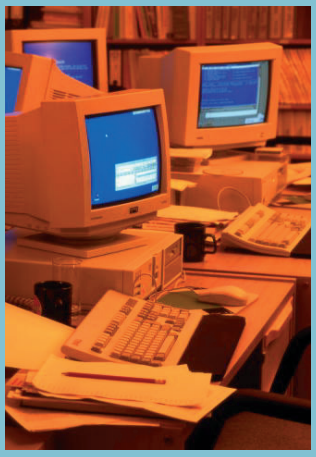
ТОЛЬЯТТИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Д.М. Ахмедханлы

ПРОГРАММИРОВАНИЕ НА TURBO PASCAL



Тольятти
Издательство ТГУ
2012

Министерство образования и науки Российской Федерации
Тольяттинский государственный университет
Институт математики, физики и информационных технологий
Кафедра «Информатика и вычислительная техника»

Д.М. Ахмедханлы

ПРОГРАММИРОВАНИЕ НА TURBO PASCAL

Учебно-методическое пособие

Тольятти
Издательство ТГУ
2012

УДК 004.43(075.8)
ББК 32.973-081.1 Я73
А954

Рецензенты:
к.т.н., профессор Волжского университета им. В.Н. Татищева
С.И. Трубачева;
к.п.н., доцент Тольяттинского государственного университета
Е.В. Панюкова.

А954 Ахмедханлы, Д.М. Программирование на Turbo Pascal : учеб.-метод. пособие / Д.М. Ахмедханлы. – Тольятти : Изд-во ТГУ, 2012. – 88 с. : обл.

Учебно-методическое пособие содержит теоретические материалы по изучению дисциплины «Информатика», методические рекомендации по выполнению практических заданий, примеры типовых задач.

Предназначено для студентов направления подготовки 270800 «Строительство» (квалификация – бакалавр) очной формы обучения.

УДК 004.43(075.8)
ББК 32.973-081.1 Я73

Рекомендовано к изданию научно-методическим советом Тольяттинского государственного университета.

ВВЕДЕНИЕ

Дисциплина «Информатика» изучается на первом курсе в течение двух семестров с применением балльно-рейтинговой системы обучения.

Федеральный государственный образовательный стандарт по дисциплине «Информатика» включает нижеперечисленные разделы и темы.

Разделы ФГОС	Семестр	Модуль
Понятие информации; общая характеристика процессов сбора, передачи, обработки и накопления информации	1	1
Алгоритмизация и программирование; языки программирования высокого уровня	1	2
Модели решения функциональных и вычислительных задач; базы данных	2	3
Основы защиты информации и сведений, составляющих государственную тайну; методы защиты информации; компьютерная графика	2	4

Цель данного учебно-методического пособия – помочь студентам в изучении следующих разделов: «Алгоритмизация и программирование»; «Языки программирования высокого уровня».

Учебно-методическое пособие содержит методические указания по изучению дисциплины, теоретические сведения по дисциплине, примеры типовых задач, варианты заданий по темам курса, вопросы к экзамену, библиографический список, глоссарий.

В методических указаниях определены цели и задачи дисциплины, даны методические рекомендации по её изучению.

I. РУКОВОДСТВО ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ

Изучение дисциплины «Информатика» согласно учебному плану предусматривает следующее распределение часов по видам учебных занятий.

1 семестр			
Лекции (час.)	Лабораторные занятия (час.)	Практические занятия (час.)	Форма контроля
12	34	–	Экзамен

1. Цели и задачи дисциплины. Компетенции

Цели изучения разделов «Алгоритмизация и программирование», «Языки программирования высокого уровня» дисциплины «Информатика» – ознакомление с основами алгоритмизации и языками программирования, применение полученных навыков в профессиональной деятельности.

В результате изучения дисциплины студент должен обладать следующими компетенциями:

- культурой мышления, способностью к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения (ОК-1);
- способностью понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной тайны (ПК-4);
- основными методами, способами и средствами получения, хранения, переработки информации, навыками работы с компьютером как средством управления информацией (ПК-5).

2. Методические рекомендации по изучению дисциплины

Тема 1. Алгоритмизация

Цель – ознакомить студентов с основными понятиями и принципами алгоритмизации.

Учебные вопросы

1. Понятие алгоритма, свойства алгоритма.
2. Базовые алгоритмические структуры.
3. Следования. Ветвления. Циклы.
4. Вложенные циклические структуры. Итерационные структуры.

Изучив данную тему, студент должен:

знать:

– способы записи алгоритмов;
– основные базовые структуры алгоритмов;

уметь:

– определять структуру алгоритма, тип;
– составлять блок-схемы.

При работе с учебным материалом темы необходимо:

– изучить разделы 1.1–1.7;
– ознакомиться с примерами типовых задач в разделе 1.8;
– ответить на вопросы для самоконтроля.

Тема 2. Основы программирования

Цель – ознакомиться со структурой программы на языке Паскаль, основными операторами, алгоритмами структуры следования.

Учебные вопросы

1. Этапы решения задач на ЭВМ.
2. Структура программы на языке Паскаль.
3. Основные операторы.

Изучив данную тему, студент должен:

знать:

– алфавит языка Паскаль, типы данных;
– операторы ввода-вывода, оператор присваивания;

уметь:

– составлять блок-схемы структуры «следование»;
– писать простые программы с использованием операторов ввода-вывода, оператора присваивания.

При работе с учебным материалом темы необходимо:

– изучить разделы 2.1–2.8;
– ознакомиться с примерами типовых задач в разделе 2.9;
– ответить на вопросы для самоконтроля;
– выбрать вариант задания из раздела III по теме «Следования»;
– составить блок-схему алгоритма, написать и отладить программу;
– предъявить результаты работы преподавателю.

Тема 3. Типовые вычислительные процессы. Ветвления

Цель – ознакомиться с алгоритмами структуры «ветвления», логическим оператором, оператором выбора.

Учебные вопросы

1. Операторы проверки условий, оператор перехода.
2. Логические операции.
3. Операции отношений.
4. Оператор выбора.

Изучив данную тему, студент должен:

знать:

– логический оператор;
– структуру вложенных логических операторов;
– структуру оператора выбора;

уметь:

– строить алгоритмы структуры «ветвления»;
– использовать вложенные логические операторы;
– использовать в программе оператор выбора.

При работе с учебным материалом темы необходимо:

– изучить разделы 3.1–3.3;
– ознакомиться с примерами типовых задач в разделе 3.4;
– ответить на вопросы для самоконтроля;
– выбрать вариант задания из раздела III по теме «Ветвления»;
– составить блок-схему алгоритма, написать и отладить программу;
– предъявить результаты работы преподавателю.

Тема 4. Циклические вычислительные процессы

Цель – ознакомиться с алгоритмами циклической структуры, операторами цикла с параметром, с предусловием, с постусловием.

Учебные вопросы

1. Циклические вычислительные процессы.
2. Оператор цикла с параметром, с предусловием, с постусловием.
3. Вложенные циклы.
4. Итерационные циклические вычислительные процессы.

Изучив данную тему, студент должен:

знать:

– операторы цикла;
– необходимые действия для организации цикла;
– правила организации вложенных циклов, оператор прерывания цикла;

уметь:

– строить алгоритмы циклической структуры;
– строить алгоритмы со структурой вложенных циклов;
– применять операторы циклов с параметром, с предусловием, с постусловием.

При работе с учебным материалом темы необходимо:

– изучить разделы 4.1–4.5;
– ознакомиться с примерами типовых задач в разделе 4.6;
– ответить на вопросы для самоконтроля;
– выбрать вариант задания из раздела III по теме «Циклы»;
– составить блок-схему алгоритма, написать и отладить программу;
– предъявить результаты работы преподавателю.

Тема 5. Операции с индексированными переменными

Цели: ознакомиться со способом хранения однотипной информации в поименованном наборе, массиве; составить понятие о способе хранения информации в двумерном массиве, матрице.

Учебные вопросы

1. Операции с индексированными переменными.
2. Одномерные массивы.
3. Двумерные массивы.

Изучив данную тему, студент должен:

знать:

– описание массивов;
– ввод-вывод массивов, хранение массивов, обработку массивов;

уметь:

– использовать индексированные переменные;
– организовать хранение информации в виде массива одномерного;
– организовать хранение информации в двумерном массиве, матрице.

При работе с учебным материалом темы необходимо:

– изучить разделы 5.1 и 5.2;
– ознакомиться с примерами типовых задач в разделах 5.3 и 5.4;
– ответить на вопросы для самоконтроля;
– выбрать вариант задания из раздела III по теме «Одномерные массивы»;
– составить блок-схему алгоритма, написать и отладить программу;
– выбрать вариант задания из раздела III по теме «Двумерные массивы»;
– составить блок-схему алгоритма;
– написать программу и отладить ее по заранее подготовленному тесту;
– предъявить результаты работы преподавателю.

Тема 6. Подпрограммы

Цели: ознакомиться со структурой программы, содержащей процедуру или функцию, использовать глобальные и локальные переменные, применять формальные и фактические параметры.

Учебные вопросы

1. Подпрограммы на языке Паскаль.
2. Организация процедур.
3. Организация функций.

Изучив данную тему, студент должен:

знать:

– структуру программы, содержащей функцию или процедуру;
– назначение формальных параметров;
– назначение фактических параметров;

уметь:

– использовать процедуры и функции;
– применять глобальные и локальные переменные;
– организовать обращение к процедуре или функции из основной программы.

При работе с учебным материалом темы необходимо:

– изучить разделы 6.1–6.4;
– ознакомиться с примерами типовых задач в разделе 6.5;
– ответить на вопросы для самоконтроля;
– выбрать вариант задания из раздела III по теме «Подпрограммы»;
– составить блок-схему алгоритма;
– написать программу и отладить ее по заранее подготовленному тесту;
– предъявить результаты работы преподавателю.

Тема 7. Языки программирования высокого уровня

Цель – ознакомиться с современными языками высокого уровня.

Учебные вопросы

1. Эволюция языков программирования.
2. Классификация языков программирования.
3. Трансляторы, компиляторы, интерпретаторы.
4. Интегрированные среды программирования.

Изучив данную тему, студент должен:

знать:

– классификацию современных языков программирования;
– назначение компиляторов и интерпретаторов;
– назначение интегрированных сред программирования.

При работе с учебным материалом темы необходимо:

– изучить разделы 7.1–7.5;
– ответить на вопросы самоконтроля.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Тема 1. Алгоритмизация

1.1. Алгоритм и его свойства. Способы записи алгоритма

Понятие алгоритма такое же основополагающее для информатики, как и понятие информации. Название «алгоритм» произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда бен Муса Хорезми (аль-Хорезми), жившего в 783–850 гг. В своей книге «Об индийском счете» он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними «столбиком», знакомые теперь каждому школьнику. В XII веке эта книга была переведена на латынь и получила широкое распространение в Европе.

Алгоритм – подробное описание последовательности действий, позволяющих решить конкретную задачу. Элементарные действия, на которые разбивается алгоритм, называются инструкциями или командами.

Свойства алгоритма:

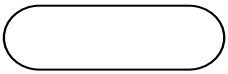
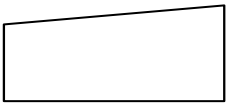
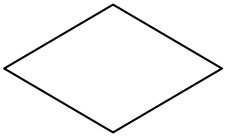
- дискретность – представление алгоритма в виде последовательности шагов;
- массовость – применимость алгоритма к некоторому множеству исходных данных;
- определенность – за конечное число шагов либо должен быть получен результат, либо доказано его отсутствие;
- однозначность – при повторном применении алгоритма к тем же исходным данным должен быть получен тот же результат.

На практике наиболее распространены следующие формы представления алгоритмов:

- словесная (запись на естественном языке);
- графическая (изображения из графических символов);
- псевдокоды (описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);
- программная (тексты на языках программирования).

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Такое графическое представление называется схемой алгоритма или блок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т. п.) соответствует геометрическая фигура. Блоки соединяются линиями переходов, определяющими очередность выполнения действий. В таблице приведены основные блоки.

Блок	Назначение
	Начало, конец алгоритма
	Ввод значений переменных с клавиатуры
	Блок действий, присвоение переменным вычислительных значений
	Проверка условия и выбор одного из двух или нескольких возможных путей дальнейшего решения
	Блок цикла
	Обращение к подпрограмме
	Вывод результатов на печать

Правила построения блок-схем:

– блок-схема выстраивается в одном направлении либо сверху вниз, либо слева направо;
– все повороты соединительных линий выполняются под углом 90 градусов.

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных блоков. Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: следование, ветвление, цикл. Характерной особенностью базовых структур является наличие в них одного входа и одного выхода. При разработке блок-схемы допускается делать любые записи внутри блоков, однако эти записи должны содержать достаточно информации для выполнения очередных действий.

1.2. Базовые алгоритмические структуры

Алгоритмы можно представлять как некоторые структуры, состоящие из отдельных базовых (основных) элементов. Естественно, что при таком подходе к алгоритмам изучение основных принципов их конструирования должно начинаться с изучения этих базовых элементов. Для их описания будем использовать язык схем. Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: ***следование, ветвление, цикл.*** Характерной особенностью базовых структур является наличие в них одного входа и одного выхода.

1.3. Базовая структура «следование»

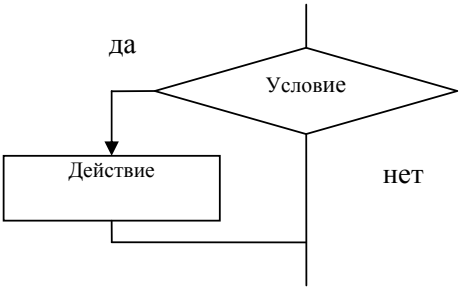
Следование – действия выполняются строго в том порядке, в котором записаны. Образуется последовательностью действий, следующих одно за другим.

1.4. Базовая структура «ветвление»

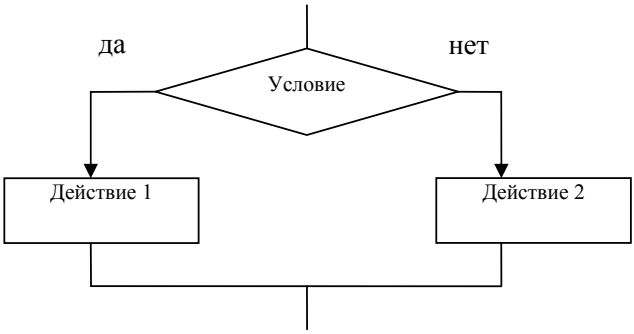
Ветвление – в зависимости от справедливости проверяемого условия (да или нет) алгоритм может пойти по одной из двух возможных ветвей. Происходит выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

Структура «ветвление» существует в трех основных вариантах:

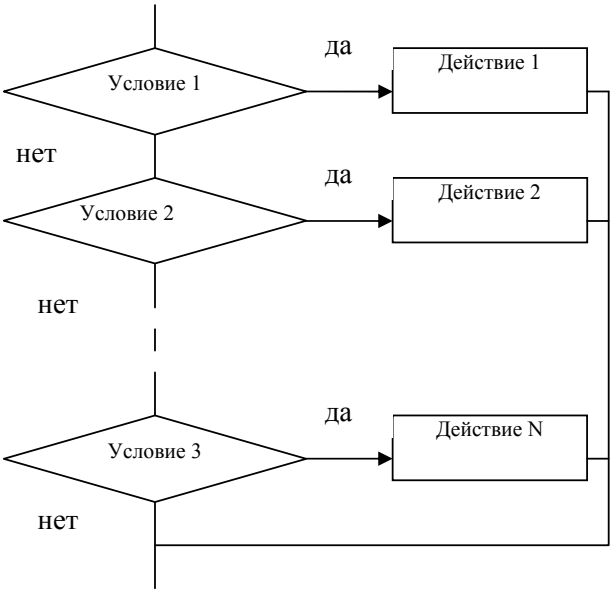
1) если – то;



2) если – то – иначе;



3) выбор.

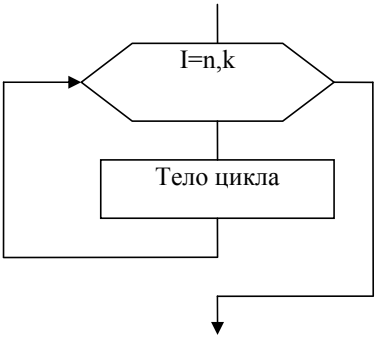


1.5. Базовая структура «цикл»

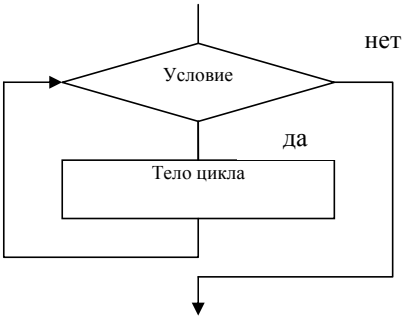
Циклы – действия повторяются многократно по одним и тем же математическим зависимостям. Обеспечивается многократное выполнение некоторой совокупности действий, которая называется телом цикла.

Циклические структуры бывают трех типов:

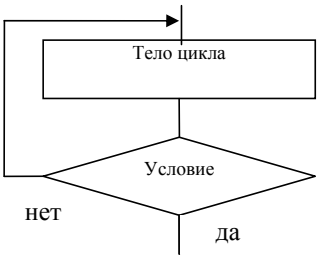
1) с параметром цикла – выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне;



2) с предусловием – выполнять тело цикла до тех пор, пока выполняется условие;



3) с постусловием – выполнять тело цикла до тех пор, пока не выполнится условие.



1.6. Итерационные циклы

Особенностью ***итерационного цикла*** является то, что число повторений тела цикла заранее неизвестно. Для его организации и используется цикл с постусловием. Выход из итерационного цикла осуществляется при выполнении некоторого условия. На каждом шаге вычислений происходит последовательное приближение к искомому результату и проверка условия достижения последнего.

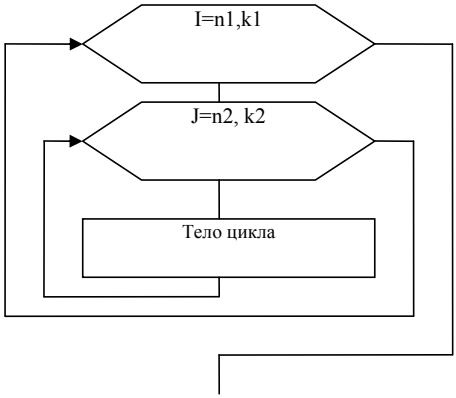
1.7. Вложенные циклы

Циклические вычислительные процессы могут быть вложенной структуры, когда один цикл (внешний) содержит внутри себя еще один (внутренний).

Возможны случаи, когда внутри тела цикла необходимо повторять некоторую последовательность операторов, т. е. организовать внутрен-

ний цикл. Такая структура получила название цикла в цикле или ***вложенных циклов***. Глубина вложения циклов (то есть количество вложенных друг в друга циклов) может быть различной.

При использовании такой структуры необходимо помнить, что параметр внутреннего цикла меняется быстрее параметра внешнего, при одном значении параметра внешнего цикла параметр внутреннего пробегает все свои возможные значения.



1.8. Примеры типовых задач по теме «Алгоритмизация»

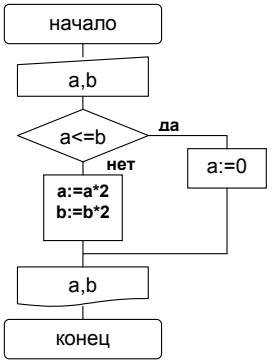
1. Вычислить и вывести значение функции $y = 4\cos^2(a\pi x)$. Исходные данные ***x*** и ***a*** ввести с клавиатуры.

Тип вычислительного процесса – следование.

Блок-схема	Пояснения
начало	Начало алгоритма
a,x	Ввод исходных данных
y:=4*cos²(a*π*x)	Вычисления
y	Вывод результата
конец	Конец алгоритма

2. Даны два действительных числа. Заменить первое число нулем, если оно меньше или равно второму, и удвоить числа в противном случае.

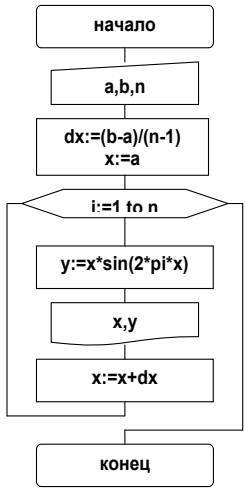
Тип вычислительного процесса – ветвление.

Блок-схема	Пояснения
	Начало алгоритма Ввод исходных данных Выбор дальнейшего пути вычислений Печать результата Конец алгоритма

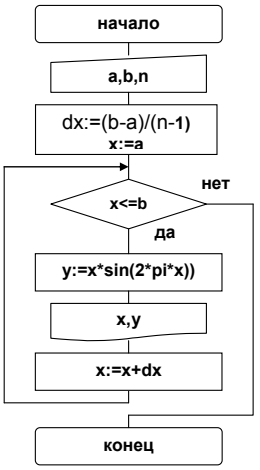
3. Составить программу расчета значений функции $y = x\cos(2\pi x)$ на интервале [a;b] в *n* равностоящих точках. Границы интервала и количество точек ввести с клавиатуры.

Представлены три блок-схемы:

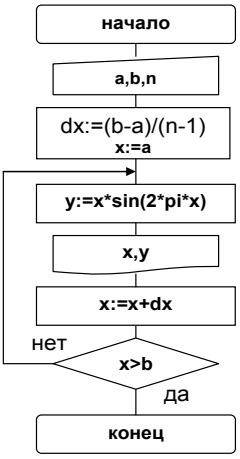
1) с параметром цикла;

Блок-схема	Пояснения
	Начало алгоритма Ввод исходных данных Вычисление шага аргумента Блок цикла Вычисление значения функции Печать результата Увеличение аргумента на шаг Конец алгоритма

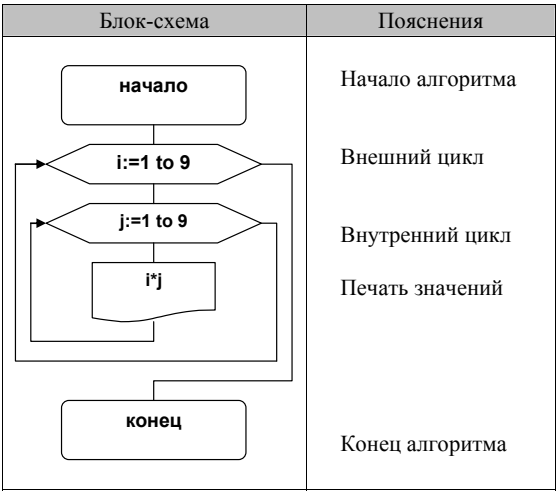
2) цикл с предусловием;

Блок-схема	Пояснения
	Начало алгоритма Ввод исходных данных Вычисление шага аргумента Проверка на вход из цикла Вычисление значения функции Печать результата Увеличение аргумента на шаг Конец алгоритма

3) цикл с постусловием.

Блок-схема	Пояснения
	Начало алгоритма Ввод исходных данных Вычисление шага аргумента Вычисление функции Печать результата Увеличение аргумента на шаг Проверка на выход из цикла Конец алгоритма

4. Вывести на экран таблицу умножения.



Вопросы для самоконтроля

1. Что понимают под алгоритмом?
2. Каковы способы записи алгоритмов?
3. В чем заключаются основные свойства алгоритма?
4. Перечислите основные алгоритмические структуры и опишите их.
5. Каковы основные принципы разработки алгоритмов?
6. Назовите основные этапы составления алгоритмов.

Тема 2. Основы программирования

Программирование – интересная, живая, быстро развивающаяся наука. Первые шаги при обучении программированию для многих оказываются очень нелегкими. Главное качество программиста – хорошее логическое мышление – развивается только в упорной и кропотливой работе.

2.1. Этапы решения задач на ЭВМ

Решение задачи разбивается на этапы.

1. Постановка задачи.
2. Формализация (математическая постановка).
3. Выбор (или разработка) метода решения.
4. Разработка алгоритма.

5. Составление программы.
6. Отладка программы.
7. Вычисление и обработка результатов.

При постановке задачи выясняется конечная цель и вырабатывается общий подход к решению задачи. Выясняется, сколько решений имеет задача и имеет ли их вообще. Изучаются общие свойства рассматриваемого физического явления или объекта, анализируются возможности данной системы программирования.

На этапе постановки задачи все объекты описываются на языке математики, выбирается форма хранения данных, составляются все необходимые формулы.

Выбор существующего или разработка нового метода решения очень важен. Метод решения записывается применительно к данной задаче на одном из алгоритмических языков (чаще на графическом).

2.2. Алгоритмический язык Pascal

Алгоритмический язык Паскаль был разработан в 1973 году швейцарским математиком Никлаусом Виртом для обучения студентов структурному программированию. Язык получил свое название в честь французского математика Блеза Паскаля (1623–1662).

Выбор Паскаля для обучения программированию объясняется рядом его достоинств. Во-первых, этот язык полно отражает идеи структурного программирования. Во-вторых, Паскаль предоставляет гибкие возможности в отношении используемых структур данных. Большое внимание в языке уделено вопросу повышения надежности программ: средства языка позволяют осуществлять достаточно полный контроль правильности использования данных различных типов и программных объектов как на этапе трансляции программ, так и на этапе их выполнения. Благодаря перечисленным возможностям, Паскаль широко применяется не только в области обучения, но и в практической работе.

Программа – это запись алгоритма на языке программирования, приводящая к конечному результату за конечное число шагов.

Программа – это детальное и законченное описание алгоритма средствами языка программирования. Исполнителем программы является компьютер. Для выполнения компьютером программа должна быть представлена в машинном коде последовательности чисел, понимаемых

процессором. Написать программу в машинных кодах вручную достаточно сложно. Поэтому сегодня практически все программы создаются с помощью языков программирования, которые по своему синтаксису и семантике приближены к естественному человеческому языку. Это снижает трудоемкость программирования. Однако листинг программы, записанный с помощью языка программирования, должен быть преобразован в машинный код. Эта операция выполняется автоматически с помощью специальной служебной программы, называемой *транслятором*.

Основные элементы программирования:

– ввод информации;
– хранение информации;
– команды обработки (операции);
– вывод данных;
– проверка условий;
– повторные выполнения (циклы);
– подпрограммы (процедуры).

2.3. Алфавит языка

Программа на Паскале записывается в виде последовательности символов, образующих алфавит языка. Алфавит включает:

– заглавные и прописные латинские буквы;
– арабские цифры;
– знаки препинания (: ; , .);
– знаки операций;
– некоторые спецсимволы.

Арифметические выражения и правила их записи

Арифметические выражения строятся из простых операндов, связанных знаками арифметических операций:

y = a + b*2

Знаки операций	Пояснения
div	деление нацело
mod	остаток от деления
*	умножение
/	деление
–	вычитание
+	сложение

Операции div и mod

Целочисленное деление div отличается от обычной операции деления тем, что возвращает целую часть частного, а дробная часть отбрасывается.

Например:

17 div 3 = 5;

8 div 2 = 4;

1 div 5 = 0.

Взятие остатка от деления mod вычисляет остаток, полученный при выполнении целочисленного деления.

Например:

17 mod 3 = 2;

8 mod 2 = 0;

1 mod 5 = 1.

При выполнении арифметических операций соблюдаются следующие правила:

1) два знака не могут следовать один за другим;

2) соблюдается следующая иерархия (приоритет) выполнения:

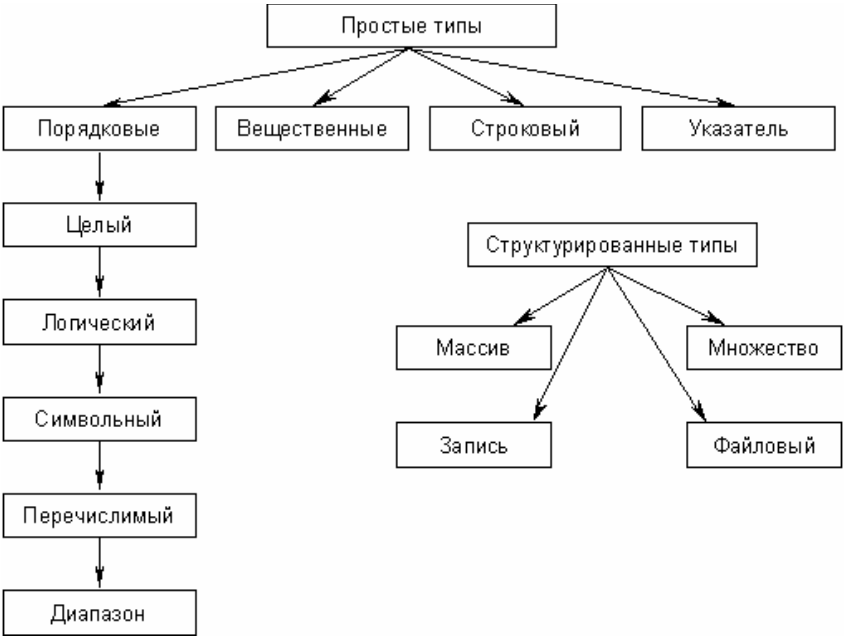
- стандартные функции;
- div, mod;
- *, /
- +, –

3) изменить иерархию выполнения арифметических операций можно с помощью скобок.

2.4. Типы данных

Переменные на языке Паскаль задаются своими именами.

Имена переменных могут содержать малые и большие буквы латинского алфавита, арабские цифры и знак подчеркивания. Переменные, используемые в программе, обязательно должны быть описаны. При описании переменных задаются имена и типы переменных. Тип – определяет объем оперативной памяти, выделяемой под хранение переменной. Для описания стандартных типов переменных используют зарезервированные английские слова. Помимо стандартных типов, можно строить и пользовательские типы данных, которые базируются только на стандартных типах.



Паскаль производит предварительную инициализацию переменных. Все переменные, описанные в программе, автоматически обнуляются.

Целые типы

Тип	Диапазон значений	Память (байт)
shortint (короткие целые)	-128÷127	1
longint (удвоенные целые)	-231÷231-1	4
integer (обычные целые)	-32768÷32767	2
word (целые положительные)	0÷65535	2
byte (целые короткие положительные)	0÷255	2

Логический тип

boolean – логические переменные занимают 1 байт памяти, могут принимать два значения:

1) True (истина);

2) False (ложь).

Логические переменные могут использоваться только в логических выражениях.

Над данными логического типа нельзя выполнять обычные арифметические операции, для них определены логические операции и операции сравнения.

Для идентификаторов имеет место:

True> False

Над логическими переменными возможны следующие логические операции:

and (и);

or (или);

not (не).

Символьный тип

char – символьные переменные, служат для хранения одного символа (буква, цифра, знаки препинания, специальные символы, непосредственно код) и занимают 1 байт памяти.

Строковый тип

string – строковые переменные, служат для хранения любой цепочки символов и занимают 255 байт памяти.

Вещественный тип

real – вещественные переменные, занимают 6 байт памяти (11 знаков после запятой).

Вещественные числа могут быть заданы в форме:

• с фиксированной точкой:

0.5 +5.0 -133.15;

• с плавающей точкой:

3.5 E 2 0.45 E -3

Форма с плавающей точкой используется для изображения очень больших или очень маленьких чисел.

2.5. Стандартные функции

Имя функции	Математическая запись	Тип результата
sin (x)	sin x	вещ.
cos (x)	cos x	вещ.
arctg(x)	arctgx	вещ.

Имя функции	Математическая запись	Тип результата
exp (x)	ex	вещ.
ln (x)	ln x	вещ.
pi	3.14	вещ.
abs (x)	| x |	вещ.
sqr (x)	x2	вещ.
sqrt (x)	$\sqrt{x}$	вещ.
trunc (x)	ближайшее наименьшее целое число	цел.
int (x)	целая часть числа	цел.
round (x)	ближайшее целое число (математическое округление)	цел.
frag (x)	дробная часть числа	вещ.
random (x)	генератор случайных чисел от 0 до x; если x – отсутствует диапазон чисел 0÷1	вещ.
odd (x)	возвращает TRUE, если x – число нечетное	лог.

Аргументом стандартной функции может быть переменная, константа, выражение, стоящее справа от имени в скобках. Для тригонометрических функций аргумент задается в радианах.

При написании формул в программе на Паскале необходимо учитывать все правила записи арифметических выражений.

Например:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.$$

Данное выражение в программе на языке Паскаль будет выглядеть:

x: = (-b + sqrt(sqr(b)–4*a*c))/(2*a).

Паскаль не допускает смешанных выражений. Допустимы выражения, в которых слева от знака присваивания – вещественная переменная, справа – целое выражение.

Если в выражении есть хотя бы одна вещественная переменная, результат будет вещественным.

Некоторые полезные формулы

Формула возведения числа в любую степень:

$$a^x = e^{x*\ln(a)} = \exp(x*\ln(a)).$$

Формула перевода числа из градусов в радианы:

$$x_{рад} = x^o * pi/180.$$

2.6. Структура программы на языке Паскаль

Программа на языке Паскаль имеет следующую структуру:

program <имя>;
<описательная часть>;
<раздел функций и процедур>;
begin
<исполнительная часть>;
end.

Зарезервированные слова:

- program <имя> – необязательная строка;
- begin – начало;
- end – конец.

<имя> – присваивается составителем программы (строится по правилам составления имен переменных).

Описательная часть программы

Все переменные, используемые в программе, должны быть описаны.

Описание начинается со служебного слова var.

Например:

```
program ff;
var
i,n: integer;
x,y,z: real;
begin;
...
```

Список переменных от типа отделяется – ":", одно описание от другого – ";", список переменных перечисляется через – ",". Если в программе используются метки, то они описываются с помощью служебного слова label. Метки могут быть числовые и символьные.

В программах на Паскале можно использовать константы, которые описываются с помощью служебного слова const.

```
Const n=100;
```

Переменная-константа (n) более в программе не описывается, ее тип определяется присвоенным ей числовым значением.

С помощью служебного слова *uses* можно подключать к программе стандартные библиотечные модули. Стандартные модули объединяют

функции определенного назначения и в случае необходимости подключаются к программе.

Например, для использования функции очистки экрана (clrscr) к программе подключают стандартный модуль crt.

Uses crt;

Исполнительная часть программы

Выполнение программы начинается именно с исполнительной части.

Отдельные инструкции, входящие в программу, называются операторами. Операторы отделяются один от другого – **";"**.

Операторы бывают трех типов:

1) пустой оператор;
2) простой оператор;
3) составной оператор.

Структура составного оператора:

begin

<оператор 1>; <оператор 2>; ...<оператор N>;

end;

2.7. Основные операторы

Оператор присваивания:

<переменная> : = <выражения>;

где «: =» – знак присваивания.

Следующие выражения читаются одинаково

x:=2; x:= 2;

y:=d+beta; y:=d+Beta;

Заглавные и прописные буквы в программе интерпретируются одинаково.

Оператор ввода:

readln (<список - ввода>);

где readln – имя оператора ввода; <список - ввода> – список имен переменных, разделенных запятыми.

Например:

readln (a,b,c);

По данному оператору с клавиатуры необходимо ввести значения переменных a, b и c.

По оператору

readln ;

компьютер ожидает нажатия любой клавиши. Используется как последний в программе, чтобы успеть записать результаты вычислений.

Оператор вывода:

writeln (<список - вывода>);

где writeln – имя оператора вывода; <список - вывода> – список переменных вывода, разделенных запятыми.

Оператор

writeln ;

Оператор writeln без списка вывода можно использовать для пропуска пустых строк при оформлении вывода результатов.

В операторе writeln можно использовать формат вывода значений переменных.

Например: writeln (' a = ', a:8:3, ' b = ' , b:3);

Первая цифра (8) после имени переменной вещественного типа определяет количество позиций, выделенных под число, включая знак и десятичную точку, а вторая цифра (3) определяет количество позиций выделенных под дробную часть числа. Цифра, стоящая после имени переменной целого типа, определяет количество позиций, отводимых под число, включая знак.

2.8. Комментарии в программе

В любом месте программы можно записать пояснительный текст – комментарий. Он не обрабатывается во время выполнения программы. Текст комментария ограничен символами { }.

```
...
{ Пояснения к программе – комментарии}
...
```

Комментарии удобно использовать в программе при отладке для временного исключения группы операторов, заключив их в фигурные скобки.

2.9. Примеры типовых задач по теме «Основы программирования»

1. Вычислить и вывести на печать значение функции *y*. Исходные данные x, a и b ввести с клавиатуры.

$$y = (a + 2b) \times \sqrt{b + 2a} \times \frac{1}{\cos x}.$$

Тип алгоритма – ***следование***.

Программа	Пояснения

program primer_1;
uses crt;
var
a, b: integer; y,x: real;
begin
clrscr;
writeln (’ введите x,a, b ’);
readln (x,a, b);
y:=(a+2*b)*sqrt(a+2*b)*(1/cos(x));
writeln (’ y=’, y:8:3);
readln; end.
	Имя программы – primer_1
	Подключение модуля crt
	{Описательная часть}
	
	{Исполнительная часть}
	Функция очистки экрана
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Вычисление функции
	Вывод результатов на экран

2. Даны два целых числа. Найти их полусумму и произведение. Тип алгоритма – ***следование***.

Программа	Пояснения

program primer_2;
uses crt;
var
a, b,p: integer; s: real;
begin
clrscr;
writeln (’ введите a, b ’);
readln (a, b);
s: = (a + b)/2;
p: = a*b;
writeln (’ s = ’, x:8:3, ’ ’, ’p = ’, y:4);
end.
	Имя программы – primer_2
	Подключение модуля crt
	{Описательная часть}
	
	{Исполнительная часть}
	Функция очистки экрана
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Вычисление полусуммы
	Вычисление произведения
	Вывод результатов на экран

Вопросы для самоконтроля

1. Что такое программа?
2. Какие символы могут содержать имена переменных?
3. Какие типы стандартных переменных допустимы на языке Паскаль?
4. Сколько байт памяти занимают переменные типа real?
5. Из каких частей состоит структура программы на Паскале?
6. Что определяет объем памяти, отводимой под каждую переменную?
7. Что такое оператор?

8. Какие типы операторов допустимы на Паскале?
9. Как операторы отделяются друг от друга?
10. Для чего используются комментарии в программе?

Тема 3. Типовые вычислительные процессы. Ветвления

Очень часто встречаются ситуации, когда требуется выбрать между двумя или более вариантами действий в зависимости от заданного условия. Такая алгоритмическая конструкция называется ветвлением.

3.1. Операторы проверки условий и перехода

Строки программы на Паскале не нумеруются. Отдельные строки в программе могут иметь метки, к которым можно переходить.

Метки должны быть описаны в программе с помощью ключевого слова label

N1, N2 ... ,

где N1, N2, ... – метки.

Метками могут быть идентификаторы или целые числа (положительные) в диапазоне 0÷9999.

Оператор перехода по метке:

goto N,

где N – метка.

Фрагмент программы с использованием меток:

```
program pr;
label 3;
var
x, y: real;
begin
3: readln (x, y);
...
go to 3;
...
end.
```

3.2. Логический оператор

if < условия > then P1 [else P2];

[else P2] – необязательная часть оператора, где <условие> – логическое выражение; P1, P2 – простые или составные операторы.

По этому оператору:

если <условие> – «истинно» (true), то выполняется P1, если – «ложно» (false), то выполняется P2. Перед структурой else «;» – не ставится.

Если else – отсутствует и <условие> – «ложно», то управление передается следующему оператору.

Операции отношения

Операция	Название
=	Равно
<>	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Логические операции

Операция	Название
and	И
or	Или

Приведем примеры использования оператора:

1) наибольшее из значений двух переменных вывести на печать:

if a>b then writeln('a=',a) else writeln(' b=',b);

2) в зависимости от значения переменной вычислить значение функции по одной из формул:

if x>=0 then y: = sin(x) else y: = -sin(x);

3) в зависимости от значения переменной выполнить определенные действия:

```
if a<0 then   begin
                b:=2*a; writeln(b);   } P1
              end
```

```
else  begin
          b:=a/2;  goto 3;      } P2
      end;
```

где P1 и P2 – составные операторы;

4) в операторе if можно одновременно проверять несколько условий:

```
if (a > b) and (a > c) then writeln('a – max');
```

при проверке нескольких условий в одном операторе каждое условие записывается в скобках;

5) на языке Паскаль допускается вложенность операторов if.

```
if n>0 then
    if (m div n)>n then m: = m – n
                   else m: = m + n;
```

Структура else – всегда относится к ближайшему оператору if.

Если n > 0 и (m div n)>n – будет выполнено m: = m-n.

Если n > 0, но (m div n)$\leq$n – будет выполнено m: = m + n.

Если $n \leq 0$ – переход к следующему оператору.

3.3. Оператор выбора

Этот оператор предназначен для замены конструкций из вложенных if.

Структура:

```
case N of
N1: P1;
N2: P2;
NN: PN;
[else P;]
end;
```

[else P;] – необязательная часть оператора,

где N – целочисленная переменная, или выражение целого типа; N1, N2, ... NN – возможные значения переменной N; P, P1, P2, ... PN – простые или составные операторы.

По этому оператору:

если значение N = N1, то выполняется P1 (после чего управление передается оператору, следующему за оператором case...of);

если значение N = N2, то выполняется P2 (после чего управление передается оператору, следующему за оператором case...of);

...

если N не принимает ни одного из перечисленных значений, управление передается оператору P;

если структура else отсутствует и N не принимает ни одного из перечисленных значений, управление передается следующему за case...of оператору.

Например:

case N of

1, 2, 5: writeln (’a’);

7 .. 10: writeln (’b’);

end;

1, 2, 5 – перечисляемые значения разделяются запятой.

7 .. 10 – интервал задается начальными и конечными значениями через две точки.

3.4. Примеры типовых задач по теме «Типовые вычислительные процессы. Ветвления»

1. Найти максимальное число среди трех неравных между собой вещественных чисел.

1. *Первый способ решения*

Программа	Пояснения

program prim_max1;
var
a, b, c: real;
begin
writeln (’введите a, b, c’);
readln (a, b, c);
if a>b then
if a>c then writeln (’max - a’)
else writeln (’max - c’)
else
if b<c then writeln (’max - b’)
else writeln (’max - c’) ;
readln;
end.
	Имя программы – prim_max1
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Поиск максимального из трёх чисел

2. *Второй способ решения*

Программа	Пояснения

program prim_max2;
var
a, b, c: real;
begin
writeln (’введите a, b, c’);
readln (a, b, c);
if (a>b) and (a>c) then writeln (’max-a’);
if (b>a) and (b>c) then writeln (’max-b’);
if (c>a) and (c>b) then writeln (’max-c’);
readln;
end.
	Имя программы – prim_max2
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Поиск максимального из трёх чисел

2. Вычислить значение функции:

$$y = \begin{cases} \sin x, 0 < x < 90 \\ 0, x \le 0 \end{cases}$$

Значение аргумента x в градусах ввести с клавиатуры.

Программа	Пояснения

Program prim_3 ;
Uses crt;
var
x, y: real;
begin
clrscr;
writeln (’Введите x’);
readln (x);
if x>90 then writeln(’Функция не определена’)
else begin
if x<0 then y: = 0
else y: = sin (x*pi/180);
writeln (’y = ’,y:8:3);
end;
readln; end.
	Имя программы – prim_3
	
	{Описательная часть}
	
	{Исполнительная часть}
	Очистка экрана
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Выбор варианта решения
	
	Вывод результата

Вопросы для самоконтроля

1. Какие алгоритмические конструкции называются ветвлениями?
2. Для чего используют оператор goto?
3. Почему при использовании нескольких условий в одном операторе if ... then ... else, каждое условие заключается в круглые скобки?
4. Перечислите основные логические операции.

5. Перечислите основные операции отношения.
6. В каких случаях прибегают к использованию оператора case ... of ?

Тема 4. Циклические вычислительные процессы

Процессы, в которых ряд действий повторяется многократно по одним и тем же математическим зависимостям, называются циклическими.

При организации циклов необходимо:

– определить параметр цикла и его начальное значение;
– изменять значение параметра цикла на каждом шаге итерации;
– организовать проверку на выход из цикла.

4.1. Оператор цикла с параметром

Структура оператора:

for i:=N to K do P ;

где i – параметр цикла; N, K – его начальное и конечное значения; P – простой или составной оператор; i,N,K – переменные или константы целого типа.

Шаг изменения параметра цикла i равен 1.

Если K < N, то имеем дело с циклом с отрицательным шагом: (-1)

for i:=N downto K do P;

Оператор цикла с параметром автоматически устанавливает начальное значение параметра цикла, изменяет значение параметра на шаг, организовывает проверку на выход из цикла. В операторе for to do перечисленные действия выполняются автоматически. В остальных операторах цикла эти действия необходимо организовать составителю программы.

Оператор for применяют в тех случаях, когда значения параметра цикла целые и меняются с шагом +1, -1.

4.2. Оператор цикла с постусловием

Структура оператора:

repeat

p1;p2 ;...pn ;

until <условие>;

где p1; p2 ; ... pn ; – любые операторы, образующие тело цикла.

По этому оператору выполняется «тело цикла», а затем проверяется <условие>, если оно не выполнилось, цикл повторяется. И так до тех пор, пока <условие> не будет выполнено. Необходимо помнить: если <условие> выполнилось с первого раза, цикл будет пройден один раз.

4.3. Оператор цикла с предусловием

Структура оператора:

while <условие> do P;

где *P* – простой или составной оператор.

По этому оператору проверяется <условие> и, если оно выполняется, то выполняется *P*, после чего опять проверяется <условие> и т. д.

Итак, *P* выполняется до тех пор, пока выполняется <условие>.

Если условие ни разу не выполнилось, *P* игнорируется, управление передается следующему оператору.

4.4. Вложенные циклы

В теле любого оператора цикла могут находиться другие операторы цикла. При этом цикл, содержащий в себе другой, называется *внешним*, а цикл, находящийся в теле первого, – *внутренним* (*вложенным*). Правила организации внешнего и внутреннего циклов такие же, как и для простых циклов.

При организации вложенных циклов необходимо помнить, что параметр внешнего цикла меняется медленнее, чем параметр внутреннего. При одном значении параметра внешнего цикла параметр внутреннего пробегает все свои возможные значения.

4.5. Оператор прерывания цикла

Для досрочного прерывания цикла можно использовать оператор goto или стандартную процедуру break.

Рассмотрим фрагмент:

```
for i:=1 to n do
begin
p:=p* i;
if p>100 then break ;
writeln('p=', p);
end;
```

Цикл будет прерван при достижении условия $p>100$ по процедуре break. Если условие не выполнится, цикл будет пройден n раз.

4.6. Примеры типовых задач по теме «Циклические вычислительные процессы»

1. Вычислить сумму ряда:

$$S = 1 + 1/2 + 1/3 + ... + 1/50.$$

Выделим переменную для накапливания суммы – sum. Значение этой переменной необходимо предварительно обнулить. В программе используется оператор цикла с параметром.

Программа	Пояснения
program prim_cikl1;	Имя программы – prim_ cikl1
var	{Описательная часть}
i: integer; sum: real;	
begin	{Исполнительная часть}
sum: =0;	
for i: = 1 to 50 do	Начало цикла
sum: = sum + 1/i;	Вычисление суммы ряда
Writeln (‘ сумма = ‘, sum);	Вывод результата на экран
end.	

2. Вычислить значение функции:

$$y = a\sin(x).$$

При $x = \overline{0.1}$ с шагом $\Delta x = 0.2$ значение переменной *a* ввести с клавиатуры.

В программе используется оператор цикла с постусловием.

Программа	Пояснения
program cikl_2;	Имя программы – prim_ cikl2
var	{Описательная часть}
y,a,x:real;	
begin	{Исполнительная часть}
writeln(‘Введите a’);	Вывод сообщения на экран
readln(a);	Ввод данных с клавиатуры
x:=0;	Определение начального значения
repeat	Начало цикла
y:=a*sin (x);	Вычисление значения функции
writeln(‘y=’,y:8:3, ‘ x=’,x:8:3); x:=x+0.2;	Вывод результата на экран
until x>1;	Увеличение параметра цикла на шаг
readln; end.	Проверка на выход из цикла

Действия, которые подчеркнуты в программе, необходимы для организации цикла.

3. Вычислить сумму ряда:

$$y = \frac{\sin x}{1!} - \frac{\sin 2x}{2!} + \frac{\sin 3x}{3!} - \ldots + (-1^{n+1}) \frac{\sin nx}{n!},$$

где n! = 1 * 2 * ... * n

Значение аргумента *x* и количество итераций *n* ввести с клавиатуры.

Введем дополнительную переменную для вычисления n! – f.

В программе используется оператор цикла с предусловием.

Программа	Пояснения
program cikl_3;	Имя программы – prim_ cikl3
uses crt;	{Описательная часть}
var x,y : real; i,n,f : integer;	
begin	{Исполнительная часть}
writeln (‘ введите x, n ‘);	Вывод сообщения на экран
readln(x, n);	Ввод данных с клавиатуры
y: =0; f: =1; i:=1;	Определение начальных значений
while i<=n do begin	Начало цикла
f:=f*I;	Подсчет n!
y:=y+sin(i*x)/f;	Вычисление значения функции
i:=i+1;	Увеличение параметра цикла на шаг
end;	
writeln (‘ y = ‘, y:8:3);	Вывод результата на экран
readln; end.	

Действия, которые подчеркнуты в программе, необходимы для организации цикла.

1. Составить программу расчета таблицы значений функции f(x) на интервале a<=x<=b в *n* равностоящих точках. Границы интервала *a, b* и количество точек *n* ввести с клавиатуры. Результаты вывести на печать.

$$f(x) = (1 - e^{-x}) \times \sin(4\pi x)$$

Найти сумму положительных значений функции f(x) на заданном интервале. Формула для расчета шага изменения аргумента:

$$dx = (b - a)/(n - 1)$$

Составим таблицу идентификаторов:

Наименование переменной	Обозначения в программе
Функция	y
Аргумент	x
Количество расчетных точек	n
Начальное значение аргумента	a
Конечное значение аргумента	b
Шаг изменения аргумента	dx
Сумма положительных значений функции	s

Программа	Пояснения

program cikl_4;
var
y,x,a,b,dx,s : real; i,n : integer;
begin
writeln(‘Введите a,b,n’);
readln(a,b,n);
dx:=(b-a)/(n-1);
s:=0; x:=a;
for i:=1 to n do
begin
y:=(1-exp(-x)) * sin(4*pi*x);
writeln(‘y=’,y:8:3);
x:=x+dx;
if y>0 then s:=s+y;
end;
writeln(‘Сумма положительных значений s=’,s:8:3);
readln;
end.
	Имя программы – prim_ cikl4
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод данных с клавиатуры
	Шаг изменения аргумента *x*
	
	Начало цикла
	Вычисление значения функции
	Вывод значений функции на экран
	Увеличение параметра цикла на шаг
	Подсчет суммы
	Конец цикла
	Вывод суммы на экран

Цикл организован с помощью оператора цикла с параметром for ... to ... do.

2. Рассмотрим задачу вывода на экран таблицы умножения, решение которой предполагает использование вложенных циклов.

Программа	Пояснения

program cikl_5;
var
i,j : byte;
begin
writeln(‘ Таблица умножения ’);
for i:=2 to 9 do
begin
for j:=1 to 9 do
writeln(i , ‘ * ‘ , j ,‘ = ’, i*j);
writeln;
end;
readln; end.
	Имя программы – prim_ cikl5
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Внешний цикл
	
	Внутренний цикл
	Вывод значений на экран
	Вывод на экран пустой строки
	Конец внутреннего и внешнего циклов

Вопросы для самоконтроля

1. Что такое цикл?
2. Какие циклы можно организовать на языке Паскаль?
3. В каких случаях предпочтительнее использовать оператор for ...to ... do для организации циклов?

4. Какой тип должна иметь переменная, которая является параметром цикла в операторе for ...to ... do ?
5. Сколько циклов будет пройдено, если в операторе while ... do условие не выполнилось с первого раза?
6. Сколько циклов будет пройдено, если в операторе repeat ... until условие выполнилось с первого раза?
7. Что такое вложенные циклы?
8. Как можно досрочно прервать цикл?

Тема 5. Операции с индексированными переменными

Массив – это поименованный набор однотипной информации.

Массив объединяет элементы одного типа данных. Всему набору данных присваивают общее имя – имя массива. Каждый элемент массива идентифицируется с помощью индекса, определяющего место этого элемента в общем наборе.

Данные в массиве сохраняются, как и в случае использования обычных неиндексированных переменных, только до конца работы программы.

Характеристики массива:

– тип – общий тип всех элементов массива;
– размерность (ранг) – количество индексов массива;
– диапазон изменения индексов – определяет количество элементов в массиве.

5.1. Массивы одномерные

Вектор (одномерный массив) – это массив, в котором элементы нумеруются одним индексом.

Описание одномерных массивов:

```
var
a: array[1...n] of <тип>;
```

где a – имя массива; n – максимальное количество элементов массива; <тип> – тип элементов массива.

Например:

```
var
mas: array [1...10] of real;
```

Описан массив с именем mas, содержащий 10 элементов вещественного типа.

Каждый элемент массива определяется с помощью индекса, стоящего справа от имени в квадратных скобках.

a [i], a [i+2], a [9].

Индекс может быть переменной, константой, либо арифметическим выражением целого типа. Размер массива должен быть задан в явном виде (как в предыдущем фрагменте) или через константу (const).

Например:

```
const n = 10;
var mas: array [1...n] of real;
```

5.2. Двумерные массивы. Матрицы

Если в массиве хранится таблица значений, то такой массив называют двумерным, а его элементы нумеруются двумя индексами – номером строки и номером столбца, на пересечении которых находится данный элемент.

В памяти компьютера все элементы массива занимают одну непрерывную область. Двумерный массив располагается в памяти по строкам.

Двумерный массив можно представить в виде матрицы:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

Описание двумерного массива:

```
var
a :array[1..n ,1..m] of <тип>;
```

где a – имя массива; n, m – количество строк и столбцов в массиве; <тип> – тип элементов массива.

Количество элементов в массиве – n×m.

Например:

```
var
mas: array [1...10,1..10] of real;
```

Описан массив с именем mas, содержащий 100 элементов вещественного типа (10 строк и 10 столбцов).

Размер массива должен быть задан в явном виде (как в предыдущем фрагменте) или через const.

Например:

```
const
n = 5, m=5;
var mas: array[1...n,1..m] of real;
```

Описан массив mas, содержащий 25 элементов целого типа (5 строк и 5 столбцов).

Каждый элемент массива определяется с помощью двух индексов, стоящих справа от имени в квадратных скобках.

a[i, j] – элемент, стоящий на пересечении *i*-й строки и *j*-го столбца;

a[i, i] – элементы главной диагонали;

a[i, 2] – элементы второго столбца.

Индекс может быть – переменной, константой, арифметическим выражением целого типа.

Если количество строк равно количеству столбцов, матрица называется *квадратной*.

Обработка двумерных массивов производится при изменении индексов элементов.

Все элементы главной диагонали квадратной матрицы удовлетворяют условию:

i = j (номер строки равен номеру столбца).

Все элементы побочной диагонали квадратной матрицы удовлетворяют условию:

i + j = n + 1(n – количество строк и столбцов).

Элементы, расположенные над главной диагональю, удовлетворяют условию:

i < j (номер строки строго меньше номера столбца).

Элементы, расположенные под главной диагональю, удовлетворяют условию:

i > j (номер строки строго больше номера столбца).

Ввод элементов двумерного массива

```
for i:=1 to n do
for j:=1 to m do          Вложенные циклы
readln(a[i]);
```

Данный фрагмент позволит ввести элементы массива по строкам.

Для ввода элементов массива по столбцам достаточно в предыдущем фрагменте поменять местами внутренний и внешний циклы.

Вывод элементов двумерного массива

```
for i:=1 to n do
begin
for j:=1 to n do
write(a[I,j],‘ ‘);
writeln; end;
```

Данный фрагмент позволит вывести элементы массива в виде матрицы.

5.3. Примеры типовых задач по теме «Массивы одномерные»

1. Составить программу подсчета суммы и произведения элементов одномерного массива вещественных чисел. Обозначим через s – сумму элементов массива, p – произведение.

Программа	Пояснения

program mas_1;
uses crt;
var a : array [1...100] of real;
n, I : integer; p, s : real;
begin
clrscr;
s: = 0; p: =1;
writeln (‘Введите размер массива
n<=100’);
readln (n);
writeln (‘введите элементы массива’);
for i:=1 to nk do
readln (a[i]);
for i:=1 to nk do
begin
s:= s + a[i];
p:= p * a[i];
end;
writeln (‘Сумма = ‘,s:8:3,’
Произведение =’,p:8:3);
readln; end.
	Имя программы – mas_1
	{Описательная часть}
	
	{Исполнительная часть}
	Функция очистки экрана
	Определение начальных значений
	Вывод сообщения на экран
	
	Ввод размера массива
	Начало цикла
	Ввод элементов массива
	
	Вычисление суммы и произведения
	элементов массива
	
	Вывод результатов на экран

При накапливании суммы переменную s необходимо обнулить, а при накапливании произведения переменной p присвоить 1.

2. Дан массив a(n). Найти максимальный элемент массива и определить его номер. Введем обозначения:

a – имя массива;

n – количество элементов в массиве;

i – индекс элементов массива;

max – максимальный элемент массива;

nmax – номер максимального элемента.

Программа	Пояснения

program mas_2;
var
a: array [1...100] of real;
i,n,nmax: integer; max: real;
begin
writeln (‘Введите размер массива n<=100’);
readln (n);
writeln (‘Введите элементы массива’);
for i:=1 to n do
readln (a[i]);
max:=a[1];
nmax:=1;
for i:=1 to n do
If a[i]>max then begin
max:=a[i]; nmax:=i;
end;
writeln (‘max =’, max, ‘ N=’, nmax);
readln; end.
	Имя программы – mas_2
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	
	Ввод размера массива
	Начало цикла
	
	Ввод элементов массива
	
	За максимальный принимается первый элемент массива
	Поиск максимального элемента
	
	Вывод результатов на экран

3. Задан массив целых чисел. Отсортировать элементы одномерного массива по возрастанию.

Существует много способов сортировки элементов массива.

Введем обозначения:

a – имя массива;

n – количество элементов в массиве;

i – индекс элементов массива;

f – флажок, f = 0 – массив отсортирован, f = 1 – массив пока не отсортирован;

p – дополнительная переменная для обмена местами соседних элементов.

Программа	Пояснения
program mas_3;	Имя программы – mas_3
var	{Описательная часть}
a: array [1...100] of integer;	
f, n, i,p : integer;	{Исполнительная часть}
begin	Вывод сообщения на экран
writeln (‘Введите размер массива n<=100’);	Ввод размера массива
readln (n);	Вывод сообщения на экран
writeln (‘Введите элементы массива’);	Ввод элементов массива
for i:=1 to n do readln (a[i]);	Внешний цикл
repeat	
f:=0;	Внутренний цикл
for i:=1 to n-1 do	Сортировка элементов
if a[i]>a[i+1] then	
begin	
p: =a[i];	
a[i]:=a[i+1];	
a[i+1]:=p; f:=1;	
end;	
until f=0;	Вывод на экран отсортированного
for i:=1 to n do writeln (a[i]);	массива
readln; end.	

В данном алгоритме используются вложенные циклы.

Внутренний цикл (for...to...do) – позволяет один раз пройтись по массиву, сравнивая соседние элементы. В случае если предыдущий элемент (a[i]) окажется больше последующего (a[i+1]), элементы меняются местами через дополнительную переменную p.

Внешний цикл (repeat...until) – повторяется до тех пор, пока массив не будет отсортирован.

В программе применяется переменная – флажок.

f = 1 – признак того, что два соседних элемента менялись местами, массив еще не упорядочен.

f = 0 – признак того, что массив упорядочен.

4. Задан одномерный массив целых чисел. Найти количество нечетных чисел среди элементов массива.

Введем обозначения:

a – имя массива;

n – количество элементов в массиве;

i – индекс элементов массива;

kol – количество нечетных чисел.

Программа	Пояснения

program mas_4;
var
a: array [1...100] of integer;
n, i,kol : integer;
begin
writeln (‘Введите размер массива n<=100’);
readln (n);
writeln (‘Введите элементы массива’);
for i:=1 to n do
readln (a[i]);
kol:=0;
for i:=1 to n do
if odd(a[i]) then kol:=kol+1;
writeln(‘Количество нечетных = ’, kol);
readln; end.
	Имя программы – mas_4
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод размера массива
	Вывод сообщения на экран
	Ввод элементов массива
	
	Подсчет количества нечетных чисел
	Вывод результата на экран

Выражение odd(a[i]) будет «истинным», если элемент массива – нечетное число.

5.4. Примеры типовых задач по теме «Двумерные массивы»

1. Составить программу подсчета суммы и произведения элементов двумерного массива вещественных чисел.

Введем обозначения:

a – имя массива;

n – количество строк в массиве;

m – количество столбцов в массиве;

i,j – индексы элементов массива;

s – сумма элементов массива;

p – произведение элементов массива.

Программа	Пояснения
program matr_1; var a:array [1...10, 1...10] of real; i, j, n, m: integer; s,p :real; begin writeln(‘ Введите n<=10 , m<=10 ’); readln (n,m); writeln(‘ Введите элементы массива по строкам ’);	Имя программы – matr_1 {Описательная часть} {Исполнительная часть} Вывод сообщения на экран Ввод размера массива Вывод сообщения на экран

for i:=1 to n do
  for j:=1 to m do
  readln(a[i, j]);
s:=0; p:=1;
for i:=1 to n do
for j:=1 to m do
begin
s:=s + a[i,j]; p:=p * a[i,j];
end;
writeln(‘s=’,s:8:3, ‘p=’,p:8:3);
readln; end.
	
	Ввод элементов массива по строкам
	
	Вложенные циклы
	
	Подсчет суммы и произведения
	
	Вывод результатов на экран

2. Задана целочисленная квадратная матрица. Определить, является ли она симметричной относительно главной диагонали.

Введем обозначения:

a – имя массива;

n – количество строк и столбцов в массиве;

i,j – индексы элементов массива;

f – признак симметричности матрицы, допустим, f = 0 – матрица симметрична, f = 1 – матрица не симметрична.

Программа	Пояснения

program matr_2;
var
a: array [1...10, 1...10] of integer;
i, j, n, f : integer;
begin
writeln(‘ Введите n<=10 ’);
readln (n);
writeln(‘ Введите элементы массива по строкам ’);
for i:=1 to n do
for j:=1 to n do
readln(a[i,j]);
f:=0
for i:=1 to n do
for j:=i to n do
if a[i,j]<>a[j,i] then f:=1;
if f=0 then writeln(‘ матрица симметрична’)
  else writeln (‘матрица не симметрична’);
readln; end.
	Имя программы – matr_2
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод размера массива
	Вывод сообщения на экран
	
	Ввод элементов массива
	
	Определение симметричности матрицы
	
	Вывод результатов на экран

При решении этой задачи сравниваются элементы, расположенные над главной диагональю (a[i,j]), с элементами, расположенными под

главной диагональю (a[j,i]), и в случае их равенства матрица считается симметричной.

3. Задана матрица вещественных чисел. Образовать одномерный массив, где каждый *j*-й элемент равен произведению элементов *j*-го столбца двумерного массива.

Введем обозначения:

a – имя двумерного массива;

b – имя одномерного массива;

n – количество строк в массиве;

m – количество столбцов в массиве;

i,j – индексы элементов массива.

Программа	Пояснения

program matr_3;
var
a: array [1...10, 1...10] of integer;
b: array [1..10] of integer;
i, j, n, f : integer;
begin
writeln(‘ Введите n<=10 , m<=10 ’);
readln (n,m);
writeln(‘ Введите элементы массива по строкам ’);
for i:=1 to n do
for j:=1 to m do
readln (a[i, j]);
writeln(‘ Исходная матрица ’);
for i:=1 to n do
begin
for j:= 1 to n do
write(a[I,j] , ‘ ‘); writeln;
end;
writeln(‘ Образованный одномерный массив ’);
for j:=1 to m do
begin
b[j]:=1;
for i:=1 to n do
b[j]:= b[j] * a[i, j];
write(b[j] , ‘ ‘);
end;
readln; end.
	Имя программы – matr_3
	{Описательная часть}
	Описание двумерного массива
	Описание одномерного массива
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод размера массива a
	Вывод сообщения на экран
	
	Ввод элементов двумерного массива
	
	Вывод на экран исходного массива в виде матрицы
	
	Формирование и вывод на экран одномерного массива b

4. Составить программу нахождения максимального среди элементов двумерного массива вещественных чисел. Определить номер строки и номер столбца, на пересечении которых был найден максимальный элемент.

Введем обозначения:

a – имя двумерного массива;

n – количество строк в массиве;

m – количество столбцов в массиве;

i,j – индексы элементов массива;

kmax, lmax – номер строки и номер столбца, на пересечении которых находится максимальный элемент.

Программа	Пояснения

program matr_4;
var
a: array [1...10, 1...10] of real;
i, j, n, m , kmax , lmax: integer;
max :real;
begin
writeln(‘ Введите n<=10 , m<=10 ’);
readln(n,m);
writeln(‘ Введите элементы массива по строкам ’);
for i:=1 to n do
for j:=1 to m do
readln(a[i,j]);
max:=a[1,1]; kmax:=1; lmax:=1;
for i:=1 to n do
for j:=1 to m do
if a[i,j]>max then
begin
max:=a[i,j];
kmax:=i;
lmax:=j;
end;
writeln(‘max=’, max:8:3);
writeln(‘строка - ‘, kmax,’ столбец - ’,lmax);
readln; end.
	Имя программы – matr_4
	{Описательная часть}
	
	{Исполнительная часть}
	Вывод сообщения на экран
	Ввод размера массива
	Вывод сообщения на экран
	
	Ввод элементов массива по строкам
	
	Поиск максимального элемента
	
	Вывод результатов на экран

Вопросы для самоконтроля

1. Что такое массив?
2. Будут ли сохранены данные массива в памяти компьютера после окончания работы программы?
3. Что такое индекс массива?
4. Какого типа должны быть переменные, являющиеся индексами массива?
5. Как можно задавать размер массива?
6. Может ли реальное количество элементов в массиве быть меньше, чем указано при описании?
7. Может ли реальное количество элементов в массиве быть больше, чем указано при описании?
8. В чем состоит разница между одномерными и двумерными массивами?
9. В каких случаях целесообразно создавать двумерные массивы?
10. Как расположены элементы двумерного массива в памяти компьютера?
11. Как определить количество элементов в двумерном массиве?
12. Как определить размер памяти в байтах, выделенной под хранение двумерного массива?
13. Какая матрица называется квадратной?
14. Что характеризует элементы, расположенные на главной диагонали, над и под главной диагональю?

Тема 6. Подпрограммы

Если в программе имеется несколько одинаковых фрагментов, то возникает вопрос: нельзя ли оформить повторяющийся фрагмент в виде отдельного блока, а затем обращаться к нему несколько раз. Аналогичная идея возникает при отладке больших программ – если разбить программу на отдельные блоки, то отладить ее по частям будет проще. На языке Паскаль подпрограммы реализуются в виде функций или процедур.

6.1. Функции и процедуры

Функции и процедуры реализуют принципы структурного программирования.

К функциям и процедурам обращаются, если необходимо:

– разбить большую задачу на несколько меньших по объему и сложности задач;
– уменьшить объем программы за счет выделения типовых программных действий в функции или процедуре;
– создать программные модули, которые могут быть использованы и в других программах.

Структура программы, содержащей процедуру (функцию):

program <имя>;
<описательная часть основной прогаммы>;
procedure <имя процедуры>;
(***function*** <имя функции>;)
<описательная часть процедуры/функции>;
begin
<исполнительная часть процедуры/функции>;
end;
begin
<исполнительная часть основной программы>;
end.

Процедуры (функции) могут содержать любые операторы языка Паскаль и повторяют структуру основной программы, только текст процедуры (функции) заканчивается «;».

6.2. Процедуры

Процедура – это независимая именованная часть программы, которую после однократного описания можно многократно вызывать по имени.

Первая строка любой процедуры:

procedure <имя>(<список формальных параметров>);

где <имя> – уникальный идентификатор, строится по правилам составления имен простых переменных; <список формальных параметров> – список имен переменных с указанием их типа, передающих информацию в процедуру и возвращающих результаты в основную программу.

Обращение к процедуре из основной программы:

<имя> (<список формальных параметров>);

где <список фактических параметров> – список переменных или констант, разделенных запятыми.

Например:

procedure prim(x:real; var s:integer; z,y:integer);

Описан заголовок процедуры, где x, z, y – передают информацию в процедуру; s – передает информацию в процедуру и возвращает результат – в основную (можно сказать, процедура изменяет значение s, хотя значение s в основной программе может быть до процедуры и не определено).

Чтобы формальный параметр возвращал результат в основную, его необходимо описать с использованием служебного слова var.

Примеры обращения к процедуре из основной программы:

prim(x, s, z, y); – имена фактических параметров x, s, z, y совпадают с именами формальных;

prim(a, s1, b, 5); – имена фактических параметров a, s1, b не совпадают с именами формальных, а последний фактический параметр является числовой константой.

Формальные и фактические параметры должны совпадать по *типу*, *количеству* и *порядку* следования. Фактические параметры должны быть описаны в основной программе. Они называются *глобальными* переменными и доступны как в основной программе, так и в процедуре. Формальные параметры отдельно в процедуре не описываются. Время существования глобальных переменных – от начала и до конца работы программы.

Промежуточные переменные, которые используются только в процедуре, описываются в процедуре и называются *локальными*. Локальные переменные не доступны в основной программе. Время существования локальных переменных – от начала и до конца работы процедуры, в которой они описаны.

6.3. Вложенные процедуры. Директива forward

Если одна подпрограмма использует другую, а та, в свою очередь, использует первую, возникает проблема размещения этих процедур в программе. Какую из них поместить в программе первой? В этих случаях прибегают к директиве forward.

Например:

procedure pp1(a,b:integer):forward;

```
procedure pp2(c: real); {Вторая процедура}
var x,y:integer;
begin
...
pp1(x, y);
end;
procedure pp1; {Первая процедура}
var z:real;
begin
...
pp2 (z);
end; ...
```

6.4. Функции

Если результатом подпрограммы является только одно значение, то имеет смысл оформить такую подпрограмму в виде функции.

Первая строка любой функции:

function <имя >(<список формальных параметров>):<тип>;

где <список формальных параметров> – список имен переменных с указанием их типа, передающих информацию в функцию; <имя> – возвращает результат в основную программу; <тип> – тип возвращаемого результата.

Обращение из основной программы к функции возможно в структуре какого-либо оператора, например оператора присваивания:

<переменная>:= <имя> (список фактических параметров);

Функция возвращает в основную программу только один результат через переменную, являющуюся именем функции. Эта переменная не должна быть описана в основной программе. В описательной части функции этой переменной должно быть присвоено какое-либо значение, иначе функция не возвращает никакого значения.

6.5. Примеры типовых задач на тему «Подпрограммы»

1. Задан одномерный массив вещественных чисел. Найти максимальный элемент массива и определить его номер.

Составим программу с использованием двух процедур:

1) процедура ввода элементов массива;

2) процедура поиска максимального элемента и определение его номера.

Для передачи массива в процедуру введем пользовательский тип mas.

Программа	Пояснения

program PP;
type
mas=array[1..100] of integer;
var
a:mas; max:real; n,max,kmax:integer;
procedure input_mas(n:integer ; var a:mas);
var i:integer;
begin
for i:=1 to n do readln(a[i]);
end;
procedure max_m(n:integer; a:mas; var max,kmax:integer);
var i:integer;
begin
max:=a[1];kmax:=1;
for i:=1 to n do
if a[i]>max then
begin max:=a[i]; kmax:=i; end;
end;
begin
writeln(‘ Введите n<=100’) ;
readln(n);
writeln (‘ Введите элементы массива ‘);
input_mas(n,a);
max_m(n,a,max,kmax);
writeln(‘max=’,max,‘kmax=’,kmax);
readln; end.
	Имя программы – PP
	{Описательная часть программы}
	Пользовательский тип данных
	
	Глобальные переменные
	Текст первой процедуры
	{Описательная часть процедуры}
	{Исполнительная часть процедуры}
	Ввод элементов массива
	
	Текст второй процедуры
	
	{Описательная часть процедуры}
	{Исполнительная часть процедуры}
	
	Поиск максимального элемента
	
	{Исполнительная часть программы}
	Вывод сообщения на экран
	Ввод размера массива
	Вывод сообщения на экран
	Обращение к первой процедуре
	Обращение ко второй процедуре
	Вывод результатов на экран

2. Составить программу вычисления n! с использованием функции, в которую вынесем подсчет факториала.

n! = 1 * 2 * 3 * ... * n

Введем обозначения:

n, k – размер и индекс массива, глобальные переменные;

p, i – значение факториала и индекс массива в функции, локальные переменные;

fun – имя функции, возвращает результат в основную программу.

Программа	Пояснения

program factorial;
var
n,k:integer;
function fun(k:integer):integer;
var p,i:integer;
begin
p:=1;
for i:=1 to k do
p:=p*i;
fun:=p;
end;
begin
writeln(‘введите n’);
readln(n);
for k: = 1 to n do
writeln(fun(k));
readln; end.
	Имя программы – factorial
	{Описательная часть программы}
	Глобальные переменные
	Текст функции
	{Описательная часть функции}
	{Исполнительная часть функции}
	
	{Описательная часть процедуры}
	{Исполнительная часть процедуры}
	
	{Исполнительная часть программы}
	Вывод сообщения на экран
	Ввод размера массива
	
	Обращение к функции

Имя функции в теле функции в операторах присваивания может стоять только слева от знака присваивания «:=», поэтому использована дополнительная переменная p для подсчета произведения.

3. Задан двумерный массив целых чисел. Определить количество четных чисел в массиве.

Составим программу с использованием двух процедур и функции:

– процедура ввода элементов массива input_mas;
– процедура вывода двумерного массива в виде матрицы print_mas;
– функция подсчета количества четных чисел kol_mas.

Программа	Пояснения

program pp_1;
type
mas=array[1..10,1..10] of integer;
var
a:mas; n,m,i,j:integer ;
procedure input_mas(n,m:integer; var a:mas);
begin
for i:=1 to n do
for j:=1 to n do readln(a[i,j]);
end;
function kol_mas(n,m:integer; a:mas):integer;
	Имя программы – pp_1
	{Описательная часть программы}
	Глобальные переменные
	
	Процедура ввода матрицы
	
	{Исполнительная часть процедуры}
	
	Функция подсчета количества четных чисел

var
k:integer ;
begin
kol:=0;
for i:=1 to n do
for j:=1 to n do
if not odd(a[i,j]) then k:=k+1;
kol:=k;
end;
procedure print_mas(n,m:integer;
a:mas);
begin
for i:=1 to n do
begin
for j:=1 to n do write(a[i,j],‘ ‘);
writeln;
end;
end;
begin
writeln(‘Введите n<=10 , m<=10’);
readln (n,m);
writeln(‘Введите элементы массива
по строкам’);
input_mas(n,m,a);
writeln(‘Исходная матрица’);
print_mas(n,m,a);
writeln(‘Количество нечетных
чисел =’,kol_mas(n,m,a));
readln; end.
	{Описательная часть функции}
	
	{Исполнительная часть функции}
	
	Процедура вывода матрицы
	
	{Исполнительная часть процедуры}
	
	{Исполнительная часть программы}
	Вывод сообщения на экран
	Ввод размера массива
	Вывод сообщения на экран
	
	Обращение к процедуре ввода массива
	
	Обращение к процедуре вывода
	массива
	Обращение к функции подсчета
	количества четных чисел в массиве

Выражение not odd(a[i,j]) истинно, если элемент массива – четное число.

Вопросы для самоконтроля

1. В каких случаях прибегают к построению подпрограмм?
2. В каком месте программы располагаются функции или процедуры?
3. Что такое процедура?
4. Какова структура процедуры?
5. Как передается информация в процедуру?
6. Каким образом возвращаются результаты работы процедуры в основную программу?

7. Какое соответствие должно быть между формальными и фактическими параметрами?
8. Какие переменные называются глобальными? Время существования глобальных переменных?
9. Какие переменные называются локальными? Время существования локальных переменных?
10. Когда используют директиву forward?
11. В каких случаях целесообразно прибегать к построению функций?
12. Как передается информация в функцию?
13. Каким образом возвращается результат работы функции в основную программу?

Тема 7. Языки программирования высокого уровня

7.1. Эволюция языков программирования

Развитие вычислительной техники сопровождается созданием новых и совершенствованием существующих средств общения программистов с ЭВМ.

Под языком программирования понимают правила представления данных и записи алгоритмов их обработки, которые автоматически выполняются ЭВМ. В более абстрактном виде язык программирования является средством создания программных моделей объектов и явлений. К настоящему времени созданы десятки различных языков программирования от самых примитивных до близких к естественному языку человека. Чтобы понимать тенденции развития языков программирования, нужно знать движущие силы их эволюции. Для выяснения этого вопроса будем рассматривать язык программирования с различных точек зрения. Во-первых, язык программирования является инструментом программиста для создания программ. Для создания хороших программ нужны хорошие языки программирования. Поэтому одной из движущих сил эволюции языков программирования является стремление разработчиков к созданию совершенных программ.

Во-вторых, процесс разработки программы можно сравнивать с промышленным производством, в котором определяющими факторами являются производительность труда коллектива программистов, себестоимость и качество программной продукции. Создаются различные техно-

логии разработки программ (структурное, модульное, объектно-ориентированное программирование и другие), которые должны поддерживаться языками программирования. Поэтому второй движущей силой эволюции языков программирования является стремление к повышению эффективности процесса производства программной продукции.

В-третьих, программы можно рассматривать как аналог радиоэлектронных устройств обработки информации, в которых вместо радиодеталей и микросхем используют конструкции языков программирования (элементная база программы). Как и электронные устройства, программы могут быть простейшими (уровня детекторного приемника) и очень сложными (уровня автоматической космической станции), при этом уровень инструмента должен соответствовать сложности изделия. Кроме того, человеку удобнее описывать моделируемый объект в терминах предметной области, а не языком цифр. Поэтому третьей движущей силой, ведущей к созданию новых, специализированных, ориентированных на проблемную область и более мощных языков программирования, является увеличение разнообразия и повышение сложности задач, решаемых с помощью ЭВМ.

В-четвертых, совершенствование самих ЭВМ приводит к необходимости создания новых и более совершенных языков программирования.

7.2. Классификация языков программирования

Первые универсальные языки

Обратимся к истокам развития вычислительной техники. Вспомним самые первые компьютеры и программы для них. Это была эра программирования непосредственно в машинных кодах, а основным носителем информации были перфокарты и перфоленты. Программисты обязаны были знать архитектуру машины досконально. Программы были достаточно простыми, что обусловливалось, во-первых, весьма ограниченными возможностями этих машин, и, во-вторых, большой сложностью разработки и, главное, отладки программ непосредственно на машинном языке.

Со времени создания первых программируемых машин человечество придумало уже более восьми с половиной тысяч языков программирования. Каждый год их число пополняется новыми. Некоторыми языками умеет пользоваться только небольшое число их собственных

разработчиков, другие становятся известны миллионам людей. Профессиональные программисты иногда применяют в своей работе более десятка разнообразных языков программирования.

Создатели языков по-разному толкуют понятие ***язык программирования.*** К наиболее распространенным утверждениям, признаваемым большинством разработчиков, относятся следующие.

Функция*:* язык программирования предназначен для написания компьютерных программ, которые применяются для передачи компьютеру инструкций по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами.

Задача: язык программирования отличается от естественных языков тем, что предназначен для передачи команд и данных от человека компьютеру, в то время как естественные языки используются для общения людей между собой. Можно обобщить определение «языков программирования» – это способ передачи команд, приказов, четкого руководства к действию; тогда как человеческие языки служат также для обмена информацией.

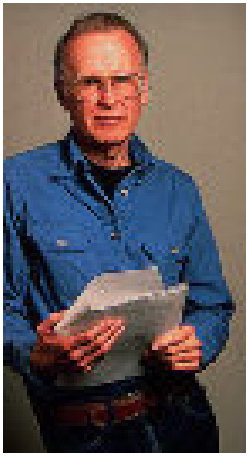
Исполнение: язык программирования может использовать специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Ассемблер

Первым значительным шагом представляется переход к языку ассемблера (assembler). Не очень заметный, казалось бы, шаг – переход к символическому кодированию машинных команд – имел на самом деле огромное значение. Программисту не надо было больше вникать в хитроумные способы кодирования команд на аппаратном уровне. Появилась также возможность использования макросов и меток, что также упрощало создание, модификацию и отладку программ. Появилось даже некое подобие переносимости – существовала возможность разработки целого семейства машин со сходной системой команд и некоего общего ассемблера для них, при этом не было нужды обеспечивать двоичную совместимость.

Здесь впервые в истории развития программирования появились два представления программы: в исходных текстах и в откомпилированном виде.

Фортран



Джон Бэкус

В 1954 году в недрах корпорации IBM группой разработчиков во главе с Джоном Бэкусом (John Backus) был создан язык программирования Fortran.

Значение этого события трудно переоценить. Это первый язык программирования высокого уровня. Впервые программист мог по-настоящему абстрагироваться от особенностей машинной архитектуры. Ключевой идеей, отличающей новый язык от ассемблера, была концепция подпрограмм. Язык Фортран использовался (и используется по сей день) для научных вычислений. Появление Фортрана было встречено еще более яростной критикой, чем внедрение ассемблера. Программистов пугало снижение эффективности программ за счет использования промежуточного звена в виде компилятора. И эти опасения имели под собой основания: действительно, хороший программист, скорее всего, при решении какой-либо небольшой задачи вручную напишет код, работающий быстрее, чем код, полученный как результат компиляции. Через некоторое время пришло понимание того, что реализация больших проектов невозможна без применения языков высокого уровня. Мощность вычислительных машин росла, и с тем падением эффективности, которое раньше считалось угрожающим, стало возможным смириться. Преимущества же языков высокого уровня стали настолько очевидными, что побудили разработчиков к созданию новых языков, все более и более совершенных.

Cobol

В 1960 году был создан язык программирования Cobol.

Он задумывался как язык для создания коммерческих приложений, и он стал таковым. На Коболе написаны тысячи прикладных коммерческих систем. Отличительной особенностью языка является возможность эффективной работы с большими массивами данных, что характерно именно для коммерческих приложений. Популярность Кобола столь высока, что даже сейчас, при всех его недостатках (по структуре и замыслу Кобол во многом напоминает Фортран) появляются новые

его диалекты и реализации. Так, недавно появилась реализация Кобола, совместимая с Microsoft.NET, что потребовало, вероятно, внесения в язык некоторых черт объектно-ориентированного языка.

PL/1

В 1964 году все та же корпорация IBM создала язык PL/1, который был призван заменить Cobol и Fortran в большинстве приложений. Язык обладал необыкновенным богатством синтаксических конструкций. В нем впервые появилась обработка исключительных ситуаций и поддержка параллелизма. Надо заметить, что синтаксическая структура языка была крайне сложной.

В силу таких особенностей разработка компилятора для PL/1 была чрезвычайно сложным делом. Язык так и не стал популярен вне мира IBM.

BASIC

В 1963 году в Дартмурском колледже (Dartmouth College) был создан язык программирования BASIC (Beginners' All-Purpose Symbolic Instruction Code – многоцелевой язык символических инструкций для начинающих). Язык задумывался в первую очередь как средство обучения и как первый изучаемый язык программирования. Он предполагался легко интерпретируемым и компилируемым. Надо сказать, что BASIC действительно стал языком, на котором учатся программировать (по крайней мере, так было еще несколько лет назад; сейчас эта роль отходит к Pascal). Было создано несколько мощных реализаций BASIC, поддерживающих самые современные концепции программирования (ярчайший пример – Microsoft Visual Basic).

Algol

В 1960 году командой во главе с Петером Науром (Peter Naur) был создан язык программирования Algol. Этот язык дал начало целому семейству Алгол-подобных языков (важнейший представитель – Pascal). В 1968 году появилась новая версия языка. Она не нашла столь широкого практического применения, как первая версия, но была весьма популярна в кругах теоретиков. Язык

Петер Наур

был достаточно интересен, так как обладал многими уникальными на тот момент характеристиками.

7.3. Дальнейшее развитие языков программирования

Создание каждого из вышеупомянутых языков (за исключением, может быть, Algol'a) было вызвано некоторыми практическими требованиями. Эти языки послужили фундаментом для более поздних разработок. Все они представляют одну и ту же парадигму программирования. Следующие языки пошли существенно дальше в своем развитии, в сторону более глубокого абстрагирования.

Pascal-подобные языки

В 1970 году Никлаусом Виртом был создал язык программирования Pascal. Язык замечателен тем, что это первый широко распространенный язык для структурного программирования (первым, строго говоря, был Алгол, но он не получил столь широкого распространения). Впервые оператор безусловного перехода перестал играть основополагающую роль при управлении порядком выполнения операторов. В этом языке также внедрена строгая проверка типов, что позволило выявлять многие ошибки на этапе компиляции.

Никлаус Вирт

C-подобные языки

В 1972 году Керниганом и Ритчи был создан язык программирования C. Он создавался как язык для разработки операционной системы UNIX. Часто C называют «переносимым ассемблером», имея в виду то, что он позволяет работать с данными практически так же эффективно, как на ассемблере, предоставляя при этом структурированные управляющие конструкции и абстракции высокого уровня (структуры и массивы). Именно с этим связана его

Керниган и Ритчи

огромная популярность и поныне. Компилятор C очень слабо контролирует типы, поэтому очень легко написать внешне совершенно правильную, но логически ошибочную программу.

Бьярн Страуструп

В 1986 году Бьярн Страуструп создал первую версию языка C++, добавив в язык C объектно-ориентированные черты. C++ продолжает совершенствоваться и в настоящее время. Так, в 1998 году вышла новая (третья) версия стандарта, содержащая в себе некоторые довольно существенные изменения. Язык стал основой для разработки современных больших и сложных проектов.

Кен Арнольд

В 1995 году в корпорации Sun Microsystems Кеном Арнольдом и Джеймсом Гослингом был создан язык Java. Он наследовал синтаксис C и C++. В Java нет указателей и множественного наследования, что сильно повышает надежность программирования.

Билл Гейтс

В 1999–2000 годах в корпорации Microsoft был создан язык C#. Он в достаточной степени схож с Java (и задумывался как альтернатива последнему), но имеет и отличительные особенности. Ориентирован в основном на разработку многокомпонентных интернет-приложений.

Языки обработки данных

Все вышеперечисленные языки являются языками общего назначения в том смысле, что они не ориентированы и не оптимизированы под использование каких-либо специфических структур данных или на применение в каких-либо специфических областях. Было разработано

большое количество языков, ориентированных на достаточно специфические применения. Ниже приведен краткий обзор таких языков.

Скриптовые языки

В последнее время в связи с развитием интернет-технологий, широким распространением высокопроизводительных компьютеров и рядом других факторов получили распространение так называемые скриптовые языки. Эти языки первоначально ориентировались на использование в качестве внутренних управляющих языков во всякого рода сложных системах.

JavaScript

Язык был создан в компании Netscape Communications в качестве языка для описания сложного поведения веб-страниц. Первоначально назывался LiveScript, причиной смены названия послужили маркетинговые соображения. Интерпретируется браузером во время отображения веб-страницы. По синтаксису схож с Java и (отдаленно) с C/C++. Имеет возможность использовать встроенную в браузер объектную функциональность, однако подлинно объектно-ориентированным языком не является.

VBScript

Язык был создан в корпорации Microsoft во многом в качестве альтернативы JavaScript. Имеет схожую область применения. Синтаксически схож с языком Visual Basic (и является усеченной версией последнего). Так же, как и JavaScript, исполняется браузером при отображении веб-страниц и имеет ту же объектно-ориентированную степень.

Perl

Язык создавался в помощь системному администратору операционной системы Unix для обработки различного рода текстов и выделения нужной информации. Развился до мощного средства работы с текстами. Является интерпретируемым языком и реализован практически на всех существующих платформах. Применяется при обработке текстов, а также для динамической генерации веб-страниц на веб-серверах.

Python

Интерпретируемый объектно-ориентированный язык программирования. По структуре и области применения близок к Perl, однако ме-

нее распространен и более строг и логичен. Имеются реализации для большинства существующих платформ.

Заключение. Языки развиваются в сторону все большей и большей абстракции. И это сопровождается падением эффективности. Повышение уровня абстракции влечет за собой повышение уровня надежности программирования. С низкой эффективностью можно бороться путем создания более быстрых компьютеров. Если требования к памяти слишком высоки, можно увеличить ее объем. Это, конечно, требует времени и средств, но это решаемо. А вот с ошибками в программах можно бороться только одним способом: их надо исправлять. А еще лучше – не совершать или максимально затруднить их совершение. И именно на это направлены все исследования в области языков программирования.

7.4. Транслятор, компилятор, интерпретатор

Транслятор (англ. *translator* – переводчик) – это программа-переводчик. Она преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд. Трансляторы реализуются в виде ***компиляторов*** или ***интерпретаторов***. С точки зрения выполнения работы компилятор и интерпретатор существенно различаются.

Компилятор (англ. *compiler* – составитель, собиратель) – читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Интерпретатор (англ. *interpreter* – истолкователь, устный переводчик) – переводит и выполняет программу строка за строкой. После того как программа откомпилирована, ни сама исходная программа, ни компилятор более не используются. В то же время программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом очередном запуске программы. Откомпилированные программы работают быстрее, но интерпретируемые проще исправлять и изменять.

Каждый конкретный язык ориентирован либо на компиляцию, либо на интерпретацию – в зависимости от того, для каких целей он создавался. Например, Паскаль обычно используется для решения довольно сложных задач, в которых важна скорость работы программ. Поэтому данный язык обычно реализуется с помощью компилятора.

С другой стороны, Бейсик создавался как язык для начинающих программистов, для которых построчное выполнение программы имеет неоспоримые преимущества. Иногда для одного языка имеется и компилятор, и интерпретатор. В этом случае для разработки и тестирования программы можно воспользоваться интерпретатором, а затем откомпилировать отлаженную программу, чтобы повысить скорость ее выполнения.

7.5. Интегрированные среды программирования

Большую часть мирового парка персональных компьютеров (ПК) составляют компьютеры, у которых внутренний язык микропроцессора совместим с набором команд первых персональных компьютеров фирмы IBM (IBM PC), что позволяет выполнять на них одни и те же программы. Такие компьютеры называют IBM-совместимые. Их насчитывается сотни миллионов, и благодаря своей массовости они стали стандартом ПК. Массовость IBM-совместимых ПК обусловило то, что фирмы-разработчики программ стали ориентировать свою продукцию на эти компьютеры, и в результате программное обеспечение для использования на IBM-совместимых ПК также стало стандартом во всем мире.

Рассмотрим понятие «*система программирования*». Как известно, программа, написанная на каком-либо алгоритмическом языке, перед выполнением на компьютере должна быть транслирована в машинные коды, для чего используются программы-переводчики (трансляторы). В середине 80-х годов разработчики программного обеспечения перешли от создания чистых трансляторов к более удобным для пользователей системам программирования, включавшим в себя, помимо транслятора, удобные средства написания, редактирования и отладки программ. К наиболее известным системам программирования относятся Turbo Pascal, Turbo-C, Turbo-Basic, Quick Basic и др. Система программирования может рассматриваться как компилятор соответствующего языка, дополненный инструментальной оболочкой для быстрой разработки программ.

Интегрированная среда – система программных средств, используемая программистами для разработки программного обеспечения.

Обычно среда разработки включает в себя текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки и отладчик. Иногда также содержит средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Хотя и существуют среды разработки, предназначенные для нескольких языков – такие как Eclipse или Microsoft Visual Studio, обычно среда разработки предназначается для одного определённого языка программирования – как, например, Visual Basic.

Отла́дчик *(debugger)* является модулем среды разработки или отдельным приложением, предназначенным для поиска ошибок в программе. Отладчик позволяет выполнять пошаговую трассировку, отслеживать, устанавливать или изменять значения переменных в процессе выполнения программы, устанавливать и удалять контрольные точки или условия остановки и т. д.

Графический интерфейс – система средств, предназначенная для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.). При этом в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или указательного устройства ввода) ко всем видимым экранным объектам.

Вопросы для самоконтроля

1. Что понимают под языками программирования?
2. К какому классу языков относится язык Ассемблер?
3. Перечислите первые алгоритмические языки.
4. Кого считают основоположником алгоритмического языка Паскаль?
5. Какие языки позволили использовать структурное программирование?
6. Раскройте основное назначение скриптовых языков.
7. Что такое транслятор?
8. Чем отличается компилятор от интерпретатора?
9. Что понимают под интегрированной средой программирования?

III. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

1. Варианты для выполнения практических заданий по теме «Следования»

Таблица вариантов заданий

Вычислить и вывести на печать значение функции Y. Исходные данные *x*, *a* и *b* ввести с клавиатуры.	
1	$y=\left(\frac{1}{\sin x}+a\right)+a^b\times\sqrt{b}$
2	$y = (a + \mathrm{tg}x) \times (b + \ln a) \times e^{-b}$
3	$y = e^{\lvert a\rvert}\times(a+\frac{1}{\sin(x)})\times\sqrt{a+b}$
4	$y = \ln b\times(a+\frac{1}{\sin(x)})\times\frac{1}{e^{-b}}$
5	$y = (a + b) \times a^{b+1} \times \mathrm{tg}x$
6	$y = tgx\times\frac{(a+b)}{\sqrt{b}}$
7	$y=\frac{1}{\cos(x)}\times\ln(b+a)\times(a+b)$
8	$y=\frac{1}{\sin(x)}\times e^{-b}\times\ln(a+b)$
9	$y = \mathrm{tg}x \times a^{(b+1)} \times \ln(a + b)$
10	$y=(b+\frac{a}{\cos(x)})\times\sqrt{a+b}\times e^{-b}$
11	$y=(a+b)\times\frac{1}{\sin x}\times\ln b$
12	$y = (a + 1) \times (b + 2) \times \mathrm{tg}x \times e^{-b}$
13	$y = (a + \mathrm{tg}x) \times (b + \ln a) \times e^{-b}$
14	$y=(a+2b)\times\sqrt{b+2a}\times\frac{1}{\cos x}$
15	$y=(a+b)\times tgx\times\frac{1}{e^{-a}}$

16	$y = \frac{\sin 2\pi x}{(a+x)b}$
17	$y = (1 - e^{-ab}) \times \sin(4\pi x)$
18	$y = b \times e^{-2a} \times \sin^2 2\pi x$
19	$y = b(1 - e^{-0,5a}) \times \cos(2\pi x)$
20	$y = (1 - \sin 2\pi x) \times \cos(2\pi x) \times e^{-ab}$
21	$y = 4 \times e^{-0,5b} \times \cos^2(a\pi x)$
22	$y = \frac{\sin 2\pi x}{\sqrt{1+a^b}}$
23	$y = (a + \mathrm{tg}x) \times (b + \ln a) \times e^{-b}$
24	$y = e^{\lvert a \rvert} \times (a + \frac{1}{\sin(x)}) \times \sqrt{a+b}$
25	$y = \frac{1}{\sin(x)} \times e^{-b} \times \ln(a+b)$
26	$y = tgx \times \frac{(a+b)}{\sqrt{b}}$
27	$y = (a + 1) \times (b + 2) \times \mathrm{tg}x \times e^b$
28	$y = \mathrm{tg}x \times a^{(b+1)} \times \ln(a + b)$
29	$y = \frac{1}{\cos(x)} \times \ln(b+a) \times (a+b)$
30	$y = (a + b) \times a^{b+1} \times \mathrm{tg}x$

2. Варианты для выполнения практических заданий по теме «Ветвления»

Таблица вариантов заданий

Вариант	Задание
1	Переменной k присвоить номер четверти плоскости, в которой находится точка с координатами *x* и *y* (*xy* = 0)
2	Если сумма трех попарно различных действительных чисел *x*, *y*, z меньше единицы, то заменить меньшее из *x* и y полусуммой двух других, в противном случае уменьшить все числа в 5 раз
3	Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3)
4	Написать программу, которая выбирает наименьшее из четырех заданных чисел

Вариант	Задание
5	Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны
6	Даны действительные числа *x*, *y* и z. Вычислить max (*x* + *y* + z, *xy*z)
7	Поменять местами значения переменных *a*, *b*, *c*, не равных между собой, таким образом, чтобы *a* > *b* > *c*
8	Меньшее из двух значений переменных вещественного типа заменить нулем, а в случае их равенства – заменить нулями оба
9	Если сумма трех попарно различных действительных чисел *x*, *y*, z меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух других, в противном случае возвести все числа в квадрат
10	Наибольшее из трех различных значений переменных целого типа *x*, *y* и z уменьшить на 3
11	Даны два действительных числа, не равных между собой. Меньшее из них заменить их полусуммой
12	Если значение ω не равно 0 и при этом котангенс от ω меньше 0,5, тогда поменять знак *y* ω, а если значение равно 0, тогда присвоить ω значение 1
13	Даны действительные числа *x*, *y* и z. Получить min (*x*, y, z).
14	Две точки плоскости заданы своими координатами. Определить, лежат ли они в одной или разных координатных плоскостях
15	Даны действительные числа *x*, *y* и z. Вычислить min ((*x* + *y* + z / 2), (*x* * *y* * z)) + 1
16	Даны действительные числа *a*, *b*, *c*. Удвоить эти числа, если *a* > *b* > *c*, и заменить их абсолютными значениями, если это не так
17	Известно, что из четырех чисел одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной *n*
18	Написать программу, которая выбирает наибольшее из четырех заданных чисел
19	Даны действительные числа *x* и *y*. Если *x* и *y* отрицательны, то каждое значение заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0,5, в противном случае извлечь из каждого квадратный корень
20	Две точки плоскости заданы своими координатами. Определить, лежат ли они в одной (распечатать ее номер) или разных координатных четвертях
21	Даны два действительных числа. Заменить первое число нулем, если оно меньше или равно второму, и удвоить оба числа в противном случае
22	Даны действительные числа *a*, *b*, *c*, *d*. Если *a* < *b* < *c* < *d*, то каждое число заменить небольшим из них; если a > b > c > d, то числа оставить без изменения; в противном случае все числа заменить их квадратами
23	Даны действительные числа *x*, *y* и z. Получить max (*x*, *y*, z)

Вариант	Задание
24	Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3)
25	Меньшее из двух значений переменных вещественного типа заменить нулем, а в случае их равенства – заменить нулями оба
26	Даны действительные числа *x*, *y* и z. Обнулить отрицательные числа
27	Известно, что из трех целых чисел одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной *n*
28	Наименьшее из трех различных значений переменных целого типа *x*, *y* и z увеличить на 3
29	Если сумма трех попарно различных действительных чисел *x*, *y*, z больше единицы, то наибольшее из этих трех чисел заменить полусуммой двух других, в противном случае возвести все числа в квадрат
30	Даны два действительных числа, не равных между собой. Наибольшее из них заменить их полусуммой

3. Варианты для выполнения практических заданий по теме «Циклы»

Таблица вариантов заданий

Выбрать функцию по варианту из таблицы. Составить программу расчета таблицы значений функции $f(x)$ на интервале $a <= x <= b$ в n равностоящих точках. Границы интервала a, b и количество точек n ввести с клавиатуры. Результаты вывести на печать	
Вариант	Функция
1	$f(x) = 5 \times (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти сумму всех положительных значений функции $f(x)$ в расчетных точках
2	$f(x) = 4 \times e^{-0,5x} \times \cos(\pi x)$ Найти сумму всех отрицательных значений функции $f(x)$ в заданном интервале
3	$f(x) = 5 \times e^{-0,5x} \times \sin(\pi x)$ Найти произведение вычисленных значений функции f(x), целая часть которых кратна 2
4	$f(x) = (1 - e^{-x}) \times \sin(4\pi x)$ Найти среднее арифметическое положительных значений функции $f(x)$
5	$f(x) = 1 + \sin(2\pi x)$ Найти количество положительных значений функции $f(x)$ на заданном интервале
6	$f(x) = 4 \times e^{-0,5x} \times \cos(\pi x)$ Найти произведение всех отрицательных значений функции $f(x)$ на заданном интервале

7	$f(x) = 5 \times (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти произведение всех положительных значений функции $f(x)$ на заданном интервале
8	$f(x) = (1 - e^{-x}) \times \sin(4\pi x)$ Найти сумму вычисленных значений функции $f(x)$, целая часть которых кратна 2
9	$f(x) = 1 + \frac{\sin 2\pi x}{1 + x}$ Найти среднее арифметическое отрицательных значений функции $f(x)$
10	$f(x) = \frac{\cos(4\pi x)}{1 + x^2}$ Найти сумму вычисленных значений функции $f(x)$, дробная часть которых > 0,5
11	$f(x) = 1 + \sin(2\pi x)$ Найти количество положительных значений функции $f(x)$
12	$f(x) = \frac{\cos(4\pi x)}{1 + x^2}$ Найти сумму вычисленных значений функции $f(x)$, дробная часть которых < 0,5
13	$f(x) = 5 \times (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти произведение вычисленных значений функции $f(x)$, абсолютные значения которых > 1
14	$f(x) = 5 \times (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти сумму вычисленных значений функции $f(x)$
15	$f(x) = \frac{\sin 4\pi x}{1 + x^2}$ Найти произведение вычисленных значений функции $f(x)$
16	$f(x) = \frac{\sin 2\pi x}{1 + x}$ Найти произведение вычисленных значений функции $f(x)$, дробная часть которых < 0,5
17	$f(x) = \frac{\cos 4\pi x}{1 + x^2}$ Найти сумму вычисленных значений функции $f(x)$, дробная часть которых > 0,5
18	$f(x) = e^{-0,5x} \times \cos^2(\pi x)$ Найти количество положительных значений функции $f(x)$ на заданном интервале
19	$f(x) = e^{-0,5x} \times \cos^2(\pi x)$ Найти количество значений функции $f(x)$, абсолютная величина целой части которых > 1

20	$f(x) = e^{-2x} \times \sin(2\pi x)$ Найти наибольшее значение функции $f(x)$ на заданном интервале
21	$f(x) = x \times \sin(2\pi x)$ Найти среднее значение функции $f(x)$
22	$f(x) = \sin(\pi x) \times \cos(\pi x)$ Найти сумму значений функции $f(x)$, абсолютная величина целой части которых < 1
23	$f(x) = (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти произведение вычисленных значений функции $f(x)$, абсолютные значения которых > 1
24	$f(x) = 5 \times e^{-0,5x} \times \sin(\pi x)$ Найти произведение вычисленных значений функции $f(x)$, целая часть которых кратна 2
25	$f(x) = (1 - e^{-x}) \times \sin(4\pi x)$ Найти наименьшее значение функции $f(x)$ на заданном интервале
26	$f(x) = \frac{\sin 2\pi x}{1 + x}$ Найти произведение значений функции в расчетных точках с четными номерами
27	$f(x) = 5 \times (1 - e^{-0,5x}) \times \cos(2\pi x)$ Найти сумму значений функции в расчетных точках с нечетными номерами
28	$f(x) = 1 + \sin(2\pi x)$ Найти наименьшее значение функции $f(x)$ на заданном интервале в расчетных точках с четными номерами
29	$f(x) = (1 - e^{-x}) \times \sin(4\pi x)$ Подсчитать количество отрицательных значений функции на заданном интервале
30	$f(x) = 4 \times e^{-0,5x} \times \cos(\pi x)$ Вычислить произведение положительных значений функции в расчетных точках с нечетными номерами

4. Варианты для выполнения практических заданий по теме «Одномерные массивы»

Таблица вариантов заданий

Вариант	Задание
1	Даны натуральное n, целые числа a1, ... , an, каждое из которых отлично от нуля. Если в массиве отрицательные и положительные элементы чередуются (+, - ,+ , - ... или -, +, -, +, ...), то ответом должен служить исходный массив. Иначе получить все отрицательные элементы массива, сохранив порядок их следования.
2	Даны натуральное n и действительные числа a1, ... , an (n – четное). Получить max (a1 + an, a2 + an-1, ... , an/2 + a(n/2) + 1).

Вариант	Задание
3	Даны натуральное число n, действительные числа a1, ... , an. Найти максимальный среди отрицательных элементов, имеющих четные индексы.
4	Даны натуральное n, действительные числа a1, ... , an. Получить числа в1, в2, ... , вn, где вi – среднее арифметическое всех элементов массива a1, ... , an, кроме ai (i = 1, 2, ... , n).
5	Даны натуральное n, действительные числа a1, ... , an. Если в результате замены отрицательных элементов массива a1, ... , an их квадратами элементы будут образовывать неубывающую последовательность, то получить сумму элементов исходного массива; в противном случае получить их произведение
6	Даны натуральное n, m, целые числа a1, ... , an, b1, ... , bm. Найти сумму тех элементов массива a1, ... , an, индексы которых совпадают со значением элементов массива b1, ... , bm
7	Даны натуральное n, действительные числа a1, ... , an (все числа попарно различны). Поменять в этом массиве местами наибольший и наименьший элементы
8	Даны натуральное n, действительные числа a1, ... , an. Получить преобразованный массив, заменяя ai нулями, если | ai | не равно max (a1, ... , an), и заменяя ai единицей в противном случае (i = 1, 2, ... , n)
9	Даны натуральное n, действительные числа a1, ... , an (все числа попарно различны). Поменять в этом массиве местами наименьший и последний элементы
10	Даны натуральное n, действительные числа a1, ... , an, b1, ... , bn. Получить новый массив c1, ... , cn, каждый элемент которого ci = max (ai, bi), i = 1, 2, ... , n
11	Даны натуральное n, действительные числа a1, ... , an, b1, ... , bn. Преобразовать b1, ... , bn по правилу: если ai < 0, то bi увеличить в 10 раз. Иначе bi заменить нулем, i = 1, 2, ... , n
12	Даны натуральное n, действительные числа a1, ... , an. Требуется умножить элементы массива a1, ... , an на квадрат ее наименьшего элемента, если ai ≥ 0. И на квадрат ее наибольшего члена, если ai < 0
13	Даны натуральное n, действительные числа a1, ... , an. Вычислить сумму тех элементов массива, индексы которых являются степенями двойки (1, 2, 4, 8, 16, ...)
14	Даны натуральные n, m, действительные числа a1, ... , an, b1, ... , bm. Элементы каждого из массивов упорядочены по не убыванию. Объединить элементы этих двух массивов в один массив c1, ... , cn + m так, чтобы они снова оказались упорядочены по не убыванию
15	Даны натуральные n, m, действительные числа a1, ... , an и b1, ... , bm. Найти наименьший среди элементов a1, ... , an, который не входит в b1, ... , bm

Вариант	Задание
16	Даны натуральные n, целые числа a1, ... , an. Найти сумму неповторяющихся элементов массива
17	Даны натуральные n, целые числа a1, ... , an. Наименьший элемент массива a1, ... , an заменить целой частью среднего арифметического всех элементов, остальные элементы оставить без изменения. Если в массиве несколько элементов со значением min (a1, ... , an), то заменить последний по порядку
18	Даны натуральные n, целые числа a1, ... , an. Преобразовать массив по правилу: все отрицательные элементы перенести в его начало, а все остальные в конец, сохраняя исходное взаимное расположение как среди отрицательных, так и среди остальных элементов
19	Даны натуральные n, действительные числа a1, ... , an (n – четное). Получить min (a1 * an, a2 * an-1, ... , a[n/2] * a[n/2]+1)
20	Даны натуральные n, действительные числа a1, ... , an. Преобразовать массив, расположив элементы в обратном порядке
21	Даны натуральное n, действительные числа a1, ... , an. Все элементы массива с четными номерами, предшествующие первому по порядку элементу со значением max (a1, ... , an), умножить на max (a1, ... , an)
22	Даны натуральное n, целые числа a1, ... , an. Найти сумму квадратов тех элементов массива, которые по модулю больше максимального элемента
23	Даны натуральное n, действительные числа a1, ... , an. Если в результате замены отрицательных элементов массива a1, ... , an их квадратами элементы будут образовывать неубывающую последовательность, то получить сумму элементов исходного массива; в противном случае получить их произведение
24	Даны натуральное n, действительные числа a1, ... , an. Найти максимальный элемент массива среди отрицательных элементов, имеющих четные индексы
25	Даны натуральное n, действительные числа a1, ... , an. Получить преобразованный массив, заменяя ai нулями, если | ai | не равно max (a1, ... , an), и заменяя ai единицей в противном случае
26	Даны натуральное n, действительные числа a1, ... , an (все числа попарно различны). Поменять в этом массиве местами наименьший и наибольший элементы
27	Даны натуральное n, действительные числа a1, ... , an. Вычислить сумму тех элементов массива, индексы которых являются степенями двойки (1, 3, 9, 27, ...)
28	Даны натуральное n, действительные числа a1, ... , an. Если в результате замены отрицательных элементов массива a1, ... , an их квадратами элементы будут образовывать невозрастающую последовательность, то получить произведение элементов исходного массива; в противном случае получить их сумму

Вариант	Задание
29	Даны натуральное число n, действительные числа a1, ... , an. Найти максимальный элемент среди отрицательных элементов, имеющих нечетные индексы
30	Даны натуральное n, действительные числа a1, ... , an. Найти произведение индексов отрицательных элементов массива

5. Варианты для выполнения практических заданий по теме «Двумерные массивы»

Таблица вариантов заданий

Вариант	Задание
1	В данной действительной матрице размером n∗m поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны
2	Дана действительная матрица размером n∗m, все элементы которой различны. В каждой строке выбирается элемент с наибольшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением
3	Дана целочисленная матрица размером n∗m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
4	Дана действительная матрица размером n∗m. Найти среднее арифметическое каждого из столбцов, имеющих четные номера
5	Дана действительная матрица размером n∗m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько)
6	Дана целочисленная матрица размером n∗m. Написать программу, позволяющую находить сумму наибольших значений элементов ее строк
7	Дана целочисленная квадратная матрица размером n∗m. Написать программу, формирующую два одномерных массива. В один переслать по строкам верхний треугольник матрицы, включая элементы главной диагонали, в другой – нижний треугольник. Полученные массивы распечатать
8	Дана целочисленная квадратная матрица размером n∗m. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали
9	Дана целочисленная квадратная матрица размером n∗m. Написать программу, позволяющую поменять местами элементы, расположенные в верхней и нижней четвертях, ограниченные главной и побочной диагоналями (за исключением элементов, расположенных на диагоналях)

Вариант	Задание
10	Задана действительная матрица размером n*m. Написать программу, позволяющую заменить все элементы, наименьшие в строке, на нули
11	Задана целочисленная матрица размером n*m. Написать программу, позволяющую находить строки с наименьшей и наибольшей суммой и выводить их на печать
12	Задана целочисленная квадратная матрица размером n*n. Написать программу, преобразующую исходную матрицу по правилу: нечетные столбцы разделить на среднее значение диагональных элементов матрицы, а четные оставить без изменения.
13	Задана действительная квадратная матрица размером n*n. Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше ее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если таких элементов нет, то ответом должно служить сообщение об этом
14	Задана целочисленная квадратная матрица размером n*n (n – четное). Написать программу, позволяющую менять местами элементы первой и второй строк, элементы третьей и четвертой строк и т. д.
15	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу прибавлением к элементам каждого столбца первой матрицы произведения элементов соответствующих строк второй матрицы
16	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу умножением элементов каждой строки первой матрицы на наибольшее из значений элементов соответствующей строки второй матрицы
17	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых – нули
18	Задан массив из целых чисел размером n и число L. Написать программу, формирующую из него матрицу, содержащую по L элементов в строке. Недостающие элементы заполнить нулями
19	Дана целочисленная матрица размером n*m (m – четное). Написать программу, позволяющую менять местами элементы первого и последнего столбцов, элементы второго и (n-1)-го столбцов и т. д. до среднего столбца (n – нечетное)
20	Дана действительная квадратная матрица размером n*n (n – четное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении с этими диагоналями
21	Дана целочисленная матрица размером n*m. Найти максимальный по модулю элемент среди отрицательных элементов нечетных столбцов

Вариант	Задание
22	Дана целочисленная матрица размером n*m и число K. Написать программу, переставляющую строки и столбцы таким образом, чтобы максимальный по модулю элемент был расположен на пересечении K-й строки и K-го столбца
23	Дана действительная матрица размером n*m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько)
24	Дана целочисленная матрица размером n*m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
25	Дана целочисленная квадратная матрица размером n*n. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали
26	В данной действительной матрице размером n*m обнулить все отрицательные элементы. Подсчитать количество обнуленных элементов
27	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых отрицательны.
28	Задана целочисленная квадратная матрица размером n*n (n – четное). Написать программу, позволяющую менять местами элементы первой и последней строк, второй и предпоследней строк и т. д.
29	Задана целочисленная матрица размером n*m. Написать программу, позволяющую находить строки с наименьшим и наибольшим произведением элементов. Вывести на печать номера этих строк
30	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу умножением элементов каждой строки первой матрицы на наименьшее из значений элементов соответствующей строки второй матрицы

6. Варианты для выполнения практических заданий по теме «Подпрограммы»

Таблица вариантов заданий

Составить программу с использованием процедур и функций	
Вариант	Задание
1	В данной действительной матрице размером n*m поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны
2	Дана действительная матрица размером n*m, все элементы которой различны. В каждой строке выбирается элемент с наибольшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением

Составить программу с использованием процедур и функций	
Вариант	Задание
3	Дана целочисленная матрица размером n*m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
4	Дана действительная матрица размером n*m. Найти среднее арифметическое каждого из столбцов, имеющих четные номера
5	Дана действительная матрица размером n*m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько)
6	Дана целочисленная матрица размером n*m. Написать программу, позволяющую находить сумму наибольших значений элементов ее строк
7	Дана целочисленная квадратная матрица размером n*m. Написать программу, формирующую два одномерных массива. В один переслать по строкам верхний треугольник матрицы, включая элементы главной диагонали, в другой – нижний треугольник. Полученные массивы распечатать
8	Дана целочисленная квадратная матрица размером n*m. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали
9	Дана целочисленная квадратная матрица размером n*m. Написать программу, позволяющую поменять местами элементы, расположенные в верхней и нижней четвертях, ограниченные главной и побочной диагоналями (за исключением элементов, расположенных на диагоналях)
10	Задана действительная матрица размером n*m. Написать программу, позволяющую заменить все элементы, наименьшие в строке, на нули
11	Задана целочисленная матрица размером n*m. Написать программу, позволяющую находить строки с наименьшей и наибольшей суммой и выводить их на печать
12	Задана целочисленная квадратная матрица размером n*n. Написать программу, преобразующую исходную матрицу по правилу: нечетные столбцы разделить на среднее значение диагональных элементов матрицы, а четные оставить без изменения
13	Задана действительная квадратная матрица размером n*n. Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше ее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если таких элементов нет, то ответом должно служить сообщение об этом
14	Задана целочисленная квадратная матрица размером n*n (n – четное). Написать программу, позволяющую менять местами элементы первой и второй строк, элементы третьей и четвертой строк и т. д.

Составить программу с использованием процедур и функций	
Вариант	Задание
15	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу прибавлением к элементам каждого столбца первой матрицы произведения элементов соответствующих строк второй матрицы
16	Даны две действительные квадратные матрицы размером n*n. Получить новую матрицу умножением элементов каждой строки первой матрицы на наибольшее из значений элементов соответствующей строки второй матрицы
17	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых – нули
18	Задан массив из целых чисел размером n и число L. Написать программу, формирующую из него матрицу, содержащую по L элементов в строке. Недостающие элементы заполнить нулями
19	Дана целочисленная матрица размером n*m (m – четное). Написать программу, позволяющую менять местами элементы первого и последнего столбцов, элементы второго и (n-1)-го столбцов и т. д. до среднего столбца (n – нечетное)
20	Дана действительная квадратная матрица размером n*n (n – четное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении с этими диагоналями
21	Дана целочисленная матрица размером n*m. Найти максимальный по модулю элемент среди отрицательных элементов нечетных столбцов
22	Дана целочисленная матрица размером n*m и число K. Написать программу, переставляющую строки и столбцы таким образом, чтобы максимальный по модулю элемент был расположен на пересечении K-й строки и K-го столбца
23	Дана действительная матрица размером n*m. Все элементы с наибольшим значением заменить нулями (таких элементов может быть несколько)
24	Дана целочисленная матрица размером n*m. Написать программу, формирующую двумерный массив по следующему правилу: элементы первой строки – в порядке возрастания индексов столбцов, элементы второй строки – в порядке убывания индексов столбцов и т. д.
25	Дана целочисленная квадратная матрица размером n*n. Написать программу, позволяющую исключать из нее столбец, в котором расположен минимальный элемент главной диагонали
26	В данной действительной матрице размером n*m обнулить все отрицательные элементы. Подсчитать количество обнуленных элементов
27	Дана целочисленная квадратная матрица размером n*n. Найти номера строк, все элементы которых отрицательны

Составить программу с использованием процедур и функций	
Вариант	Задание
28	Задана целочисленная квадратная матрица размером n∗n (n – четное). Написать программу, позволяющую менять местами элементы первой и последней строк, второй и предпоследней строк и т. д.
29	Задана целочисленная матрица размером n∗m. Написать программу, позволяющую находить строки с наименьшим и наибольшим произведением элементов. Вывести на печать номера этих строк
30	Даны две действительные квадратные матрицы размером n∗n. Получить новую матрицу умножением элементов каждой строки первой матрицы на наименьшее из значений элементов соответствующей строки второй матрицы

Форма контроля

В ходе подготовки к экзамену необходимо освоить теоретический материал, представленный в учебно-методическом пособии «Информатика», ответить на все вопросы самоконтроля в конце каждого раздела.

Выполнить все упражнения и задания по вариантам из всех разделов.

Вопросы к экзамену

1. Алгоритм и его свойства.
2. Базовые алгоритмические структуры.
3. Основные блоки для построения алгоритмов.
4. Правила построения блок-схем.
5. Этапы решения задач на ЭВМ.
6. Алгоритмический язык Паскаль. Алфавит языка. Типы данных.
7. Стандартные функции.
8. Арифметические выражения и правила их записи.
9. Структура программы на языке Паскаль. Описательная и исполнительная части программы.
10. Типы данных.
11. Понятие оператора. Пустой, простой и составной операторы.
12. Операторы присваивания, ввода-вывода. Комментарии в программе.
13. Логический оператор. Оператор выбора.
14. Операторы цикла с параметром, с предусловием, с постусловием.
15. Оператор прерывания цикла.
16. Вложенные циклы.
17. Итерационные циклические структуры.
18. Операции с индексированными переменными.
19. Ввод-вывод массивов.
20. Одномерные массивы. Двумерные массивы.
21. Организация подпрограмм. Процедуры. Функции.
22. Локальные и глобальные переменные.
23. Формальные и фактические параметры.
24. Вложенные процедуры.

Библиографический список

Основная литература

1. Андреева, Т.А. Программирование на языке Pascal : учеб. пособие / Т.А. Андреева. – М. : Интернет-Ун-т информ. технологий : БИНОМ ; Лаб. знаний, 2006. – 234 с.
2. Баженова, И.Ю. Введение в программирование : учеб. пособие / И.Ю. Баженова, В.А. Сухомлин. – М. : Интернет-Ун-т информ. технологий ; БИНОМ . Лаб. знаний, 2007. – 326 с.
3. Карпов, Ю.Г. Теория и технология программирования. Основы построения трансляторов : учеб. пособие для вузов / Ю.Г. Карпов. – СПб. : БХВ-Петербург, 2008. – 270 с.
4. Панкратова, Л.П. Контроль знаний по информатике : тесты, контрольные задания, экзаменационные вопросы, компьютерные проекты / Л.П. Панкратова, Е.Н. Чулак. – СПб. : БХВ-Петербург, 2004. – 440 с.

Дополнительная литература

5. Сырецкий, Г.А. Информатика: фундаментальный курс : учеб. для вузов. Т. 1. Основы информационной и вычислительной техники / Г.А. Сырецкий. – СПб. : БХВ-Петербург, 2005. – 822 с.
6. Шапорев, С.Д. Информатика: теоретический курс и практические занятия : учеб. для вузов / С.Д. Шапорев. – СПб. : БХВ-Петербург, 2008. – 469 с.

Интернет-ресурсы

1. Обучение в Интернете. Бесплатное дистанционное обучение информатике, телекоммуникациям, основам электронного бизнеса [Электронный ресурс]. – URL : http//stream-time.ru
2. Суркова, Е.В. Лабораторный практикум по программированию на языке PASCAL [Электронный ресурс] / Е.В. Суркова. – Ульяновск, 2007. – URL : http://www.ph4s.ru/bookprogramir_4

Глоссарий

Алгоритм – подробное описание последовательности действий, позволяющих решить конкретную задачу. Элементарные действия, на которые разбивается алгоритм, называются инструкциями или командами.

Алфавит языка – программа на Паскале записывается в виде последовательности символов, образующих алфавит языка.

Блок-схема – при графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий. Такое графическое представление называется схемой алгоритма или блок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т. п.) соответствует геометрическая фигура.

Ветвления – тип вычислительного процесса, когда в зависимости от справедливости проверяемого условия (да или нет) алгоритм может пойти по одной из двух возможных ветвей. Происходит выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к общему выходу, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

Вложенные циклы – возможны случаи, когда внутри тела цикла необходимо повторять некоторую последовательность операторов, т. е. организовать внутренний цикл. Такая структура получила название цикла в цикле или ***вложенных циклов***.

Данные – это информация, представленная в формализованном виде и предназначенная для обработки ее техническими средствами, например ЭВМ.

Интерпретатор (англ. *interpreter* – истолкователь, устный переводчик) **–** переводит и выполняет программу строка за строкой. После того как программа откомпилирована, ни сама исходная программа, ни компилятор более не используются. В то же время программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом очередном запуске программы. Откомпилированные программы работают быстрее, но интерпретируемые проще исправлять и изменять.

Информация – это сведения, снимающие неопределенность об окружающем мире, которые являются объектом хранения, преобразования, передачи и использования.

Исполнительная часть программы – часть программы, содержащая инструкции (***операторы***) компьютеру на выполнение. Именно с исполнительной части начинается выполнение программы.

Итерационные циклы – особенностью итерационного цикла является то, что число повторений тела цикла заранее неизвестно. Для его организации используются циклы с постусловием. Выход из итерационного цикла осуществляется по выполнению некоторого условия.

Компилятор (англ. *compiler* – составитель, собиратель) – читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется. Локальные переменные – промежуточные переменные, которые используются только в процедуре, описываются в процедуре и называются локальными. Локальные переменные не доступны в основной программе. Время существования локальных переменных – от начала и до конца работы процедуры, в которой они описаны.

Массив – это поименованный набор однотипной информации. Массив объединяет элементы одного типа данных. Всему набору данных присваивают общее имя – имя массива. Каждый элемент массива устанавливается с помощью индекса, определяющего место этого элемента в общем наборе. Данные в массиве сохраняются, как и в случае использования обычных неиндексированных переменных, только до конца работы программы.

Описательная часть программы – часть программы, которая используется для описания переменных, констант, пользовательских типов, меток.

Паскаль – широко распространенный язык для структурного программирования. Впервые оператор безусловного перехода перестал играть основополагающую роль при управлении порядком выполнения операторов. В этом языке также внедрена строгая проверка типов, что позволило выявлять многие ошибки на этапе компиляции.

Программа – это запись алгоритма на языке программирования, приводящая к конечному результату за конечное число шагов. Программа – это детальное и законченное описание алгоритма средствами языка программирования.

Процедура – это независимая именованная часть программы, которую после однократного описания можно многократно вызывать по имени.

Следование – тип вычислительного процесса, когда действия выполняются строго в том порядке, в котором записаны. Образуется последовательностью действий, следующих одно за другим.

Структура данных – это программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных. Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих интерфейс структуры данных. В вычислительной технике данные обычно отличают от программ. Программа является набором инструкций, которые детализируют вычисление или задачу, которая производится компьютером. Данные – это всё отличное от программного кода.

Тип переменной – определяет объем оперативной памяти, выделяемой под хранение переменной.

Транслятор (англ. *translator* – переводчик) – это программа-переводчик. Она преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд. Трансляторы реализуются в виде ***компиляторов*** или ***интерпретаторов***. С точки зрения выполнения работы компилятор и интерпретатор существенно различаются.

Цикл – тип вычислительного процесса, когда действия повторяются многократно по одним и тем же математическим зависимостям. Обеспечивает многократное выполнение некоторой совокупности действий, которая называется телом цикла.

Цикл с параметром – выполняется тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне.

Цикл с постусловием – тело цикла выполняется до тех пор, пока *не выполнится* условие.

Цикл с предусловием – тело цикла выполняется до тех пор, пока *выполняется* условие.

Язык программирования – правила представления данных и записи алгоритмов их обработки, которые автоматически выполняются ЭВМ. В более абстрактном виде язык программирования является средством создания программных моделей объектов и явлений внешнего мира.

СОДЕРЖАНИЕ

Учебное издание

Ахмедханлы Дилара Микаил кызы

ПРОГРАММИРОВАНИЕ НА TURBO PASCAL

Учебно-методическое пособие

Технический редактор *З.М. Малявина*
Корректор *Г.В. Данилова*
Вёрстка: *Л.В. Сызганцева*
Дизайн обложки: *Г.В. Карасева*

Подписано в печать 03.05.2012. Формат 60×84/16.
Печать оперативная. Усл. п. л. 5,12.
Тираж 100 экз. Заказ № 1-40-11.

Издательство Тольяттинского государственного университета
445667, г. Тольятти, ул. Белорусская, 14