Знакомство с языком Паскаль

Общие сведения о языке Паскаль

Язык программирования Паскаль был разработан швейцарским ученым, профессором Никлаусом Виртом в 1971 г. Вирт назвал свой язык Паскалем в честь французского религиозного философа, математика и физика XVII века Блеза Паскаля. Давая название своему новому языку программирования, Вирт имел в виду прежде всего то обстоятельство, что Паскаль в юности изобрел механическое счетно-решающее устройство, которое назвал "Паскалиной". Первоначально язык Паскаль был задуман как средство для обучения программированию, но в дальнейшем – после расширения языка и разработки высокоэффективных компиляторов (один из них – Турбо-Паскаль) – он стал использоваться как язык для профессионального программирования. В настоящее время Паскаль является одним из наиболее распространенных языков программирования высокого уровня.

Одним из наиболее известных диалектов языка Паскаль, использующихся на IBM PC-совместимых компьютерах, является Турбо-Паскаль, разработанный французом Филиппом Каном в 1982 году.

- **Паскаль** - это универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации.

- Простейшее предложение языка программирования, которое может выполнить одно определенное законченное действие называется **оператором.**

- Чтобы написать программу нужно знать **синтаксис (правило записи элементов**) и **семантику (правила применения и смысла элементов).**

Структура программы на Паскале

I. Заголовок

Содержит лишь одну строку - **Program** <имя программы>

Пример заголовка: **program raschet;**

II. Раздел описания

- **Uses** - список библиотечных модулей
- **Label** - описание меток
- **Const –** описание постоянных
- **Type –** описание типов данных
- **Var -** описание переменных
- **Procedure –** описание процедур
- **Function** – описание функции

III. Раздел операторов

Все действия и команды располагаются в этой основной части. Это раздел должен начинаться с **Begin** и заканчиваться **End.**

begin
< операторы >
end.

Заголовок программы состоит из зарезервированного слова program и имени программы (со списком параметров, заключенных в круглые скобки). Завершается заголовок точкой с запятой.

Заголовок необязателен, но желателен, чтобы по нему можно было быстро распознать программу.

Пример заголовка: program raschet;

Раздел описания переменных начинается со слова **Var** (variables – переменные) за которым идет список имен переменных через запятую. Тип указывается после двоеточия. В стандарте языка Паскаль существуют два числовых типа величин: **вещественный** и **целый**. Слово **integer** обозначает целый тип (является идентификатором целого типа). Вещественный тип обозначается словом **real.**

Например:

```
const  pi:=3.14159
var a, b : integer; c, d : real;
```

Пунктуация Паскаля

- Необходимо строгое соблюдение правописания (синтаксиса) программы. В частности, в Паскале однозначно определено назначение знаков пунктуации.
- **Точка с запятой (;)** ставится в конце заголовка программы, в конце раздела описания переменных, является разделителем операторов. Перед словом end точку с запятой можно не ставить.
- **Запятая (,)** является разделителем элементов во всевозможных списках: списке переменных в разделе описания, списке вводимых и выводимых величин.
- Строгий синтаксис в языке программирования необходим потому, что компьютер является формальным исполнителем программы. Если, допустим, разделителем в списке переменных должна быть запятая, то любой другой знак будет восприниматься как ошибка.

- В программу на Паскале можно вставлять комментарии. **Комментарий** - это пояснение к программе, которое записывается в фигурных скобках. В комментариях можно использовать русские буквы. На исполнение программы комментарий никак не влияет.
- Заметим, что в Паскале нет различия между строчными и прописными буквами. Например, для Паскаля тождественны следующие варианты записи: **begin, Begin, BEGIN, BeGiN.** Использование строчных или прописных букв - дело вкуса программиста.

Ввод исходных данных с клавиатуры происходит с помощью оператора **read** (read - читать) или **readln** (read line - читать строку):

read(<список переменных>);

или **readln**(<список переменных>);

При выполнении команды ввода компьютер ожидает действий пользователя. Пользователь набирает на клавиатуре значения переменных в том порядке, в каком они указаны в списке, отделяя их друг от друга пробелами. В конце нажимается клавиша <ВВОД> (<Enter>). Разница в выполнении операторов **readln** и **read** состоит в том, что после выполнения ввода по оператору **readln** экранный курсор перемещается в начало новой строки, а по оператору **read** этого не происходит.

Вывод результатов происходит по оператору **write** (write - писать) или **writeln** (write line - писать в строку):

write(<список вывода>);

или **writeln**(<список вывода>);

Результаты выводятся на экран компьютера в порядке их перечисления в списке. Элементами списка вывода могут быть константы, переменные, выражения.

Разница в выполнении операторов **writeln** и **write** состоит в том, что после выполнения вывода по оператору **writeln** экранный курсор перемещается в начало новой строки, а по оператору **write** этого не происходит.

.Оператор условия

```
IF (< условие>) THEN
                    begin
                    < оператор 1 >;
                   < оператор 2 >;
                                    …
                end
                  ELSE
                      begin
                        < оператор 1 >;
                          < оператор 2 >;
                                          …
                      end;
```

Конструкция ИНАЧЕ (ELSE) может отсутствовать.

```
{ Программа для вычисления действительных корней квадратного уравнения, если они, конечно, имеются... }
program First ;
var
a,b,c : real; {переменные для хранения коэффициентов}
x1,x2 : real; {переменные для хранения корней уравнения}
D: real; {переменная для хранения дискриминанта}
begin
writeln ('Введите коэффициенты квадратного уравнения');
writeln;
write ('a = '); readln(a);
write ('b = '); readln(b);
write ('c = '); readln(c);
d := b*b-4*a*c; {вычислим значение дискриминанта}
if D < 0 then
writeln('Действительных корней нет')
else
if D = 0 then
writeln('x= ' , -b/(2*a):5:0)
else
begin
x1 := (–b+sqrt(D))/(2*a); {вычислим 1-й корень}
x2 := (–b–sqrt(D))/(2*a); {вычислим 2-й корень}
writeln('x1 = ', x1:3:0); {напечатаем 1-й корень}
writeln('x2 = ', x2:3:0); {напечатаем 2-й корень}
end;
end;
end. {В конце программы ставим точку!}
```