ВВЕДЕНИЕ

В предлагаемом учебном пособии рассматриваются вопросы использования изучаемого теоретического материала по курсу «Алгоритмизация и программирование» для решения на практических занятиях и при самостоятельном изучении. Настоящее пособие рекомендуется студентам при подготовке к занятиям, контрольным работам.

Целью настоящего учебника является изложение базовых положений теории алгоритмов и основ программирования. Каждое практическое занятие содержит теоретический материал, решение основных задач и список задач для самостоятельного выполнения поданной теме. К основным алгоритмическим конструкциям и ряду решенных практических заданий приводятся блок-схемы.

В результате изучения материала студент должен:

знать

— основные понятия теории алгоритмов, базовые алгоритмические конструкции;

— методы построения и реализации алгоритмов;

— основные конструкции и структуры данных языка Pascal;

— основные подходы к организации сложных структурированных данных и способы их реализации;

уметь

— разрабатывать алгоритмы обработки данных на основе базовых алгоритмов, применяемых к данным различного типа;

владеть

— навыками практического программирования на языке высокого уровня Pascal;

— методами декомпозиции сложных задач и реализацией отдельных элементов в виде подпрограмм.

Фактический материал учебного пособия неоднократно апробирован автором, а также учениками и студентами, и как показала практика, дает хороший результат при применении его на первых курсах и в старших классах.

Авторы

Практическая работа № 1

Структура программы. Процедуры ввода-вывода, оператор присваивания

Структура программы на языке Pascal

Структура программы на языке PASCAL состоит из следующих частей:

1) заголовок;
2) описательный блок (раздел описаний):
- ✓ раздел меток — label;
- ✓ раздел констант — const;
- ✓ раздел типов — type;
- ✓ раздел переменных — var;

3) блок процедур и функций;
4) исполнительный блок (блок основных операторов).

```
Program <имя программы> (Input, Output);
    Uses <имя 1>[,<имя 2>...];
    Label m1,m2,...;
    Const [<константа 1 = значение 1>,...,< константа n = значение n >];
    Type [<имя типа1 = тип>,...,< имя типа n = тип>];
    Var <переменная 1> [,<переменная 2>,...,<переменная n>] :<тип>;
     Procedure <имя процедуры>[( параметры)];
          Begin
               <тело процедуры>;
          End;
     Function <имя> (аргументы):<тип значения>;
           Begin
               <тело функции>;
           End;
     Begin
          <тело программы>;
     End.
```

В разделе меток объявляемые метки могут быть перечислены в произвольном порядке независимо от того, в каком порядке эти

метки встречаются в разделе операторов, при этом любая метка в разделе операторов может быть указана только один раз.

В разделе констант объявляемые константы — это переменные, которые не изменяются в процессе выполнения программы. Константе дается определенное имя, которое используется в качестве ее синонима.

В разделе типов описываются создаваемые новые типы. Стандартные типы (integer, real, char и т. д.) считаются уже описанными.

В разделе переменных объявляемые переменные — это величины, которые изменяются в процессе выполнения программы. Каждая переменная, до работы с ней должна быть описана, т. е. указан тип переменной. Попытка присвоить переменной значение иного типа в процессе выполнения программы считается ошибкой. Любая используемая ячейка памяти требует описания. При описании ячейке памяти дается имя (идентификатор), по которому с ячейкой работают, и тип, показывающий, какие значения могут находиться в данной ячейке. Когда объявлена переменная, то в ней изначально находиться произвольное значение. Для изменения значения переменной используется оператор присваивания.

Блок процедур и функций содержит подпрограммы, оформленные специальным образом.

В основном блоке программы (разделе операторов) задаются действия, которые должны быть выполнены по определенному алгоритму. Последовательность операторов программы заключается в операторные скобки:

```
Begin
    Оператор1;
    Оператор2;
    Оператор3;
    ……
    ОператорN;
End.
```

Весь раздел операторов представляет собой один составной оператор.

Составной оператор — последовательность произвольных операторов, заключаемых в операторные скобки **Begin … End.** Допускается произвольная глубина вложения составных операторов.

```
Begin {начало раздела операторов}
    begin
        begin
            begin
                …
            end
        end
    end
End. {конец программы}
```

Существуют правила, по которым ставится точка с запятой:

— точка с запятой разграничивает операторы;

— точка с запятой не ставится перед **else** (трактуется как ошибка);

— точка с запятой не ставится перед **until** (трактуется как пустой оператор);

— точка с запятой не ставится перед **end** (в зависимости от версии языка программирования).

Программы строятся на основе базовых структур: следование, ветвление и цикл. Особенность базовых структур заключается в том, что у каждой базовой структуры ровно один вход и ровно один выход. В результате структурная программа также имеет только один вход и один выход.

Процедуры ввода/вывода языка Pascal

Для выполнения ввода/вывода информации существуют четыре стандартные процедуры:

Read (x1, x2, x3); — ввод переменных x_1, x_2, x_3, которые последовательно принимают значения, вводимые с клавиатуры.

Readln (x1, x2, x3); — ввод переменных x_1, x_2, x_3, которые последовательно принимают значения, вводимые с клавиатуры, и после этого происходит переход на новую строку.

Readln; — процедура задержки экрана.

Write (x1, x2, x3); — вывод на экран значения переменных x_1, x_2, x_3.

Writeln (x1, x2, x3); — вывод на экран значения переменных x_1, x_2, x_3 с переходом курсора на новую строку.

Writeln; — осуществляется переход на новую строку.

Например:

✓ **Write** ('x1='); — процедура вывода выводит на экран комментарий.

✓ **Write** ('x1=', x); — процедура вывода выводит на экран комментарий (текст, заключённый в апострофах) и значение переменной x.

✓ **Write** (x1+x2); — процедура вывода выводит на экран результат вычисления арифметического выражения, заключённого в скобках.

Оператор присваивания

Оператор присваивания — это один из основных операторов языка Turbo Pascal. В левой части указывается имя переменной, правая часть — это выражение того же типа, что и переменная. Символы «:=« связывают левую и правую части оператора присваивания и означают «присвоить значение». Данные символы рассматриваются как один специальный символ и пишутся слитно.

Например: a := b + c;

Оператор присваивания работает следующим образом: сначала вычисляется выражение в правой части, а затем результат вычисления кладется в переменную, стоящую в левой части.

Примеры

1. Составить программу «Приветствие».

```
Program example_1_1;
 begin
      writeln (‘Здравствуй, компьютер!’);
      write (‘Привет,’); writeln (‘студент.’)
 end.
```

2. Найти сумму двух чисел

```
Program example_1_2;                {заголовок программы}
Var X,Y, Sum: Real;                 {раздел объявления переменных}
Begin                               {начало основного блока программы}
  Write (‘Введите числа X и Y’);    {вывод сообщения на экран}
  Readln (X,Y);                     {чтение значений двух чисел с клавиатуры}
  Sum:=X+Y;                         {определение суммы}
  Writeln (‘Сумма чисел X и Y равна’, Sum)   {вывод результата}
End.                                {конец программы}
```

Практическая работа № 2

Простые типы данных. Линейные алгоритмические конструкции

Тип	Описание
Shortint, Integer, Longint, Byte, Word	Целочисленные данные
Real, Single, Double, Extended, Comp	Вещественные данные
Char	Символьные данные. Символ занимает 1 байт
Boolean	Логические данные. Занимает 1 байт, имеет два значения: false (ложь) и true (правда).

В целочисленном типе данных существует пять подтипов, различающихся:

— множеством значений;
— количеством занимаемой памяти.

Тип	Название	Диапазон значений	Размер
Shortint	Короткое целое	–128,127	8 бит
Integer	Целое	–32768, 32767	16 бит
Longint	Длинное целое	–2147483648, 2147483647	32 бита
Byte	Байт	0, 255	8 бит
Word	Слово	0, 65535	16 бит

Типы данных для представления вещественных чисел:

Тип	*Название*	*Диапазон значений*	*Размер*
Real	Вещественный	2.9E–39, 1.7E+38	6 байт
Single	Одинарный	1.5E–35, 3.4E+38	4 байта
Double	Двойной	5.0E–324, 1.7E+308	8 байт
Extended	Расширенный	3.4E–4932, 1.1E+4932	10 байт
Comp	Комплексный	–9.2E+18, 9.2E+18	8 байт

Операции, используемые для работы с целыми переменными

Знак операции	Назначение	Приоритет
+	Сложение	2
–	Вычитание	2
×	Умножение	1
Div	Целая часть от деления	1
Mod	Остаток от деления	1

M **mod** N = M – ((M **div** N) × N), если M >= ((M **div** N) × N)
M **mod** N = M – ((M **div** N) × N) + N, если M < ((M **div** N) × N)

Примеры:

9 div 2 = 4 (–9) div 2 = –3
3 div 5 = 0 (–7) div (–2) = 3
9 mod 2 = 1 (–14) mod 3 = 1
3 mod 5 = 3 (–10) mod 5 = 0

Операции, используемые для работы с вещественными переменными

Знак операции	Назначение	Приоритет
+	Сложение	2
–	Вычитание	2
×	Умножение	1
/	Деление	1

Арифметические операции применимы только к величинам целых и вещественных типов. При выполнении операции деления целых или вещественных чисел получаем вещественное число.

Значения в переменных вещественного типа хранятся приближенно. При задании вещественного числа отдельно задаются мантисса и порядок. Такой способ задания вещественного числа называется экспоненциальным форматом.

Например: A := 5.46E12;

Вещественной переменной разрешено присваивать значения целого типа. Целой переменной нельзя присваивать значения вещественного типа.

Стандартные функции и операции

Функция	Тип аргумента	Тип функции	Назначение
Abs (x)	целое вещественное	целое вещественное	|x| — абсолютная величина x
Sqr (x)	Целое вещественное	целое вещественное	x^2 — возведение величины x в квадрат
Trunc (x)	вещественное	целое	Выделение целой части величины x
Round (x)	вещественное	целое	Округление величины x до целого числа
Succ (x)	порядковое	порядковое	Следующее за величиной x число
Pred (x)	порядковое	порядковое	Предыдущее перед величиной x число
Int (x)	вещественное	вещественное	Выделение целой части

			величины x

Продолжение табл.

Функция	Тип аргумента	Тип функции	Назначение
Frac (x)	вещественное	вещественное	Выделяет дробную часть величины x
Random (x)	Целое	целое	Случайное число от 0 до x – 1
Randomize			Оператор, позволяющий генерировать новую последовательность случайных чисел при новом запуске программы на выполнение
Ln (x)	Вещественное	Вещественное	ln(x) — натуральный логарифм величины x
Cos (x)	Вещественное	Вещественное	cos(x) — косинус величины x
Sin (x)	Вещественное	Вещественное	sin(x) — синус величины x
Exp (x)	Вещественное	Вещественное	e^x — экспонента величины x
ArcTan (x)	Вещественное	Вещественное	arctg(x) — арктангенс величины x
Sqrt (x)	Вещественное	Вещественное	$\sqrt{x}$ — корень квадратный величины x

Round (x) = **Trunc** (x + 0,5), если x >= 0
Round (x) = **Trunc** (x – 0,5), если x < 0

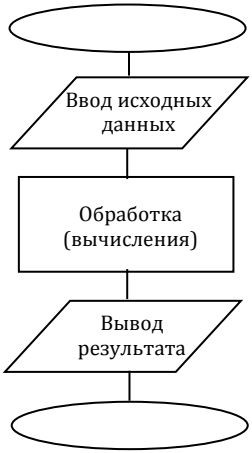
Примеры:

trunk (5,234) = 5	round (5,234) = 5
trunk (–5,234) = –5	round (–5,234) = –5
trunk (–5,934) = –5	round (–5,934) = –6
frac (123,456) = 0,456	int (123,456) = 123,0
frac (–123,456) = –0,456	int (–123,456) = –123,0

Для получения случайных чисел используют функцию Random (x). Данная функция формирует случайное число от 0 до X целого или вещественного типа. Перед обращением к функции ее целесообразно инициализировать, используя процедуру Randomize. X — параметр, указывающий диапазон значений случайного числа. Оно изменяется в пределах от 0 до X. Результат в этом случае имеет тип Word (диапазон значений: 0 … 65535). Если параметр X не задан, то результат будет типа Real в пределах $0{,}0 \le X < 1{,}0$.

Линейные алгоритмические конструкции

Программы с линейной структурой являются простейшими и используются, как правило, для реализации простых вычислений по формуле. Алгоритм с линейной структурой — это последовательность инструкций, которые выполняются друг за другом. В программах данного типа, как правило, используются три оператора: оператор ввода, присваивания, оператор вывода информации. Алгоритм программы с линейной структурой может быть представлен в виде схемы:



Примеры

1. Составить программу, вычисляющую значение переменной m при данных i, j, k: $m = \frac{i+j}{k+1}$.

```
Program example_2_1;
   Var i, j, k: Integer; m: Real;
   Begin
        Write ('Введите значения для i, j и k =>:');
        Readln (i, j, k);
        m:= (i+j)/(k+1);
        Writeln ('Значение для m=', m)
    End.
```

2. Вычислить площадь треугольника по формуле Герона.

```
Program example_2_2;
```

```
Var s, a, b, c, p: Real;
Begin
    Write (' Введите значения a, b, c =>:');
    Readln (a, b, c);
    p:= (a + b + c) / 2;
     s: = sqrt(p × (p – a) × (p – b) × (p – c));
     Writeln ('s=', s)
End.
```

3. Вычислить расстояние между двумя точками с координатами (x1, y1) и (x2, y2).

```
Program example_2_3;
    Var x1, y1, x2, y2, c: Real;
    Begin
       Write (' Введите значения x1, y1, x2, y2 =>:');
       Readln (x1, y1, x2, y2);
        c:= sqrt ((x2 – x1) × (x2 – x1) + (y2 – y1) × (y2 – y1));
        Writeln ('c=', c)
    End.
```

Задачи для самостоятельной работы

1. Дан радиус окружности, подсчитать длину окружности.
2. Дан радиус окружности, подсчитать площадь круга.
3. Дан прямоугольный треугольник с катетами a и b. Найти гипотенузу c.
4. Дан произвольный треугольник. Известны стороны a и b и угол между ними. Найти третью сторону c.
5. Дан произвольный треугольник со сторонами a, b и c. Найти площадь треугольника.
6. Вычислить объём шара радиуса R.
7. Найти среднее арифметическое и среднее геометрическое трёх заданных чисел.
8. Найти расстояние между двумя точками с данными координатами.
9. По ребру найти площадь грани, площадь боковой поверхности и объём куба.
10. Вычислить периметр и площадь правильного 10-угольника, вписанного в окружность заданного радиуса.
11. Для заданного целого числа a напечатать следующую таблицу:
    a
    a^3 a^6
    a^6 a^3 a

12. Даны два действительных числа a и b. Получить их сумму, разность и произведение.
13. Дана длина ребра куба. Найти объём куба и площадь его боковой поверхности.
14. Даны два действительных положительных числа. Найти среднеарифметическое и среднегеометрическое этих чисел (или их модулей).
15. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
16. Смешано V_1 литров воды температуры t_1 и V_2 литров воды температуры t_2. Найти объём и температуру образовавшейся смеси.
17. Определить периметр правильного n-угольника, описанного около окружности радиуса r.
18. Дана сторона равностороннего треугольника. Найти площадь этого треугольника.
19. Даны гипотенуза и катет прямоугольного треугольника. Найти катет и радиус вписанной окружности.
20. Найти сумму членов арифметической прогрессии по данным значениям: a, d, n.
21. Треугольник задан длинами сторон. Найти:
    а) длины высот;
    б) длины биссектрис;
    в) длины медиан;
    г) радиусы вписанной и описанной окружностей.
22. Вычислить расстояние между двумя точками с координатами x_1, y_1 и x_2, y_2.
23. Даны целые числа k, m, действительные числа x, y, z. При k<m2 или заменить модулем соответственно значения x, y или z, а два других уменьшить на 0,5.
24. Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.
25. Дано действительное число x. Получить целую часть x; затем число x, округлённое до ближайшего целого; затем число без дробных цифр.
26. Даны действительные числа x, y. Вычислить расстояние от точки плоскости с координатами (x, y) до границы квадрата с вершинами:
    а) (–0.5, –0.5), (–0.5, 0.5), (0.5, 0.5), (0.5, –0.5);
    б) (0, 0), (0, 1), (1, 1), (1, 0).
    Имеется в виду минимум расстояний от данной точки до точек квадрата.

27. Даны целые (либо вещественные) числа x_1, y_1, x_2, y_2, x_3, y_3. Известно, что точки с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3) являются тремя вершинами некоторого прямоугольника. Найти координаты четвёртой вершины.

Практическая работа № 3

Условный оператор. Оператор многозначного ветвления

При описании разветвляющихся процессов обычно используют понятие условного и безусловного перехода. Если в программе требуется нарушить порядок выполнения операторов без предварительных проверок каких-либо условий, переход называется безусловным. Для реализации такого перехода служит оператор **goto n** (n — метка). В Паскале метка должна быть описана в разделе **label**, например:

label m, metka, 123

Однако современный стиль программирования предполагает как можно более редкое применение безусловного перехода, а еще лучше — полный отказ от него.

Ветвление — это выбор и выполнение одной из двух последовательностей инструкций в зависимости от ответа на заданный вопрос.

Условный оператор ***IF*** предназначен для изменения порядок выполнения операторов в зависимости от истинности или ложности некоторого условия. Он предписывает выполнять некоторое действие только в том случае, когда выполняется заданное условие. Это условие записывается в виде логического выражения, а действие, которое нужно выполнить, задается в виде последовательности операторов.

Выполнение условного оператора начинается с вычисления значения логического выражения, записанного в условии. Простые условия записываются в виде равенств и неравенств. Сложные условия составляют из простых с помощью логических операций:

And — логическое «и»,

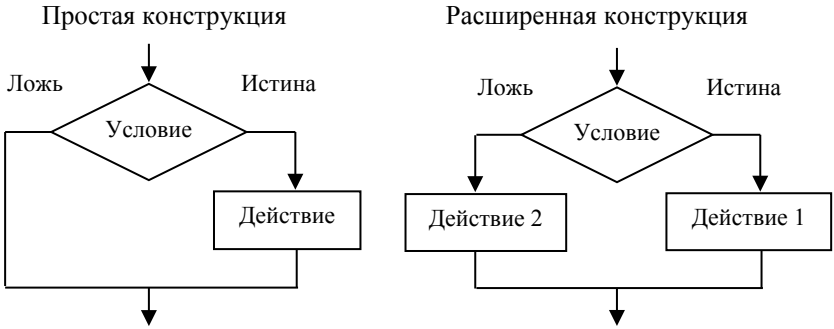
Or — логическое «или»,

Not — логическое «не».

В случае использования логических связок простые условия необходимо заключать в скобки. Например, условие $0 < a < 10$ записывается в следующем виде: $(0 < a)$ and $(a < 10)$.

Вещественные числа не рекомендуется сравнивать на точное равенство, т. к. они представлены в памяти приближенно.

Существует две конструкции оператора ветвления — простая и расширенная:



Укороченная развилка (краткая форма оператора)

If <условие> **then** <*оператор1*>;
<*оператор2*>;
<*оператор3*>;
...
<*операторN*>;

Если логическое выражение ***истинно***, то выполняется <*оператор1*>, в противном случае программа переходит к выполнению операторов, следующих за оператором ***if*** (<*оператор2*>, <*оператор3*>,..., <*операторN*>).

Если условие ***ложно***, то **не выполняется** <*оператор1*>, а сразу выполняется следующие за ним операторы <*оператор2*>, <*оператор3*>,..., <*операторN*>.

Полная развилка (полная форма оператора)

If <условие> **then** <*оператор1*>
else <*оператор2*>;
<*оператор3*>;
...
<*операторN*>;

Если логическое выражение ***истинно***, то выполняется <*оператор1*>, затем последовательность <*оператор3*>,..., <*операторN*>.

Если условие ***ложно***, то выполняется <*оператор2*>, затем выполняется следующие по порядку операторы <*оператор3*>,..., <*операторN*>.

Любая встретившаяся часть **ELSE** соответствует ближайшей к ней «сверху» части **THEN** условного оператора. Перед **ELSE** не ставится точка с запятой, т. к. это один оператор *if … then … else…;*.

Если за **ELSE** и **THEN** необходимо выполнить группу операторов, то тогда используются операторные скобки **begin** … **end**. Такая структура называется вложенной.

```
If <условие> then begin
                    <оператор1>;
                    <оператор2>;
                    …
                    <операторN>;
                end
               else begin
                    <оператор1>;
                    <оператор2>;
                    …
                    <операторN>;
                end
```

Примеры

1. Даны действительные числа x и y. Получить max(x,y).

```
Program example_3_1;
 Var x,y:real;
 Begin
     Write ('Введите x и y');
     Readln (x,y);
     If x>y then m:=x else m:=y;
     Writeln ('Максимальное m=', m)
  End.
```

2. Выбрать наименьшее значение из трех различных чисел.

Способ 1.

```
Program example_3_2;
 Var a, b, c, min :real;
 Begin
     Write ('Введите a, b, c');
     Readln (a, b, c);
     min := a;
     If b < min then min := b;
     If c < min then min := c;
```

```
    Writeln ('min = ',min)
End.
```

Способ 2.

```
Program example_3_3;
 Var a, b, c, min :real;
 Begin
    Write ('Введите a, b, c');
    Readln (a, b, c);
    if (a < b) and (a < c) then min := a
                           else if b < c then min := b
                                         else min := c;
    Writeln ('min = ',min)
End.
```

3. Составить программу для вычисления значения Y по заданному значению X,

Если y= $\begin{cases} 210, \text{при } x \le 10; \\ x^3, \text{при } x \ge 10. \end{cases}$.

```
Program example_3_4;
 Var x, y :real;
 Begin
    Write ('Введите x = ');
    Readln (y);
    min := a;
    If x <= 10 then y := 210 else y := x × x × x;
    Writeln ('при x = ',x, ' y = ', y)
 End.
```

4. Решить квадратное уравнение вида $ax^2 + bx + c = 0$.

```
Program example_3_5;
 Var a, b, c, d, x1, x2 :real;
 Begin
    Write('Введите коэффициенты a, b, c : ');
    Readln(a, b, c);
    d := b×b – 4×a×c;
    If d > 0 then begin
                        x1 := (–b + sqrt(d)) / (2×a);
                        x2 := (–b – sqrt(d)) / (2×a);
                  Writeln ('x1 = ',x1, ' x2 = ', x2)
               end
          else
```

```
            if d= 0 then begin
                        x := –b / (2×a);
                        Writeln ('x = ',x)
                      end
            else Writeln ('уравнение не имеет корней')
  End.
```

4. Проверить, является ли число точным квадратом.

```
 Program example_3_6;
Var a : integer;
Begin
    Write ('Введите a = ');
    Readln (a);
    if sqr ( round ( sqrt (a) ) ) = a
    then Writeln (a, '— точный квадрат')
    else Writeln (a, '— неточный квадрат')
End.
```

Оператор выбора *CASE*

В случае необходимости разветвить вычислительный процесс в зависимости от выполнения или невыполнения того или иного условия на более чем две ветви используется оператор выбора (случая, селектора, переключателя). Его использование оказывается более удобным по сравнению с использованием оператора if, если необходимо осуществить проверку более сложных условий, чем ДА/НЕТ. Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметр, по которому осуществляется выбор, служит *ключ выбора* — выражение любого порядкового типа.

```
Case S of
    C1: <Оператор1> ;
    C2: <Оператор2> ;
     . . .
    CN: <ОператорN>
    Else <Оператор>
End;
```

S — выражение порядкового типа значение которого вычисляется;

C1, C2, …, CN — константы, с которыми сравнивается значение выражения S;

Если выражение выбора отвечает условиям списка выражений блока CASE, выполняются операторы из этого блока.

<Оператор1>, <Оператор2>, <Оператор N> — операторы — *Список варианта*, из которых выполняется тот, с константой которого совпадает значение выражения S. Ветвь оператора *else* является необязательной. Если она отсутствует, и значение выражения S не совпадает ни с одной константой, весь оператор рассматривается как пустой.

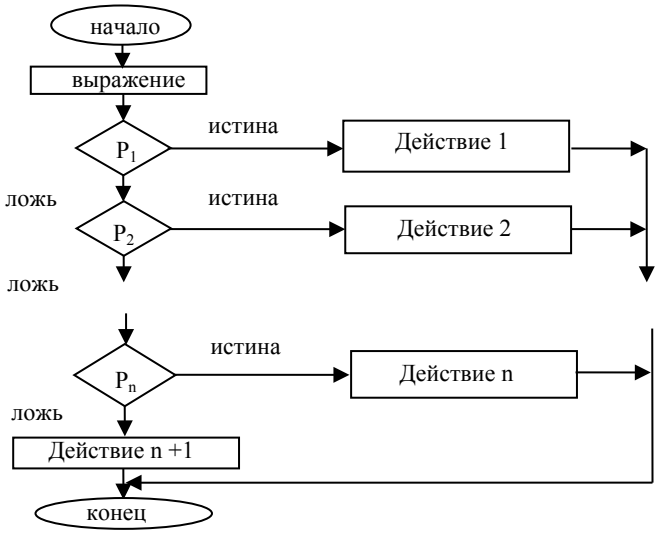
Если для нескольких констант нужно выполнить один и тот же оператор, их можно перечислить через запятую, сопроводив их одним оператором.

Оператор выбора работает следующим образом. В начале вычисляется значение выражения *Ключ выбора*, затем в последовательности операторов *Список варианта* отыскивается такой, которому предшествует константа, равная вычисленному значению. Найденный оператор выполняется, после чего оператор выбора завершает свою работу. Если в *Списке варианта* не будет найдена константа, соответствующая вычисленному значению *Ключа выбора*, управление передается оператору, стоящему за словом **else**.

Часть **else** <оператор> можно не использовать. Тогда при отсутствии в списке выбора нужной константы ничего не произойдет, и оператор выбора завершит свою работу.

Любому из операторов списка выбора может предшествовать не одна, а несколько констант выбора, разделенных запятыми.

Схематически такую конструкцию можно изобразить следующим образом:



Этот оператор представляет собой обобщение условного оператора:

If <условие1> **then** <оператор1> **else**
  If <условие2> **then** <оператор2> **else**
  ……..
        If < условиеN> **then** <операторN>;

Примеры

1. Ввести число от 1 до 100, если введенное число попадет в диапазон [1...10], определить его четность.

```
Program example_3_7;
 Var i: integer;
 Begin
      Write ('Введите число');
      Readln ( i );
      Case i of
            2, 4, 6, 8: Writeln (‘Четная цифра’);
            1, 3, 5, 7, 9: Writeln (‘Нечетная цифра’);
            10..100: Writeln (‘Число от 10 до 100’);
             Else Writeln (‘Отрицательное число или больше 100’)
       end;
 end.
```

2. Вывести текст в зависимости от введенного числа.

```
Program example_3_8;
 Var N: integer;
 Begin
      Write ('Введите число');
      Readln ( N );
      Case N of
           –32768 .. 0 : Writeln(‘<= 0’);
           2 .. 9 : Writeln(‘2 – 9’);
           1, 10 : Writeln(‘1, 10’);
           Else Writeln (‘> 10’)
      end;
 end.
```

3. Программа вводит два числа в первой строке и один из знаков +, –, ×, / во второй строке. Выводится результат соответствующего арифметического действия.

```
Program example_3_9;
  Var operation : char; x, y, z : real;
  Begin
       Write ('Введите числа');
       Readln ( x, y);
       Write ('Введите операцию');
       Readln ( operation);
       Case operation of
            '+' : Writeln(a,' + ',b,' = ',a + b);
            '–' : Writeln(a,' – ',b,' = ',a – b);
            '×' : Writeln(a,' × ',b,' = ',a × b);
            '/' : if b = 0 then Writeln ('Error : b = 0')
                           else Writeln (a,' / ',b,' = ',a / b);
       end;
  end.
```

Оператор безусловного перехода

Формат оператора перехода языка Pascal:

Goto <метка>;

Метка — это произвольный идентификатор, позволяющий именовать некоторый оператор программы, и таким образом ссылаться на него. Метка располагается непосредственно перед помеченным оператором и отделяется от него двоеточием.

Действие оператора безусловного оператора состоит в передаче управления соответствующему помеченному оператору. Правила при использовании меток:

— метка, на которую ссылается **Goto**, должна быть описана в разделе описаний;

— метки, описанные в процедуре (функции), локализуются в ней, поэтому передача управления извне процедуры (функции) на метку внутри нее невозможна.

Примеры

1. Алгоритм Евклида.

```
Program example_3_10;
Label 1, 2;
  Var x, y, m, n, c : integer;
  Begin
       Writeln (введите числа m, n =>);
       Readln (m, n);
```

```
x := m;
y := n;
1 : If x = y then goto 2;
    If x > y then x := x + y
            else y: = y – x;
    goto 1;
2 : c := x;
    Writeln (‘НОД двух чисел ’, m, ‘ и ’, n, ‘ = ’, c)
End.
```

Список задач

1. Даны действительные числа x, y. Получить:
   а) max (x,y),
   б) min (x,y),
   в) max (x,y), min (x,y).
2. Даны действительные числа x, y, z. Вычислить:
   а) max (x + y + z, x · y · z),
   б) $\min^2 (x + y + z/2, x \cdot y \cdot z) + 1$.
3. Даны действительные числа a, b, c. Проверить выполняется ли неравенство a < b < c.
4. Найти min значение из трёх величин, определяемых арифметическими выражениями
   a = sin (x), b = cos (x), c = ln (x) при заданных значениях x.
5. Даны действительные числа a, b, c. Удвоить эти числа, если a > b > c и заменить их абсолютными значениями, если это не так.
6. Даны два действительных числа. Заменить первое число нулём, если оно меньше или равно второму, и оставить числа без изменения иначе.
7. Даны действительные числа x, y. Меньшее из этих двух чисел заменить их полусуммой, а большее — их удвоенным произведением.
8. Даны три целых числа, найти среднее из них. Средним назовем число, которое больше наименьшего из данных чисел, но меньше наибольшего.
9. Подсчитать сумму только положительных из трех данных чисел.
10. Даны три числа. Подсчитать количество чисел, равных нулю.
11. Найти произведение двух наибольших из трех введенных с клавиатуры чисел.