Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Лабораторная работа 11

"Динамические структуры данных"

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Тарасов C.В.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

Написать программу, в которой реализуется работа с двусвязным списком.

Для каждого варианта разобрать следующие функции:

1. Удаление данных из списка

2. Печать списка

3. Удаление элемента в начале списка

4. Удаление элемента в конце списка

5. Добавление элемента в начало списка

6. Добавление элемента в конец списка

7. Добавление элемента после элемента с заданным номером

8. Удаление элемента по номеру

9. Добавление K элементов после элемента с заданным номером

10. Удаление k элементов после элемента с заданным номером

**Анализ задачи**

**Связный список, содержащий два поля указателя – на следующий элемент и на предыдущий, называется двусвязным.**

Для реализации поставленной задачи и функций, создадим структуры – List и Node.

struct Node {

int data;

Node\* next;

};

struct List {

Node\* head;

int count;

};

**Программный код**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

Node\* prev;

};

struct List {

Node\* head;

Node\* tail;

int count;

};

void show\_list(List& list);

void push\_front(List& list, int value);

int pop\_front(List& list);

void push\_back(List& list, int value);

int pop\_back(List& list);

int add\_after(List& list, int value, int number);

void removeelement(List& list, int number);

int add\_several(List& list, int number, int k);

void clear\_list(List& list);

int main()

{

setlocale(LC\_ALL, "Russian");

List list;

list.head = NULL;

list.tail = NULL;

list.count = 0;

int n, value, number, k, flag;

do {

system("cls");

cout << "Введите 0 для завершения работы\n";

cout << "Введите 1 для показа списка\n";

cout << "Введите 2 для добавления элемента в начало списка\n";

cout << "Введите 3 для добавления элемента в конец списка\n";

cout << "Введите 4 для удаления элемента в начале списка\n";

cout << "Введите 5 для удаления элемента в конце списка\n";

cout << "Введите 6 для добавления элемента после элемента с заданным номером\n";

cout << "Введите 7 для удаления элемента с заданным номером\n";

cout << "Введите 8 для добавление k элементов после элемента с заданным номером\n";

cout << "Введите 9 для удаления k элементов после элемента с заданным номером\n";

cout << "Введите 10 для удаления данных из списка\n";

cout << "============================================================================\n";

cin >> n;

switch (n)

{

case 1:

{

show\_list(list);

cout << "Введите что бы продолжить\n"; cin >> value;

break;

}

case 2:

{

cout << " Введите значение элемента, для вставки в начало "; cin >> value;

push\_front(list, value);

break;

}

case 3:

{

cout << " Введите значение элемента, для вставки в конец "; cin >> value;

push\_back(list, value);

break;

}

case 4:

{

cout << " Удаление в начале списка \n";

flag = pop\_front(list);

if (flag == -1) cout << " Список пуст\n";

else cout << " Элемент в начале списка со значением - " << flag << " был удален \n";

break;

}

case 5:

{

cout << " Удаление в конце списка \n";

flag = pop\_back(list);

if (flag == -1) cout << " Список пуст\n";

else cout << " Элемент в конце списка со значением - " << flag << " был удален \n";

break;

}

case 6:

{

cout << " Введите номер элемента, после которого нужно вставить элемент "; cin >> number;

cout << " Введите значение элемента для вставки "; cin >> value;

flag = add\_after(list, value, number - 1);

if (flag == -1) cout << " Что-то введено неверно :( ";

break;

}

case 7:

{

cout << " Введите номер элемента, который нужно удалить ";cin >> number;

removeelement(list, number);

break;

}

case 8:

{

cout << " Введите номер элемента, после которого нужно вставить элементы "; cin >> number;

cout << "Введите количество элементов для вставки "; cin >> k;

flag = add\_several(list, number - 1, k);

if (flag == -1) cout << " Что-то введено неверно :(\n";

break;

}

case 9:

{

cout << "Введите количество элементов для удаления "; cin >> k;

cout << " Введите номер элемента,после которого нужно удалить ";cin >> number;

for (int i = 0;i < k;i++)

{

removeelement(list, number);

}

break;

}

case 10:

{

clear\_list(list);

cout << "Данные удалены ";

break;

}

case 0:

cout << " Завершение работы ";

break;

}

} while (n != 0);

}

// Функция печати списка

void show\_list(List& list) {

Node\* temp = list.head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

// Функция добавления элемента в начало списка

void push\_front(List& list, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = list.head;

newNode->prev = nullptr;

if (list.head != nullptr) {

list.head->prev = newNode;

}

list.head = newNode;

if (list.tail == nullptr) {

list.tail = newNode;

}

++list.count;

}

// Функция удаления элемента в начале списка

int pop\_front(List& list) {

if (list.head == nullptr) {

return -1;

}

int result = list.head->data;

Node\* temp = list.head;

list.head = list.head->next;

if (list.head != nullptr) {

list.head->prev = nullptr;

}

else {

list.tail = nullptr;

}

delete temp;

return result;

}

// Функция добавления элемента в конец списка

void push\_back(List& list, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

newNode->prev = list.tail;

if (list.tail != nullptr)

{

list.tail->next = newNode;

}

list.tail = newNode;

if (list.head == nullptr) {

list.head = newNode;

}

list.count++;

}

// Функция удаления элемента из конца списка

int pop\_back(List& list) {

if (list.head == nullptr) {

return -1;

}

Node\* temp = list.tail;

int result = temp->data;

if (list.head == list.tail) { // список содержит только один элемент

list.head = nullptr;

list.tail = nullptr;

}

else {

list.tail = temp->prev;

list.tail->next = nullptr;

}

delete temp;

list.count--;

return result;

}

// Функция добавления элемента после элемента с заданным номером

int add\_after(List& list, int value, int number) {

if (list.head == nullptr || number >= list.count) {

return -1; // ничего не делаем, если список пустой, number неверный

}

Node\* current = list.head;

for (int i = 1; i <= number && current != nullptr; i++) {

current = current->next; // перебираем элементы до того, который имеет номер number

}

if (current == nullptr) {

return -1; // number выходит за пределы списка

}

Node\* newNode = new Node;

newNode->data = value;

newNode->next = current->next;

current->next = newNode;

newNode->prev = current;

if (newNode->next != nullptr) {

newNode->next->prev = newNode;

}

else {

list.tail = newNode; // если добавляемый элемент последний, то он становится хвостом списка

}

list.count++;

return 0;

}

// Функция удаления элемента по номеру

void removeelement(List& list, int number) {

if (number < 1 || number > list.count) {

return;

}

if (number == 1) {

Node\* temp = list.head;

list.head = list.head->next;

delete temp;

list.count--;

return;

}

if (number == list.count)

{

Node\* temp = list.tail;

list.tail = list.tail->prev;

list.tail->next = nullptr;

delete temp;

list.count--;

return;

}

Node\* temp = list.head;

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

temp = temp->next;

}

Node\* delNode = temp->next;

temp->next = delNode->next;

temp = delNode->next;

temp->prev = delNode->prev;

delete delNode;

list.count--;

}

// Функция добавления K элементов после элемента с заданным номером

int add\_several(List& list, int number, int k) {

if (list.head == nullptr || number >= list.count || k <= 0) {

return -1; // ничего не делаем, если список пустой, number неверный, или k не положительное число

}

Node\* current = list.head;

for (int i = 1; i <= number && current != nullptr; i++) {

current = current->next; // перебираем элементы до того, который имеет номер number

}

if (current == nullptr) {

return -1; // number выходит за пределы списка

}

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

int value;

cout << "Введите значение для вставки элемента\n"; cin >> value;

Node\* newNode = new Node;

newNode->data = value;

newNode->next = current->next;

current->next = newNode;

newNode->prev = current;

if (newNode->next != nullptr) {

newNode->next->prev = newNode;

}

else {

list.tail = newNode; // если добавляемый элемент последний, то он становится хвостом списка

}

list.count++;

current = newNode;

}

return 0;

}

// Функция очистки списка

void clear\_list(List& list) {

Node\* temp = list.head;

while (temp != nullptr) {

Node\* delNode = temp;

temp = temp->next;

delete delNode;

}

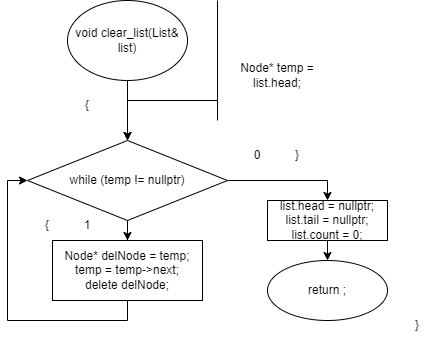
list.head = nullptr;

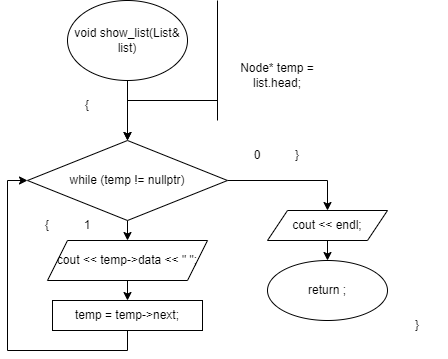
list.tail = nullptr;

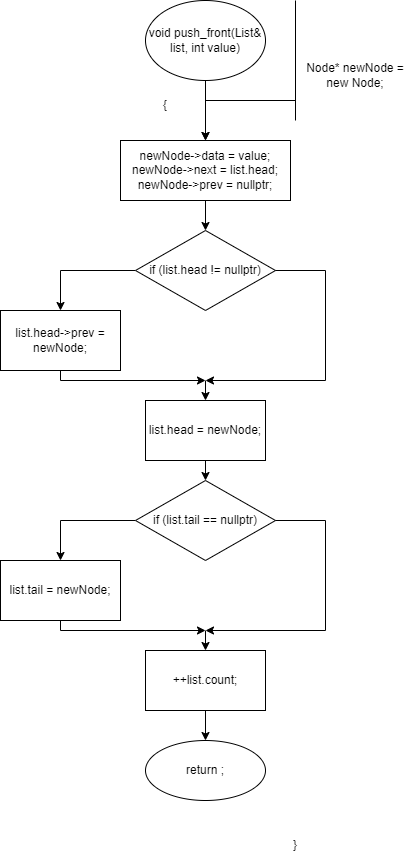
list.count = 0;

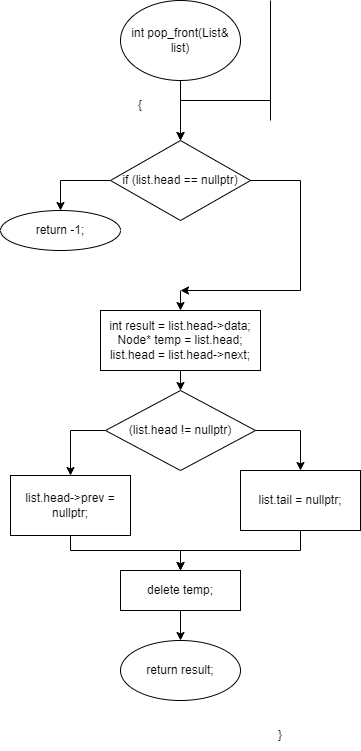
}

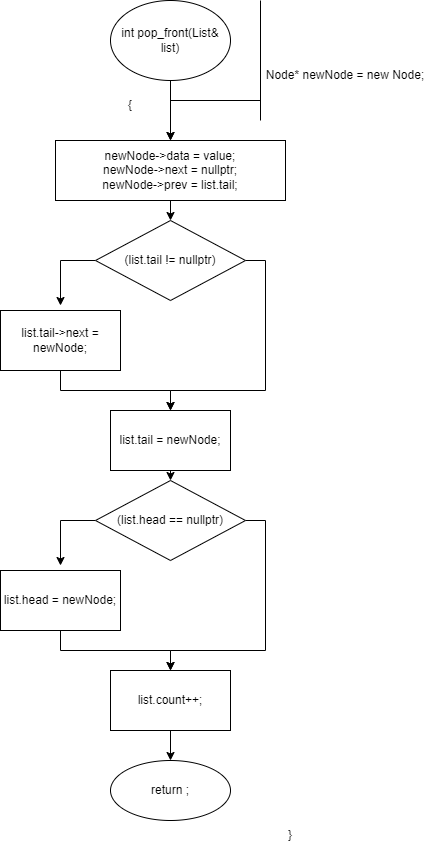
**Блок-схемы**

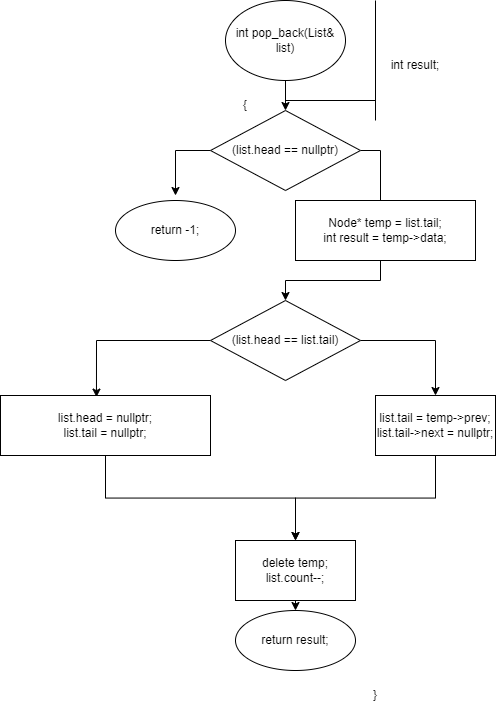
****

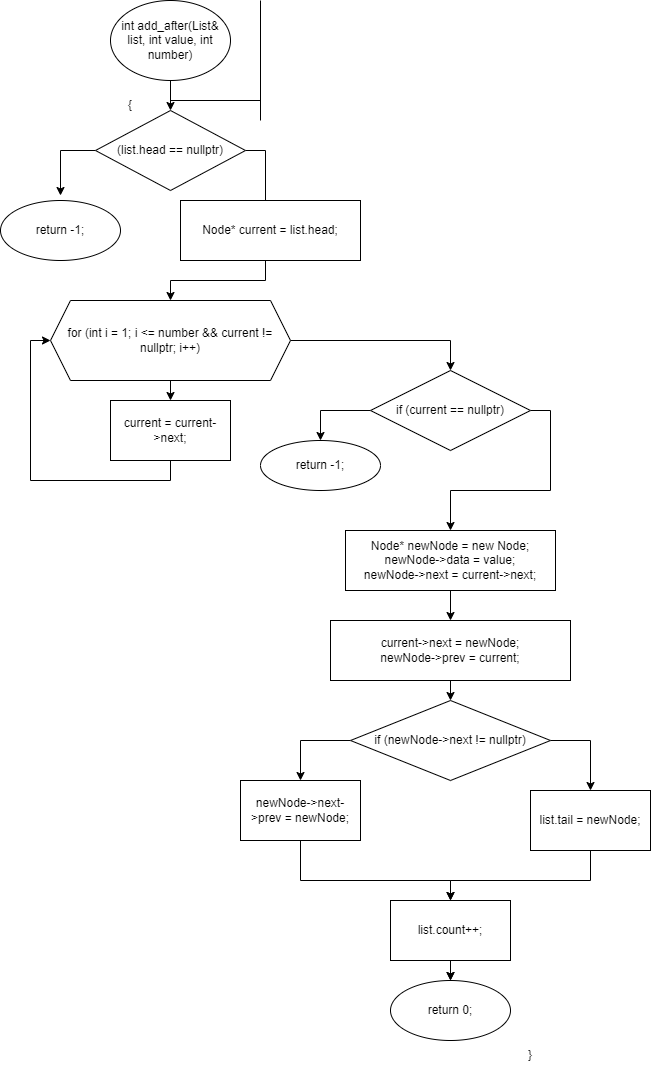
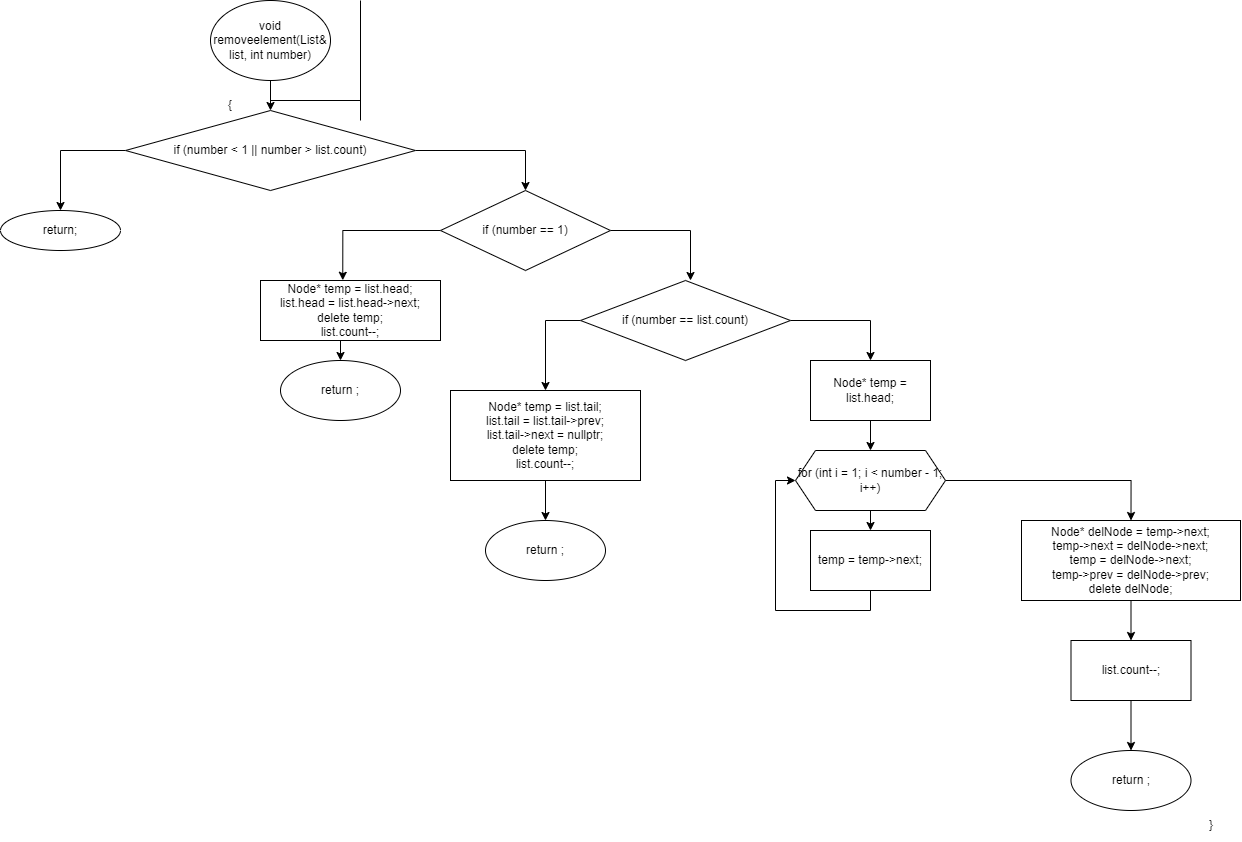
****

****

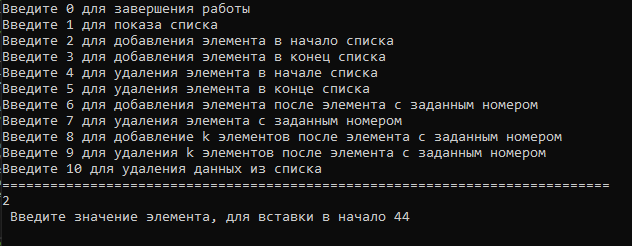
****

****

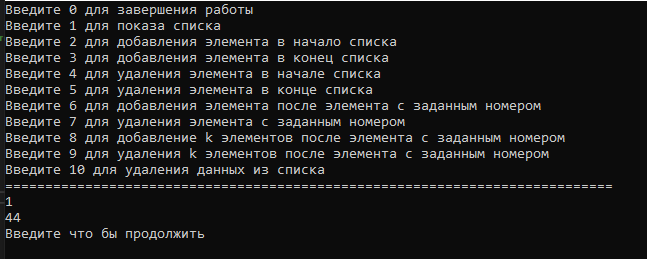
****

****

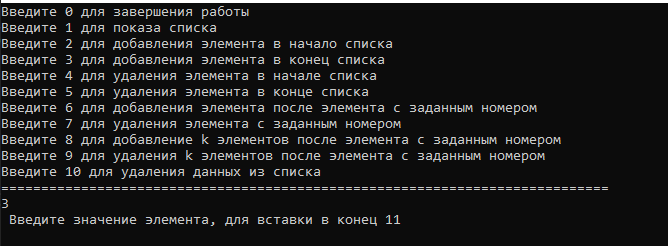
**Результат работы программы**

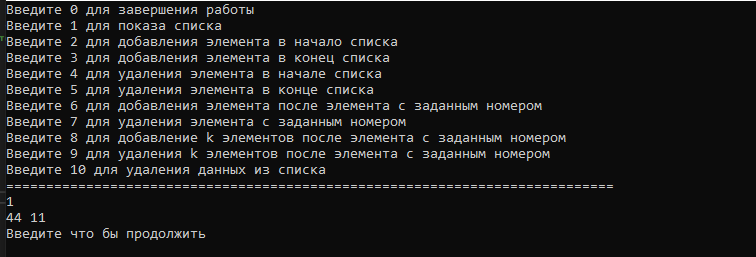


Добавление в начало

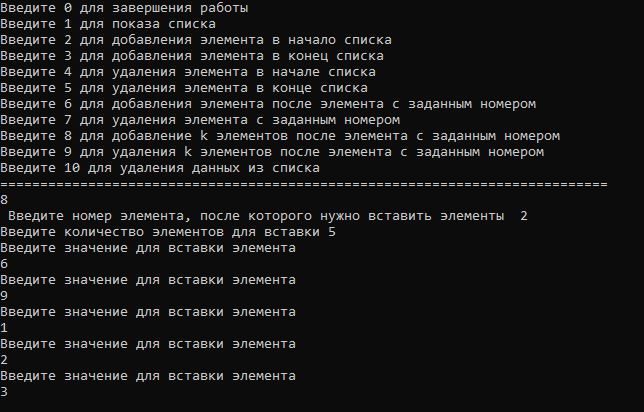


Показ списка

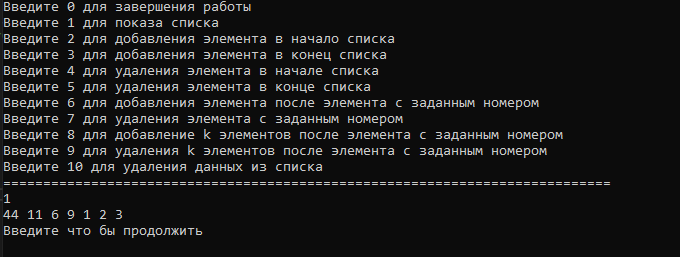
Добавление в конец



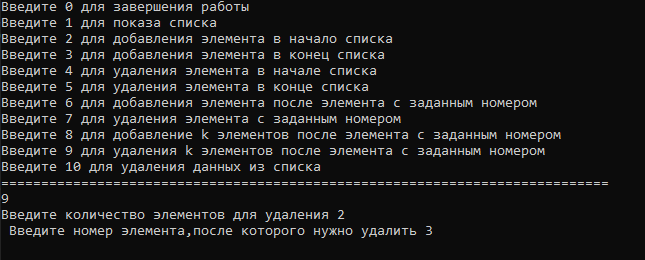
Показ списка



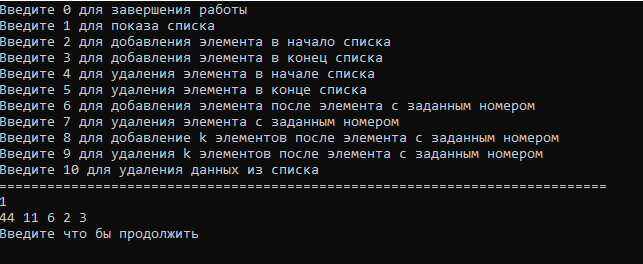
Добавление К элементов



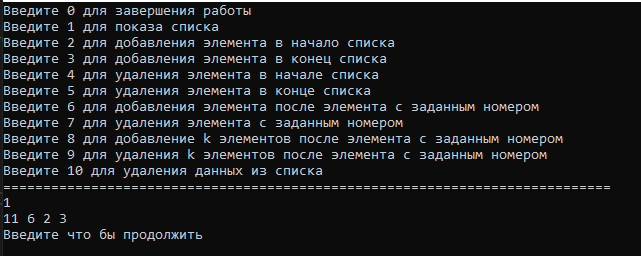
Показ списка



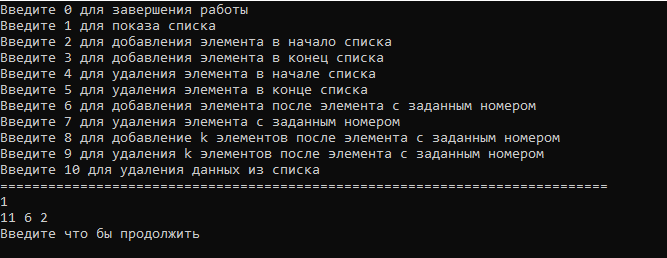
Удаление к элементов



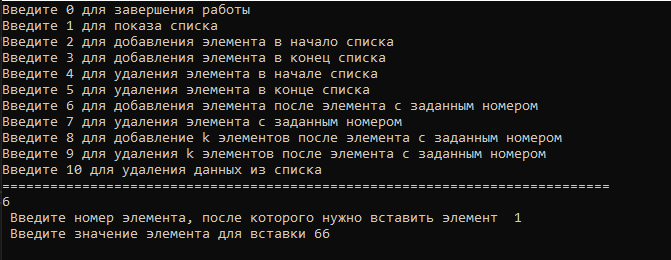
Показ списка



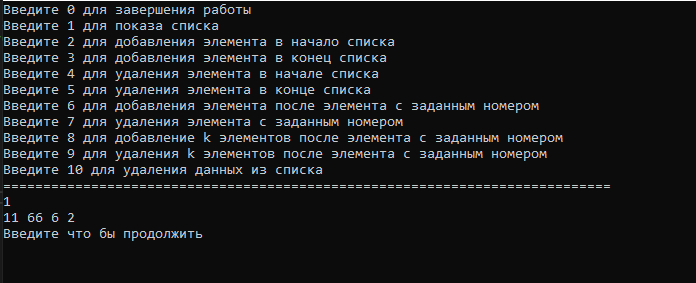
Удаление в начале списка



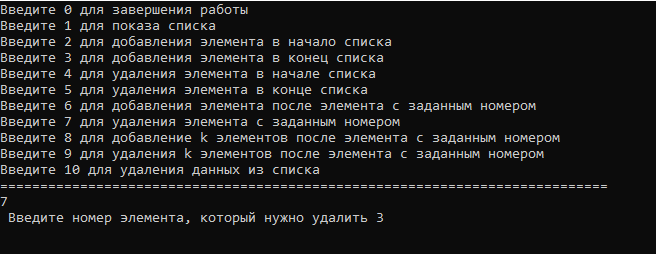
Удаление в конце списка



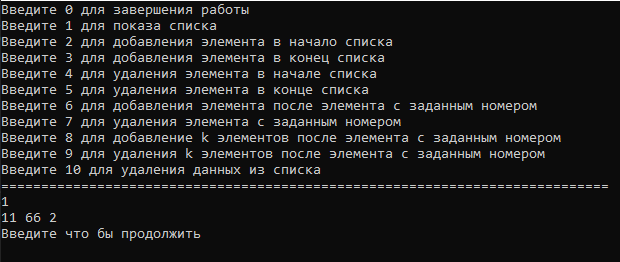
Добавление элемента после элемента с заданным номером



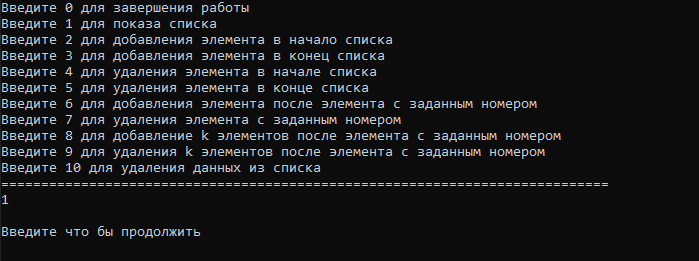
Показ списка



Удаление элемента с заданным номером



Показ списка



Удаление данных из списка