ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

• ИЗДАТЕЛЬСТВО ТГТУ •

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТАМБОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Методические указания к выполнению лабораторных работ
по курсу «Основы алгоритмизации и программирования»
для студентов специальности 021100 всех форм обучения

Тамбов
• Издательство ТГТУ •
2004

УДК [34:681.31] (075)
ББК Х.С51
О75

Утверждено Редакционно-издательским советом университета

Рецензент
Кандидат технических наук, доцент
М.Ю. Серегин

Составители:
И.П. Рак, *А.В. Терехов*, *А.В. Селезнев*

О75 Основы алгоритмизации и программирования: Метод. указ. / Сост.: И.П. Рак, А.В. Терехов, А.В. Селезнев. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2004. 24 с.

Представлены методические указания для выполнения лабораторных работ по дисциплине «Основы алгоритмизации и программирования».

Предназначены для студентов специальности 021100 «Юриспруденция» всех форм обучения.

УДК [34:681.31] (075)
ББК Х.С51

УЧЕБНОЕ ИЗДАНИЕ

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Составители: РАК Игорь Петрович,
ТЕРЕХОВ Алексей Васильевич,
СЕЛЕЗНЕВ Андрей Владимирович

Редактор Е. С. Мордасова
Инженер по компьютерному макетированию Т. А. Сынкова

Подписано к печати 26.01.2004.
Формат 60 × 84/16. Гарнитура Times. Бумага газетная. Печать офсетная.
Объем: 1,39 усл. печ. л.; 1,45 уч.-изд. л.
Тираж 30 экз. С. 65^{M}

Издательско-полиграфический центр
Тамбовского государственного технического университета
392000, Тамбов, ул. Советская, 106, к. 14

ЗАПИСЬ АЛГОРИТМОВ С ПОМОЩЬЮ БЛОК-СХЕМ

Цель работы: научиться составлять алгоритм программы и записывать его с помощью блок-схем.

Основные понятия

Алгоритм – некоторая конечная последовательность правил (предписаний), определяющая процесс преобразования исходных и промежуточных данных в результат решения задачи.

Разрабатываемый алгоритм должен обладать следующими свойствами:

- *массовостью*, позволяющей решать не одну задачу, а целый класс задач;
- *детерминированностью*, однозначно определяющей выполняемые действия (промежуточные и окончательные результаты разных пользователей должны быть одинаковыми при одинаковых исходных данных);
- *результативностью*, позволяющей получить результат после конечного числа шагов.

По используемой структуре управления вычислительным процессом алгоритмы классифицируют следующим образом:

- линейной структуры;
- разветвляющейся структуры;
- циклической структуры;
- смешанной (комбинированной) структуры.

Алгоритм линейной структуры – алгоритм, в котором все действия выполняются последовательно друг за другом.

Алгоритм разветвляющейся структуры – алгоритм, в котором в зависимости от выполнения некоторого логического условия вычислительный процесс должен идти по одной или другой ветви.

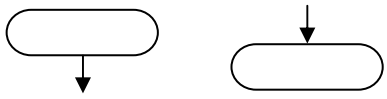
Алгоритм циклической структуры – алгоритм, содержащий многократно выполняемые участки вычислительного процесса, называемые циклами. Если алгоритм содержит цикл, внутри которого размещен один или несколько других циклов, то такой алгоритм называется алгоритмом со структурой вложенных циклов.

Существует много способов записи алгоритмов, отличающихся друг от друга наглядностью, компактностью, степенью формализации и другими показателями. Наибольшее распространение получил графический способ (блок-схем).

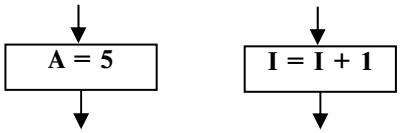
Схема алгоритма представляет собой последовательность блоков, предписывающих выполнение определенных действий, и связи между ними. Она может выполняться с разной степенью детализации.

Запись блоков

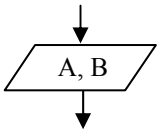
1 Начало и конец алгоритма (программы):



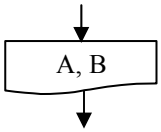
2 Блок присваивания:



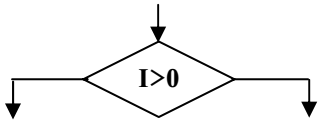
3 Ввод данных с клавиатуры:



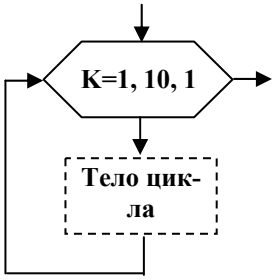
4 Вывод информации на экран (на печать):



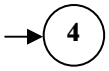
5 Блок ветвления (проверки условия):



6 Блок цикла с параметром:



7 Нумерация блоков, значок перехода на блок с указанным номером:



Пример

Напишите блок-схему алгоритма программы, которая запрашивает с клавиатуры целое число N и если это число больше 10, то вычисляет и выводит на экран произведение всех целых чисел от 1 до N, иначе выводит на экран значение N.

Алгоритм решения задачи:

1 Ввод исходных данных (N).

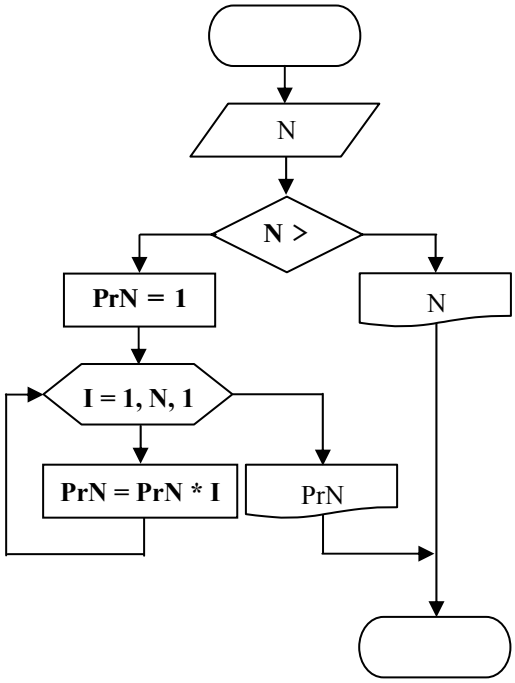
2 Проверка условия N > 10.

3 В случае выполнения условия вычисляется произведение всех целых чисел от 1 до N.

4 Вывод на экран полученного произведения и завершение выполнения программы.

5 В противном случае вывод на экран значения N и завершения выполнения программы.

Блок-схема:



ЗАДАНИЯ

НАПИШИТЕ БЛОК-СХЕМУ АЛГОРИТМА ПРОГРАММЫ, КОТОРАЯ:

1) классифицирует компьютерную сеть. Программа запрашивает у пользователя число компьютеров в сети и в зависимости от введенного количества выводит класс сети (если число ЭВМ меньше 256 – то это сеть класса С, от 256 до 65535 – сеть класса В, свыше 65535 – сеть класса А);

2) запрашивает у пользователя номер одного из весенних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц весенним;

3) выводит на экран приглашение: «Который час?», вводит с клавиатуры число X, имеющее смысл времени суток, и печатает слова «Доброе утро», «Добрый день», «Добрый вечер» или «Доброй ночи» в зависимости от введенного времени. Программа должна реагировать на ввод неправильного времени: меньше 0 или больше 24;

4) запрашивает у пользователя размер хищений (р.), определяет и выводит на экран масштаб в соответствии с принятой классификацией (например, если размер хищений меньше 1000 р. – «мелкий», от 1000 до 10 000 – «крупный», свыше 100 000 – «особо крупный»);

5) запрашивает произвольное число N, вычисляет сумму всех целых чисел от 1 до N. Если полученная сумма больше 10, то выводит на экран ее значение, иначе выводит на экран сообщение «Сумма меньше 10»;

6) выводит на экран приглашение: «Введите месяц», вводит с клавиатуры число X, имеющее смысл месяца, и печатает слова «Зима», «Весна», «Лето» или «Осень» в зависимости от введенного месяца. Программа должна реагировать на ввод неправильного месяца: меньше 1 или больше 12;

7) запрашивает с клавиатуры два произвольных числа X и Y. Если X больше Y, то вычисляет и выводит на экран сумму всех целых чисел от 1 до X, иначе вычисляет и выводит на экран произведение всех целых чисел от X до Y;

8) запрашивает у пользователя номер одного из летних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц летним;

9) вводит с клавиатуры 10 чисел и выводит на экран максимальное и минимальное из них;

10) запрашивает с клавиатуры два произвольных числа X и Y, вычисляет их разность. Если разность больше 10, то вычисляет и выводит на экран сумму всех целых чисел от Y до X, иначе выводит на экран значение разности;

11) выводит на экран приглашение: «Введите день недели», вводит с клавиатуры число X, имеющее смысл дня недели, и печатает слова «Рабочий день», «Короткий день», «Выходной» в зависимости от введенного дня. Программа должна реагировать на ввод неправильного дня недели: меньше 1 или больше 7;

12) запрашивает произвольное число N, вычисляет произведение всех целых чисел от 1 до N. Если полученное произведение больше 50, то выводит на экран ее значение, иначе выводит на экран сообщение «Произведение меньше 50»;

13) запрашивает у пользователя номер одного из осенних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц осенним;

14) запрашивает с клавиатуры два целых числа, их сумму и произведение и выводит на экран сообщение о правильности сделанных пользователем вычислений;

15) запрашивает с клавиатуры два произвольных числа X и Y. Если X больше Y, то вычисляет и выводит на экран произведение всех целых чисел от 1 до X, иначе вычисляет и выводит на экран сумму всех целых чисел от 1 до Y;

16) запрашивает с клавиатуры два произвольных числа X и Y, вычисляет их разность. Если разность больше 10, то вычисляет и выводит на экран произведение всех целых чисел от Y до X, иначе выводит на экран значение разности;

17) запрашивает с клавиатуры два целых числа, их разность и частное (результат деления) и выводит на экран сообщение о правильности сделанных пользователем вычислений;

18) запрашивает у пользователя номер одного из зимних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц зимним;

19) запрашивает с клавиатуры число X. Если X меньше 10, то вычисляет и выводит на экран квадрат числа X, а если больше или равно, то вводилось новое число Y, а затем вычисляет и выводит на экран значение суммы X и Y;

20) запрашивает с клавиатуры два произвольных числа X и Y. Если разность X и Y больше 0, то вычисляет и выводит на экран сумму всех целых чисел от Y до X, иначе вычисляет и выводит на экран разность всех целых чисел от Y до X.

Лабораторная работа 2

ЗНАКОМСТВО СО СРЕДОЙ Borland Pascal

Цель работы: научиться работать со средой Borland Pascal, познакомиться со структурой программы в Pascal и действиями, выполняемыми над данными.

Основные понятия

Процессор компьютера может выполнять только те команды, которые ему знакомы. Эти команды представляются двоичным кодом. Однако создать файл с такими командами вручную очень сложно. Поэтому для облегчения программирования разработаны специальные языки – ***алгоритмические языки*** или ***языки программирования***. Программист с помощью текстового редактора создает файл с текстом программы, записанной на алгоритмическом языке. Но чтобы получить готовую для выполнения программу, этот текст нужно преобразовать в коды команд процессора. Для такого преобразования служат специальные программы-переводчики, их называют ***компиляторы*** и ***трансляторы*** (от англ. слов *translate* – переводить, *compile* – составлять).

С момента появления первых ЭВМ было придумано много разных алгоритмических языков: Basic, Pascal, C, Fortran, Prolog, Assembler и т.д. При этом существуют и развиваются различные версии компиляторов для этих языков (Turbo Pascal, Borland Pascal версий 5.0, 6.0, 7.0, и т.д.).

Ознакомимся с языками программирования на примере языка Pascal.

Структура программы. Программа на языке Pascal состоит из заголовка программы и двух частей: раздела описаний и раздела операторов.

PROGRAM *Name*; {*Заголовок программы*}
  {*Раздел описаний*}
BEGIN
  {*Раздел операторов*}
END.

В Pascal игнорируется различие в высоте букв (заглавные или строчные), т.е. *Name*, *NAME*, *name* одно и то же.

Слово PROGRAM зарезервировано в Pascal, т.е. не может использоваться ни в каких иных целях, кроме как для объявления имени программы. Заголовок программы необязателен и игнорируется компьютером.

Первая строка заканчивается особым разделителем – точкой с запятой. Этот разделитель в языке Pascal отмечает конец оператора или описания. Использование особого разделителя позволяет располагать несколько операторов в одной строке.

Зарезервированное слово BEGIN сигнализирует компилятору о начале другой части программы – раздела операторов. Завершает всю программу зарезервированное слово END. Точка оповещает компилятор о конце текста программы.

Обязательной частью являются лишь тело программы (раздел операторов). Заголовок программы является хотя и необязательным, но желательным элементом.

Раздел описаний состоит из следующих подразделов:

- описания меток (LABEL);
- описания внешних модулей (USES);
- описания типов (TYPE);
- описания констант (CONST);
- описания переменных (VAR);
- описания функций (FUNCTION);
- описания процедур (PROCEDURE).

Порядок размещения подразделов произвольный, можно создавать несколько одинаковых подразделов

Подраздел описания меток. Метка – точка перехода. Используется в операторе безусловного перехода. Данный подраздел начинается со слова LABEL, за которым следует список меток:

LABEL 1,77, 190;

В программе в качестве меток могут использоваться целые числа без знака. Сама метка ставится в телепрограммы перед оператором и отделяется от него двоеточием.

Подраздел описания внешних модулей. Внешние модули – это наборы констант, типов данных, переменных, процедур и функций, которые вы можете использовать в своей программе. На каждый модуль есть описание его содержимого. Подключение модуля к вашей программе осуществляется строчкой:

USES *Модуль*,

где USES – зарезервированное слово; *Модуль* – имя подключаемого модуля.

Подраздел описания типов. Среди типов данных различают стандартные (предопределенные разработчиками языка) и пользовательские (определяемые программистом в своей программе). Основные (стандартные) типы данных, используемые в языке Pascal следующие:

- целые числа;
- вещественные числа;
- логический тип;
- символьный тип;
- строковый тип.

Программист может описать свой тип, на основе этих базовых в разделе описания типов, который начинается словом TYPE. Затем для каждого типа следует конструкция вида:

Имя типа = (*идентификатор1*, *идентификатор2*, … , *идентификаторN*).

Рассмотрим простые типы данных, каждый из которых определяет упорядоченное множество значений:

Целые типы	**Shortint** (–128 ... 127, 1 байт) **Integer** (–32767 ... 32768, 2 байта) **Longint** (–2147483648 ... 2147483647, 4 байта) **Byte** (0 ... 255, 1 байт) **Word** (0 ... 65535, 2 байта)
Вещественные типы	**Real** (занимает 6 байт, диапазон от 2.9E–39 до 1.7E+38 по модулю, точность 11 – 12 значащих цифр) **Single** (занимает 4 байта, диапазон от 1.5E–45 до 3.4E+38 по модулю, точность 7 – 8 значащих цифр) **Double** (занимает 8 байт, диапазон от 5.0E–324 до 1.7E+308 по модулю, точность 15 – 16 значащих цифр) **Extended** (занимает 10 байт, диапазон от 3.4E–4932 до 1.1E+4932 по модулю, точ-

	ность19 – 20 значащих цифр) **Comp** (занимает 8 байт, диапазон от –9.2E–18 до 9.2E+18, хранятся точно, поскольку это целые числа)
Логический тип (Boolean)	Переменные логического типа занимают в памяти один байт и могут принимать одно из двух значений *False* – ложное или *True* – истинное
Символьный тип (Char)	Символьный тип позволяет работать с символами, которые записываются двумя способами: в одинарных кавычках или по их коду, например 'a', 'B', '*' или, что, то же самое, #97, #130, #42. В отличие от текста программы на Паскале, символы, соответствующие строчным и заглавным буквам различаются
Строка (String)	Строка символов, занимает MAX+1 байт, где MAX – максимальное количество символов в строке

Вещественные числа хранятся неточно. Каждый из имеющихся вещественных типов гарантирует правильное хранение только определенного количества значащих цифр, их называют верными цифрами.

Подраздел описания констант. Константами в языке считаются такие объекты программы, которые не могут изменять своего значения.
В языке Pascal константы могут иметь собственное имя (идентификатор). Константы описываются следующим образом:

CONST *Имя* = *Значение*,

где CONST – зарезервированное слово; *Имя* – имя константы; *Значение* – значение константы.

В Pascal есть зарезервированные константы. Например, число π, идентификатор pi (pi = 3.14159265).

Подраздел описания переменных. Переменная – объект программы, который может изменять свое значение в процессе выполнения (счета). Переменная представляет собой некоторый адрес ячейки памяти, по которому будет размещен тот или иной элемент данных, с которым оперирует компьютерная программа. В процессе решения переменная может изменять свои значения, но в каждый момент времени в памяти ЭВМ хранится только одно «текущее» значение. Объем памяти, отводимый под переменную, определяется типом данных (Real, Integer и т.д.). Описание переменных производится следующим образом:

VAR *Имя* : *тип*,

где Var – зарезервированное слово; *Имя* – идентификатор переменной; *тип* – тип переменной.

В настоящее время в программировании принято записывать имена переменных с использованием так называемой венгерской нотации. Венгерская нотация основывается на том, что имена переменных и функций записываются полными словами или словосочетаниями или их сокращениями, но так, чтобы по имени можно было понять назначение переменной или действие, выполняемое функцией.

Процедуры (подпрограммы) и функции представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные именем.

Основные правила выбора идентификаторов (имен)

1 Идентификатор может состоять из букв латинского алфавита (a…z), цифр (0…9), знака подчеркивания;

2 Идентификатор не может содержать специальные знаки:

- знаки пунктуации . () [] .. (разделение границ диапазона) : ; ‘ (апостроф) = $ # (признак кода числа);
- знаки операций + – * / > < @.

3 Идентификатор не может начинаться с цифры.

4 Идентификатор не может совпадать ни с одним из зарезервированных слов.

5 Длина идентификатора может быть произвольной, но значащими считаются первые 63 символа.

6 Заглавные и строчные буквы в идентификаторах считаются одинаковыми.

Действия, выполняемые над данными

Арифметические действия	
Арифметические действия выполняются над данными вещественного и целого типов	+ – сложение – – вычитание * – умножение / – деление вещественное DIV – деление целочисленное MOD – вычисление остатка от целочисленного деления
Операции отношения (сравнения)	
Результат любой операции соотношения имеет логический тип, т.е. либо *True* – истина, либо *False* – ложь	= – равно <> – не равно < – меньше > – больше <= – меньше или равно >= – больше или равно
Логические операции	
Логические операции над данными логического типа дают результат того же типа, над данными целого типа – результат целого типа	NOT – логическое НЕ (превращает *True* в *False*, а *False* в *True*) AND – логическое И (логическое умножение) OR – логическое ИЛИ (логическое сложение) XOR – исключающее или (*True* если операнды разные)

Принцип действия этих операций можно проиллюстрировать такой таблицей:

A	NOT A	B	NOT B	A AND B	A OR B	A XOR B
False	*True*	*False*	*True*	*False*	*False*	*False*
False	*True*	*True*	*False*	*False*	*True*	*True*
True	*False*	*False*	*True*	*False*	*True*	*True*
True	*False*	*True*	*False*	*True*	*True*	*False*

Из идентификаторов, констант, знаков арифметических, логических операций и операций сравнения составляются выражения.

Правила составления выражений

1 Два символа арифметических операций не должны стоять рядом; исключение составляет знак «– » перед отрицательной константой.

2 Нельзя опускать знак умножения.

3 Круглые скобки определяют очередность выполнения операций.

4 Число левых и правых скобок должно быть одинаково.

5 При отсутствии скобок вычисление выражения выполняется согласно приоритету операций:

- унарные операции: NOT, + (увеличение на 1), – (уменьшение на 1);
- операции типа умножения: *, /, DIV, MOD, AND;
- операции типа сложения: +, –, OR;
- операции отношения =, <, >, <=, >=, <>.

Система программирования Borland Pascal

Система программирования Borland Pascal представляет собой единство компилятора с языка программирования Pascal и некоторой инструментальной программной оболочки, способствующей повышению эффективности создания программ.

Назначение среды: написание и редактирование текстов программ, загрузка с диска и сохранение на диске программ на языке Pascal, компиляция, запуск и отладка программ, а также многое другое.

«Горячие» клавиши

F1	Помощь
F2	Запись текста программы в файл на диске
F3	Загрузка текста программы с диска
F10	Вход в главное меню
Alt + X	Выход из системы

ТЕКСТОВЫЙ РЕДАКТОР. *ТЕКСТОВЫЙ РЕДАКТОР СРЕДЫ БОРЛАНД ПАСКАЛЬ ПРЕДОСТАВЛЯЕТ ПОЛЬЗОВАТЕЛЮ УДОБНЫЕ СРЕДСТВА СОЗДАНИЯ И РЕДАКТИРОВАНИЯ ТЕКСТОВ ПРОГРАММ.*

<table>
<tr><th colspan="2">Клавиши перемещения курсора</th><th colspan="2">Клавиши удаления и замены</th></tr>
<tr><td>→</td><td>На один символ вправо</td><td rowspan="2">⇐ или BackSpace</td><td rowspan="2">Удаление символа перед курсором</td></tr>
<tr><td>←</td><td>На один символ влево</td></tr>
<tr><td>↑</td><td>На одну строку вверх</td><td>Delete</td><td>Удаление символа над курсором</td></tr>
<tr><td>↓</td><td>На одну строку вниз</td><td>Ctrl + Y</td><td>Удаление текущей строки</td></tr>
<tr><td>Home</td><td>В начало строки</td><td>Insert</td><td>Вкл./выкл. режима замены символа</td></tr>
<tr><td>End</td><td>На конец строки</td><td colspan="2" rowspan="2">Работа с блоками</td></tr>
<tr><td>Pg Up</td><td>На страницу вверх</td></tr>
<tr><td>Pg Dn</td><td>На страницу вниз</td><td>Shift + стрелка</td><td>Выделение блока</td></tr>
</table>

Ctrl + Home	В начало страницы	Ctrl + Insert	Забрать выделенный блок в буфер
Ctrl + End	На конец страницы	Shift + Insert	Вставить блок из буфера
Ctrl + Pg Up	В начало текста	Ctrl + K, Ctrl + Y	Удалить выделенный блок
Ctrl + Pg Dn	В конец текста		

Компиляция и отладка программ

F9	Компиляция программы
Ctrl + F9	Компиляция программы с последующим запуском
Alt + F5	Переключиться в экран выполнения программы
F7	Шаг отладки с заходом в подпрограммы
F8	Шаг отладки без захода в подпрограмму
F4	Выполнение программы до текущего местоположения курсора
Ctrl + F4	Просмотреть содержимое переменной
Ctrl + F2	Выход из режима отладки

ПРИМЕР. *РАСЧЕТ КОРНЕЙ КВАДРАТНОГО УРАВНЕНИЯ.*

Общий вид: $ax^2 + bx + c = 0$.

Исходные данные: a, b, c.

Найти: корни x_1 и x_2.

Алгоритм:
1 Вводим a, b, c.
2 Вычисляем дискриминант $D = b^2 – 4ac$.
3 Проверяем условие $D < 0$, если «да», то выводим на экран сообщение «Действительных корней нет» и завершаем выполнение программы.
4 В противном случае, вычисляем корни уравнения x_1 и x_2.
5 Выводим на экран x_1 и x_2.

Блок-схема: (Составить самим).

Программа:

```
PROGRAM QuadrEq;
VAR a, b, c, x1, x2, d : real;
                    BEGIN
        WRITELN('Введите коэффициенты уравнения');
        READLN(a, b, c);
        d:=b*b–4*a*c;
        IF d<0 THEN WRITELN('Действительных корней нет')
        ELSE
                BEGIN
                        x1:=(-b+sqrt(d))/2*a;
                        x2:=(-b-sqrt(d))/2*a;
                        WRITELN ('x1=', x1, ' x2=', x2);
```

```
                END;
END.
```

Лабораторная работа 3

ОПЕРАТОР УСЛОВНОГО ПЕРЕХОДА

Цель работы: познакомиться с операторами ввода/вывода, присвоения, безусловного перехода, составными и пустыми операторами языка Pascal. Научиться применять оператор условного перехода.

Основные понятия

Операторы ввода-вывода. Для ввода данных в Паскале используются операторы READ и READLN. При вводе данных с клавиатуры действия, выполняемые этими двумя операторами, будут практически одинаковыми. Разница только в том, что после ввода READLN переводит курсор на новую строку. Синтаксис оператора READLN:

READLN (a1, a2, …),

где a1, a2 … – список переменных, в которые осуществляется ввод данных.

Оператор READLN работает следующим образом: программа останавливается в ожидании ввода, пользователь вводит данные в соответствии со списком переменных, перечисленных в скобках. Данные, вводимые с клавиатуры, друг от друга отделяются пробелами или вводятся через *Enter*. Ввод заканчивается нажатием клавиши *Enter*. Далее программа распределяет данные по переменным a1, a2, …

Если в программе встретится оператор READLN без списка переменных, то программа будет ожидать нажатие клавиши *Enter* (т.е. ввода пустой строки). Обычно такой оператор ставится в конце программы, чтобы можно было сначала оценить результаты, а потом нажатием *Enter* завершить программу, т.е. для организации паузы.

Вывод данных на экран осуществляется посредством операторов WRITE и WRITELN. Отличие в работе оператора WRITE и WRITELN, заключается в том, что оператор WRITELN после вывода значений всех переменных и констант из списка осуществляет перевод курсора на новую строку экрана. Синтаксис оператора WRITELN:

WRITELN(a1, a2, …),

где a1, a2, … – список вывода, в котором кроме имен переменных можно писать строковые константы (последовательность символов в апострофах) и даже выражения (выводятся их результаты).

Все операторы в Pascal отделяются друг от друга символом «;».

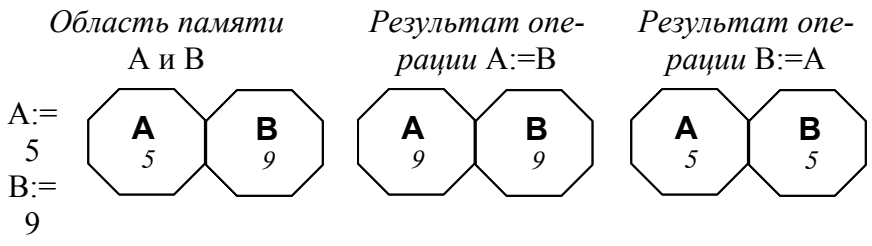
Оператор присваивания. Синтаксис оператора присваивания:

Имя переменной := *Выражение*.

Переменная (левая часть) и выражение (правая часть) должны быть одного типа.

Данное выражение следует читать: «К присвоить А». Понимается это так, что значение, хранимое в области памяти с именем А, помещается в область памяти с именем К.

Лучше всего продемонстрировать действие операции присвоения на примере. Допустим, значения 5 и 9 присваиваются переменным А и К.

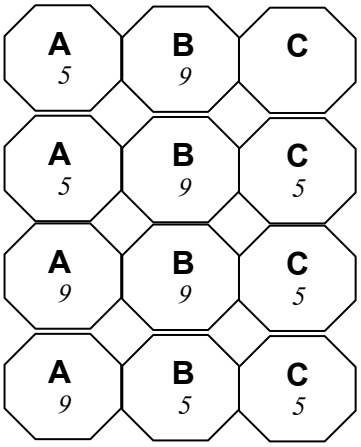


Следует всегда помнить, что в программировании переменные – это не только абстрактные имена, но и конкретные области памяти, которые хранят значения.

Допустим, необходимо сделать так, чтобы A = 9, B = 5. Если на писать A:=B, то получится результат:

A:=B

При этом значение 5 будет потеряно безвозвратно. Поэтому нужна третья переменная C, в которой временно будет храниться значение переменной A.



Дополнительная переменная C

Сохраняем значение A: B:=A

Значение B помещаем в A: A:=B

Первоначальное значение A перемещаем в B: B:=C

Составной оператор и пустой оператор. Составной оператор – это последовательность произвольных операторов программы, заключенная в операторные скобки – зарезервированные слова BEGIN … END. Составные операторы – важный инструмент Pascal, дающий возможность писать программы по современной технологии структурного программирования (без операторов перехода GOTO).

Pascal допускает произвольную глубину вложенности составных операторов:

```
BEGIN
   ……
   BEGIN
         ……
         BEGIN
               ……
         END;
         ……
   END;
   ……
END.
```

Поскольку BEGIN и END представляют собой структурные скобки, то после BEGIN и перед END ставить знак «;» не обязательно.

В программе может применяться пустой оператор, не выполняющий никакого действия. Например – ; ;.

Оператор условного перехода. Оператор условного перехода (условный оператор) позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом условный оператор – это средство ветвления вычислительного процесса.

Структура условного оператора имеет вид:

IF *условие* **THEN** *оператор1* **ELSE** *оператор2*,

где IF, THEN, ELSE – зарезервированные слова («если», «то», «иначе»); *оператор1*, *оператор2* – любые операторы языка Паскаль (в том числе и составные).

Условный оператор работает по следующему алгоритму. Вначале вычисляется *условие*, если результат *True* (истина), то выполняется *оператор1*, а *оператор2* пропускается; если результат *False* (ложь), то, наоборот, *оператор1* пропускается, а выполняется *оператор2*.

Оператор IF может быть неполным, т.е. часть «ELSE *оператор2*» может быть опущена. Тогда при значении *True* условного выражения выполняется *оператор1*, в противном случае он пропускается.

Если *оператор1* и *оператор2* – составные, то условный оператор будет иметь вид:

```
IF условие THEN
   BEGIN
   ……
   END
ELSE
   BEGIN
   ……
   END.
```

Пример. Напишите программу, определяющую наименьшее значение из двух чисел, введенных с клавиатуры.

```
PROGRAM Minimum;
VAR a, b, min : real;
BEGIN
   WRITELN('Введите два числа');
   READLN(a, b);
   IF a<b Then min:=a ELSE min:=b;
   WRITELN(min);
END.
```

Задания

Используя оператор условного перехода, напишите программу, которая:

1) классифицирует компьютерную сеть. Программа запрашивает у пользователя число компьютеров в сети и в зависимости от введенного количества выводит класс сети (если число ЭВМ меньше 256 – то это сеть класса С, от 256 до 65535 – сеть класса В, свыше 65535 – сеть класса А);
2) запрашивает у пользователя номер одного из весенних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц весенним;
3) выводит на экран приглашение: «Который час?», вводит с клавиатуры число X, имеющее смысл времени суток, и печатает слова «Доброе утро», «Добрый вечер», «Добрый день» в зависимости от введенного времени. Программа должна реагировать на ввод неправильного времени: меньше 0 или больше 24;
4) запрашивает с клавиатуры у пользователя размер хищений (р.), определяет и выводит на экран масштаб в соответствии с принятой классификацией (например, если размер хищений меньше 100 р. – «мелкий», от 100 до 1000 – «крупный», свыше 1000 – «особо крупный»);
5) выводит на экран приглашение: «Введите месяц», вводит с клавиатуры число X, имеющее смысл месяца, и печатает слова «Зима», «Весна», «Лето», «Осень» в зависимости от введенного месяца. Программа должна реагировать на ввод неправильного месяца: меньше 1 или больше 12;
6) запрашивает у пользователя номер одного из летних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц летним;
7) вводит с клавиатуры три числа и выводит на экран максимальное из них;
8) выводит на экран приглашение: «Введите день недели», вводит с клавиатуры число X, имеющее смысл дня недели, и печатает слова «Рабочий день», «Короткий день», «Выходной» в зависимости от

введенного дня. Программа должна реагировать на ввод неправильного дня недели: меньше 1 или больше 7;

9) запрашивает у пользователя номер одного из осенних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц осенним;

10) запрашивает с клавиатуры два целых числа, их сумму и произведение и выводит на экран сообщение о правильности сделанных пользователем вычислений;

11) вводит с клавиатуры три числа и выводит на экран минимальное из них;

12) запрашивает с клавиатуры два целых числа, их разность и частное (результат деления) и выводит на экран сообщение о правильности сделанных пользователем вычислений;

13) запрашивает у пользователя номер одного из зимних месяцев, и выводит количество дней в этом месяце. Программа должна проверять, является ли введенный месяц зимним;

14) вводит с клавиатуры число X. Если X меньше 10, то вычисляет и выводит на экран квадрат числа X, а если больше или равно, то вводит новое число Y, а затем вычисляет и выводит на экран значение суммы X и Y;

15) запрашивает с клавиатуры три произвольных числа и выводит на экран сообщение о том, сколько из них отрицательных, положительных и нулевых.

Лабораторная работа 4

ОДНОМЕРНЫЕ МАССИВЫ

Цель работы: научиться работать с одномерными массивами и операторами цикла WHILE...DO и REPEAT...UNTIL.

Основные понятия

Массив – это упорядоченный набор переменных, которым присвоено одно имя. К необходимости применения массивов мы приходим каждый раз, когда требуется связать и использовать целый ряд родственных
величин.

В одномерном массиве элементы располагаются в последовательных ячейках памяти, т.е. массив занимает непрерывную область памяти. Каждый элемент массива имеет свой номер, являющийся целым числом и называющийся индексом элемента. Иногда массив так и называют *переменные с индексами*. Для использования в программе массива требуется предварительно описать его в разделе описания переменных:

Имя : ARRAY [*Мин_индекс*…*Макс_индекс*] OF *Тип*,

где *Имя* – имя массива, ARRAY; OF – зарезервированные для описания массивов слова; *Мин_индекс*, *Макс_индекс* – минимальное и максимальное значения индекса массива; *Тип* – тип переменных массива. Минимальное и максимальное значения индекса массива определяют его размерность.

Доступ к элементу одномерного массива в программе осуществляется путем указания имени массива и после него в квадратных скобках номера этого элемента в массиве.

Для ввода данных в массив и вывода из массива используются циклы. Цикл – многократно выполняемый участок вычислительного процесса.

Паскаль позволяет использовать три различных оператора для организации циклов:

- оператор цикла с параметром;
- оператор цикла с предварительной проверкой условия;
- оператор цикла с последующей проверкой условия.

В данной работе познакомимся с операторами цикла с предварительной и последующей проверкой условий.

Оператор цикла с предварительной проверкой условия

WHILE *условие* **DO** *оператор*,

где WHILE, DO – зарезервированные слова [пока (выполняется условие), делать]; *условие* – выражение логического типа; *оператор* – произвольный оператор языка Паскаль.

Оператор WHILE работает следующим образом: если выражение *условие* имеет значение *True*, то выполняется *оператор*, после чего вычисление выражения *условие* и его проверка повторяются. Если *условие* имеет значение *False*, оператор WHILE прекращает свою работу. Поскольку значение логического выражения проверяется в начале каждой итерации, то тело цикла может не выполниться ни разу. Таким образом, в этом цикле логическое выражение – это условие продолжения работы в цикле.

Оператор цикла с последующей проверкой условия

REPEAT *тело_цикла* **UNTIL** *условие*,

где REPEAT, UNTIL – зарезервированные слова (повторять до тех пор, пока не будет выполнено *условие*); *тело_цикла* – произвольная последовательность операторов языка Паскаль; *условие* – выражение логического типа.

Оператор REPEAT работает следующим образом: выполняются операторы, входящие в *тело*_цикла, после этого вычисляется значение логического выражения условие: если False, то выполнение операторов *тело_цикла* повторяется, в противном случае оператор REPEAT завершает свою работу. Тело цикла обязательно выполняется хотя бы один раз. Таким образом, в этом цикле логическое выражение – это условие выхода из цикла.

Пример. Программа поиска наибольшего значения в одномерном массиве, размерностью 10.

```
PROGRAM Maximum;
VAR i, n : integer;
Massiv : ARRAY[1..10] OF Real;
  Begin
        i:=1;
        WHILE i<=10 DO
        BEGIN
        WRITE('Введите ', i, '-й элемент: ');
        READLN(Massiv[i]);
        i:=i+1;
        END;
        n:=1;
        i:=2;
        REPEAT
        IF Massiv[i]>Massiv[n] THEN n:=i;
        i:=i+1;
        UNTIL i>10;
        WRITELN('Максимальный элемент ', n, ' = ', Massiv[n]:7:4);
  END.
```

Задания

Напишите программу, которая ввод данных в одномерный целочисленный массив осуществляет посредством цикла WHILE...DO, а вывод – REPEAT...UNTIL (в скобках указана размерность массива):

1) заменить все отрицательные элементы массива нулями (12);
2) увеличить элементы массива с четными индексами на «1» (11);
3) сделать все положительные элементы массива отрицательными (10);
4) заменить каждый отрицательный элемент произведением всех ненулевых элементов массива (8);
5) заменить первый элемент массива максимальным (11);
6) увеличить каждый ненулевой элемент массива на «3» (12);
7) заменить все элементы массива, которые больше 10, на 100 (12);
8) увеличить все положительные элементы массива на единицу (12);
9) заменить все нулевые элементы массива на единицу (11);
10) заменить минимальный элемент массива на единицу (12);
11) увеличить элементы массива с нечетными индексами на «2» (10);
12) увеличить каждый третий элемент массива в 2 раза (12);
13) заменить последний элемент массива минимальным (10);
14) заменить все элементы массива, которые делятся без остатка на «3», на единицы (9);
15) увеличить все ненулевые элементы массива в 3 раза (11);
16) заменить каждый положительный элемент суммой всех элементов массива (8);
17) уменьшить каждый ненулевой элемент массива на 2 (11);
18) заменить каждый второй элемент массива на «1» (15);
19) увеличить все положительные элементы массива, которые меньше 10, на «10» (10);
20) заменить все элементы массива, которые меньше среднего значения, на «3» (12).

Лабораторная работа 5

ОБРАБОТКА ДВУХМЕРНЫХ МАССИВОВ

Цель работы: научиться работать с двухмерными массивами.

Основные понятия

Массив – это упорядоченный набор переменных, которым присвоено одно имя. К необходимости применения массивов мы приходим каждый раз, когда требуется связать и использовать целый ряд родственных
величин.

В одномерном массиве элемент определяется при помощи одного индекса, поэтому такой массив можно представить в виде строки. Доступ к элементам двухмерного массива осуществляется посредством двух индексов, что позволяет представить двухмерный массив в виде таблицы, в которой первый индекс определяет номер строки, а второй – номер столбца. На пересечении столбца и строки находится определенный элемент.

A	1	2	3	4	5
1					
2					
3					
4					
5					

Данный двухмерный массив в программе описывается следующим образом:

VAR A : ARRAY [1..5, 1..5] OF REAL.

Элемент, расположенный в 5-ой строке и первом столбце имеет имя A[5, 1].

Оператор цикла с параметром. Оператор цикла FOR организует выполнение одного оператора заранее определенное число раз. Синтаксис оператора:

FOR *пар_цик* **:=** *нач_знач* **TO** *кон_знач* **DO** *оператор*,

где FOR, TO, DO – зарезервированные слова («для», «до», «выполнить»); *пар_цик* – параметр цикла – переменная типа *Integer*; *нач_знач* – начальное значение параметра цикла; *кон_знач* – конечное значение параметра цикла; *оператор* – произвольный оператор языка Паскаль.

На первом шаге цикла параметр принимает значение *нач_знач*. В этот же момент происходит вычисление *кон_знач* – значения параметра на последнем шаге цикла. После каждого выполнения тела цикла, если параметр цикла не равен *кон_знач*, происходит изменение параметра на следующее большее или меньшее значение в зависимости от формы оператора FOR.

В случае *нач_знач* > *кон_знач* в первой форме оператора или *нач_знач* < *кон_знач* во второй его форме ошибки не происходит, но цикл не выполняется ни разу. После завершения работы цикла значение параметра остается равным *кон_знач*.

На месте *нач_знач* и *кон_знач* могут находиться выражения целого типа (n+2, 2*k+n и т.д.), а *оператор* может быть составным оператором.

Для того чтобы установить шаг наращивания параметра цикла –1 вместо служебного слова TO пишется DOWNTO.

Для работы с двухмерным массивом в программе используется алгоритм с вложенными циклами.

Пример. Программа заполнения двухмерного массива 3 × 5 и поиска в нем наименьшего элемента.

```
PROGRAM Minimum;
      VAR i, j : Integer;
      M : ARRAY [1..3, 1..5] OF Real;
      min : Real;
  BEGIN
        FOR i:=1 TO 3 DO
        FOR j:=1 TO 5 DO
        BEGIN
        WRITE('M[',i,',',j,']=');
        READLN(M[i,j]);
        END;
        min:=M[1,1];
        FOR i:=1 TO 3 DO
        FOR j:=1 TO 5 DO
        IF M[i,j]<min THEN min:=M[i,j];
        WRITELN('Минимальный элемент=', min:7:4);
  END.
```

ЗАДАНИЯ

Напишите программу, в которой нужно:

1) вывести столбец и строку двухмерного массива 4 × 5, на пересечении которых расположен максимальный элемент;
2) определить номера строк двухмерного массива 5 × 3, содержащих только положительные элементы;
3) для каждого столбца двухмерного массива 3 × 5 рассчитать произведение ненулевых элементов;
4) подсчитать произведение неотрицательных элементов в двухмерном массиве 6 × 3;

5) в двухмерном массиве 6 × 3 для каждой нечетной строки определить произведение положительных элементов;

6) подсчитать произведение отрицательных элементов в двухмерном массиве 4 × 5;

7) для каждой строки двухмерного массива 5 × 4 рассчитать среднее значение;

8) определить количество элементов в двухмерном массиве 3 × 6, которые больше 2, но меньше 6;

9) в двухмерном массиве 3 × 6 для каждого четного столбца определить сумму элементов;

10) вывести столбец и строку двухмерного массива 5 × 4, на пересечении которых расположен минимальный элемент;

11) определить номера столбцов двухмерного массива 3 × 5, содержащих только отрицательные элементы;

12) для каждой строки двухмерного массива 5 × 4 определить максимальное значение;

13) определить количество положительных элементов в двухмерном массиве 5 × 4;

14) в двухмерном массиве 5 × 4 для каждой нечетной строки определить сумму элементов;

15) подсчитать произведение положительных элементов в двухмерном массиве 3 × 6;

16) определить количество элементов в двухмерном целочисленном массиве 6 × 3, делящихся на «3» без остатка;

17) для каждой строки двухмерного массива 6 × 4 посчитать количество положительных элементов;

18) определить номера строк двухмерного массива 5 × 4, сумма элементов которых больше 50;

19) определить номера столбца двухмерного массива 4 × 3, произведение элементов которых меньше 15;

20) в двухмерном массиве 5 × 3 определить количество элементов, которые больше 10.