МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Программирование

Отчет по лабораторной работе № 4

Линейные списки

Выполнили студенты группы М3О-209Б-22

Кожевников А. В., Томчук Д. М.

Проверила доцент, к.т.н., Дмитриева Е.А.

Москва 2023 г.

Оглавление

[Структурные схемы алгоритмов 4](#_Toc151046153)

[Функция main 4](#_Toc151046154)

[Функция create 6](#_Toc151046155)

[Функция make 9](#_Toc151046156)

[Функция findElem 10](#_Toc151046157)

[Функция print 11](#_Toc151046158)

[Функция delList 12](#_Toc151046159)

[Функция findMax 13](#_Toc151046160)

[Код программы 15](#_Toc151046161)

[Тестирование программы 19](#_Toc151046162)

[Тестирование некорректных режимов 19](#_Toc151046163)

[Тестирование корректных режимов 20](#_Toc151046164)

[Вывод 22](#_Toc151046165)

**Задание**

Реализовать заданный согласно варианту задания линейный список, состоящий из 20 элементов заданного типа. Интерфейс должен включать **для всех вариантов** следующие операции**:**

- создание списка;

- вывод на экран и/или в файл значений элементов списка с их индексами (номерами);

- удаление списка,

а также **дополнительные операции** (вариант - 4):

* Поиск в списке элемента с максимальным значением с получением его номера в списке (повторное вхождение одного и того же значения запрещено)
* Включение нового элемента в позицию списка с заданным в программе номером.
* Удаление элемента из позиции списка с заданным в программе номером.
* Вид списка – однонаправленный, вещественный.

Отчет должен содержать:

- задание на лабораторную работу, соответствующее варианту;

- схемы алгоритмов;

- текст программы и всех функций;

- результаты работы программы.

# Структурные схемы алгоритмов

## Функция main

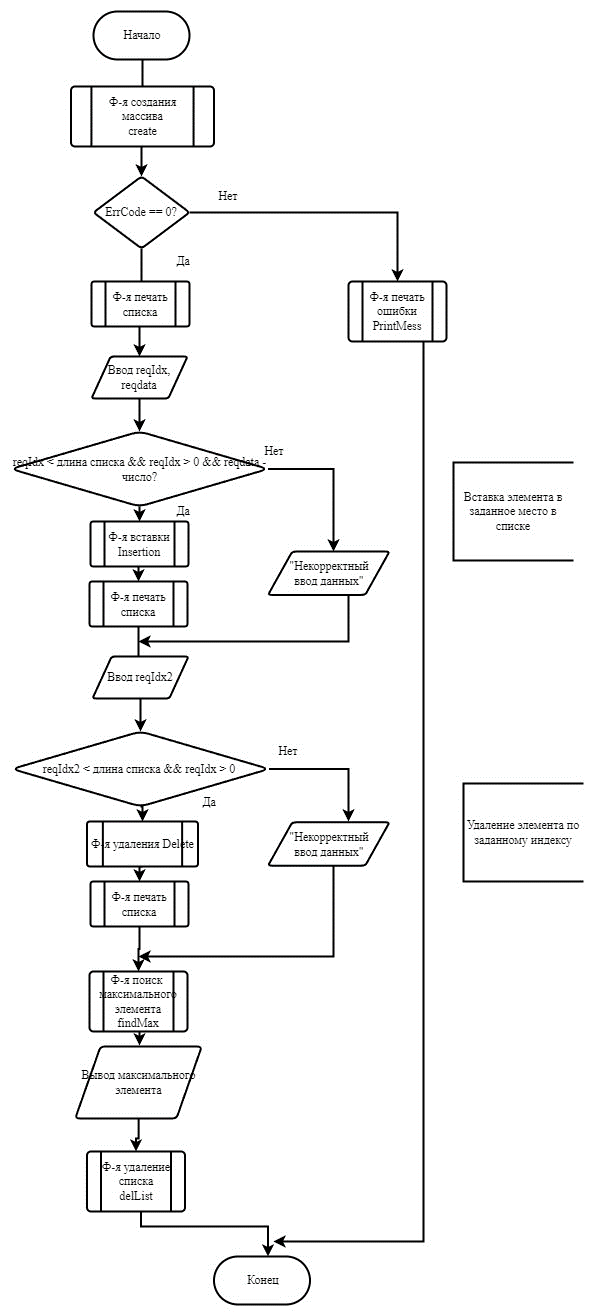


Рисунок 1. Структурная схема алгоритма функции main.

## Функция create

1. Назначение:

Создание списка из элементов, представленных в файле

2. Прототип функции:

int create(List\*& head, List\*& end, const char\* fileName)

3. Обращение к функции:

ErrCode = create(head, end, fileName)

4. Блок-схема:

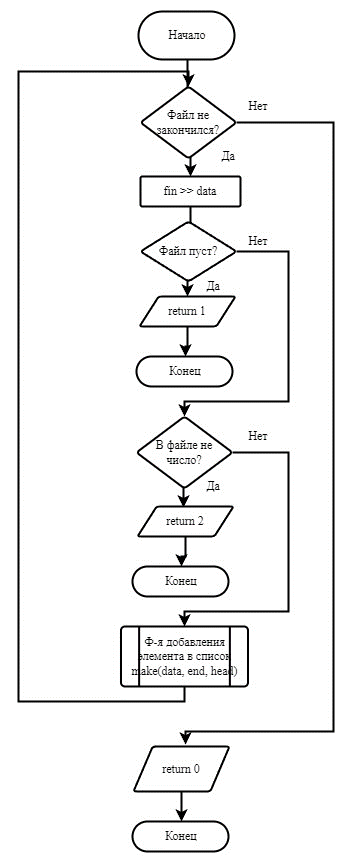


Рисунок 2. Структурная схема алгоритма функции create

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | List\*& | Указатель на первый элемент списка | Вход |
| end | List\*& | Указатель на последний элемент списка | Вход |
| fileName | Const char\* | Имя файла | Вход |

## Функция make

1. Назначение:

Добавление уникального элемента в конец списка

2. Прототип функции:

void make(float a, List\*& end, List\*& head)

3. Обращение к функции:

make(data, end, head)

4. Блок-схема:

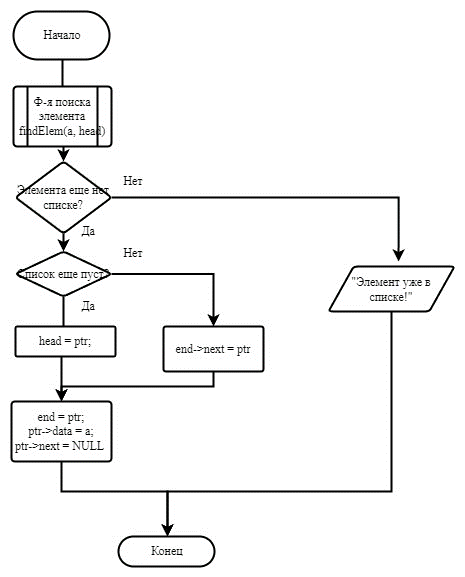


Рисунок 3. Структурная схема алгоритма функции make

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| a | Float | Новый элемент списка | Вход |
| head | List\*& | Указатель на первый элемент списка | Вход |
| end | List\*& | Указатель на последний элемент списка | Вход |

## Функция findElem

1. Назначение:

Поиск указанного элемента в списке

2. Прототип функции:

List\* findElem(float a, List\* head)

3. Обращение к функции:

findElem(a, head)

4. Блок-схема:

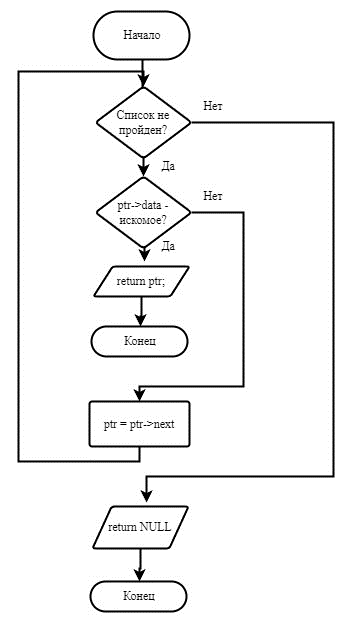


Рисунок 4. Структурная схема алгоритма функции findElem

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | List\*& | Указатель на первый элемент списка | Вход |
| a | float | Искомое | Вход |
| ptr | List\*& | Указатель на найденный элемент | Выход |

## Функция print

1. Назначение:

Печать всего списка

2. Прототип функции:

void print(List\* head)

3. Обращение к функции:

print(head)

4. Блок-схема:

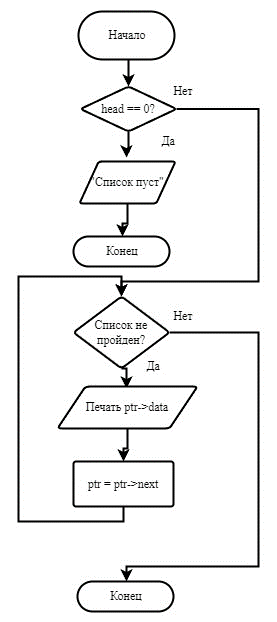


Рисунок 5. Структурная схема алгоритма функции print

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | List\* | Указатель на первый элемент списка | Вход |

## Функция delList

1. Назначение:

Удаление всего списка

2. Прототип функции:

void delList(List\*& head)

3. Обращение к функции:

delList(head)

4. Блок-схема:

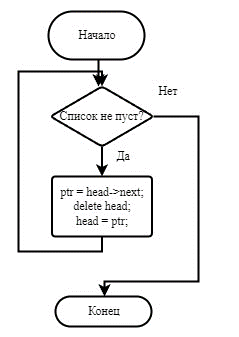


Рисунок 6. Структурная схема алгоритма функции delList

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | List\* | Указатель на первый элемент списка | Вход |

## Функция findMax

1. Назначение:

Поиск максимального элемента в списке и возврат его индекса

2. Прототип функции:

int findMax(List\* head)

3. Обращение к функции:

findMax(head)

4. Блок-схема:

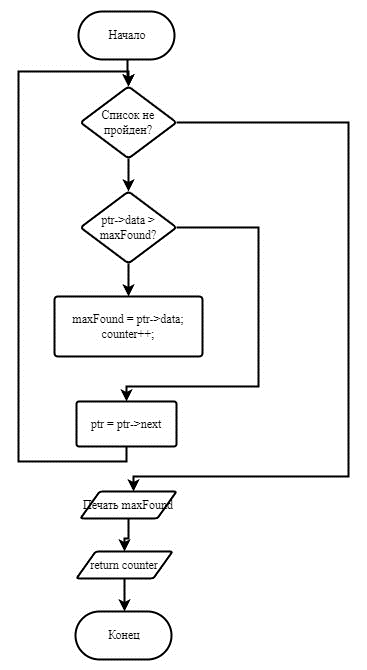


Рисунок 7. Структурная схема алгоритма функции findMax

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| counter | int | Индекс максимального элемента в списке | Выход |
| head | List\* | Указатель на первый элемент списка | Вход |

## Функция Delete

1. Назначение:

Удаление элемента списка по номеру

2. Прототип функции:

void Delete (List\*& head, int index);

3. Обращение к функции:

void Delete(head, reqIdx2);

4. Блок-схема:

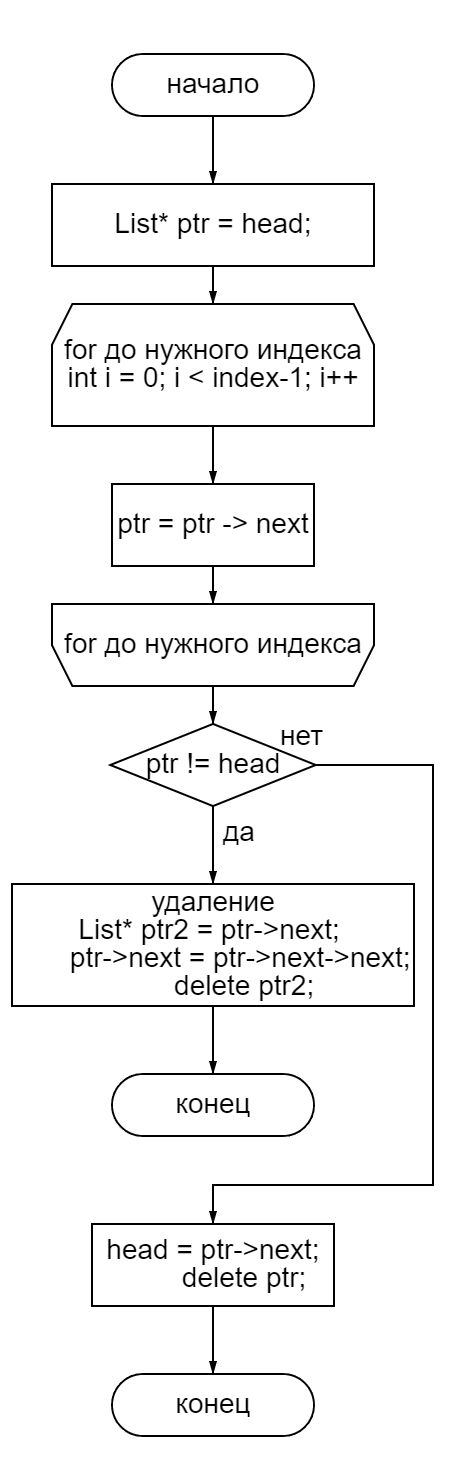


Рисунок 7. Структурная схема алгоритма функции Delete

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| index | int | Индекс элемента для удаления в списке | Вход |
| head | List\* | Указатель на первый элемент списка | Вход |

## Функция Insertion

1. Назначение:

Добавление элемента списка по номеру

2. Прототип функции:

void Insertion (float a, List\* head, int index);

3. Обращение к функции:

Insertion(reqdata, head, reqIdx);

4. Блок-схема:

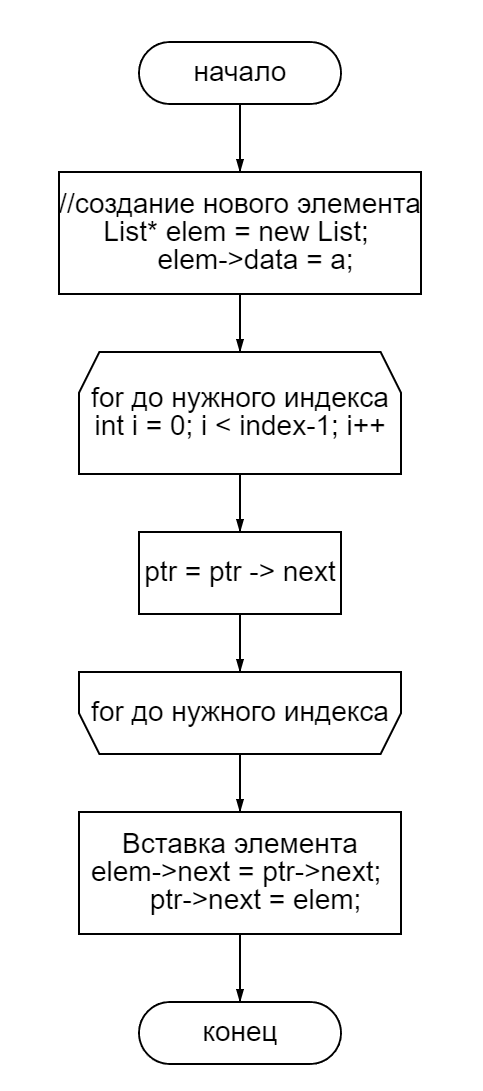


Рисунок 7. Структурная схема алгоритма функции Insertion

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| index | int | Индекс элемента для добавления в списке | Вход |
| head | List\* | Указатель на первый элемент списка | Вход |
| a | float | Значения элемента | Вход |

# Код программы

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Программирование \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Project type :Win64 Console Application \*

\*Project name :lab\_sorting.sln \*

\*File name :lab\_sorting.cpp \*

\*Language :CPP, MSVS 2022 \*

\*Programmers :Кожевников Артем Вадимович, Томчук Дмитрий Максимович, М3О-209Б-22 \*

\*Modified By : \*

\*Created :14.11.2023 \*

\*Last revision:16.11.2023 \*

\*Comment : \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <fstream>

using namespace std;

// С Т Р У К Т У Р Ы

struct List {

float data = NULL;

List\* next = NULL;

}; // Структура списка

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int create(List\*& head, List\*& end, const char\* fileName); // Создание списка

void make(float a, List\*& end, List\*& head); // Добавление нового элемента

void print(List\* head); // Печать всего списка

void delList(List\*& head); // Удаление списка

List\* findElem(float a, List\* head); // Найти элемент в списке

int findMax(List\* head); // Поиск максимального элемента с возвратом его номера

void Insertion(float a, List\* head, int index); // добавлений элемента

void Delete(List\*& head, int index); // Удаление элемента по его индексу

int count\_size(List\* head); // Посчитать длину списка

void PrintMess(int code); // Сообщение об ошибке

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Г Л О Б А Л Ь Н Ы Е П Е Р Е М Е Н Н Ы Е \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Файлы тестов

//const char\* fileName = "corr1.txt";

const char\* fileName = "corr2.txt";

//const char\* fileName = "incorr1.txt";

//const char\* fileName = "incorr2.txt";

//const char\* fileName = "incorr3.txt";

int ErrCode = 0; // Код ошибки

int size; // Длина списка

float reqdata; int reqIdx; // Новый элемент списка и его индекс в списке, запрашиваемые у пользователя

int reqIdx2; // Индекс элемента на удаление

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л А В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "Russian");

List\* head = NULL;

List\* end = NULL;

ErrCode = create(head, end, fileName);

if (!ErrCode)

{

print(head);

// Добавление нового элемента

cout << "Введите индекс места, в которое нужно поставить новый элемент: "; cin >> reqIdx;

cout << endl;

cout << "Введите элемент, который требуется вставить: "; cin >> reqdata;

cout << endl;

if (reqIdx<count\_size(head) && reqIdx >= 0 && !cin.fail()) // Индекс за пределами или не число?

{

Insertion(reqdata, head, reqIdx);

print(head);

} // if

else cout << "Некорректный ввод данных!" << endl;

// Удаление элемента

cout << "Введите индекс элемента, который нужно удалить: ";

cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Очистка буфера

cin >> reqIdx2;

cout << endl;

if (reqIdx2<count\_size(head) && reqIdx2 >= 0) // Индекс за пределами?

{

Delete(head, reqIdx2);

print(head);

} // if

else cout << "Некорректный ввод данных!" << endl;

// Поиск максимального элемента

int res = findMax(head); // Индекс максимального элемента

cout << "Индекс максимального элемента = " << res << endl;

// Удаление списка

delList(head);

print(head);

} // if

else PrintMess(ErrCode);

} // main

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int create(List\*& head, List\*& end, const char\* fileName)

{

float data;

ifstream fin(fileName);

while (!fin.eof())

{

fin >> data;

if (fin.eof() && fin.fail())

{

fin.close(); return 1; // Массив пуст

}// if

if (fin.fail())

{

fin.close(); return 2;

} // Передали не число

make(data, end, head);

} // while

fin.close();

return 0; // все ок

} // create

void make(float a, List\*& end, List\*& head) // Добавление нового элемента

{

if (!findElem(a, head)) // Если a в списке еще не было

{

List\* ptr = new List; // Новый элемент

if (!head) head = ptr; // Если это первый элемент

else { end->next = ptr; } // У конца появилось продолжение

ptr->data = a;

end = ptr; // Делаем этот элемент последним

ptr->next = NULL; // Следующих элементов списка нет

} // if

else cout << "Элемент a = " << a << " уже содержится в списке!" << endl;

} // make

List\* findElem(float a, List\* head) // Найти элемент в списке

{

List\* ptr = head;

while (ptr)

{

if (ptr->data == a) return ptr;

ptr = ptr->next;

}

return nullptr;

} // findElem

void print(List\* head) // Печать всего списка

{

List\* ptr = head;

if (!ptr) cout << "Список пуст";

while (ptr)

{

cout << "\t" << ptr->data; // Считывание данного

ptr = ptr->next; // Переход к следующему

}

cout << endl;

} // print

int count\_size(List\* head) // Посчитать размер списка

{

int counter = 0;

while (head)

{

counter++;

head = head->next;

} // while

return counter;

} // counter\_size

int findMax(List\* head) // Поиск максимального

{

List\* ptr = head;

int counter = -1; // Индекс

float maxFound = -pow(10, 38); // Очень маленькое значение

while (ptr)

{

if (ptr->data > maxFound)

{

maxFound = ptr->data;

counter++;

} // if

ptr = ptr->next; // Переход дальше

} // while

cout << "Наибольшее значение = " << maxFound << endl;

return counter;

} // findMax

void Insertion(float a, List\* head, int index) { // Вставка элемента в список на указанное место

List\* elem = new List;

elem->data = a;

List\* ptr = head;

for (int i = 0; i < index-1; i++) {

ptr = ptr -> next;

}

elem->next = ptr->next;

ptr->next = elem;

} // Insertion

void Delete(List\*& head, int index) { // Удаление элемента по индексу

List\* ptr = head;

for (int i = 0; i < index -1; i++) {

ptr = ptr->next;

} // for i

if (ptr != head)

{

List\* ptr2 = ptr->next;

ptr->next = ptr->next->next;

delete ptr2;

} // if

else {

head = ptr->next;

delete ptr;

} // else

} // Delete

void delList(List\*& head) // Удаление всего списка

{

List\* ptr;

while (head) // Удаление, начинающееся с "головы"

{

ptr = head->next;

delete head;

head = ptr;

} // while

head = NULL;

} // delList

void PrintMess(int code) // Печать сообщения об ошибке

{

switch (code)

{

case 1: cout << "Файл пуст" << endl; break;

case 2: cout << "В файле не цифра" << endl; break;

} // switch }

# Тестирование программы

## Тестирование некорректных режимов

Тест 1

Цель: Проверить работу программы при пустом файле

Исходные данные: Пустой файл

Ожидаемый результат: «Файл пустой»

Полученный результат:



Тест 2

Цель: Проверить работу программы при наличии букв в файле

Исходные данные:



Ожидаемый результат:

«В файле не цифра»

Полученный результат:



Тест 3

Цель: Проверить работу программы при отсутствии данных в файле

Исходные данные:



Ожидаемый результат:

«Файл пустой»

Полученный результат:



Тест 4

Цель: Проверить работу программы при попытке задать букву для удаления из списка

Исходные данные:

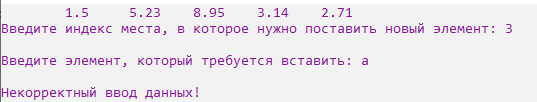
файл corr1.txt;

reqdata = a

Ожидаемый результат:

«Некорректный ввод данных»

Полученный результат:



Тест 5

Цель: Проверить работу программы при некорректно указанном индексе для удаления элемента

Исходные данные:

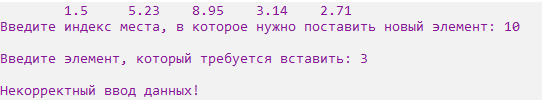
файл corr1.txt;

reqIdx = 10

Ожидаемый результат:

«Некорректный ввод данных»

Полученный результат:



Тест 6

Цель: Проверить работу программы при попытке вставить элемент за пределами списка

Исходные данные:

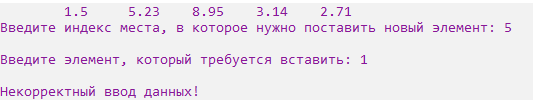
Файл corr1.txt

reqIdx = 5

Ожидаемый результат:

«Некорректный ввод данных»

Полученный результат:



## Тестирование корректных режимов

Тест 1

Цель: Проверить работу программы при корректных входных данных из вещественных чисел

Исходные данные:

Файл corr1.txt

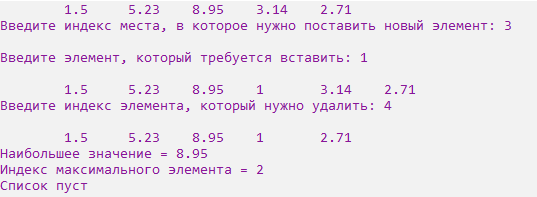
reqData = 1, reqIdx = 3, reqIdx2 = 4

Ожидаемый результат:

Список = [1.5, 5.23, 8.95, 1, 2.71]

Наибольшее значение = 8.95, индекс = 2

Полученный результат:



Тест 2

Цель: Проверить работу программы при корректных входных данных из целых чисел

Исходные данные:

Файл corr2.txt

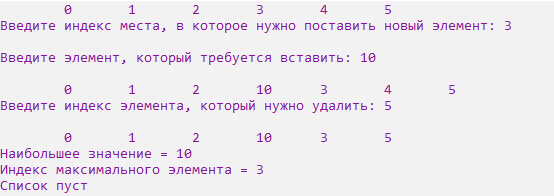
reqData = 10, reqIdx = 3, reqIdx2 = 5

Ожидаемый результат:

Список = [0, 1, 2, 10, 3, 5]

Наибольшее значение = 10, индекс = 3

Полученный результат:



Вывод по тестам: программа работает корректно.

# Вывод

Вывод по результатам сопоставительного анализа:

Анализ времени работы всех алгоритмов

* В наилучшем случае (упорядоченная по возрастанию последовательность) алгоритм IS работает быстрее QS
* В среднем и наихудшем случаях быстрее работает алгоритм QS
* Таким образом, в среднем и наихудшем случаях QS – более эффективный алгоритм. В наилучшем случае IS – более эффективен

Анализ количества сравнений в алгоритмах QS и IS

* В наилучшем случае QS имеет большее количество сравнений и перестановок чем IS
* В среднем и наилучшем случаях QS имеет меньше сравнений и перестановок, чем IS
* Таким образом, с точки зрения количества операций, а. следовательно, и быстроты работы, алгоритм QS эффективнее IS.