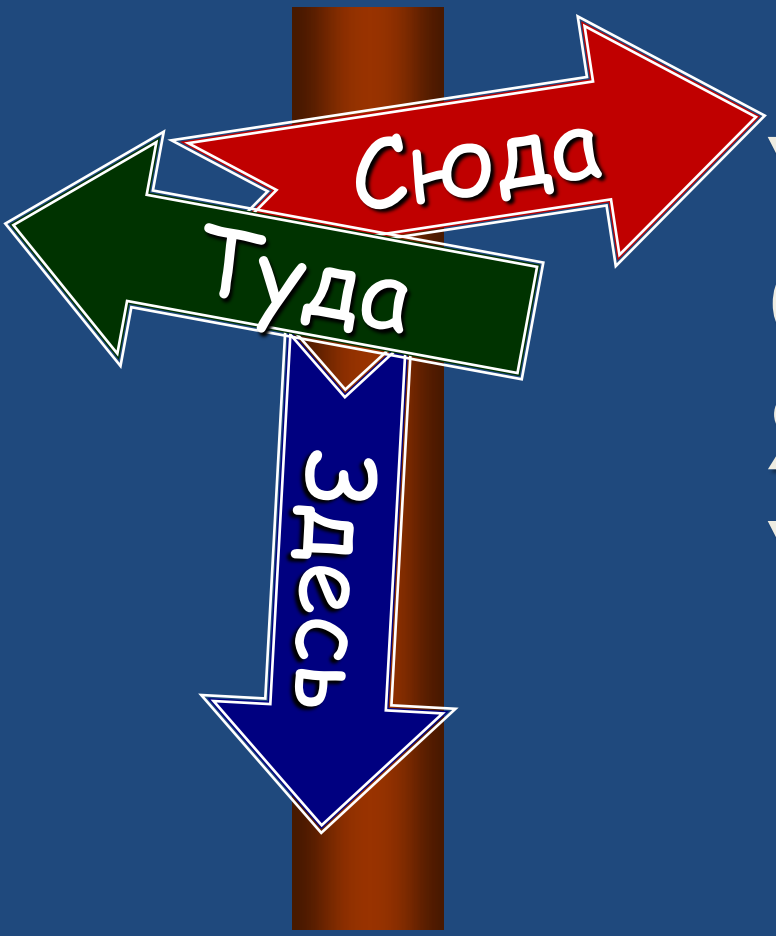


УКАЗАТЕЛИ И ССЫЛОЧНЫЕ ТИПЫ ЯЗЫКА ВЫСОКОГО УРОВНЯ PASCAL

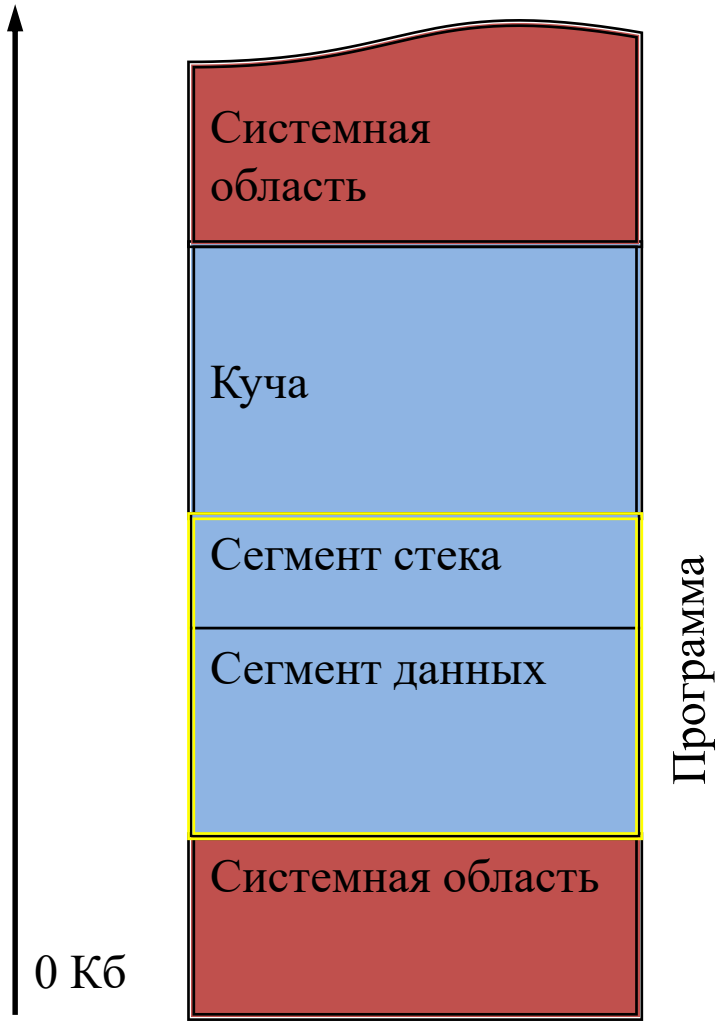
Чем больше указателей, тем труднее поиски.
В. Домиль

Основы программирования

Содержание

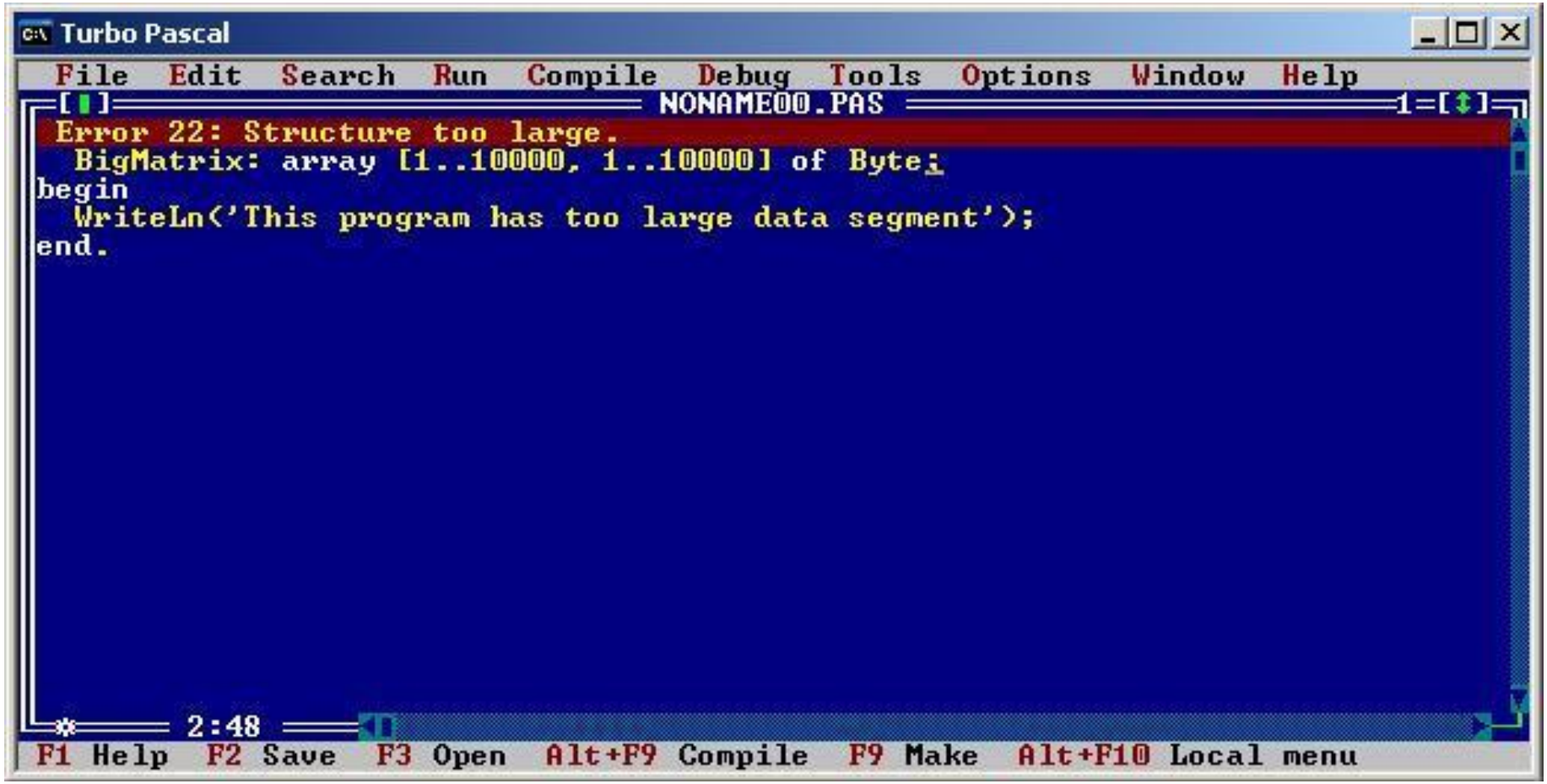
- Распределение памяти программы
- Указатели и ссылочные типы
- Статические и динамические переменные
- Управление кучей

Распределение памяти



- *Сегмент данных* содержит глобальные данные программы и подключенных библиотек.
- *Сегмент стека* содержит локальные переменные всех подпрограмм, причем только на время их выполнения.
- *Куча (heap)* – память, организованная в виде списка свободных участков.

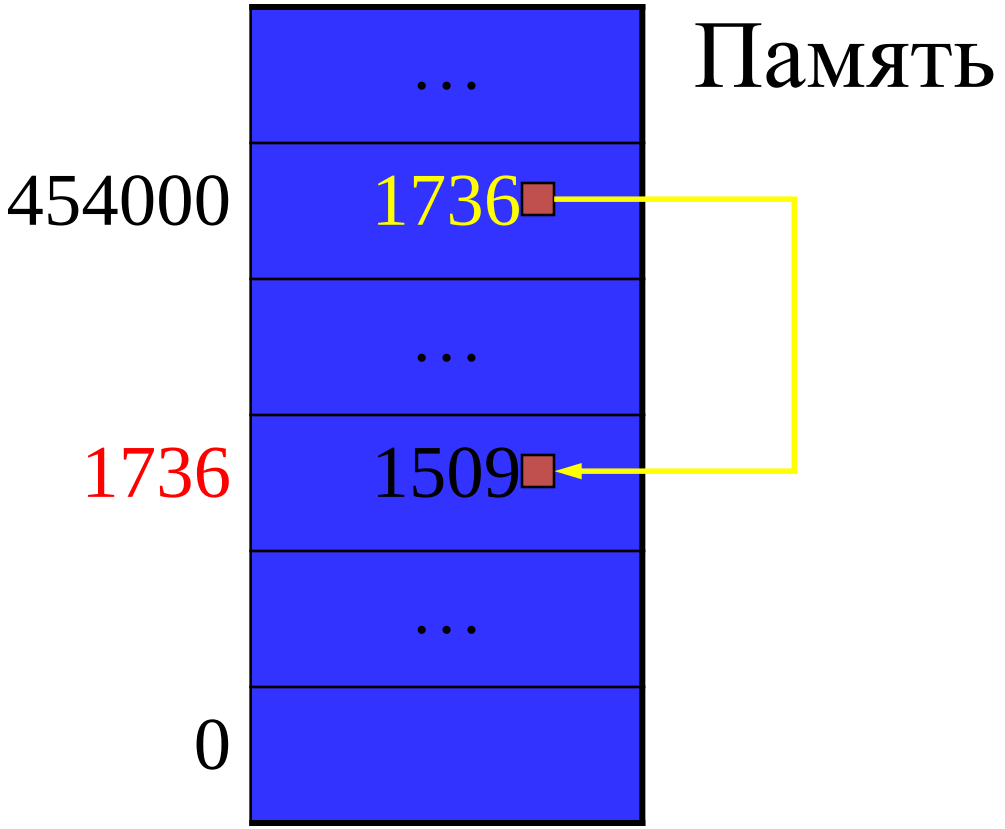
Сегмент данных



Сегмент стека

Указатели

- *Указатель* – значение, задающее *адрес* другого значения в памяти.



Виды указателей

- *Типизированные указатели* задают адрес другого значения, имеющего предопределенный тип данных.
- *Нетипизированные указатели* задают адрес другого значения, которое имеет любой тип данных.

Ссылочные типы

- *Ссылочные типы* используются для описания указателей.
- *Типизированные* указатели

```
type
  <Имя>=^<ИД базового типа>;
```

Базовый тип – любой тип данных.

- *Нетипизированные* указатели

```
var
   P: Pointer;
```

Операции и константы для ссылочных типов

- *Пустая ссылка* (не указывающая ни на какое значение) – **NIL**.
- Операции отношения
  - **=** – указывают на один и тот же адрес
  - **<>** – указывают на разные адреса.

Пример: объявление ссылочных типов


```
type
 Pinteger=^Integer;
 Preal=^Real;
 Pnode=^TNode;
 Tnode=record
  Info: Real;
  Next: PNode;
 end;
```

```
type
 UntypedPtr=Pointer;
 TNode=record
  Info: Real;
  Next: Pointer;
 end;
var
 P1: Pointer;
 P2: UntypedPtr;
```

Классификация переменных программы

- *Статические переменные*
  - создаются во время компиляции программы
    - явно с помощью декларации **var**
    - неявно как формальные параметры подпрограмм
  - не уничтожаются во время работы
  - размещаются в сегменте данных или в сегменте стека
- *Динамические переменные*
  - явно создаются и явно уничтожаются *динамически* во время выполнения программы.
  - размещаются в *куче*.

Статические переменные

- Для получения адреса статической переменной используется *операция взятия адреса* **@**.
  - @MyVar выдает адрес переменной MyVar.
- Для получения доступа к переменной по указателю используется *операция разыменования (раскрытия) ссылки* **^**.
  - MyPtr^ обозначает переменную, на которую ссылается указатель MyPtr.

Пример: работа со статическими переменными


```
var
 Ptr: ^Integer;
 i: Integer;
begin
 i := 1;
 Ptr=@i;
 Ptr^ := Ptr^+1;
 WriteLn('i=', i); { i=2 }
end.
```

Динамические переменные

- *Выделение памяти*
  New(<Имя переменной ссылочного типа>)
  Переменной присваивается значение указателя на вновь созданную динамическую переменную.
- *Освобождение памяти*
  Dispose(<Имя переменной ссылочного типа>)
  Память, связанная с динамической переменной, возвращается в кучу.
  Сравните Dispose(P); *и* P:=NIL;

Пример: работа с динамическими переменными


```
var
   i, j: PInteger;
   r: PReal;
begin
   New(i);
   New(r);
   i^ := 2;
   r^ := i^*5.25;
   j := i;
   WriteLn('i^=', i^); { i^=2 }
   WriteLn('r^=', r^:5:2); { r^=10.50 }
   WriteLn('j^=', j^); { j^=2 }
   Dispose(i);
   Dispose(r);
end.
```

Ошибки при работе с указателями

Примеры ошибочных операторов	Причина ошибки
`r := Sqr(r^);` `r := Sqr(r);`	Указателю ссылочного типа нельзя присвоить значение выражения типа, который не является ссылочным и не совместим с ним по присваиванию.

`r^:= i; r:= i^;`
`i^:=NIL;`
`i^:=r^; i^:=3.5;`
`i:=r; r:=i;`
	Типы идентификатора и выражения в операторе присваивания должны быть совместимы по присваиванию.

`i:=j; j:=NIL;`
`Write(i^, j^);`
`i:=j; Dispose(i);`
`Write(i^, j^);`
	Обращение к указателю после его уничтожения.

Управление кучей

- MemAvail
  размер суммарного свободного пространства в куче в байтах.
- MaxAvail
  размер наибольшего непрерывного свободного блока в куче в байтах.
- SizeOf(<Идентификатор типа>)
  размер внутреннего представления значений данного типа в байтах.

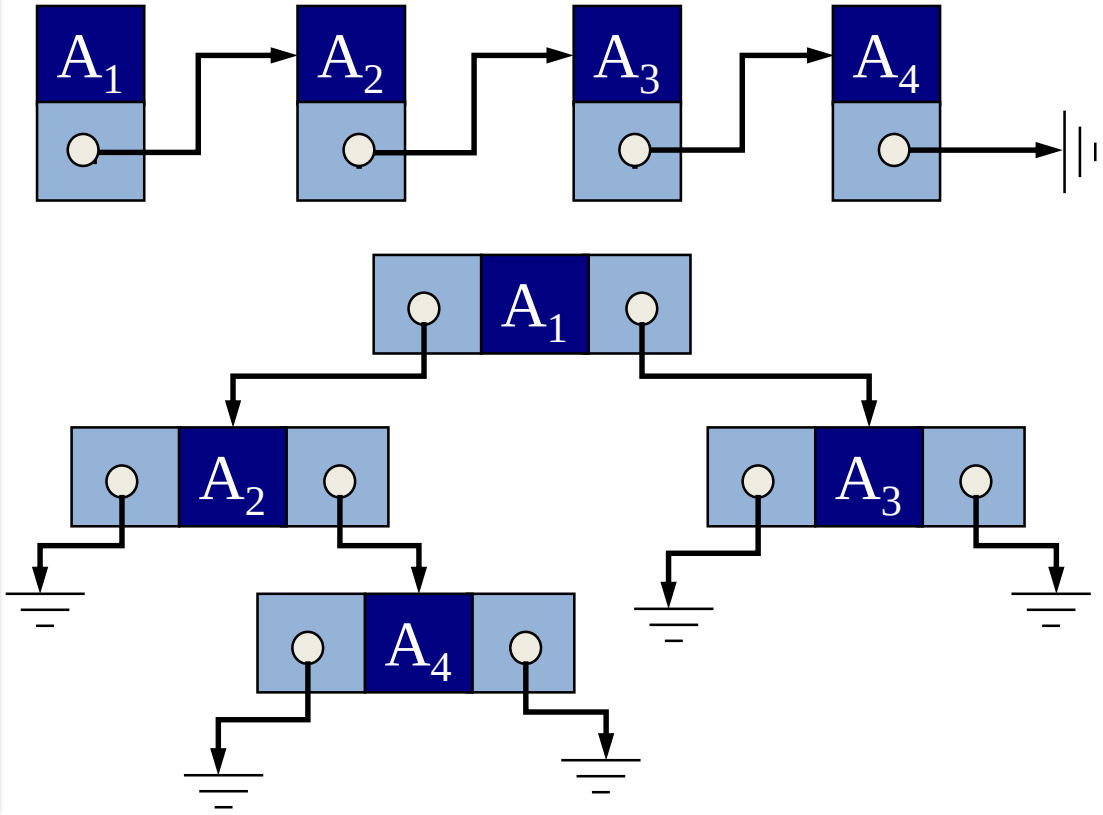
Пример: управление кучей


```
function MakeNode(Info: Real): PNode;
var
  P: PNode;
begin
  if MaxAvail<SizeOf(TNode) then
     P := Nil
  else begin
     P := New(PNode);
     P^.Info := Info;
     P^.Next := Nil;
  end;
  MakeNode := P;
end;
```

Применение ссылочных типов

- Ссылочные типы применяются *для реализации списковых и древовидных структур*.

```
type
  TInfo=Integer;
  PListNode=^TListNode;
  TListNode=record
    Info: TInfo;
    Next: PListNode;
  end;
  PTreeNode=^TTreeNode;
  TTreeNode=record
    Info: TInfo;
    Left, Right: PTreeNode;
  end;
```



Заключение

- Программа, загруженная в оперативную память, включает в себя сегмент данных и сегмент стека.
- Сегмент данных содержит глобальные переменные программы.
- Сегмент стека содержит локальные переменные подпрограмм, причем только на время их выполнения.

Заключение

- Указатель – значение, задающее адрес другого значения в памяти. Ссылочные типы используются для описания указателей.
- Статические переменные размещаются в сегменте данных или в сегменте стека и описываются явно, с помощью декларации var, или неявно, как формальные параметры процедур и функций.
- Динамические переменные создаются и уничтожаются динамически во время выполнения программы, в куче.