Государственное бюджетное профессиональное
образовательное учреждение
«Нижегородский автомеханический техникум»

Основы программирования на алгоритмическом языке Паскаль

учебное пособие

по дисциплине «Основы программирования»

для студентов дневного отделения

специальности 09.02.03

Программирование в компьютерных системах

Нижний Новгород

2020

Содержание

Введение

Методическая разработка «Основы программирования на алгоритмическом языке Паскаль» предназначена для студентов дневного отделения специальности 09.02.03. Программирование в компьютерных системах в качестве учебного пособия при изучении раздела «Алгоритмический язык программирования Паскаль» дисциплины «Основы программирования».

В учебном пособии нашли отражение такие вопросы, как «Элементы языка Паскаль», «Операторы языка Паскаль», «Структурированные типы данных», «Подпрограммы в Паскале», что дает возможность получить теоретические знания по данным темам, приведены примеры программ, задачи для самостоятельного решения. Используя данное пособие, студенты могут подготовить домашнее задание, самостоятельно изучить пропущенный материал или поработать дополнительно. Представленные темы соответствуют рабочей программе дисциплины «Основы программирования».

В результате изучения данного пособия студент должен:

уметь:

- реализовывать построенные алгоритмы в виде программ на языке программирования Паскаль;

знать:

- типы данных;
- базовые конструкции языка программирования Паскаль.

Методическая разработка может быть использована в качестве раздаточного материала на занятиях, и помочь высвободить время на уроках для практических занятий по темам.

Основная часть

Алгоритмический язык высокого уровня Паскаль был создан в 1968-1971 гг. Никлаусом Виртом (швейцарский ученый, специалист в области информатики, один из известнейших теоретиков в области разработки языков программирования, профессор информатики) в швейцарском федеральном институте в Цюрихе. Родоначальником языка Паскаль стал язык программирования АЛГОЛ. Свое название язык программирования получил в честь великого французского математика и физика Блеза Паскаля, который в 1642 году изобрел счетную машину для выполнения арифметических операций, так называемое «паскалево колесо».

Созданный специально для обучения навыкам структурного программирования язык оказался чрезвычайно удачным и сразу же привлек внимание специалистов. В короткое время язык Паскаль приобрел широкую популярность во всем мире. К основным достоинствам языка следует отнести:

1. Легкость реализации на ПК.

2. Возможность достаточно полного контроля правильности программы, как на этапе компиляции, так и на этапе выполнения.

3. Возможность удовлетворения требований структурного программирования, суть которого заключается в том, что с помощью нескольких конструкций можно выразить в принципе любые алгоритмы.

4. Наличие набора структурных типов данных: массивов, записей, множеств, файлов.

К недостаткам можно отнести отсутствие операции возведения в степень.

Язык программирования Паскаль – это универсальный язык, позволяющий решать самые разнообразные задачи обработки информации.

Во всех языках программирования определены способы организации данных и действий над данными. Кроме того, существуют элементы языка, включающие в себя множество символов (алфавит), лексемы (лексема - последовательность допустимых символов языка программирования, имеющая смысл для транслятора. Транслятор рассматривает программу как последовательность лексем) и другие изобразительные средства программирования. Несмотря на разнообразие языков программирования, их изучение происходит приблизительно по одной схеме. Это обусловлено общностью структуры языков высокого уровня (рисунок1).

Рисунок 1 - Структура языков программирования высокого уровня

Следует отметить сходство изучения естественных языков и языков программирования. Во-первых, для того чтобы читать и писать на иностранном языке, надо знать *алфавит* этого языка. Во-вторых, следует знать правила написания слов и предложений, т.е. *синтаксис* языка. В-третьих, необходимо понимать смысл слов и фраз, чтобы адекватно реагировать на них. Смысловое содержание языковой конструкции называется *семантикой*. Например, в салоне самолета засветилось табло «Fasten belts!» (Застегните ремни!). Зная английскую грамматику, можно правильно прочитать эту фразу, однако не понять её смысл и, следовательно, не выполнить соответствующих действий. Из грамотно написанных слов можно составить абсолютно бессмысленную фразу.

Таким образом, любой язык программирования образуют три основные составляющие: алфавит, синтаксис и семантика.

Соблюдение правил в языке программирования должно быть более строгим, чем в разговорном языке. Человеческая речь содержит значительное количество избыточной информации, то есть, не расслышав какое-то слово, можно не понять смысл фразы в целом. Слушающий или читающий человек может додумать, дополнить, исправить ошибки в воспринимаемом контексте.

Компьютер же – это автомат, воспринимающий всё «всерьез». В текстах программ нет избыточности, и компьютер сам не исправит даже очевидной (с точки зрения человека) ошибки. Он может лишь указать место, которое «не понял» и вывести замечание о предполагаемом характере ошибки. Исправить же ошибку должен программист.

ЭЛЕМЕНТЫ ЯЗЫКА ПАСКАЛЬ

Алгоритмический язык – это набор символов языка (алфавит языка), система правил составления из этих символов конструкций языка (синтаксис языка) и система правил истолкования этих конструкций (семантика языка).

Алфавит языка Паскаль содержит следующие символы:

- строчные и прописные буквы латинского алфавита;
- арабские цифры от 0 до 9;
- специальные символы (+, -, *, ; , :, «, /, ‘, =, [ ], ( ), < >, !, ?, пробел и др.);
- буквы русского алфавита (для написания комментариев и подсказок).

Комментарии могут быть включены в любое место программы. Комментарий – это любая последовательность символов, заключенных в фигурные скобки – {комментарии} или в ограничители вида – (*комментарий*). Комментарий не определяет никаких действий программы и является лишь пояснительным текстом. Программист пишет комментарии не для ПК, а для себя. Комментарий придает тексту программы большую ясность. Хорошо откомментированные программы называют самодокументированными. Во многих таких программах объем комментариев превышает объем вычислительных операторов.

Текст программы, записанный с помощью символов алфавита языка, представляет собой последовательность строк.

Для обозначения объекта программы используется идентификатор.

Идентификатор (имя) – это любая последовательность букв и цифр, начинающаяся с буквы. Рекомендуемая максимальная длина идентификатора - 8 символов.

ДАННЫЕ В ПАСКАЛЕ

Данные – это формализованное представление информации, пригодной для обработки. Каждый элемент данных в программе является либо переменной, либо константой.

Константа - это величина, значение которой не меняется в процессе выполнения программы. Константы могут быть представлены либо непосредственно своими

значениями, либо идентификатором. В последнем случае к ней можно обращаться по имени.

Синтаксис определения констант – это ***раздел описания констант***, запись которого начинается со служебного слова CONST. Тип константы определяется её записью.

```
CONST
    имя = ± константа;
```

Например:
```
CONST
    E =2.72;
```

Переменные – это величины, значения которых могут изменяться в процессе выполнения программы. Переменные должны быть представлены в программе идентификатором.

Синтаксис определения переменных - это ***раздел описания переменных***, запись которого начинается со служебного слова VAR (variables – переменные). С помощью описания устанавливается не только факт существования переменной, но и задается её тип. В одном разделе можно описать несколько переменных.

```
VAR
    идентификатор 1, идентификатор 2, …, идентификатор N:тип;
```

Например:
```
VAR
    A, B, C: INTEGER;
```

ТИПЫ ДАННЫХ

Любая переменная или константа в программе связана с определением типов данных, причем эта связь не может меняться во время выполнения программы.

Тип - это множество значений плюс множество операций над ними.

1. СТАНДАРТНЫЕ ТИПЫ ДАННЫХ (скалярные).
   1. Целый тип – INTEGER.
   2. Вещественный тип – REAL.
   3. Символьный тип – CHAR.
   4. Логический тип (булевский) – BOOLEAN.

Целый тип - INTEGER

Значение – все целые числа в диапазоне от -2147483648 до 2147483647.

Константы целого типа – это любые числа, записанные без десятичной точки: Например: 55, -14.

Переменные, принимающие в качестве своих значений константы целого типа, относятся к целому типу.

Вещественный тип - REAL

Значение – все числа, которые имеют дробную часть.

Константы вещественного типа могут быть представлены в двух формах:

1) с фиксированной точкой (-27.3);

2) с плавающей точкой (4E – 05 ($4*10^{-5}$), 0.62E+02 ($0.62*10^{2}$)) – знак и число, стоящие после символа E указывают, на сколько знаков вправо или влево необходимо сместить десятичную запятую.

Переменные, принимающие в качестве своих значений числа с фиксированной или плавающей точкой, относятся к вещественному типу.

Символьный тип - CHAR

Константа символьного типа – это любой символ алфавита, заключенный в апострофы:

Например: ‘W’, ‘№’

Переменные, принимающие в качестве своих значений константы символьного типа, относятся к символьным переменным.

Строковая константа – строка символов, заключенная в апострофы.

Например: ‘TRUE’, ‘Язык программирования Паскаль’

Логический тип - BOOLEAN

Значение – истина или ложь.

Константы логического типа – это TRUE (истина) и FALSE (ложь).

Переменные, принимающие в качестве своих значений константы логического типа, относятся к логическим переменным.

2. ТИПЫ ДАННЫХ, ОПРЕДЕЛЯЕМЫЕ ПОЛЬЗОВАТЕЛЕМ.

В языке Паскаль программист имеет возможность определять новые типы данных. Причем делать это нужно как непосредственно при описании переменных, так и в специальном ***разделе описания типов***, который обязательно должен располагаться перед разделом описания переменных.

```
TYPE
   идентификатор=тип;
```

Перечисляемый тип

Определяется, как упорядоченный набор идентификаторов, заданных путем их перечисления.

```
TYPE
      перечисляемый тип=(значение типа 1, …, значение типа N);
VAR
      идентификатор переменной:перечисляемый тип;
```

Например:

```
TYPE
      CHISLA=(0,1,2,3,4,5,6,7,8,9);
VAR
      A:CHISLA;
```

Ограниченный тип

Определяется путем наложения ограничения на стандартный тип или определенный ранее пользователем тип.

```
TYPE
   ограниченный тип::=минимальная константа..максимальная константа;
VAR
   идентификатор переменной:ограниченный тип;
```

: : - могут отсутствовать.

Ограничения определяются заданием диапазона: минимальная константа - нижняя граница диапазона, максимальная константа - верхняя граница диапазона.

Константы должны быть одного и того же типа. Тип константы может быть любым стандартным типом кроме вещественного типа.

Например:

```
TYPE
    CHISLA=0..9;
VAR
    A:CHISLA;
```

Если константы имеют стандартный тип, то описание ограниченности типа можно делать в разделе переменных.

Например:

```
VAR
    A:0..9;
```

СТРУКТУРА ПРОГРАММЫ

Программа на языке Паскаль состоит из заголовка программы, который начинается со служебного слова PROGRAM и собственно программы. Заканчивается программа точкой.

PROGRAM имя;

ПРОГРАММА.

Программа содержит раздел подключения модулей, разделы описаний и раздел операторов.

Раздел подключения модулей начинается со служебного слова USES, за которым следует список имен модулей, перечисляемых через запятую.

Разделы описаний:

1. Раздел меток.

LABEL

число 1, число 2, …, число N;

где число 1, …, число N - метки;

Метка представляет собой идентификатор или целое число без знака. Любой оператор в программе можно выделить, поставив перед ним метку, метка от оператора отделяется двоеточием (метка: оператор;).

2. Раздел констант.

```
CONST
    имя=± константа;
```

3. Раздел типов.

```
TYPE
    идентификатор=тип;
```

4. Раздел переменных.

```
VAR
    идентификатор 1, идентификатор 2, …, идентификатор N:тип;
```

5. Раздел подпрограмм (процедур или функций).

```
PROCEDURE (FUNCTION)
  раздел подпрограмм;
```

Этот раздел присутствует в программе, если программист помимо стандартных процедур и функций определяет свои, являющиеся самостоятельными программными единицами, к которым осуществляется обращение из основной программы с помощью указания имени этой процедуры (или функции) и ее параметров.

Любой из вышеперечисленных разделов может отсутствовать.

Раздел операторов представляет собой, так называемый составной оператор, который включает в себя последовательность выполняемых операторов, разделенных точкой с запятой, и ограниченных операторными скобками – BEGIN и END.

ВЫРАЖЕНИЯ

1. Арифметические выражения.

Арифметическое выражение задает правило вычисления значения переменных. Выражение строится из констант, переменных, операций, функций, разделенных знаками арифметических операций и круглыми скобками. Тип выражения совпадает с типом результата.

При составлении выражений следует выполнять следующие правила:

1. Нельзя записывать подряд два знака арифметических операций.

   нельзя: A+-B

   нужно: A+(-B)

2. Не допускаются верхние и нижние индексы, «двухэтажные» дроби.

   нельзя: x_1, x^2, $\frac{1}{2}$

   нужно: x1, SQR(X), 1/2

3. Можно использовать скобки только круглого типа.

4. Вычисления выполняются слева направо, в соответствии со старшинством операций. Если присутствуют круглые скобки, то сначала выполняются действия в скобках.

Порядок выполнения действий в арифметическом выражении:

1. Вычисление значений функций.
2. Смена знака (-).
3. Умножение *, деление /, операция DIV, операция MOD.
4. Сложение +, вычитание -.

Операция возведения в степень в языке Паскаль отсутствует. Для выполнения этой операции используется следующая математическая формула:

$$x^y = e^{y \cdot \ln(x)}$$

На Паскале это будет выглядеть так:

X^Y =EXP(Y*LN(X))

Например: X^5 = EXP(5*LN(X))

2. Логические выражения.

Логическое выражение задает правило для вычисления логического значения TRUE и FALSE. Выражение строится из логических данных (логических функций, констант, переменных, операций отношения (сравнения)), связанных логическими операциями.

Порядок выполнения действий:

1. Операция отрицания - NOT.
2. Умножение, деление, операции DIV, MOD, логическое умножение – AND.
3. Сложение, вычитание, логическое сложение – OR.
4. Операции отношения (<, >, <=, >=, =, <>) - имеют самый низкий приоритет, поэтому их следует заключать в круглые скобки.

Выражения с AND и OR вычисляются по короткой схеме:

X AND Y, если X ложно, то все выражение ложно, и Y не вычисляется;

X OR Y, если X истинно, то все выражение истинно, и Y не вычисляется.

Таблица основных стандартных функций языка и некоторых операций

Функция	Назначение	Тип аргумента	Тип функции
PI	Число π=3,1415926536	-	REAL
ABS(X)	Абсолютное значение X (|x|)	REAL INTEGER	REAL INTEGER
SQR(X)	Квадрат X (x^2)	REAL INTEGER	REAL INTEGER
SIN(X)	Синус X	REAL INTEGER	REAL REAL
COS(X)	Косинус X	REAL INTEGER	REAL REAL
ARCTAN(X)	Арктангенс X	REAL INTEGER	REAL REAL
EXP(X)	Экспонента X (e^x)	REAL INTEGER	REAL REAL
EXP10(X)	10 в степени X (10^x)	REAL INTEGER	REAL REAL
LN(X)	Натуральный логарифм X (ln x)	REAL INTEGER	REAL REAL
LOG(X)	Десятичный логарифм X (lg x)	REAL INTEGER	REAL REAL
SQRT(X)	Квадратный корень из X ($\sqrt{x}$)	REAL INTEGER	REAL REAL
TRUNC(X)	Ближайшее целое, не превышающее X по модулю	REAL	INTEGER
ROUND(X)	Округление X в сторону ближайшего целого	REAL	INTEGER
ODD(X)	TRUE – если X – нечетное FALSE – если X– четное	INTEGER	BOOLEAN
ORD(X)	Определение номера символа языка Пас-	CHAR	INTEGER

Функция	Назначение	Тип аргумента	Тип функции
	каль в десятичной системе счисления		
CHR(X)	Определение символа языка Паскаль по его порядковому номеру	INTEGER	CHAR
A DIV B	Вычисление частного при делении A на B	INTEGER	INTEGER
A MOD B	Вычисление остатка при делении A на B	INTEGER	INTEGER
INT(X)	Целая часть X	INTEGER REAL	REAL REAL
FRAC(X)	Дробная часть X	INTEGER REAL	REAL REAL
PRED(X)	Нахождение элемента, являющегося предыдущим для данного (в перечне допустимых элементов)	INTEGER BOOLEAN CHAR	INTEGER BOOLEAN CHAR
SUCC(X)	Нахождение элемента, являющегося следующим для данного (в перечне допустимых элементов)	INTEGER BOOLEAN CHAR	INTEGER BOOLEAN CHAR
RANDOM	Выбирается случайное число от 0 до 1		REAL
RANDOM(X)	Выбирается случайное число от 0 до X-1		INTEGER

PRED(TRUE)=FALSE

SUCC(FALSE)=TRUE

Если аргумент целый, то Y:=PRED(X) ≈ Y:=X-1

Y:=SUCC(X) ≈ Y:=X+1

ОПЕРАТОРЫ ЯЗЫКА ПАСКАЛЬ

Операторы языка Паскаль предназначены для обмена информацией между пользователем и ПК. Все данные, поставляемые компьютеру, организуются в файлы. Файлами являются:

программа пользователя;

наборы исходных данных, для этой программы;

наборы результатов выполнения программы;

Для ввода в программу исходных данных и вывода результатов используются

два стандартных файла: INPUT – для ввода данных;

OUTPUT – для вывода данных.

Можно считать, что вычислительный процесс осуществляется преобразованием текстового файла INPUT в файл OUTPUT. Эти файлы могут записываться в заголовке программ.

```
PROGRAM P1 (INPUT, OUTPUT);
```

По умолчанию они имеют следующие описания.

```
TYPE
   TEXT = PACKED FILE OF CHAR;
VAR
   INPUT, OUTPUT: TEXT;
```

ПУСТОЙ ОПЕРАТОР

Синтаксис оператора:

```
N:;
```

где N – метка.

Пустой оператор не задает в программе никаких действий, чаще всего встречается с меткой и ставится в конце составного оператора или программы.

СОСТАВНОЙ ОПЕРАТОР

Синтаксис оператора:

```
BEGIN
      оператор 1;
      оператор 2;
      …….. ;
      оператор N
 END;
```

Составной оператор задает последовательность операторов, заключенную в операторные скобки BEGIN и END. Точка с запятой не ставится после слова BEGIN, между последним оператором составного оператора и словом END. Выполнение составного оператора заключается в последовательном выполнении входящих в его состав операторов. Выход из составного оператора осуществляется либо через его

закрывающуюся операторную скобку, либо с помощью оператора перехода по метке, находящейся вне составного оператора.

ОПЕРАТОРЫ ВВОДА

Синтаксис операторов:

READ (V1,V2, ...,VN);

READLN (V1,V2, ...,VN); READ LINE (читать строку)

где V1, V2, …, VN – список переменных

Оператор ввода обеспечивает чтение данных из стандартного файла INPUT и присвоение прочитанных значений соответствующим переменным в порядке их следования. Типы вводимых значений должны соответствовать типу соответствующих переменных.

В качестве элементов списка ввода можно использовать только переменные вещественного, целого и символьного типов. Вводимые значения отделяются друг от друга пробелами или располагаются на строках разного уровня.

Оператор READLN выполняет действия, аналогичные оператору READ, но после ввода последней переменной из списка ввода осуществляется переход к началу новой строки файла INPUT. Оператор READLN без списка ввода реализует переход к началу новой строки.

Пример 1: Пусть дан фрагмент программы.

```
PROGRAM VVOD;
  VAR
     K, M, N: INTEGER;
     A1,A2,X,Y:REAL;
     BEGIN
          READ (A1,K,M);
          READ (X,Y,N);
          READ (A2)
     END.
```

Протокол работы:

2 10 15 2.5 8.3 1 4.8

В результате выполнения операторов READ переменные примут значения:

A1=2; K=10 и т. д.

ОПЕРАТОРЫ ВЫВОДА

Синтаксис операторов:

WRITE (E1,E2, ...,EN);

WRITELN (E1,E2, ...,EN); WRITE LINE (писать в строку)

где E1, E2, ..., EN - список выводимых выражений.

Оператор вывода осуществляет вывод значений выражений, указанных в списках вывода, в стандартный файл OUTPUT. Вид печатаемой единицы данных определяется типом соответствующего выражения.

Оператор WRITELN выполняет те же действия, что и оператор WRITE, но после вывода последнего выражения из списка вывода осуществляется переход на новую строку. Для того чтобы пропустить на экране строку нужно использовать оператор WRITELN без параметров, т. е. WRITELN.

При выводе на экран нескольких чисел в строку они не отделяются друг от друга пробелами.

Пример:

```
WRITE ('ЯП Паскаль',5*3);
WRITELN (7<10);
WRITELN (sqr(4));
```

Протокол работы:

```
ЯП Паскаль15TRUE
16
```

УПРАВЛЕНИЕ ФОРМОЙ ВЫВОДА ДАННЫХ

Если программиста не устраивает стандартная форма вывода, то он использует средства управления печатью (указатели форматов вывода). Для этого в Паскале используют две положительные целые величины:

– *ширина поля вывода* (количество позиций на экране, отводящихся для вывода данного выражения), которая указывается через двоеточие после соответствующего выражения в операторе вывода;

– *точность представления* (количество цифр в дробной части числа), которая указывается через двоеточие после ширины поля вывода.

В этом случае оператор вывода имеет вид:

WRITE(LN) (E1:M1:N1,E2:M1:N1, ...,EN:MN:NN);

где M_I (I=1,N) – это ширина поля вывода,

N_I (I=1,N) – точность представления.

Если E_I – выражение типа CHAR, то выводится символ и столько пробелов перед ним, чтобы общее количество знаков равнялось ширине поля.

Если E_I – типа INTEGER, то печать производится аналогично типу CHAR.

Если E_I – типа REAL и не задана точность представления, то число полностью выводится на экран.

Если E_I типа BOOLEAN, то печатается TRUE или FALSE и перед ними печатается столько пробелов, какова общая ширина поля.

```
WRITE ('A':3);                          Протокол работы:
WRITELN (2:1,5<7,2>7:8);                _ _A2TRUE_ _ _FALSE
```

ОПЕРАТОР ПРИСВАИВАНИЯ

Синтаксис оператора:

V:=E;

где V – переменная, E – выражение.

Оператор присваивания выполняется следующим образом: вычисляется значение выражения E, стоящего в правой части оператора, и это значение присваивается переменной V, стоящей в левой части.

При использовании оператора необходимо помнить, что тип выражения должен соответствовать типу переменной.

Допускаются следующие различия типов:

1) Переменной вещественного типа можно присвоить значение целого типа.
2) Переменной целого типа можно присвоить значение ограниченного типа.

Пример:

Вычислить процент выполнения плана выпуска продукции по предприятию.

```
PROGRAM PRIMER1;
  USES CRT;
  VAR
       P,F,PR:REAL;
   BEGIN
       WRITELN ('Введите план и фактический выпуск продукции');
       READLN (P, F);
```

```
    PR:=F/P*100;
    CLRSCR;
    WRITELN ('План выпуска продукции составляет:',P:5:2);
    WRITELN ('Фактический выпуск продукции составляет:',F:5:2);
    WRITELN ('Процент выполнения плана равен',PR:5:2)
END.
```

Задачи для самостоятельного решения:

1. Даны семь чисел, найти их среднее арифметическое.
2. Вычислить дробную часть среднего арифметического трех заданных положительных чисел.
3. Вычислить длину окружности, площадь круга и объем шара одного и того же заданного радиус.
4. Вычислить периметр и площадь прямоугольного треугольника по динам двух катетов.
5. По длине двух сторон треугольника и углу между ними вычислить длину третьей стороны и площадь треугольника.
6. Ввести любой символ языка Паскаль и определить его порядковый номер, а также предыдущий и последующий символы (использовать функции ORD, PRED и SUCC).

БЕЗУСЛОВНЫЙ ОПЕРАТОР

Синтаксис оператора:

GOTO метка;

Безусловный оператор служит для изменения порядка выполнения операторов и для перехода на оператор с меткой, которая указана в этом операторе.

Оператор GOTO следует применять в исключительных случаях; частое его использование затрудняет чтение программы и свидетельствует о невысокой культуре программирования.

С помощью оператора GOTO нельзя:

1. Передать управление во внутрь составного оператора.
2. Передать управление во внутрь оператора цикла.

3. Передать управление во внутрь оператора выбора.

4. Передать управление в процедуру.

5. Передать управление из одной ветви условного оператора в другой условный оператор.

УСЛОВНЫЙ ОПЕРАТОР

Синтаксис оператора:

1. Полная форма

```
IF условие THEN оператор 1 ELSE оператор 2;
```

2. Краткая форма

```
IF условие THEN оператор 1;
оператор 2;
```

Условный оператор используется при создании программ, в которых в зависимости от проверки какого-либо условия, определяется один или несколько вариантов возможных действий.

Условие – это логическое выражение, которое может принимать значение истина или ложь. Если условие истинно, то выполняется оператор, следующий за словом THEN, т.е. оператор 1; если условие ложно, то выполняется оператор, следующий за словом ELSE, т. е. оператор 2 - для полной формы оператора, либо оператор, следующий за условным - для сокращенной формы.

Для наглядности условный оператор следует записывать, сдвигая альтернативные части по отношению к условию.

```
IF условие
  THEN оператор 1
  ELSE оператор 2;
```

После слов THEN и ELSE могут стоять и условные операторы. Для определения соответствия слов IF и ELSE следует руководствоваться следующим правилом: **конструкция ELSE относиться к ближайшему IF, для которого не установлено соответствие.**

Например:
```
IF условие 1
   THEN IF условие 2
            THEN IF условие 3
                     THEN оператор 1
                      ELSE оператор 2
            ELSE оператор 3
    ELSE оператор 4;
```

Если после слов THEN и ELSE надо выполнить несколько операторов, то для этого используют операторные скобки BEGIN и END. Оператор в этом случае будет называться **составным условным оператором**.

```
IF условие
   THEN BEGIN
            оператор 1;
            оператор 2;
            ……;
            оператор N
        END;
```

Примеры:

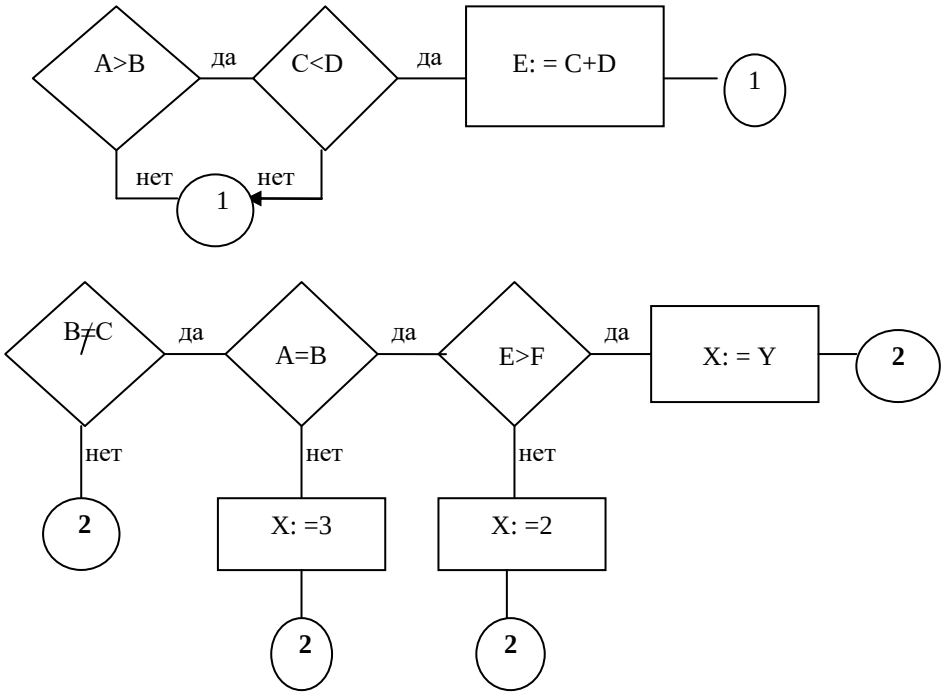
1. Вычислить переменную Z в зависимости от условия:

$$Z=\begin{cases} x^2, & \text{если } x<3 \\ x^2-2x+5, & \text{если } 3\le x\le 4 \\ x-2, & \text{если } x>4 \end{cases}$$

```
PROGRAM PRIMER2;
      USES CRT;
      LABEL
            1;
      VAR
            X,Z:REAL;
            OTV:CHAR;
      BEGIN
          1:CLRSCR;
            WRITELN (‘Введите произвольное число X’);
            READLN (X);
            CLRSCR;
            IF X<3
              THEN Z:=SQR (X)
```

```
        ELSE IF X<=4
                THEN Z:=SQR(X)-2*X+5
                ELSE Z:=X–2;
      WRITELN (‘При значении X =’,X:5:2);
      WRITELN (‘Значение Z=’,Z:5:2);
      WRITELN (‘Будете вводить новые данные (Y/N)?’);
       READLN (OTV);
      IF (OTV=’Y’) OR (OTV=’y’)
          THEN GOTO 1
END.
```

2. Записать условные операторы.



Задачи для самостоятельного решения:

1. $$Y=\begin{cases} X, \text{ если } X \geq 0 \\ -X, \text{ если } X < 0 \end{cases}$$

2. $$Y=\begin{cases} -X^2, & \text{если } X \leq 0 \\ 0, & \text{если } 0 \leq X \leq ½ \\ X - ½, & \text{если } X > ½ \end{cases}$$

3. Если X и Y отрицательные, то каждое из них возвести в квадрат, иначе меньшее из них заменить абсолютной величиной.

4. Даны три числа a, b, c. Найти min из этих чисел и проверить его на равенство 0. Если min=0, то повторить ввод всех трех чисел.

5. Найти остаток от деления целого выражения C=K*(A+B) на 4. Вывести на печать сообщение об остатке. Если остаток равен 0, то выражение C оставить без изменения. Если равен 1 или 3, уменьшить C на величину остатка. Если остаток равен 2, то увеличить C на величину остатка. Новое значение C вывести на печать.

6. Найти корни квадратного уравнения $a \cdot x^2 + b \cdot x + c = 0$.

7. Написать программу для определения подходящего возраста для вступления в брак, используя следующее соображение: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется соответственно как удвоенный возраст девушки минус 14. Данные для проверки работы программы задать самостоятельно.

ОПЕРАТОР ВАРИАНТА (ВЫБОРА) CASE

Синтаксис оператора:

```
CASE выражение OF
        список меток 1: оператор 1;
        список меток 2: оператор 2;
        ……..;
        список меток N: оператор N
[ELSE оператор]
END;
```

где CASE – вариант или выбор, OF – из, END – конец;

выражение – это выражение любого скалярного типа, кроме REAL.

список меток – это список разделенных запятыми значений выражения или одно из его значений.

Список меток должен содержать хотя бы одно значение выражения, указанного после ключевого слова CASE, следовательно, метки и выражение должны иметь

одинаковый тип. Метка - это необязательно целое число и она не описывается в разделе LABEL и на нее нельзя ссылаться в операторе GOTO.

Оператор выбора является обобщением условного оператора, т. е. дает возможность выполнить один из нескольких операторов в зависимости от значения некоторого выражения.

Оператор выполняется следующим образом:

1. Вычисляется значение выражения.
2. Среди меток находится это значение.
3. Выполняется тот оператор, метка которого равна значению выражения.
4. После выполнения оператора управление передается в конец оператора CASE.

Пример:

1. Определить площади различных геометрических фигур:

$$S=\begin{cases} a\cdot b, \text{ если n=1,} \\ \frac{a\cdot h}{2}, \text{ если n=2,} \\ \frac{(a+b)\cdot h}{2}, \text{ если n=3,} \\ \pi\cdot r^2, \text{ если n=4,} \\ \frac{\pi\cdot r^2\cdot\varphi}{360}, \text{если n=5.} \end{cases}$$

```
PROGRAM PRIMER3;
        USES CRT;
        LABEL
                1;
        VAR
                A,B,H,FI,S,R:REAL;
                N:INTEGER;
        BEGIN
                CLRSCR;
                WRITELN (‘Введите значения переменных A,B,H,FI,R’);
                READLN (A,B,H,FI,R);
                1:WRITELN (‘Введите N от 1 до 5’);
                READLN (N);
                        IF (N<1) OR (N>5)
```

```
                    THEN BEGIN
                                WRITELN (‘Повторите ввод’);
                                GOTO 1
                            END;
        CASE N OF
            1:S:=A*B;
            2:S:=A*H/2;
            3:S:=(A+B)*H/2;
            4:S:=PI*SQR(R);
            5:S:=PI*SQR(R)*FI/360
        END;
        WRITELN (‘Площадь фигуры равна ',S:7:2)
END.
```

Задачи для самостоятельного решения:

1. Составить программу, которая переводит числовую оценку знаний учащегося в ее словесный эквивалент (2 – «неудовлетворительно», 3 – «удовлетворительно», 4 – «хорошо», 5 – «отлично»). В случае неправильного ввода дать сообщение «Нет такой оценки».
2. Составить программу, которая бы по введенному номеру времени года (1 – зима, 2 – весна, 3 – лето, 4 - осень) выдавала соответствующие этому времени года месяцы, количество дней в каждом из месяцев.
3. Составить программу, которая выводит на экран меню:
    1. Первое
    2. Второе
    3. Третье

и в зависимости от выбранного пункта выдает одну из надписей: «Получите суп», «Получите картошку», «Получите компот», «Оставайтесь голодными».

ОПЕРАТОРЫ ЦИКЛА

В языке Паскаль существует 3 вида операторов цикла:

1. Оператор цикла с постусловием;
2. Оператор цикла с предусловием;

3. Оператор цикла с параметром.

ОПЕРАТОР ЦИКЛА С ПОСТУСЛОВИЕМ

Синтаксис оператора:

```
REPEAT
        оператор 1;
        оператор 2;
         ……
        оператор N
UNTIL  условие;
```

где REPEAT (повторять), UNTIL (до тех пор, пока) – служебные слова;

операторы – любые операторы языка Паскаль;

условие – логическое выражение.

Служебные слова REPEAT и UNTIL по действию похожи на операторные скобки BEGIN и END: между ними можно помещать группу операторов, отделяя их друг от друга точкой с запятой. Точка с запятой не ставится перед словом UNTIL.

Операторы в цикле REPEAT будут выполняться до тех пор, пока условие ложно, т.е. проверка условия производится после очередного выполнения цикла, что обеспечивает его выполнение хотя бы один раз.

Пример:

Найти сумму $S=1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{N}+\ldots$ Вычисления закончить, как только очередное слагаемое станет меньше ε.

```
PROGRAM PRIMER4;
        USES CRT;
        LABEL
                1;
        VAR
                S,EPS:REAL;
                N:INTEGER;
                OTV:CHAR;
        BEGIN
             1:CLRSCR;
                WRITELN (‘Введите точность вычислений’);
                 READLN (EPS);
```

```
        S:=1;
        N:=2;
        REPEAT
              S:=S+(1/N);
              N:=N+1
        UNTIL   (1/N)<EPS;
        WRITELN (‘Сумма =’,S:5:2);
        WRITELN (‘Будете вводить новые данные (Y/N)?’);
        READLN (OTV);
        IF (OTV=’Y’) OR (OTV=’y’)
           THEN GOTO 1
END.
```

ОПЕРАТОР ЦИКЛА С ПРЕДУСЛОВИЕМ

Синтаксис оператора:

```
WHILE условие
    DO оператор;
```

где WHILE (пока), DO (выполнить) – служебные слова;

оператор – любой оператор языка Паскаль;

условие – логическое выражение.

Оператор в цикле WHILE выполняется до тех пор, пока условие истинно. Если условие ложно, то выполняется оператор, следующий за WHILE. Если условие ложно с самого начала, то оператор не выполняется ни разу. Условие вычисляется и анализируется перед каждым выполнением цикла. Тело цикла может состоять из нескольких операторов. В этом случае необходимо использовать операторные скобки BEGIN и END.

Пример:

Найти сумму $S=1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{N}+\ldots$ Вычисления закончить, как только очередное слагаемое станет меньше ε. Использовать оператор цикла с предусловием.

```
WHILE (1/N)>EPS
    DO BEGIN
```

```
            S:=S+(1/N);
            N:=N+1;
END;
```

ОПЕРАТОР ЦИКЛА С ПАРАМЕТРОМ

Синтаксис оператора:

```
FOR  V:=E1  TO [DOWNTO] E2  DO
    оператор;
```

где FOR (для), TO (до), DOWNTO (вниз до), DO (выполнить) – служебные слова;

V – параметр цикла;

E1, E2 – граничные выражения, определяющие границу изменения параметра цикла (начальное и конечное значения параметра цикла).

Оператор цикла с параметром используется в том случае, если заранее известно число повторений цикла. Тип параметра цикла V и граничных выражений E1 и E2 должен совпадать, может быть любым, кроме вещественного. Тело цикла может состоять из нескольких операторов, в этом случае необходимо использовать операторные скобки BEGIN и END. После выхода из цикла значение параметра цикла становится неопределенным.

Оператор цикла выполняется следующим образом:

1. Вычисляются значения выражений E1 и E2 (один раз перед входом в цикл).
2. Параметру цикла V присваивается значение выражения E1.
3. Значение параметра цикла сравнивается со значением выражения E2. Если параметр цикла меньше или равен этому значению, то выполняется тело цикла, в противном случае выполнение цикла заканчивается – при использовании TO. При использовании DOWNTO тело цикла выполняется в том случае, если параметр цикла больше или равен значению E2.
4. Параметр цикла принимает следующее значение, полученное с помощью функции SUCC (в случае TO) или PRED (в случае DOWNTO). Происходит возврат к пункту 3.

S=1+2+…+100

```
S:=0;
FOR I:=1 TO 100 DO
  S:=S+I;
```

```
S:=0;
FOR I:=100 DOWNTO 1 DO
   S:=S+I;
```

Пример:

1. Вычислить переменную A:

$$A=\sum_{i=1}^{20} i^2+\prod_{n=10}^{1} n^3$$

```
PROGRAM PRIMER5;
        USES CRT;
        VAR
              S,P,I,N,A:INTEGER;
        BEGIN
              CLRSCR;
              S:=0;
              FOR I:=1 TO 20 DO
                   S:=S+SQR(I);
              P:=1;
              FOR N:=10 DOWNTO 1 DO
                   P:=P*SQR(N)*N;
              A:=S+P;
              WRITELN (‘Сумма=’,A:7);
        END.
```

Задачи для самостоятельного решения:

1. $N!=1\cdot 2\cdot 3\cdot ...\cdot N$

2. $Y=x^{10}=x\cdot x\cdot x\cdot x\cdot x\cdot x\cdot x\cdot x\cdot x\cdot x$

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ

Структурированный тип данных - это множество значений или переменных с одним общим именем.

Переменная, состоящая из нескольких компонентов, называется структурированной переменной.

РЕГУЛЯРНЫЙ ТИП (МАССИВЫ)

Для массивов в языке Паскаль характерно следующее:

1. Число элементов массива определяется при его описании и в дальнейшем не изменяется.

2. Каждый элемент массива может быть явно обозначен и к нему имеется прямой доступ.

Синтаксис описания одномерного массива:

```
TYPE
        имя типа=ARRAY [тип индекса] OF тип элементов;
VAR
        имя массива:имя типа;
```

Тип индекса может быть любым стандартным типом, кроме REAL.

Пример: Описать вещественный массив A(10).

```
TYPE
      VECTOR=ARRAY [1..10] OF REAL;
VAR
     A:VECTOR;
```

Можно обойтись без определения типа массива. Тогда описание массива имеет вид:

```
VAR
     имя массива: ARRAY [тип индекса] OF тип элементов;
```

При использовании ограниченного типа в качестве типа индекса можно употреблять символ константы:

```
CONST
       N=10;
VAR
       A: ARRAY [1..N] OF REAL;
```

Синтаксис описания двумерного массива:

```
TYPE
    имя типа=ARRAY [тип инд.1] OF ARRAY [тип инд.2] OF тип элементов;
VAR
    имя массива:имя типа;
```

Пример: Описать целочисленную матрицу B(20, 10).

```
TYPE
    MATRICA=ARRAY [1..20] OF ARRAY [1..10] OF INTEGER;
VAR
    B:MATRICA;
```

Описание можно сократить, указывая только имя массива и диапазон изменения индексов для каждой размерности массива.

Например:
```
VAR
    B:ARRAY [1..20, 1..10] OF INTEGER;
```

Обращение к элементам массива производится при помощи указания имени массива, за которым в квадратных скобках следуют номера индекса.

Например: $a_5 \rightarrow A[5]$

$b_{15,8} \rightarrow B[15, 8]$

Примеры:

1. Дан вектор X(N). Найти количество нулевых элементов, сумму положительных и произведение отрицательных элементов.

```
PROGRAM PRIMER6;
    USES CRT;
    LABEL
        1;
    VAR
        X:ARRAY [1..50] OF REAL;
        S, P:REAL;
        K,N,I:INTEGER;
        OTV:CHAR;
```

```
BEGIN
        1:CLRSCR;
        WRITELN (‘Введите размерность вектора’);
        READLN (N);
        (*ввод вектора*)
        WRITELN (‘Введите вектор X’);
        FOR I:=1 TO N DO
           BEGIN
                WRITE (‘X [‘,I:2,‘]=>’);
                READLN (X[I])
           END;
        (*вывод вектора*)
        WRITELN (‘Вектор X’);
        FOR I:=1 TO N DO
           WRITE (X[I]:5:2,‘ ’);
        WRITELN;

        K:=0;
        S:=0;
        P:=1;
        FOR I:=1 TO N DO
              IF X[I]=0
                THEN K:=K+1
                ELSE IF X[I]>0
                           THEN S:=S+X[I]
                           ELSE P:=P*X[I];
        WRITELN (‘Количество элементов вектора, равных нулю =’,K:2);
        WRITELN (‘Сумма положительных элементов вектора =’,S:5:2);
        WRITELN (‘Произведение отрицательных элементов =’,P:5:2);
        WRITELN (‘Вы завершили свою работу или будете работать еще?’);
        WRITELN (‘Если работа не закончена – введите Y(y)’);
        WRITELN (‘В противном случае введите N(n)’);
        READLN (OTV);
        IF (OTV=’Y’) OR (OTV=’y’)
          THEN
                GOTO 1
 END.
```

2. Найти максимальный элемент матрицы B(N,M) и номер строки и столбца, на пересечении которых он находится.

```
 PROGRAM PRIMER7;
      USES CRT;
      LABEL
```

```
        1;
VAR
        B:ARRAY [1..50, 1..50] OF REAL;
        MAX:REAL;
        NST,NSTR,N,M,I,J:INTEGER;
        OTV:CHAR;
BEGIN
        1:CLRSCR;
        WRITELN (‘Введите количество строк матрицы’);
        READLN (N);
        WRITELN (‘Введите количество столбцов матрицы’);
        READLN (M);
        (*ввод матрицы*)
        WRITELN (‘Введите матрицу B’);
        FOR I:=1 TO N DO
           FOR J:=1 TO M DO
                BEGIN
                     WRITE (‘B[‘,I:2,‘,’,J:2,’]=>’);
                     READLN (B[I,J])
                END;
        (*вывод матрицы*)
        WRITELN (‘Исходная матрица B’);
        FOR I:=1 TO N DO
           BEGIN
                FOR J:=1 TO M DO
                     WRITE (B[I, J]:5:2,‘ _’);
                WRITELN
           END;
        MAX:=B[1,1];
        NSTR:=1;
        NST:=1;
        FOR I:=1 TO N DO
              FOR J:=1 TO M DO
                   IF MAX<=B[I,J]
                      THEN BEGIN
                                   MAX:=B[I,J];
                                   NSTR:=I;
                                   NST:=J
                              END;
        WRITELN (‘Максимальный элемент матрицы’,MAX:5:2);
        WRITELN (‘Номер строки’,NSTR:2);
        WRITELN (‘Номер столбца’,NST:2);
        WRITELN (‘Будете вводить новые данные (Y/N)?’);
        READLN (OTV);
        IF (OTV=’Y’) OR (OTV=’y’)
```

```
                THEN GOTO 1
        END.
```

Задачи для самостоятельного решения:

1. Дан вектор X(n). Найти число нулевых, число и сумму отрицательных, число и сумму положительных элементов.
2. Найти число и сумму положительных элементов вектора, начиная с третьего элемента.
3. Найти произведение и количество элементов вектора, принадлежащих отрезку [-2;1].
4. Найти максимальный (минимальный) элемент вектора.
5. Найти сумму и количество элементов матрицы, принадлежащих отрезку [2;5]. Элементы равные 25 уменьшить на 10.
6. В матрице C(N,N) отрицательные элементы заменить их квадратами, подсчитать их количество.
7. В матрице найти произведение элементов, больших трех, и количество отрицательных элементов.
8. В матрице подсчитать сумму и количество элементов, меньших пяти, элементы, большие 10 умножить на 5.

УПАКОВАННЫЕ МАССИВЫ

Элементы массива запоминаются в последовательно расположенных ячейках памяти. Такой способ расположения эффективен для запоминания целых и вещественных чисел. При работе с другими типами элементов такой способ недостаточно экономен.

Для экономии памяти в языке Паскаль существуют средства, позволяющие более компактно расположить данные в памяти. Это делается с помощью упаковки данных. Например, массивы символьной информации упаковываются по два симво-

ла в одну ячейку. Компилятор делает это, если при описании массива указать ключевое слово PACKED перед ключевым словом ARRAY.

Синтаксис описания упакованного массива:

```
TYPE
        имя типа= PACKED  ARRAY [тип индекса] OF тип элементов;
VAR
        имя массива: имя типа;
```

СТРОКОВЫЙ ТИП ДАННЫХ (STRING)

Строковый тип обобщает понятие символьных массивов, позволяя динамически изменять длину строки. Строка – это последовательность символов. Каждый символ занимает 1 байт памяти. Количество символов в строке называется ее длиной.

Строковые величины могут быть константами и переменными.

Синтаксис описания строковых констант:

```
CONST
    идентификатор='значение';
```

Например:
```
CONST
        S1='Язык программирования Паскаль';
        S2='IBM PC - computer';
```

Синтаксис описания строковых переменных:

```
VAR
   идентификатор:STRING [ максимальная длина строки];
```

Максимальная длина строки может быть задана целым числом или константой целого типа. Это число находится в пределах от 1 до 255. Указание максимальной длины строки может быть опущено, в этом случае подразумевается число 255.

Например:
```
VAR
        S1:STRING [80];
        S2:STRING;
```

Строковые переменные занимают в памяти ПК на 1 байт больше, чем указанная в описании ее длина. Нулевой байт содержит значение текущей длины строки. Если строковой переменной не присвоено никакого значения, ее текущая длина равна нулю.

Символы внутри строки индексируются (нумеруются), начиная с единицы. К отдельным символам строки можно обращаться с помощью индексов, которые записываются в квадратных скобках после имени строки. Индекс может быть положительной константой, переменной величиной и выражением целого типа. Значение индекса не должно выходить за границы описания.

Например: S[5], S1[i], S2[k+1]

Тип STRING и тип CHAR совместимы. Строки и символы могут употребляться в одних и тех же выражениях.

Строковые выражения строятся из строковых констант, переменных, функций и знаков операций.

Над строковыми данными выполняются **следующие операции**: присваивание, сцепление (сложение), отношение.

1. Операция присваивания имеет вид:

имя строковой переменной:= строковое выражение;

В операциях присваивания можно использовать переменные типа CHAR.

Если длина строки выражения превышает максимально допустимую длину строки, то лишние символы отбрасываются.

Например:

```
VAR
    S1, S2: STRING [10];

S1:='ABCD';
S2:=S1;                    ('ABCD')
```

```
VAR
    S1: STRING [10];
    S2: STRING [2];

S1:='ABCD';
S2:=S1 ;                   ('AB')
```

2. Операция сцепления (сложения, +) объединяет несколько символьных строк в одну результирующую строку.

Например: ‘Язык программирования’ + ‘Паскаль’ =>

‘Язык программирования Паскаль’

```
PROGRAM STROKI;
   VAR
      S1, S2: STRING [15];
      C: CHAR;
   BEGIN
      S1:= ‘ABCD’;
      C:= S1[3];
      S2:= S1+S1+C;
      WRITELN (‘C=’, C);
      WRITELN (S2)
 END.
```

Протокол работы:

```
C=C
ABCDABCDC
```

3. Операции отношения (сравнения) применяются для сравнения двух строк, в результате чего получается логическая величина (True или False). Данная операция имеет более низкий приоритет, чем операция сцепления.

Сравнение строк производится слева направо до первого несовпадающего символа, и большей считается та строка, в которой первый несовпадающий символ имеет больший номер в таблице символьной кодировки.

Если строки имеют различную длину, но в общей части символы совпадают, считается, что более короткая строка меньше, чем более длинная.

Строки считаются равными, если они совпадают по текущей, а не по объявленной длине, и содержат одни и те же символы.

Например:

‘cosm1’<’cosm2’ True

‘pascal’>’PASCAL’ True

‘ключ_’<>‘ключ’ True

‘windows’=’windows’ True

ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

```
VAR
  S, S2 : STRING;
  N, Poz : INTEGER;
```

1. LENGTH (S) – определяет текущую длину строки S. Результатом является значение целого типа.

Пример:

Значение S	Выражение	Результат
‘test-5’	Length(S)	6
‘(A+B)*C’	Length(S)	7

2. COPY (S, Poz,N) - копирует из строки S строку длиной N символов, начиная с позиции Poz.

Пример:

Значение S	Выражение	Результат
‘ABCDEFG’	Copy(S,2,3)	‘BCD’
‘ABCDEFG’	Copy(S,4,4)	‘DEFG’

3. DELETE (S,Poz,N) - удаляет N символов из строки S, начиная с позиции Poz. Результат строкового типа.

Пример:

Исходное значение S	Оператор	Конечное значение S
‘abcdefg’	Delete(S,3,2)	‘abefg’
‘abcdefg’	Delete(S,2,6)	‘a’

4. INSERT (S1, S2, Poz) - вставляет строку S1 в строку S2, начиная с позиции Poz. Результат - строка S2 (если она содержит символов больше, чем объявлено, то лишние символы отбрасываются).

Пример:

Исходное значение S2	Оператор	Конечное значение S2
‘ЭВМ PC’	Insert(‘IBM-‘,S2,5)	‘ЭВМ IBM-PC’
‘Рис. 2’	Insert(‘№’,S2,6)	‘Рис. №2’

5. CONCAT (S1, S2, …, SN) – выполняет сцепление (конкатенцию) строк S1, S2, …, SN в одну строку. Результат строкового типа.

6. POS (S1, S2) – обнаруживает первое появление в строке S2 подстроки S1. Результат – целое число, равное номеру позиции, где находится первый символ подстроки S1. Если в строке S2 подстроки S1 не обнаружено, то результат равен 0.

Пример:

Значение S2	Выражение	Результат
‘abcdef’	Pos(‘cd’,S2)	3
‘abcdcdef’	Pos(‘cd’,S2)	3
‘abcdef’	Pos(‘k’,S2)	0

Пример:

Ввести строку длиной не менее 15 символов. Подсчитать общее количество введенных символов строки и количество знаков “+”, начиная с 7 символа по 15.

```
PROGRAM PRIMER8;
    USES CRT;
    LABEL
        1;
    VAR
        S:STRING [100];
        K,KP,I:INTEGER;
        OTV: CHAR;
    BEGIN
      1:CLRSCR;
      WRITELN (‘Введите строку длиной не менее 15 символов’);
      READLN (S);
      WRITELN (‘Введенная строка символов’);
```

```
    WRITELN (S);
    K:=LENGTH(S);
    WRITELN (‘Количество символов в введенной строке=’,K:3);
    KP:=0;
    FOR I:=7 TO 15 DO
        IF S[I]=‘+’
          THEN KP:=KP+1;
    WRITELN (‘Количество символов +, начиная с 7 по 15 символ=’,KP:2);
    WRITELN (‘Будете вводить новые данные (Y/N)?’);
    READLN (OTV);
    IF (OTV=’Y’) OR (OTV=’y’)
       THEN GOTO 1
END.
```

Задачи для самостоятельного решения:

1. Дана строка S, состоящая не более чем из 40 символов. В этой строке все вхождения букв ‘ABC’ заменить на ‘123’.

2. Используя функции для работы со строками получить из слова: а) «ВЕЛИЧИНА» слово «НАЛИЧИЕ»; б) «СТРОКА» слово «СЕТКА».

3. Сформировать символьную строку, состоящую из n звездочек (n – целое число, $1 \leq n \leq 255$).

4. Дан текст, в конце которого стоит точка. Подсчитать количество слов в тексте (слово – подстрока, не содержащая пробелов).

КОМБИНИРОВАННЫЙ ТИП (ЗАПИСИ)

При решении экономических и информационных задач, которые обрабатывают ведомости, документы, списки и т.д., возникает необходимость объединять различные типы данных в одну группу. Для этого в языке Паскаль введено понятие записи (RECORD).

Запись – это совокупность ограниченного числа логически связанных компонент различного типа.

Каждая компонента записи называется полем.

Запись, как и другие переменные, объявляется в разделе описания и используется в разделе операторов. Описание записи можно делать как в разделе описания типов TYPE, так и в разделе описания переменных VAR.

Синтаксис описания записи:

```
TYPE
        имя типа=RECORD
                имя элемента 1:тип;
                имя элемента 2:тип;
                …..
                имя элемента N:тип
        END;
VAR
        имя записи:имя типа;
```

Служебное слово RECORD (запись) выполняет роль открывающейся операторной скобки, END - закрывающейся операторной скобки. Внутри операторных скобок описываются элементы записи.

Для обращения к компонентам (полям) записи необходимо указать идентификатор записи, за которым ставится точка, а затем идентификатор поля (составное имя).

```
имя записи.имя поля
```

Имена полей внутри записи не должны повторяться. В языке Паскаль нет ни одной операции, которая воспринимала бы запись как нечто целое. Однако значение записи можно пересылать в другие переменные записи с помощью операторов присваивания.

Пример:

```
 TYPE
        STUDENT= RECORD
          FIO: STRING [20];
           D1, D2: 2..5;
           PR: INTEGER
        END;
VAR
        GRUPPA: ARRAY [1..25] OF STUDENT;
```

В данном примере был создан тип STUDENT, который представляет собой запись, включающую фамилию, оценки по двум дисциплинам и количество пропусков. Эти данные по каждому студенту объединены в группу, т.е. получен массив записей GRUPPA. Этот массив содержит 25 записей типа STUDENT. Обращение к отдельным элементам записей: GRUPPA [15]. FIO

GRUPPA [24]. PR

GRUPPA [13]. D1

При обращении к компонентам записи необходимость указывать составные имена приводит к удлинению программы и громоздкости. Для устранения этого неудобства в языке Паскаль используется **оператор присоединения**, который позволяет осуществлять доступ к компонентам записи, таким образом, как, если бы они были простыми переменными.

Синтаксис оператора:

```
WITH
      имя записи DO
         BEGIN
               операторы, содержащие имена элементов записи
         END;
```

Внутри этого оператора к компонентам записи можно обращаться только с помощью имени компонент.

Пример:

Для каждого студента указаны фамилия, оценки по пяти дисциплинам. Требуется вычислить средний балл каждого студента.

```
PROGRAM    PRIMER9;
  USES CRT;
  LABEL 1;
  TYPE
     VED = RECORD
         FIO: STRING[30];
         B1, B2, B3, B4, B5: 2..5;
        SB: REAL
    END;
```

```
VAR
    GRUPPA: ARRAY[1..25] OF VED;
     I, K: INTEGER;
    OTV: CHAR;
BEGIN
     1: CLRSCR;
     WRITELN('ВВЕДИТЕ КОЛИЧЕСТВО СТУДЕНТОВ В ГРУППЕ');
    READLN(K);
    FOR I := 1 TO K DO
       BEGIN
          WRITELN('ВВЕДИТЕ ДАННЫЕ ', I:2, '-ГО СТУДЕНТА:');
          WITH GRUPPA[I] DO
               BEGIN
                  WRITELN ('ВВЕДИТЕ ФАМИЛИЮ И ИМЯ СТУДЕНТА');
                  READLN (FIO);
                  WRITELN ('ВВЕДИТЕ 5 ОЦЕНОК ПО ДИСЦИПЛИНАМ');
                  READLN (B1);
                  READLN (B2);
                  READLN (B3);
                  READLN (B4);
                  READLN (B5);
                  SB:=(B1+B2+B3+B4+B5)/5
               END;
         END;
{ВЫВОД ВЕДОМОСТИ НА ЭКРАН}
    CLRSCR;
    WRITELN('-------------------------------------------------');
    WRITELN('|       ФИО         | 1 | 2 | 3 | 4 | 5 | СР.БАЛЛ |');
    WRITELN('-------------------------------------------------');
     FOR I := 1 TO K DO
       BEGIN
          WITH GRUPPA[I] DO
             BEGIN
                WRITE ('|', FIO:20, '|');
                WRITE (B1:2, ' |');
                WRITE (B2:2, ' |');
                WRITE (B3:2, ' |');
                WRITE (B4:2, ' |');
                WRITE(B5:2, ' |');
                WRITE(SB:7:2, '  |');
                WRITELN
             END
       END;
     WRITELN('-------------------------------------------------');
     WRITELN('БУДЕТЕ ВВОДИТЬ НОВЫЕ ДАННЫЕ (Y/N)?');
```

```
        READLN(OTV);
        IF (OTV = 'y') OR (OTV = 'Y')
            THEN GOTO 1
  END.
```

Задачи для самостоятельного решения:

1. Дана ведомость учащихся, занимающихся в кружке по информатике, и их оценка по информатике. Определить количество учащихся, занимающихся на «5», и количество учащихся с фамилией, начинающееся на букву «А».
2. Дана ведомость учащихся, и количество их пропусков за семестр (в часах). Определить количество учащихся, не имеющих пропусков, и количество учащихся, имеющих пропусков более 20 часов.

МНОЖЕСТВЕННЫЙ ТИП (МНОЖЕСТВА)

В языке Паскаль множеством называется конечный (от 0 до 255) неупорядоченный набор элементов (возможно пустой), различающихся между собой значениями, но принадлежащих одному и тому же простому типу, который называется базовым типом для данного множества.

В качестве базового типа может использоваться любой скалярный тип кроме вещественного REAL.

Имя дается всему множеству в целом. В отличие от элементов массива элементы множества не пронумерованы, не упорядочены. Каждый отдельный элемент множества не идентифицируется, и с ним нельзя выполнить какие-либо действия. Действия могут выполняться только над множеством в целом. Элементами множества могут быть константы, переменные, выражения, значения которых принадлежат базовому типу. При записи элементы множества заключаются в квадратные скобки.

Например, [3,4,7,9,12] – множество из пяти целых чисел;

[1..100] – множество целых чисел от 1 до 100;

[‘A’..’Z’,’?’,’!’] – множество, содержащее все прописные латинские буквы и знаки ? и !.

[ ] – пустое множество, т.е. множество, не содержащее никаких элементов.

Порядок записи элементов множества в нем не имеет значения.

Например, [1,2,3] эквивалентно множеству [3,2,1];

Каждый элемент во множестве учитывается только один раз.

Например, [1,2,3,4,2,3,4,5] эквивалентно множеству [1..5].

Множества определяются ключевым словом SET OF и следующим за ним базовым типом.

Множества могут быть объявлены как в разделе описания типов TYPE, так и в разделе описания переменных VAR.

Синтаксис описания множества:

```
TYPE
    имя типа = SET  OF базовый тип;
VAR
    имя множества: имя типа;
```

Мощность множества показывает, сколько элементов входит в данное множество.

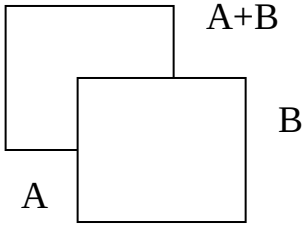
Например,

[ ]=0	[1,2,3,4,5]=5
[4]=1	[1,2,3..5]=5
[‘0’..’9’,’E’,’+’]=12	[1..5]=5

Операции над множествами:

1. Объединение множеств (+).

Объединением двух множеств называется множество, состоящее из элементов, принадлежащих обоим множествам.



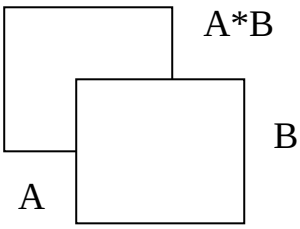
Например:

[1, 9] + [1..3] = [1, 2, 3, 9]

[‘A’..’C’] + [‘D’, ‘E’] = [‘A’..‘E’]

2. Пересечение множеств (*).

Пересечением двух множеств называется множество, состоящее из элементов, принадлежащих одновременно обоим множествам.



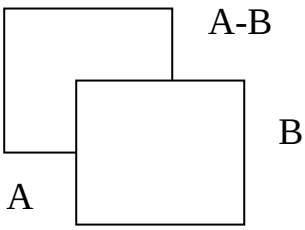
Например:

[0..4] * [5, 6] = [ ]

[‘A’..’F’] * [‘B’, ‘D’] = [‘B’..‘D’]

3. Разность (дополнение) множеств (-).

Разностью двух множеств называется множество, состоящее из тех элементов первого множества, которые не принадлежат второму множеству.



Например:

[1, 5, 9] - [2, 4, 8, 9] = [1, 5]

[‘A’, ‘B’, ‘C’, ‘D’] - [‘A’, ‘C’] = [‘B’, ‘D’]

4. Операция присваивания (:=).

S1:=S2 – множество S1 получит текущее значение множества S2.

Например, пусть M:=[3,4,7,9]

тогда M=[4,7,3,3,9]

5. Операции отношения и включения множеств:

= - равенство (совпадение) двух множеств

< > - неравенство множеств

Два множества считаются равными, если равны все их значения. Если множества отличаются хотя бы одним значением, то они не равны.

<= - проверка на вхождение множества из левого операнда в множество из правого операнда

>= - проверка на вхождение множества из правого операнда в множество из левого операнда

Все операции вырабатывают значения TRUE или FALSE в зависимости от проверки.

Например, [1,2,3]=[1,2] False

[1,2,3]<>[1,2,2] True

[1,2,3]>=[1,2] True

[3]=[1..100] True

6. Проверка принадлежности множеству (IN).

Правый операнд должен быть множеством, левый – значением базового типа множества. Эта операция вырабатывает значение TRUE, если значение элемента входит во множество, и значение FALSE в противном случае.

Например, 2 IN [1..10,12] True

5 IN [1,2,7,10] False

Данную операцию удобно использовать для исключения более сложных проверок, например оператор вида

IF (SIM='A') OR (SIM='B') OR (SIM='X')

THEN оператор;

можно заменить гораздо более наглядной и компактной формой

IF SIM IN ['A','B','X']

THEN оператор;

Порядок выполнения операций:

1) пересечение *
2) объединение + , разность -
3) вхождение IN, равно = , неравно < >, <=, >=

Ввод множества осуществляется при помощи операции объединения (+), а вывод – при помощи операции проверки принадлежности (IN).

Пример:

Даны два множества, состоящие из 10 элементов. Получить множество Y=X1+X2+(X1-X2).

```
PROGRAM PRIMER10;
    USES CRT;
    LABEL
        1;
```

```
VAR
        X1, X2, Y: SET OF 1..10;
        I, N1, N2: INTEGER;
        OTV: CHAR;
BEGIN
        1:CLRSCR;
        WRITELN ('Введите два множества');
        X1:=[ ];
        X2:=[ ];
        Y:=[ ];
        FOR I:=1 TO 10 DO
            BEGIN
                WRITE ('Введите',I:2,' элемент первого множества: ');
                READLN (N1);
                X1:=X1+[N1]
            END;
        FOR I:=1 TO 10 DO
            BEGIN
                WRITE ('Введите',I:2,' элемент второго множества:  ');
                READLN (N2);
                X2:=X2+[N2]
            END;
        WRITELN;
        WRITELN ('Первое множество');
        FOR I:=1 TO 10 DO
            IF I IN X1
              THEN WRITE (I:2,'  ');
        WRITELN;
        WRITELN ('Второе множество');
        FOR I:=1 TO 10 DO
            IF I IN X2
              THEN WRITE (I:2,'  ');
        WRITELN;
        Y:=X1+X2+(X1-X2);
        WRITELN ('Полученное множество');
        FOR I:=1 TO 10 DO
            IF I IN Y
              THEN WRITE (I:2,'  ');
        WRITELN;
        WRITELN ('Будете работать ещё? (Y/N)?');
        READLN (OTV);
        IF (OTV='Y') OR (OTV='Y')
          THEN GOTO 1
END.
```

Задачи для самостоятельного решения:

1. Дана символьная строка. Требуется определить в ней количество знаков препинания.

2. Дана строка из строчных русских букв. Между соседними словами – запятая, за последним словом – точка. Составить программу, печатающую в алфавитном порядке: а) все гласные буквы, которые входят в каждое слово; б) все согласные буквы, которые не входят ни в одно слово.

ПОДПРОГРАММЫ В ПАСКАЛЕ

Подпрограммы (ПП) – это часть основной программы, оформленная в виде, допускающем многократное обращение к ней из разных точек основной программы.

В языке Паскаль выделяют два вида ПП:

1. Процедура PROCEDURE.

2. Функция FUNCTION.

Любая программа может содержать несколько процедур и функций. ПП в свою очередь также может содержать ПП.

Процедуры и функции объявляются в разделе описания вслед за разделом переменных.

Выполнение программы начинается с операторов основной программы. При необходимости вызывается ПП, и начинают действовать ее операторы. Затем управление передается в основную программу, которая продолжает выполняться.

ПП оформляются подобно основной программе, т.е. состоят из заголовка и тела (блока).

Телом является блок, состоящий из разделов описаний и разделов операторов.

Все имена, представленные в разделе описаний основной программы, называют глобальными. Они действуют как в разделе операторов основной программы, так и в любой ПП.

Имена, представленные в разделе описаний ПП, называют локальными. Они действуют только в рамках ПП и недоступны операторам основной программы.

ПП-ПРОЦЕДУРЫ

Заголовок ПП-процедуры содержит имя процедуры и список формальных параметров, который может отсутствовать.

PROCEDURE имя процедуры (список формальных параметров);

PROCEDURE имя процедуры;

С помощью параметров осуществляется передача исходных данных в процедуру, а также передача результатов работы обратно в вызывающую ее основную программу. В списке формальных параметров перечисляются идентификаторы, и для каждого из них определяется тип. Если в теле процедуры есть идентификаторы, которые не являются формальными параметрами, то они считаются глобальными по отношению к данному описанию.

Вызов и выполнение процедуры осуществляется при помощи оператора процедуры, который имеет вид:

имя процедуры (список фактических параметров);

имя процедуры;

В Паскале различают четыре вида формальных параметров:

1. Параметры-значения.
2. Параметры-переменные.
3. Параметры-процедуры.
4. Параметры-функции.

Если списку формальных параметров не предшествует никакой символ, то они все являются параметрами-значениями.

Фактическим параметром соответствующему параметру-значению должно быть выражение.

В качестве начального значения для формального параметра пересылается текущее значение соответствующего фактического параметра. Значение формального параметра может меняться при выполнении процедуры, однако влияние на значение фактического параметра не оказывает. Следовательно, параметр-значение не может быть результатом выполнения процедуры. Параметр-значение можно передавать только в одном направлении: из программы в процедуру.

В случае использования параметров-значений, формальный параметр – это просто локальная переменная, которой в начале присваивается значение соответствующего фактического параметра. После этого связи между фактическими и формальными параметрами нет. Поэтому невозможно случайно или преднамеренно испортить значение переменной, существующей вне процедуры.

Если в списке формальных параметров списку идентификаторов предшествует VAR, то параметры этого списка называются параметрами-переменными.

В этом случае фактическими параметрами также являются переменные. Если некоторый параметр процедуры представляет собой результат ее выполнения, то он обязательно должен специфицироваться как формальный параметр-переменная.

Если в качестве формальных параметров выступает массив, и он соответствует формальному параметру-переменной, то это означает, что операторы, составляющие тело процедуры могут не только использовать этот массив, но и менять его.

Если списку формальных параметров предшествует служебное слово PROCEDURE, то параметры этого списка называются параметрами-процедурами.

Фактическим параметром в этом случае должен быть идентификатор процедуры.

Если списку формальных параметров предшествует служебное слово FUNCTION, то параметры этого списка называются параметрами-функциями.

Фактическим параметром в этом случае должен быть идентификатор функции.

Между формальными и фактическими параметрами должно быть полное соответствие, т.е.

- формальных и фактических параметров должно быть одинаковое количество;

- порядок следования формальных и фактических параметров должен быть один и тот же;

- тип каждого фактического параметра должен совпадать с типом соответствующего ему формального параметра.

Процедура возвращает результат в основную программу не только при помощи параметров-переменных, но и непосредственно изменяя глобальные переменные.

Локальные переменные порождаются при каждом входе в процедуру и уничтожаются при выходе из этой процедуры, т.е. они существуют только при выполнении процедуры.

ПП-ФУНКЦИИ

Заголовок ПП-функции имеет вид:

FUNCTION имя функции (список формальных параметров): тип функции;

FUNCTION имя функции: тип функции;

За заголовком функции следует тело функции, оформленное в виде блока (раздел описаний и раздел операторов).

Функция, как и процедура, может содержать несколько операторов, несколько входных параметров, но результат ее выполнения только один.

Этот единственный результат обозначается именем функции и передается в основную программу.

Обращение к функции происходит с помощью указателя функции, за которым в круглых скобках следует список фактических параметров, разделяемых запятыми. Этот указатель функции может использоваться только в выражениях.

В теле функции обязательно должен присутствовать оператор присваивания, в левой части которого стоит имя функции.

При описании функции в ее заголовке могут быть указаны:

1. Параметры-значения.
2. Параметры-процедуры.
3. Параметры-функции.

Параметры-значения используются для передачи исходных данных в ПП, в списке формальных параметров перечисляются через запятую с обязательным указанием их типов.

В качестве соответствующего фактического параметра может быть использовано любое выражение идентичного типа (константы или переменные).

В заголовке функции параметры-переменные использовать не рекомендуется, т.к. если результатов выполнения ПП несколько, то целесообразнее применять ПП-процедуру.

Спецификация параметра-процедуры – это заголовок процедуры, а параметра-функции – это заголовок функции, например:

FUNCTION PR(A,B:REAL; FUNCTION PR:REAL):REAL;
параметр-функция

FUNCTION PR(A,B:REAL; PROCEDURE PR):REAL;
параметр-процедура

Пример:

Даны вектор A(10) и вектор B(10). Найти сумму и произведение элементов каждого вектора.

```
PROGRAM PRIMER11;
  USES CRT;
  TYPE
    MASSIV=ARRAY [1..10] OF REAL;
  VAR
    A,B:MASSIV;
    SA,PA,SB,PB:REAL;
  PROCEDURE VVOD(K:INTEGER;
                          VAR
                                X:MASSIV);
    VAR
          I:INTEGER;
    BEGIN
      FOR I:=1 TO K DO
        BEGIN
          WRITE ('X[',I:2,']=>');
          READLN(X[I])
        END
    END;
   PROCEDURE VIVOD(K:INTEGER;
                           VAR
                                X:MASSIV);
    VAR
```

```
            I:INTEGER;
  BEGIN
     FOR I:=1 TO K DO
       WRITE (X[I]:5:2,' ');
     WRITELN
  END;
PROCEDURE SHET(K:INTEGER;
                        VAR
                              S1,P1:REAL;
                              X1:MASSIV);
  VAR
            I:INTEGER;
  BEGIN
     S1:=0;
     P1:=1;
     FOR I:=1 TO K DO
       BEGIN
          S1:=S1+X1[I];
          P1:=P1*X1[I]
       END
  END;
BEGIN
   WRITELN('Введите вектор A');
   VVOD (10,A);
   WRITELN(' Введите вектор B');
   VVOD (10,B);
   WRITELN('Исходный вектор A');
   VIVOD (10,A);
   WRITELN('Исходный вектор B');
   VIVOD (10,B);
   SHET (10,SA,PA,A);
   WRITELN('Сумма элементов вектора A=',SA:5:2);
   WRITELN('Произведение элементов вектора A=',PA:7:2);
   SHET (10,SB,PB,B);
   WRITELN('Сумма элементов вектора B=',SB:5:2);
   WRITELN('Произведение элементов вектора B=',PB:7:2);
END.
```

Задачи для самостоятельного решения:

Даны два вещественных вектора A и B. Логической переменной C присвоить значение TRUE, если у вектора A количество положительных элементов больше, чем у вектора B, и значение FALSE в противном случае.

Заключение

Программирование все в большей степени становится занятием для профессионалов. В понятие «компьютерная грамотность» входит, прежде всего, навык использования многообразных средств информационных технологий. Решая ту или иную информационную задачу, необходимо выбрать адекватное программное средство. Это могут быть электронные таблицы, системы управления базами данных, математические пакеты и т.п. И только в том случае, когда подобные средства не дают возможности решить задачу, следует прибегать к универсальным языкам программирования.

Задачей при первоначальном изучении программирования является освоение основ структурной методики программирования, которая остается основой программистской культуры. Для этой цели наиболее подходящим средством является язык программирования Паскаль.

Данное учебное пособие предназначено для освоения базовых понятий языков программирования высокого уровня в их реализации на Паскале, что является необходимым для освоения общих и профессиональных компетенций.

Методическая разработка может быть рекомендована для студентов, изучающих дисциплину «Основы программирования», а также в помощь преподавателям, читающим данную дисциплину.

Список использованных источников

1 Голицына О. Л., Попов И. И. «Основы алгоритмизации и программирования»: Учеб. пособие. – М.: ФОРУМ: ИНФРА-М, 2002.

2 Гуда А.Н. Информатика. - М.: Издательско-торговая корпорация «Дашков и К°»; Ростов н/Д: Наука-Спектр, 2009.

3 Долинер Л.И. Основы программирования в среде PascalABC.NET: учебное пособие/Л.И.Долинер. [Электронный ресурс]. Екатеринбург: Изд-во Урал. ун-та, 2014. - 128 с. (Федеральная государственная информационная система «Национальная электронная библиотека» htpp://нэб.рф/)

4 Культин Н.Б. Turbo Pascal в задачах и примерах - СПб.: БХВ-Петербург, 2010.

5 Павловская Т.А. Паскаль. Программирование на языке высокого уровня. Учебник, 2007.

6 Пильщиков В.Н. Сборник упражнений по языку Паскаль / В.Н. Пильщиков. – М.: Наука, 1989.

7 Семакин И.Г. Основы алгоритмизации и программирования: учебник для студ. учреждений сред. проф. образования / И.Г. Семакин, А.П. Шестаков. — М.: Издательский центр «Академия», 2013. - 304с.