Реализация процедур и функций в Паскале

Паскаль относится к процедурным языкам программирования. Программы, написанные на процедурных языках, представляют собой последовательность операторов – инструкций, которые выполняются одна за другой по тексту программы. Для небольших программ при этом не требуется какой-то дополнительной организации, но если размер программы большой, то ее текст становится громоздким и неудобным для восприятия человеком - **неструктурированным**.

Для структурирования таких программ их разбивают на **отдельные блоки, которые называют *подпрограммами*.**

В небольших программах подпрограммы удобно использовать в тех случаях, когда одинаковые действия, выполняемые группой операторов над значениями некоторых переменных, требуется выполнить несколько раз в различных местах программы.

В этом случае использование подпрограмм дает возможность сократить текст программы, т.к. соответствующая последовательность действий записывается один раз, а вызывается несколько раз

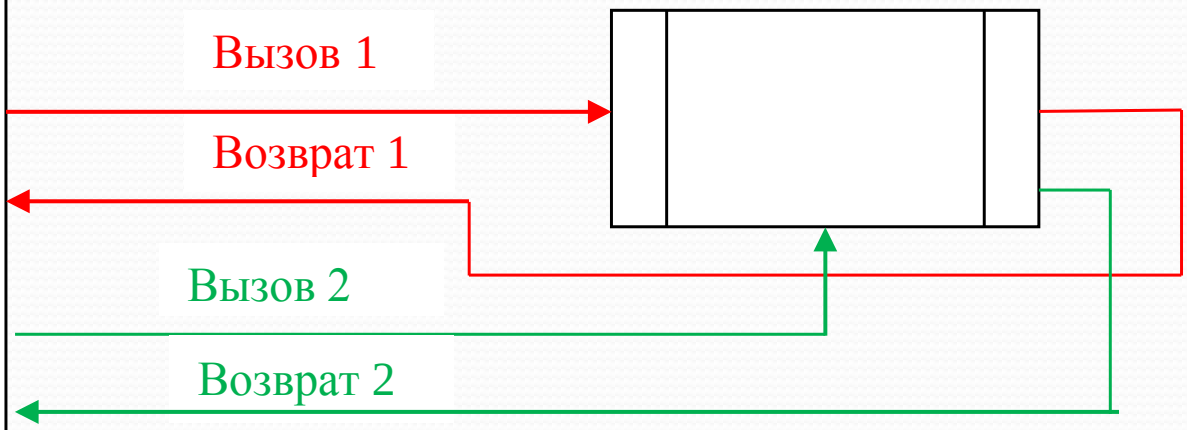
Взаимодействие основной программы и подпрограммы

Основная программа

Подпрограмма

оператор 1
оператор 2
вызов подпрограммы
оператор 3

вызов подпрограммы
оператор 4



В языке Паскаль различают **два вида подпрограмм: процедуры и функции.**

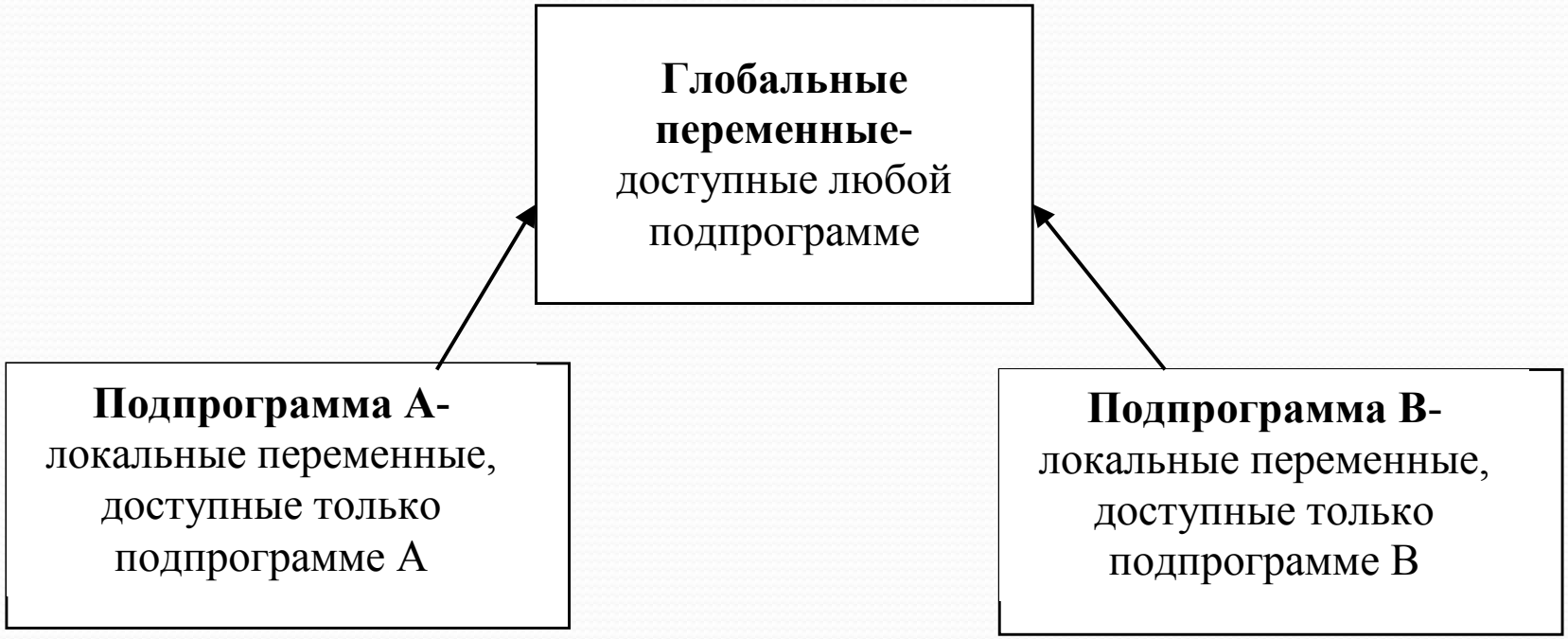
Результатом работы процедуры может быть какое-то значение переменной или несколько значений переменных, либо просто какие-то действия (например, по выводу текста или картинки на экран или принтер).

Функция является частным случаем процедуры, а именно, результатом работы функции всегда является одно скалярное значение, которое передается имени функции. Поэтому имя функции можно использовать в качестве операнда каких-то выражениях.

ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

Переменные, описанные в основной программе, называют ***глобальными переменными***. Переменные, описанные в процедуре, называются ***локальными переменными***.

схематически:



Допускается любой уровень вложенности процедур и функций. Любая программа, процедура и функция представляет собой блок со своей областью описаний и может содержать внутри этого блока описание других процедур и функций, а так же обращений к ним.

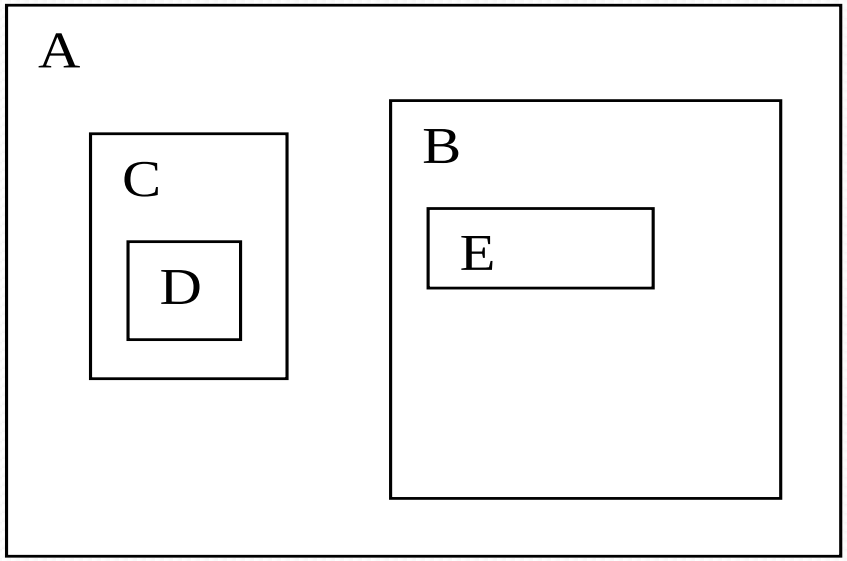
Блок, содержащий в своем разделе описаний другой блок, называется внешним. Блок, содержащийся в разделе описаний другого блока, называется внутренним или подблоком.

Объекты, описанные внутри какого-либо подблока, являются по отношению к нему локальными и недоступны внешним блокам.

Объекты, описанные в некотором внешнем блоке, доступны и могут быть использованы в любом его подблоке, то есть они являются глобальными по отношению к этим подблокам.

Только объекты, локальные для некоторого блока, являются глобальными для всех его подблоков (под объектами понимаются имена констант, переменных, типов процедур и функций).

Схематически изобразим структуру блоков некоторой программы



Здесь A – основная программа, B,C,D,E – подпрограммы (процедуры или функции).

Переменные, описанные в A, являются глобальными по отношению ко всем процедурам и функциям, описанные в C - глобальные для D и т.д.. Процедура E доступна в B, но недоступна в C и т.д.

Различают **стандартные (встроенные)** процедуры и функции, и процедуры и функции, **определяемые пользователем**. Стандартные процедуры и функции содержатся в библиотечных модулях системы программирования. В языке Pascal (Delphi) эти модули подключаются к программе при помощи ключевого слова

uses <имя модуля>.

модуль **system**,
Модуль **crt**
Модуль **graph (graphABC)**

для их подключения в программе на Pascale необходимо записать после имени программы следующую строчку:

program <имя программы>;
uses crt, graphABC;

Кроме стандартных процедур и функций пользователь может разрабатывать свои процедуры и функции. Они описываются в разделе описаний основной программы и по своей структуре аналогичны программе на языке Паскаль. После того, как эти процедуры и функции описаны пользователем, можно обращаться к ним по имени (вызывать их) в основной программе также как и к стандартным процедурам и функциям.

Рассмотрим более подробно работу с процедурами и функциями, определяемыми пользователем, в языке Паскаль.

Процедуры

- **Описание процедуры имеет вид:**
- ***Procedure*** *<имя процедуры> [(< список формальных параметров >)];*
- *< раздел описаний процедуры >*
- ***Begin***
- *< тело процедуры >*
- ***End;***

- **Обращение к процедуре (вызов ее в основной программе) имеет вид:**
- ***< имя процедуры >*** *[(список фактических параметров)];*

Формальные параметры - это имена переменных, которые используются при описании процедуры в ее заголовке и также в теле процедуры.

Фактические параметры - это значения (константы, выражения или имена переменных), которые подставляются на место формальных параметров при обращении к процедуре.

Количество формальных и фактических параметров, их последовательность и их типы должны быть согласованы.

Различают ***параметры-значения и параметры-переменные.*** Перед параметрами-переменными в описании процедуры ставится служебное слово **Var.**

Параметры-значения используются для передачи значений переменных из основной программы в процедуру, то есть служат как бы аргументами процедуры (входными параметрами).

Параметры-переменные возвращают результат работы процедуры в основную программу, то есть являются как бы выходными переменными для процедуры.

При изменении значения формального параметра-переменной во время выполнения процедуры одновременно происходит изменение значения фактического параметра. А при изменении формального параметра-значения изменение соответствующего ему фактического параметра не происходит.

Говорят, что обмен значениями между фактическими и формальными параметрами-переменными происходит ***по значению***, а параметрами-значениями ***по ссылке***.

В принципе, параметры-переменные можно использовать и в качестве входных переменных процедур, особенно если надо сэкономить память.

При описании процедуры, в списке формальных параметров, имена переменных одного типа указываются через запятую, после последнего имени ставится двоеточие и указывается тип параметра.

Параметры разных типов разделяются точкой с запятой, например:

Procedure pr1(x,y:integer; z:real; var t:real);

- **Пример 1.**Составить программу, выводящую на экран четыре горизонтальных линии из символов ” * ”:
- 1 линия-7 символов ,2 линия-19 символов,3 и 4 линии- любое количество символов (<80, соответствующие числа обозначим k3 и k4 и будем вводить их с клавиатуры).

- В результате работы программы на экране должны получиться следующие линии:
- * * * * * * *
- * * * * * * * * * * * * * * * * * * *
- * * * * * . . . . . * * * . . . . . * * * - всего k3
- * * * . . . . * * * - всего k4

Анализ программы:

1) Как вывести на экран линию из n звездочек?

Одну звездочку - write(‘*’);

n звездочек - цикл с параметром, тело цикла – write(‘*’);

For i : = 1 to n do write (‘ * ‘);

Затем перевести строку - Writeln;

2) Надо повторить этот цикл 4 раза, *при этом первый раз* n=7, *второй раз* n=19, *третий* - n=k3, *четвертый* - n=k4.

Для того, чтобы в программе четыре раза не повторять этот цикл, оформим его в виде ***процедуры,*** для которой ***параметром - значением будет число n.*** *В основной программе вызовем эту процедуру 4 раза, задавая соответствующие* ***фактические значения n***

```
Program pr_line;
    var  k3, k4 : integer ;
    {Описание процедуры}
    Procedure zvezd ( n : integer );
        Var  i : integer ;
    Begin
      For i : = 1 to n do
         Write ( ‘ * ‘ );
      Writeln; { для перевода строки}
    End ; {конец процедуры}
  Begin { Основная программа }
    Write (‘ Введите k3 и k4  ‘ );
    Readln( k3, k4 );
    Zvezd( 7 ); {первый вызов процедуры}
    Zvezd ( 19 ); {второй вызов процедуры }
    Zvezd ( k3 ); {третий вызов процедуры}
    Zvezd ( k4 ); {четвертый вызов процедуры}
  End.
```

Пример 2.Составить программу поиска максимального из четырех чисел, используя процедуру максимума из двух чисел .

- Анализ процедуры:
- **Параметры-значения и параметры-переменные?**
- в процедуре поиска максимума из двух чисел будет два *параметра-значения – обозначим их* ***x,y*** *(аргументы процедуры),* из которых ищется максимальное, и *один результат (параметр-переменная* ***z****)* - значение максимума из двух чисел.
- То есть заголовок описания процедуры поиска max из двух чисел будет иметь вид:
- ***procedure max_2 (x,y:real; var z:real);***
- Анализ основной программы:
- Для поиска максимума из нескольких чисел будем последовательно находить максимум из двух и ***результат помещать в переменную m;***
- на каждом следующем шаге ***сравнивать следующее число с m и хранить в m***. То есть в основной программе будем три раза обращаться к процедуре *max_2:*

Текст программы:

```
program max_4;
                var a, b, c, d, m : real;
                procedure max_2 (x, y : real; var z : real);
                begin
                        if x >y then z:=x else z:=y;
                end;
 begin
            write ( 'Введите 4 числа     ');
            readln (a,b,c,d);
            max_2 (a, b, m);  {первый вызов, в m-max из 2-х}
            max_2 (m, c, m); {второй  вызов, в m-max из 3-х}
            max_2 (m, d, m); {третий вызов, в m-max из 4-х}
            writeln ('max=',m)
 end.
```

- ***Самостоятельно 2 способ:*** находим max_2 из 1 и 2 числа, затем из 3 и 4 чисел, затем max_2 из получившихся двух максимумов

Функции

В том случае, когда результатом работы процедуры является одно скалярное значение, эту процедуру можно оформить в виде функции.

Описание функции имеет вид:

Function ***<имя>*** *(<список формальных параметров>): <тип результата>*

<раздел описаний>

begin

<тело функции>

end;

Обращение к функции (ее вызов) имеет вид:

<имя функции> *(<список фактических параметров>)*

Результат работы функции присваивается ее имени, поэтому **в теле функции обязательно должен быть оператор присваивания вида:**

<имя функции>:=<значение результата>

После того, как функция описана, её можно использовать **в качестве операнда по имени в правой части оператора присваивания** при вычислении значений каких-то выражений.

То есть использование функций, определенных пользователем, аналогично использованию встроенных стандартных функций (sin(x), cos(x) и т.д.).

Пример: Составить программу вычисления значения выражения

$$C = \frac{m!n!}{(m-n)!}, \quad \text{где } (m > n)$$

- Здесь трижды надо вычислить факториал, поэтому оформим вычисление факториала произвольного числа в виде функции.
- Функция для вычисления факториала будет иметь один формальный параметр, обозначим его буквой k.
- Для вычисления k! необходимо организовать цикл с параметром i, который будет изменяться от 1 до k, а **в теле** цикла накапливать произведение **P := P*i**. Начальное присваивание: P=1.
- Переменные P и i будут локальными переменными и описываются в разделе описаний функции:

```
Program  primer_func;
        Var  m,n:integer;  C:real; {глобальные переменные}

        Function fact (k:integer):longint;
                Var i:integer; P:longint; {локальные переменные}
        Begin {тело функции}
                P:=1;
                For i:=1 to k do
                        P:=P*i;
                Fact:=P
        End;
Begin {основная программа}
        Write (‘m,n=’);
        Readln (m,n);
        C:=fact (m) * fact(n) /fact(m-n);
        Writeln (c)
End.
```