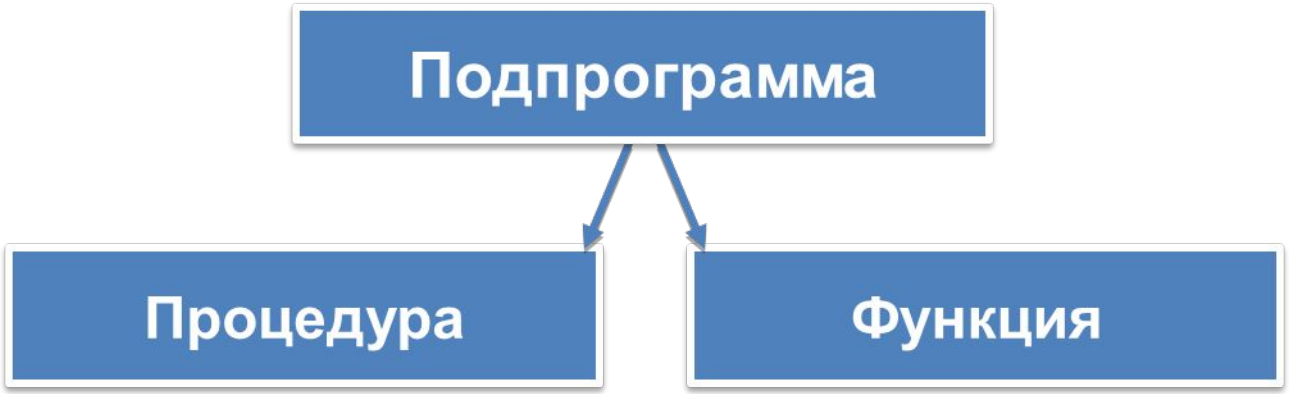
Процедуры и функции в Pascal’е

Вспомогательный алгоритм - подпрограмма

Подпрограммы применяются когда:

1. часть алгоритма неоднократно повторяется в программе;
2. можно использовать фрагменты разработанных ранее алгоритмов;
3. для разбиения крупных программ на части в соответствии с модульным принципом программирования.

В паскале реализовано два типа подпрограмм **процедуры** и **функции**.



Процедуры и функции

- **Процедура(функция)** представляет собой последовательность операторов, которая имеет **имя**, **список параметров** и может быть вызвана из различных частей программы.

- Имя процедуры в тексте программы называется ***вызовом.***

- ***Вызов активирует процедуру (функцию)*** - начинают выполняться её операторы.

- После выполнения процедуры программа продолжается с оператора стоящего за вызовом.

- Отличие ***процедур*** от ***функций*** в том, что функции возвращают значение.

Описание процедур и функций

Все процедуры или функции должны быть описаны в разделе описаний основной программы.

Описание процедуры имеет вид:

```
procedure имя (список формальных параметров);
    раздел описаний локальных параметров
    begin
      операторы тела процедуры
    end;
```

Описание функции имеет вид:

```
function имя (список формальных параметров): тип значения функции;
раздел описаний локальных параметров
begin
  операторы тела функции
end;
```

ОБЛАСТЬ ДЕЙСТВИЯ ПЕРЕМЕННЫХ ПРОЦЕДУР И ФУНКЦИЙ

- **Глобальные переменные** - переменные, описанные в основной программе и работающие во всей программе.

!!!!Обмен информацией между основной программой и подпрограммой может осуществляться только с помощью глобальных переменных.

- **Локальные переменные** – описаны в подпрограмме и существуют только в течении работы подпрограммы.
- **Параметры.** Переменные, с помощью которых осуществляется связь между основной программой и подпрограммами. Так передаются значения от основной программы к подпрограмме и наоборот.
- **Формальные параметры** – указываются при описании процедуры (функции). Каждый параметр является локальным по отношению к описываемой процедуре (функции), т.е. к нему можно обращаться только в пределах данной процедуры (функции).
- **Фактические параметры** – это конкретные значения формальных параметров, которые передаются при обращении к процедуре (функции).

!!!! Число и тип формальных и фактических параметров должны совпадать с точностью до их следования!

Параметры процедур и функций

Список ФОРМАЛЬНЫХ ПАРАМЕТРОВ состоит из одной или нескольких секций, разделенных символом " **;** ".

Секция состоит из списка переменных, перечисляемых через запятую, знака "**:**" и типа.

……..(X,Y:integer; S:real)

В ПРОЦЕДУРЕ секция может предваряться служебным словом **var - тогда параметры передаются по ссылке, (экономия памяти и времени)**

……..(X,Y:integer; var S:real)

Если **var** отсутствует параметры передаются значениями.

Список формальных параметров вместе с окружающими скобками может отсутствовать.

Раздел описаний локальных параметров

1. Раздел описаний процедуры или функции ***устроен так же, как и раздел описаний программы***.

2. Здесь описываются ***локальные переменные, константы*** *и* вложенные процедуры и функции.

3. Все такие локальные объекты доступны лишь внутри данной подпрограммы и не видны извне.

ПАРАМЕТРЫ ПРОЦЕДУРЫ

- **Параметры - значения** – Передача параметров по значению. Копия значения фактического параметра становится значением соответствующего формального параметра. При этом при изменении параметра в процедуре значение соответствующего параметра в основной программе не измениться.

- **Параметры - переменные** – в процедуру передается адрес фактического параметра. Любые операции с формальным параметром в процедуре приведут к изменению фактического параметра в основной программе. Таким образом, **процедура тоже может возвращать какие-то значения в основную программу или несколько значений.**

Решение задач с использованием подпрограмм
ВВОД ЭЛЕМЕНТОВ МАССИВА

ПРОЦЕДУРА :

```
{}
Procedure INP (N1:integer; VAR b:ZZ);  {N1- параметр-значение,
                                        b- параметр-переменная}
 Var k: integer;  {локальные переменные}
 begin
   Randomize;
   For k:=1 to N1 do
     b[k]:=random(100)-50;

End;
```

Основная программа

```
const N=10;
Type ZZ=array[1..N] of integer;   {описание
                                   глобального массива a}
Var a:ZZ; i: integer;
BEGIN
 INP (N,a);  {вызов процедуры с указанием
              фактических значений параметров}
For i:=1 to N do
 write (a[i]:4);
END.
```

Нахождение суммы элементов массива

ФУНКЦИЯ:

```
function SUM (N2:inferer; b:ZZ):integer;
 var k1,s: integer;
 begin
   s:=0;
   for k1:=1 to n do
    s:=s+b[k1];
   SUM:=s; {передача выч. значения имени
                                        функции}
 end;
```

Основная программа

```
const N=10;
Type ZZ=array[1..N] of integer;   {описание
                                   глобального массива a}
Var a:ZZ;
      i, sum1: integer;
BEGIN
 INP (N,a);
For i:=1 to N do
 write (a[i]:4);
 writeln (sum1(N,a)); {вывод суммы элементов массива}
END.
```

ДОМАШНЕЕ ЗАДАНИЕ:

СОСТАВИТЬ ПОДПРОГРАММЫ:

1. Нахождения минимального элемента
2. Сортировки массива