1 ОРГАНИЗАЦИЯ ДВУНАПРАВЛЕННЫХ СПИСКОВ

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

9.1 Подготовка к работе

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

9.2 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. Для двунаправленных списков необходимо организовать связь элемента с предыдущим и последующим элементами, поэтому для этих целей понадобятся два адресных поля.

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:

- перемещение по списку возможно в обоих направлениях. Это упрощает работу со списком, в частности, вставку и удаление.
- если какая-то из ссылок станет неверной при некоторых сбоях, целостность списка можно восстановить по другой ссылке.

Пример 9.1 даёт представление о работе с двунаправленным списком.

```
#include <iostream>
```

```
#include<string.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 << "INPUT LIST"<<endl;
        ch = cin.get();                         //ввод символа в переменную ch
        if (ch!='.')
        {
                p=new (List);                   //создать первый элемент
                beg=p;                          /*запомнить в переменную beg,
адрес начала списка*/
                p->key=ch ;                     //заполнить   поле данных
                p->pred=0;  p->next=0;          //заполнить адресные поля
                ch = cin.get();                 //ввод символа в переменную ch
                while (ch != '.')               // пока не будет введена точка

                {
                        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;
                        ch = cin.get();
                }
        }
return beg;
}

/*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;
}

//функция вывода элементов списка в обратном порядке
void print_word(List *beg, List *end)
{
	end=end->pred;
	while(end!=beg->pred)			//пока не конец вывода
		{
			cout<<end->key;
			end=end->pred;	//перейти на предыдующий
		}
}

//добавление в конец списка заданного символа
List* ins_end_List(List*beg, char c)
{
	List *p=beg;
	while(p->next)	p=p->next; /*поставить указатель p на послений
элемент списка*/
	List *r=new(List);		//созать новый элемент
	r->key=c;			// заполнить   поле данных пробелом
	p->next=r;			//связать последний элемент с новым
	r->pred=p;			//связать новый  элемент с последним
	r->next=0;
	return beg;
}

//функция возвращает true, если список отсортирован по возрастанию
bool sort_list(List *beg, List *end) {
	bool n=true;
	List*p=beg,*t=beg->next;	/*указатели p и t хранят адреса двух
соседних элементов списка*/
	while(t!=end)			//пока не конец слова
		{
			if (p->key>=t->key) {n=false;break;}
			p=p->next;		//перейти на следующий
			t=t->next;		 //перейти на следующий
		}
	return n;
}

/* функция определяет  слова, символы которых упорядочены по
возрастанию, и выводит такие слова наоборот*/
List* word_List(List*beg)
{
	List *p,*begin, *end;
//добавление в конец списка пробела
	p=ins_end_List(beg, ' ');
//выделение слов
	while(p->next)  //пока не конец списка
	{
```

```
                while (p->key == ' '&& p->next)         p=p->next; /*пропустить
пробелы*/
                begin = p;              /*указатель begin указывает на первый
символ слова*/
                if (p->next)            // если не конец списка
                {
                while (p->key != ' ' && p->next) p=p->next;  /* пропустить все
символы слова*/
                end = p;                /*указатель end указывает на символ,
следующий за последним символом слова*/
                        if (sort_list(begin,end))     /* если символы слова
упорядочены по возрастанию */
                        //вывод слова в обратном порядке
                                {
                                        print_word(begin,end);
                                        cout<<"\n";
                                }
                }
        }
        return beg;
}

void main()
{
        List*beg;
        int i,k,n;
        do
        {
                cout<<"1.Make list\n";
                cout<<"2.Print list\n";
                cout<<"3.Word  List\n";
                cout<<"4 Exit\n";
                cin>>i;
                switch(i)
                {
                        case 1:   {beg=Init(); break;}
                        case 2:   {n=print_list(beg);
                                   cout<<"n="<<n<<"\n";
                                   break;
                                   }
                        case 3:    {beg= word_List(beg);break;}
                }
        }
        while(i!=4);
}
```

9.3 Варианты заданий

Сформировать двунаправленный список символов, заканчивая ввод точкой. Слова в этом списке разделены пробелами. Выполнить указанные действия.

1. В каждом слове удалить первую букву.

2. Подсчитать количество слов, которые начинаются и заканчиваются одним и тем же символом.
3. Подсчитать количество слов, не содержащих заданный символ.
4. Подсчитать количество слов, имеющих длину меньше средней длины всех слов списка.
5. Найти слово максимальной длины.
6. Проверить последнее слово строки на симметричность.
7. Исключить из списка слова, сумма кодов символов которых кратна заданному числу.
8. Сформировать однонаправленный список из слов, содержащих цифры.
9. Сформировать однонаправленный список из слов с четными номерами.
10. Удалить из списка все односимвольные слова и пробелы, следующие за ними.
11. Сформировать однонаправленный список из слов, все символы которых упорядочены по возрастанию.
12. Сформировать однонаправленный список из слов, длина которых превышает 7.
13. Сформировать однонаправленный список из слов, которые содержат только буквы латинского алфавита.
14. Удалить из списка те слова, которые начинаются и заканчиваются цифрами.
15. Если в слове есть два одинаковых рядом стоящих символа, то удалить один из них.
16. Отметить самое короткое слово списка символом * в начале и в конце слова.
17. Если первое слово списка симметрично, то вывести все слова списка в обратном порядке.
18. Во всех словах нечетной длины удалить среднюю букву.
19. Продублировать все односимвольные слова списка.
20. Удалить слова, состоящие только из цифр.
21. Подсчитать количество симметричных слов списка.
22. Подсчитать количество служебных слов, которые используются для описания типов данных.
23. Сформировать однонаправленный список из слов, четной длины.
24. Сформировать однонаправленный список из симметричных слов.

9.4 Контрольные вопросы

1. В чем различие статических и динамических переменных?
2. Как описать элемент двунаправленного списка?
3. Как удалить элемент из двунаправленного списка?
4. Как вставить элемент в двунаправленный список?
5. Что такое пустая ссылка и как она применяется в двунаправленных списках?
6. Преимущества двунаправленного списка?