем их уточнения и дополнения. Контроль способствует формированию умений и навыков, рационально организовывать учебную деятельность.
Развивающая	Контроль содействует развитию внимания, памяти, мышления, воображения, умению сопоставить и систематизировать имеющиеся знания, делать выводы, обобщения, приводить доказательства.
Воспитательная	Контроль дисциплинирует студента, воспитывает у него чувство ответственности за свою работу, приучает к систематическому труду, стимулирует регулярную активную учебную деятельность.
Методическая	Контроль позволяет оценить методы преподавателя, увидеть его сильные и слабые стороны, выбрать оптимальные варианты обучающей деятельности (контроль учит не только студента, но и преподавателя).

Для организации текущего контроля в УМК предусмотрены контрольные вопросы по темам, проверочные и самостоятельные работы, а также на каждой лабораторной работе каждый студент получает оценку за выполненные задания в соответствии с указанными баллами в каждом задании на лабораторную работу.

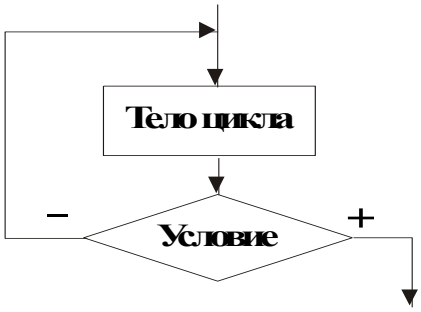
Самостоятельная работа по теме: "Циклы"
Вариант - 1

Задание 1. Тест

1. Если число повторений оператора заранее известно, то используют...
   а) оператор выбора б) цикл с условием
   в) цикл с параметром г) оператор ветвления
2. Сколько раз выполниться тело цикла?

```
B:=5;
While B =5 do
    B: =B+1;
Write (B);
```

   а) 0 раз
   б) 1 раз
   в) 3 раза
   г) много раз
3. Какой оператор соответствует блок-схеме?

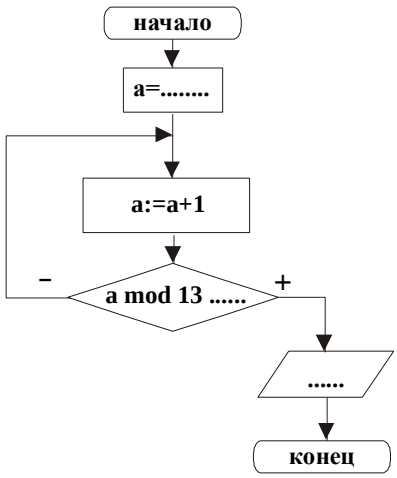


а) While <условие> do
<тело цикла>;
б) Repeat
<тело цикла>;
Until <условие>;
в) If <условие> Then <действие>;
г) for <*параметр*>:=*нач. знач.* to *кон. знач.* do
<тело цикла>;

4. Какое из нижеуказанных утверждений верно?
   а) выполнение оператора в цикле с условием не зависит от условия;
   б) цикл с предусловием может не выполнять тело цикла;
   в) цикл While всегда выполняется хотя бы раз;
   г) While и Repeat - это циклы с параметром.

Задание 2. Заполни пробелы, допущенные при решении следующей задачи: Составьте программу, которая находит наименьшее число кратное 13 и превосходящее 3000.

Решение:




```
Program demo;
Var      a: integer;
Begin
         a:= 3000;
.....
         a:=a+1;
..... a mod 13 =0;
         writeln (a);
end.
```

Задание 3. Составить блок- схему и программу решения задачи:

Вычислить значение суммы, заданной по формуле: S = sin x + sin 2x + sin 3x + ….. +sin 15x,
где x вводится с клавиатуры.

Задание 4. Составить блок- схему и программу решения задачи:

Объем озера ежедневно уменьшается на 7 %. Высохнет ли озеро на ¾ через 1 год (количество дней в году взять равным 365)?

Самостоятельная работа по теме: "Циклы"
Вариант - 2

Задание 1. Тест

1. Неоднократное исполнение оператора можно организовать, используя...

а) оператор выбора б) неоднократный ввод оператора

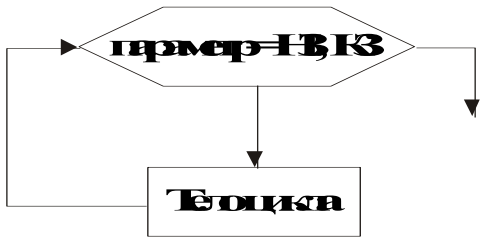
в) цикл г) оператор ветвления

2. Сколько раз выполниться тело цикла?

M:=3;
Repeat
M: =M+1;
Until M > 6;

а) 0 раз
б) 1 раз
в) 3 раза
г) много раз

3. Какой оператор соответствует блок-схеме?



а) While <условие> do
<тело цикла>;
б) Repeat
<тело цикла>;
Until <условие>;
в) If <условие> Then <действие>;
г) for <*параметр*>:=*нач. знач.* to *кон. знач.* do
<тело цикла>;

4. Какое из нижеуказанных утверждений верно?

а) цикл с условием всегда обязательно заканчивается словом Until;

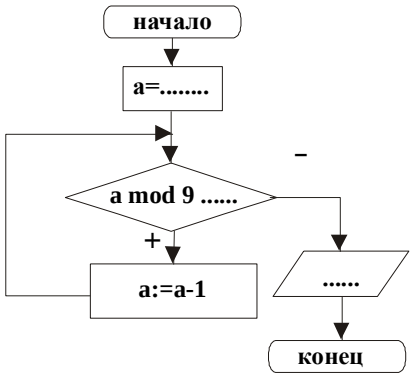
б) если необходимо несколько раз выполнить оператор, то программу запускают на исполнение несколько раз;

в) цикл Repeat может не выполнять тело цикла;

г) While - это цикл с условием.

Задание 2. Заполни пробелы, допущенные при решении следующей задачи: Составьте программу, которая находит наибольшее число кратное 9 и не превосходящее 5000.

Решение:




```
Program demo;
Var     a: integer;
Begin
        a:= 5000;
..... a mod 9 <>0 .....
        a:=a-1;
        writeln (a);
end.
```

Задание 3. Составить блок- схему и программу решения задачи:

Вычислить значение произведения, заданного по формуле: $P = \frac{1}{2} * \frac{1}{4} * \frac{1}{6} * \ldots\ldots * \frac{1}{50}$

Задание 4. Составить блок- схему и программу решения задачи:

Урожай в данный год равен 20 ц с 1 га. Каждые два года он увеличивается на 5% от предыдущего года. Достигнет ли через 25лет урожайность 40 ц с га?

Самостоятельная работа по теме: "Циклы"
Вариант - 3

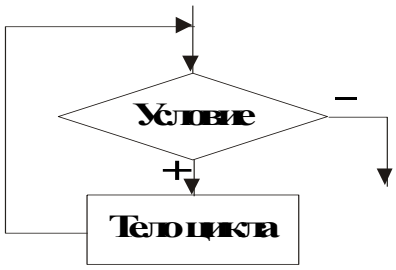
Тест:

1. Если задано только условие повторение оператора (или окончания), то используют...
   а) цикл с параметром б) цикл с условием
   в) оператор выбора г) оператор ветвления
2. Сколько раз выполниться тело цикла?

```
A:=0;
For K:=1 to 3 do
   A: =A+1;
```

   а) 0 раз
   б) 1 раз
   в) 3 раза
   г) много раз

3. Какой оператор соответствует блок-схеме?

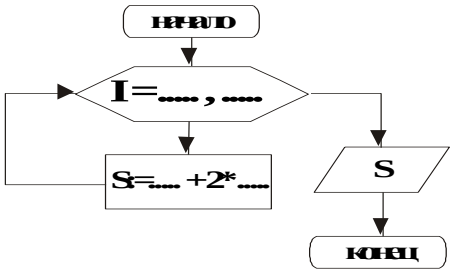


   а) While <условие> do
      <тело цикла>;
   б) Repeat
      <тело цикла>;
      Until <условие>;
   в) If <условие> Then <действие>;
   г) for <*параметр*>:=*нач. знач.* to *кон. знач.* do
   <тело цикла>;

4. Какое из нижеуказанных утверждений верно?
   а) цикл с условием всегда обязательно заканчивается словом Until;
   б) Repeat, While и For - это циклы с условием.
   в) если необходимо неоднократно выполнить некоторый оператор, то используют - цикл;
   г) если необходимо несколько раз выполнить оператор, то программу запускают на исполнение несколько раз;

Задание 2. Заполни пробелы, допущенные при решении следующей задачи: Составьте программу, которая находит сумму всех четных чисел до 300.

Решение:




```
Program demo;
Var      i, s: integer;
Begin
.......  i:= 1 ..... 150 do
         s:=s+2 * .....;
         writeln (s);
end.
```

Задание 2 Составить блок- схему и программу решения задачи:

Вычислить значение произведения, заданного по формуле: $S = \frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\ldots\ldots+\frac{n}{n+1}$,

вычислить для n =15

Задание 3 Составить блок- схему и программу решения задачи:

На елочный базар поставляются елки высотой не менее 3 м. Известно, что елку посадили в марте месяце и ее высота была 80 см ежемесячно елка вырастает на 20% от высоты предыдущего месяца. Можно ли будет срубить елку для продажи на елочном базаре в декабре месяце.

Самостоятельная работа по теме «Одномерные массивы»

1. Дан целочисленный массив A размера 10. Вывести номер первого из тех его элементов A[i], которые удовлетворяют двойному неравенству: A[1] < A[i] < A[10]. Если таких элементов нет, то вывести сообщение об этом.
2. Дан целочисленный массив размера N. Преобразовать его, прибавив к четным числам первый элемент. Первый и последний элементы массива не изменять.
3. Дан массив размера N. Найти количество элементов равных минимальному.
4. Дано вещественное число R и массив размера N. Найти количество элементов массива, больших данного числа.
5. Дан массив размера N. Преобразовать его, вставив перед каждого положительного элемента нулевой элемент.
6. Поменять местами минимальный и максимальный элементы массива размера 10.
7. Дан массив размера N. Преобразовать его, вставив после$_2$ каждого отрицательного элемента нулевой элемент.
8. Дан целочисленный массив A размера 10. Вывести номер последнего из тех его элементов A[i], которые удовлетворяют двойному неравенству: A[1] < A[i] < A[10]. Если таких элементов нет, то вывести сообщение об этом.
9. Дан массив, состоящий из 20 элементов. Проверить, все ли элементы этого массива имеют значение больше заданной величины R.
10. Проверить, действительно ли количество положительных элементов одномерного массива больше количества его отрицательных элементов.
11. Сообщить индекс максимального и минимального элементов одномерного массива.
12. Написать программу , заменяющую 3 средних элемента массива на разность минимального массива и первого.
13. Дан массив, состоящий из 20 элементов. Проверить, сколько элементов этого массива имеют значение меньшее среднего арифметического элементов массива.
14. Напишите программу, выводящую количество нулевых элементов одномерного массива.
15. Одномерный массив A длиной 25 элементов заполнить случайными числами. Определить количество элементов, значения которых лежат в диапазоне [y1..y2].

Самостоятельная работа по теме «Двумерные массивы»

1. Дано число k<10 и матрица размера 5 x 6. Заменить элементы столбца, содержащего элемент равный k на 1.
2. Дано число k (0 < k < 11) и матрица размера 4 x 10. Найти сумму и произведение элементов k-го столбца данной матрицы.
3. Дана матрица размера 5 x 10. Во второй строке найти количество элементов, больших 2.
4. Дана матрица размера 5 x 10.Поменять местами минимальный и максимальный элемент в матрице.
5. Дана матрица 5 x 5 найти минимальный элемент главной диагонали и заменить им угловые элементы.
6. Дана матрица размера 5 x 10. Вывести номер элемента равного R.
7. Дана квадратная матрица порядка M. Найти сумму элементов ее побочной диагонали и заменить ею саму диагональ.
8. Дана квадратная матрица порядка M. Заменить нулями элементы матрицы, лежащие выше главной диагонали.
9. Дана матрица размера 5 x 5. Найти среднее арифметическое четных элементов матрицы.
10. Дано число k<10 и матрица размера 4 x 6. Заменить элементы строки, содержащей элемент равный k на нули.
11. Для элементов главной диагонали найти минимальный элемент, вывести сумму элементов столбца, где этот элемент расположен.
12. Вывести индексы нулевых элементов матрицы размера 5 x 6.
13. Дана матрица размера 4 x 5. Сообщить количество элементов равных сумме индексов.
14. Найти количество отрицательных элементов в предпоследней строке матрицы размерностью 5 x 6.
15. Найти произведение ненулевых элементов матрицы размерностью 4 x 5.

Проверочная работа «Подпрограммы»
Выберите правильный ответ

1. Что представляют собой процедуры?

a. Служат для задания совокупности действий, направленных на изменение внутренних по отношению к ним программой обстановки

b. Служат для задания совокупности действий, направленных на изменение внешней по отношению к ним программой обстановки

c. Служат для определения алгоритма вычисления нового значения некоторого простого или ссылочного типа

d. Заключается в том, чтобы определить алгоритм вычисления нового значения некоторого простого или ссылочного типа

2. Что представляют собой функции?

a. Служат для задания совокупности действий, направленных на изменение внешней по отношению к ним программой обстановки

b. Служат для создания алгоритма вычисления нового значения некоторого простого или ссылочного типа

c. Определяют алгоритм вычисления нового значения некоторого простого или ссылочного типа

d. Служат для задания совокупности действий, приводящих к решению задачи

3. Какое из ниже приведенных описаний процедур верно?

a. procedure Input(x:real;): integer;
b. procedure Input(x,y; z:real);
c. procedure Input(x,y:integer;z:integer);
d. procedure Input(x,y:integer;z:array [1..3] of integer);

4. Какое из ниже приведенных описаний функций неверно?

a. function Output(x:real;var z:integer): integer;
b. function Output(x,y; z:real): real;
c. function Output(x,y:integer;z:integer):boolean;
d. неверно а и с;

5. Дано следующее описание блока программы:

```
procedure Sum(x:real);
  var
    i: integer;
begin
  i:=4;
  x:=i+2;
  Writeln(x);
end;
var
  x: real;
begin
  x:=3;
  y:=Sum(x);
end.
```

Какие ошибки допущены в данном описании?

a. В данном блоке нет ошибок
b. Неправильное описание процедуры
c. Неправильный вызов процедуры
d. Неправильное описание и неправильный вызов процедуры

6. Дано следующее описание блока программы:

```
procedure Sum(a,b:real):boolean;
  var
    i: integer;
begin
  i:=4;
  x:=i+2;
  Writeln(x);
end;
var
  x: real;
begin
  x:=3;
  Sum(x);
end.
```

Какие ошибки допущены в данном описании?

a. В данном блоке нет ошибок
b. Неправильное описание процедуры
c. Неправильный вызов процедуры
d. Неправильное описание и неправильный вызов процедуры

7. Дано следующее описание блока программы:

```
Function Sum(a,b:real):boolean;
  var
    i: integer;
begin
  i:=4;
  x:=i+2;
  If x>0 then Sum:=true
  else Sum:=False;
end;
```

```
var
  x: real;
begin
  x:=3;
  Sum(x,y);
end.
```

Какие ошибки допущены в данном описании?

a. В данном блоке нет ошибок
b. Неправильное описание функции
c. Неправильный вызов функции
d. Неправильное описание и неправильный вызов функции

8. Дано следующее описание блока программы:

```
Function Sum(a,b:real):boolean;
  var
    i: integer;
begin
  i:=4;
  x:=i+2;
  If x>0 then Sum:=true
  else Sum:=False;
end;
var
  x,y: real;
begin
  x:=3;
  y:=2;
  writeln(Sum(x,y));
end.
```

Какие ошибки допущены в данном описании?

a. В данном блоке нет ошибок
b. Неправильное описание функции
c. Неправильный вызов функции
d. Неправильное описание и неправильный вызов функции

9. Какие переменные называются локальными по отношению к подпрограмме?

a. Переменные, которые описаны в подпрограмме
b. Переменные, которые не видны в теле основной программы
c. Переменные, которые описаны перед подпрограммой
d. Правильны a и b

10. Какие переменные называются глобальными?

a. Переменные, которые описаны в начале программы
b. Переменные, которые видны во всех процедурах и функциях и теле основной программы
c. Переменные, которые изменялись в подпрограмме влекут изменения в теле основной программы
d. Все правильны

РУБЕЖНЫЙ КОНТРОЛЬ

В качестве рубежного контроля по дисциплине выступают контрольные работы по темам и ежемесячная аттестация по дисциплине.

Контрольная работа №1
«Основы работы в среде программирования»

Вариант 1

1. Сцепление модулей – это
   (а) мера его зависимости по данным от других модулей;
   (б) мера его независимости от предыстории обращений к нему;
   (в) мера его внутренних связей;
   (г) мера его параметров и переменных.
2. Структурное программирование _____
3. Качество ПО Мобильность – это
   (а) характеристики ПО, которые позволяют минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации;
   (б) способность ПО быть перенесенным из одной среды (окружения) в другую, в частности, с одной ЭВМ на другую;
   (в) характеристики ПО, которые позволяют минимизировать усилия пользователя по подготовке исходных данных;
   (г) способность ПО быть перенесенным из одной производственной обстановки в другую, в частности, с одной ЭВМ на другую.
4. Раскрыть принцип модульности разработки программ _____
5. Как обозначить комментарий в тексте программы?
6. Нарисуйте блок-схемы, реализующие основную алгоритмическую конструкцию цикла (с параметром, с постусловием).
7. Каким сочетанием клавиш можно запустить программу на исполнение:
   (а) Ctrl- F1;
   (б) *Alt-X* ;
   (в) Ctrl – F9;
   (г) Alt-F5.

Вариант 2

1. Программный модуль - это
   (а) любой фрагмент описания процесса, оформляемый как самостоятельная часть программы в разделе описаний;
   (б) определенная функция задачи, встроенная в систему программирования как основная;
   (в) любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт
   (г) заданная функция задачи, встроенная и загружаемая в память вместе с системой программирования.
2. Идея нисходящего структурного программирования «сверху-вниз» _____
3. Синтаксис языка программирования
   (а) правила написания объектов языка;
   (б) символы, используемые в написании объектов языка;
   (в) сопоставление объектов языка и их функций;
   (г) правила использования объектов языка.
4. Раскрыть принцип «по умолчанию»разработки программ _____
5. Запишите операторные скобки в программе, написанной на языке Паскаль.
6. Раскрыть свойства алгоритма – результативность и массовость.
7. Какой клавишей можно сохранить программный код в файл:
   (а) F10;
   (б) F3
   (в) F2;
   (г) F6.

Вариант 3

1. Выдерите неверный ответ
   Критериями качества ПО принято считать
   (а) эффективность;
   (б) простота использования;
   (в) функциональность;
   (г) надежность в использовании.
2. Модульное программирование _____
3. Качество ПО Сопровождаемость – это
   (а) характеристика ПО, минимизирующая усилия по внесению изменений, по модификации в соответствии с потребностями пользователей;
   (б) способность ПО выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей;
   (в) характеристики ПО, которые позволяют минимизировать усилия пользователя по подготовке исходных данных;
   (г) выделении в алгоритмах групп по частоте использования для последующего создания условий их быстрого выполнения.
4. Раскрыть частотный принцип разработки программ _____
5. Каким знаком разделяются операторы в программе, написанной на языке Паскаль?
6. Дать понятие алгоритма и перечислить его свойства.
7. Каким сочетанием клавиш можно вызвать справку по выбранному оператору в окне Паскаль:
   (а) Ctrl-F9;
   (б) Ctrl-Page Up
   (в) Ctrl - F1;
   (г) Ctrl- C.

Контрольная работа
«Основные алгоритмические конструкции»

Вариант 1

1. Дана последовательность операторов:

```
a:=1; b:=1;
While a+b<8 Do
Begin a:=a+l; b:=b+2 End;
S:=a+b;
```

Определите значения переменных a, b, и s после завершения этой последовательности операторов.

2. Тело цикла в программе

```
p:=1; a:=1;
while p<16 Do
Begin
  a:=2*a; p:=p*a;
End;
```

выполнится _____раз.

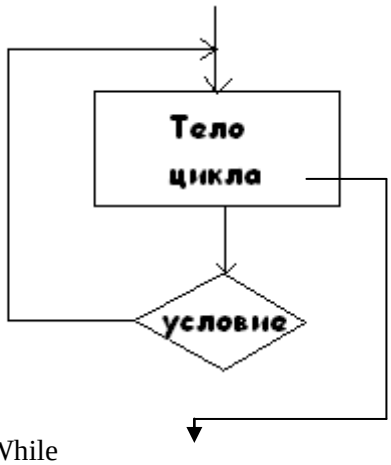
3 Что выведется на экран в результате выполнения фрагмента программы:

```
s:=0;
FOR j:=10 TO 15 DO
begin
   s:=s+2*j;
   write(‘ j=’, j:2, ’ s=’,s:4)
end;
```

4. В каком из операторов допущена синтаксическая ошибка:
   a. For i=1 to 20 do p:=p+1;
   b. While s<3 do s:=s-3;
   c. Repeat k:=k+1 until k<7;

d. For I:=10 downto 5 do p:=p+1;

5. Тип переменных для параметра цикла FOR:
   a. целочисленный тип
   b. логический тип
   c. натуральный тип
   d. верного ответа нет
6. Какой цикл называют циклом с предусловием?
   a. Repeat
   b. While
   c. For
7. Какой цикл изображен на блок схеме?



   a. While
   b. Repeat
   c. For
8. Параметр цикла For может получить значения…
   a. 1,2,3,4,5
   b. 2,4,6,8,10
   c. 1,3,5,7,9
9. Требуется подсчитать сумму натуральных чисел от 5 до 125. Какое условие нужно использовать в цикле While?
   a. i>125
   b. i<125
   c. i<=125.
10. Сколько раз исполнится цикл:
    i:=4;
    while i<10 do i:=i+3;
   a. 2 раза;
   b. 3 раза;
   c. 4 раза.
11. Чему будет равна переменная sum после выполнения фрагмента программы:
    i:=15;
    sum:=0;
    While i>5 do begin
    sum:=sum+i/5;
    i:=i-5
    end;
   a. 5
   b. 6
   c. 30
12. Сколько раз исполнится следующий цикл:
    i:=12;
    Repeat
     i:=i-2
    Until i>4;
   a. 1
   b. 5

c. Бесконечное количество раз

13. Когда применяется оператор с циклом?

a. Когда в алгоритме много раз повторяются одни и теже действия
b. Когда в зависимости от условия мы поступаем так или иначе
c. Когда команды алгоритма идут одна за другой и выполняются по одному разу

14. Установите соответствие:

1. While
2. do
3. for
4. to

Укажите порядок следования вариантов ответа:

a. делать
b. пока
c. для
d. до

15. Когда окончится выполнение цикла

while a<b do
a:=a+1;

a. Когда a станет больше b
b. Когда a станет равно b
c. Цикл не закончится
d. Сразу закончится

Вариант 2

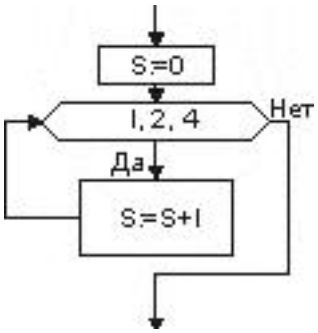
1. Определите значения переменных a и b после выполнения операторов:

a : =1 ; b : =1 ;
While a<=3 Do
a:=a+l;
b:=b+l;

2. Тело цикла в программе

p:=1; a:=1;
while p<60 do a:=2*a; p:=p*a;
выполнится _____ раз.

3. Чему будет равен значение S после выполнения алгоритма.



4. В каком из операторов допущена смысловая ошибка:

a. For i:=1 to 20 do p:=p+1;
b. While s<3 do s:=s-3;
c. Repeat k<1 until k:=k+1;
d. For I:=10 downto 5 do p:=p+1;

5. Сколько раз будет выполняться цикл For i:=7 to 12 do…;

a. 5 раз;

b. 6 раз;
c. 4 раза;
d. 7 раз.

6. Чему будет равна переменная sum после выполнения фрагмента программы:

```
sum:=0;
for i:=7 to 9 do
 sum:=sum+i;
```

a. 15
b. 24
c. 16
d. 23

7. Какой цикл называют циклом с постусловием?

a. Repeat
b. While
c. For

8. Параметр цикла For может получить значения…

a. 9,7,5,3,1,0
b. 7,6,5,4,3,2,1
c. 10,8,6,4,2,0
d. 8,9,7,6,5,3

9.Сколько раз исполнится цикл:

```
i:=6;
while i<18 do i:=i+3;
```

a. 2 раза;
b. 3 раза;
c. 4 раза;
d. 5 раз.

10. Команда while означает

a. Для
b. До тех пор
c. Пока
d. Если

11. Какое число будет выведено на экран:

```
a:=0;
for i:=1 to 5 do
a:=a+2;
write(a);
```

a. 7
b. 10
c. 12
d. 8

12. Что будет напечатано на экране в результате выполнения следующей программы:

```
a:=2;b:=5;
while a<b do
 begin
   writeln(a, ’, ‘,b) ;
   a:=a+3;
end;
```

a. 2, 5
b. 8, 5
c. 3, 5
d. 5, 5

13. Определить значение переменной S после выполнения следующих команд:

```
S:=0; i:=1; repeat s:=s+i*i; i:=i-1; until i<=1;
```

a. 0

b. 55
c. 25
d. 1

14.Begin и and в цикле с предусловием и цикле с параметром можно опустить, когда
a. в теле цикла один оператор;
b. всегда;
c. на усмотрение прогрпммиста;
d. в справочных случаях.

15. Какое из нижеуказанных утверждений верно?
а) выполнение оператора в цикле с условием не зависит от условия;
б) цикл с предусловием может не выполнять тело цикла;
в) цикл While всегда выполняется хотя бы раз;
г) While и Repeat - это циклы с параметром.

Вариант 3

1. Определите значение переменной s после выполнения следующих операторов:
a) s:=0; i:=0; While i<5 Do Inc(i); s:=s + 100 Div i;
b) s:=0; i:=0; While i<5 Do Begin inc(i);s:=s+100 Div i End;
c) b) s:=0; i:=1; While i<1 Do Begin inc(i);s:=s+100 Div i End;

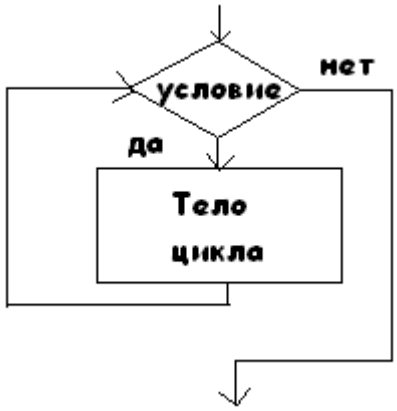
2. Тело цикла в программе

```
m:=36; n:=56;
While (m<>0)or(n<>0) Do
    If m>n then m:=m mod n Else n:=n mod m;
```
выполнится _____ раз.

3. Сколько раз исполнится следующий цикл:
```
i:=21;
Repeat
 i:=i-5
Until i>21
```
a. 1
b. 21
c. Бесконечное количество раз
d. 2

4. Чему будет равна переменная sum после выполнения фрагмента программы:
```
sum:=0;
i:=3;
Repeat
Sum:=sum+3;
i:=i+2
Until i>11;
```
a. 12
b. Цикл бесконечный
c. 15
d. 14

5. Какой цикл изображен на блок схеме?



a. Repeat
b. For
c. While

6. Сколько раз будет выполняться цикл For i:=4 to 8 do…;

a. 5 раз;
b. 6 раз;
c. 4 раза;
d. 7 раз.

7. Требуется подсчитать сумму натуральных чисел от 2 до 22. Какое условие нужно использовать в цикле While?

a. i<23
b. i>22
c. i>=22
d. i<22

8. Оператор Repeat означает

a. Повторение;
b. Для;
c. Пока;
d. Если.

9. В результате выполнения этой программы

```
for i:=1 to n do
write('Привет');
```

на экране появится:

a. Привет
b. Ничего не появится
c. n раз слово Привет
d. 10 раз слово Привет

10. Какая из команд не является командой алгоритма с циклом?

a. while
b. for
c. do
d. else

11. Что будет напечатано на экране в результате выполнения следующей программы:

```
a:=1; b:=5;
while a<b do
begin
  a:=a+2;
  b:=b-1;
end;
  writeln(a,', ',b) ;
```

a. 1, 5
b. 4, 4
c. 3, 4
d. 5, 3

12. Что будет выведено на экран в результате выполнения программы:

```
var i: integer; ss,tt:string;
begin
ss:='krokodil'; tt:='';
for i:=4 to 7 do
  tt:=tt+copy(ss,i,2);
  write(tt); readln;
end.
```

a. kooddiil
b. koddiill
c. kkoodiil
d. kokodiil

13. В каком цикле тело цикла будет выполнено хотя бы один раз?
while условие do тело цикла ;

a. repeat тело цикла until условие ;
b. for параметр:= начальное значение downto конечное значение do тело цикла ;
c. for параметр:= начальное значение to конечное значение do тело цикла.

14. Сколько раз выполниться тело цикла?

```
M:=3;
Repeat
   M: =M+1;
Until M > 6;
```

a) 0 раз
b) 1 раз
c) 3 раза
d) много раз

15. Как записывается цикл с параметром уменьшающимся на 1?

a. while условие do тело цикла ;
b. repeat тело цикла until условие ;
c. for параметр:= начальное значение downto конечное значение do тело цикла ;
d. for параметр:= начальное значение to конечное значение do тело цикла.

Контрольная работа
«Массивы»

Варианты заданий с 1 по 6. Вариант содержит три задачи по одной из каждой части.

Часть 1

1. В массиве X(N) найти максимальный элемент среди положительных элементов массива.
2. В массиве X(N) найти процент положительных элементов и вывести сообщение.
3. В целочисленном массиве X(N) найти max1 – максимальный элемент среди четных элементов массива
4. Найти S1 – сумму четных элементов массива и S2 – сумму положительных элементов массива.
5. В массиве X(N) найти количество элементов, которые меньше значения среднего арифметического элементов массива.
6. Найти номер минимального положительного элемента массива X(N).

Часть 2

Рассматривая строку как одномерный массив, решите задачу

1. Дано предложение. Определить количество букв «н», предшествующих первой запятой, если ее нет, то во всем предложении.

2. Дано предложение. Определить правильно ли в енм записаны буквосочетания «жи» и «ши»

3. Дано предложение. Заменить все символы, расположенные от второй запятой до третьей на «+», если запятых нет, то сообщить об этом.

4. Дано предложение. Определить номера позиций букв «к» в нем.

5. Дано предложение. Удалить из негоодинаковые соседние буквы.

6. Дано слово. Получить новое слово, образованное четными буквами исходного слова.

Часть 3

1. Определить сумму и количество четных чисел расположенных вне диагоналей матрицы B(n,n).

2. Задана матрица A(n,n). Зеркально отразить ее относительно главной диагонали.

3. В каждой строке матрицы F(k,k) элемент, лежащий на главной диагонали, заменить значением минимального элемента всей матрицы.

4. В матрице X(n,n) найти положение максимального элемента в каждой строке.

5. Задана матрица A(n,n). Первый элемент каждого четного столбца заменить средним арифметическим элементов матрицы.

6. Задана матрица A(n,m). Обнулить те строки, где находится максимальный элемент матрицы.

Контрольная работа
«Массивы»

Варианты заданий с 1 по 6. Вариант содержит три задачи по одной из каждой части.

Часть 1

7. В массиве X(N) найти максимальный элемент среди положительных элементов массива.

8. В массиве X(N) найти процент положительных элементов и вывести сообщение.

9. В целочисленном массиве X(N) найти max1 – максимальный элемент среди четных элементов массива

10. Найти S1 – сумму четных элементов массива и S2 – сумму положительных элементов массива.

11. В массиве X(N) найти количество элементов, которые меньше значения среднего арифметического элементов массива.

12. Найти номер минимального положительного элемента массива X(N).

Часть 2

Рассматривая строку как одномерный массив, решите задачу

7. Дано предложение. Определить количество букв «н», предшествующих первой запятой, если ее нет, то во всем предложении.

8. Дано предложение. Определить правильно ли в енм записаны буквосочетания «жи» и «ши»

9. Дано предложение. Заменить все символы, расположенные от второй запятой до третьей на «+», если запятых нет, то сообщить об этом.

10. Дано предложение. Определить номера позиций букв «к» в нем.

11. Дано предложение. Удалить из негоодинаковые соседние буквы.

12. Дано слово. Получить новое слово, образованное четными буквами исходного слова.

Часть 3

7. Определить сумму и количество четных чисел расположенных вне диагоналей матрицы B(n,n).

8. Задана матрица A(n,n). Зеркально отразить ее относительно главной диагонали.

9. В каждой строке матрицы F(k,k) элемент, лежащий на главной диагонали, заменить значением минимального элемента всей матрицы.

10. В матрице X(n,n) найти положение максимального элемента в каждой строке.

11. Задана матрица A(n,n). Первый элемент каждого четного столбца заменить средним арифметическим элементов матрицы.

12. Задана матрица A(n,m). Обнулить те строки, где находится максимальный элемент матрицы.

Контрольная работа №5

«Объектно-ориентированное программирование»

Вариант 1		**Вариант 2**
Вопрос 1 Дать понятие ООП *Вопрос 2* Описание (абстракция), которое показывает, как построить существующую во времени и пространстве переменную…. *Вопрос 3* Это свойство, которое позволяет отличить объект от других объектов того же или других классов *Вопрос 4* Инкапсуляция – это… *Вопрос 5* Что позволяет создавать новые объекты, изменяя или дополняя свойства прежних? *Вопрос 6* Когда вы строите новый класс, наследуя его из существующего класса, можно:		*Вопрос 1* Раскрыть понятие ООП *Вопрос 2* Осязаемая сущность, которая четко проявляет свое поведение, имеет состояние, поведение и паспорт (средство для его однозначной идентификации) – это... *Вопрос 3* … объединяет все его поля данных (статический компонент, т.е. неизменный) и текущие значения каждого из этих полей (динамический компонент, т.е. обычно изменяющийся). *Вопрос 4* Объект характеризуется как совокупностью всех своих свойств и их текущих значений, так и совокупностью

- - - *Вопрос 7* Свойство родственных объектов (т.е. объектов, имеющих одного общего родителя) решать схожие по смыслу проблемы разными способами? *Вопрос 8* В чем заключается понятие перекрытие метода? *Вопрос 9* Дать понятие статистического метода. *Вопрос 10* К каким методам или свойствам можно получить доступ либо из содержащего их класса, либо из его подкласса. Никакому внешнему коду такой доступ не предоставляется?		допустимых для этого объекта действий. Указанное объединение в едином объекте как «материальных» составных частей, так и действий, манипулирующих этими частями называется… *Вопрос 5* Наследование – это… *Вопрос 6* Когда вы строите новый класс, наследуя его из существующего класса, можно: - - - *Вопрос 7* Назначение полиморфизма *Вопрос 8* В чем заключается понятие перекрытие метода? *Вопрос 9* Дать понятие виртуального метода. *Вопрос 10* Общедоступным свойством или методом называется?

ГАОУ СПО Стерлитамакский колледж строительства, экономик и права

Комплект

контрольно-измерительных материалов

по учебной дисциплине

«Основы программирования»

2 семестр

основной профессиональной образовательной программы (ОПОП)

по специальности (специальностям) СПО

230115 Программирование в компьютерных системах

базовой подготовки

Стерлитамак, 2012

Содержание

1. Паспорт комплекта контрольно- измерительных материалов

1.1. Область применения

Комплект контрольно-измерительных материалов предназначен для проверки результатов освоения учебной дисциплины «Основы программирования» (далее УД), относящейся к общепрофессиональному циклу основной профессиональной образовательной программы (далее ОПОП) по специальности СПО 230115 Программирование в компьютерных системах

Комплект контрольно-измерительных материалов позволяет оценивать освоение умений и усвоение знаний:

Освоенные умения, усвоенные знания	**Показатели оценки результата**	**№№ заданий для проверки**
1	**2**	**3**
В результате изучения учебной дисциплины обучающийся должен **знать**:		
этапы решения задачи на компьютере	Перечисляет этапы решения задачи на компьютере	
типы данных	Формулирует типы данных	**1.1-1.2**
базовые конструкции изучаемых языков программирования	Воспроизводит базовые конструкции изучаемых языков программирования	**1.3-1.5**
принципы структурного и модульного программирования	Формулирует принципы структурного программирования	**1.6-1.8 2.1-2.2**
	Формулирует принципы модульного программирования	
Знать принципы объектно-ориентированного программирования	Формулирует принципы объектно-ориентированного программирования	**2.3**
В результате изучения учебной дисциплины обучающийся должен **уметь:**		
реализовывать алгоритмы в виде программ на конкретном языке программирования	. Программа, построенная на основе простейших алгоритмов, написана в соответствии с правилами программирования и соответствует поставленной задаче	**3.1**

1.2. Система контроля и оценки освоения программы УД

Система контроля и оценки освоения программы УД соответствует положению об итоговой и промежуточной аттестации в ГАОУ СПО СКСЭиП

1.2.1. Организация контроля и оценки освоения программы УД

Контроль и оценка освоения программы УД осуществляется в форме:
- текущего контроля:
на занятиях в форме устных, письменных опросов и проверочных работ;
практических работ за ПК;
- промежуточного контроля:
ежемесячная аттестация по текущим оценкам, по изученным темам «Интерфейс системы программирования», «Основные алгоритмические конструкции», «Массивы», «Подпрограммы», «Реализация ООП» контрольная работа;
- рубежный контроль
По итогам первого семестра накопительная оценка;
- итогового контроля:
итоговый контроль проводится на экзамене во втором семестре изучения дисциплины;
Условием допуска к экзамену является положительная ежемесячная аттестация по курсу дисциплины, положительная накопительная оценка за первый семестр изучения.

Оценка освоения **знаний** осуществляется с помощью:

1) заданий с выбором ответа из фиксированного набора вариантов;
2) открытого теста со свободным кратким ответом.

Оценка освоенных **умений** осуществляется с помощью практического задания открытого типа с развернутым ответом.

2. Комплект материалов для оценки сформированности умений и знаний

2.1. Инструкция для обучающегося

Экзаменационная работа состоит из трех частей, которые различаются по содержанию, сложности и числу заданий. Определяющим признаком каждой части работы является форма заданий:

- часть 1 содержит задания с вариантами ответов;
- часть 2 содержит задания с кратким ответом;
- часть 3 содержит задания с развернутым ответом.

Часть 1 включает 8 вопросов.

Задания с выбором ответа из списка предложенных ответов направлены на проверку усвоения теоретической части изученного курса. В них необходимо выбрать букву правильного ответа в зависимости от формулировки задания и внести в бланк ответов.

Часть 2 включает 3 вопроса.

Задания с кратким ответом, предназначены для проверки базовых умений и знаний обучающегося по изученной дисциплине. В них необходимо дать краткий ответ словом, словосочетанием или числом в виде обыкновенной дроби. Задание с кратким ответом считается выполненным, если верный ответ зафиксирован в бланке ответов в той форме, которая предусмотрена инструкцией по выполнению задания.

Часть 3 включает 1задачу с развернутым ответом.

При выполнении заданий с развернутым ответом в бланке ответов должна быть записана программа решения с пояснениями.

Для экономии времени пропускайте задание, которое не удаётся выполнить сразу, и переходите к следующему. Если после выполнения всей работы у Вас останется время, Вы сможете вернуться к пропущенным заданиям. Баллы, полученные Вами за выполненные задания, суммируются. Постарайтесь выполнить как можно больше заданий и набрать наибольшее количество.

Условия выполнения заданий

При выполнении практического задания можно воспользоваться компьютером и черновиком.

Справочные материалы не требуются.

Оцениваемые умения и знания**:** уметь писать программу на изученном языке программирования, использовать основные алгоритмические конструкции, массивы, подпрограммы, модули и основы ООП для решения поставленной задачи, выполнять отладку и редактирование программного кода.

Место проведения: полигон вычислительной техники

Используемое оборудование: компьютер

Максимальное время выполнения задания.

На выполнение письменной работы отводится 2 часа (120 минут). Рекомендуемое время выполнения каждого задания:

для каждого задания 1 части – 2 минуты;

для каждого задания 2 части – 4-5 минут;

для каждого задания 3 части – до 60 минут.

Желаем успеха!

2.2 Задания для оценки освоения умений и усвоения знаний

ПРИЛОЖЕНИЕ В

Бланк для внесения ответов ПРИЛОЖЕНИЕ А

3. Пакет для эксперта

3.1 Инструкция для эксперта

Количество вариантов заданий для обучающихся, сдающих экзамен: 6.

Каждый вариант состоит из трёх частей и включает 13 заданий. Одинаковые по форме представления и уровню сложности задания сгруппированы в определённые части работы.

Часть 1 содержит 8 заданий с выбором ответа, базового уровня сложности. Их обозначение в работе: 1.1, 1.2,…1.8, предназначены для проверки знаний.

Часть 2 содержит 4 задания с кратким ответом, повышенного уровня сложности. Их обозначение в работе: 2.1, 2.2, 2.3, предназначены для проверки знаний и умений.

Часть 3 содержит 1 задачу с развёрнутым ответом и предназначена для проверки умений.

Время выполнения каждого задания:

Для каждого задания 1 части – 2-3 минуты;

Для каждого задания 2 части – 4-5 минут;

Для задания 3 части – до 60 минут;

Общая продолжительность выполнения заданий на экзамене составляет 120 минут.

Условия выполнения заданий

При выполнении практического задания можно воспользоваться компьютером и черновиком.

Справочные материалы не требуются.

Оцениваемые умения и знания**:** уметь писать программу на изученном языке программирования, использовать основные алгоритмические конструкции, массивы, подпрограммы, модули и основы ООП для решения поставленной задачи, выполнять отладку и редактирование программного кода.

Место проведения: полигон вычислительной техники

Используемое оборудование: компьютер

Рекомендации по проведению оценки**:**

1. Ознакомьтесь с заданиями для студентов, сдающих экзамен, оцениваемыми знаниями и умениями, показателями оценки
2. Создайте доброжелательную обстановку, но не вмешивайтесь в ход (технику) выполнения задания.
3. Соберите выполненные задания через 120 минут после начала выполнения и проверьте правильность выполнения задания.

3.1. Ответы на задания части 1 и части 2 проверяются сопоставлением с ключом, ответы на задания части 3 проверяются сопоставлением с модельным ответом.

3.2. Верное выполнение каждого задания части 1 оценивается 1 баллом. За выполнение задания ставится 0 баллов, если:

а) указан неправильный ответ;

б) указаны 2 или несколько ответов, среди которых может быть и правильный;

в) ответ отсутствует.

3.3. В части 2 верное выполнение заданий оценивается 2 баллами. Ставится 1 балл, если в ответе допущена 1 ошибка. Ставится 0 баллов, если: а) в ответе допущено более 1 ошибки; б) ответ отсутствует.

3.4. Задания части 3 (с развёрнутым ответом) предусматривают проверку конечных шагов решения задачи и оцениваются по набранным баллам в соответствии с модельным ответом.

4. Суммируйте баллы, полученные обучающимся за верно выполненные задания.
5. Поставьте оценку, руководствуясь следующей шкалой:

Сумма баллов	% выполнения заданий	Оценка
16-18 баллов	91-100%	Отлично
13-15 баллов	76-90%	Хорошо
10-12 баллов	61-75%	удовлетворительно
1-9 баллов	Менее 60%	неудовлетворительно

6. Перенесите № вариантов, набранные баллы обучающимся и выставленные им оценки в оценочную ведомость.

ПАКЕТ ЭКЗАМЕНАТОРА

Показатели оценки результатов освоения программы учебной дисциплины

Номер задания	**Оцениваемые умения и знания**	**Показатели оценки результата (требования к выполнению задания)**
1.1 — 1.2 Закрытый тест с выбором одного варианта из фиксированного набора вариантов	Знание этапов решения задачи на компьютере, типы данных	Соответствие ответа правилам решения задачи на ПК и правилам работы с типами данных
1.2 — 1.5 Закрытый тест с выбором одного варианта из фиксированного набора вариантов	Знание базовых конструкции изучаемых языков программирования	Соответствие ответа правилам использования базовых конструкций языка программирования
1.6 — 1.8 Закрытый тест с выбором одного варианта из фиксированного набора вариантов	Знание принципов структурного и модульного программирования	Соответствие ответа принципам структурного и модульного программирования
2.1-2.2 Открытый тест со свободным кратким ответом	Знание принципов структурного и модульного программирования	Соответствие ответа правилам составления программ в соответствии с принципами структурного и модульного программирования
2.3 Открытый тест со свободным кратким ответом	Знание принципов объектно-ориентированного программирования	Основные понятии ООП указаны верно.
3.1. Практическое задание	Умение реализовывать алгоритмы в виде программ на конкретном языке программирования	Программа написана в соответствии с правилами программирования и решает поставленную задачу..

Рекомендации по проведению оценки**:**

1. Ознакомьтесь с заданиями для экзаменующихся, оцениваемыми умениями и знаниями и показателями оценки.
2. Оцените результаты выполнения заданий
   - Части 1 и 2 сопоставлением с ключом,
   - Части 3 сопоставлением с модельным ответом.

Ключ к тексту и модельные ответы ПРИЛОЖЕНИЕ Б

3.2. Оценочная ведомость

Министерство образования РБ

ГАОУ СПО Стерлитамакский колледж строительства, экономики и права

Оценочная ведомость

для итогового контроля в форме экзамена
по учебной дисциплине **«Основы программирования»** обучающихся в группе 2 курса _____
по специальности 230115 «Программирование в компьютерных сетях» базового уровня подготовки

№ п/п	№ вариан-та	Фамилия, имя, отчество обучающегося	Количество набранных баллов	Оценка
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				

Дата проведения экзамена «____»_____20 __года
Время, отведённое на проведение экзамена 120 мин.
Эксперт_____(_____) Преподаватель _____ (_____)
Подпись *ФИО* *Подпись* *ФИО*

ПРИЛОЖЕНИЕ А. Бланки ответов для экзаменующихся

Бланк ответов по учебной дисциплине «Основы программирования» Вариант № __ ФИО студента_____ Группа _____ Дата выполнения_____
Часть 1 *Напишите букву выбранного ответа напротив номера вопроса*
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
ЧАСТЬ 2 *Напишите краткий ответ напротив номера вопроса*
2.1
2.2
2.3
ЧАСТЬ 3 *Напишите программу для решения поставленной задачи с записью выводимого результата*
3.1

ПРИЛОЖЕНИЕ Б. Ключи, модельные ответы для экзаменатора

задание\вариант	Вариант 1	Вариант 2	Вариант 3	Вариант 4	Вариант 5	Вариант 6
Задание 1						
5.	в	б	в	а	б	б
6.	а	а	а	а	а	а
7.	а	а	а	б	б	г
8.	г	а	а	г	а	г
9.	г	б	в	а	г	а
10.	г	в	а	г	в	а
11.	а	б	г	а	б	а
12.	б	а	а	а	а	б
Задание 2						

2.1.	Begin
	For i:=1 To n Do
	For j:=1 To m Do
	x[i,j]:=-25+Random(51);
	End;
		Begin
		For i:=1 To n Do
		Begin {ввод i-ой строки массива}
		For j:=1 To n Do
		Write(x[i,j]);
		Writeln; {переход на начало следующей строки}End;
			min:=mas[1]
			for j:=2 to n do
			begin
			if mas[i]<min then min:=mas[i];end;
			writeln(min);
				for I:=l to n do
				begin
				write (T[I])
				end;
					begin
					sum := 0;
					for i := 1 to n do
					
					Inc(sum, a[i,n+1-i]);
					end;
						Метод пузырька, поиска наименьшего, сортировка вставками

2.2.	Формальные параметры подпрограммы указывают, с какими параметрами следует обращаться к этой подпрограмме (количество параметров, их последовательность, типы). Они задаются в заголовке подпрограммы в виде списка формальных параметров, разбитого на группы, разделенные точками с запятыми
	При обращении к подпрограмме формальные параметры заменяются на соответствующие факти-
		1) функция выдает 1 значение, процедура может и больше;
		2) вызываем процедуру в теле программы как обычный оператор, имя функции всегда присваивается переменной
			procedure exchange (var a,b: integer);
			var c: integer;
			begin
			if a > b then begin
			c := a;
			a := b;
			b := c;
			end;
			end;
				В процедуре это переменные записанные после var в списке параметров процедуры, а в функции это имя функции	идентификаторы, введенные внутри процедуры, функции для описания переменных, констант, типов, процедур, наз локальными для данного блока. Константы, переменные, типы, описанные в блоке program, называются глобальными.	Все формальные параметры можно разбить на три категории: параметры-значения; параметры-переменные; параметры-процедуры и параметры-функции.

	ческие вызывающей программы или подпрограммы					
2.3.	К общедоступным свойствам и методам можно получать доступ из любого контекста.	позволяет использовать одни и те же функции для решения разных задач. Полиморфизм выражается в том, что под одним именем скрываются различные действия, содержание которых зависит от типа объекта	Это такое отношение между объектами, когда один объект повторяет структуру и поведение другого	Класс	Предполагает соединение в одном объекте данных и функций, которые манипулируют этими данными	Изменяя алгоритм того или иного метода в потомках объекта, программист может придавать этим потомкам отсутствующие у родителя специфические свойства. Для изменения метода необходимо *перекрыть* его в потомке, т.е. объявить в потомке одноименный метод и реализовать в нем нужные действия.

№ Варианта	Ответ 3.1.

1.	type
	mass = array[1..15, 1..15] of integer;
	var i,j,n:integer; mas: mass;
	function summa(K: integer; t: mass):integer;
	var s:integer;
	begin
	s:=0;
	For i:=1 to k do
	s:=s+t[i,i];
	summa:=s;
	end;
	
	begin
	Write('n = ');
	Readln(n);
	Randomize;
	For i:=1 to n do
	For j:=1 to n do
	mas[i,j]:=Random(3);
	For i:=1 to n do begin
	For j:=1 to n do
	write (mas[i,j]);
	writeln;
	end;
	Writeln(summa(n,mas));
	Readln;
	end.

2.	 program sortirovka; uses crt; var y : array[1..50] of integer; i,n,b:integer; begin writeln (‘введите количество элементов массива’);readln (n);

	Randomize;
	for i:=l to n do
	y[i]):=Random(2);
	for i:=l to n do
	write (y [i],' ');
	
	for j:=l to n-1 do
	for i:=l to n-j do
	if y[i] > y[i + l] then
	begin
	b:=y[i];
	y[i]:=y[i+l];
	y[i+l];=b;
	end;
	for i:=l to n do
	write (y[i],' ');
	end.

3.	Procedure KORNI (a,b,c: real, var x1,x2: real, var s: char);
	var d: real;
	begin
	d:=sqr(b)-4*a*c;
	if d>=0 then begin
	x1:= (-b+Sqrt(d))/(2*a);
	x2:= (-b-Sqrt(d))/(2*a);
	s:= ‘’;
	end
	else s:= ‘n’;
	end;
	var n,m,k,x1,x2:real; f:char;
	begin
	writeln('введите коэффициенты уравнения a,b,c'); read(n,m,k);
	korni(n,m,k,x1,x2,f);
	if f='n' then writeln('действительных корней нет') else writeln('x1=',x1:3:3,' x2=',x2:3:3);
	readln;
	end.

4.	var a: integer; n: byte; rez: longint;
	Function stepen(x, y : integer) : longint;
	Var St : LongInt; I : byte;
	Begin
	St := 1;
	For I := 1 To y Do St := St * x;
	stepen := St;
	End;
	begin
	writeln('введите число a'); read(a);
	writeln('введите степень n'); read(n);
	rez:= stepen (a,n);
	readln;
	end.

5.	Function NOD (m,n: word): word;
	var m,n:integer;
	prod:char;
	begin
	write('M= '); readln(m);
	write('N= '); readln(n);
	repeat
	if n>m then n:=n mod m else m:=m mod n;
	until (m=0)or(n=0);

	NOD: =m+n; End;

6.	uses crt;
	var a,b,k1,k2:integer;
	function stepen(m, n:integer):integer;
	var i, rez:integer;
	begin
	rez:=1;
	for i:=1 to n do
	rez:=rez*m;
	stepen:=rez;
	end;
	begin
	clrscr;
	writeln('введите степени k1, k2');
	readln(k1,k2);
	writeln('введите числа b,a');
	readln(b,a);
	writeln(b,' в степени ',k1,' + ',a+4,' в степени ',k2,' = ', stepen(a+4,k2)+ stepen(b,k1));
	readln;
	end.

Макс балл	**4**
ОШИБКИ	Не записан ответ после выполнения программы -1 Присутствуют недочеты: неверно записан(ы) оператор(ы), не доопределены переменные -2 Программа не записана. Записана программа, не соответствующая задаче 0

ПРИЛОЖЕНИЕ В Задания для экзаменующихся

Вариант № 1
ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Выполнение цикла заканчивается:

```
while a<b do
a:=a+1;
```

(1) Когда a станет больше b;

(2) Когда a станет равно b;

(3) Цикл не закончится;

(4) Сразу закончится.

1.2. .Алгоритм какого типа выполняет группу операторов в зависимости от условия

(1) циклический;

(2) разветвляющийся;

(3) вспомогательный;

(4) линейный;

1.3. В приведенном фрагменте программы (*N* типа LongInt, *N* > 0)

```
P := 1;
While P <= N Do
Begin
        Left := N Div (P * 10) * (P * 10);
        Right := N Mod P;
        K := ((N Mod (P * 10) Div P + 1) Mod 10) * P;
        N := Left + K + Right; P := P * 10
  End;
```

натуральное число *N* изменяется по следующему правилу

(1) в каждый разряд прибавляется 1;

(2) из каждого разряда вычитается 1;

(3) в каждый разряд прибавляется 1, если значение в разряде — не девять, иначе заменяется на нуль;

(4) каждая девятка в десятичной записи числа заменяется на нуль.

1.4. Требуется подсчитать сумму натуральных чисел от 2 до 22. Условие используемое в цикле While

(1) i<23;

(2) i>22;

(3) i>=22;

(4) i=23.

1.5. Операции div и mod используются для переменных типа:

(1) Char;

(2) Integer;

(3) String;

(4) Real.

1.6. Для прекращения выполнения тела цикла в текущей итерации и перехода к началу новой итерации цикла используют оператор

(1) Go to;

(2) Exit;

(3) Continue;

(4) Break.

1.7. Задан двумерный массив X[1..n, 1..m]. Функция

```
Function Check (X: Myarray): Boolean;
Var i, j : Integer; t : Boolean;
Begin   t := True; i := 1;
   While t And (i <= n) Do
Begin j := 1; While (j <= m) And (X[i, j] <> 0) Do Inc (j);
              t := (j = m + 1); Inc (i)
  End;
Check := Not t
End;
```

возвращает значение

(1) True, если все элементы массива ненулевые;

(2) True, если в массиве есть элемент, равный нулю;

(3) False, если в массиве есть элемент, равный нулю;

(4) Not t.

1.8. Отличие процедуры от функции при описании заключается в следующем:

(1) У функции должен обязательно быть указан ее тип;

(2) У процедуры должен обязательно быть указан ее тип;

(3) В количестве формальных переменных;

(4) Работа с разными типами переменных.

ЧАСТЬ 2

В бланк ответов напишите ответ напротив номера вопроса

- Запишите программный блок заполнения двумерного массива случайными числами_
- Описать формальные и фактические переменные_____
- Общедоступным свойством или методом называется _____

ЧАСТЬ 3

Напишите программу для решения поставленной задачи с записью выводимого результата

Написать функцию вычисления суммы элементов матрицы на главной диагонали. Вывести матрицу и вычисленную сумму на экран

ЗАДАНИЯ ДЛЯ ЭКЗАМЕНУЮЩИХСЯ

Вариант № 2

ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Каждое выражение(каждый оператор) в программе отделяется друг от друга

(1) точкой с запятой;

(2) двоеточием;

(3) новой строкой;

(4) тире.

1.2. Программа - это

(1) Набор команд на языке программирования;

(2) Алгоритм решения задачи, записанный на языке программирования;

(3) Машинные коды;

(4) Набор команд, выполняющих какие-то действия.

1.3. Циклический алгоритм с известным числом повторений:

(1) For i = a to b do

(2) While условие do

(3) Repeat..until условие

(4) Case переменная of

1.4. Сколько раз исполнится цикл: *i:=6; while i<18 do i:=i+3;*

(1) 2 раза;

(2) 3 раза;

(3) 4 раза;

(4) Ни одного.

1.5. Фрагмент программы, написанный правильно:

(1) *Repeat* s:=s+a;

until s > 10;

(2) *Repeat*: begin s:=s+a; end;

until s > 10;

(3) *Repeat*: s:=s+a

until s > 10;

(4) *Repeat* begin s:=s+a *end*

until s > 10;

1.6. Задан двумерный массив X[1..n, 1..m]. Процедура

```
Procedure Sub (Var X: Myarray);
Var i, j: Integer;
Begin For i := 1 To n Do
        For j := 1 To m Div 2 Do X[i, 2 * j] := X[i, 2 * j] + X[i, 1];
End;
```

(1) к элементам столбцов в первой половине матрицы прибавляет элементы первого столбца соответствующей строки;

(2) добавляет к матрице еще M столбцов с элементами, равными соответствующим элементам первого столбца;

(3) к элементам четных столбцов прибавляет элементы первого столбца соответствующей строки;

(4) к элементам четных строк прибавляет элементы первой строки соответствующего столбца.

1.7. Выберите правильную запись для создания пользовательского типа данных , описывающего массив одномерный:

(1) Type mass =Array[1.. 5] Of real;

(2) Type mass :Array[1.. 5] Of real;

(3) Type mass =Array[1.. 5] ;

(4) Type mass =Array Of real;

1.8. Локальными по отношению к подпрограмме называются переменные, которые :

(1) описаны как аргумент подпрограммы;

(2) изменяются в основной программе;

(3) описаны перед подпрограммой;

(4) описаны после заголовка подпрограммы.

ЧАСТЬ 2

<u>В бланк ответов напишите ответ напротив номера вопроса</u>

2.1. Запишите блок вывода двумерного массива на экран в виде таблицы _____

2.2. Перечислите различия между процедурой и функцией _____

2.3. Назначение полиморфизма _____

ЧАСТЬ 3

<u>Напишите программу для решеия поставленной задачи с записью выводимого результата</u>

Напишите программу на языке Паскаль. Задан массив A (n). Отсортировать его по возрастанию (любым методом).

ЗАДАНИЯ ДЛЯ ЭКЗАМЕНУЮЩИХСЯ

Вариант № 3

ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Итерационный цикл с истинным условием выхода из него:

(1) For i = a to b do

(2) While условие do

(3) Repeat..until условие

(4) Case переменная of

1.2. **Сколько раз исполнится следующий цикл:**

```
i:=12;
Repeat
 i:=i-2
Until i>4;
```

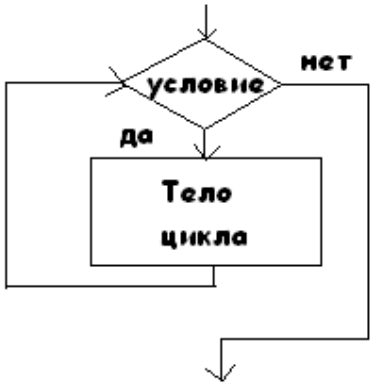
(1) 1;

(2) 5;

(3) Бесконечное количество раз;

(4) 4.

1.3. На предложенной блок схеме изображен цикл



(1) Repeat;

(2) For ;

(3) While;

(4) Case.

1.4. Вещественный тип данных:

(1) String;

(2) Char;

(3) Integer;

(4) Real.

1.5. Задан линейный массив M[1..n].

```
Function Control (M: Myarray): Boolean;
Var I : Integer;
Begin I := 1;
        While (I <= n) And (M[I] > 0) Do Inc(I);
        Control := (I <= n);
End;
```

Если в данном массиве все элементы положительные, приведенная функция возвращает значение

(1) *n*;

(2) True;

(3) False;

(4) *I* <= *n*;

1.6. Переход за последний оператор процедуры осуществляется оператором

(1) Go to;

(2) Exit;

(3) Continue;

(4) Break.

1.7. Укажите способ заполнения массива в следующем фрагмене программы

```
Program M4;
Cons
N=40;
Var
A : Array [1..N] Of Integer; I : Integer;
Begin
For I:=1 To N Do
Begin
A[I]:= Trunc(Random*101)-50
Write(A[I],'VVV')
End
End.
```

(1) Вручную с клавиатуры;

(2) По формуле;

(3) Массив не заполнен;

(4) Автоматически.

1.8. Какое из ниже приведенных описаний процедур верно?

(1) procedure Input(x:real;): integer;

(2) procedure Input(x,y; z:real);

(3) procedure Input(x,y:integer;z:integer);

(4) procedure Input(x,y:integer;z:array [1..3] of integer);

ЧАСТЬ 2

В бланк ответов напишите ответ напротив номера вопроса

2.1. Запишите фрагмент программы, выполняющей поиск минимального элемента в одномерном массиве _____

2.2. Запишите процедуру, выполняющую обмен значениями двух чисел _____

2.3. Наследование – это _____

ЧАСТЬ 3

Напишите программу для решения поставленной задачи с записью выводимого результата

Напишите подпрограмму на языке Паскаль для вычисления корней квадратного уравнения $ax^2+bx+c=0$ или проверки их отсутствия

ЗАДАНИЯ ДЛЯ ЭКЗАМЕНУЮЩИХСЯ

Вариант № 4

ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Итерационный цикл с ложным условием выхода из него:

(1) For i = a to b do

(2) While условие do

(3) Repeat..until условие

(4) Case переменная of

1.2. Чему будет равна переменная sum после выполнения фрагмента программы:

```
sum:=0;
for i:=5 to 8 do
 sum:=sum+i;
```

(1) 18;

(2) 13;

(3) 8;

(4) 26.

1.3. Оператор нахождения остатка от деления двух чисел:

(1) Div;

(2) Writeln;

(3) Mod;

(4) Crt.

1.4. Цикл с предусловием выполняется так:

(1) выполняется тело цикла, изменяется параметр цикла, проверяется условие продолжения выполнения цикла;

(2) изменяется параметр цикла, проверяется условие продолжения выполнения цикла, выполняется тело цикла;

(3) проверяется условие продолжения выполнения цикла, выполняется тело цикла; 4) тело цикла выполняется *N* раз (*N* — натуральное);

(4) определяется, сколько раз должен быть выполнен цикл, и далее цикл с предусловием сводится к циклу с параметром.

1.5. Значения переменных *p* и *d* после выполнения фрагмента алгоритма

```
k := 47; Case k Mod 9 Of
            5: Begin d := k; p := True End;
            0..2: Begin d := 2; p := False End;
            8: Begin d := 1; p := False End
            Else Begin d := 1; p := True End
    End;
```

Равны

(1) P = True, d = 1;

(2) p = False, d = 2;

(3) p = False, d = 3;

(4) p = True, d = 2.

1.6. Элементы массива p[1..5] равны соответственно 1, –1, 5, 2, 4. Значение выражения *p[1] * p[3] – p[2 * p[2] + p[p[5] – p[2]]]*

равно

(1) 8;

(2) 12;

(3) –12;

(4) 6.

1.7. Предложенный алгоритм определяет для массива X[1..n]

```
P:= 0;
For k:= n downto 1 do
If  X[k] < >T  then  P:=k;
```

(1) количество элементов массива, не равных T;

(2) номер последнего элемента массива, равного T;

(3) номер последнего элемента массива, не равного T;

(4) номер первого элемента массива, не равного T.

1.8. Пусть дан заголовок функции:

function f (a:integer; b:real; c: char):real;

верный вызовов функций:

(1) R = f (4.5, 7, ‘r’);

(2) R = f (3, 7.89, ‘5’);

(3) R = f (15, ‘*’);

(4) R = f (53, 3, ‘A’).

ЧАСТЬ 2

В бланк ответов напишите ответ напротив номера вопроса

2.1. Запишите блок вывода одномерного массива на экран в строку _____

2.2. Запишите как обозначается переменная, возвращаемая в основную программу в процедуре и в функции _____

2.3. Описание (абстракция), которое показывает, как построить существующую во времени и пространстве переменную….

ЧАСТЬ 3

Напишите программу для решения поставленной задачи с записью выводимого результата

Написать функцию, вычисляющую x^y для целых чисел (используя умножение).

ЗАДАНИЯ ДЛЯ ЭКЗАМЕНУЮЩИХСЯ

Вариант № 5

ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Из перечисленных ниже в программе обязателен раздел

(1) Var;

(2) Const;

(3) Type;

(4) Label;

(5) Begin … End.

1.2. Выражение вида k*exp(ln(n)) вычисляет в среде Паскаль:

(1) степень n числа k;

(2) степень n числа;

(3) произведение k*n;

(4) экспонету от суммы k и n.

1.3. Оператор, позволяющий выполнить выбор из нескольних вариантов:

(1) For i = a to b do

(2) While условие do

(3) Repeat..until условие

(4) Case переменная of

1.4. Переменная sum после выполнения фрагмента программы будет равна:

```
sum:=0;
i:=1;
Repeat
Sum:=sum+3;
i:=i-1
Until i>11;
```

(1) 3;

(2) Цикл бесконечный;

(3) 6;

(4) 0.

1.5. Считать с клавиатуры значение переменной a:

(1) a: integer;

(2) var a;

(3) writeln(a);

(4) readln(a).

1.6. Фрагмент программы вида:

```
a='in';
```

```
b='out';

k:= length (b) - length (a);

for i:=1 to k do if (a[i]<b[i]) then k:=k+1;

write (k);
```

выведет на экран значение k, равное ...

(1) 1;

(2) 2;

(3) 3;

(4) 4.

1.7. Следующий алгоритм для массива X[1..n, 1..m] определяет

```
S:= 0;
For i:= 1 to n do
For  j:= 1 to m do
If  X[i, j ] <0  then S:=S+X[i, j ];
```

(1) минимальный элемент массива;

(2) сумму отрицательных элементов массива;

(3) максимальный элемент массива;

(4) количество отрицательных элементов массива.

1.8. Процедура – это подпрограмма:

(1) Служит для задания совокупности действий, направленных на изменение внутренних по отношению к ним программой обстановки;

(2) Служат для задания совокупности действий, направленных на изменение внешней по отношению к ним программой обстановки;

(3) Служат для определения алгоритма вычисления нового значения некоторого простого или ссылочного типа;

(4) Заключается в том, чтобы определить алгоритм вычисления нового значения некоторого простого или ссылочного типа.

ЧАСТЬ 2

В бланк ответов напишите ответ напротив номера вопроса

2.1. Запишите фрагмент программы, вычисляющей сумму элементов побочной диагонали двумерного массива_____

2.2. Локальные и глобальные переменные: где описываются и как работают_____

2.3. Инкапсуляция – это _____

ЧАСТЬ 3

Напишите программу для решения поставленной задачи с записью выводимого результата

Написать функцию нахождения наибольшего общего делителя (НОД) двух неотрицательных целых чисел.

ЗАДАНИЯ ДЛЯ ЭКЗАМЕНУЮЩИХСЯ

Вариант № 6

ЗАДАНИЯ

ЧАСТЬ 1

В бланк ответов напишите букву выбранного ответа напротив номера вопроса

1.1. Оператор выполняющий выбор действий из истинности и ложности условия:

(1) If условие then ... else...

(2) While условие do

(3) Repeat..until условие

(4) Case переменная of

1.2. Выделение части программы и оформление ее специальным образом для дальнейшего обращения к ней из других программ является принципом:

(1) Структурного программирования;

(2) Модульного программирования;

(3) Объектно-ориентированного программирования;

(4) Обычного программирования.

1.3. Синтаксические и логические ошибки в программе устраняются на этапе

(1) Алгоритмизация вычислительного процесса;

(2) Отладка программы;

(3) Составление программы;

(4) Решение задачи на ЭВМ и анализ результатов.

1.4. Типы параметров, используемых в подпрограммах

(1) Вспомогательные, основные;

(2) Типовые, порядковые;

(3) Фактические, формальные;

(4) Базовые, пользовательские;

1.5. В чем сущность механизма использования подпрограмм?

(1) Использование подпрограмм помогает облегчить процесс написания кода;

(2) Подпрограммы в основном используются для уменьшения исходного кода программы;

(3) Подпрограммы реализуют вспомогательные алгоритмы для дальнейшего их использования в основной программе;

(4) Подпрограмма исползуется для красоты программы.

1.6. Фрагмент программы, написанный правильно:

(1) *While* s < 10 *do*:

s:=s+a;

(2) *While* s < 10 *do*:

begin s:=s+a; end;

(3) *While* s < 10 *do*

s:=s+a;

(4) *While* s < 10 *do*

s:=s+a

1.7. Оператор целочисленного деления:

(1) Readln;

(2) Div;

(3) Var;

(4) Mod.

1.8. Дано следующее описание блока программы:

```
var  x: real;
procedure Sum(x:real);
var  i: integer;
begin
i:=4;  x:=i+2;  Writeln(x);
end;
begin
x:=3;  y:=Sum(x);
end.
```

Ошибки допущенные в данном описании:

(1) В данном блоке нет ошибок;

(2) Неправильное описание процедуры;

(3) Неправильный вызов процедуры;

(4) Неправильное описание и неправильный вызов процедуры.

ЧАСТЬ 2

<u>В бланк ответов напишите ответ напротив номера вопроса</u>

2.1. Перечислите и раскройте суть способов упорядочивания одномерного массива ____

2.2. запишите виды формальных параметров и чем они отличаются друг от друга _____

2.3. Раскройте суть понятия перекрытие метода _____

ЧАСТЬ 3

<u>Напишите программу для решения поставленной задачи с записью выводимого результата</u>

Разработать функцию Stepen (вещественное число в степени k) используя умножение, и использовать эту функцию для вычисления выражений вида: $b^{k1}+(a+4)^{k2}$. То есть – программа запрашивает значения a, b, k1 , k2, вычисляет выражение и выводит на экран.

РАЗДЕЛ 5 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Методические рекомендации для студентов по организации самостоятельной работы

Введение

В современный период востребованы высокий уровень знаний, социальная мобильность, профессионализм специалистов, готовность к самообразованию и самосовершенствованию. В связи с этим должны измениться подходы к планированию, организации учебно-воспитательной работы, в том числе и самостоятельной работы студентов. Прежде всего, это касается изменения характера и содержания учебного процесса, переноса акцента на самостоятельный вид деятельности, который является не просто самоцелью, а средством достижения глубоких и прочных знаний, инструментом формирования у студентов активности и самостоятельности.

Целью методических рекомендаций является повышение эффективности учебного процесса, в том числе благодаря самостоятельной работе, в которой студент становится активным субъектом обучения, что означает:

- способность занимать в обучении активную позицию;
- готовность мобилизовать интеллектуальные и волевые усилия для достижения учебных целей;
- умение проектировать, планировать и прогнозировать учебную деятельность;
- привычку инициировать свою познавательную деятельность на основе внутренней положительной мотивации;
- осознание своих потенциальных учебных возможностей и психологическую готовность составить программу действий по саморазвитию.

Виды самостоятельной работы студентов

Репродуктивная самостоятельная работа	Самостоятельное прочтение, просмотр, конспектирование учебной литературы, прослушивание лекций, магнитофонных записей, заучивание, пересказ, запоминание, Интернет-ресурсы, повторение учебного материала и др.
Познавательно-поисковая самостоятельная работа	Подготовка сообщений, докладов, выступлений на семинарских и практических занятиях, подбор литературы по дисциплинарным проблемам, написание рефератов, контрольных, курсовых работ и др.
Творческая самостоятельная работа	Написание рефератов, научных статей, участие в научно-исследовательской работе, подготовка дипломной работы (проекта). Выполнение специальных заданий и др., участие в студенческой научной конференции. Организация и контроль самостоятельной работы

Для успешного выполнения самостоятельной работы студентов необходимо планирование и контроль со стороны преподавателей.

Аудиторная самостоятельная работа выполняется студентами на лекциях, лабораторных занятиях, и, следовательно, преподаватель должен заранее выстроить систему самостоятельной работы, учитывая все ее формы, цели, отбирая учебную и научную информацию и средства (методических) коммуникаций, продумывая роль студента в этом процессе и свое участие в нем.

Вопросы для самостоятельной работы студентов, указанные в рабочей программе дисциплины, предлагаются преподавателями в начале изучения дисциплины. Студенты имеют право выбирать дополнительно интересующие их темы для самостоятельной работы.

1. Основные мотивы самостоятельной работы студентов

Активная самостоятельная работа студентов возможна только при наличии серьезной и устойчивой мотивации. Самый сильный мотивирующий фактор – подготовка к дальнейшей эффективной профессиональной деятельности. Среди внутренних факторов, способствующих активизации самостоятельной работы выделяют следующие:

1. *Полезность* выполняемой работ заключается в том, что результаты самостоятельной работы могут быть использованы на семинарских и практических занятиях, лабораторном практикуме, при подготовке публикации. Другим вариантом использования фактора полезности является активное применение результатов работы в профессиональной подготовке. Так, например, если студент получил задание на дипломную работу на одном из младших курсов, он может выполнять самостоятельные задания по ряду дисциплин гуманитарного и социально-экономического, естественно-научного и общепрофессионального циклов дисциплин, которые затем войдут в его выпускную квалификационную работу.
2. *Творческая деятельность.* Это может быть участие в научно-исследовательской, опытно-конструкторской или методической работе, проводимой на той или иной кафедре.
3. *Участие* в олимпиадах по учебным дисциплинам, конкурсам научно-исследовательских или прикладных работ и т.д.
4. *Участие* в научно – практических конференциях.
5. *Подготовка публикаций* для сборников тезисов и докладов научно-практических конференций, журналов, учебных пособий и т.д.
6. *Участие в грантовых конкурсах.*

2. Способы самостоятельной работы при чтении учебной и научной литературы

При остром недостатке времени у студентов встает вопрос об оптимизации обучения, то есть такой организации учебного процесса, которая обеспечила бы условия для продуктивного самообучения и самовоспитания. Важнейшую роль здесь играет овладение способами самостоятельной работы. Речь идет о том, что чтобы прежде всего научиться рациональному использованию времени при работе с книгой.

В этом особенно нуждаются первокурсники, которые еще недостаточно владеют навыками умственной деятельности, обеспечивающей успешное обучение.

Начинать самостоятельные занятия следует с первых же дней учебы в вузе. Первые дни семестра важны, чтобы включиться в работу, установить определенный равномерный ритм на весь семестр. Чтобы выполнить весь объем самостоятельной работы, необходимо заниматься самостоятельно по 4-5 часов ежедневно, кроме выходных дней.

Под ритмом работы понимают ежедневные занятия в одни и те же часы, при чередовании их с перерывами для отдыха. Вначале для организации ритмичной работы требуется сознательное напряжение воли, затем принуждение снимается, возникает привычка и работа становиться потребностью.

Ритмичная работа позволяет студенту заниматься много, не уставая, не снижая производительности и не перегружая себя. Для этого необходимо:

- Сменять один вид работы другим, что позволяет сохранять высокую работоспособность, поскольку при однообразных видах занятий человек утомляется больше, чем при работе разного характера.
- Заниматься несколькими предметами в один и тот же день не всегда целесообразно, поскольку при каждом переходе нужно вновь концентрировать внимание и затрачивать время.
- Умение сосредотачиваться – необходимое условие для умственного труда, иначе работа оказывается малопродуктивной и даже бесполезной.
- Начинать занятия немедленно, как только сел за стол. Следует начинать с уверенностью, что вскоре придет сосредоточенное состояние, но если внимание наступает не сразу или нарушается на время, нужно выяснить и устранить причины этого.
- Нужно научиться не прерывать внимания, пока читаемое не получит логического завершения, пока не будет пройден какой либо этап. Нередко внимание отвлекается посторонними мыслями, которые во время занятий следует решительно отгонять. Перерыв в занятиях следует приурочить к концу изучения параграфа, раздела или главы книги, та как в этом случае не будет потери времени при возобновлении работы. Умение сосредоточиться, углубиться в работу приобретается в результате практики, создающей определенные навыки.

- Повысить производительность умственного труда может порядок на рабочем месте и обстановка, благоприятствующая работе.

Большая часть самостоятельной работы студента состоит в **изучении литературы**. Одна из задач студента – научиться самостоятельно работать с книгой, а это требует определенных затрат энергии и времени. Поэтому надо научиться делать эту работу рационально, то есть необходимо учиться читать.

Как работать с учебной и научной книгой

Методы эффективной работы с книгой в целях развития интеллекта можно условно разделить на две группы:

1. Правильная организация процесса чтения
2. Повышение скорости чтения и восприятия.

В комплексе оба метода могут в 2-3 раза сократить время прочтения различных материалов.

При чтении текста мозг формирует «свою трактовку содержания» прочитанного. Происходит перекодирование сообщения на языке собственных мыслей читателя. Мозг выделяет «ядерное», сущностное значение из текста. Эффективность такой перекодировки зависит от осмысления и внимательности чтения.

Как показали эксперименты, знание и умелое применение некоторых упражнений дают возможность извлекать «ядерное» значение в тексте быстро и надежно. Эти упражнения основаны на использовании дифференциального алгоритма чтения. Центральное место в этом алгоритме занимает «блок доминанта». Это слово в переводе с латинского языка означает «господствующий, основной, главный». Что же такое доминанта применительно к тексту?

Доминанта – главная смысловая часть текста. Она выражается своими словами, на языке собственных мыслей, является результатом переработки текста, его осмысления в соответствии с индивидуальными особенностями читателя, выявления основного замысла автора.

Дифференциальный алгоритм чтения в соответствии с блоками позволяет реализовать логико-семантический анализ текста: вначале выделить ключевые слова, затем построить смысловые ряды и, наконец, выделив цепь знаний, сформулировать доминанту. Именно так и только так (по О.А. Андрееву) можно увидеть главное, действительно, проникнуть в суть вещей, явлений, излагаемых автором.

Возможны три основных способа чтения.

- Первый способ – артикуляция или проговаривание вслух (или почти вслух) того, что читаешь. Скорость такого чтения невелика.
- Второй способ – чтение про себя, при котором речевой процесс проявлен в форме внутренней речи, то есть без открытой артикуляции. Текст, при этом усваивается более эффективно. Способ в принципе допускает быстрое чтение.
- Третий, наиболее совершенный способ чтения – тоже молча, но в условиях максимального сжатия внутренней речи, при котором она проявляется в виде коротких залпов ключевых слов и смысловых рядов, адекватно отражающих смысл текста.

Итак, артикуляция замедляет процесс чтения и от нее необходимо избавиться. Однако не приведет ли сокращение артикуляции при повышении скорости чтения к снижению качества восприятия и осмысления полученной информации?

Как показали исследования психологов, иногда при чтении слова могут быть заменены наглядными представлениями, пространственными схемами. Целые группы слов – одним словом.

Быстро читающие люди обладают способностью, не проговаривания читаемый текст, сразу улавливать и фиксировать замысел автора, а затем усваивать его на уровне внутренней речи. В этом случае, несмотря на высокую скорость чтения, происходит глубокое понимание и усвоение прочитанного, так как основная идея понятна с самого начала. Задачу научиться такому чтению можно решать в два этапа. Первый предполагает сокращение артикуляции, если она ярко выражена, второй – овладение приемами чтения, при которых текст воспринимается крупными информационными блоками.

Как известно, людей по способу восприятия и переработки информации делят на три типа: зрительный, слуховой и кинестетический. Люди зрительного типа при чтении используют код наглядных образов, тогда как люди слухового типа применяют менее производительный код речедвижений. Наблюдения за людьми, читающими быстро, показывают, что они, как правило, относятся к зрительному типу. Вот пример, как описывает О.Бальзак процесс быстрого чтения: «Впитывание мысли в процессе чтения достигло у него способности феноменальной. Взгляд его охватывал семь – восемь строчек сразу, и разум постигал смысл со скоростью, соответствующей скорости глаз. Часто одно-единственное слово позволяло ему усвоить смысл целой фразы».

Направленным обучением можно практически любого здорового человека научить в процессе чтения использовать код наглядных зрительных образов при соответствующем сокращении артикуляции.

С опорой на работу Л.Г. Одинцова «Как научиться хорошо учиться» (М., 1996) приводим следующие рекомендации по работе с книгой.

- В тексте всегда есть элементы, нахождение и использование которых позволяет извлечь требуемую информацию наиболее быстро. Например, при чтении учебника в первую очередь отыскивается наиболее важная информация данной главы, параграфа, а она часто следует после слов: в итоге, в результате, выводы и т.д.
- Попробуйте в процессе чтения мысленно заглянуть вперед, представить себе, о чем будет идти речь, к какому выводу придет автор, как далее будет строиться изложение и т.д. например, если описывается одна сторона явления, то, очевидно, далее будет описана и другая и т.д. Это позволяет предварительно подготовиться к последующей информации.
- Хорошим упражнением по развитию навыков «предвидения» является остановка чтения в момент, когда, по вашему мнению, заканчивается какая-то часть текста. Попытайтесь предугадать содержание следующей части.
- До начала чтения текста важно собрать о нем как можно больше информации, чтобы точнее представить, что можно получить из данного текста и как лучше работать с ним. Это помогут сделать название, автор, издательство, аннотация, оглавление, предисловие и заключение. Предварительное ознакомление с книгой перед настоящим чтением позволяет сберечь время и труд.

Как правило, предисловие пишется крупным специалистом в данной области, и поэтому излагаемая проблема показывается как бы целиком, в общем плане, без подробностей. А это позволяет лучше сориентироваться, начинать чтение, зная основную цель автора.

- Перед углубленным чтением любого текста (статьи, книги, конспекта, лекции перед экзаменом) сначала бегло просмотрите его целиком. При этом постарайтесь выявить основные стержневые идеи, наиболее крупные части и логику их изложения. Лишь после такого просмотра переходите к более детальному чтению.
- Перед чтением статьи или параграфа учебника попробуйте проделать следующее: прочитайте внимательно первый абзац, потом бегло просмотрите первые или последние фразы следующих абзацев (в них обычно содержится основная информация), обратите внимание на курсивы, разрядки, подзаголовочный текста и, наконец, внимательно прочтите один-два последних абзаца; постарайтесь выявить основное направление текста и его построение.
- Прочитав в тексте интересную идею, полезно остановить свое внимание на ней, прислушаться к тем мыслям, которые она у вас вызвала, подумать о тех последствиях, которые из нее вытекают, попытаться развивать ее дальше.
- Существенно замедляют чтение регрессии – частые возвратные движения глаз, многократное повторное прочитывание материала. Возвратиться к уже прочитанному, но недостаточно хорошо понятому участку лучше всего, когда прочитан законченный смысловой фрагмент текста и сделана хотя бы попытка его осмысления, а не в процессе чтения предложения.
- Любой текст не однороден по своей информационной насыщенности. В некоторых предложениях, абзацах сконцентрировано очень много информации, например, формулируются основные положения, ведущие идеи и т.д., а другие служат лишь иллюстрацией, фоном. Таким образом, текст имеет «смысловой рельеф». Чем точнее читатель умеет определить степень важности каждого отрезка теста и приспособить к «смысловому барьеру» способ своего чтения (то есть замедлить и углубить в более важных местах и ускорять в менее важных), тем продуктивнее чтение. Постарайтесь гибко варьировать способ работы с текстом в соответствии с его «смысловым барьером».
- Чтобы чтение было эффективным, попробуйте по прочитанному всегда отвечать на 6 вопросов: «*Кто делает? Что делает? Когда? Почему? Где? Как?»*

Освойте технику быстрого чтения по специальной методике, например по книге О.А. Андреева, Л.Н. Хромова. Учитесь быстро читать (М., 1991).

Большое значение при чтении учебной и научной литературы имеет умение запоминать прочитанный материал, а для этого необходимо тренировать память. Существуют приемы, позволяющие тренировать память, которыми необходимо овладеть, что позволит повысить эффективность работы с учебной и научной литературой.

Тренировка памяти. В учебной деятельности важно не только, и не столько быстро читать, но и усваивать материал, сохранять в памяти. Память прекрасно тренируема и управляема. Однако прежде чем ее развивать, подумайте, какая именно память вам нужна: на идеи, на логику изложения материала, на схемы и формулы. Это разные виды памяти и развивать их надо по-разному.

Наблюдая за собой, выясните, как вам легче запомнить информацию – если вы ее видите, слышите или записываете. В дальнейшем постарайтесь так организовать работу, чтобы максимально использовать ведущий тип своей памяти.

Если у вас хорошая **зрительная память**, то хорошо запоминаются рисунки, расположение информации на странице, цвет и т.д. помогите себе, выделяя цветными карандашами отдельные места конспекта, обводя рамками, делая значки, пометки на полях, представляя зрительно отдельные аспекты текста.

При хорошей **слуховой памяти** лучше запоминается звучащая речь. Используйте эту особенность, выделяя интонацией, тембром голоса отдельные места текста, слушая его в записи на магнитофоне, рассуждая в слух и т.д.

В случае **памяти на движение** помогает повторная сокращенная запись запоминаемого материала, например выводов, основных положений текста, рисование таблиц, графиков, схем, а при выполнении лабораторных работ лучше все потрогать и проделать самому.

Наряду с использованием ведущего типа памяти, специально позаботьтесь и о развитии отстающих, так как при многих видах профессиональной деятельности они также могут потребоваться.

Использование приемов логического, осмысленного запоминания в несколько раз повышает продуктивность деятельности. Например, при запоминании лекции, глав учебников особенно действенным является основные аспекты содержания, но и запомнить логику – целесообразную связь отдельных частей материала.

Постарайтесь с первого курса развивать память на то, что непосредственно касается вашей будущей профессии. Это и основной круг идей данной отрасли знаний, и методы, и наиболее интересные факты, и фамилии ведущих специалистов области и т.д. при этом лучше не ждать, что запомнится само, а специально стараться запомнить нужное.

Вообще установка на запоминание, особенно длительное, положительно сказывается на прочности и точности сохранения материала в памяти. Прикажите себе запомнить надолго, а не так как нерадивый студент, спешно «набивающий» себе голову информацией непосредственно перед экзаменом с единственной целью – удержать выученное на один – два дня.

Любая информация запоминается лучше, если в ней намечены какие-то спорные моменты – ориентиры. И как по камушкам переходят реку, так и по этим ориентирам потом легче воспроизвести содержание. При запоминании текста выделяйте «смысловые опорные пункты», которые легко запоминаются, но с которыми тесно связаны целые фрагменты материала. Это может быть крылатая фраза, яркая цитата, пример, идея и т.д.

Материал запоминается непроизвольно, то есть легко и без затраты специальных усилий, если он является целью какой-либо поисковой деятельности. Например, если вы задались вопросом и нашли ответ на то, что долго искали, или нашли подтверждение гипотезы, которую вы сами выдвинули, то это запоминается само собой. Отсюда вывод – организуйте свою деятельность так, чтобы предмет запоминался, являлся целью этой деятельности. Например, ищите, выделяйте в тексте наиболее важные его положения – и они запомнятся, делите текст на части, анализируйте связи между ними – и запомнится логика текста.

При повторении курса лекций, запоминая материал по отдельным темам или даже вопросам, не забывайте повторить связь между ними. Именно тогда в голове укладывается система знаний, которая гораздо эффективнее, чем разрозненные обрывки.

В процессе развития памяти старайтесь не использовать стихийно сложившиеся мнения, механическое зазубривание, а применяйте научно обоснованные методы сознательной и рациональной организации развития памяти и поиск новых приемов.

Предпосылкой хорошей памяти являются осознание человеком своей деятельности и разграничение информации на ту, которая решающим образом помогает скорейшему достижению своих целей, и на менее существенную информацию. Начинайте любое дело с четкой и ясной формулировки его цели; определите, какая информация может оказать решающее воздействие на ее достижение, и сконцентрируйтесь на ней.

Прочному запоминанию способствует многомодельность восприятия, то есть запоминаемый текст читается, проговаривается и прослушивается. Везде, где это возможно, постарайтесь использовать три приема (слух, зрение и чувства) обработки запоминаемой информации сразу несколько органов чувств.

Не очень осмысленную вами информацию, которую, тем не менее, надо запомнить, можно удержать с помощью ассоциативных приемов мнемотехники, суть которых в том, что новое связывается с известным не прямо, а через цепочку дополнительных промежуточных ассоциации (помните цвета спектра – «Каждый охотник желает знать, где сидит фазан»). Везде, где трудно запомнить прямо, найдите дополнительный связующий мостик. Такими «связующими» мостиками являются буквальные «узелки» на память, завязываемые многими на носовом платке. В течение дня человек неизбежно пользуется носовым платком, а там – узелок, «напоминающий», что нужно не забыть сделать определенное дело.

Память будет работать прекрасно, если наряду с имеющимися приемами вы будете придумывать все новые, адекватные различным видам информации. Если такая работа привычна для вас, то с каждым годом память будет становиться все более мощной и продуктивной.

3. Формы ведения записей

Самостоятельная работа с книгой может быть успешной, если текст не только прочитан, но и законспектирован. Существует несколько **форм записей**, но любая форма записи не даст нужного результата, если не будет пробуждать мысли того, кто ее ведет, если отсутствует активная работа ума и формирование своих выводов из прочитанного.

Выбор формы записи зависит от индивидуальных особенностей человека, его образованности и опыта. При этом не меньшую роль играет назначение записей, то есть то, какие задачи ставит перед собой человек (для самообразования, для выступления на семинаре, для использования в будущем).

Введение записей мобилизует наряду со зрительной памятью, также и моторную память. Кроме того, у человека, систематически ведущего записи изучаемой литературы, создается свой фонд материалов для быстрого повторения и мобилизации накопленных знаний.

Все записи должны быть убористыми и компактными. Интервалы между строками должны быть достаточными, чтобы вписывать дополнения. Рекомендуется вести записи ручкой, а карандашом или ручкой другого цвета пользоваться для отметок и выделений при последующей работе. Полезно также датировать записи.

Записи могут носить различный характер: план, выписки, тезисы, аннотирование, конспектирование, реферирование.

1. План - наиболее краткая формой записи.. Это перечень вопросов, рассматриваемых в книге или статье. План обычно раскрывает структуру произведения, логику автора, способствует лучшей ориентации в содержании.

Так, составленным планом можно воспользоваться, чтобы вспомнить прочитанное или быстро отыскать в книге нужное место. Представление об основных пунктах плана дает оглавление книги, поэтому во многих случаях наименования глав и разделов можно использовать в качестве пунктов. Составление плана приучает логически мыслить, вырабатывать умение сжато и последовательно излагать суть вопроса в письменной и устной форме.

Существует два способа составления плана: работа над ним по ходу чтения и составление плана после ознакомления с произведением. При этом план получается более последовательным и стройным.

Трудность составления плана состоит в том, что надо выяснить для себя, прежде всего, построение изучаемого текста, ход мыслей автора и лишь затем изложить содержание работы кратко и ясно. При всей своей краткости план дает представление о содержании прочитанного. Следует учесть, что форма плана не исключает цитирования отдельных мет и обобщений. Различают простой и развернутый план. В отличие от простого плана развернутый план не только содержит перечисление вопросов, но и раскрывает основные идей произведения, может включать выдержки из него, схемы, таблицы. Планом, особенно развернутым, необходимо пользоваться при написании выступления или статьи.

В целом развернутый план дает гораздо большее представление о произведении, его основных идеях, задачах, которые в нем решаются. Он может включать положения, замечания, собственные мысли студента.

Важно знать, что составление планов помогает вырабатывать способность к отвлеченному, абстрактному мышлению, но наибольшую пользу составление плана даст подготовленным лицам, которые бывают достаточно лишь взглянуть на перечень основных вопросов, чтобы воспроизвести содержание прочитанного.

2. Тезисы – более сложная и совершенная форма записи, чем составление плана.

Это сжатое изложение основных мыслей прочитанного произведения или подготовляемого вступления. Особенностью тезисов является их утвердительный характер.

В них сосредотачивается самое главное, только выводы и обобщения, в них меньше доказательств, иллюстрации и пояснений. Тезисы не должны повторять дословно текст, но в ряде мест могут быть близки к нему, воспроизводя некоторые характерные выражения автора, важные для понимания хода его мыслей. Составление тезисов помогает глубже понять основные идеи произведения, выделить главное в нем; приучают сжато, точно и четко сформулировать свои мысли, повышает культуру речи и письма. При составлении тезисов учитывают следующее. Прежде всего, если произведение небольшое, необходимо внимательно изучать его в целом, если большое – изучать по главам и разделам. Затем, когда будут ясны основные идеи, кратко и последовательно излагать их в виде пунктов.

Различают простые и сложные, развернутые тезисы. Если записывают только утверждение чего – либо, такой тезис называют простым, а сложным тезисом будет выражение главной мысли, содержащее, кроме утверждения, еще и краткое ее доказательство.

Часто тезисы формулируются самим автором как выводы и обобщения в заключении книги или разделах книги. Нередко тезисы выделяются в тексте другим шрифтом.

Рекомендуется делать тезисные записи своими словами, причем можно записывать один абзац за другим, учитывая смысловую связь между ними. Но в большинстве случаев следует составлять сводный тезис, сложный по форме. При этом объединяется несколько утверждений, тесно связанных между собой.

Тезисы по содержанию очень близки к **конспекту**, но конспект носит более описательный характер, и его положения не столь категоричны, как в тезисах. Кроме того, конспект представляет собой более полную форму записи.

Следует отметить, что различие между формами записей условно, но в любой форме запись – важнейшая часть самостоятельной работы с книгой.

3. Выписки. Это записи текста из книги: теоретических положений, статистических данных, имеющих по мнению читателя важное значение.

Достоинство выписок состоит в точности воспроизведения текста книги, удобстве пользования записями при последующей работе, в накоплении обобщений и фактического материала. Выписки полезны для повторения, освежения в памяти прочитанного, для быстрой мобилизации своих знаний, когда необходимо в короткий срок вспомнить материал. Выписки выделяют из текста самое главное и тем самым помогают глубже понять его. Без них трудно обойтись при подготовке доклада, реферата, выступления. Выписки следует рассматривать как составную часть тезисов и конспектов.

Выписывать текст можно и по ходу чтения и после его завершения. В последнем случае надо замечать места, которые потом будут выписаны. Необходимо каждую выписку снабжать ссылкой на источник с указанием соответствующей станицы. Это нужно, чтобы в последствии можно было быстро найти в книге соответствующее место. Целесообразно выписывать из текста только такие места, в которых содержится самое главное, суть вопроса. Выписки должны быть ориентированы на изучение произведения в целом, а не отдельных мест, поскольку положения, вырванные из общего контакта, понимаются нередко совсем не так, как этого хотел автор. Иначе говоря, отдельно взятые, лишенные пояснений выдержки могут быть не поняты или поняты неправильно.

Выписки бывают дословные (цитаты) и «свободные», когда мысли автора излагаются своими словами. Следует учесть, что большие отрывки, которые трудно цитировать, целесообразнее в краткой форме переложить своими словами, но «яркие» и важные места лучше выписывать дословно. Каждую цитату следует заключать в кавычки. Если ее берут из середины предложения, то после вводных кавычек ставят три точки. Ставят их и в конце цитаты, если из предложения опущены последние слова.

Следует знать, что какого-либо единого метода выписок, годного для всех случаев, не существует, поскольку у каждого человека свои особенности мышления и восприятия, свои подход к теме. Все это влияет на содержание и характер выписок.

Выписки рекомендуется хранить в картотеке, конвертах или папках, на которых следует обозначить общую тему.

4. Аннотация – еще одна форма записи, являющаяся кратким обобщением содержания книги. Ею удобно пользоваться, если имеется намерение вернуться к изучаемому произведению. Аннотация может быть необходима и для того, чтобы не забыть о нем.

Для составления аннотации надо сначала полностью прочитать и глубоко продумать произведение. При всей своей краткости аннотация может содержать отдельные фрагменты авторского текста, а не только оценку книги или статьи.

5. Резюме очень близко к аннотации. Это запись, являющаяся краткой оценкой прочитанного материала. Различие между ними состоит в том, что аннотация сжато характеризует произведение в целом, а резюме концентрирует внимание на его выводах, главных итогах.

6. Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка

на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – непременное правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответсвтии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становиться малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В **текстуальном конспекте** сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект – это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению **тематического конспекта** предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

Следующим методом самостоятельной работы с книгой является **реферирование** на определенную тему. Слово реферат употребляется в двух различных значениях:

1. Краткое изложение содержания книги, научной работы;

2.Доклад за заданную тему на основе критического образа литературных источников.

7. Реферат – это один из самых сложных видов самостоятельной работы с книгой, а для этого следует овладеть более простыми приемами работы – разработкой плана, составлением тезисов и конспектов.

Подготовка реферата и выступление с его изложением углубляет знания, расширяет кругозор, приучает логически, творчески мыслить, развивать культуру речи.

При просмотре литературы намечается ориентировочный план реферата, в который включается обычно 3-4 основных вопроса или раздела. Каждом из разделов формулируются подвопросы, помогающие последовательно раскрыть содержание проблемы.

В процессе изучения материала формулировки подвопросов и разделов обычно уточняются. При реферировании следует делать выписки, записывать мысли, возникающие при чтении; следует также точно записывать и определения тех понятий, которые будут использованы в реферате. Из прочитанной литературы нужно заимствовать не буквальный текст, а важнейшие мысли, идеи, теоретические положения; можно цитировать небольшие отрывки, приводить диаграммы, схемы, чертежи, но главное – высказывать собственные соображения по вопросам реферата. Приведенные выше советы следует рассматривать как примерные, предполагающие и другие подходы, поскольку у каждого человека вырабатываются свои приемы и навыки составления рефератов. Большую помощь в работе над рефератом оказывают предисловия к монографиям и сборникам. В них можно найти сведения о цели издания, а также о существующих пробелах в исследовании.

При разработке плана реферата важно учитывать, чтобы каждый его пункт раскрывал одну из сторон избранной темы, а все пункты в совокупности охватывали тему целиком. Различают несколько композиционных решений реферата: во-первых, хронологическое, когда тема раскрывается в исторической последовательности; во-вторых, описательное, при котором тема расчленяется на составные части, в целом раскрывающие определенное явление; в-третьих, аналитическое, когда тема исследуется в ее причинно-следственных связях и взаимозависимых проблемах. Важно следить за тем, чтобы каждый пункт плана был соотнесен с главной темой и не содержал повторения в других пунктах. Важными разделами реферата является вступление и заключение. Во вступлении надо обосновать актуальность темы, обозначить круг составляющих ее проблем, четко и кратко определить задачу своей работы. В заключении делаются краткие выводы, подводятся итоги. В конце реферата должен быть приложен список литературы.

В отличие от тематического конспекта реферат требует большей творческой активности, самостоятельности в обобщении изученной литературы, умения логически стройно изложить материал, оценить различные точки зрения на исследуемую проблему и высказать о ней собственное мнение. В реферате важно связать теоретические положения с практикой.

Итак, реферат – это самостоятельное произведение автора, которое должно свидетельствовать о знании литературы по данной теме, ее основной проблематике, отражать точку зрения автора реферата на эту проблематику, его умение осмысливать явления жизни на основе теоретических знаний.

При оценке реферата обычно руководствуются следующими критериями:

1. Удалось ли его автору раскрыть сущность данной проблемы;
2. Сумел ли автор показать связь рассматриваемой проблемы с жизнью;
3. Проявил ли автор самостоятельность и творческий подход в изложении реферата;
4. Можно ли считать реферат логически стройным и т.д.

4. Как слушать и конспектировать лекции

Основы знаний закладываются на лекциях, им принадлежит ведущая роль в учебном процессе. На лекциях дается самое важное, основное в изучаемой дисциплине. Основные задачи, стоящие перед лектором: помочь студентам понять основы и усвоить материал на самой лекции, дать указания на то, что требует наибольшего внимания, учить правильному мышлению и создавать ясное представление о методологии изучаемой науки.

Лекции являются эффективным видом занятий для формирования у студентов способности быстро воспринимать новые факты, идеи, обобщать их, а также самостоятельно мыслить.

Лектор излагает теоретический и практический материал, относящийся к основному курсу. Из большого числа монографий, учебников, сборников лектор выбирает самое главное, помогает усвоить логику рассуждений. Интонацией голоса и манерой изложения лектором подчеркивает наиболее существенное, выделяет главное и второстепенное.

Лектор может приводить наблюдения и факты из своего личного опыта, что придает материалу убедительность, повышает интерес к предмету лекции, способствует его усвоению.

Важно помнить, что лекция – это творческий процесс, в котором участвуют одновременно и лектор, и студенты, поэтому она требует атмосферы сотрудничества и уважительного отношения к труду лектора.

Студенту следует научиться понимать и основную идею лекции, а также, следуя за лектором, участвовать в усвоении новых мыслей. Но для этого надо быть подготовленным к восприятию очередной те-

мы. Время, отведенное на лекцию, можно считать использованным полноценно, если студенты понимают роль лектора, задачи лекции, если работают вместе с лектором, а не бездумно ведут конспект.

Подготовленным можно считать такого студента, который, присутствуя на лекции, усвоил ее содержание, а перед лекцией припомнил материал раздела, излагаемого на ней или просмотрел свой конспект, или учебник.

Перед лекцией необходимо прочитывать конспект предыдущей лекции, а после окончания крупного раздела курса рекомендуется проработать его по конспектам и учебникам.

Для наиболее важных дисциплин, вызывающих наибольшие затруднения, рекомендуется перед каждой лекцией просматривать содержание предстоящей лекции по учебнику с тем, чтобы лучше воспринять материал лекции. В этом случае предмет усваивается настолько, что перед экзаменом остается сделать немногое для закрепления знаний.

Важно помнить, что ни одна дисциплина не может быть изучена в необходимом объеме только по конспектам. Для хорошего усвоения курса нужна систематическая работа с учебной и научной литературой, а конспект может лишь облегчить понимание и усвоение материала.

Основная задача при слушании лекции – учиться мыслить, понимать идеи, излагаемые лектором. Большую помощь при этом может оказать конспект. Передача мыслей лектора своими словами помогает сосредоточить внимание, не дает перейти на механическое конспектирование. Механическая запись лекции приносит мало пользы.

Ведение конспекта создает благоприятные условия для запоминания услышанного, т.к. в этом процессе принимают участие слух, зрение и рука. Конспектирование способствует запоминанию только в том случае, если студент понимает излагаемый материал. При механическом ведении конспекта, когда просто записывается слова лектора, присутствие на лекции превращается в бесполезную трату времени.

Некоторые студенты полагают, что при наличии учебных пособий, учебников нет необходимости вести конспект. Такие студенты нередко совершают ошибку, так как не используют конспект как средство, позволяющее активизировать свою работу на лекции или полнее и глубже усвоить ее содержание.

Определенная часть студентов считает, что конспекты лекции могут заменить учебники, поэтому они стремятся к дословной записи лекции и нередко не задумываются над ее содержанием. В результате при разборе учебного материала по механической записи требуется больше труда и времени, чем при понимании и кратком конспектировании лекции.

Конспект ведется в тетради или на отдельных листах. Записи в тетради легче оформить, их удобно брать с собой на лекцию или практические занятия. Рекомендуется в тетради оставлять поля для дополнительных записей, замечаний и пунктов плана. Но конспектирование в тетради имеет и недостаток: в нем мало места для пополнения новыми материалами, выводами и обобщениями. В этом отношении более удобен конспект на отдельных листах (карточках). Из него нетрудно извлечь отдельную необходимую запись, конспект можно быстро пополнить листами, в которых содержатся новые выводы, обобщения, фактические данные. При подготовке выступлений, докладов легко подобрать листки из различных конспектов и свести их вместе. В результате такой работы конспект может стать тематическим.

Но вести конспект на отдельных листках или карточках более трудоемко, чем в тетради. Карточки легко рассыпать и перепутать, приходиться обзаводиться ящичками для хранения карточек, возникает необходимость на каждом листке писать его порядковый номер.

Но затрата труда и времени окупается преимуществами конспектирования на карточках перед конспектом в тетради.

Рекомендуется делать такие карточки, которые помещаются в обычный почтовый конверт. Карточки удобно тасовать, менять при необходимости их последовательность, раскладывать на столе для обзора. Например, учителю истории карточки служили бы долго. Перед уроком можно взять соответствующий конверт и найти в нем материал по узловым вопросам темы, записанный еще в вузе.

При конспектировании допускается сокращение слов, но необходимо соблюдать меру. Каждый студент обычно вырабатывает свои правила сокращения. Но если они не введены в систему, то лучше их не применять, т.к. случайные сокращения ведут к тому, что спустя некоторое время конспект становится непонятным.

Следует знать, что не существует какого-либо единого, годного для всех метода конспектирования. Каждый ведет записи так, как ему представляется наиболее целесообразным и удобным. Собственный метод складывается по мере накопления опыта, но во всех случаях надо стремится к тому, чтобы конспективные записи были краткими и наилучшим образом содействовали глубокому усвоению изучаемого материала. Известный отечественный педагог В.А. Сухомлинский, рекомендовал учиться думать над конспектом уже на лекции и работать над записями ежедневно хотя бы в течение 2 часов. Он советовал также делить конспект на две графы: в первой кратко записывать изложенные лекции, а во второй – то, над чем надо подумать; сюда же следовало заносить узловые, главные вопросы, над которыми надо подумать постоянно,

связывая с этим повседневное чтение. Он подчеркивал, что узловые вопросы предмета будут программой, на основе которой припоминается весь материал.

5. Использование компьютера в процессе самостоятельной работы студентов.

Наиболее комплексный ряд заданий, выполняемых студентом в процессе учебы в вузе, развивающих самостоятельность – это написание реферативных, курсовых и дипломных работ, выполнение которых требует применения всего спектра знаний, умений и навыков, приобретенных студентом в процессе обучения. Алгоритм, методика и формы выполнения этих работ практически одинаковы, они различаются содержанием и глубинной проработанности материала. И реферат, и курсовая, и дипломная работы должны выполняться в соответствии с действующими требованиями ГОСТов.

На современном этапе никто уже не представляет себе самостоятельную работу без использования международной информационной сети – **Интернет**. Необходимость использования Интернета возникает не только при подготовке к практическим и семинарским занятиям, но, в большей степени, при написании различных исследовательских и творческих работ. Многие современные монографии, периодические журналы изданы только в электронном виде и с ними можно познакомиться только в Интернете.

Написание работ творческого и исследовательского характера требует знания и умения применять различные компьютерные технологии. Можно предложить следующий алгоритм работы по написанию исследовательских и творческих работ с использованием компьютера.

- Первый этап заключается в наборе материала на компьютере. Для этого необходимо, чтобы на компьютер были установлен текстовый и графический редакторы для набора текста и выполнения различных рисунков, графиков или схем. Если материал неоднородный, т.е. содержит графики, схемы, чертежи, текст, то для этих целей лучше выбрать интегрированный пакет, который позволяет совмещать различного формата файлы (например Word, PageMaker и др.). Цитаты из книг и журналов можно переснимать на сканере – удобно и быстро. Здесь как раз и понадобиться база данных, которая значительно упростит работу с выбранной литературой.
- Второй этап - корректировка ошибок, недочетов. Практика показывает, что чтение с листа более привычно и корректировать удобнее файлы, имея распечатанный образец перед собой.
- Третий этап - печать начисто. Откорректированный и исправленный текст необходимо не забыть проверить на орфографию (по возможности и стилистику) перед тем как распечатать. Чертежи лучше выводить на бумагу на графопостроителе.
- Четвертый этап - рецензия специалистов, работающих в данной области.
- Пятый этап - защита курсовых или дипломных работ на кафедре или в лаборатории. Желательно использовать презентационные компьютерные программы (например, Power point) при ответе – это увеличит наглядность доклада и использовать презентационные средства типа Proxima – проектор, позволяющий выводить на экран содержимое дисплея. Можно также использовать телевизор вместо монитора при наличии специального блока сопряжения.

Почти на всех этапах студент работает самостоятельно. За время выполнения исследования у него развиваются:

1. Навыки и методы работы с литературой: ее анализ, отбор необходимого материала.
2. Навыки и методы работы с персональным компьютером: профессиональный набор текста, выполнение рисунков и чертежей, схем и др.
3. Исследовательские навыки и др.

Поиск в Интернете.

1. **Поиск информации в Интернете** лучше всего начинать с работы в **Интернет-каталоге**.

Один из наиболее полных и хорошо систематизированных каталогов в русскоязычном секторе Интернета находится на сайте www.aport.ru. Есть много других Интернет-каталогов: www.yandex.ru, www.list.ru, www.rambler.ru (русскоязычные), www.altavista.com (англоязычный) и др. Выбор каталога зависит от вкусов пользователя, степени проработанности его тематической структуры, скорости доступа к ресурсам каталога и т.д.

2. Чтобы попасть на эту страничку, вам надо вписать URL(адрес) данного сайта в адресную строку вашего Интернет-обозревателя (браузера), которая находится в верхней части окна.

3. Перед вами откроется главная страница поисковой системы, например «Апорт».

4. Находим на этой странице ссылку на подкаталог «Наука и образование» и кликаем на ней мышью. Теперь мы попадаем на следующую страницу каталога, где пользователю предлагается выбрать интересующую его рубрику.

5. Ищем на этой странице ссылку на рубрики. Кликаем на нее. Загружается следующая страница, на которой будут ссылки на подрубрики. Под списком рубрик появятся ссылки на конкретные Интернет-ресурсы. Вы выбираете интересующий вас ресурс (при этом можно пользоваться краткой аннотацией, рейтингом популярности сайта, информацией о времени его последнего обновления) и кликаете на его ссылке. Откроется новое окно браузера, в которое будет загружен выбранный вами сайт.

Помимо **тематического поиска** в любом Интернет-каталоге есть **контекстный поиск.**

Часто бывает так, что всю страницу сохранять необязательно, так как интерес вызывают лишь отдельные ее элементы. Текстовая часть страницы без графики и средств мультимедиа сохраняется как файл языка HTML. Часто имеет смысл сохранять только текст, так как любые графические объекты занимают много места на дискетах и жестких дисках компьютера.

Если вам необходимо сохранить только графические элементы страницы (рисунки, фотографии и т.д.), достаточно кликнуть на интересующей вас картинке правой клавишей мыши. Появится диалоговое окно, в котором следует выбрать пункт «Сохранить рисунок как».

Следует помнить, что вы не сможете редактировать составленные в формате HTML Интернет-страницы. В случае если вам по каким-то причинам нужно **внести** в них **правку**, дополнить своими материалами, включить в готовый текстовый документ, то нужно:

1. Выделить мышью в окне браузера необходимый фрагмент текста (весь текст выделяется после нажатия на команды меню «правка» и «выделить все»);
2. Копировать его (команда «копировать» на вкладке «Правка» / «edit»);
3. Вставить в нужный текстовый файл в программе Word (команда «Вставить» / «Paste»).

Вы также можете **распечатать** нужные вам страницы:

1. Одновременно нажмите клавиши ctrl и P или выберите команду «Печать» / «Print» на вкладке «Файл».
2. Если вам не нужно распечатывать всю страницу, то вы можете распечатать ее фрагмент. Для этого вы выделяете интересующий участок страницы мышью, а в диалоговом окне печати, указывая диапазон печати, выберите пункт «Выделенный фрагмент».
3. Если вы хотите вернуться на предыдущую страницу, достаточно кликнуть мышкой кнопку «Назад», которая находится в левом верхнем углу окна вашего браузера. Обратный шаг можно сделать, нажав на стрелку «Вперед».

Для того чтобы в следующий раз точно попасть на нужную вам страницу Интернета, совсем не обязательно переписывать ее адрес, часто громоздкий и сложный. Достаточно всего лишь добавить ссылку на страницу в папке «Избранное» (она расположена вверху экрана, на рабочей панели браузера). Если вы хотите запомнить много страниц и к тому же систематизировать их, то направляйтесь на специальный сайт www.zakladki.ru, где вы сможете сохранить гиперссылку на любую Интернет-страницу. В этом случае вы сможете работать не только со ссылками, подобранными вами, но и другими пользователями (при условии, что доступ к ним не закрыт паролем).

6. Подготовка к экзаменам

Экзаменационная сессия – очень тяжелый период работы для студентов и ответственный труд для преподавателей. Главная задача экзаменов – проверка качества усвоения содержания дисциплины.

На основе такой проверки оценивается учебная работа не только студентов, но и преподавателей: по результатам экзаменов можно судить и о качестве всего учебного процесса. При подготовке к экзамену студенты повторяют материал курсов, которые они слушали и изучали в течение семестра, обобщают полученные знания, выделяют главное в предмете, воспроизводят общую картину для того, чтобы яснее понять связь между отдельными элементами дисциплины.

Экзаменам, как правило, предшествует сдача зачетов. К экзаменам допускаются только те студенты, которые сдали зачеты.

При подготовке к экзаменам основное направление дают программы курса и конспект, которые указывают, что в курсе наиболее важно. Основной материал должен прорабатываться по учебнику, поскольку конспекта недостаточно для изучения дисциплины. Учебник должен быть проработан в течение

семестра, а перед экзаменом важно сосредоточить внимание на основных, наиболее сложных разделах. Подготовку по каждому разделу следует заканчивать восстановлением в памяти его краткого содержания в логической последовательности.

До экзамена обычно проводится консультация, но она не может возместить отсутствия систематической работы в течение семестра и помочь за несколько часов освоить материал, требующийся к экзамену. На консультации студент получает лишь ответы на трудные или оставшиеся неясными вопросы. Польза от консультации будет только в том случае, если студент до нее проработает весь материал. Надо учиться задавать вопросы, вырабатывать привычку пользоваться справочниками, энциклопедиями, а не быть на иждивении у преподавателей, который не всегда может тут же, «с ходу» назвать какой-либо факт, имя, событие.

На экзамене нужно показать не только знание предмета, но и умение логически связно построить устный ответ.

Получив билет, надо вдуматься в поставленные вопросы для того, чтобы правильно понять их. Нередко студент отвечает не на тот вопрос, который поставлен, или в простом вопросе ищет скрытого смысла. Не поняв вопроса и не обдумав план ответа, не следует начинать писать. Конспект своего ответа надо рассматривать как план краткого сообщения на данную тему и составлять ответ нужно кратко. При этом необходимо показать умение выражать мысль четко и доходчиво.

Отвечать нужно спокойно, четко, продуманно, без торопливости, придерживаясь записи своего ответа.

На экзаменах студент показывает не только свои знания, но и учится владеть собой. После ответа на билет могут следовать вопросы, которые имеют целью выяснить понимание других разделов курса, не вошедших в билет. Как правило, на них можно ответить кратко, достаточно показать знание сути вопроса. Часто студенты при ответе на дополнительные вопросы проявляют поспешность: не поняв смысла того, что у них спрашивают, начинают отвечать и нередко говорят не по сути.

Студент должен знать, что на экзамене осуществляется не только контроль и выставляется оценка, но это еще и дополнительная возможность, систематизация знаний. Если говорить о сверхзадаче экзаменатора, то она состоит в уяснении не только и не столько того, что студент выучил, сколько того, чему он научился и что останется у него после экзамена, поскольку этот остаток будет характеризовать образовательный уровень студента.

Следует помнить, что необходимым условием правильного режима работы в период экзаменационной сессии является нормальный сон, поэтому подготовка к экзаменам не должна быть в ущерб сну. Установлено, что сильное эмоциональное напряжение во время экзаменов неблагоприятно отражается на нервной системе и многие студенты из-за волнений не спят ночи перед экзаменами. Обычно в сессию студенту не до болезни, так как весь организм озабочен одним - сдать экзамены. Но это еще не значит, что последствия неправильно организованного труда и чрезмерной занятости не скажутся потом. Поэтому каждый студент помнить о важности рационального распорядка рабочего дня и о своевременности снятия или уменьшения умственного напряжения.

Список используемой литературы

Основная

7. Гост 19.701-90 (ИСО 5807-85) схемы алгоритмов, программ, данных и систем.
8. Канцедал С.А. Алгоритмизация и программирование: учебное пособие. – М.: ИД «Форум»: ИНФРА-М, 2008 – 352 с.
9. Программирование на языке Паскаль: задачник/ под редакцией Усковой О.Ф. – СПб.: Питер, 2005. – 336 с.
10. Установочный диск с развернутой системой помощи языка программирования Visual Basic, 2010
11. Культин, Н.Б. Turbo Pascal в задачах и примерах: учебное пособие. – БХВ., 2007 – 256 с.
12. Павловская, Т.А. Паскаль. Программирование на языке высокого уровня. - СПб: Питер, 2007 – 393 с.

Дополнительная

3. В.В. Фаронов Турбо Паскаль. Начальный курс. Учебное пособие-М.: «Нолидж», 1996.-613 с.
4. С.А. Абрамов и др. Задачи по программированию. — М.: Наука, 1988. – 210 с.

Интернет-ресурсы

4. Методы программирования
   http://www.tstu.ru/education/elib/pdf/2006/kulakov.pdf
5. Структурное программирование
   http://digital.sibsutis.ru/Progr/StrProgr.htm
6. Модульное программирование
   http://digital.sibsutis.ru/Progr/ManyMod.htm