

## Тема 8: ОДНОНАПРАВЛЕННЫЕ СПИСКИ

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

### Теоретические сведения

Рассмотрим структуру данных – *однонаправленный список*.

**Список** – это конечная совокупность данных одного типа, связанных между собой с помощью указателей. Наиболее простой динамической структурой является линейный однонаправленный список, элементами которого служат объекты структурного типа (рис. 8.1).



Рис. 8.1. Линейный однонаправленный список

Элемент однонаправленного списка состоит из двух частей: самого данного (возможно сложного) и указателя на следующий элемент списка. Во главе списка находится указатель *beg* («корень»), который указывает на первый элемент списка. Указателю в последнем элементе списка обычно присваивается значение *NULL*, служащее признаком конца списка. Каждый элемент списка, кроме последнего, содержит указатель на следующий за ним элемент. На каждый элемент списка, кроме первого, имеется указатель от одного элемента – предшествующего. Таким образом, с помощью указателей можно пройти по всем элементам списка в одном направлении.

Такая структура является динамической, она может изменяться в процессе выполнения программы.

Описание простейшего элемента однонаправленного списка выглядит следующим образом:

```
struct имя_типа
{
    информационные поля;
    адресное поле;
};
```

Элемент списка содержит по меньшей степени одно информационное поле. Информационное поле – это поле любого, ранее объявленного или стандартного, типа. Информационных полей может быть несколько.

Адресные поля служат для связи элементов списка между собой. Адресное поле – это указатель на объект того же типа, что и определяемая структура, в него записывается адрес следующего элемента списка или признак конца списка.

Пример описания однонаправленного списка, элементы которого содержат целые числа, в котором поле *a* – информационное, а поле *next* – адресное.

```
struct List
{
    int a;           //информационное поле
    List* next;      //адресное поле
};
```

**Создание списка.** Для построения списка необходимо использовать не менее двух указателей, например: *beg* и *p*. Переменная *beg* содержит указатель на начало списка, значением *beg* в процессе создания списка всегда будет указатель на первый элемент уже построенной части списка. Переменная *p* будет содержать указатель на последний из построенных элементов списка для присоединения к нему очередного (следующего) элемента списка.

Первый элемент списка можно сформировать с помощью операторов:

```
beg= new List; /*сформировать первый элемент*/
cout << "ВВОДИТЕ ЭЛЕМЕНТ СПИСКА"<<endl;
cin>>beg->a;   /*занести значение в первый элемент списка*/
beg->next = NULL;
```

По первому оператору будет выделено место в памяти для размещения формируемого элемента списка, а следующие операторы формируют значения его полей *a* и *next*.

Для построения второго элемента списка надо снова обратиться к процедуре *new*. Указателю будет присвоена ссылка на построенный элемент. Но теперь в качестве такого указателя нельзя использовать переменную *beg*:

- потому что, эта переменная должна ссылаться на начало (первый элемент) списка;
- в этом случае будет уничтожен первый элемент, а ведь к нему надо присоединить второй элемент – так, чтобы в поле *next* первого элемента была занесена ссылка на второй элемент.

Поэтому для формирования последующих элементов списка будем использовать переменную *p*, которая должна всегда указывать на последний из сформированных элементов списка. Поэтому после операторов, формирующих первый элемент, запишем оператор:

```
List*p=beg;
```

Теперь указатель *p* также указывает на первый элемент списка. В дальнейшем он будет указывать на последний построенный элемент. В этом случае формирование каждого очередного элемента списка можно будет производить одной и той же последовательностью операторов:

```
p->next= new List;      /* сформировать следующий элемент, связав его с предыдущим*/
p=p->next;              /* p присвоить адрес сформированного элемента списка*/
cin>> p->a ;            /* занести значение в текущий элемент списка*/
p->next = NULL;
```

Оператор *p=p->next;* используется для перехода от одного элемента к следующему по порядку элементу (последовательный перебор всех элементов списка).

**Пример 8.1.** Функция формирования списка из *n* элементов.

```
List * Init()
{
    List *beg = new List;
    int n;
    cout << "ВВЕДИТЕ КОЛИЧЕСТВО ЭЛЕМЕНТОВ СПИСКА"<<endl;
    cin>>n;
    cout << "ВВОДИТЕ " <<n <<" ЭЛЕМЕНТОВ СПИСКА"<<endl;
    cin>>beg->a ;
    beg->next = NULL;
    List *p = beg;
    for(int i=1;i<n;i++)
    {
        p->next = new List;
        p=p->next;
        cin>>p->a ;
        p->next=NULL;
    }
    return beg;
}
```

**Задача 8.1.** Создать список из *n* вещественных чисел, вывести его на печать. Все отрицательные элементы заменить значением последнего элемента.

```
#include <iostream>
using namespace std;
struct List
{
    float a;
    List* next;
};

// функция формирование списка List * Init()

//печать элементов списка и возвращение адреса последнего элемента списка
List * Print(List *b)
{
    List *end;
    cout << "\nСПИСОК"<<endl;
    while(b)
```

```

    {
        cout << b->a << " \t ";           //печать текущего элемента списка
        end=b;
        b = b->next;                       //переход на следующий элемент списка
    }
    return end;                           //end хранит адрес последнего элемента списка
}

//замена отрицательных элементов списка значением последнего элемента
List * Zamena(List *b, List*e)
{
    List *p = b;
    while(p)
    {
        if(p->a<0) p->a=e->a;
        p = p->next;                     //переход на следующий элемент списка
    }
    return b;
}

int main ()
{
    setlocale(LC_ALL , "russian");
    List *beg=NULL,*end;
    beg=Init();
    end=Print(beg);
    beg=Zamena(beg,end);
    Print(beg);
    return 0;
}

```

**Задание 8.1.** Создать список из  $n$  целых чисел, вывести его на печать. Поменять местами первый и максимальный элементы списка.

Для **добавления элемента в начало списка** необходимо создать новый элемент ( $t$ ), связать его с первым элементом и установить указатель  $beg$  на первый элемент списка.

**Пример 8.2.** Функция добавления в начало списка.

```

List * Add_begin(List *beg, const int a) //добавление в начало списка
{
    List *t = new List;                //создание нового элемента списка
    t->a=a;                             //занесение значения в новый элемент списка
    t->next = beg;                     //связывание нового элемента списка с первым элементом
    beg=t;                             //установка указателя beg на начало списка
    return beg;
}

```

Для **добавления элемента в конец списка** необходимо установить указатель ( $t$ ) на последний элемент списка, создать новый элемент ( $p$ ), связать последний элемент списка и новый элемента, занесение в адресное поле нового элемента NULL.

**Пример 8.3.** Функция добавления в конец списка.

```

List * Add_end(List *beg, const int a) //добавление в конец списка
{
    List *t=beg;                       //установка указателя t на начало списка

    while(t->next)                     //установка указателя t на последний элемент списка
        t = t->next;
    List *p = new List;                //создание нового элемента списка
    p->a=a;                             //занесение значения в новый элемент списка
    t->next=p;                         //связывание последнего элемента списка и нового элемента
    p->next=NULL;                     //занесение в адресное поле нового элемента NULL
}

```

```

    return beg;
}

```

Для добавления элемента в  $k$ -ю позицию (за исключением первой и последней) списка необходимо установить указатель ( $t$ ) на элемент перед вставляемым (позиция  $k-1$ ), создать новый элемент ( $p$ ), связать новый элемент и  $k+1$  элемент списка, связать  $k-1$  элемента списка и новый элемент..

**Пример 8.4.** Функция добавления в  $k$ -ю позицию списка (за исключением первой и последней).

```

List * Add_middle(List *beg, const int a)    //добавление в середину списка
{
    List *t=beg;                            //установка указателя t на начало списка
    int k,i=0;
    cout << "ВВЕДИТЕ НОМЕР ПОЗИЦИИ ДЛЯ ВСТАВКИ ЭЛЕМЕНТА СПИСКА" << endl;
    cin >> k;
    //установка указателя t на k-1 элемент списка (на элемент перед вставляемым)
    while(i<k-2)
    {
        t = t->next;    //проход по списку
        i++;
    }
    List *p = new List;    //создание нового элемента списка
    p->a=a;                //занесение значения в новый элемент списка
    p->next=t->next;        //связывание нового элемента и k+1 элемент списка
    t->next=p;              //связывание k-1 элемента списка и нового элемента
    return beg;
}

```

Ниже приведена функция, демонстрирующая удаление элемента с заданным значением.

**Пример 8.5.** Функция удаления элемента с заданным значением.

```

List * Delete(List *beg, const int a)    //удаление элемента списка
{
    //если список пуст
    if (beg==NULL)
    {
        return beg;
    }

    //если список не пуст
    List *t = beg;    //установка указателя t на начало списка
    // удаление первого элемента списка
    if(t->a==a)
    {
        beg=t->next;    //сдвиг указателя beg на второй элемент списка
        delete t;        //удаление первого элемента списка
        return beg;
    }

    // удаление внутри списка (и последнего элемента списка)
    List *t1 = t->next;    //установка указателя t1 на элемент, следующий за t (на второй)
    while(t1)
    {
        if(t1->a==a)
        {
            t->next=t1->next;    //связывание предыдущего со следующим за удаляемым
            delete t1;            //удаление элемента списка
            t1=t->next;
            return beg;
        }
        else {

```

```

        t=t1;
        t1=t1->next;
    }
    return beg;
}

```

//переход на следующий элемент  
//переход на следующий элемент

**Задача 8.2.** Построить список символов до первой точки. Создать из исходного списка новый, исключив все символы-цифры.

```

#include <iostream>
using namespace std;
struct List
{
    char a;
    List* next;
};

// построение списка символов до первой точки
List * Init()
{
    List *beg;
    char ch;
    cout << "ВВОДИТЕ ЭЛЕМЕНТЫ СПИСКА"<<endl;
    cin>> ch;
    if (ch!='.')
    {
        beg = new List;
        beg->a = ch;
        beg->next = NULL;
        List *p = beg;
        cin>> ch;
        while (ch != '.')
        {
            p->next = new List;
            p=p->next;
            p->a = ch;
            cin>> ch;
            p->next=NULL;
        }
    }
    return beg;
}

//печать списка
void Print(List *b)
{
    while(b)
    {
        cout << b->a << " \t ";
        b = b->next;
    }
}

//создание из исходного списка нового без символов-цифр
List * new_List(List *b)
{
    List *p1, *f1;
    while (b && b->a >='0'&& b->a <='9') b = b->next; //поиск элемента отличного от цифр
    if (b == NULL) return b; //новый список создать нельзя;
    else
    {
        p1= new List;
    }
}

```

//печать текущего элемента списка  
//переход на следующий элемент списка

//формирование 1-го элемента нового списка

```

    p1->a=b->a;
    f1=p1;           // f1 указатель на начало нового списка
    b=b->next;
    while (b != NULL)
    {
        if ( b->a <'0' || b->a >'9') // если элемент списка не цифра
        {
            p1->next = new List;    //формирование остальных элементов нового списка
            p1=p1->next;
            p1->a=b->a;
            b = b->next;
        }
        else b = b->next;
    }
    p1->next= NULL;
}
return f1;
}
int main ()
{
    setlocale(LC_ALL, "russian");
    List *beg=NULL, *beg_n;
    beg=Init();
    cout << "\nИСХОДНЫЙ СПИСОК" << endl;
    Print(beg);
    beg_n=new_List(beg);
    if (beg_n == NULL) cout << "\nНовый список создать нельзя";
    else
    {
        cout << "\nНОВЫЙ СПИСОК" << endl;
        Print(beg_n);
    }

    return 0;
}

```

**Задание 8.2.** Построить список целых чисел до ввода отрицательного значения. Создать из исходного списка новый, исключив из списка значения меньше, чем среднее арифметическое значение для этого списка.

**Понятие о стеке и очереди.** *Стек* – это структура данных, в которой элемент, записанный последним, считывают (он является доступным к обработке) первым. Принцип «последний пришел – первый ушел» используется во многих технических устройствах и в быту: вспомните рожок от автомата, посадку пассажиров в вагон с одной дверью и т.д. Стек используют в программировании для реализации рекурсии. Рекурсия выполняется так: сначала все вызовы накапливаются (аналогия такая: пружина сжимается), а потом выполняются вложенные процедуры или функции (пружина распрямляется).

*Очередь* – это структура данных, в которой элемент, записанный первым, считывают первым. Здесь действует принцип «первый пришёл – первый ушел», хорошо известный в быту: очередь за билетами, очередь на обслуживание и т.п.

Максимально допустимые размеры стека и очереди – важные характеристики реализации языка программирования, определяющие круг задач, которые можно решить. Стеки и очереди описывают и создают в памяти с помощью типа данных – указателя. Соответствующие описания и примеры процедур их обработки можно отыскать в справочниках.

### Методические указания

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

### Аудиторные и домашние задания

1. Построить список из символов до появления первой точки. Поменять местами первый и последний символ списка.

2. Построить список из символов до появления первой точки. Заменить строчные латинские буквы прописными.
3. Построить список из  $n$  целых чисел. Вычислить произведение первого, наибольшего и наименьшего чисел списка.
4. Создать список из  $n$  вещественных чисел. Заменить отрицательные элементы значением последнего элемента списка.
5. Создать список из  $n$  вещественных чисел. Сколько в списке чисел с минимальным значением среди положительных.
6. Дано натуральное число  $n$ . Построить список целых чисел  $a_1 \dots a_{2n}$ . Выяснить, верно ли, что для  $i = 1, \dots, n$  выполнено:  $a_i + a_{n+i} > 10$ .
7. Дано натуральное число  $n$ . Построить список целых чисел  $a_1 \dots a_{2n}$ . Выяснить, верно ли, что для  $i = 1, \dots, n$  выполнено:  $a_i = -a_{n+i}$ .
8. Дано натуральное число  $n$ . Построить список вещественных чисел  $x_1 \dots x_{2n}$ . Вычислить  $(x_1 - x_{n+1})^2 + (x_2 - x_{n+2})^2 + \dots + (x_n - x_{2n})^2$ .
9. Создать список из  $n$  вещественных чисел. Поменять местами наибольший и наименьший элементы списка.
10. Создать список из символов, вводимых с клавиатуры до появления символа «точка». Определить, сколько символов цифр входит в полученный список.
11. Создать список символов. Вставить после каждой цифры символ '\*».
12. Создать список из  $n$  вещественных чисел. Если список упорядочен по убыванию, продублировать его первые и последние элементы.
13. Построить список из  $n$  целых чисел. Удалить из списка отрицательные числа.
14. Создать список из  $n$  вещественных чисел. Удалить из списка наибольший и наименьший элементы.
15. Создать список из  $n$  вещественных чисел. Если список упорядочен по возрастанию, удалить первые  $k$  элементов списка.
16. Ввести список символов, заканчивая ввод точкой. Слова в этом списке разделены пробелами. В каждом слове удалить первую букву.
17. Сформировать новый список из элементов списка целых чисел, имеющих значения больше среднего арифметического для положительных элементов исходного списка.
18. Построить список из  $n$  целых чисел. Сформировать новый список из элементов исходного, стоящих на четных позициях.

### Контрольные вопросы

1. Что такое связанные структуры, стек, очередь, список?
2. Сколько указателей требуется для работы с линейным односвязным списком?
3. Как описывается однонаправленный список?
4. Какие действия необходимо выполнить для создания линейного односвязного списка?
5. Как распечатать значения линейного односвязного списка?
6. Какие особенности вставки и удаления элементов списка?