Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

Московский Авиационный Институт

(национальный исследовательский университет)

Институт №3

Системы управления, информатика и электроэнергетика

Кафедра 304

Отчет по лабораторной работе

По учебной дисциплине

«**Структуры и алгоритмы обработки данных**»

На тему

«**Линейные списки**»

Группа M30-224 Б-18

Выполнили:

Мариненко Г. В.  
Рогачев В. Н.

Проверил

[Силаев Андрей Владимирович](https://mai.ru/education/schedule/ppc.php?guid=70916f74-1d9a-11e0-9baf-1c6f65450efa)

Москва 2019

Вариант 2

Разработать однонаправленный список с целочисленным типом данных.

1. Поиск в списке наличия элемента с заданным значением с получением его номера в списке. Повторное вхождение элемента- разрешено
2. Включение нового элемента в начало списка
3. Удаление элемента из конца списка

/\*Целочисленный однонаправленный список

поиск в списке наличия элемента с заданным значение с получением его номера в списке( повторное вхождение разрешено)

включение нового элемента в начало списка

удаление элемента из конца списка\*/

#include <iostream>

#include <string>

#include <fstream>

**using** **namespace** std;

**struct** ListElem //описание элемента списка

{

ListElem\* NextElem; //ссылка на следующий элемент

**int** data; //данные типа int

**int** position; //позиция элемента в списке

ListElem() //вызывается при создании элемента структуры

{

cout << "Элемент " << **this** << " создан" << endl;

}

~ListElem() //вызывается при удалении элемента структуры

{

cout << "Элемент" << **this** << " удален" << endl;

}

};

**struct** List //список

{

ListElem\* head = **nullptr**; //начало списка

ListElem\* tail = **nullptr**; //конец списка

**void** CreateListFromFile(string FNAME) //функция создания списка из файла

{

ifstream fin(FNAME);

**int** CounterStr = 0; //счетчик кол-ва строк

string str;

**while** (!fin.eof()) //считаем кол-во строк до конца файла

{

getline(fin, str);

**if** (str.length() > 0)

{

CounterStr++;

}

}

**if** (CounterStr == 0)

{

cout << "Файл пустой!" << endl;

**return**;

}

fin.seekg(0); //устанавливаем курсор на начало файла

**for** (**int** i = 0; i < CounterStr; i++) //заполнение списка

{

ListElem\* NewElem = **new** ListElem();//выделение памяти под новый

элемент списка

fin >> NewElem->data; //считывание из файла данных элемента

**if** ((head == **nullptr**) && (tail == **nullptr**)) //если список пустой

(когда хвост и голова еще NULL 1-я иттерация

{

head = NewElem;

tail = NewElem;

}

**else**

{

tail->NextElem = NewElem; //в хвосте меняем nullptr на ссылку

нового элемента

tail = NewElem; //новый элемент становится хвостом

}

tail->NextElem = **nullptr**; //в хвосте ссылка на следующей

элемент отсутствует

}

AddIndex(); //функция переиндексации элементов

fin.close();

cout << "Дискриптор:\n"<<"Начало списка: "<<head << " Конец списка: "

<< tail << endl;

}

**void** PrintList() //функция вывода списка на экран

{

**if** ((head == **nullptr**) && (tail == **nullptr**)) // если список пустой

{

cout << "Список пустой!"<<endl;

**return**;

}

**else**

{

ListElem\* CurrentElem = head; //начинаем с начала списка

**while** (CurrentElem != **nullptr**) //пока элементы не кончились

{

cout << CurrentElem->position<<" "<< CurrentElem->data << endl; //выписываем индекс и данные элемента

CurrentElem = CurrentElem->NextElem; //переход к следующему элементу

}

}

}

**void** AddIndex() //функция индексации элементов списка

{

**int** index = 0;

**if** ((head == **nullptr**) && (tail == **nullptr**)) //если список пустой

{

cout << "Список пуст!" << endl;

}

**else**

{

ListElem\* CurrentElem = head; //начинаем с начала списка

**while** (CurrentElem != **nullptr**) //пока элементы не кончились

{

CurrentElem->position = index;

index++;

CurrentElem = CurrentElem->NextElem;//переход к следующему элементу

}

}

}

**void** AddElem(ListElem\* NewElem) //функция добавления элемента в начало списка

{

**if** ((head == **nullptr**) && (tail == **nullptr**))

{

head = NewElem;

tail = NewElem;

head->NextElem = **nullptr**; //Ссылка на следующий элемент

tail->NextElem = **nullptr**; //Ссылка на следующий элемент

}

**else**

{

NewElem->NextElem = head; //Первому элементу присваиваем значение нового

элемента, который NextElem через ссылку

head = NewElem; //Присваиваем первому элементу значение нового элемента,

который ранее давал ссылку на следующий элемент

}

AddIndex();

}

**void** DeleteElem() //функция удаления элемента с конца списка

{

ListElem\* Temp = head; // начинаем с начала списка

ListElem\* LastElem = tail;

**if** ((head == **nullptr**) && (tail == **nullptr**)) //если список пустой

{

cout << "Список пуст!" << endl;

**return**;

}

**if** (head == tail)

{

**delete** head;

head = **nullptr**;

tail = **nullptr**;

cout << "Список удален" << endl;

**return**;

}

**else**

{

**while** (Temp->NextElem != LastElem)

{

Temp = Temp->NextElem;

}

tail = Temp;

**delete** LastElem;

}

tail->NextElem = **nullptr**;

AddIndex();

}

**void** SearchbyIndex(**int** Value) //Функция поиска по индексу

{

**bool** IsFound = **false**;

**if** ((head == **nullptr**) && (tail == **nullptr**))// если список пустой

{

cout << "Список пустой!" << endl;

**return**;

}

**else**

{

ListElem\* CurrentElem = head; //Начинаем с начала списка

**while** ((CurrentElem != **nullptr**)) //Пока элементы не кончились

{

**if** (CurrentElem->data == Value) //Нашли нужный элемент

{

cout<< "Индекс искомого элемента: "<<CurrentElem->position

<< " Его значение: " << CurrentElem->data << endl;

IsFound = **true**;

}

CurrentElem = CurrentElem->NextElem; //переход к следующему элементу, текущему элементу присваивается ссылка следующего

}

**if** (IsFound == **false**) //не нашли

{

cout << "Элемент не найден " << endl;

}

}

}

};

**int** main()

{

setlocale(LC\_ALL, "Rus");

**int** choice=0;

**int** Value = 0;

List\* First = **new** List();

cout << "1.Coздание списка " << endl;

cout << "2.Вывод на экран " << endl;

cout << "3.Добавить элемент в начало" << endl;

cout << "4.Удалить элемент из конца списка " << endl;

cout << "5.Найти элемент по его значению " << endl;

cout << "6.Завершить работу " << endl;

ListElem\* AddNewElem;

**while** (choice < 7)

{

cout << "Выберите действие ";

cin >> choice;

**switch** (choice)

{

**case** 1: First->CreateListFromFile("Text.txt");

**break**;

**case** 2: First->PrintList();

**break**;

**case** 3: AddNewElem=**new** ListElem();

cout << "Введите новое число int типа ";

cin >> AddNewElem->data;

First->AddElem(AddNewElem);

**break**;

**case** 4: First->DeleteElem();

**break**;

**case** 5:

cout << "Введите искомое значение элемента ";

cin >> Value;

First->SearchbyIndex(Value);

**break**;

**case** 6:

cout << "Работа завершена! "<<endl;

**return** 0;

**break**;

**default**:

**break**;

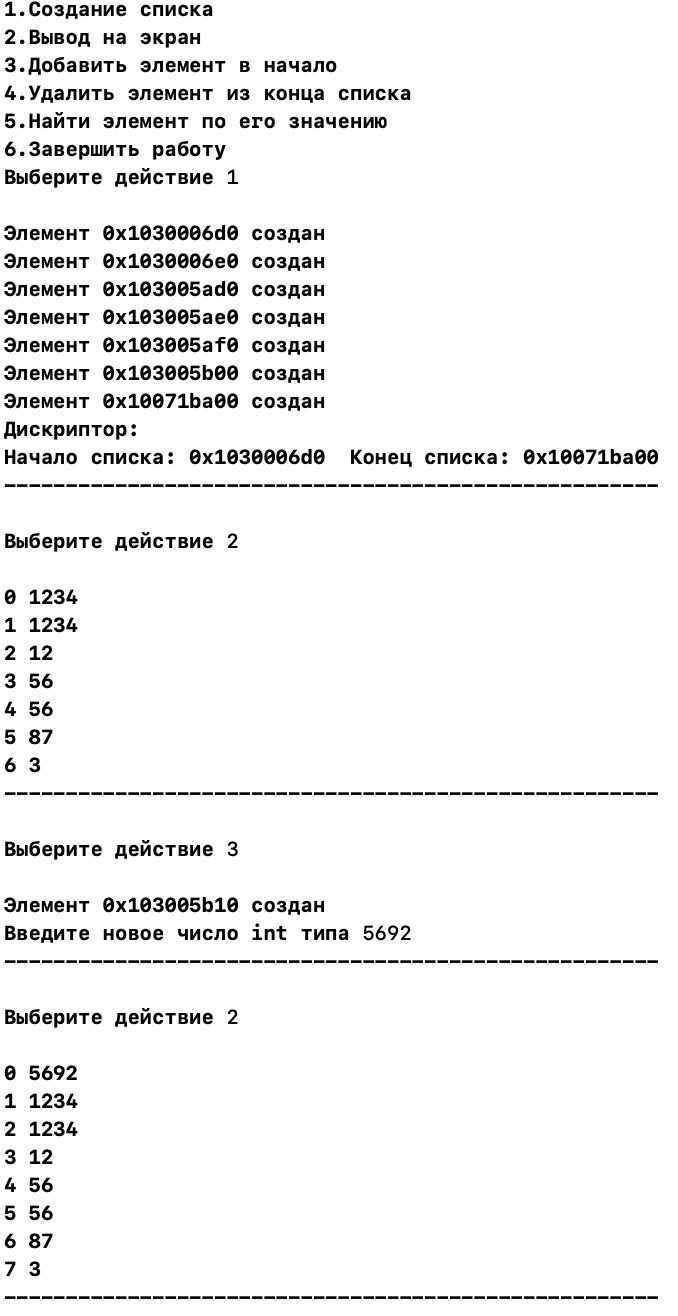
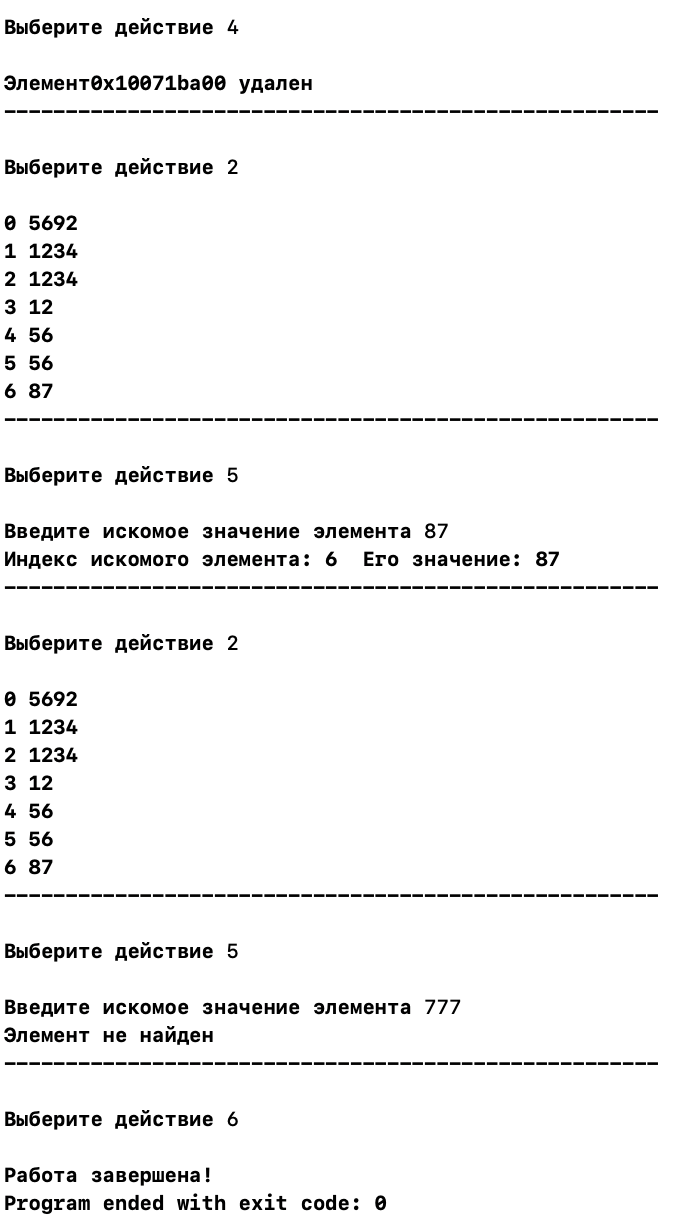
}

}

**return** 0;

}

**Пример работы программы:**

**  
**

**Вывод:**На основе полученных знаний, разработали программы, выполняющую работу с однонаправленным списком с целочисленным типом данных. Каждый элемент списка имел явные связи. Были использованы возможности списка:

1. Поиск в списке наличия элемента с заданным значением с получением его номера в списке. Повторное вхождение элемента- разрешено
2. Включение нового элемента в начало списка
3. Удаление элемента из конца списка

Дескриптор содержал в себе указатель на «голову» и «хвост» списка в памяти компьютера.