Лекция 2. Модульное программирование

Процедуры и функции: описание и использование. Виды параметров подпрограмм: значения, переменные, константы, открытые, процедурные. Рекурсия. Модули: описание и использование. Стандартные модули Паскаля.

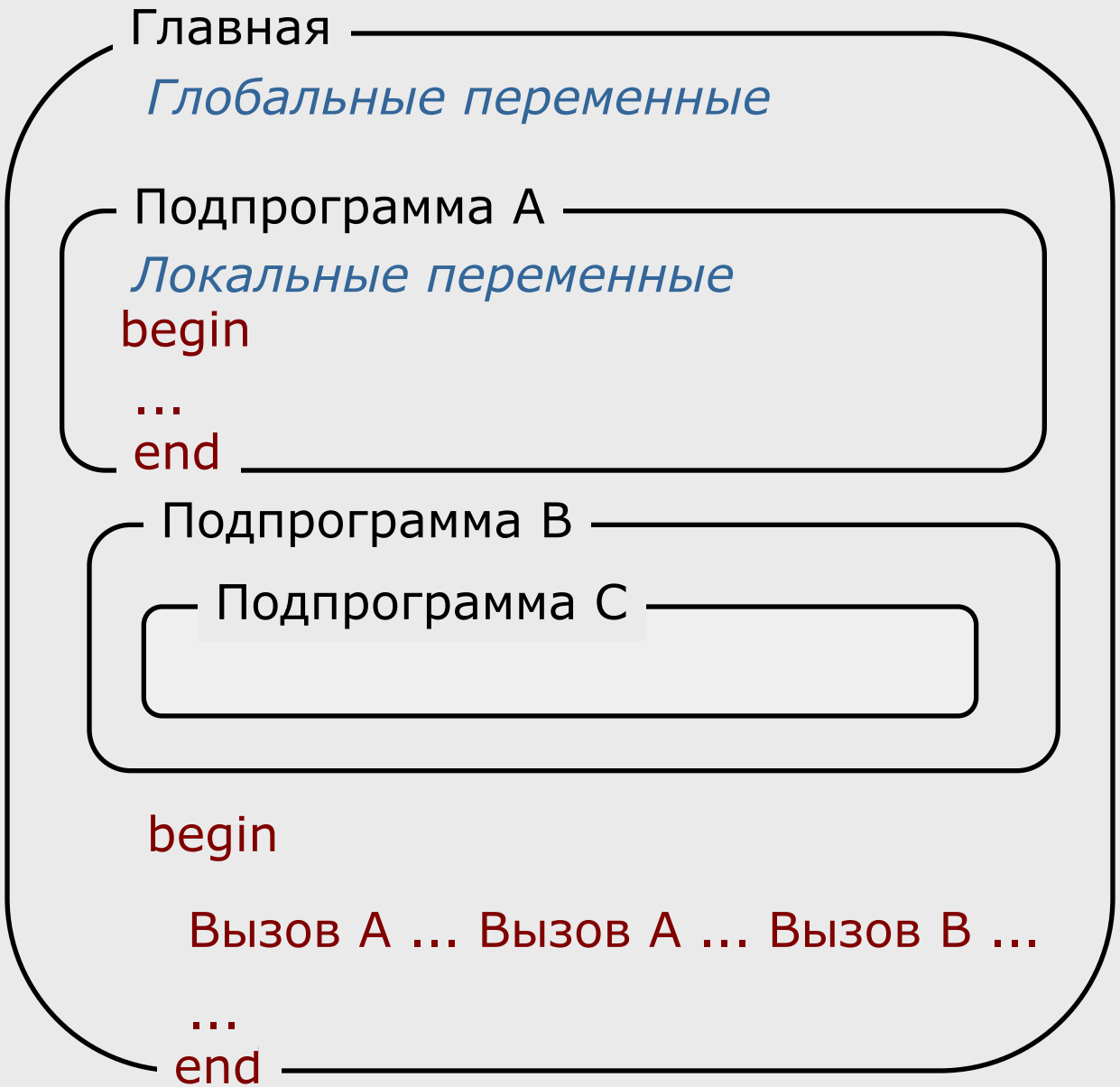
Павловская Т.А. (СПбГУИТМО)

Подпрограммы

Общие положения

- Подпрограмма — это фрагмент кода, к которому можно обратиться по имени
- Логические законченные части программы оформляются в виде подпрограмм
- Подпрограмма записывается один раз, а вызываться может столько раз, сколько необходимо
- Одна и та же подпрограмма может обрабатывать различные данные, переданные ей в качестве параметров.

Структура программы

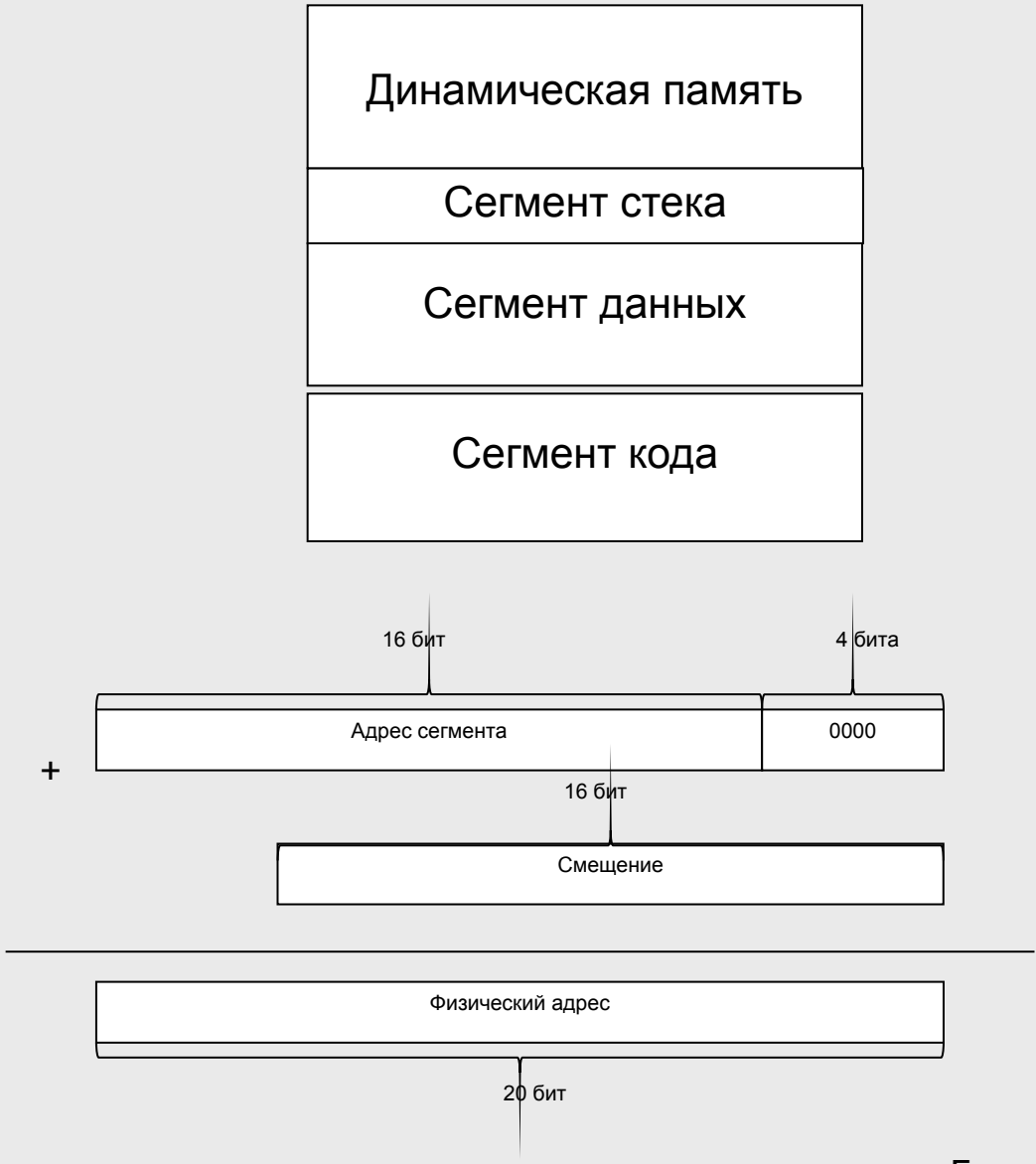


Структура программы в оперативной памяти

В IBM PC-совместимых компьютерах память условно разделена на сегменты.

Адрес каждого байта составляется из номера сегмента и смещения.

Компилятор формирует **сегмент кода**, в котором хранится программа в виде машинных команд, и **сегмент данных**, в котором выделена память под глобальные переменные программы.



Виды переменных

- *Глобальными* называются переменные, которые описаны в главной программе. Время жизни глобальных переменных — с начала программы и до ее завершения. Располагаются в сегменте данных.
- В подпрограммах описываются *локальные* переменные. Они располагаются в сегменте стека, причем распределение памяти происходит в момент вызова подпрограммы, а ее освобождение — по завершении подпрограммы.
- Локальные переменные автоматически не обнуляются.

Вызов

Для того чтобы подпрограмма выполнилась, ее надо **вызвать**.

Вызов подпрограммы записывается в том месте программы, где требуется получить результаты работы.

Процедура вызывается с помощью отдельного оператора, а функция — в правой части оператора присваивания, например:

```
inc(i); writeln(a, b, c);        { вызовы процедур }

y := sin(x) + 1;                 { вызов функции }
```

Процедуры

заголовок

```
procedure <имя> [(список параметров)];
<разделы описаний>
begin
    <раздел операторов>
end;
```

Пример процедуры

Вычислить разность между средними арифметическими значениями элементов двух вещественных массивов

```
program dif_average;
const n = 3;
type mas = array[1 .. n] of real;
var   a, b : mas;
      i      : integer;
      dif, av_a, av_b : real;
```

```
procedure average(x : mas; var av : real)
var i : integer;
begin
      av := 0;
      for i := 1 to n do av := av + x[i];
      av := av / n;
end;
```

{Главная программа}

```
begin      for i := 1 to n do read(a[i]);
       for i := 1 to n do read(b[i]);
       average(a, av_a);                  average(b, av_b);

       dif := av_a - av_b;
       writeln('Разность значений ', dif:6:2)
end.
```

Функции

заголовок

Описание функции:

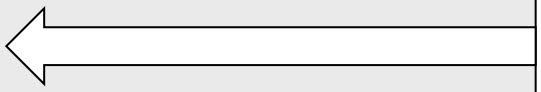
```
function <имя> [(список параметров)] : <тип>;
<разделы описаний>
begin
    <раздел операторов>
    <имя> := <выражение>;
end;
```

Простейший вызов функции:

имя переменной := имя (список аргументов);

Пример функции

```
program dif_average1;
const n = 3;
type mas = array[1 .. n] of real;
var   a, b : mas;
      i      : integer;
      dif    : real;
```



```
function average(const x : mas): real;
var    i : integer;
       av : real;
begin
       av := 0;
       for i := 1 to n do av := av + x[i];
       average := av / n;
end;
```

```
{Главная программа}
begin      for i := 1 to n do read(a[i]);
       for i := 1 to n do read(b[i]);
       dif := average(a) - average(b);
       writeln('Разность значений ', dif:6:2)
end.
```

Виды параметров подпрограмм

Обмен данными

- Через глобальные переменные
- Через параметры

Параметры передаются:

- По значению
- По адресу (var или const)

Типы параметров:

- Значения
- Переменные
- Константы
- Бестиповые
- Открытые
- Процедурные
- Объекты

Параметры и аргументы

- При вызове подпрограммы после ее имени в скобках указываются аргументы, то есть те конкретные величины, которые передаются в подпрограмму
- Список аргументов как бы накладывается на список параметров и замещает их, поэтому **аргументы должны соответствовать параметрам по количеству, типу и порядку следования.**
- Для каждого параметра обычно задается его имя, тип и способ передачи.
- Либо тип, либо способ передачи могут не указываться.
- В заголовке подпрограммы нельзя вводить описание нового типа.

```
y := sh(x, 1e-5);
```

```
function sh(x, e: real): real;
```

```
procedure a(var m: mas;
x: real; var y);
```

Параметр-значение

- **имя : тип;**

Например:

- procedure P(x : integer);

При вызове подпрограммы на месте параметра, передаваемого по значению, может находиться выражение:

P(x); P(3); ~~P(1.2);~~ ~~P(a + b/2);~~ P(a + b div 2);

{ var x, a, b : integer; }

Тип выражения должен быть совместим по присваиванию с типом параметра.

Параметр-переменная

var имя : тип;

Например:

- procedure P(var x : integer);

При вызове подпрограммы на месте параметра-переменной может находиться только ссылка на переменную точно того же типа:

P(a); P(x);

Параметр-константа

const имя : тип;

Например:

- procedure P(const x : integer);

При вызове подпрограммы на месте параметра может быть записано выражение, тип которого совместим по присваиванию с типом параметра.

Пример

```
var a, b, c, d, e : real;
procedure XaXa(a, b: real; c : real; var d : real);
    var e: real;
    begin  c := a + b; d := c; e := c;
    writeln ('c=', c, ' d=', d, ' e=', e);
end;
begin
    a:=3; b :=5;
    XaXa(a, b, c, d);
    writeln ('c=', c, ' d=', d, ' e=', e);
end.
```

```
c= 8 d= 8 e= 8
c= 0 d= 8 e= 0
```

Пример

Заголовок процедуры имеет вид:

```
Procedure P(a:real; b:char; var c:real);
```

Переменные в вызывающей программе описаны так:

```
Var a:integer; b,c:char; d,x:real;
--------------------------
```

- P(a, b, c); В какой строке ошибка?
- P(d+a, c, x);
- P(x, ’c’, d);
- P(a, b, a+1);

Итоги

- Для передачи в подпрограмму исходных данных испольуются параметры-значения и параметры-константы. Параметры составных типов (массивы, записи, строки) предпочтительнее передавать как константы.

- Результаты работы процедуры следует передавать через параметры-переменные, результат функции — через ее имя.

Открытый массив

- Может быть *только одномерным* и состоять из элементов любого типа, кроме файлового.
- На место открытого массива можно передавать одномерный массив любой размерности, состоящий из элементов такого же типа
- Элементы массива нумеруются с нуля. Номер максимального элемента в массиве можно определить с помощью функции High.

Пример. Функция, определяющая максимальный элемент любого целочисленного массива.

```
function max_el(const mas : array of integer) : integer;
var i, max : integer;
begin
    max := mas[0];
    for i := 0 to High(mas) do
        if mas[i] > max then max := mas[i];
    max_el := max;
end;
```

Открытые строки

Строки произвольной длины можно передавать:

- по значению;
- на место параметра-константы.

Для передачи в подпрограмму строк любой длины по адресу используются:

- специальный тип OpenString, называемый *открытой строкой;*
- тип string при включенном режиме {$P+}.

```
type s20 = string[20];
var s1 : string[40];
    s2 : string[10];
procedure P(const x : s20; y : string;
            var z : openstring);
...
begin  ...  P(s2, s1, s1); ...
end.
```

Бестиповые параметры

```
  Function EQ( var x, y; size: word ): boolean;
{С помощью функции EQ можно сравнить две любые величины}
  Type Bytes = array[ 0 .. MaxInt ] of byte;

  Var xb: Bytes absolute x;
      yb: Bytes absolute y;     n : integer;

  Begin
    n :=0;
    While (n < size) and (xb[n] = yb[n]) do inc(n);
    EQ := n = size;
  End;
```

Пусть, например, в программе описаны переменные:

```
var a, b : array [1 .. 10] of byte;
    x    : real;
    c    : string;
```

Следующие обращения к функции EQ будут корректны:

```
EQ(a, b, sizeof(a))    { сравнение двух массивов }

EQ(a[2], b[5], 4) { сравнение 2–5 элементов массива "a" с
   5–8 элементами массива "b", соответственно }

EQ(c, x, sizeof(real)) { сравнение первых 6 байт строки c с
   переменной x }
```

Параметры процедурного типа

```
procedure tabl_fun(x, Xk, dX : real; f : fun);
begin
    writeln(' ------------------------- ');
    writeln('|      X        |       Y       |');
    writeln(' ------------------------- ');

    while x <= Xk do begin
     writeln('|', x:9:2,'    |', f(x):9:2,'    |');
     x := x + dX;
    end;

    writeln(' ------------------------- ');
end.
```

Работа с параметром

1. Определить процедурный тип

```
type fun = function(x : real) : real;
```

2. Задать для подпрограмм, предназначенных для передачи в качестве параметра, ключ компилятора {$F+}, определяющий дальнюю адресацию

```
{$F+}
function Q(x : real) : real;
begin
    Q := 2 * x / sqrt(1 – sin(2 * x));
end;
{$F–}
```

3. Задать в заголовке подпрограммы параметр процедурного типа:

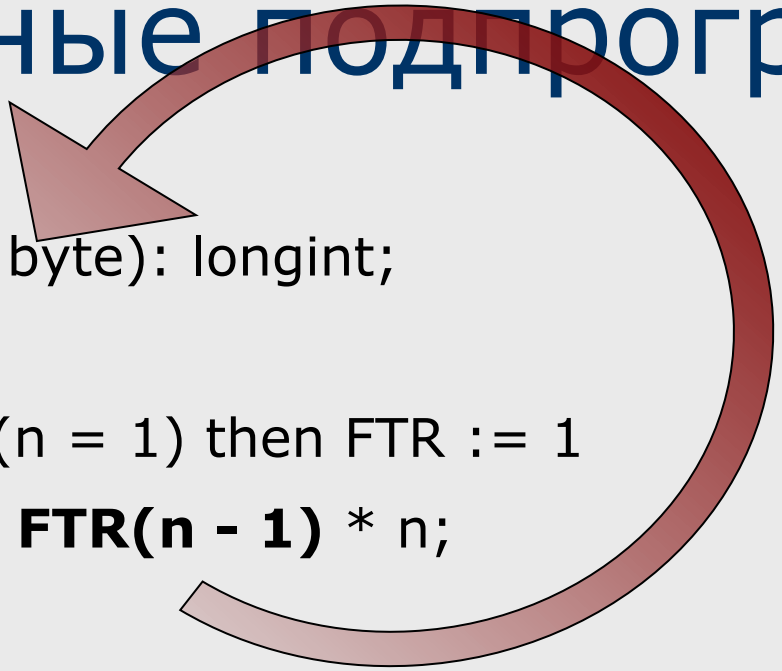
```
procedure tabl_fun(x, Xk, dX : real; f : fun);
```

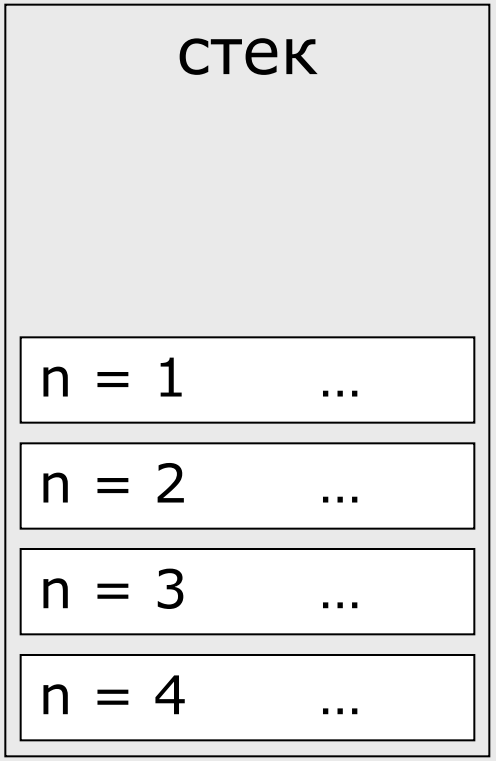
4. Передать в подпрограмму имя конкретной функции/процедуры:

```
tabl_fun(0, 10, d, Q);
```

Рекурсивные подпрограммы

```
function FTR(n : byte): longint;
begin
    if (n = 0) or (n = 1) then FTR := 1
    else   FTR := FTR(n - 1) * n;
end;
----------
A := FTR(4);
```





Модули

Структура модуля

```
unit имя;              { заголовок модуля }

interface
    { интерфейсная секция модуля }
    { описание глобальных элементов модуля
      (видимых извне) }

implementation
     { секция реализации модуля }
     { описание локальных (внутренних) элементов
      модуля }

begin
          { секция инициализации }
     { может отсутствовать }
End;
End.
```

Состав модуля

- *В интерфейсной секции модуля* определяют константы, типы данных, переменные, а также заголовки процедур и функций.
- *В секции реализации* описываются подпрограммы, заголовки которых приведены в интерфейсной части. Кроме того, в этой секции можно определять константы, типы данных, переменные и внутренние подпрограммы.
- *Секция инициализации* предназначена для присваивания начальных значений переменным, которые используются в модуле.

Пример интерфейсного раздела:

```
Interface
{подключаемые модули}
Uses  AnotherUnit;
{константы}
Const
  PI=3.14159265;
  E=2.71828182;
{пользовательские типы данных}
Type
  TMyType=array[-3..7] of real;
{переменные}
Var
  temp:TMyType;
{процедуры и функции}
Procedure Fill(var x:TMyType);
Function Find(const x:TMyType; const Value:real):Boolean;

Implementation
```

Пример модуля

```
unit Average;
interface
   const n = 10;
   type mas = array[1 .. n] of real;
   procedure aver_mas(x : mas;
           var av : real);
implementation
   procedure aver_mas(x : mas;
           var av : real);
     var i : integer;
   begin
           av := 0;
           for i := 1 to n do
           av := av + x[i];
           av := av / n;
   end;
end.
```

Использование модуля в программе:

```
program dif_average;
uses Average;
var a, b : mas;
        i : integer;
        dif, av_a, av_b : real;
begin
     for i := 1 to n do read(a[i]);
     for i := 1 to n do read(b[i]);
     aver_mas(a, av_a);
     aver_mas(b, av_b);
     dif := av_a – av_b;
     writeln('Разность:', dif:6:2);
end.
```

Пример модуля 2

```
Unit primer1;
Interface
Const n=50;
Type
Massiv=array[1..n]of integer;
Procedure Vvod(Var m:byte; Var X:massiv);
Procedure Vyvod(m:byte;X:massiv);
Procedure SumKol(m:byte;X:massiv;Var Sum,Kol:integer);
Implementation
Procedure Vvod(Var m:byte; Var X:massiv);
var i:integer;
begin
For i:=1 to m do
x[i]:=random(20)-10;
end;
Procedure Vyvod(m:byte; X:massiv);
var i:integer;
begin
For i:=1 to m do
write(X[i]:5);
writeln
end;
Procedure SumKol(m:byte;X:massiv;Var Sum,Kol:integer);
var i:integer;
begin
Sum:=0;
Kol:=0;
For i:=1 to m do
If X[i]>0 then
 begin
  Sum:=Sum+X[i];
  Inc(Kol);
 end;
end;
end.
```

Использование модуля в программе:

```
Uses primer1;
Var A,B:massiv;
Na,Nb:byte;
Kol_A, kol_B, sum_A, Sum_B:integer;
Begin
Write ('Введите число элементов в массиве A
(<50)');
readln(Na);
Vvod(na,A);
Writeln('Массив A:');
Vyvod(na,A);
SumKol(na,A,Sum_A,Kol_A);
writeln('Sum_A=',Sum_A,'':2, 'Kol_A=',Kol_A);
Write ('Введите число элементов в массиве B
(<50)');
readln(Nb);
Vvod(nb,B);
Writeln('Массив B:');
Vyvod(nb,B);
SumKol(nb,B,Sum_B,Kol_B);
writeln('Sum_B=',Sum_B,'':2, 'Kol_B=',Kol_B);
end.
```

Стандартные модули Паскаля

Стандартные модули Pascalabc.net

- Crt
- GraphABC
- ABCObjects
- ABC Sprites