# 16. Указатели (Pointers).

Для всех элементов программы – переменных, констант, а также для кода самой программы, компилятор выделяет определенную область в оперативной памяти, и эта область составляет 65536 байта. Этого явно недостаточно для обработки больших массивов данных. С другой стороны, объем оперативной памяти гораздо больше и определяется характеристиками данного компьютера. Турбо-Паскаль предоставляет возможность доступа к этой части ОП с помощью т.н. *динамических структур данных.* Эти структуры используются также для хранения временных промежуточных данных сложной структуры, особенно при работе с графическими и звуковыми средствами компьютера. Кроме того, динамические структуры позволяют различные организации данных, когда их размерность или область значений предварительно не определены.

В отличие от статической памяти, которая распределяется компилятором, распределение динамической памяти осуществляется непосредственно программистом. При этом, ни тип, ни количество данных не известны наперед, и к ним невозможно обращаться по имени (как в случае статических элементов программы), а только с помощью указания адреса их размещения. Эта возможность обеспечивается с помощью динамического структурированного типа данных — *указателя (Pointer).* Фактически указатель представляет адрес памяти, где хранится связанная с ним величина. Переменная типа «Pointer» в качестве своего значения содержит адрес байта оперативной памяти, где размещены соответствующие данные.

Адресация ОП организована следующим образом: адрес задаётся двумя словами (типа «Word» по 16 битов в каждой) – *сегмента* и *смещения.* Минимальная длина сегмента 65536 байта и соответствующий участок памяти начинается с адреса кратного 16. Смещение определяет количество байтов, которые нужно отсчитать от начала сегмента до нужного адреса. Следовательно, сегмент определяет адрес с точностью до 16 байтов, а смещение – с точностью до байта. Внутренняя структура указателя представлена так же совокупностью двух слов – сегмента и смещения. Переменная–указатель содержит адрес первого из последовательности байтов, в которой размещается

содержимое структурных данных, присоединенных к данному указателю.

## *16.1. Типы указателей и их объявление.*

В Турбо-Паскале поддерживаются два вида указателей – *типизированные* и *нетипизированные*. Типизированный указатель используется для представления адреса переменной или элемента программы определенного типа и объявляется с помощью символа «^»:

```
p: ^ Тип;
```

Например,

```
Type
    intPointer = ^integer;
    chainRecs = Record a,b,c:Integer;
                       next: ChainRecs
            End;
Var rpr: ^Real;
    ipr: int_pointer;
    recpr: chainRecs;
```

Нетипизированный указатель не связан с определенным типом и объявляется с помощью ключевого слова «Pointer»:

```
pp:Pointer;
```

Нетипизированные указатели очень удобны для размещения и манипулирования динамическими данными, точная структура и природа которых либо наперед неизвестна, либо меняется в процессе выполнения программы. Нетипизированные указатели предоставляют больше свободы при формировании программ. Например, с целью страховки в Турбо-Паскале не допускается передача значения между указателями, связанными с различиными типами данных, но с помощью нетипизированного указателя можно обойти это препятствие. Пусть

```
Var rpr : ^REAL;
          ipr: ^Integer;
          pp:Pointer;
```

Присвоение

```
ipr:=rpr;
```

недопустимо, но его можно реализовать с помощью нетипизированного указателя «pp» следующим образом:

```
pp:=rpr;
ipr:=pp;
```

## *16.2. Манипулирование динамической памятью.*

Как можно использовать и что можно делать с помощью указателей? Перечислим основные действия, которые можно реализовать на указателях:

Выделение, зарезервирование нового адреса;
Запись информации по этому адресу;
Освобождение адреса;
Стирание информации по адресу.

Эти возможности позволяют манипулировать данными, структура которых создается и модифицируется по ходу выполнения программы. Это обстоятельство является большим преимуществом относительно статических структур данных, когда эта структура должна быть зафиксирована наперед, а в случае малейших изменений программа должна быть переписана заново.

Насколько сложными структурами можно манипулировать, зависит от сложности самой структуры данных, для которых выделяется память, а также от возможности программных средств реализовать их обработку.

С помощью указателей доступна для использования вся не занятая оперативная память компьютера, которая называется «куча» (Heap). Информация о занятой и свободной областях ОП хранится в переменных: *HeapOrg* (адрес начала кучи, т.е. свободного пространства), *FreePtr* (адрес конца кучи), *HeapPtr* (адрес границы незанятой текущей программой динамической области):

```
Var HeapOrg, HeapPtr, FreePtr : Pointer;
```

Кроме того, определены две функции, выдающие информацию о динамической памяти, это

**MemAvail:LongInt** – объем общей свободной области в байтах, и

**MaxAvail:LongInt** – объем наибольшей непрерывной свободной области в байтах.

## 16.2.1. Типизированные указатели.

*Выделение памяти* для хранения значения переменной любого типа осуществляется с помощью процедуры

*New(tpr);*

где *tpr* — типизированный указатель. Рассмотрим фрагмент:

```
1. Type intPointer = ^integer;
2. Var tpr: intPointer;
3.     i:Inteqer;
4. Begin
5.                ...
6.       New(tpr);
7.       tpr^=10;
8.       i:= tpr^ + 10;
9. ...
10. End.
```

В первой строке объявляется тип — указатель для представления адреса локации целочисленной величины, а во второй строке — переменная *tpr* данного типа.

В третьей строке объявляется целочисленная переменная *i*.

В шестой строке определяется значение переменной *tpr* (по величине *HeapPtr),* и в оперативной памяти выделяется место, в данном случае 2 байта, для записи целого числа. На этом этапе память только зарезервирована, она заполняется только после выполнения инструкции в седьмой строке. Как только место в динамической памяти выделено, значение переменной *HeapPtr* увеличивается на 2 (так как для представления целочисленной переменной требуется 2 байта памяти).

В седьмой строке происходит заполнение участка памяти, отведенного под *tpr,* и туда записывается целое число 10.

В восьмой строке в вычислениях используется целое число, которое хранится в ОП по адресу *tpr* и к которому можно обратиться по имени соответствующего указателя, следуемого символом «^», т.е. *tpr^*.

## СХЕМАТИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ РАСПРЕДЕЛЕНИЯ ПАМЯТИ.

После того, как данные, записанные в ОП, обработаны, место можно *освободить* с помощью процедуры

**Dispose(tpr);**

в результате которой системе возвращается память, до тех пор отведенная для переменной *tpr.* При этом значение самой переменной *tpr* не изменяется.

Рассмотрим еще две процедуры для манипулирования типизированными указателями:

**Mark(tpr)** – в переменной *tpr* запоминает, фиксирует адрес текущей границы кучи между занятой и свободной областями, т.е. текущее значение переменной *HeapPtr.*

**Release(tpr)** – освобождает, стирает оперативную память, начиная с адреса аргумента *tpr*.

### 16.2.2. Нетипизированные указатели.

При работе с нетипизированными указателями, которые создают или освобождают часть кучи, приходится реально указывать величину требуемой памяти:

**GetMem(p:Pointer, size: Byte);** аналог **New**
**FreeMem(p:Pointer, size:Byte);** аналог **Dispose**

Следующие две функции возвращают части адреса соответственно — сегмент и смещение:

**Seg(x):Word;**
**Ofs(x): Word;**

Следующая функция образует адрес — значение переменной типа pointer из сегмента и смещения:

**Ptr(seg, ofs:Word):Pointer;**

Функция **Addr(x):Pointer;** возвращает адрес переменной *x* в памяти.

Функции **CSeg:Word; DSeg:Word;** возвращают значения регистров **CS** и **DS** соответственно.

Функция SizeOf(x):Byte; возвращает величину памяти, занимаемой переменной *x* в памяти.