Тема 9: ДВУНАПРАВЛЕННЫЕ СПИСКИ

Цель занятия: ознакомление с динамической структурой данных – двунаправленным списком; получение практических навыков в создании и обработке списков.

Теоретические сведения

Двунаправленный список состоит из элементов данных, каждый из которых содержит ссылки как на следующий, так и на предыдущий элементы. На рис. 9.1 показана организация двунаправленного списка.

Рис. 9.1. Двунаправленный список

Построение двусвязного списка выполняется аналогично построению односвязного за исключением того, что необходимо установить две ссылки. Поэтому в структуре данных должны быть описаны два указателя связи.

Описание простейшего элемента двунаправленного списка выглядит следующим образом:

```
struct List                 //описание структуры
{
        int key;            //ключевое поле
        List* pred,*next; //адресные поля
};
```

Наличие двух ссылок вместо одной предоставляет несколько преимуществ. Наиболее важное из них состоит в том, что перемещение по списку возможно в обоих направлениях. Это упрощает работу со списком, в частности, вставку и удаление. Помимо этого, пользователь может просматривать список в любом направлении. Еще одно преимущество имеет значение только при некоторых сбоях. Поскольку весь список можно пройти не только по прямым, но и по обратным ссылкам, то в случае, если какая-то из ссылок станет неверной, целостность списка можно восстановить по другой ссылке.

Рассмотрим **пример**, описывающий создание двунаправленного списка, удаление из списка элемента с заданным номером, добавление элемента с заданным номером, печать полученных списков.

```
#include "stdafx.h"
#include <iostream>
#include<string.h>
#include <conio.h>
using namespace std;
struct List                 //описание структуры
{
        int key;            //поле данных
        List* pred,*next; //адресные поля
};
//формирование списка из n элементов
List*make_list()
{
        int n;
        cout<<"n-?";
        cin>>n;
        List *p,*r,*beg;
        p=new (List);               //создать первый элемент
        beg=p;                      //запомнить адрес в переменную beg, в которой хранится начало списка
        cout<<"key-?";
        cin>>p->key;                //заполнить    поле данных
        p->pred=0;   p->next=0;     //заполнить адресные поля
        for(int i=1;i<n;i++)        /*добавить элементы в конец списка*/
        {
                r=new(List);                //новый элемент
                cout<<"key-?";
                cin>>r->key;                // заполнить    поле данных
                p->next=r;                  //связать начало списка с r
```

```
                r->pred=p;              //связать r с началом списка
                r->next=0;              /*обнулить последнее адресное поле*/
                p=r;                    /*передвинуть p на последний элемент списка*/
        }
return beg;                             //вернуть начало списка
}
/*печать списка, на который указывает указатель beg*/
void print_list(List *beg)
{
        if (beg==0)                     //если список пустой
        {
                cout<<"The list is empty\n";
                 return;
         }
        List*p=beg;
        while(p)                        //пока не конец списка
        {
                cout<<p->key<<"\t";
                p=p->next;              //перейти на следующий
        }
        cout<<"\n";
}

//удаление элемента с номером k
List* del_List(List*beg, int k)
{
        List *p=beg;
        if(k==1)                        //удалить первый элемент
        {
                beg=beg->next;          /*переставить начало списка на следующий элемент*/
                if(beg==0) return   0;  /*если в списке только один элемент*/
                beg->pred=0;            /*обнулить адрес предыдущего элемента */
                delete p;               //удалить первый элемент
                return beg;             //вернуть начало списка
        }
//если удаляется элемент из середины списка
        for(int   i=0;i<k-2 && p!=0; i++,p=p->next); /*пройти по списку либо до элемента с предыдущим
номером, либо до конца списка*/
        if(p==0||p->next==0)return   beg;  //если в списке нет элемента с номером k
        List*r=p->next;                 //встать на удаляемый элемент
        p->next=r->next;                //изменить ссылку
        delete r;                       //удалить r
        r=p->next;                      //встать на следующий
        if(r!=0) r->pred=p;             /*если   r   существует, то связать элементы*/
        return beg;                     //вернуть начало списка
}

//добавление элемента с заданным номером
List* add_List(List *beg,int k)
{
        List *p;
        p=new(List);            /*создать   новый   элемент   и   заполнить поле данных*/
        cout<<"key-?";
        cin>>p->key;
        p->pred=0;              //обнулить адрес предыдущего
        p->next=0;               //обнулить адрес следующего
        if(k==1)                //если добавляется первый элемент
        {
                p->next=beg;    //добавить перед beg
```

```
                beg->pred=p;      //связать список с добавленным элементом
                beg=p;            //запомнить первый элемент в beg
        return beg;               //вернуть начало списка
        }
        List*r=beg;               //встать на начало списка
        for(int  i=0;i<k-2 && r->next!=0; i++,r=r->next);    /*пройти по списку либо до конца
списка, либо до элемента с номером k-1*/
        p->next=r->next;          //связать p с элементом, следующим за вставляемым
        if (r->next!=0) r->next->pred=p;    /*если элемент не последний, то связать элемент, следующий за
вставляемым, с p */
        p->pred=r;                //связать p и r
        r->next=p;
        return beg;               //вернуть начало списка
}
void main()
{
        List*beg;
        int i,k;
        do
        {
        cout<<"1.Make list\n";
        cout<<"2.Print list\n";
        cout<<"3.Add List\n";
        cout<<"4.Del List\n";
        cout<<"5.Exit\n";
        cin>>i;
        switch(i)
                {
                        case 1:
                        {beg=make_list();break;}
                        case 2:
                        {print_list(beg);break;}
                        case 3:
                        {
                                cout<<"\nk-?";cin>>k;
                                beg=add_List(beg,k);
                                break;
                        }
                        case 4:
                        {
                                cout<<"\nk-?";cin>>k;
                                beg=del_List(beg,k);
                                break;
                        }
                }
        }
        while(i!=5);
}
```

```
#include "stdafx.h"
#include <iostream>
#include<string.h>
#include <conio.h>
using namespace std;
struct List             //описание структуры
{
char key;         //поле данных
```

```
   List* pred,*next; //адресные поля
   };
   // построение списка символов до первой точки
   List * Init()
   {
       List *p,*r,*beg;
       char ch;
       cout << "ВВОДИТЕ  ЭЛЕМЕНТЫ СПИСКА"<<endl;
       cin>> ch;
       if (ch!='.')
       {
              p=new (List);        //создать первый элемент
              beg=p;                      //запомнить адрес в переменную beg, в которой хранится
начало списка
              p->key=ch ;          //заполнить   поле данных
              p->pred=0;  p->next=0;      //заполнить адресные поля
              cin>> ch;
              while (ch != '.')
              {
                     r=new(List);         //новый элемент
                     r->key=ch;           // заполнить   поле данных
                     p->next=r;           //связать начало списка с r
                     r->pred=p;           //связать r с началом списка
                     r->next=0;           /*обнулить   последнее   адресное поле*/
                     p=r;
                     cin>> ch;
              }
       }
   return beg;
   }

   //формирование списка из n элементов
   List*make_list()
   {
       int n;
       cout<<"n-?";
       cin>>n;
       List *p,*r,*beg;
       p=new (List);        //создать первый элемент
       beg=p;                      //запомнить адрес в переменную beg, в которой хранится начало
списка
       cout<<"key-?";
       cin>>p->key;         //заполнить   поле данных
       p->pred=0;  p->next=0;      //заполнить адресные поля
       for(int i=1;i<n;i++)/*добавить элементы в конец списка*/
       {
              r=new(List);         //новый элемент
              cout<<"key-?";
              cin>>r->key;         // заполнить   поле данных
              p->next=r;           //связать начало списка с r
              r->pred=p;           //связать r с началом списка
              r->next=0;           /*обнулить   последнее   адресное поле*/
              p=r;                 /*передвинуть  p  на   последний   элемент списка*/
       }
       return beg;                        //вернуть начало списка
   }

   /*печать списка, на который указывает указатель beg*/
   int print_list(List *beg) //функция возвращает количество элементов списка
   {
       int n=0;
```

```
        if (beg==0)                //если список пустой
        {
               cout<<"The list is empty\n";
               return n;
        }
        List*p=beg;
        while(p)                          //пока не конец списка
        {
               cout<<p->key;
               n++;
               p=p->next;          //перейти на следующий
        }
        cout<<"\n";
        return n;
    }
    List* del1_List(List*beg)
    {   List *p=beg, *begin, *end;
        int count = 0;
        while(p)
        {
               while (p->key == ' '&& p)         p=p->next;                          //пропустить
пробелы
               begin = p;                         // номер первого символа слова
               while (p->key != ' ' && p)        p=p->next;                                  //
пропустить все символы слова
               end = p;     // номер символа,  следующего за последним символом слова
               while(begin->next!=end)                          //пока не конец списка
               {
                      cout<<begin->key;
                      begin=begin->next;         //перейти на следующий
               }
               cout<<"\n";
        }
        return beg;
    }
    //удаление элемента с номером k
    List* del_List(List*beg, int k)
    {
    List *p=beg;
    if(k==1)                          //удалить первый элемент
    {
    beg=beg->next;           /*переставить   начало   списка на следующий элемент*/
    if(beg==0) return  0;    /*если в списке только один элемент*/
    beg->pred=0;             /*обнулить   адрес   предыдущего элемента */
    delete p;                //удалить первый элемент
    return beg;              //вернуть начало списка
    }
    //если удаляется элемент из середины списка
    for(int  i=0;i<k-2 && p!=0; i++,p=p->next); /*пройти по списку либо до элемента с предыдущим
номером, либо до конца списка*/
    if(p==0||p->next==0)return  beg;      //если в списке нет элемента с номером k
    List*r=p->next;                //встать на удаляемый элемент
    p->next=r->next;               //изменить ссылку
    delete r;                      //удалить r
    r=p->next;               //встать на следующий
    if(r!=0) r->pred=p;            /*если  r  существует, то связать элементы*/
    return beg;                    //вернуть начало списка
    }

    //добавление элемента с заданным номером
    List* add_List(List *beg,int k)
```

```
   {
   List *p;
   p=new(List);            /*создать  новый   элемент  и   заполнить поле данных*/
   cout<<"key-?";
   cin>>p->key;
   p->pred=0;        //обнулить адрес предыдущего
   p->next=0;         //обнулить адрес следующего
   if(k==1)                //если добавляется первый элемент
   {
   p->next=beg;      //добавить перед beg
   beg->pred=p;      //связать список с добавленным элементом
   beg=p;            //запомнить первый элемент в beg
   return beg;             //вернуть начало списка
   }
   List*r=beg;             //встать на начало списка
   for(int  i=0;i<k-2 && r->next!=0; i++,r=r->next);    /*пройти    по    списку    либо    до
конца   списка, либо до элемента с номером k-1*/
   p->next=r->next;        //связать p с элементом, следующим за вставляемым
   if (r->next!=0) r->next->pred=p;      /*если  элемент  не  последний,  то  связать  элемент,
следующий за вставляемым, с p */
   p->pred=r;        //связать p и r
   r->next=p;
   return beg;             //вернуть начало списка
   }
   void main()
   {
   List*beg;
   int i,k,n;
   do
   {
   cout<<"1.Make list\n";
   cout<<"2.Print list\n";
   cout<<"3.Add List\n";
   cout<<"4.Del List\n";
   cout<<"5.Exit\n";
   cin>>i;
   switch(i)
   {
   case 1:
   {beg=Init();break;}
   case 2:
   {n=print_list(beg);
   cout<<"n="<<n<<"\n";break;}
   case 3:
   {
   cout<<"\nk-?";cin>>k;
   beg=add_List(beg,k);
   break;
   }
   case 4:
   {
   cout<<"\nk-?";cin>>k;
   beg=del1_List(beg);
   break;
   }
   }
   }
   while(i!=5);
   }
```

Методические указания

При подготовке к занятию необходимо изучить: описания двунаправленных списков; различные способы формирования и просмотра списков; особенности вставки и удаления элементов списка.

Аудиторные и домашние задания

1. Ввести двунаправленный список из *n* вещественных чисел. Заменить все нулевые элементы списка суммой первого и последнего элементов.
2. Создать двунаправленный список из *n* вещественных чисел. Определить число вхождений в список его наибольшего элемента.
3. Построить двунаправленный список символов до появления первой точки. Поменять местами первый и последний символ списка.
4. Ввести N и список действительных чисел $A_1,A_2...A_n$. Получить max(A_1+A_n ,A_2+A_{n-1} ..., $A_n + A_1$).
5. Дано натуральное число *n*. Создать список целых чисел $b_1,\ldots,b_n$. Выяснить, верно ли, что для *i=1*,…,*n* выполнено $b_i = b_{n-i+1}$.
6. Дано натуральное число *n*. Создать список из *n* целых чисел. Выяснить, верно ли, что для *i=1*,…,*n* выполнено $a_i > a_{n-i+1}$.
7. Дано натуральное число *n*. Построить список вещественных чисел $x_1,\ldots,x_{2n}$. Вычислить $(x_1 + 2x_{2n})(x_2 + 2x_{2n-1})\ldots(x_n + 2x_n).$
8. Дано натуральное число *n*. Создать список вещественных чисел $a_1,\ldots,a_n$. Вычислить $\left(\sqrt{|a_1|} - a_n\right)^2 + \left(\sqrt{|a_2|} - a_{n-1}\right)^2 + \ldots + \left(\sqrt{|a_{n-1}|} - a_1\right)^2.$
9. Дано натуральное число *n*. Построить список вещественных чисел $x_1,\ldots,x_n$. Вычислить $\left(\frac{1}{|x_1|+1} + x_n\right)\left(\frac{1}{|x_2|+1} + x_{n-1}\right)\ldots\left(\frac{1}{|x_n|+1} + x_{n-i+1}\right).$
10. Создать двунаправленный список из *n* вещественных чисел. Удалить первый и последний элементы списка.
11. Построить двунаправленный список из *n* целых чисел. Удалить из списка элемент с максимальным значением.
12. Построить двунаправленный список из *n* целых чисел. Добавить в начало списка элемент, содержащий сумму всех элементов списка, а в конец – элемент, содержащий произведение всех элементов списка.
13. Построить двунаправленный список символов. Вставить перед каждым символом цифры звездочку.
14. Создать двунаправленный список из *n* вещественных чисел. Переписать в однонаправленный список все отрицательные элементы исходного списка.

Контрольные вопросы

1. Особенности организации двунаправленного списка?
2. Сколько указателей требуется для работы с линейным двунаправленным списком?
3. Как описывается двунаправленный список?
4. Какие действия необходимо выполнить для создания линейного двунаправленного списка?
5. Как распечатать значения линейного двунаправленного списка?
6. Какие особенности вставки и удаления элементов двунаправленного списка?