Комитет по образованию Администрации г. Подольска
Муниципальное образовательное учреждение «Лицей № 26»

«Методика решения и составления задач на Паскале с использованием динамических типов данных»

Кривко-Красько Сергей Васильевич,
учитель информатики

2013

Введение

Изучение основ программирования в средней школе, как на базовом, так и на профильном уровне (1) включает изучение лишь статических типов данных. В то же время в профессиональном программировании широко используются динамические типы данных. Поэтому знакомство с ними наиболее успешных учащихся, представляется необходимым. Следует также учесть, что нередко олимпиадные задачи по информатике эффективно решаются именно с использованием динамических типов.

Статические данные размещаются в памяти на стадии компиляции и перераспределения памяти на стадии выполнения не допускается. Это создавало некоторые неудобства при работе с массивами (приходилось объявлять максимальный размер памяти под массивы). Выделение памяти для переменных на стадии выполнения программы стало возможным только при использовании нового типа данных - указателей (ссылок). Значением указателя (переменной ссылочного типа) является не значение переменной (величины), а адрес области памяти (первой ячейки) в которой находится переменная.

Для указателей область памяти выделяется статически (как обычно), а для переменных, на которые они указывают, - динамически, то есть на стадии выполнения программы. Для хранения динамических переменных выделяется специальная область памяти, называемая "кучей".

Для объявления указателей в ЯП Паскаль используется специальный символ "^", после которого указывается тип динамической (базовой) переменной:

type <имя_типа> = ^ <базовый тип>;

var <имя_переменной>: <имя_типа>; или

<имя_переменной>: ^ <базовый тип>;

Например:

```
type int = ^integer;
var x,y: int;        {*Указатель на переменную целого типа*}
    a: ^real;    {*Указатель на переменную вещественного типа*}
```

Выделение оперативной памяти (в «куче») для динамической переменной

базового типа осуществляется с помощью оператора New(x), где x - соответствующий указатель.

Обращение к динамическим переменным выполняется так:

<имя_переменной>^

Пример: x^:=15

В процессе выполнения число 15 (два байта) записывается в область памяти, адрес которой является значением указателя x.

Оператор Dispose(x) освобождает память, занятую динамической переменной, при этом значение указателя x становится неопределенным.

Зарезервированное слово nil обозначает константу ссылочного типа, которая ни на что не указывает.

Основные зарезервированные слова и операторы для работы с динамическими переменными Оператор (зарезервированное слово) назначение

New(x) выделение оперативной памяти для динамической переменной

Dispose(x) освобождает память, занятую динамической переменной

nil константа ссылочного типа, которая ни на что не ссылается (записывается в последнем элементе списка)

x^:=...оператор записи значения в выделенную ячейку памяти

Линейный список

Наиболее эффективно использование указателей при работе с так называемыми динамическими структурами данных.

К динамическим структурам относятся:

- линейный списк;
- стек;
- очередь;
- граф;
- дерево.

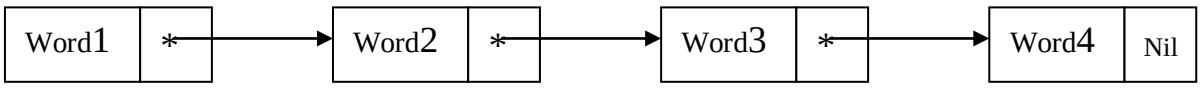
Рассмотрим применение указателей

Списком называется структура данных, каждый элемент которой при помощи указателя связан со следующим элементом. Каждый элемент списка, как правило содержит два поля: поле для хранения данных (содержательная часть) и поле для ссылки на следующий элемент списка.

Поле ссылки последнего элемента всегда имеет значение nil. Поле данных может иметь сложную структуру. Для работы с примерами, обозначим поле данных как word и тип хранимых данных - string, а поле для ссылки на другой элемент списка обозначим как next.

Указатель на начало списка является отдельной переменной.

Пример линейного списка.



Основные операции со списком:

- вывод элементов списка;

- вставка элемента в список;

- удаление элемента из списка.

Создание упорядоченного списка

Рассмотрим следующую задачу: создание упорядоченного списка (упорядочение производится по содержательной части элементов списка, а так как в примере содержательная часть списка – текст, то тексты упорядочиваются в алфавитном порядке – по возрастанию)

Для вставки необходимо найти элемент, содержательная часть которого больше содержательной части нового элемента. Отдельно надо рассмотреть ситуации, когда новый элемент приходится вставлять в начало или конец списка.

В начале программы надо описать типы и переменные:

```
program Orded_List;
type  str10=string[10];
      pt=^element;   (*Указатель на элемент списка*)
      element=Record
      word:str10; (*текстовое поле*)
      next:pt;  (*указатель на следующий элемент*)
      end;
var  first:pt; (*Указатель на первый элемент списка*)
     w:str10; (Переменная для ввода текста*)
```

Для вывода элементов списка на экран понадобится процедура:

```
Procedure List_write(x:pt);
  begin
    while x<>Nil do
       begin
          writeln(x^.word);
          x:=x^.next;
       end;
  end;
```

И самая главная процедура – вставка элемента в упорядоченный список:

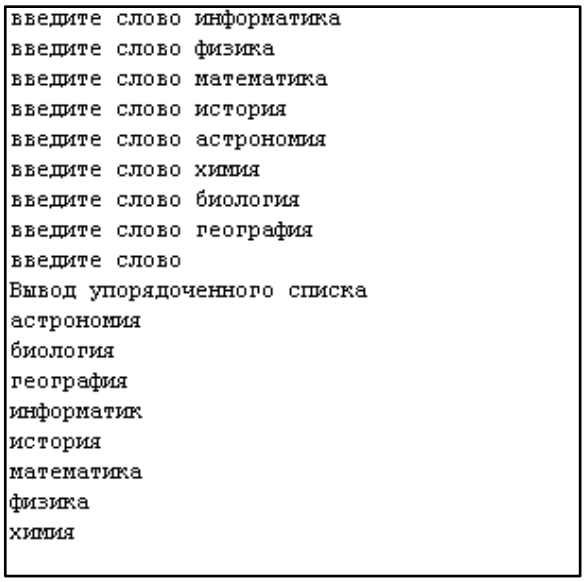
```
Procedure List_Insert(var first:pt;w:str10);
  Var new_x, x:pt;
  begin
    New(new_x); new_x^.word:=w;
 (*Отвели место в памяти элементу и записали в него текст*)
  if first=Nil then  (*если список был изначально пуст*)
     begin
        first:=new_x; new_x^.next:=ni;
         (*сделали новый элемент первым*)
     end
  else
     if w<first^.word then
(*если новый элемент должен быть перед 1-ым элементом*)
       begin
         new_x^.next:=first;     first:=new_x;
(*сделали новый элемент первым *, его ссылка – на тот, что был
первым*)
       end
     else
(*ищем место, куда надо вставить новый элемент*)
       begin
          x:=first;
          while (x^.next<>Nil) and (w> x^.next^.word)  do
             x:=x^.next;
          new_x^.next:=x^.next; x^.next:=new_x;
       end;
  end;
```

И основная часть программы:

```
begin
    first:=Nil;
    repeat
       write(Введите текст '); readln(w);
       if w<>' ' then List_Insert(first,w);
    until w=' '; (* выход из цикла по вводу пробела*)
    Writeln(Вывод упорядоченного списка');
    List_write(first);
end.
```

Программа выполнена в PascalABC.

Скриншот окна результатов:



Список литературы

1. *Программы для общеобразовательных учреждений: Информатика. 2-11 классы / Составитель М.Н.Бородин.* М. : БИНОМ. Лаборатория знаний, 2007. стр. 448.

2. **Окулов С.М.** *Основы программирования.* М. : БИНОМ. Лаборатория знаний, 2004. стр. 424.

3. **Иванова Г.С.** *Основы программирования.* М. : Из-во МГТУ им. Н.Э.Баумана, 2001. стр. 392.