Динамические переменные

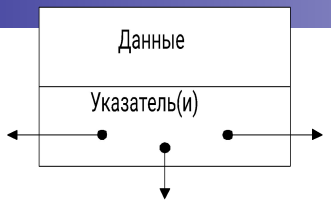
Модули

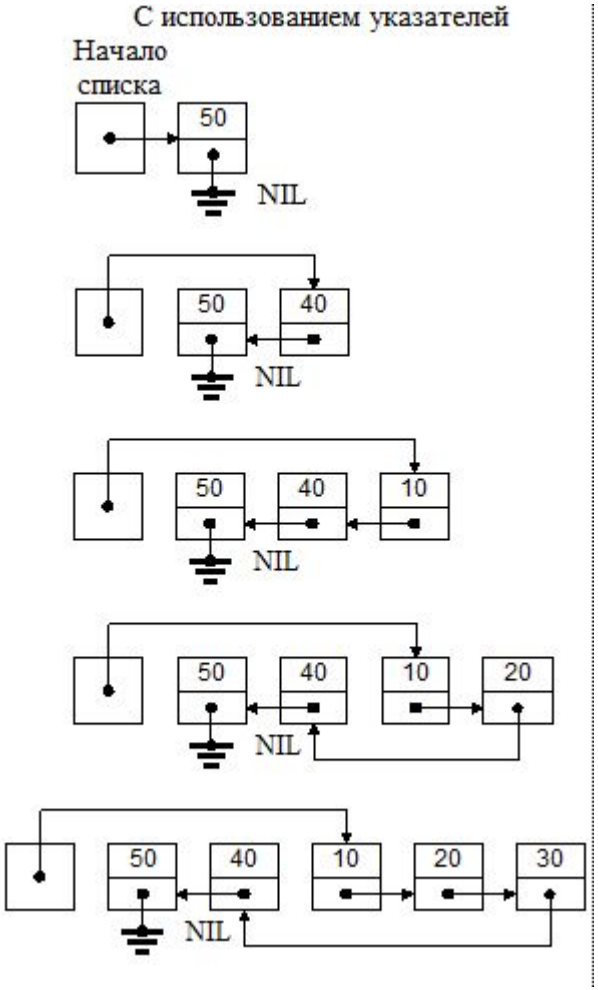
- В языке Паскаль, как практически во всех современных языках высокого уровня, есть переменные, которые создаются и уничтожаются в процессе выполнения программы. Они не входят в явные описания программы и, следовательно, к ним нельзя обращаться с помощью имен. Память для них выделяется только динамически в ходе работы программы.
- Доступ к динамическим переменным осуществляется с помощью указателей (или ссылок), которые становятся определенными после создания динамического объекта.
- Динамические переменные очень широко используется в программировании, более того, современные операционные системы имеют функции поддержки динамических областей памяти.
- Динамические переменные используются при заранее неизвестном количестве элементов обрабатываемой информации: строк текста, таблиц, больших матриц данных и т. п.

Указатели

- Тип указателя сам по себе не является динамической структурой данных, часто его называют **ссылочным типом**. *Это обычное четырехбайтное число, содержащее адрес, с которого начинается динамическая переменная*. **Адрес** в архитектуре процессоров фирмы Intel – два целых беззнаковых числа, определяющие номер сегмента памяти и смещение внутри этого сегмента. Указатели используются для установления отношений или связей между динамическими структурами данных. Эти связи могут быть весьма сложными.

Узел





- При наличии одного указателя, показывающего на следующие данные, структура называется списком (в математике это является связанным списком). В общем случае в узле может содержаться несколько указателей, тогда структуру называют деревом (от генеалогического дерева). Список же является вырожденным деревом. Кроме того, на каждый узел может указывать произвольное число указателей.
- В стандартном языке Паскаль указатели должны ссылаться на *однотипные элементы* данных, то есть являются типизированными. Существует специальное значение указателя **NIL** (**пустой указатель**), принадлежащее всем типам указателей. В этом случае указатель не указывает ни на какой элемент. Это значение применяется для обозначения конца списка или ветви дерева (по аналогии в функцией EOF).
- Пример общего случая использования динамической структуры: ввод данных, располагаемых в порядке возрастания: 50, 40, 10, 20, 30.

Описание указателей:

Type <имя_указателя>=^<базовый_тип>

Например:

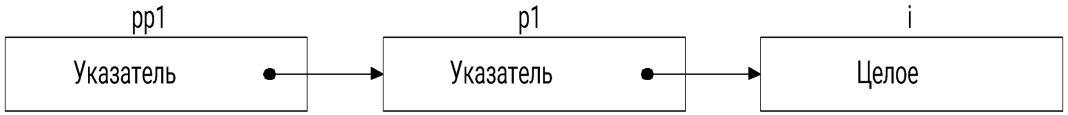
```
Var  z:^real;
Type poin=^T;
  T=Record
      str:string;
      pnt:poin;
  end;
Var  p,q:Poin;
```

Здесь p и q – указатели на переменную типа T (запись), чтобы обратиться к самой переменной, записывается p^, q^.

- **Замечание 1.** Правило языка Паскаль: имя любого типа сначала должно быть определено, и лишь затем использовано. Единственное исключение – при определении ссылочного типа можно использовать имя типа, который определяется далее в тексте программы.
- **Замечание 2.** В программах, использующих указатели, обычно нельзя обойтись без раздела описания типов, поскольку при определении ссылочного типа должно использоваться имя базового типа, как правило сложного, например записи.

Если необходимы указатели на различные типы элементов, то должны быть и различные типы указателей, то есть каждый указатель может ссылаться только на элементы своего типа.

- Для того чтобы присвоить переменной ссылочного типа определенное значение, необходимо воспользоваться **операцией взятия адреса (указателя)**, – символа амперсанд «@» и переменной базового типа, например:
- z:=@x;
- Ссылочные типы можно образовывать от любых других типов, поэтому допустимо определение вида «указатель на указатель». Например, фрагмент программы:
- Type p1=^integer;
- Var pp1:^p1;
- i:integer;
- Begin
- p1:=@i;
- pp1:=@p1;
- приводит к следующим связям:



Операции

- Над значениями ссылочных типов допускаются только две операции сравнения на равенство и неравенство. Они проверяют, ссылаются ли два указателя на одно и то же место в памяти, или нет, например:
- Sign:=P1=P2;
- If P1<>nil then ...

Работа с динамическими переменными

Доступ, то есть обращение к динамическим переменным, возможен по двум схемам.

Рассмотрим ситуацию, возникшую после присваивания:

p1:=@i;

Первый вариант очевиден:

i:=i+2;

Но, хотя на *i* ссылается указатель, переменная описана как статическая.

Для реализации косвенного доступа к переменной через указатель используется **разыменование**. То есть, чтобы по указателю получить доступ к переменной, необходимо после указателя поставить знак «^». Так, *запись p1^ означает: «переменная, на которую ссылается p1».* Поэтому операторы

i:=i+2 и p1^:=p1^+2

полностью эквивалентны. Разыменование имеет тип, совпадающий с базовым типом, в частности p1^ является переменной целого типа (по описанию выше).

Разыменование допустимо для любых ссылочных типов, например, возможны следующие конструкции:

j:=pp1^^;

f:=h.l^.g^.p;

Основные действия над динамическими переменными

— это их создание и уничтожение.

- Первое реализуется стандартной процедурой

New (<указатель>)

При выполнении этой процедуры происходят следующие действия:

В динамической области памяти отводится место для переменной по размеру базового типа указателя.

Указателю присваивается адрес этой переменной.

- Пример.

```
Var I,J: ^integer;  { определяем два указателя на целую переменную }
NEW (J);  { выделяем память для целой переменной и присваиваем ее адрес указателю }
I:=J;     { I и J указывают на одну и ту же переменную (J:=I —запрещено) }
J^:=5;    { запись числа в выделенную память }
I^:=6;    { стало и J^=6 }
I:=NIL;   { I ни на что не указывает }
J:=NIL;   { доступа к переменной больше нет, она не нужна }
```

- Стандартная функция **MaxAvail** без параметров, возвращает максимальный размер непрерывного участка свободной памяти, пригодного для размещения переменной.
- В общем случае для структурированных типов при определении размера выделяемой памяти можно использовать функцию **SizeOf** (определение размера переменной):

```
If  MaxAvail >= SizeOf (TypeDat) then ...
```

- Для освобождения памяти, выделенной под динамическую переменную, используется процедура, обратная по действию процедуре New:

Dispose (<указатель>);

Указатели без типа

Турбо Паскаль содержит стандартный ссылочный тип, который позволяет не конкретизировать базового типа и считается совместимым со всеми ссылочными типами.

- Var <имя_указателя> : pointer;

Работа с ними подразумевает их преобразование в указатели, ссылающиеся на значения конкретного типа, например, при необходимости организовать список, состоящий из записей различных типов.

Для создания и удаления переменных с такими указателями соответственно используются процедуры:

- **GetMem** (<указатель>,<размер>);
- **FreeMem** (<указатель>,<размер>);

Размер участка памяти указывается в байтах, обычно с применением функции SizeOf:

- **GetMem** (Ptr,**SizeOf**(R));

. Модули

- **Модуль** – это совокупность или коллекция программных ресурсов, предназначенных для использования другими программами или модулями. **Ресурсы** – это любые программные объекты Паскаля, – константы, переменные, типы, подпрограммы.
- Основное отличие модуля от программы заключается в том, что его объекты только используются другими программами, но сам модуль выполнить нельзя.
- Все программные ресурсы модуля делятся на две части:
- 1). Интерфейс. Здесь находятся видимые объекты, то есть те, которые можно использовать в других программах, и предназначенные именно для этих целей.
- 2). Реализация. Сюда помещают рабочие объекты, называемые невидимыми или скрытыми. Например, если модуль содержит подпрограмму универсального применения, то вызываемые этой подпрограммой процедуры и функции, содержащиеся в модуле, и используемые ею переменные могут иметь чисто внутренний характер и не должны использоваться другими программами.

Таким образом, общая структура модуля следующая:

- **UNIT** <имя модуля>
- **Interface**
- <описание видимых объектов>
- **Implementation**
- <описание скрытых объектов>
- [Begin {Initialization}
- <операторы инициализации>
- { Finalization
- <завершающие операторы> }]
- end.

Стандартные модули

Обычно все используемые модули находятся в текущем каталоге или в системном библиотечном файле **TURBO.TPL (Turbo Pascal Library)**. В этот файл можно добавлять и свои модули, но в стандартном варианте там находятся 5 модулей, содержащих все системные константы, типы, процедуры и функции:

- SYSTEM,
- DOS,
- CRT,
- PRINTER,
- OVERLAY.

Остальные модули размещаются в отдельных файлах TPU.

- GRAPH,
- STRINGS,
- WINDOS,
- TURBO3 и GRAPH3