Министерство образования и науки Российской Федерации

*Санкт-Петербургский государственный электротехнический университет*

**Центр Новых Информационных Технологий**

**Кафедра МОЭВМ.**

**Отчёт по лабораторной работе:**

## **«Списки»**

Преподаватель: Самойленко В.П.

Выполнил: студент гр.5383

Допира В. Е.

Санкт-Петербург

2016

**Лабораторная работа 6**

**Списки**

**Вариант 4**

**1.Задача**

Л1-список в связанной памяти

12) Найти номера всех элементов с заданным значением.

**2.Спецификация программы**

**2.1 Входные данные**

Место и форма представления данных

Список, представленный в программе с помощью класса, вводится в файл «putin.txt».

**2.2Выходные данные**

Программа выводит на экран меню. Пользователь может совершать следующие действия:

1. Создание списка

2. Поиск

3. Удаление списка

4. Вывод списка на экран

5. Выход

**3.Алгоритм**

Программа выполняет следующую последовательность действий:

|  |
| --- |
| Объявление структуры и класса со списком  typedef int TData;  struct Node  {  TData data; // проблемные данные  Node \*next;  };  class L1List  { private:  Node \*head; // начало списка  Node \*curr, // текущий элемент  \*predCurr; // элемент, предшествующий текущему  public:  // конструктор  L1List(){head=curr=predCurr=NULL;};  // деструктор  ~L1List() {c1earList();};  // идентификация состояния списка  int null() {return head == NULL;};  int endOfList(){return curr==NULL;};    void insertAfter (TData item);  Node\* addNode( TData item, Node \*ptrNext );    void Show();  int find (TData item);    void deleteEl();  void c1earList();  }; |
| Show - функция отображения списка на экране ()  {  Определяем указатель, который изначально равен адресу начала списка  До тех пор пока не встретит пустое значение()  {  Выведет элемент x из списка  Указываем, что далее нам нужен следующий элемент  }  } |
| insertAfter – функция создания списка ()  {  Выделение памяти для структуры Node  Вставить в пустой список или в начало списка элементы поочередно, обращение к функции addNode  Иначе вставить внутрь списка или в его конец элементы поочередно, обращение к функции addNode  (вставленный элемент становится текущим, переход к следующему элементу)  } |
| addNode – функция вставки элементов ()  {  Выделение памяти для новой структуры Node  Переход к следующему элементу  Возвращение этого элемента списка  } |
| find – функция вывода номера в списке ()  {  Если список пуст, вернуть 0  Пока не конец списка()  {  Если текущий элемент равен искомому, вывод его номера и присвоение флагу 1  Переход к следующему элементу  }  Вернуть флаг  } |
| deleteEl – функция удаления элемента ()  {  Выйти, если список пуст или конец списка  Если текущий элемент равен первому, удалить первый элемент  Сдвиг следующего на место первого и снова удаление  Иначе  Удалить внутренний или последний узел  } |
| c1earList– функция удаления списка ()  {  Если список уже пуст, выход  Удаление элементов с помощью функции deleteEl до конца списка  } |
| Корректное отображение Кириллицы  Вывод меню действий со списком:  1. Создание списка  2. Поиск  3. Удаление списка  4. Вывод списка на экран  5. Выход  Пока не выход из программы ()  {  Пользователь выбирает действие из предложенных  switch(операция)  {  case 1:  {  Считать данные из файла "putin.txt"  Если файл не открыт, сообщить об этом  Иначе  { Пока не конец файла ()  {  Обращение к элементу для создания списка olist.insertAfter();  }  Закрытие файла}  Выход  }  case 2:  Ввод элемента для поиска  Вывод номера в списке с помощью olist.find  Если элемент не найден, сообщить об этом  Выход    case 3:  Удаление с помощью olist.c1earList();  Выход    case 4:  Вывод списка на экран olist.Show();  Выход    case 5:  Завершение работы со списком  В других случаях вывод сообщения: «Выбранная операция не найдена»  }  }  } |

**Функции, используемые в программе**

void L1List::Show();

Назначение: Функция отображения списка на экране

void L1List::insertAfter(TData item);

Назначение: создания списка

Параметры: item – вставляемый элемент списка из файла

Node\* L1List::addNode(TData item, Node \*ptrNext );

Назначение: вставка элементов

Параметры: item –элемент списка из файла

\*ptrNext - ссылка на следующий элемент

Возвращаемое значение: следующий элемент списка

int L1List::find (TData item);

Назначение: Вывод номера в списке

Параметры: item – искомый элемент списка из файла

Возвращаемое значение: флаг

void L1List::deleteEl();

Назначение: удаление элемента

void L1List::c1earList();

Назначение: удаление списка

**4. Набор тестов**

|  |  |
| --- | --- |
| **Файл** | **Полученный результат** |
| не существует | Файл не может быть открыт! |
| 12 60 4 4 5 60 0 4 7 60 | Действия со списком:  1. Создание списка  2. Поиск  3. Удаление списка  4. Вывод списка на экран  5. Выход  Выберите действие из предложенных: 0  Выбранная операция не найдена  Выберите действие из предложенных: 1  Создание списка  Выберите действие из предложенных: 4  Вывод списка на экран  12 60 4 4 5 60 0 4 7 60  Выберите действие из предложенных: 2  Найти номера всех элементов с заданным значением  Введите элемент для поиска 4  Номер в списке:  3 4 8  Выберите действие из предложенных: 2  Найти номера всех элементов с заданным значением  Введите элемент для поиска 60  Номер в списке:  2 6 10  Выберите действие из предложенных: 2  Найти номера всех элементов с заданным значением  Введите элемент для поиска 8  Номер в списке:  Элемент не найден  Выберите действие из предложенных: 3  Удаление списка  Выберите действие из предложенных: 5  Выход из программы |

**5. Текст программы:**

#include <cstddef>

#include <stdexcept>

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

typedef int TData;

struct Node

{

TData data; // проблемные данные

Node \*next;

};

class L1List

{ private:

Node \*head; // начало списка

Node \*curr, // текущий элемент

\*predCurr; // элемент, предшествующий текущему

public:

// конструктор

L1List(){head=curr=predCurr=NULL;};

// деструктор

~L1List() {c1earList();};

// идентификация состояния списка

int null() {return head == NULL;};

int endOfList(){return curr==NULL;};

void insertAfter (TData item);

Node\* addNode( TData item, Node \*ptrNext );

void Show();

int find (TData item);

void deleteEl();

void c1earList();

};

void L1List::Show() //Функция отображения списка на экране

{

curr=head; //Определяем указатель, который изначально он равен адресу начала списка

while (!endOfList()) //До тех пор пока не встретит пустое значение

{

cout<<curr->data<<" "; //Выведет элемент x из списка

curr=curr->next; //Указываем, что далее нам нужен следующий элемент

}

cout<<endl;

}

void L1List::insertAfter(TData item)

{

Node \*newNode;

if (null()||endOfList())

// вставить в пустой список или в нача-ло списка

{

newNode = addNode(item, head);

predCurr= head = curr = newNode;

}

else

// вставить внутрь списка или в его конец

{

newNode = addNode(item, curr);

predCurr -> next = newNode;

// вставленный элемент становится текущим

predCurr=curr= newNode;

}

}

Node\* L1List::addNode(TData item, Node \*ptrNext )

{

Node \*newLst = new Node;

newLst-> data = item;

newLst-> next= NULL;

ptrNext= newLst;

return ptrNext;

}

int L1List::find (TData item)

{

if (null( )) return 0; // список пуст

curr = head;

int j=1;

int flag=0;

while(!endOfList())

{

if ( curr-> data == item)

{cout<<j<<endl;

flag=1;}

curr = curr -> next;

j++;

}

return flag;

}

void L1List::deleteEl()

{

// выйти, если список в состоянии Null или EOList

if (null()||endOfList()) return;

if (curr == head)

// удалить первый элемент

{ head = head -> next;

delete curr;

curr = head;

return;

}

else {

// удалить внутренний или последний узел

predCurr -> next = curr -> next;

delete curr;

curr = predCurr -> next;

}

}

void L1List::c1earList()

{

if (null()) return; // список уже пуст

curr = head;

while (!null())

// удалить текущий (первый) элемент списка

deleteEl();

}

int main()

{

setlocale(LC\_ALL, "rus"); // корректное отображение Кириллицы

L1List olist;

TData i;

int operation=0;

cout<<"Действия со списком:\n";

cout<<"1. Создание списка\n";

cout<<"2. Поиск\n";

cout<<"3. Удаление списка \n";

cout<<"4. Вывод списка на экран\n";

cout<<"5. Выход\n";

while(operation!=5)

{ cout<<endl<<"Выберите действие из предложенных: ";

cin>>operation;

switch(operation)

{

case 1:

{

cout<<" Создание списка "<<endl;

ifstream fin("putin.txt");

if (!fin.is\_open()) // если файл не открыт

cout << "Файл не может быть открыт!\n"; // сообщить об этом

else

{ while(!fin.eof())

{fin>>i;

olist.insertAfter(i);

}

fin.close(); //закрытие файла

}

break;

}

case 2:

cout<<" Найти номера всех элементов с заданным значением "<<endl;

cout<<"Введите элемент для поиска ";

cin >>i;

cout<<endl<<"Номер в списке: "<<endl;

if (olist.find(i)==0) cout<<"Элемент не найден\n";

break;

case 3:

cout<<" Удаление списка "<<endl;

olist.c1earList(); break;

case 4:

cout<<" Вывод списка на экран "<<endl;

olist.Show(); break;

case 5:

cout<<" Выход из программы "<<endl;

break;

default: cout<<"Выбранная операция не найдена"<<endl;

}

}

return 1;

}

**6. Вывод**

В ходе данной работы были получены практические навыки работы со списками, заданными с помощью классов, на языке программирования С++.