Процедуры и функции.

Писать большие программы как нечто единое целое, без ***расчленения*** на относительно самостоятельные фрагменты (блоки), т.е. без ***структурирования***, просто невозможно. Практически во всех языках программирования имеются средства ***структурирования***. Такие языки называются ***процедурно-ориентированными***.

Такой метод имеет преимущество по сравнению с «потоковой записью программы», которое заключается в следующем: каждый блок в программе существует в ***единственном*** экземпляре, в то время как обращаться к нему можно ***многократно*** из разных точек программы.

Процедурой называется наименованная последовательность инструкций. Упоминание этого ***имени*** в тексте программы приводит к активизации процедуры и называется её ***вызовом***. Сразу, после активизации процедуры, начинают выполняться входящие в ***неё*** операторы. После выполнения последнего из них управление возвращается обратно в основную программу, и выполняются операторы стоящие ***непосредственно*** за оператором вызова процедуры.

Функция отличается от процедуры тем, что результат её работы возвращается в виде ***значения*** этой функции, и, следовательно, вызов функции может использоваться наряду с другими операндами в выражениях.

Переменные, которые описаны в разделе описаний процедуры (функции), недоступны основной программе и другим подпрограммам. Но в процедуре (функции) доступны переменные, которые были описаны в основной программе. Перед вызовом, процедура (функция) должна быть описана.

Существуют три вида переменных:

- **глобальные** – переменные, описанные в основной программе (доступны всегда);
- **локальные** – переменные, описанные в подпрограммах (доступны только в текущей подпрограмме);
- **формальные** – переменные, которые передаются подпрограмме в виде параметров.

Форма записи процедуры:

```
Procedure <имя_процедуры>;
 <раздел описаний>
Begin
 <операторы>
End;
```

Пример:

```
Program Procedure_Demo;
Var
  I : Integer; {глобальная переменная}

 Procedure Povtor;
  Var
   C : Integer; {локальная переменная}
  Begin
   For C:=1 To I Do Writeln('Повторяю.');
  End;

BEGIN
 Write('Введите число повторений:');
 Readln(I);
 Povtor;
END.
```

Форма записи функции:

Функция – это аналогия процедуры, но она должна иметь тип, и в отличие от процедуры, которая только выполняет, функция выполняет и возвращает в своём имени какой-то результат, ранее описанного типа.

```
Function <имя_функции> : <тип>;
 <раздел описаний>
Begin
 <операторы>
End;
```

Внутри исполнительной части функции должен **обязательно** встречаться оператор **`<имя_функции>:=<выражение>;`**

Пример:

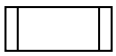
```
Program Function_Demo;
 Var
  C : Char;

Function Yes_No : Boolean;
 Begin
  Case C Of
   'y', 'Y', 'д', 'Д' : Yes_No:=True
   Else
     Yes_No:=False
   End;
 End;

BEGIN
 Write('Ты любишь играть? ');
 Readln(C);
 If Yes_No Then Writeln('Я тоже, иногда.')
  Else Writeln('Зря, однако.')
END.
```

Как видно из примера, имя функции выступает как бы в роли ***переменной*** логического типа, поэтому ей надо ***присваивать*** какое-то значение.

Оно выступает в роли переменной, к которой присваивается какое-то значение, но оно **не может** участвовать в «***правых***» частях выражений.

В блок-схеме алгоритма вызов подпрограммы или функции обозначается значком: 

Пример алгоритма с использованием подпрограммы: