1 ОРГАНИЗАЦИЯ ОДНОНАПРАВЛЕННЫХ СПИСКОВ

Цель работы - получить практические навыки работы с переменными ссылочного типа и динамическими переменными; освоить различные способы организации, формирования и обработки списков.

8.1 Подготовка к работе

При подготовке к лабораторной работе необходимо изучить особенности описания и использования в программах ссылочных типов данных (указателей), механизм создания и уничтожения динамических переменных, описания однонаправленных списков, различные способы формирования и просмотра списков, особенности вставки и удаления элементов списка.

8.2 Теоретические сведения

Использование списков целесообразно в тех случаях, когда трудно или невозможно предсказать число объектов, обрабатываемых программой. В подобных ситуациях используются динамические объекты, которые создаются не заранее, а в моменты, определяемые логикой программы. Созданные элементы связываются с уже существующими объектами с помощью указателей.

Для описания элемента списка используется структура. Все поля элемента списка по назначению делятся на две группы:

- информационные;
- адресные.

В информационных полях помещаются те данные, ради которых и строится список. 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.

Для организации однонаправленных списков достаточно использовать одно адресное поле, для двунаправленных списков необходимо организовать связь элемента с предыдущим и последующим элементами, поэтому для этих целей понадобятся два адресных поля.

Описание простейшего элемента однонаправленного списка выглядит следующим образом:

```
struct имя_типа
{
информационное поле;
адресное поле;
};
```

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
};
```

Если количество элементов списка известно, то список может формироваться с помощью цикла *for*, причем первый элемент создается вне этого цикла. Так как элементы списка – динамические переменные, то очередной элемент возникает при выполнении процедуры ***new***. В последнем элементе списка адресное поле *next* является пустой ссылкой ***NULL***.

Функция формирования списка из n элементов:

```
List* make_list(int n)
{
        List*  beg= new List;           /*сформировать   первый элемент*/
        cout << "ВВОДИТЕ  " <<n <<"  ЭЛЕМЕНТОВ СПИСКА"<<endl;
        cin>>beg->a ; /*занести значение  в первый элемент списка*/
        beg->next = NULL;
        List*p=beg;             //p присвоить адрес первого элемента списка
        for (int i=1;i<n;i++)
        {
                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;
        }
        return beg;                     //вернуть адрес начала списка
}
```

Для последующего просмотра списка текущий указатель нужно установить на начало списка *p=beg*, а перемещаться по списку (переходить на следующий элемент) можно, выполняя в цикле оператор *p=p->next*;. Например, для нахождения суммы всех элементов списка, служит следующая функция:

```
int Summa (List *beg)
{
        List *p=beg;        // p установить на начало списка
        int s=0;
        while  (p)          //пока не достигнут конец   списка:
        {
                s+= p->a;           //нахождение суммы элементов списка
                p=p->next;          //переход на следующий элемент списка
        }
        return s;
};
```

Для вывода элементов списка можно использовать следующую функцию:

```
void Print (List *beg)
{
        cout<<"\nList\n";
```

```
        List *p = beg;                  // p установить на начало списка
        while(p)
        {
          cout << p->a << "\t";         //печать текущего элемента списка
          p = p->next;                  //переход на следующий элемент списка
        }
}
```

Для освобождения памяти можно использовать следующую функцию:

```
List* Free(List *beg)
{
        if (beg == NULL)  return beg;           //если список пуст
        List *p = beg, *t;   //t указатель на элемент, который будет удаляться
        while(p)
                {
                        t=p;
                        p=p->next;              //переход на следующий элемент
                        delete t;               //удаление  элемента  списка
                }

        beg=NULL;
        return beg;                     //вернуть адрес начала списка
}
```

8.3 Варианты заданий

Ввести элементы списка и выполнить указанные действия.

1. Для N элементов списка целых чисел вычислить $(X_1+X_n),(X_2+X_{n-1})...(X_n+X_1)$.
2. Из списка действительных чисел переписать в новый список те элементы исходного, которые принадлежат диапазону [-2; 3,5].
3. Прибавить ко всем элементам списка максимальный элемент этого списка.
4. Из списка символов $S_1,S_2...S_n$ получить список S_{m+1} ,S_{m+2} ...S_n ,S_1,...S_m (m<n) .
5. Если в списке действительных чисел есть хотя бы один отрицательный элемент, то все положительные числа умножить на 0.1.
6. В последовательности символов S_1,S_2...,первый символ S_1 отличен от точки, а среди символов S_2,S_3... имеется хотя бы одна точка. Для символов S_1...Sn, предшествующих первой точке, получить последовательность S_n,S_{n-1} ...S_1.
7. В списке целых чисел $A_1,A_2,...A_n$ поменять местами элемент, стоящий на первом месте, с элементом, значение которого минимально.
8. Для натурального N и действительных $A_1...A_{2n}$ получить $A_1 A_{2n} + A_2 A_{2n-1} +...A_n A_n$;
9. Выяснить, имеются ли среди целых чисел списка совпадающие.
10. Получить из списка $Z_1,Z_2...Zn$ последовательность $Z_1 Z_2...Z_n$, $Z_n...Z_1$.
11. Сформировать список символов, заканчивая ввод при появлении символа точка. Все символы с кодами, большими, чем заданный, заменить пробелами.

13. Для списка из N действительных чисел найти min|Ai-Acp|, где Acp – среднее арифметическое всех элементов списка.
14. В последовательности символов $S_1,S_2...S_n$ m - номер первого пробела. Получить список S_{m+1} ,S_{m+2} ...S_n,S_m,...S_1.
15. В последовательности целых чисел поменять местами элементы с минимальным и максимальным значениями.
16. Для натурального N и списка целых чисел $A_1...A_{2n}$ найти те пары чисел, для которых выполняется $A_i+A_{2n-i+1} >20$.
17. Дан список символов $S_1,S_2...Sn$, предшествующих первой точке (n заранее неизвестно). Получить последовательность $S_1,S_3...S_n$, если n нечетное и последовательность $S_2,S_4..S_n$, если n четное.
18. Для списка X_1 $X_2..Xn$ вычислить $X_1X_n + X_2 X_{n-1} +...+X_nX_1$.
19. Преобразовать список $A_1...A_n$, расположив вначале отрицательные элементы, а затем положительные, сохранив прежний порядок их следования.
20. Ввести список символов, заканчивая ввод точкой. Слова в этом списке разделены пробелами. Если количество слов нечетно, удалить первое слово.
21. Из последовательности целых чисел $A_1...A_n$ получить последовательность $X_1,Y_1,X_2,Y_2,X_3,Y_3...$, где $X_1,X_2..$ -взятые в порядке следования четные члены последовательности $A_1..A_n$, а $Y_1,Y_2..$- нечетные члены.
22. Вывести на печать элементы списка целых чисел, имеющие значения больше среднего арифметического для положительных элементов.
23. Для списка вещественных чисел $X_1X_2..X_n$ вычислить $(X_1+X_2+2X_n)(X_2+X_3++2X_{n-1})...(X_{n-1} +X_n+2X_2)$.
24. Ко всем отрицательным элементам списка A_1 $A_2..A_n$ прибавить максимальный элемент.
25. Если список целых чисел упорядочен по возрастанию, то получить новый список из положительных элементов исходного.
26. Переставить первый элемент списка в конец списка.
27. Поменять местами максимальный и последний элементы списка.
28. Удалить из списка символов все цифры.
29. В однонаправленном списке изменить порядок следования элементов на обратный.
30. Получить новый список, в котором 1 элемент равен сумме первого и последнего элементов исходного, 2-ой элемент – сумме второго и предпоследнего и т.д.

8.4 Контрольные вопросы

1. В чем различие статических и динамических переменных?
2. Как создаются и удаляются динамические переменные?
3. Как описать элемент списка?
5. Как удалить элемент из списка?
6. Как вставить элемент в список?
7. Что такое пустая ссылка и как она применяется в списках?