

## Блочная структура программы



*Архитектор не может похоронить свои ошибки, как врач.*

*Ф. Райт*

Компьютерные науки

© М.Л. Цымблер

---

---

---

---

---

---

---

---

## Содержание

- Область действия декларации
- Правила видимости
- Локальные и глобальные переменные
- Распределение памяти

Блочная структура программы

© М.Л. Цымблер

2

---

---

---

---

---

---

---

---

## Определяющее и использующее вхождение

- *Определяющее вхождение идентификатора* – вхождение идентификатора в объявление переменной, метки и др.
- *Использующее вхождение идентификатора* – вхождение идентификатора в оператор.
- Пример:  
`var A: Real; { Определяющее вхождение A }`  
`...`  
`A:=A+B; { Использующее вхождение A }`

Блочная структура программы

© М.Л. Цымблер

3

---

---

---

---

---

---

---

---

### Область действия декларации

- *Область действия декларации* – это часть текста программы, начинающаяся с данной декларации и завершающаяся концом текущего блока.
- Понятие области действия декларации необходимо для связывания использующих вхождений идентификаторов с правильными определяющими вхождениями данных идентификаторов.

---

---

---

---

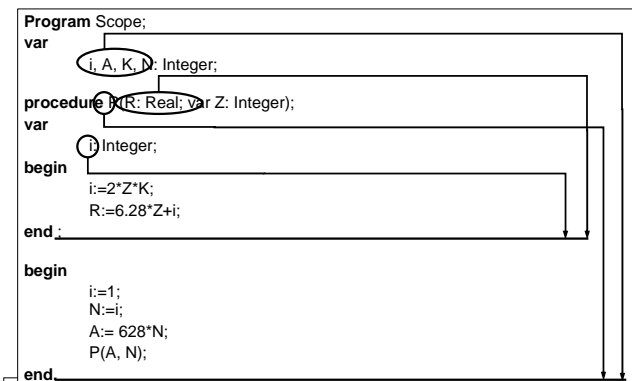
---

---

---

---

### Пример: область действия декларации




---

---

---

---

---

---

---

---

### Связывание использующих вхождений идентификатора с определяющим вхождением

- Каждое использующее вхождение идентификатора связывается с тем определяющим вхождением данного идентификатора, которое имеет **наименьшую** область действия.

---

---

---

---

---

---

---

---

### Пример: область действия декларации

```

Program Scope1;
var
    i, A, K, N: Integer;
procedure P(A: Real; N: Integer);
var
    i: Integer;
begin
    N:=4*N;
    A:=6.28*N;
end;
begin
    i:=1;
    N:=i;
    A:=628*N;
    P(A, N);
end.
    
```

---

---

---

---

---

---

---

---

### Правила видимости

- Идентификатор, определенный в блоке, является *видимым* в данном блоке (может быть использован в операторах и выражениях данного блока).
- Идентификатор, определенный во внешнем блоке, может быть переопределен во внутреннем блоке. В этом случае объект, связанный с внешним определением, становится *невидимым* во внутреннем блоке.

---

---

---

---

---

---

---

---

### Пример: видимость программных объектов

```

Program Scope2;
var
    K, i, A, N: Integer;
procedure P(A: Real; N: Integer);
const
    K:=413;
var
    i: Integer;
begin
    N:=i*K;
    A:=6.28*N;
end;
begin
    K:=314;
    i:=1;
    N:=i;
    A:=628*N;
    P(A, N);
end.
    
```

---

---

---

---

---

---

---

---

### Локальные и глобальные переменные

- Переменная называется *локальной по отношению к данному блоку*, если блок содержит связанное с ней определяющее вхождение, и *глобальной* – в противном случае.

Блочная структура программы © М.Л. Цымблер 10

---

---

---

---

---

---

---

---

### Пример: локальные и глобальные переменные

```

Program LocalAndGlobal;
const
  N=10;
type
  TArr=array[1..N] of Real;
var
  A, B, C, Res: Real;
procedure InpData(var A, B, C: Real);
begin
  ...
end;
procedure Calculate(A, B, C: Real; var Res: Real);
var
  i,j,k: Integer;
function Search(A: TArr; E:Real): Integer;
begin
  ...
end;
begin
  ...
  procedure OutData(Res: Real);
  begin
    ...
  end;
  begin
    InpData(A,B,C);
    Calculate(A,B,C,Res);
    OutData(Res);
  end;
end.
    
```

Блочная структура программы © М.Л. Цымблер 11

---

---

---

---

---

---

---

---

### Глобальные переменные вредны

- Использование глобальных переменных снижает читаемость программы и может привести к *побочным эффектам*.
- *Побочный эффект подпрограммы* – выполнение подпрограммой действий, результат которых обнаруживается вне данной подпрограммы, например, изменение значений глобальных по отношению к ней переменных.
- Следует избегать, где это возможно, **использования глобальных переменных** (использовать локальные переменные и/или параметры подпрограмм).

Блочная структура программы © М.Л. Цымблер 12

---

---

---

---

---

---

---

---

### Пример: побочный эффект

```

Program SideEffectsBad;
var
  i: Integer;
  C: Real;
function Deg(A: Real; N: Integer):
Real;
{Вычисляет A^N}
begin
  C:=1;
  for i:=1 to N do
    C:=C*A;
  Deg := C;
end;

function Sum(N: Integer): Real;
{Вычисляет 1/2+1/4+1/8+...+1/(2^N)}
begin
  C:=0;
  for i:=1 to N do
    C:=C+1/Deg(2,i);
  Sum:=C;
end;

begin
  WriteLn('0.875=', Sum(3):5:3);
end.
    
```

**Глобальная переменная**

**Побочный эффект**

0.875=8.125

Блочная структура программы © М.Л. Цымблер 13

---

---

---

---

---

---

---

---

### Пример: нет побочного эффекта

```

Program NoSideEffects;
function Deg(A: Real; N: Integer):
Real;
{Вычисляет A^N}
var
  i: Integer;
  C: Real;
begin
  C:=1;
  for i:=1 to N do
    C:=C*A;
  Deg := C;
end;

function Sum(N: Integer): Real;
{Вычисляет 1/2+1/4+1/8+...+1/(2^N)}
var
  i: Integer;
  C: Real;
begin
  C:=0;
  for i:=1 to N do
    C:=C+1/Deg(2,i);
  Sum:=C;
end;

begin
  WriteLn('0.875=', Sum(3):5:3);
end.
    
```

**Локальная переменная**

0.875=0.875

Блочная структура программы © М.Л. Цымблер 14

---

---

---

---

---

---

---

---

### Распределение памяти

Системная область

Куча

Сегмент стека

Сегмент данных

Системная область

0 Кб

Программа

Блочная структура программы © М.Л. Цымблер 15

---

---

---

---

---

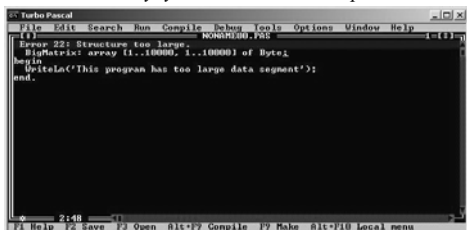
---

---

---

## Сегмент данных

- Сегмент данных содержит глобальные переменные программы и всех используемых модулей **unit**.
- Максимальный размер сегмента данных 64 Кб. Если под глобальные переменные требуется больше памяти, необходимо использовать кучу и динамические переменные.



```

Turbo Pascal
File Edit Search Run Compile Debug Tools Options Window Help
1-1-1
Error 22: Structure too large.
HighMatrix: array (1..10000, 1..10000) of Byte;
begin
writeln('This program has too large data segment');
end.
2:48
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu
    
```

---

---

---

---

---

---

---

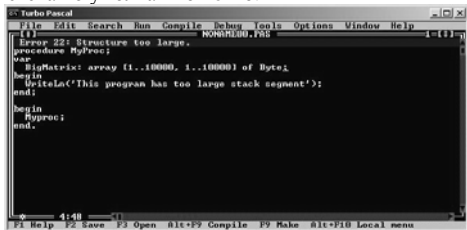
---

---

---

## Сегмент стека

- Сегмент стека содержит локальные переменные всех процедур и функций, причем только на время их выполнения.
- Размер сегмента стека определяется директивой компилятора \$M, и может изменяться от 1 Кб до 64 Кб. Размер сегмента стека по умолчанию 16 Кб.



```

Turbo Pascal
File Edit Search Run Compile Debug Tools Options Window Help
1-1-1
Error 22: Structure too large.
HighMatrix: array (1..10000, 1..10000) of Byte;
procedure MyProc;
begin
writeln('This program has too large stack segment');
end;
begin
MyProc;
end.
4:48
F1 Help F2 Save F3 Open Alt+F9 Compile F9 Make Alt+F10 Local menu
    
```

---

---

---

---

---

---

---

---

---

---

## Куча

- Куча (*heap*), динамическая память – память, организованная в виде списка свободных участков. Используется при работе с динамическими переменными.

---

---

---

---

---

---

---

---

---

---