Министерство образования и науки Украины

Донбасская государственная машиностроительная академия

Кафедра прикладной математики

Конспект лекций

по дисциплине

«Информатика»

для студентов всех специальностей
заочной формы обучения

Часть 1

Утверждено на заседании
методического совета ДГМА
Протокол № 1 от 20.10. 2005

Рекомендовано для дальнейшего
использования в учебном процессе
методическим советом ФАМИТ
Протокол № 6 от 20.02. 2012

Краматорск 2005

УДК 004.43

ББК 22.18

Конспект лекций по дисциплине “Информатика” для студентов всех специальностей заочной формы обучения. Часть 1/ Сост. И.А. Гетьман - Краматорск: ДГМА, 2005. - 40 с.

Рассмотрены основные понятия, позволяющие освоить приемы и методы алгоритмизации задач, получить навыки составления и отладки программ для ЭВМ на языке PASCAL для студентов заочного отделения. Рассмотрены примеры выполнения контрольной работы.

Составители Гетьман И.А., ассист.

Отв. за выпуск Черномаз В.Н., доцент.

Содержание

Введение

Основная цель данных методических указаний - приобретение студентами практических навыков при взаимодействии с ПЭВМ и решении задач с использованием языка программирования **Pascal**.

Во время подготовки к выполнению очередного задания студенту необходимо:

проработать теоретический материал по теме задачи;

разработать блок-схему алгоритма и составить программу по своему варианту задания.

Студент должен решить поставленную задачу на ЭВМ, оформить отчет по задаче и защитить его.

Внимание!

Правила техники безопасности запрещают студентам включать или выключать какие-либо устройства, вскрывать оборудование, снимать защитные щиты, а также прикасаться к токоведущим частям устройств.

После выполнения работы студент обязан привести рабочее место в порядок.

Порядок выполнения элементарных действий при выполнении любой работы на ЭВМ.

В систему программирования **Turbo Pascal** фирмы Borland встроен простой, но достаточно удобный текстовый редактор для создания текстов программ. Не выходя из него, можно компилировать программы, находить ошибки и тут же их исправлять, компоновать программы из отдельных частей, запускать отлаженную программу. Для этого предназначено системное меню, активизируемое нажатием клавиши ***F10.*** Некоторые команды дублируются функциональными клавишами.

Ниже описаны действия при выполнении основных функций при разработке программ.

а) Загрузка системы Turbo Pascal

Войти в личный каталог студента, содержащий индивидуальные тексты его программ;

активизировать файл **turbo.exe** (обычно это делается через пользовательское меню с использованием клавиши ***F2***), при этом в верхней части экрана появится меню команд системы.

б) Создание (ввод) новой программы

Нажать клавишу ***F10*** (вход в меню);

установить курсор на команду **FILE** и активизировать ее путем нажатия на клавишу ***Enter***;

в появившейся на экране рамке с перечнем команд установить курсор на команду **NEW** и активизировать ее (откроется новое окно);

ввести программу;

по окончании ввода программы нажать клавишу ***F10*** и активизировать снова команду меню ***File***;

в появившемся перечне команд активизировать команду **Save As**;

указать имя файла и нажать клавишу ***Enter*** (запись файла на диск);

для компиляции и выполнения программы можно либо с помощью клавиши ***F10*** войти в меню системы **Turbo Pascal** и активизировать команду ***Run*** , либо нажать клавиши ***Ctrl+F9***;

для наблюдения за результатами выполнения программы нажать клавиши ***Alt+F5.***

в) Корректировка (исправление) программы

Если в программе обнаружены ошибки, то для их устранения следует выполнить следующие операции:

устранить ошибки в программе;

новый вариант программы записать на диск, для чего нажать клавишу ***F10,*** активизировать команду ***File,*** в подменю выбрать команду ***Save*** или нажать клавишу ***F2***;

для компиляции, выполнения и наблюдения за результатами выполнить ранее описанные команды.

г) Сохранение программы

Если необходимо записать программный файл, то надо в опции ***File*** активизировать команду ***Save As***.

Для сохранения текущего программного файла в опции ***File*** активизируется команда ***Save*** (***F2***).

д) Загрузка в редактор ранее сохраненного текста

Нажать клавишу ***F10*** и активизировать команду ***File***;

в подменю выбрать команду ***Open*** (***F3***) и активизировать ее;

на запрос системы указать имя загружаемого файла или выбрать из предложенного списка.

Если после выхода из системы **Turbo Pascal** не было записи нового файла, то при повторном входе в нее предыдущий файл может быть сохранен в редакторе, тогда его загрузка не требуется.

е) Распечатка программы и результатов ее работы

После отладки программы, т. е. получения результатов вычислений на экране дисплея, необходимо вывести на принтер ее текст и результаты. Вывод текста программы на принтер можно осуществить следующим образом:

установить курсор на верхнюю строку выводимого текста (слева от текста) и нажать клавиши ***Ctrl+K-B*** (нажимаются одновременно клавиши ***Ctrl+K***, а затем – клавиша ***B***);

установить курсор ниже последней строки выводимого текста на одну строку (слева от текста) и нажать клавиши ***Ctrl+K-K***;

нажать клавиши ***Ctrl+K-P***. Текст будет распечатан.

Для вывода результатов решения задачи на принтер необходимо в программу внести следующие изменения:

после заголовка программы ввести оператор **Uses printer**;

в операторах вывода данных перед списком вывода записать **lst;**

запустить программу на выполнение (***Ctrl+F9)***. Результаты будут распечатаны.

ж) Работа с фрагментами текста

Система **Turbo Pascal** версии 7.0 допускает многооконную работу, т.е. одновременно может быть загружено несколько программ, каждая - в отдельное окно. Переключение между окнами осуществляется с помощью клавиши ***F6***. Удаление текущего окна – ***Alt+F3*** . Для доступа к другим функциям работы с окнами следует с помощью клавиши ***F10*** войти в меню системы и активизировать команду **Windows.**

Редактор **Turbo Pascal** допускает работу с фрагментами текста. Любой фрагмент можно, используя буфер обмена, скопировать в любое место программы в любом окне:

- выделить фрагмент с помощью клавишам – стрелок при нажатой клавише ***Shift*** (не отпускать, пока не будет завершено выделение);
- копировать выделенный фрагмент в буфер обмена - ***Ctrl+Ins*** или ***Shift+Del***, в последнем случае выделенный фрагмент будет удален из программы;
- копировать содержимое буфера обмена в программу в точку, где находится курсор ***Shift+Ins***;
- удалить выделенный фрагмент – ***Ctrl+Del***.

Доступ к командам копирования возможен через меню: нажав клавишу ***F10,*** выбрать команду **Edit,** появится подменю команд редактирования текста.

Примечание: Приведенная выше последовательность операций не оптимальна, но наиболее проста для первого знакомства с ПЭВМ. Удачи вам!

1 Понятие алгоритма. Блок-схема

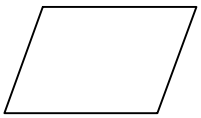
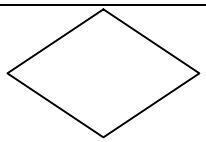
Алгоритм – конечная последовательность предписаний, однозначно определяющая процесс преобразования исходных данных в результат решения задачи.

В процессе разработки алгоритма могут использоваться различные способы его описания. Наиболее распространенные:

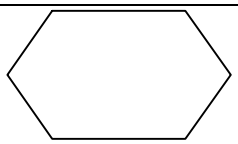
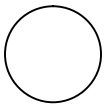
- словесная запись;
- графические схемы алгоритмов (блок-схемы);
- псевдокод (формальные алгоритмические языки);
- структурограммы.

Блок-схема – это графическое представление алгоритма, дополненное элементами словесной записи. На блок-схеме каждый пункт алгоритма изображается соответствующей геометрической фигурой. В таблице 2 приведены графические элементы, на которых компонуются блок-схемы, их названия и символы.

Таблица 1 – Графические элементы блок-схем

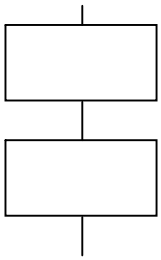
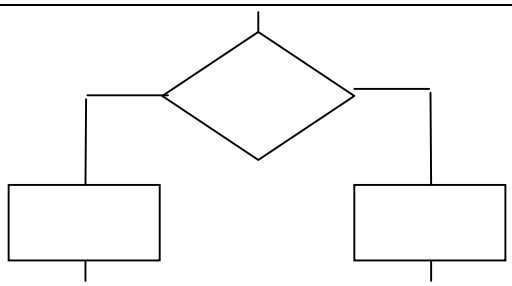
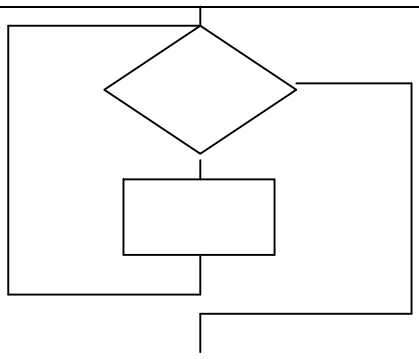
Название блока	**Блок**	**Отображаемая функция**
Начало-конец		Начало, конец, вход – выход в программах
Блок ввода-вывода		Ввод данных либо вывод результатов на экран
Блок вывода		Вывод данных на печать
Процесс		Вычисление или последовательность вычислений
Предопределенный процесс		Выполнение подпрограммы
Альтернатива		Проверка условий

Продолжение таблицы 1

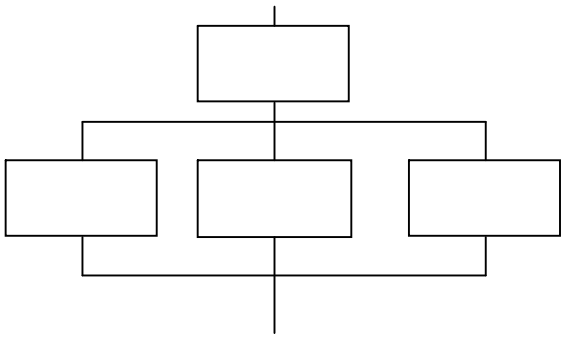
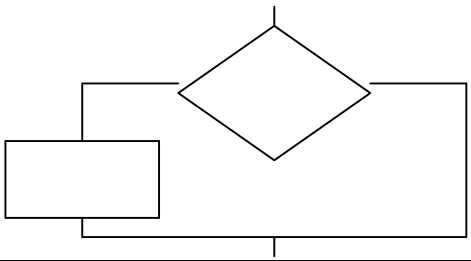
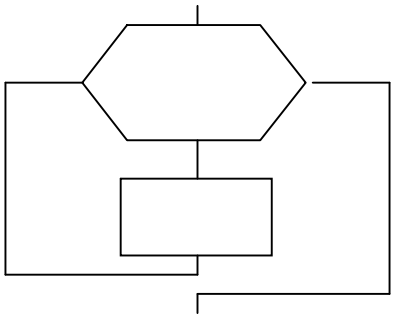
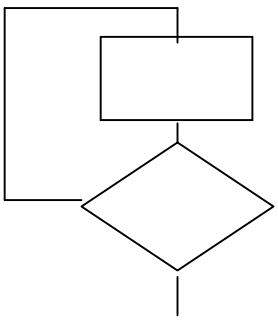
Название блока	**Блок**	**Отображаемая функция**
Модификация		Начало цикла
Соединитель		Разрыв линий потока информации в пределах одной страницы

В таблице 3 приведены основные базовые элементарные структуры для составления блок-схем.

Таблица 2 – Базовые структуры блок-схем

Название типа структуры	**Изображение**
Основные	
Последовательность	
Разветвление (выбор)	
Цикл с предусловием	

Продолжение таблицы 2

Название типа структуры	**Изображение**
Дополнительные	
Выбор варианта	
Сокращенная запись разветвления	
Цикл с параметрами	
Цикл с постусловием	

2 Основные конструкции языка Pascal

2.1 Алфавит языка

В программе можно использовать только символы, входящие в алфавит языка:

а) 26 латинских букв (как прописные, так и строчные);

б) цифры от 0 до 9;

в) знаки препинания: точка (**.**), точка с запятой (**;**), двоеточие (**:**), запятая (,), апостроф (’);

г) знаки арифметических операций: плюс (**+**), минус (**-**), умножение (*****), деление (**/**);

д) знаки операций отношения: больше (>), меньше (<), равно (=);

е) скобки: круглые **( )** , фигурные **{ }** , квадратные **[ ]** ;

ж) пробел.

В тексте, заключенном в апострофы, можно использовать любые символы, имеющиеся на клавиатуре дисплея.

В языке Pascal имеется такое понятие, как комментарий - это взятая в фигурные скобки любая последовательность символов, не содержащая закрывающей фигурной скобки.

2.2 Данные

2.2.1Виды данных

Данные бывают двух видов - константы и переменные.

Константы - это такие данные, значения которых, будучи определенными перед выполнением программы, не изменяются при ее выполнении.

Переменные - это такие данные, значения которых могут изменяться в процессе выполнения программы.

Каждое данное (как константа, так и переменная) относится к какому-либо типу. Под типом данного понимается множество его допустимых значений. Тип данного определяет также множество допустимых над ним действий.

В языке Pascal существует несколько групп типов. Мы рассматриваем только две из них: простые и структурированные типы данных.

Из группы простых типов данных, в свою очередь, мы рассмотрим только шесть основных типов (табл.3):

Таблица 3 – Типы данных

Тип	Имя типа	Содержание
Целый	**INTEGER**	Целые числа в интервале от32768 до 32767
Действительный (вещественный)	**REAL**	Вещественные числа в интервале от 10^{-38} до 10^{38}
Логический	**BOOLEAN**	Логические данные. Могут принимать значения TRUE (истина) или FALSE (ложь)
Символьный (литерный)	**CHAR**	Данные символьного типа. Могут принимать значения одной литеры из набора символов в ЭВМ
Перечисляемый		Набор значений, которые может принимать данное
Диапазон		Подмножество упорядоченных значений, определяемое минимальным и максимальным значениями

Среди типов данных, используемых в языке Pascal, есть стандартные (предопределенные) типы и типы, определяемые пользователем.

К стандартным типам, не требующим предварительного определения, относятся типы INTEGER, REAL, CHAR, BOOLEAN и STRING. Все остальные используемые в программе типы должны быть определены либо в разделе описания типов, либо в разделе описания переменных.

2.2.2 Идентификаторы

Идентификаторы - это имя любого объекта программы: переменной, константы, типа, метки и т.д. Идентификатор может включать буквы, цифры и символ подчеркивания. (Пробелы в идентификаторе недопустимы.) Начинаться идентификатор должен с буквы (буквы только латинские). Прописные и строчные буквы в идентификаторе не различаются: так NAME и name будут идентичны. Длина идентификатора может быть любой, но существенными являются только первые 63 символа.

2.2.3Константы

Тип константы простого типа однозначно определяется ее значением и явно не описывается.

Описание констант имеет вид:

CONST < Имя константы > = Значение;

Имя константы содержит максимум 40 символов и должно начинаться с буквы (используются только латинские буквы).

Пример: CONST PT=31.592; TT=’S’; A=7;

Числа с дробной частью записываются в двух формах: основной и полулогарифмической (с порядком). Вместо запятой ставится десятичная точка. Вместо основания степени 10 ставится буква E, что позволяет четко отделить мантиссу от показателя степени и записать все символы числа в одной строке. Незначащий нуль перед десятичной точкой может быть отброшен. В записи числа, имеющего только целую часть, десятичная точка не используется. Знак "+" может быть опущен.

Примеры записи чисел приведены в табл. 4.

Таблица 4 – Примеры записи чисел

Число	**Запись в *Pascal***
2,87	2.87
-0,315	-0.315 или -.315
184	184 или +184
210000	2.1E+5, 2.1E5, 21E4 …
-0,00045	-0.00045, -.00045, -45E-5, -.45E-3
-1000	-1E+3, -1E3, -E3

В Pascal по умолчанию определены специальные константы:

PI=3.14159265… ; MAXINT - наибольшее целое число, равное 32767.

2.2.4 Переменные

Раздел описания переменных начинается с ключевого слова VAR, после которого перечисляются переменные, массивы и их типы. Общий вид:

VAR < Имя переменной > , … , < имя переменной > : < тип переменной >;

Имя переменной - описывается аналогично именам констант.

Например:

```
VAR  TOP :  INTEGER;
        X , Y : REAL;
        A : ARRAY[1..10] OF INTEGER;
        B  : ARRAY[1..5,1..7] OF REAL;
```

2.3 Стандартные функции

В процессе решения задач часто возникает необходимость в вычислении элементарных функций. Для обращения к функции необходимо в выражении записать идентификатор функции и в круглых скобках - аргумент. Аргументами функции могут быть константы, переменные, функции или выражения. Для тригонометрических функций угол следует задавать в радианах. Различают стандартные арифметические функции, применяемые в **Pascal** (табл. 5) и стандартные функции преобразования (табл.6).

Таблица 5 – Стандартные арифметические функции

Функция	Математическая запись	Запись в **Pascal**
Синус	***sin(x)***	**SIN (X)**
Косинус	***cos(x)***	**COS (X)**
Арктангенс	***arctan(x)***	**ARCTAN (X)**
Абсолютное значение	$\|x\|$	**ABS (X)**
Корень квадратный	$\sqrt{X}$	**SQRT (X)**
Вычисление экспоненты	e^X	**EXP (X)**
Натуральный логарифм	***ln (x)***	**LN (X)**
Возведение в квадрат	x^2	**SQR (X)**
Присвоение знака	***sign (x)***	**SGN (X)**
Вычисление числа π	π	**PI**

Для возведения переменной x в некоторую степень a используют известное равенство

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

Тогда на языке Pascal оно выглядит: **EXP (A * LN (X))** .

Таблица 6 - Стандартные функции преобразования

Функция	Содержание функции
TRUNC (X)	Вычисляет целую часть аргумента X

Продолжение таблицы 6

Функция	Содержание функции
ROUND (X)	Определяет округленное значение X
ORD (X)	Определяет порядковый номер аргумента X в упорядоченном множестве значений, определяемом типом X
CHR (X)	Определяет символ, порядковый номер которого равен аргументу X
SUCC (X)	Выдает значение, следующее за аргументом X в списке значений, определяемом для типа X
PRED (X)	Выдает значение, предшествующее аргументу X в списке значений, определяемом для типа X

Функции **ORD(X)** , **SUCC(X)** и **PRED(X)** определены только для порядкового типа.

2.4 Выражения

Различают выражения арифметические, строковые и выражения типа отношения.

2.4.1 Арифметические выражения

Арифметические выражения определяют последовательность вычисления значения. Выражения могут включать в себя константы, переменные, стандартные функции, которые разделяются скобками и знаками операций.

Знаки арифметических операций: "+" (сложение), "-" (вычитание), "*" (умножение), "/" (деление), DIV (деление на целое), MOD (определение остатка).

Действия выполняются слева направо с соблюдением следующего старшинства:

а) выражения в скобках;

б) мультипликативные операции (* , / , DIV , MOD , AND, NOT);

в) аддитивные операции (+ , - , OR).

Тип результата операций выражения зависит от типов операндов, участвующих в операции. Тип результата + , - , * является INTEGER , если оба операнда имеют тип INTEGER, и REAL - в противном случае.

Результатом операции / всегда является тип REAL, а результат операций MOD, DIV всегда имеет тип INTEGER, так как аргументы могут быть только типа INTEGER.

Пример.

Математическая запись:

а) $\sqrt{1 + LN(1{,}3X) + COS(A - T)}$; б) $2^{x}COS(BX) - 3^{x}SIN(BX)$.

Запись на **Pascal**:

а) SQRT (1+LN (1.3 * X) + COS (A - T));

б) EXP(X * LN (2))* COS (B * X) - EXP(X *LN (3))* SIN (B * X).

2.4.2 Отношения

Выражения типа отношений служат для установления отношений между двумя числовыми или строковыми значениями. Допустимы 6 операций: "=" (равно), "< >" (не равно), "<" (меньше), ">" (больше), "<=" (меньше или равно), ">=" (больше или равно).

Возможно выполнение логических операций: NOT - отрицание, AND - умножение (логическое и), OR - сложение (логическое или).

Пример: A = B;

(A > 0) AND (B > 0).

2.5 Структура программы

Программа на языке Pascal состоит из заголовка, блока и заканчивается точкой. Блок, в свою очередь, содержит раздел описаний и раздел операторов. Раздел операторов включает в себя последовательность исполняемых операторов, разделенных точкой с запятой (;) и ограниченных операторными скобками - служебными словами **BEGIN END**.

Общая структура программы

PROGRAM < имя > ;

USES CRT;

LABEL

< метка >, ... , < метка >;

CONST

< имя константы > = < константа >;

⋮

< имя константы > = < константа >;

TYPE

< имя типа > = < тип >;

⋮

< имя типа > = < тип >;

VAR

< имя переменной >,...,< имя переменной >: < тип >;

⋮

< имя переменной >,...,< имя переменной >: < тип >;

PROCEDURE < заголовок процедуры >;

< блок >;

FUNCTION < заголовок функции >;

< блок >;

BEGIN

< оператор >;

⋮

< оператор >

END.

Приведенные в программе элементы имеют значения:

PROGRAM - заголовок программы.

LABEL - раздел описания меток. Метка - целое число без знака, содержащее не более четырех цифр, или обычный идентификатор. Метка от оператора отделяется двоеточием. Метка используется для пометки оператора, на который осуществляется переход. Все используемые в программе метки должны быть определены в разделе описания меток.

CONST - раздел описания констант. Служит для присвоения константам некоторых значений.

TYPE - раздел описания типов. Служит для определения простых и структурных типов данных, задаваемых пользователем.

VAR - раздел описания переменных.

PROCEDURE, FUNCTION - разделы описания процедур и функций. Эти разделы присутствуют в программе, если помимо стандартных процедур и функций в программе определяются свои, являющиеся самостоятельными программными единицами, к которым осуществляется обращение из основной программы.

Обязательным является только раздел операторов. Все остальные элементы программы могут отсутствовать.

Заголовок программы носит чисто декоративный характер и игнорируется компиляторами.

В Pascal порядок размещения разделов описаний произвольный, единственное правило, которое необходимо выдержать - можно использовать лишь те идентификаторы, которые перед этим были определены.

2.6 Операторы

2.6.1 Запись операторов

Операторы - это описание каких-либо действий над переменными и константами. Тело программы можно рассматривать как последовательность операторов.

Операторы отделяются друг от друга точкой с запятой. В одной строке программы можно записать несколько операторов. И, наоборот, один оператор может размещаться на нескольких строках.

2.6.2 Составной оператор

Составной оператор - это совокупность последовательно выполняемых операторов, заключенных в операторные скобки BEGIN и END.

Внутри операторных скобок операторы также отделяются друг от друга точкой с запятой. BEGIN - это не оператор. Поэтому после него точка с запятой не ставится. Перед END точка с запятой допускается, но ее наличие не обязательно.

Составной оператор может понадобиться в тех случаях, когда в соответствии с правилами построения конструкций языка допустимо использование только одного оператора, а требуется выполнить несколько операторов. Таким единственным оператором может выступать составной оператор, в который входит ряд операторов, выполняющих требуемые действия.

В дальнейшем везде, где будет сказано, что можно использовать один оператор, им может быть составной оператор.

3 Операторы присваивания, ввода, вывода. Программирование линейного вычислительного процесса

3.1 Теоретические сведения

3.1.1 Оператор присваивания

Оператор присваивания служит для вычисления значения выражения и присваивания его переменной.

Формат оператора:

имя переменной := выражение ;

Вычисляется значение выражения, записанного справа, а затем переменной присваивается это значение.

Имя переменной и результат должны принадлежать одному типу. Исключение может составлять случай, когда выражение имеет значение целого типа, а имя результата - действительного типа.

3.1.2 Оператор ввода

Ввод информации осуществляется при помощи операторов:

READ(b1,b2, ... , bn);

READLN(b1,b2, ... , bn);

READLN; ,

где **b1,b2, ... , bn** - имена переменных, значения которых вводятся.

Оператор READ(b1,b2, ... ,bn); осуществляет ввод данных.

Оператор READLN(b1,b2, ... ,bn); осуществляет ввод данных и обеспечивает переход к началу новой строки. Переменные вводятся последовательно в одной строке и отделяются друг от друга запятыми.

Оператор READLN; обеспечивает пропуск одной строки и переход к началу новой строки.

3.1.3 Оператор вывода

Для вывода информации используются операторы:

WRITE(b1,b2, ... , bn);

WRITELN(b1,b2, ... , bn);

WRITELN; ,

где **b1,b2, ... , bn** - выражения, значения которых выводятся. (В частности, это могут быть идентификаторы переменных или констант).

Оператор WRITE(b1,...,bn); выполняет вывод значений, соответствующих выражений, размещая выводимые значения в одной строке.

Оператор WRITELN(b1,b2, ... ,bn); выполняет вывод значений и после вывода последнего значения осуществляет переход к новой строке.

Оператор WRITELN; обеспечивает пропуск строки и переход к началу новой строки.

Значения выражений в списке операторов вывода могут принадлежать к целому, вещественному, символьному или логическому типам.

Общий вид записи операторов вывода значений целого типа:

WRITE(b : m);

WRITELN(b : m); ,

а для вывода действительного типа:

WRITE(b : m : n);

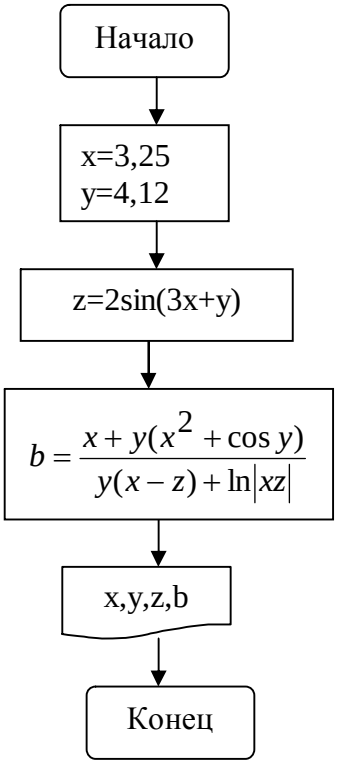
WRITELN(b : m : n); ,

где **b** - арифметическое выражение; **m** - поле, отводимое под значение; **n** - часть поля, отводимого под дробную часть числа.

3.2 Пример

Составить программу для вычисления функций: $b = \frac{x + y(x^2 + \cos y)}{y(x - z) + \ln|xz|}$, где z=2sin(3x+y); x=3,25; y=4,12.

```
program P_1;
uses crt;
  const x=3.25;y=4.12;
  var z,b:real;
begin
  z:=2*sin(3*x+y);
  b:=(x+y*(sqr(x)+cos(y)))/(y*(x-z)+ln(abs(x*z)));
  writeln('При x=',x:4:2,' y=',y:5:2,' z=',z:5:2,'
b=',b:5:2);
 end.
```



4 Программирование разветвляющегося вычислительного процесса

4.1 Теоретические сведения

Разветвления в программе возникают при необходимости выбора одного из нескольких возможных путей в решении задачи.

Для организации разветвлений в программах используются операторы перехода (условный и безусловный) и выбора.

Оператор безусловного перехода имеет вид:

GOTO n; ,

где **n** - метка оператора.

Оператор условного перехода позволяет изменить порядок выполнения операторов в программе в зависимости от определенных условий. Общий вид оператора условного перехода:

IF < условие > THEN

< оператор 1 >

ELSE

< оператор 2 >;

Если условие, заданное в операторе **IF,** истинно, то выполняется **THEN** - ветвь, т. е. оператор (простой или составной), стоящий после **THEN (< оператор 1 >)**. В противном случае выполняется **ELSE**-ветвь, оператор (он тоже может быть составным), стоящий после **ELSE (< оператор 2 >**). После выполнения одной из ветвей работа программы продолжается с оператора, следующего за **IF**. Используется также усеченный формат оператора условного перехода:

IF < условие > THEN < оператор >;

При его использовании **THEN**-ветвь операторов выполняется при истинном условии, если же условие ложно, она игнорируется и сразу выполняется оператор, стоящий за оператором **IF**.

Напомним, что если в какой-то ветви требуется выполнить более одного оператора, из них необходимо образовать составной оператор, т.е. заключить эти операторы в операторные скобки **BEGIN** и **END**.

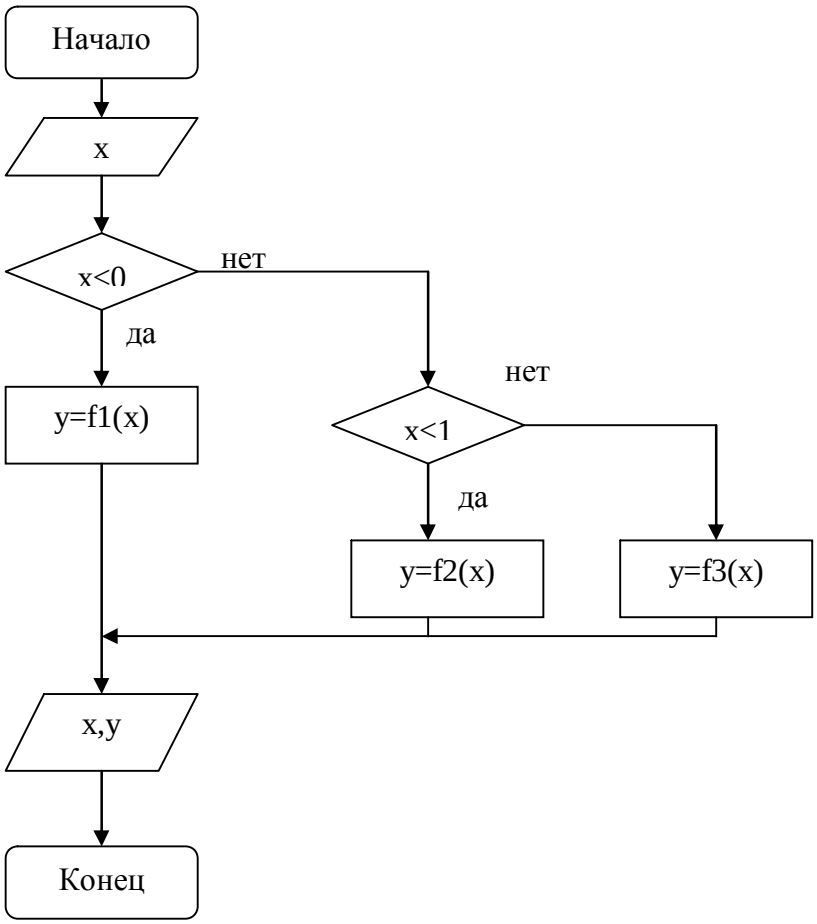
4.2 Пример

Вычислить значение функции

$$\mathbf{Y}=\begin{cases} (2x+1)/x^5 & \text{если } x<0, \\ e^{x+1}+\cos(x), & \text{если } 0\le x\le 1, \\ 3\ln\sqrt[5]{\sin(x)+x^2}, & \text{если } x>1. \end{cases}$$

Значение x запрашивать в диалоге.

```
program  p_2;
uses crt;
   var    x,  y , m : real;
begin
   clrscr;
   writeln('введите значение x ');readln(x);
   clrscr;
   if  x<0  then  y:=(2*x+1)/(x*x*x*x*x)
        else
        if  x=0  then y:=exp(x+1)+cos(x)
             else
                begin
                   m:=sin(x)+sqr(x);
                   y:=3*ln(exp(ln(m)/5))
                end;
   writeln('   при x= ',x:6:3,'   y= ',y:12:5);
   r:=readkey
end.
```



5 Программирование алгоритмов циклической структуры

5.1 Теоретические сведения

Циклическим называется вычислительный процесс, содержащий многократные вычисления по одним и тем же математическим зависимостям, но для различных значений входящих в него переменных.

5.1.1 Цикл WHILE ("пока")

Цикл с проверкой условия в начале цикла. С помощью конструкции **WHILE...DO** можно реализовать циклический процесс, состоящий из ряда операторов, который выполняется до тех пор, пока выполняется определенное условие.

WHILE < условие > DO

< ***Оператор*** >;

Если в цикле необходимо выполнить более одного оператора, то их следует заключить в операторные скобки begin end , т.е. образовать из них составной оператор.

До тех пор, пока соблюдается условие, последовательно выполняется тело цикла (< оператор >). Если условие не соблюдается, то выполнение программы продолжается, начиная с оператора, следующего за циклом.

5.1.2 Цикл REPEAT

Цикл с проверкой условия в конце цикла. Тело цикла выполняется до тех пор, пока не станет истинным условие.

REPEAT

< ***Оператор 1>;***

. . .

< ***Оператор n>;***

UNTIL < условие >;

5.1.3 Цикл со счетчиком

Общий вид записи:

при увеличении значения параметра -

FOR i := m1 TO m2 DO <Оператор>;

при уменьшении значения параметра -

FOR i := m1 TO m2 DOWTO <Оператор>; ,

где **i** - переменная порядкового типа , которая изменяется при повторении цикла, ее часто называют управляющей переменной;

m1 - выражение, задающее начальное значение счетчика;

m2 - выражение, задающее конечное значение счетчика;

DO - изменение параметра цикла с шагом +1;

DOWTO - изменение параметра цикла с шагом -1.

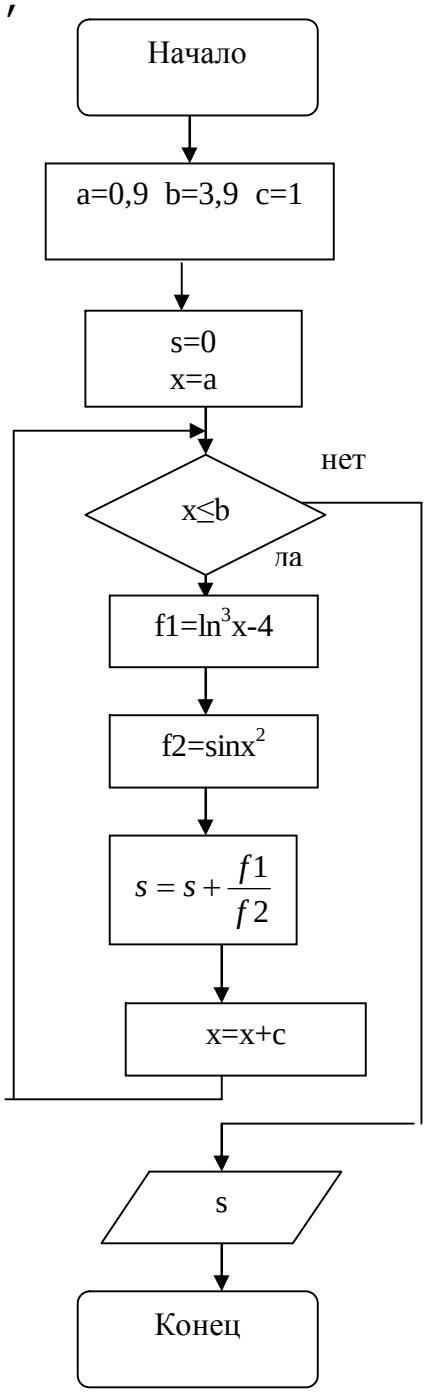
5.2 Пример

Найти сумму ряда $y = \sum \frac{\ln^3 x - 4}{\sin(x)^2}$, где 0,9≤x≤3,9, x меняется с шагом h=1:

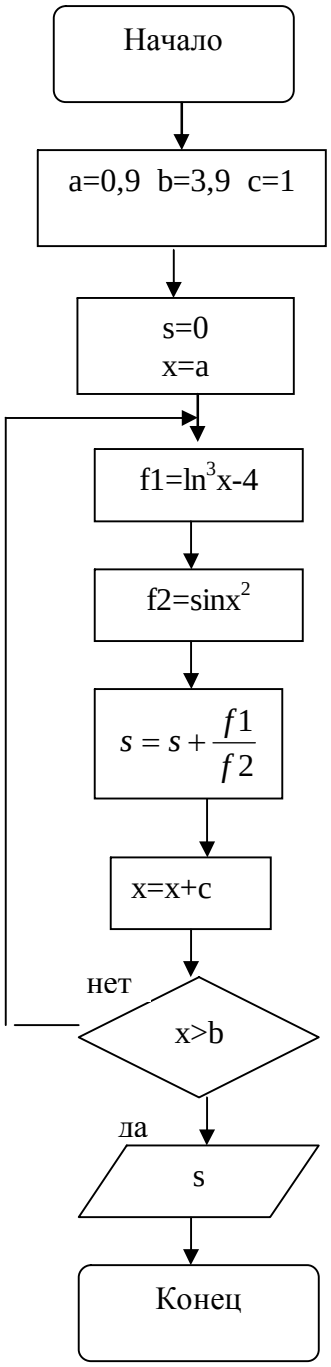
а) используя цикл WHILE;

б) используя цикл REPEAT.

а)
```
program p_3a;
uses crt;
  const a=0.9;b=3.9;c=1;
  var x,s,f1,f2:real;
begin
 x:=a; s:=0;
 while x<=b do
    begin
        f1:=sqr(ln(x))*ln(x)-4;
        f2:=sin(sqr(x));
        s:=s+f1/f2;
        x:=x+c;
    end;
  writeln('Y=',y:7:3);
end.
```



б)
```
program P_3b;
uses crt;
  const a=0.9;b=3.9;c=1;
  var x,s,f1,f2:real;
begin
 x:=a; s:=0;
 repeat
        f1:=sqr(ln(x))*ln(x)-4;
        f2:=sin(sqr(x));
        s:=s+f1/f2;
        x:=x+c;
 until x>b;
writeln('Y=',y:7:3);
end.
```



6 Табулирование функций

6.1 Теоретические сведения

Для табулирования функций используется цикл с заданным числом повторений, вычисленным по формуле

$$\mathbf{n} = \left[\frac{x_k - x_n}{h} \right] + 1 \,,$$

где x_k, x_n - конечное и начальное значения аргумента;

h - шаг изменения аргумента;

знак "[ ]" означает, что берется целая часть от деления.

Перед первым выполнением цикла необходимо задать начальное значение аргумента, а затем **n** раз выполнять вычисления и печать значений аргумента и функции.

При каждом новом выполнении цикла необходимо изменять аргумент на величину шага.

6.2 Пример

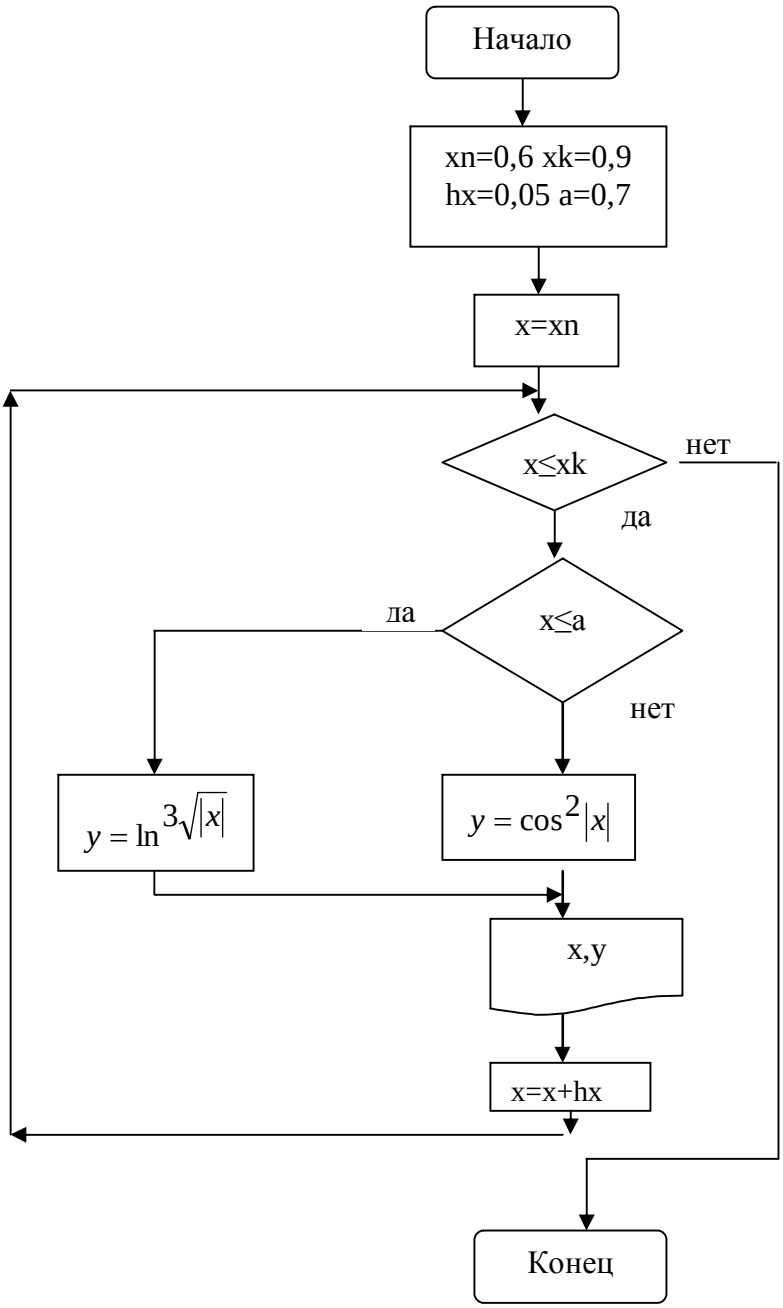
Вычислить таблицу значений функции

$$y = \begin{cases} \ln^3 \sqrt{|x|}\,, & \text{если } x \le 0{,}7, \\ \cos^2 |x|\,, & \text{если } x > 0{,}7. \end{cases}$$

для значений аргумента x в интервале от 0,6 до 0,9 с шагом 0,05.

```
program p_4;
use crt;
  var xn,xk,hx,a,x,y:real;
begin
  xn:=0.6; xk:=0.9; hx:=0.05; a:=0.7;
  writeln(' ТАБЛИЦА ЗНАЧЕНИЙ X,Y');
  writeln('-----------------------');
  writeln('!    X     !     Y    !');
  writeln('-----------------------');
  x:=xn;
  while x<=xk do
    begin
       if x<=a then
           y:=sqr(ln(sqrt(abs(x))))*ln(sqrt(abs(x)))
```

```
            else
            y:=sqr(cos(abs(x)));
        writeln('!',x:9:2,' !',y:9:4,' !');
        x:=x+hx;
    end;
  writeln('-----------------------');
end.
```



7 Нестандартные и ограниченные типы данных. Оператор выбора варианта

7.1 Теоретические сведения

Язык Pascal предоставляет возможность для задания дополнительных типов, отличных от стандартных. Новые типы описываются в специальном разделе типов или определяются непосредственно при описании переменных.

7.1.1 Перечисляемый тип

Такой тип задается перечислением значений, которые может принимать переменная. Общая форма задания перечисляемого типа такова :

TYPE T = (A1, A2, . . . , AN); ,

где **T** - имя нового типа; **A1, A2, ... , AN** определяют константы нового типа данных. Последовательность значений, составляющая перечисляемый тип, упорядочена.

Например, новый перечисляемый тип COLOR можно определить следующим образом :

```
TYPE  COLOR = ( RED,  BLUE,  BLACK);
```

Для значений перечисляемых типов определены стандартные функции SUCC, PRED, ORD (см. разд.2 “Основные конструкции языка Pascal”).

7.1.2 Ограниченный тип данных (тип диапазон)

Отрезок значений любого порядкового типа может быть определен как ограниченный тип (тип диапазон).

Общий вид

TYPE T = MIN .. MAX; ,

где **T** - имя типа; **MIN, MAX** - левая и правая границы диапазона. Например:

```
TYPE   DEN = 1 .. 31;
       GOD = 1900 .. 2000;
VAR    X : DEN;  Y : GOD;
       C : 1 .. 12;
```

7.1.3 Оператор выбора варианта

Оператор выбора варианта является обобщением условного оператора: он дает возможность выполнить один из нескольких операторов в зависимости от значения некоторого выражения, называемого селектором. В общем случае оператор имеет вид:

CASE < селектор > OF

< список меток 1 > : < Оператор 1 >;

< список меток 2 > : < Оператор 2 >;

⋮

< список меток n > : < Оператор n >;

[ELSE

< оператор >]

END; ,

где **< селектор > -** выражение любого типа, кроме вещественного;

< оператор >- любой оператор языка, в том числе и составной;

< список меток > - список разделенных запятыми значений выражения < селектор > или одного его значения.

Оператор выбора варианта выбирает для исполнения тот оператор, одна из меток которого равна текущему значению выражения < селектор >. Если значение выражения < селектор > не совпадает ни с одной из меток, выполняется оператор, соответствующий ELSE . Ветвь ELSE не обязательна. По окончании выполнения выбранного оператора управление передается в конец оператора CASE.

7.2 Пример

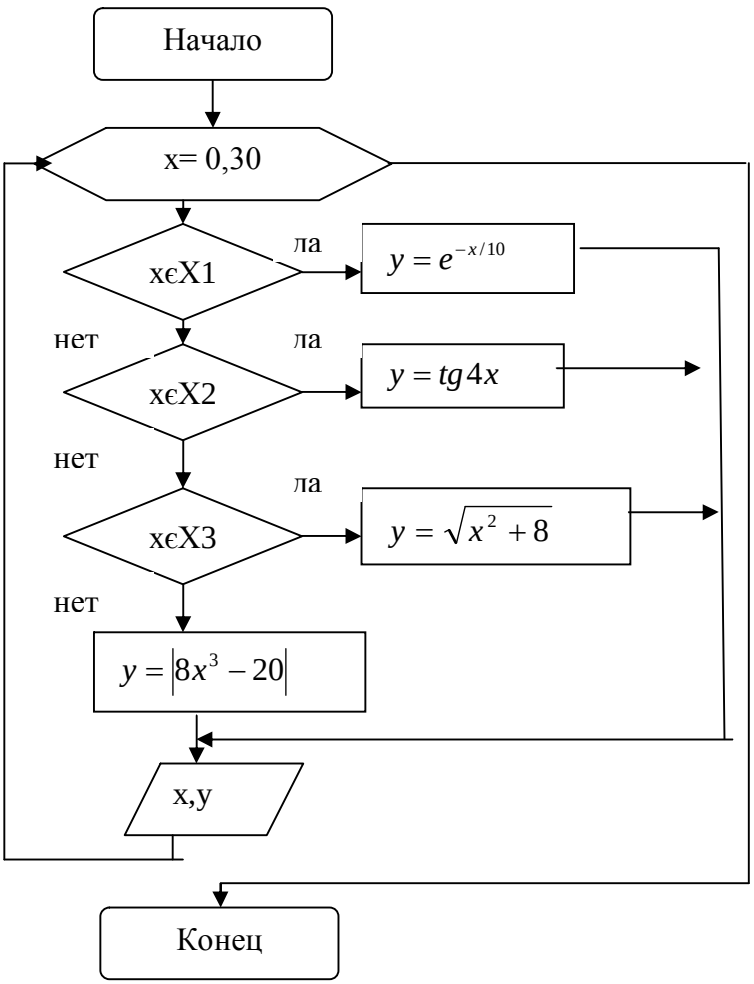
Вычислить таблицу значений функции

$$y = \begin{cases} e^{-x/10}, & \text{если } x \in X1, \\ \operatorname{tg} 4x, & \text{если } x \in X2, \\ \sqrt{x^2+8}, & \text{если } x \in X3, \\ \left|8x^3 - 20\right|, & \text{если } x \in X4. \end{cases}$$

для целочисленных значений аргумента x в интервале от 0 до 30.

Множество X1 – четные числа из интервала [10,20], X2 –нечетные числа из интервала [10,20], X3 – [1,8], X4 – остальные.

```
program p_5;
uses crt;
 type arg=0..30;
 var x:arg;y:real;
BEGIN
 ClrScr;
 for x:=0 to 30 do
  begin
    case x of
        10,12,14,16,18,20: y:=exp(-x/10);
        11,13,15,17,19: y:=sin(4*x)/cos(4*x);
        1,2,3,4,5,6,7,8: y:=sqrt(sqr(x)+8);
                         else y:=abs(8*sqr(x)*x-20);
    end;
    writeln('При x=',x:3,' y=',y:5:2);
  end;
end.
```



8 Селективная обработка одномерных массивов

8.1 Теоретические сведения

До сих пор мы имели дело с простейшими типами данных. Базируясь на простых типах, можно строить более сложные - структурированные типы данных, в частности, массивы. Массив - это структура, состоящая из фиксированного числа компонентов одного типа. Его можно рассматривать как упорядоченный набор данных одного типа, имеющий одно общее имя. Массив характеризуется своим именем и размерностью.

Общий вид описания типа массив:

TYPE T = ARRAY [T1,T2, ..., Tk] OF TC;

где **T -** имя массива ;

T1, ..., Tk - типы индексов;

TC - тип компонентов / базовый тип.

Количество индексов k - размерность массива. Индексы могут быть любого типа, кроме REAL и INTEGER.

После ввода типа массива можно вводить переменные этого типа:

VAR M: T; ,

где M - идентификатор массива.

Массивы можно описывать в разделе описаний переменных непосредственно:

VAR M: ARRAY [T1, T2, ... Tn] of TC.

Здесь массив с именем M состоит из восьми элементов (M1,M2, ... , M8) вещественного типа.

Доступ к любому элементу массива осуществляется указанием идентификатора массива, за которым в квадратных скобках следует индексное выражение. Индексное выражение должно давать значения, лежащие в диапазоне, определяемом типом индекса. Для объявленного выше массива M в программе доступны следующие индексные переменные: M[1], M[2], ... , M[8].

Количество **k** индексов, необходимое при обращении к элементу массива, определяет k - мерность массива. При k = 1 массив называется *одномерным,*. при k = 2 - *двумерным.* Одномерный массив соответствует понятию вектора (линейной таблицы, строки), двумерный массив - понятию матрицы (набора векторов, прямоугольной таблицы).

Селективная обработка массива - это выделение из массива элементов, удовлетворяющих условию, и обработка выделенных фрагментов. Часто из выделенных элементов формируют новый (рабочий) массив и его обрабатывают.

Наиболее часто встречаются такие условия обработки элементов массива:

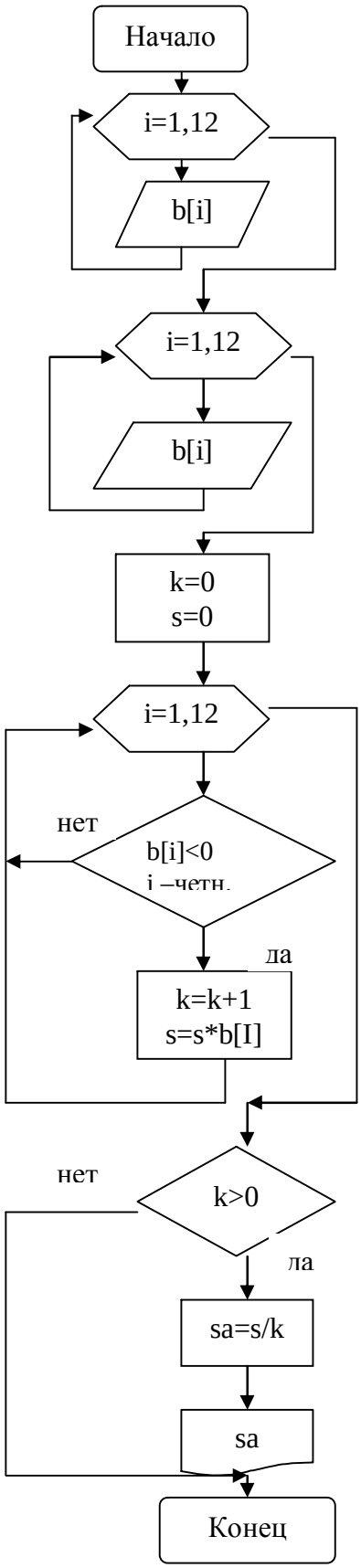
четные	A[i] mod 2 = 0 ;
нечетные	A[i] mod 2 <> 0 ;
кратные k	A[i] mod k = 0 ;
некратные k	A[i] mod k <> 0 ;
на четных местах	i mod 2 = 0 ;
на нечетных местах	i mod 2 <> 0 ;
положительные	A[i] > 0 ;
отрицательные	A[i] < 0 ;
в интервале (x1,x2)	(A[i]>x1) and (A[i]<x2).

8.2 Пример

Вычислить среднее арифметическое отрицательных элементов массива B(12), стоящих на четных местах.

```
program p_6;
 uses crt;
 type mas=array[1..12] of integer;
 var b:mas;i,k,s:integer;sa:real;
BEGIN
 clrscr;
 writeln('Введите 12 элементов массива B:');
 for i:=1 to 12 do read(b[i]);
 writeln('Исходный массив B:');
 for i:=1 to 12 do write(b[i]:4);
 writeln;
 s:=0;
 k:=0;
 for i:=1 to 12 do
 if (b[i]<0) and (i mod 2=0) then
  begin
    s:=s+b[i];
    k:=k+1;
  end;
 writeln('Сумма отрицательных с четными индексами
s=',s:6);
 writeln('Количество k=',k:4);
```

```
 if k>0 then
  begin
    sa:=s/k;
    write('Среднее арифметическое sa=',sa:5:2);
  end
 else write('Отрицательных с четными индексами нет');
end.
```



9 Нахождение наибольшего и наименьшего элементов

9.1 Теоретические сведения

Нахождение наибольшего и наименьшего значений массива выполняется в цикле, в котором текущий элемент массива сравнивается с наибольшим (наименьшим) из всех предыдущих элементов массива. Если текущий элемент окажется больше (меньше) наибольшего (наименьшего) из предыдущих, то его надо считать новым наибольшим (наименьшим). В противном случае наибольшее (наименьшее) сохраняет старое значение, т.е.

$$Y_{max} = \begin{cases} Y_i, & \text{если } Y_i > Y_{max}, \\ Y_{max}, & \text{если } Y_i \le Y_{max}. \end{cases}$$

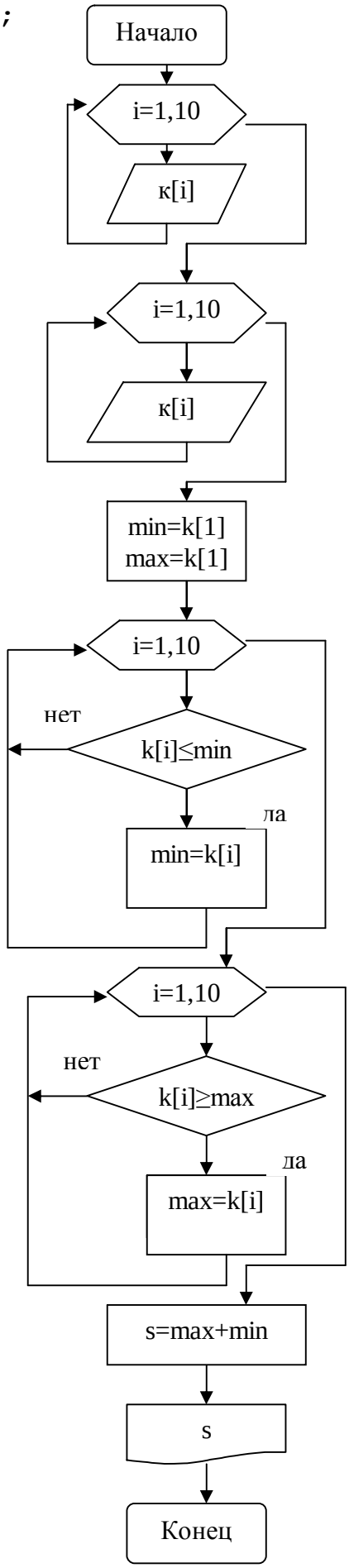
$$Y_{min} = \begin{cases} Y_i, & \text{если } Y_i < Y_{min}, \\ Y_{min}, & \text{если } Y_i \ge Y_{min}. \end{cases}$$

9.2 Пример

Найти сумму наибольшего и наименьшего элементов массива.

```
program p_8;
 uses crt;
  type mas=array[1..10] of integer;
 var k:mas;i,min,max,s:integer;
begin
 clrscr;
 writeln('Ведите массив K:');
 for i:=1 to 10 do  read(k[i]);
 writeln;
 writeln('Выведите массив K:');
 for i:=1 to 10 do write(k[i]:3);
 writeln;
min:=k[1];
max:=k[1];
for i:=1 to 10 do
begin
if k[i]<min then min:=k[i];
if k[i]>max then max:=k[i];
end;
writeln('Минимальный элемент =',min:4);
writeln('Максимальный элемент =',max:4);
s:=max+min;
```

```
writeln('Сумма =',s:4);
end.
```



10 Вложенные циклы. Обработка двухмерных массивов

10.1 Теоретические сведения

Если телом цикла является циклическая структура, то такие циклы называют вложенными или сложными. Цикл, содержащий в себе другой цикл, называют внешним. Цикл, содержащийся в теле другого цикла, называют внутренним.

Двумерный массив характеризуется именем и размерностью, где первый индекс определяет максимальное число строк, а второй - максимальное число столбцов. Элемент массива - переменная, расположенная на пересечении соответствующих столбца и строки. При обработке таких массивов обычно и применяют вложенные циклы.

В Pascal многомерные массивы могут определяться последовательно: сначала объявляется один массив, затем второй, элементами которого являются объявленные ранее массивы, и т.д. Один массив вкладывается в другой, и степень такого вложения неограниченна. Например, для матрицы A(2,3)

```
TYPE
        STROKA = ARRAY [1..3]  OF  REAL;
        MATR = ARRAY [1..2]  OF  STROKA;
VAR
        V : STROKA;
        A : MATR;
```

Переменная A имеет смысл двумерного массива из двух строк, в каждую из которых включено по три элемента.

Описание A можно сократить, исключив определение типа STROKA в определении типа MATR :

```
TYPE
        MATR = ARRAY [1..2] OF ARRAY [1..3] OF REAL;
VAR
        A : MATR;
```

или

```
TYPE
        MATR = ARRAY [1..2,1..3] OF REAL;
VAR
        A : MATR;
```

Если указанный тип используется для определения одного массива в программе, то удобно объявление массива в разделе описания переменных:

```
VAR
    A : ARRAY [1..2,1..3] OF REAL;
```

Ссылка на элемент матрицы A, лежащий на пересечении I -й строки и J - го столбца, имеет вид A[I,J].

При обработке матриц часто приходится выделять элементы:

k- й строки	-	A[k,j] j=1, … , m ,	
k- й колонки	-	A[i,k] i=1, … , n .	

а для квадратных матриц (m=n) также:

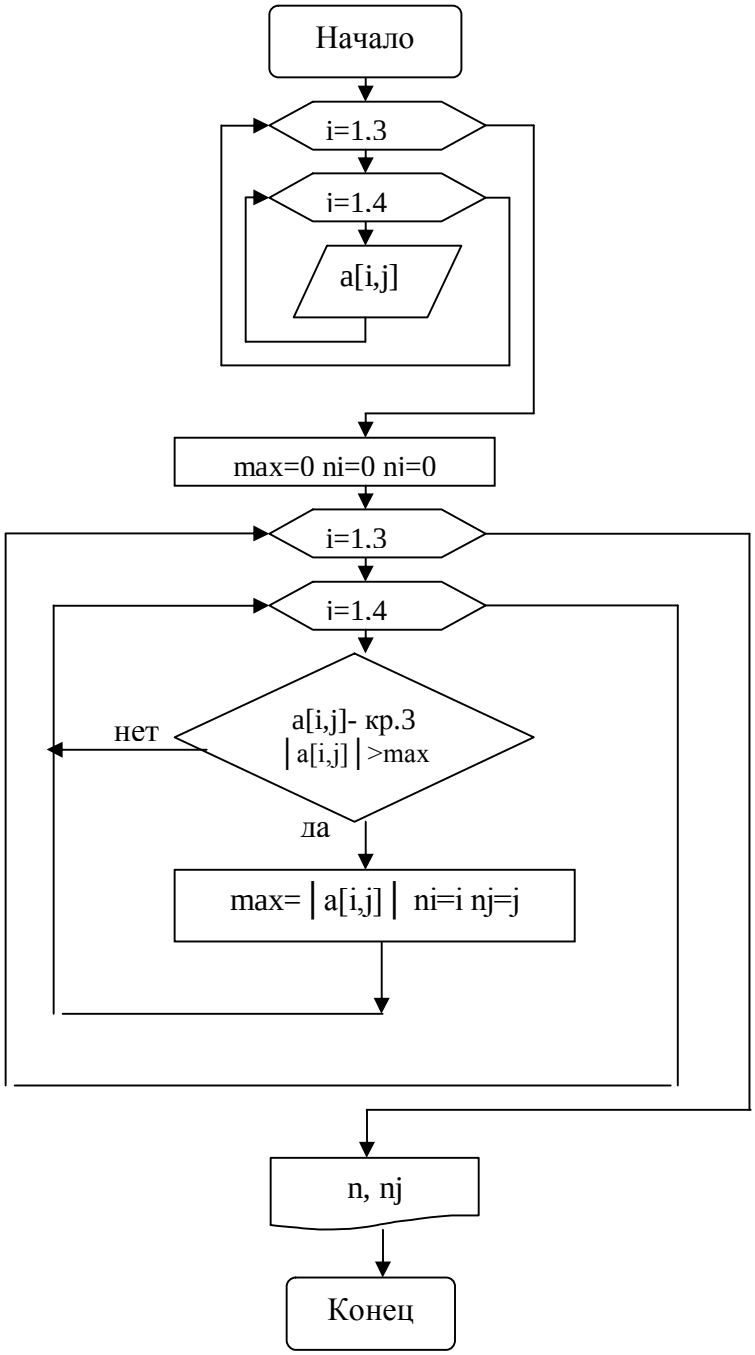
главной диагонали	-	A[i,i]	i=1, … , n ,
побочной диагонали	-	A[i,n+1-i]	i=1, … , n ,
наддиагональные	-	A[i,j]	i>j ,
поддиагональные	-	A[i,j]	i<j .

10.2 Пример

Найти индексы максимального по модулю кратного 3 элемента матрицы.

```
program p_8;
uses crt;
   const n=3;m=4;
   type matr=array[1..n,1..m]of integer;
   var i,j,indi,indj,max:integer;
   a:matr;
begin
   writeln('введите массив размера ',n,' x ',m);
   for i:=1 to n  do
   begin writeln;
   for j:=1 to m do read(a[i,j]);
   end;
   writeln('контрольный вывод массива:');
   for i:=1 to n do
   begin writeln;
   for j:=1 to m do write( ' ',a[i,j])
   end;
   max:=0;ni:=0;nj:=0;
   for i:=1 to n do
   for j:=1 to m do
   if (a[i,j] mod 3 =0) and (abs(a[i,j])>max) then
   begin
```

```
            max:=abs(a[i,j]);ni:=i;nj:=j
        end;
        writeln;
        writeln('искомый элемент стоит на пересечении');
        writeln('ni,'-ой строки и ',nj,' -го столбца ');
end.
```



Список литературы

1 Паскаль: Учеб. пособие В.С. Новичков, Н.И. Парфилова, А.Н. Пылькин. - М.: Высш. школа, 1990. - 223 с.

2 Практикум по основам программирования. Язык Паскаль Н.Д. Васюкова, В.В. Тюляева. - М.: Высш. школа, 1991.- 160 с.

3 Белецкий Я. Турбо-Паскаль с графикой для персональных компьютеров. - М.: Машиностроение, 1991. - 320 с.

4 Вычислительная техника и программирование: Практикум по программированию В.Е. Алексеев, А.С. Ваулин, Г.Б. Петрова.- М.: Высш. школа, 1991. - 324 с.

5 Введение в язык Паскаль Г.В. Абрамов, Н.П. Трифонов, Г.Н. Трифонова. - М.: Наука, 1988. - 320 с.

6 Программирование на языке Паскаль Г.Л. Семашко, А.И. Салтыков. - М.: Наука,1988. - 128 с.

7 Начала программирования на языке Паскаль С.А. Абрамов, Е.В. Зима. - М.: Наука,1987. - 112 с.

8 Перминов О.Н. Язык программирования Паскаль. - М.: Радио и связь, 1989. - 128 с.

9 Программирование на языке Паскаль для персональных ЭВМ ЕС А.Н. Вальвачев, В.С. Крисевич. - Минск: Высш. школа.,1989. - 223 с.

10 Сердюченко В.Я. Розробка алгоритмів та програмування мовою TURBO-PASCAL. – Харків: Парітет, 1995. – 352 с.

Конспект лекций

по дисциплине

«Информатика»

для студентов всех специальностей
заочной формы обучения

Часть 1

Составитель Ирина Анатольевна Гетьман

Редактор Нелли Александровна Хахина

Подп. в печать Формат 60х84/16.
Ризограф. печать. Усл. печ. л. 2,5 Уч.-изд. л. 1,81
Тираж экз. Заказ №

ДГМА. 84313, г. Краматорск, ул. Шкадинова, 72