**Программирование на языке Паскаль**

***Структура программы на Турбо Паскаль***

Программа на языке Паскаль состоит из заголовка, разделов описаний и раздела операторов. Заголовок программы содержит имя программы, например:

***Program PRIM;***

Описания могут включать в себя:

* раздел подключаемых библиотек (модулей);
* раздел описания меток;
* раздел описания констант;
* раздел описания типов;
* раздел описания переменных;
* раздел описания процедур и функций.

**Раздел описания модулей** определяется служебным словом **USES** и содержит имена подключаемых модулей (библиотек) как входящих в состав системы Turbo Pascal, так и написанных пользователем. Раздел описания модулей должен быть первым среди разделов описаний. Имена модулей отделяются друг от друга запятыми:

***uses CRT, Graph;***

Любой оператор в программе может быть помечен меткой. Имя метки задается по правилам образования идентификаторов Турбо Паскаль. В качестве метки также могут использоваться произвольные целые числа без знака, содержащие не более четырех цифр. Метка ставится перед оператором и отделяется от него двоеточием. Все метки, используемые в программе, должны быть перечислены в **разделе описания меток**, например:

***label 3, 471, 29, Quit;***

Описание констант позволяет использовать имена как синонимы констант, их необходимо определить в **разделе описания констант**:

***const K= 1024; MAX= 16384;***

В **разделе описания переменных** необходимо указать все переменные, используемые в программе, и определить их тип:

***var P,Q,R: Integer;***

***A,B: Char;***

***F1,F2: Boolean;***

Описание типов, процедур и функций будет рассмотрено ниже. Отдельные разделы описаний могут отсутствовать, но следует помнить, что в Паскаль - программе должны быть обязательно описаны все компоненты программы.

**Раздел операторов** представляет собой ***составной оператор***, который содержит между служебными словами

***begin.......end***

последовательность операторов. Операторы отделяются друг от друга символом **;**. Текст программы заканчивается символом ***точка.***

Кроме описаний и операторов Паскаль - программа может содержать комментарии, которые представляют собой произвольную последовательность символов, расположенную между открывающей скобкой комментариев { и закрывающей скобкой комментариев }.

***Пример 1***

Данная программа запрашивает с клавиатуры два числа, находит их сумму и выводит ответ.

***program Primer; {вычисление суммы двух чисел}***

***var***

***x,y,s: integer;***

***begin***

***WriteLn('Введите через пробел два числа ');***

***ReadLn(x,y);***

***s := x + y;***

***WriteLn('Сумма чисел равна ',s);***

***end.***

***Константы***

***Константа*** – это неизменяемая величина. И конечно, это совершенно не обязательно числовая величина.

***Объявление константы***

Объявляется константа в блоке объявлений следующим образом:

***const***

*a=79;*

*d=4.5;*

*s=’hh’;*

Константа ***обязательно*** имеет значение. Нельзя включить в объявление константы переменные величины, так как их значение в блоке объявлений еще не определено. Объявление вполне может содержать выражения, в которых также будут присутствовать уже объявленные ранее константы.

***const***

*a=’fgfgfg’;*

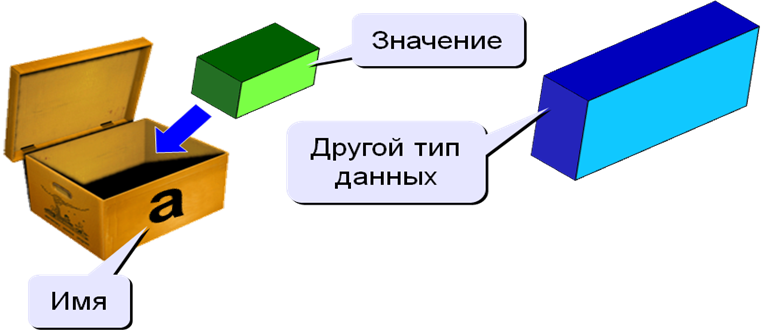
*s=a+’dsd’;*

*w=5.6;*

*q=w+7;*

***Переменные***

***Переменная*** – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.



**В именах переменных МОЖНО использовать**

* + латинские буквы (A-Z) (заглавные и строчные буквы не различаются);
  + цифры (имя не может начинаться с цифры);
  + знак подчеркивания **\_;**

**В именах НЕЛЬЗЯ использовать**

* + русские буквы
  + пробелы
  + скобки, знаки +, =, !, ? и др.

***Объявление переменных:***

***тип - целые***

**Выделение места в памяти**

***Var***iable-переменная

**var a, b, c : integer;**

***Список имен переменных***

***Типы переменных:***

Для обработки ЭВМ данные представляются в виде величин и их совокупностей. С понятием величины связаны такая важная характеристика, как ее тип.

***Тип определяет:***

* возможные значения переменных, констант, функций, выражений, принадлежащих к данному типу;
* внутреннюю форму представления данных в ЭВМ;
* операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.



В языке Паскаль тип величины задают заранее. Все переменные, используемые в программе, должны быть объявлены в разделе описания с указанием их типа. Обязательное описание типа приводит к избыточности в тексте программ, но такая избыточность является важным вспомогательным средством разработки программ и рассматривается как необходимое свойство современных алгоритмических языков высокого уровня.

***Иерархия типов в языке Паскаль такая:***

* ***Простые*** 
  + *Порядковые* 
    - Целые
    - Логические
    - Символьные
    - Перечисляемые
    - Интервальные
  + *Вещественные*
* ***Структуированные*** 
  + Массивы
  + Строки
  + Множества
  + Записи
  + Файлы
* ***Указатели***

***Простые типы данных***

В таблице приведены простые типы данных Турбо Паскаль, объем памяти, необходимый для хранения одной переменной указанного типа, множество допустимых значений и применимые операции.

|  |  |  |  |
| --- | --- | --- | --- |
| **Идентификатор** | **Длина (байт)** | **Диапазон значений** | **Операции** |
| **Целые типы** | | | |
| ***integer*** | 2 | -32768..32767 | +, -, /, \*, Div, Mod, >=, <=, =, <>, <, > |
| ***byte*** | 1 | 0..255 | +, -, /, \*, Div, Mod, >=, <=, =, <>, <, > |
| ***word*** | 2 | 0..65535 | +, -, /, \*, Div, Mod, >=, <=, =, <>, <, > |
| ***shortint*** | 1 | -128..127 | +, -, /, \*, Div, Mod, >=, <=, =, <>, <, > |
| ***longint*** | 4 | -2147483648..2147483647 | +, -, /, \*, Div, Mod, >=, <=, =, <>, <, > |
| **Вещественные типы** | | | |
| ***real*** | 6 | 2,9x10-39 - 1,7x1038 | +, -, /, \*, >=, <=, =, <>, <, > |
| ***single*** | 4 | 1,5x10-45 - 3,4x1038 | +, -, /, \*, >=, <=, =, <>, <, > |
| ***double*** | 8 | 5x10-324 - 1,7x10308 | +, -, /, \*, >=, <=, =, <>, <, > |
| ***extended*** | 10 | 3,4x10-4932 - 1,1x104932 | +, -, /, \*, >=, <=, =, <>, <, > |
| **Логический тип** | | | |
| ***boolean*** | 1 | true, false | Not, And, Or, Xor, >=, <=, =, <>, <, > |
| **Символьный тип** | | | |
| ***char*** | 1 | все символы кода ASCII | +, >=, <=, =, <>, <, > |

## *Структурированные типы данных*

В Турбо Паскале пять структуированных типов:

* *массивы;*
* *строки;*
* *множества;*
* *записи;*
* *файлы;*

### *Перечисляемый тип данных*

Перечисляемый тип задаётся перечислением тех значений, которые он может получать. Каждое значение именуется некоторым идентификатором и располагается в списке, обрамлённом круглыми скобками. Применение перечисляемых типов делает программы нагляднее. Например:

**TYPE**

**DAYS = (MONDAY, TUESDAY, WEDNESDAY);**

**VAR**

**DAY : DAYS;**

**BEGIN**

**…**

**IF DAY = MONDAY THEN WRITELN ('КАКОЙ ТЯЖЁЛЫЙ ДЕНЬ.');**

**…**

**END.**

Переменные любого перечисляемого типа можно объявить без предварительного описания типа. Например:

**VAR**

**COL : (BLACK, WHITE, RED);**

### *Интервальный тип данных*

***Отрезок (диапазон значений)*** любого порядкового типа может быть определен как ***интервальный (ограниченный) тип***. Отрезок задается диапазоном от минимального до максимального значения констант, разделенных двумя точками. В качестве констант могут быть использованы константы, принадлежащие к целому, символьному, логическому или перечисляемому типам.

***Примеры отрезков:***

***1..10***

***-15..25***

***'a'..'z'***

Минимальное и максимальное значения констант называются нижней и верхней границами отрезка, определяющего интервальный тип. Нижняя граница должна быть меньше верхней.

Над переменными, относящимися к интервальному типу, могут выполняться все операции и применяться все стандартные функции, которые допустимы для соответствующего базового типа.

***Как записать значение в переменную?***



**а := 5;**



***Оператор присваивания***

**При записи нового значения старое стирается!**

**Оператор** – это команда языка программирования инструкция).

**Оператор присваивания** – это команда для записи нового значения в переменную.

***Математические операции***

В таблице приведены основные математические операции Турбо Паскаль.

|  |  |  |
| --- | --- | --- |
| **Символ операции** | **Название операции** | **Пример** |
| **\*** | умножение | 2\*3 (результат: 6) |
| **/** | деление | 30/2 (результат: 1.5E+01) |
| **+** | сложение | 2+3 (результат: 5) |
| **-** | вычитание | 5-3 (результат: 2) |
| **div** | целочисленное деление | 5 div 2 (результат: 2) |
| **mod** | остаток от деления | 5 mod 2 (результат: 1) |

***Логические операции***

Над логическими аргументами в Турбо Паскаль определены следующие операции:

* ***NOT*** - логическое отрицание ("НЕ")
* ***AND*** - логическое умножение ("И")
* ***OR*** - логическое сложение ("ИЛИ")
* ***XOR*** - логическое "Исключающее ИЛИ"

Результаты выполнения этих операций над переменными А и В логического типа приведены в таблице истинности.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **not A** | **A and B** | **A or B** | **A xor B** |
| true | true | false | true | true | false |
| true | false |  | false | true | true |
| false | true | true | false | true | true |
| false | false |  | false | false | false |

***Операции отношения***

К операциям отношения в Турбо Паскаль относятся такие операции, как:

* **>** - больше
* **<** - меньше
* **=** - равно
* **<>** - не равно
* **>=** - больше или равно
* **<=** - меньше или равно

В операциях отношения могут принимать участие не только числа, но и символы, строки, множества и указатели.

***Приоритет операций***

Порядок вычисления выражения определяется старшинством (приоритетом) содержащихся в нем операций. В языке Паскаль принят следующий приоритет операций:

* унарная операция not, унарный минус -, взятие адреса @
* операции типа умножения:  **\*  /  div  mod  and**
* операции типа сложения:  ***+  -  or  xor***
* операции отношения:  ***=  <>  <  >  <=  >=  in***

Порядок выполнения операций переопределить можно с помощью скобок. Например, 2\*5+10 равно 20, но 2\*(5+10) равно 30.

***Основные математические функции***

В этом разделе приведены основные математические функции, встроенные в системную библиотеку Турбо Паскаль.

***Abs(X)***  Возвращает абсолютное значение числа X.

***Cos(X)*** Возвращает косинус числа X, где X - угол в радианах

***Sin(X)*** Возвращает синус числа X, где X - угол в радианах

***ArcTan(X)*** Возвращает арктангенc числа X.

***Exp(X)*** Возвращает число, равное eX.

***Ln(x)*** Возвращает число, равное натуральному логарифму от числа X.

***Pi***  Число Пи.

***Sqr(X)*** Возвращает число, равное квадрату числа X.

Функций **тангенс** и **котангенс** в Турбо Паскале нет. Для их вычисления используйте выражение sin(x)/cos(x) (или cos(x)/sin(x) для котангенса). Функции возведения в произвольную степень в Турбо Паскале нет. Используйте многократное умножение для возведения в целочисленную степень, либо функции **Exp** и **Ln** для возведения в вещественную степень: .



*Примеры:*

|  |  |
| --- | --- |
| Выражение | Запись на Паскале |
| X3 | X\*X\*X |
| X3,5 | Exp(3.5\*Ln(X)) |
| U2x | Exp(2\*x\*Ln(U)) |

***Sqrt(X)*** Возвращает число, равное квадратному корню из числа X.

***Trunc(X)***  Возвращает число, равное целой части числа X.

(Происходит отбрасывание дробной части числа X.

Результат выполнения имеет тип Longint).

***Frac(X)*** Возвращает число, равное дробной части числа X.

***Int(X)*** Возвращает число, равное целой части числа X.

Результат выполнения функции - real.

***Round(X)*** Функция округляет число X. Возвращаемое значение имеет тип Longint.

***Random(X)*** Возвращает случайное целое число в диапазоне 0..X.

***Random***  Если аргумент опущен, то возвращается случайное вещественное число из интервала (0; 1).

Перед использованием random в программах рекомендуется сначала инициализировать генератор псевдослучайных чисел процедурой Randomize. В противном случае при каждом запуске программы будет генерироваться одна и та же последовательность случайных чисел.

*Пример. Вывод на экран 5 случайных чисел в диапазоне -10..10.*

***var i: integer;***

***begin***

***randomize;***

***for i:=1 to 5 do writeln(random(21)-10);***

***end.***

***Inc(X,Y)*** Увеличивает значение числа X на Y. Если число Y не указано, то увеличение происходит на 1.

***Dec(X,Y)*** Уменьшает значение числа X на Y. Если число Y не указано, то уменьшение происходит на 1.

**Операторы ввода и вывода**

***Ввод данных***

Процедура производит последовательное чтение К значений исходных данных и присваивает эти значения переменным А1, А2, ..., АК.

***Read (A1, A2,...AK);***

В этом случае после ввода данных происходит перевод курсора на следующую строку:

***ReadLn (A1, A2,...AK);***

Пустой оператор ReadLn позволяет приостановить работу программы до нажатия Enter:

***ReadLn;***

Значения исходных данных могут отделяться друг от друга пробелами и нажатием клавиш табуляции и Enter.

***Не допускается разделение вводимых чисел запятыми!***

Например: **Read(a,b);** - *ввод значений двух переменных*

***Вывод данных***

Для вывода результатов работы программы на экран используются процедуры

Вывод значений переменных А1, А2,..., АК в строку экрана:

***Write (A1,A2,...AK);***

После вывода данных на экран, курсор перейдет на новую строку:

***WriteLn (A1,A2,...AK);***

При выводе вещественных чисел необходимо форматирование:

***WriteLn (A:К:М);***

где ***А*** - переменная или выражение действительного типа, ***К*** - ширина поля вывода, ***М*** - число цифр дробной части выводимого значения. К и М - выражения или константы целого типа. В этом случае действительные значения выводятся в форме десятичного числа с фиксированной точкой.

**Например:**

**Write(a); {** *вывод значения переменной* ***а* }**

**Writeln(a); {** *вывод значения переменной* ***а***  *и переход на новую строку***}**

**Write(‘Hello!’); {** *вывод текста***}**

**Write(‘c=’,c); {** *вывод текста и значения переменной* ***с* }**

**Оператор условного перехода**

**Задача:** Ввести два целых числа и вывести на экран наибольшее из них.

**Идея решения:** надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

**Особенность:** действия исполнителя зависят от некоторых условий

(*если … иначе …*).

*Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются* ***разветвляющимися.***

да

нет

**a, b**

**a>b**

**max:=b**

**max:=a**

**Вывод max**

**Полная форма условного оператора**

**if** *условие* **then** *оператор 1* **else** *оператор 2*;

**Блок «начало»**

**Блок «ввода»**

**Блок «решение»**

**program qq;**

**var a, b, max: integer;**

**begin**

**writeln('Введите два целых числа');**

**read ( a, b );**

**if a > b then max:=a**

**else**

**max:=b;**

**writeln ('Наибольшее число ', max);**

**end.**

**Полная форма ветвления**

**Блок «вывода»**

**Блок «конец»**

***Условие*** - это логическое выражение, в зависимости от которого выбирается одна из двух альтернативных ветвей алгоритма. Если значение условия истинно (TRUE), то будет выполняться ***оператор 1*,** записанный после ключевого слова ***then***. В противном случае будет выполнен ***оператор 2***, следующий за словом ***else***, при этом *оператор 1* пропускается. После выполнения указанных операторов программа переходит к выполнению команды, стоящей непосредственно после оператора ***if.***

**Необходимо помнить, что перед ключевым словом else точка с запятой никогда не ставится!**

Следует помнить, что синтаксис языка допускает запись **только одного оператора** после ключевых слов then и else, поэтому группу инструкций обязательно надо объединять в **составной оператор** (***заключать в операторные скобки begin ... end***). В противном случае возникает чаще всего логическая ошибка программы, когда компилятор языка ошибок не выдает, но программа тем не менее работает неправильно.

**program qq;**

**var a, b, max: integer;**

**begin**

**writeln('Введите два целых числа');**

**read ( a, b );**

**if a > b then max:=a;**

**writeln ('Наибольшее число ', max);**

**end.**

**Неполная форма условного оператора**

**if** *условие* **then** *оператор 1*;

**a, b**

**a>b**

**max:=a**

**Вывод max**

**Оператор выбора**

Часто возникают ситуации, когда приходится осуществлять выбор одного из нескольких альтернативных путей выполнения программы. Несмотря на то, что такой выбор можно организовать с помощью оператора if .. then, удобнее воспользоваться специальным оператором выбора. Его формат:

**case *выражение* of**

***вариант* : *оператор*;**

**...**

***вариант* : *оператор*;**

**end;**

***или***

**case *выражение* of**

***вариант* : *оператор*;**

**...**

***вариант* : *оператор*;**

**else *оператор***

**end;**

*выражение*, которое записывается после ключевого слова case, называется ***селектором***, оно может быть любого перечисляемого типа. *вариант* состоит из одной или большего количества констант или диапазонов, разделенных запятыми. Они должны принадлежать к тому же типу, что и селектор, причем недопустимо более одного упоминания *варианта* в записи инструкции case. Из перечисленного множества *операторов* будет выбран только тот, перед которым записан *вариант*, совпадающий со значением селектора. Если такого *варианта* нет, выполняется *оператор*, следующий за словом else (если он есть).

***Пример***

**case ch of**

**'A'..'Z', 'a'..'z' : WriteLn('Буква');**

**'0'..'9' : WriteLn('Цифра');**

**'+', '-', '\*', '/' : WriteLn('Оператор');**

**else WriteLn('Специальный символ')**

**end;**