МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

 Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Списки. Стеки. Очереди

Отчет по лабораторной работе

по дисциплине

Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капранов С. Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

**Задача 21.**

Написать процедуру, которая меняла бы в односвязном списке крайние элементы.

**Входные данные**

Программе через консоль передаются элементы списка, количество которых задаёт пользователь.

**Результат работы программы**

Результатом является список, элементы которого равны тем, которые ввёл пользователь, крайние элементы поменяны местами.

**Код программы**

#include<iostream> //подключение библиотеки ввода-вывода

using namespace std; // использование пространства имён std

template <typename T> // создание шаблонного класса list

class List {

public: //публичная секция класса

List() { // конструктор по умолчанию

head = nullptr; //указатель на первый элемент списка

}

~List() { //деструктор

delete [] head;

}

void push\_back(T data ) { //метод добавления элемента в конец

// списка

if (head == nullptr) { //если список пустой, то выделяем

//память и добавляем элемент

head = new Node<T>(data);

}

else { //иначе доходим до последнего

//элемента, выделяем память и добавляем элемент

Node<T> \*cur= head; while (cur->Next != nullptr) {

cur = cur->Next;

}

cur->Next = new Node <T>(data);

}

}

T& operator[](const int index) {//перегрузка оператора, для

//обращения к элементам списка

int counter = 0;

Node <T>\* cur = head;

while (cur != nullptr) {

if (counter==index){

return cur->value;

}

cur = cur->Next;

counter++;

}

}

void swap() { //метод, меняющий местами крайние элементы

T buf = head->value;

Node<T>\* cur = head;

while (cur->Next != nullptr) {

cur = cur->Next;

}

head->value = cur->value;

cur->value = buf;

}

private: //приватная секция класса

template <typename T> //создание внутреннего класса для

//хранения элемента списка

class Node {

public:

T value; //значение элемента

Node\* Next; //указатель на следующий элемент

Node(T val = T(), Node\* n = nullptr) { //конструктор

//с параметрами

value = val;

Next = n;

}

};

Node<T> \*head; //член класса list, хранит указатель

//на первый элемент

};

int main() { //точка входа в программу

setlocale(LC\_ALL,"ru"); //включение русского языка

List <float> lst;

int n;

float a;

cout << "Введите количество элементов в списке"<<endl;

cin >> n;

for (int i = 0; i < n; i++){ //запись элементов в список

cout<<"Введите элемент списка под номером "<<i+1<<endl;

cin >> a;

lst.push\_back(a);

}

cout << "Текущий список" << endl; //вывод списка на консоль

for (int i = 0; i < n; i++) {

cout << lst[i]<<endl;

}

cout << "Список после перестановки крайних элементов "<<endl;

lst.swap(); //использование метода перестановки крайних элементов

//вывод списка на консоль

for (int i = 0; i < n; i++) {

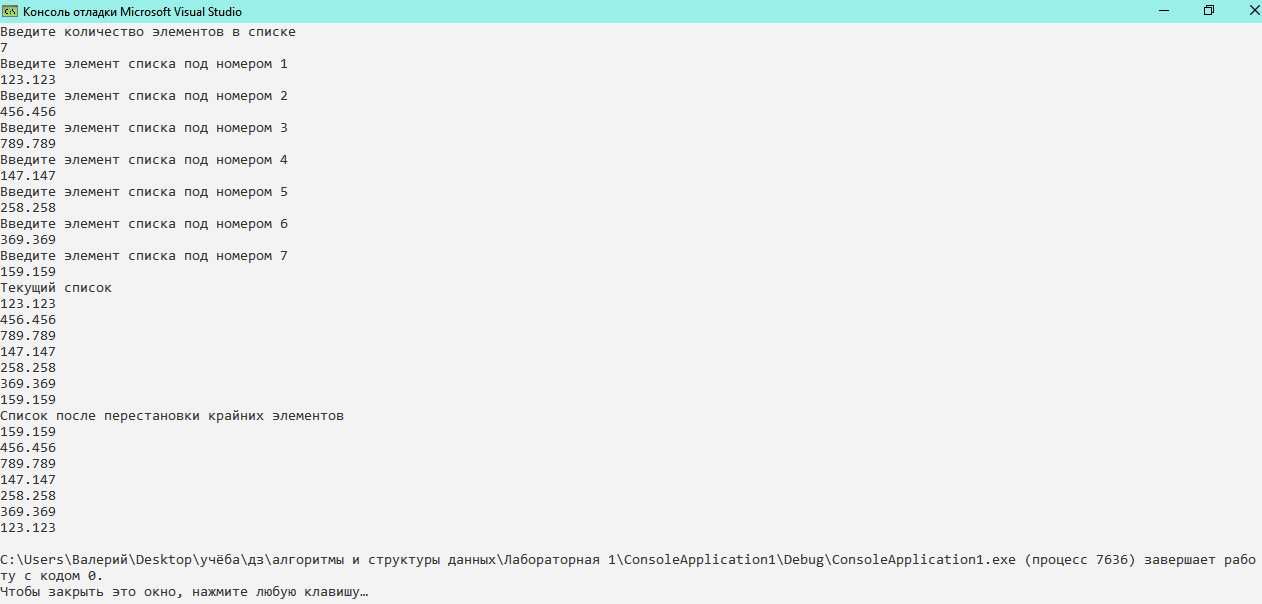
cout << lst[i]<<endl;

}

return 0;

}

**Результат работы программы**

****

**Функция для перестановки крайних элементов**

template <typename T>

void swaping(List<T>\* lst) {

T buf = lst->head->value;

Node <T>\* cur = lst->head;

while (cur->Next != nullptr) {

cur = cur->Next;

}

lst->head->value = cur->value;

cur->value = buf;

}

Использование:

List <float> lst;

/заполнение списка/

swaping(&lst);