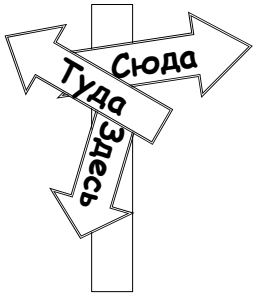
Указатели и ссылочные типы



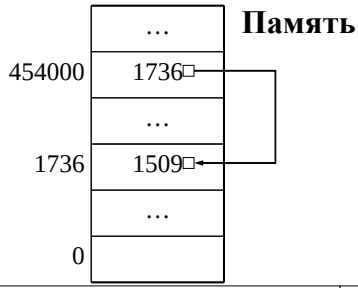
Если на клетке со слоном видишь указатель «буйвол», не верь глазам своим.
Козьма Прутков

Содержание

- Указатели и ссылочные типы
- Статические и динамические переменные
- Управление кучей

Указатели

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



Типизированные vs нетипизированные указатели

- *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>=^<Идентификатор 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** или неявно, как формальные параметры подпрограмм) и 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 специальной области памяти, называемой *кучей*.

Статические переменные

- Для получения адреса статической переменной используется *операция взятия адреса* @. Например, @MyVar выдает адрес переменной MyVar.
- Для получения доступа к переменной по указателю используется *операция разыменования (раскрытия) ссылки* **^**. Например, 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.

Указатели и ссылочные типы © М.Л. Цымблер 13

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.

Указатели и ссылочные типы © М.Л. Цымблер 14

Пример: управление кучей

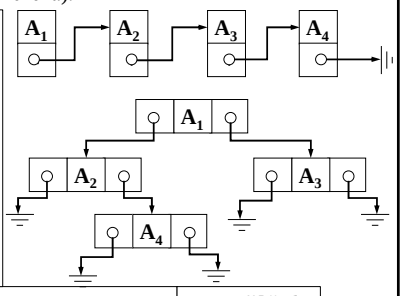
```
function MakeNode(Info: Real): PNode;
var
        P: PNode;
begin
        if MaxAvail<SizeOf(TNode) then
                P := Nil
        else begin
                P := New(PNode);
                P^.Info := Info;
                P^.Next := Nil;
        end;
        MakeNode := P;
end;
```

Указатели и ссылочные типы © М.Л. Цымблер 15

Применение ссылочных типов

- Ссылочные типы применяются в основном *для реализации списковых и древовидных структур* с большим количеством элементов (общий размер больше, чем могут вместить сегменты данных и/или стека).

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



Указатели и ссылочные типы © М.Л. Цымблер 16