Практическое занятие № 10 «Динамические структуры данных»

Учебные цели:

- получение умений и навыков работы с динамическими структурами данных.

Воспитательные цели:

- воспитание познавательного интереса, активности, целеустремленности, настойчивости, активности, наблюдательности, интуиции, сообразительности;
- формировать диалектико-материалистическое мировоззрение;
- формировать навыки самостоятельности и дисциплинированности;
- стимулировать активную познавательную деятельность обучаемых, способствовать формированию у них творческого мышления.

Категория слушателей: 2,3 курс.

Время: 90 мин.

Место проведения: дисплейный класс.

Материально-техническое обеспечение:

1) персональный компьютер *IBM PC* с операционной системой Windows XP; 2) среда разработки приложений *Visual C++.NET*.

ПЛАН ЛАБОРАТОРНОГО ЗАНЯТИЯ

Учебные вопросы	Время, мин
Вступительная часть	5
1. Основные сведения о ссылочном типе данных	15
2. Линейные списки, изучение основных операций	65
3. Выполнение индивидуального задания	5
Заключительная часть	

Элементы теории

Абстрактные структуры данных предназначены для удобного хранения и доступа к информации. Они предоставляют удобный интерфейс для типичных операций с хранимыми объектами, скрывая детали реализации от пользователя. Это весьма удобно и позволяет добиться большей модульности программы. Абстрактные структуры данных иногда делят на две части: *интерфейс*, набор операций над объектами, который называют АТД (*абстрактный тип данных*) и

реализацию. Языки программирования высокого уровня(Паскаль, Си..) предоставляют удобный интерфейс для чисел: операции +, *, = .. и т.п, но при этом скрывают саму реализацию этих операций, машинные команды.

Статические структуры относятся к разряду непримитивных структур, которые, фактически, представляют собой структурированное множество примитивных, базовых, структур. Например, вектор может быть представлен упорядоченным множеством чисел. Поскольку по определению статические структуры отличаются отсутствием изменчивости, память для них выделяется один раз и ее объем остается неизменным до уничтожения структуры.

Динамические структуры по определению характеризуются отсутствием физической смежности элементов структуры в памяти непостоянством и непредсказуемостью размера (числа элементов) структуры в процессе ее обработки.

Поскольку элементы динамической структуры располагаются по непредсказуемым адресам памяти, адрес элемента такой структуры не может быть вычислен из адреса начального или предыдущего элемента. Для установления связи между элементами динамической структуры используются указатели, через которые устанавливаются явные связи между элементами. Такое представление данных в памяти называется связным. Элемент динамической структуры состоит из двух полей:

- *информационного поля* или *поля данных*, в котором содержатся те данные, ради которых и создается структура; в общем случае информационное поле само является интегрированной структурой - вектором, массивом, другой динамической структурой и т.п.;
- *поле связок*, в котором содержатся один или несколько указателей, связывающий данный элемент с другими элементами структуры;

Когда связное представление данных используется для решения прикладной задачи, для конечного пользователя «видимым» делается только содержимое информационного поля, а поле связок используется только программистом-разработчиком.

Достоинства связного представления данных − в возможности обеспечения значительной изменчивости структур;

- размер структуры ограничивается только доступным объемом машинной памяти;
- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
- большая гибкость структуры.

Вместе с тем связное представление не лишено и *недостатков*, основные из которых:

- на поля связок расходуется дополнительная память;

- доступ к элементам связной структуры может быть менее эффективным по времени.

Последний недостаток является наиболее серьезным и именно им ограничивается применимость связного представления данных. Если в смежном представлении данных для вычисления адреса любого элемента нам во всех случаях достаточно было номера элемента и информации, содержащейся в дескрипторе структуры, то для связного представления адрес элемента не может быть вычислен из исходных данных. Дескриптор связной структуры содержит один или несколько указателей, позволяющих войти в структуру, далее поиск требуемого элемента выполняется следованием по цепочке указателей от элемента к элементу. Поэтому связное представление практически никогда не применяется в задачах, где логическая структура данных имеет вид вектора или массива - с доступом по номеру элемента, но часто применяется в задачах, где логическая структура требует другой исходной информации доступа (таблицы, списки, деревья и т.д.).

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется *линейным*. *Длина списка* равна числу элементов, содержащихся в списке, список нулевой длины называется пустым списком. Линейные связные списки являются простейшими динамическими структурами данных.

Графически связи в списках удобно изображать с помощью стрелок. Если компонента не связана ни с какой другой, то в поле указателя записывают значение, не указывающее ни на какой элемент. Такая ссылка обозначается специальным именем - nil.

На рис. 1 приведена структура *односвязного списка*. На нем поле INF - информационное поле, данные, NEXT - указатель на следующий элемент списка. Каждый список должен иметь особый элемент, называемый указателем начала списка или головой списка, который обычно по формату отличен от остальных элементов. В поле указателя последнего элемента списка находится специальный признак nil, свидетельствующий о конце списка.

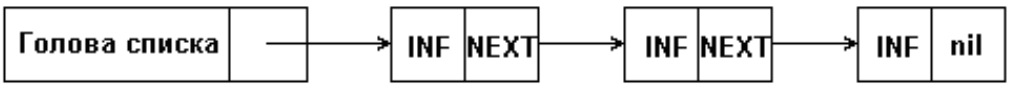


Рис. 1: Представление односвязного списка в памяти

Двусвязный список характеризуется наличием пары указателей в каждом элементе: на предыдущий элемент и на следующий (рис. 2).

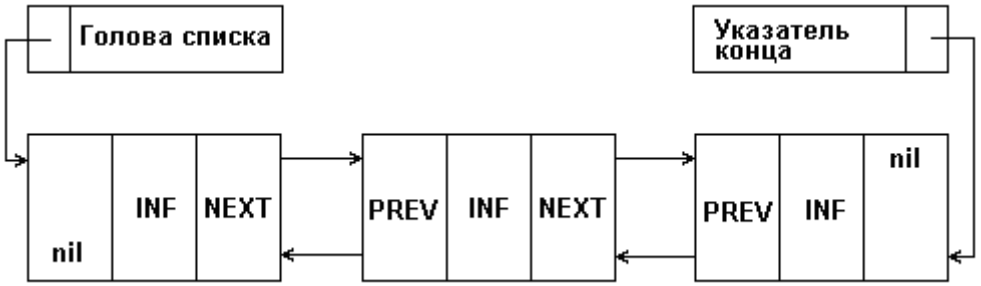


Рис. 2: Представление двусвязного списка в памяти

Очевидный плюс в том, что от данного элемента структуры мы можем пойти в обе стороны. Таким образом упрощаются многие операции. Однако на указатели тратится дополнительная память.

Выделим *типовые операци*и над списками:

- добавление звена в начало списка;
- удаление звена из начала списка;
- добавление звена в произвольное место списка, отличное от начала (например, после звена, указатель на которое задан);
- удаление звена из произвольного места списка, отличного от начала (например, после звена, указатель на которое задан);
- проверка, пуст ли список;
- очистка списка;
- печать списка.

Реализуем выделенный набор операций в виде отдельных модулей на C++. Подключив эти модули, можно решить большинство типовых задач на обработку списка. Пусть список объявлен следующим образом:

```
struct node
     {  int info;
       struct node *next;
     };
typedef node *NodePtr; // указатель на тип node - шаблон
NodePtr head = NULL; // указатель на голову списка
```

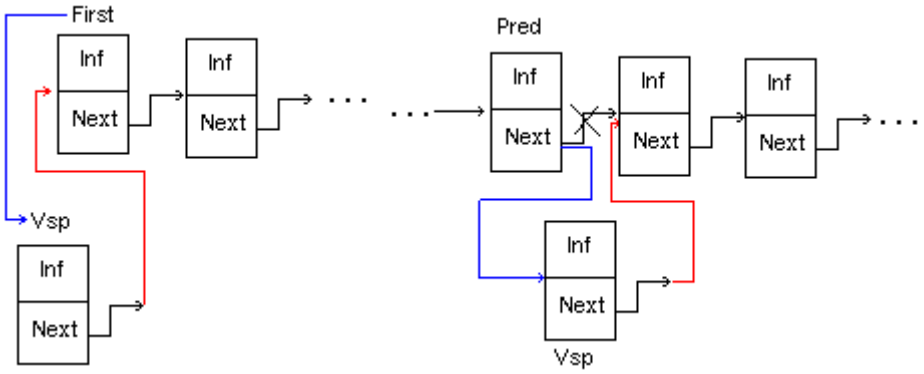
Первые четыре действия сначала реализуем отдельно, снабдив их иллюстрациями.

1. Добавление звена в начало списка (рис. 3а).

```
NodePtr v,p;
. . .
v = new node;
v->info = 0;
v->next = head->next;
head = v;
```

2. Добавление элемента в середину списка (рис. 3б).

```
NodePtr v, p;
. . .
v = new node;
v->info = 3;
v->next = p->next;
p->next = v;
```



а) б)

Рис. 3. Добавление звена в начало и середину списка

3. Удаление звена из начала списка (рис. 4а).

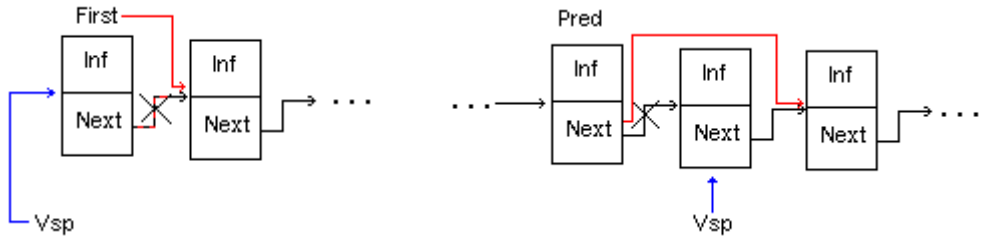
```
NodePtr d; // указатель на удаляемый элемент
. . .
d = head;
head = head->next;
delete d;
```

4. Удаление элемента из середины списка (рис. 4б).

```
NodePtr d; // указатель на удаляемый элемент
. . .
p->next = d->next; // фактически это p->next = NULL
delete d;
```

5. Удаление элемента с конца списка

```
NodePtr d; // указатель на удаляемый элемент
. . .
p->next = d->next; // фактически это p->next = NULL
delete d;
```



а) б)

Рис. 4. Удаление звена из начала и середины списка

Вопросы для самопроверки

1. Для чего применяются абстрактные структуры данных?
2. Статические структуры.
3. Динамические структуры.
4. Перечислите достоинства и недостатки связного представления данных.
5. Что называется списком?
6. Виды списков.
7. Как в С++ описывается линейный список?

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОТРАБОТКЕ УЧЕБНЫХ ВОПРОСОВ

Задание 1. В соответствии с Вашим вариантом составьте схему алгоритма, напишите и отладьте программу для формирования обработки динамического списка, считая, что длина списка задана.

Номер варианта	Задание
1.	Используя динамическую структуру список, подсчитать количество цифр в заданном наборе символов. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
2.	Найти сумму четных элементов списка, состоящего из не менее чем из двух элементов. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
3.	Используя динамическую структуру список, подсчитать сумму чисел кратных 10 в списке. Записать их в отдельный динамический массив. Полученный массив вывести на экран.

4.	Используя динамическую структуру список, подсчитать количество четных чисел в заданном наборе символов. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
5.	Используя динамическую структуру список, подсчитать произведение отрицательных чисел в списке. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
6.	Сформировать список из элементов целого типа. Четные элементы возвести в квадрат. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
7.	Используя динамическую структуру список, подсчитать количество элементов лежащих в диапазоне от 20 до 50. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
8.	Используя динамическую структуру список, увеличить в 2 раза нечетные числа. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
9.	Используя динамическую структуру список, подсчитать произведение элементов массива с номерами 3, 6, 9, 12 и т.д. Записать их в отдельный динамический массив. Полученный массив вывести на экран.
10.	Используя динамическую структуру список, проверить является ли он упорядоченным по возрастанию набором чисел. Если нет, то сформировать динамический массив и записать в него начало списка, которое является упорядоченной по возрастанию. Например, список: 1, 2, 3, 4, -8, 6, 7, 35, динамический массив: 1, 2, 3, 4.

Пример выполнения задания 1.

Создать список из 50 элементов и заполнить его случайными символами. Узнать сколько в полученном списке английских строчных букв и записать их в отдельный динамический массив. Вывести полученный массив на экран.

Листинг 1.

```
#include "stdafx.h"
#include <iostream>
//#include <stdlib.h>  RAND_MAX (32767)
// для изменения диапазона изменения случайных чисел
#include <ctime>
using namespace std;
int main()
```

```
{
     struct node
     { unsigned char info;
     struct node *next;
     };
     srand(time(0));// без этого числа будут одинаковые
     //srand - Sets a random starting point.
typedef node *NodePtr; // указатель на тип node - шаблон
NodePtr head = NULL;
NodePtr tek; // указатель на текущий элемент
NodePtr hvost; // указатель на "хвост" списка
int N = 50; // количество элементов в списка
int cnt = 1; // счетчик элементов в списка
if (head == NULL)
{ head = new node;
head->info = char(rand()%255); // случайный символ
head->next = NULL;
hvost = head;
}
for (int i = 2; i<=N; i++)
{ tek = new node;
++cnt;
tek->info =char(rand()%255);
tek->next=NULL;
hvost->next=tek;   // в данном случае - NULL
 hvost = tek;
}
// Вывод списка на экран
cout<<"Spisok simvolov"<<endl;
tek = head;
int *bukva=0;
int *temp=0;
int k=0;
for (int i = 1; i<=N; i++)
{
     cout << tek->info << ' ';
if ((tek->info>=97) && (tek->info<=122))
     {
          k++;
          if (k==1)
          {
          bukva = new int [k];
          bukva[0]=tek->info;
          //создаем дополнительный массив
          temp = new int [k];// на кол-во элементов в массиве с
данными
          temp[0] = bukva[0];
          }
          else
          {
          delete [] bukva; // удаляем массив
```

```
          bukva= new int[k]; // созадем новый массив на 1
больше
          for(int i=0;i<k-1;i++)
          bukva[i]=temp[i];
          bukva[k-1] = tek->info;// добавляем элемент в конец
          delete [] temp;
          temp = new int [k];
          //копируем все данные из массива bukva в массив temp
     for(int i=0;i<k;i++)
     temp[i] = bukva[i];
          }
     }
tek = tek->next;
}
cout<<endl<<"Kol-vo bukv="<<k<<endl;
if (k!=0)
     {
     cout<<"Spisok strochnyh angliyskih bukv"<<endl;
     for(int i=0;i<k;i++)
     cout << char(bukva[i])<<'\t';
     cout<<endl;
     }
delete [] bukva;
delete [] temp;
return 0;
}
```

Контрольные вопросы

Приведите программный код C++, реализующий следующие операции со списками:

1) добавление звена в начало списка;
2) удаление звена из начала списка;
3) добавление звена в произвольное место списка;
4) удаление звена из произвольного места списка;
5) проверка, пуст ли список;
6) очистка списка;
7) печать списка.