Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа**

**«Информационные динамические структуры»**

Выполнила:

студентка группы ИВТ-23-2Б

Сычева Евгения Андреевна

Проверила:

Доцент кафедры ИТАС

О. А. Полякова

2024 г.

**Постановка задачи:**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Разработать следующие функции:

1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

Задача №1:

Записи в линейном списке содержат ключевое поле типа \*char(строка символов). Сформировать однонаправленный список. Удалить K элементов из конца списка. Добавить элемент перед элементом с заданным номером.

Задача №2:

Записи в линейном списке содержат ключевое поле типа \*char(строка символов). Сформировать двунаправленный список. Удалить K элементов из конца списка. Добавить элемент перед элементом с заданным номером.

Задача №3:

Записи в стеке содержат ключевое поле типа \*char(строка символов). Сформировать стек. Удалить K элементов из конца стека. Добавить элемент перед элементом с заданным номером.

Задача №4:

Записи в очереди содержат ключевое поле типа \*char(строка символов). Сформировать очередь. Удалить K элементов из конца очереди. Добавить элемент перед элементом с заданным номером.

**Алгоритм работы:**

1. Написать функцию для создания списка. Функция может создавать пустой список, а затем добавлять в него элементы.
2. Написать функцию для печати списка. Функция должна предусматривать вывод сообщения, если список пустой.
3. Написать функции для удаления и добавления элементов списка в соответствии со своим вариантом.
4. Выполнить изменения в списке и печать списка после каждого изменения.
5. Написать функцию для записи списка в файл.
6. Написать функцию для уничтожения списка.
7. Записать список в файл, уничтожить его и выполнить печать (при печати должно быть выдано сообщение "Список пустой").
8. Написать функцию для восстановления списка из файла.
9. Восстановить список и распечатать его.
10. Уничтожить список.

**Код алгоритма для задачи №1:**

#include <iostream>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

using namespace std;

int n;

struct Node

{

string key;

Node\* next;

};

struct LinkedList

{

Node\* last, \* current, \* head;

void CL() //создание списка

{

cout << "Введите количество элементов: ";

cin >> n;

last = new Node;

cout << "Введите первый элемент: ";

cin >> last->key;

last->next = NULL;

head = last;

for (int i = 1; i < n; i++)

{

current = new Node;

cout << "Введите " << i + 1 << " элемент: ";

cin >> current->key;

last->next = current;

last = current;

}

last->next = NULL;

}

void SL() //печать списка

{

if (n)

{

current = head;

while (current != NULL)

{

cout << current->key << " ";

current = current->next;

}

cout << "\n------------\n";

}

else

{

cout << "Список пуст\n";

}

}

void DFL() //удаление элементов

{

if (head == nullptr) {

return;

}

int K;

cout << "Введите количество элементов: ";

cin >> K;

Node\* temp = head;

Node\* prev = nullptr;

int count = 0;

while (temp->next != nullptr) {

prev = temp;

temp = temp->next;

count++;

}

if (count < K) {

cout << "Недостаточно элементов в списке для удаления" << endl;

return;

}

temp = head;

for (int i = 0; i < count - K; i++) {

prev = temp;

temp = temp->next;

}

if (prev == nullptr) {

delete temp;

head = nullptr;

}

else {

delete temp;

prev->next = nullptr;

}

}

void AIL() //добавление элементов

{

int l;

cout << "Введите номер элемента, перед которым вставить элемент: ";

cin >> l;

last = l > 1 ? head : NULL;

for (int i = 0; i < l - 2; i++)

{

last = last->next;

}

Node\* E = last == NULL ? head : last->next;

l = 1;

n += l;

if (l)

{

current = new Node;

cout << "Введите элемент для добавления: ";

cin >> current->key;

if (last != NULL)

{

last->next = current;

}

else

{

head = current;

}

last = current;

}

for (int i = 1; i < l; i++)

{

current = new Node;

cout << "Введите " << i + 1 << " элемент";

cin >> current->key;

last->next = current;

last = current;

}

if (last != NULL)

{

last->next = E;

}

}

void WF() //запись списка в файл

{

ofstream F1("F1.txt");

current = head;

F1 << " " << n;

while (current != NULL)

{

F1 << " " << current->key;

current = current->next;

}

F1.close();

}

void RF() //восстановление списка из файла

{

ifstream F1("F1.txt");

F1 >> n;

head = new Node;

F1 >> head->key;

last = head;

while (F1.get() && !F1.eof())

{

current = new Node;

F1 >> current->key;

last->next = current;

last = current;

}

last->next = NULL;

F1.close();

}

void RL() //уничтожение списка

{

last = head;

while (last != NULL)

{

current = last->next;

delete last;

last = current;

}

n = 0;

}

};

int main()

{

system("chcp 1251 >NULL");

int a;

LinkedList lst;

while (true)

{

cout << "Введите номер задания в меню:\n 1. Создание списка\n 2. Удалить K элементов с конца\n 3. Добавить элемент в список с заданным ключем\n 4. Печать списка\n 5. Запись списка в файл\n 6. Уничтожение списка\n 7. Восстановление списка в файл\n";

cin >> a;

system("cls");

switch (a)

{

case 1:

lst.CL();

break;

case 2:

lst.DFL();

break;

case 3:

lst.AIL();

break;

case 4:

lst.SL();

break;

case 5:

lst.WF();

break;

case 6:

lst.RL();

break;

case 7:

lst.RF();

break;

}

}

return 0;

}

**Код алгоритма для задачи №2:**

#include <iostream>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

using namespace std;

int n;

struct Node {

string key;

Node\* next;

Node\* prev;

};

struct LinkedList {

Node\* last, \* current, \* head, \* tail;

void CL() {//CreateList

cout << "Введите количество элементов: ";

cin >> n;

last = new Node;

cout << "Введите первый элемент: ";

cin >> last->key;

last->next = NULL;

last->prev = NULL;

head = last;

for (int i = 1; i < n; i++) {

current = new Node;

cout << "Введите " << i + 1 << " элемент: ";

cin >> current->key;

current->prev = last;

last->next = current;

last = current;

}

tail = last;

last->next = NULL;

}

void SL() {//ShowList

if (n) {

current = head;

while (current != NULL) {

cout << current->key << ' ';

current = current->next;

}

cout << '\n';

current = tail;

while (current != NULL) {

cout << current->key << ' ';

current = current->prev;

}

cout << "\n----------------\n";

}

else {

cout << "Список пуст!\n";

}

}

void AIL() {//добавление элементов

int l;

cout << "Введите номер элемента: ";

cin >> l;

if (l > n / 2) {

last = tail;

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

last = last->prev;

}

Node\* E = last->next;

l = 1;

n += l;

if (l) {

current = new Node;

cout << "Введите элемент: ";

cin >> current->key;

current->prev = last;

last->next = current;

last = current;

}

E->prev = last;

last->next = E;

}

else {

last = l > 1 ? head : NULL;

for (int i = 0; i < l + 2; i++) {

last = last->next;

}

Node\* E = last == NULL ? head : last->next;

l = 1;

n += l;

if (l) {

current = new Node;

cout << "Введите элемент: ";

cin >> current->key;

if (last != NULL) {

current->prev = last;

last->next = current;

}

else {

current->prev = NULL;

head = current;

}

last = current;

}

if (last != NULL) {

E->prev = last;

last->next = E;

}

}

}

void DFL() {//удаление элементов из списка

if (head == nullptr) {

return;

}

int K;

cout << "Введите количество элементов которые надо удалить: ";

cin >> K;

Node\* temp = head;

Node\* prev = nullptr;

int count = 0;

while (temp->next != nullptr) {

prev = temp;

temp = temp->next;

count++;

}

if (count < K) {

cout << "Недостаточно элементов в списке для удаления" << endl;

return;

}

while (K > 0) {

Node\* toDelete = temp;

temp = temp->prev;

if (prev == nullptr) {

head = temp;

}

else {

prev->next = temp;

}

delete toDelete;

K--;

if (temp == nullptr) {

break;

}

}

}

void WF() {//WriteFile

ofstream F("F.txt");

current = head;

F << ' ' << n;

while (current != NULL) {

F << ' ' << current->key;

current = current->next;

}

F.close();

}

void RF() {//ReadFile

ifstream F("F.txt");

F >> n;

head = new Node;

F >> head->key;

head->prev = NULL;

last = head;

while (F.get() && !F.eof()) {

current = new Node;

F >> current->key;

current->prev = last;

last->next = current;

last = current;

}

last->next = NULL;

tail = last;

F.close();

}

void RL() {//RemoveList

last = head;

while (last != NULL) {

current = last->next;

delete last;

last = current;

}

n = 0;

}

};

int main() {

system("chcp 1251 > NULL");

int a;

LinkedList lst;

while (true) {

cout << "Введите номер команды меню:\n 1. Создание списка\n 2. Удалить K элементов с конца\n 3. Добавить элемент в список с заданным ключем\n 4. Печать списка\n 5. Запись списка в файл\n 6. Уничтожение списка\n 7. Восстановление списка из файла\n";

cin >> a;

system("cls");

switch (a) {

case 1:

lst.CL();

break;

case 2:

lst.DFL();

break;

case 3:

lst.AIL();

break;

case 4:

lst.SL();

break;

case 5:

lst.WF();

break;

case 6:

lst.RL();

break;

case 7:

lst.RF();

break;

}

}

return 0;

}

**Кол алгоритма для задачи №3:**

#include <iostream>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

using namespace std;

int n;

struct Node {

string key;

Node\* prev;

};

struct LinkedList {

Node\* last, \* current, \* tail, \* last2, \* current2, \* tail2;

void CL() {//CreateList

cout << "Введите количество элементов: ";

cin >> n;

last = NULL;

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

current = new Node;

cout << "Введите" << ' ' << i + 1 << " элемент: ";

cin >> current->key;

current->prev = last;

last = current;

}

tail = last;

}

void SL() {//ShowList

if (n) {

last2 = tail;

tail = tail->prev;

last2->prev = NULL;

while (tail != NULL) {//переписываем все элементы из стека 1 в стек 2, таким образом переворачивая исходный стек

current2 = tail;

tail = tail->prev;

current2->prev = last2;

last2 = current2;

}

tail = last2;//замена стека 1 стеком 2

last2 = NULL;

while (tail != NULL) {//выводим элементы и восстанавливаем исходный стек в стек 2

cout << tail->key << ' ';

current2 = tail;

tail = tail->prev;

current2->prev = last2;

last2 = current2;

}

cout << "\n----------------\n";

tail = last2;//заменяем стек 1 на стек 2

}

else {

cout << "Стек пуст!\n";

}

}

void AIL() {//AddIntoList

int l;

cout << "Введите номер элемента: ";

cin >> l;

last2 = tail;

tail = tail->prev;

last2->prev = NULL;

for (int i = 1; i < n - l + 1; i++) {//Перемещаем все элементы во стек 2 до нужного номера включительно

current2 = tail;

tail = tail->prev;

current2->prev = last2;

last2 = current2;

}

tail2 = last2;

l = 1;

n += l;

last = tail;

for (int i = 0; i < l; i++) {//Вводим новые K элементов в стек 1

current = new Node;

cout << "Введите элемент: ";

cin >> current->key;

current->prev = last;

last = current;

}

tail = last;

while (tail2 != NULL) {//возвращаем обратно списанные элементы

current = tail2;

tail2 = tail2->prev;

current->prev = last;

last = current;

}

tail = last;

}

void DFL() {//DeleteFromList

int K;

cout << "Введите количество элементов, которые надо удалить: ";

cin >> K;

current = NULL;

last = tail;

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

{

tail = last->prev;

delete last;

n--;

}

}

void WF() {//WriteFile

ofstream F("F.txt");

F << ' ' << n;

last2 = tail;

tail = tail->prev;

last2->prev = NULL;

while (tail != NULL) {//переписываем все элементы из стека 1 в стек 2, таким образом переворачивая исходный стек

current2 = tail;

tail = tail->prev;

current2->prev = last2;

last2 = current2;

}

tail = last2;//замена стека 1 стеком 2

last2 = NULL;

while (tail != NULL) {//записываем элементы в файл и восстанавливаем исходный стек в стек 2

F << ' ' << tail->key;

current2 = tail;

tail = tail->prev;

current2->prev = last2;

last2 = current2;

}

tail = last2;//заменяем стек 1 на стек 2

F.close();

}

void RF() {//ReadFile

ifstream F("F.txt");

F >> n;

last = NULL;

while (F.get() && !F.eof()) {

current = new Node;

F >> current->key;

current->prev = last;

last = current;

}

tail = last;

F.close();

}

void RL() {//RemoveList

last = tail;

while (last != NULL) {

current = last->prev;

delete last;

last = current;

}

n = 0;

}

};

int main() {

system("chcp 1251 > NULL");

int a;

LinkedList lst;

while (true) {

cout << "Введите номер команды меню:\n 1. Создание стека\n 2. Удалить K элементов с конца стека\n 3. Добавить элемент в стек с заданным ключем\n 4. Печать стека\n 5. Запись стека в файл\n 6. Уничтожение стека\n 7. Восстановление стека из файла\n";

cin >> a;

system("cls");

switch (a) {

case 1:

lst.CL();

break;

case 2:

lst.DFL();

break;

case 3:

lst.AIL();

break;

case 4:

lst.SL();

break;

case 5:

lst.WF();

break;

case 6:

lst.RL();

break;

case 7:

lst.RF();

break;

}

}

return 0;

}

**Код алгоритма для задачи №4:**

#include <iostream>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <fstream>

using namespace std;

int n;

struct Node {

string key;

Node\* next;

};

struct LinkedList {

Node\* last, \* current, \* head, \* last2, \* current2, \* head2;

void CL() {//CreateList

cout << "Введите количество элементов: ";

cin >> n;

last = new Node;

cout << "Введите 1 элемент: ";

cin >> last->key;

last->next = NULL;

head = last;

for (int i = 1; i < n; i++) {

current = new Node;

cout << "Введите" << ' ' << i + 1 << " элемент: ";

cin >> current->key;

last->next = current;

last = current;

}

last->next = NULL;

}

void SL() {//ShowList

if (n) {

last2 = head2 = head;

cout << head->key << ' ';

head = head->next;

last2->next = NULL;

while (head != NULL) {

cout << head->key << ' ';

last2->next = head;

last2 = last2->next;

head = head->next;

last2->next = NULL;

}

cout << "\n----------------\n";

head = head2;//замена очереди 1 очередью 2

}

else {

cout << "Очередь пуста!\n";

}

}

void AIL() {//AddIntoList

int l;

cout << "Введите номер элемента: ";

cin >> l;

if (l > 1) {

last2 = head2 = head;

head = head->next;

head2->next = NULL;

for (int i = 1; i < l - 1; i++) {//Перемещаем все элементы во вторую очередь до нужного номера

current2 = head;

head = head->next;

current2->next = NULL;

last2->next = current2;

last2 = current2;

}

l = 1;

n += l;

for (int i = 0; i < l; i++) {//Вводим новый элемент во вторую очередь

current2 = new Node;

cout << "Введите элемент: ";

cin >> current2->key;

last2->next = current2;

last2 = current2;

}

while (head != NULL) {//Списываем оставшиеся элементы из первой очереди во второй

last2->next = head;

last2 = last2->next;

head = head->next;

last2->next = NULL;

}

head = head2;//замена очереди 1 очередью 2

}

else {//Вторая очередь начинаниется с новых элементов

l = 1;

n += l;

if (l) {

current2 = new Node;

cout << "Введите элемент: ";

cin >> current2->key;

last2 = head2 = current2;

last2->next = NULL;

while (head != NULL) {//Списываем оставшиеся элементы из первой очереди во второй

last2->next = head;

last2 = last2->next;

head = head->next;

last2->next = NULL;

}

head = head2;//замена очереди 1 очередью 2

}

}

}

void DFL() {//DeleteFromList

string l;

l = n;

current = NULL;

last = head;

bool replace = false;

if (last->key == l) {//Если нужный элемент является первым, то просто удаляем его из очереди

head = last->next;

delete last;

replace = true;

n--;

}

else {

last2 = head2 = last;

last = last->next;

head2->next = NULL;

while (last != NULL) {

current2 = last;

last = last->next;

current2->next = NULL;

if (current2->key != l) {

last2->next = current2;

last2 = current2;

}

else {

delete current2;

n--;

}

}

head = head2;//замена очереди 1 очередью 2

}

}

void WF() {//WriteFile

ofstream F("F.txt");

F << ' ' << n;

last2 = head2 = head;

F << ' ' << head->key;

head = head->next;

last2->next = NULL;

while (head != NULL) {

F << ' ' << head->key;

last2->next = head;

last2 = last2->next;

head = head->next;

last2->next = NULL;

}

head = head2;//замена очереди 1 очередью 2

F.close();

}

void RF() {//ReadFile

ifstream F("F.txt");

F >> n;

head = new Node;

F >> head->key;

last = head;

while (F.get() && !F.eof()) {

current = new Node;

F >> current->key;

last->next = current;

last = current;

}

last->next = NULL;

F.close();

}

void RL() {//RemoveList

last = head;

while (last != NULL) {

current = last->next;

delete last;

last = current;

}

n = 0;

}

};

int main() {

system("chcp 1251 > NULL");

int a;

LinkedList lst;

while (true) {

cout << "Введите номер команды меню:\n 1. Создание стека\n 2. Удалить K элементов с конца стека\n 3. Добавить элемент в стек с заданным ключем\n 4. Печать стека\n 5. Запись стека в файл\n 6. Уничтожение стека\n 7. Восстановление стека из файла\n";

cin >> a;

system("cls");

switch (a) {

case 1:

lst.CL();

break;

case 2:

lst.DFL();

break;

case 3:

lst.AIL();

break;

case 4:

lst.SL();

break;

case 5:

lst.WF();

break;

case 6:

lst.RL();

break;

case 7:

lst.RF();

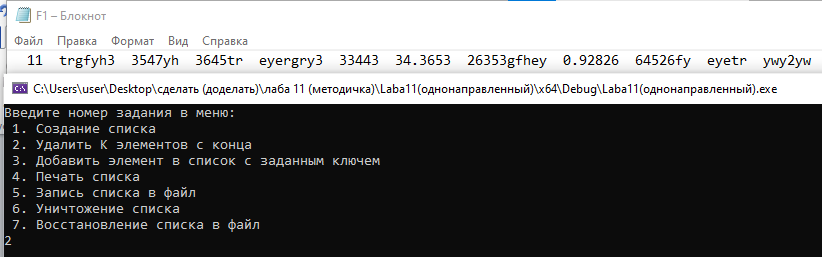
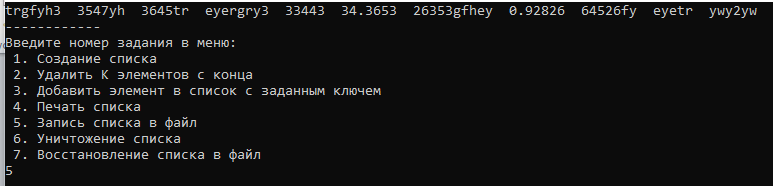
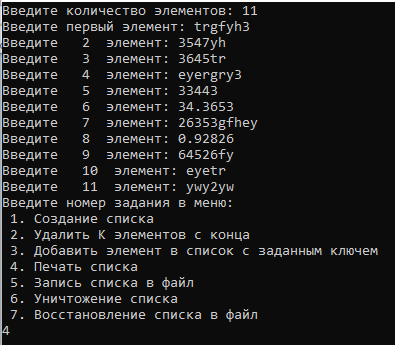
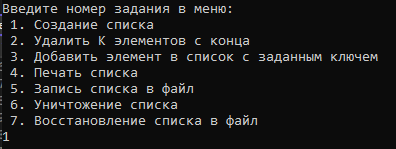
break;

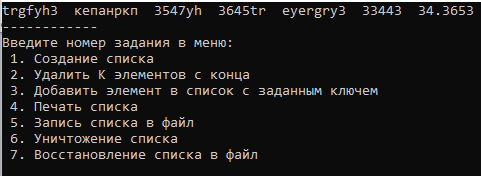
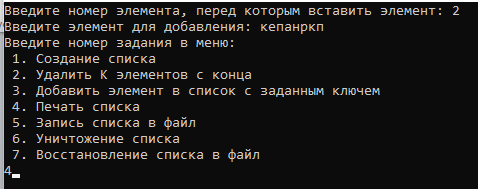
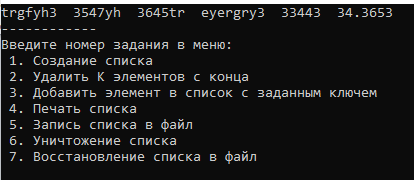
}

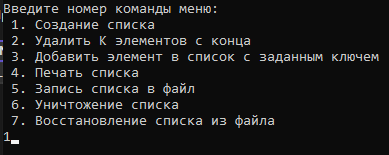
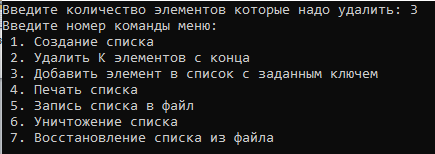
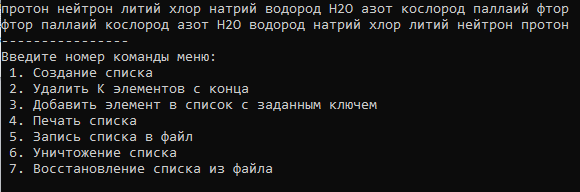
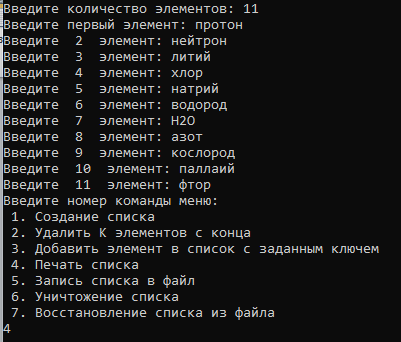
}

return 0;

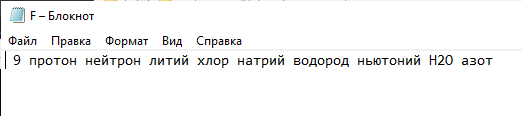
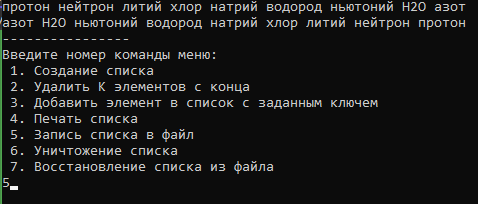
}

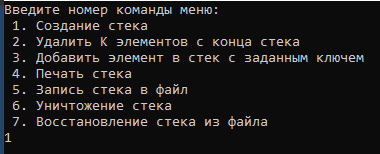
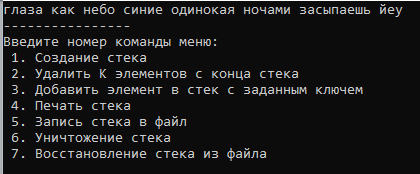
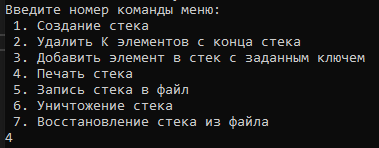
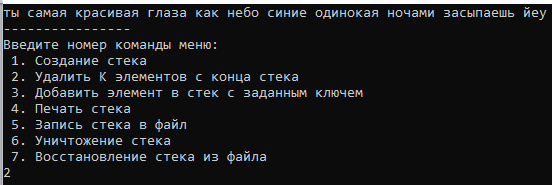
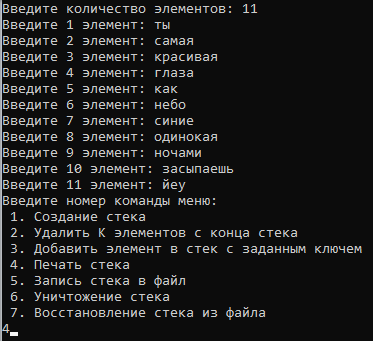
**Результат работы компилятора для задачи № 1:**

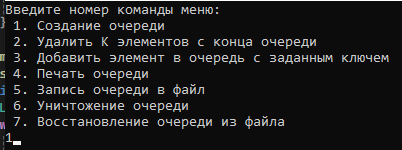
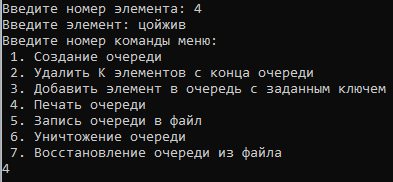
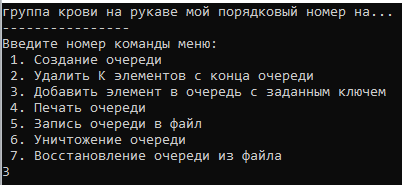
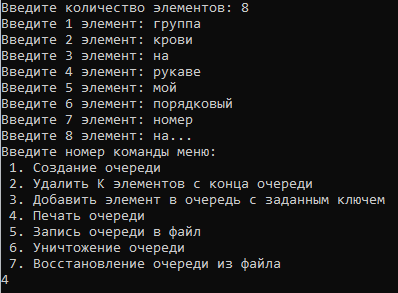
****

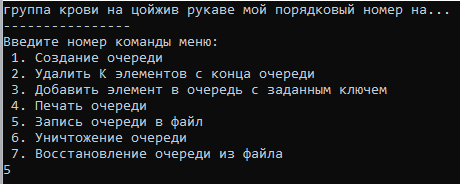
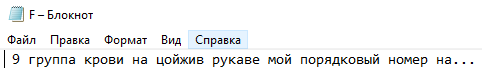
**Результат работы компилятора для задачи № 2:** ****

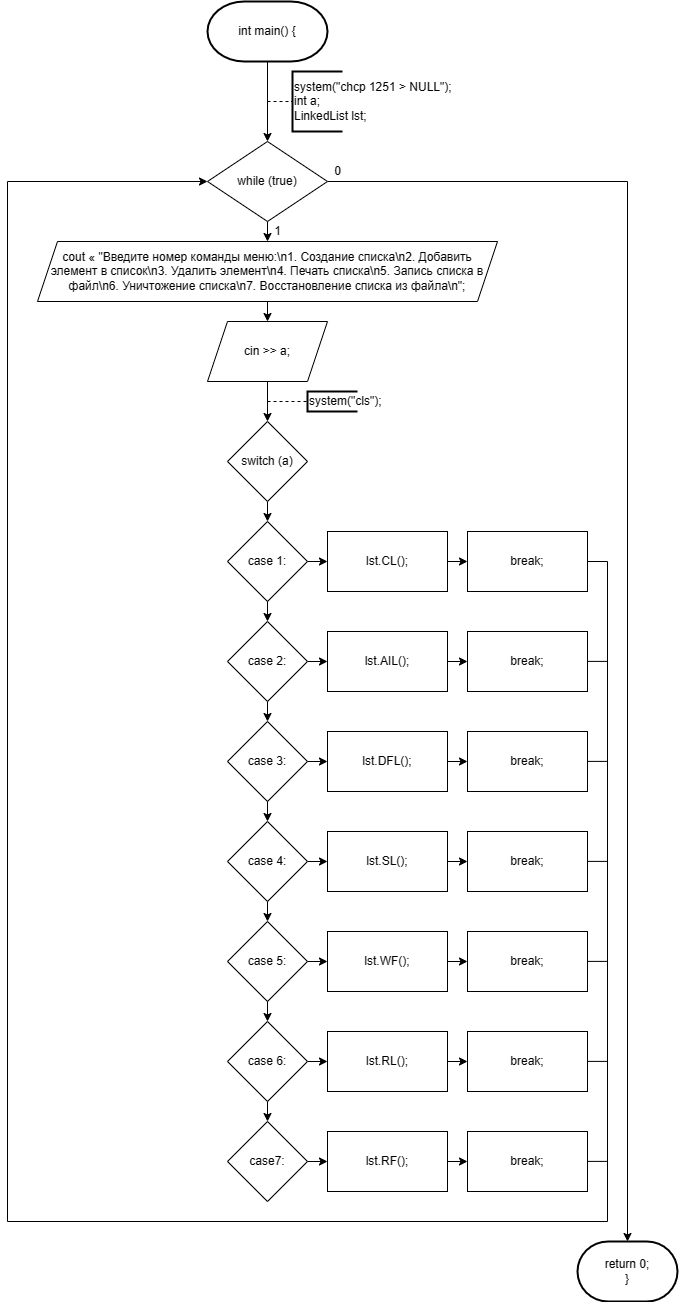
****

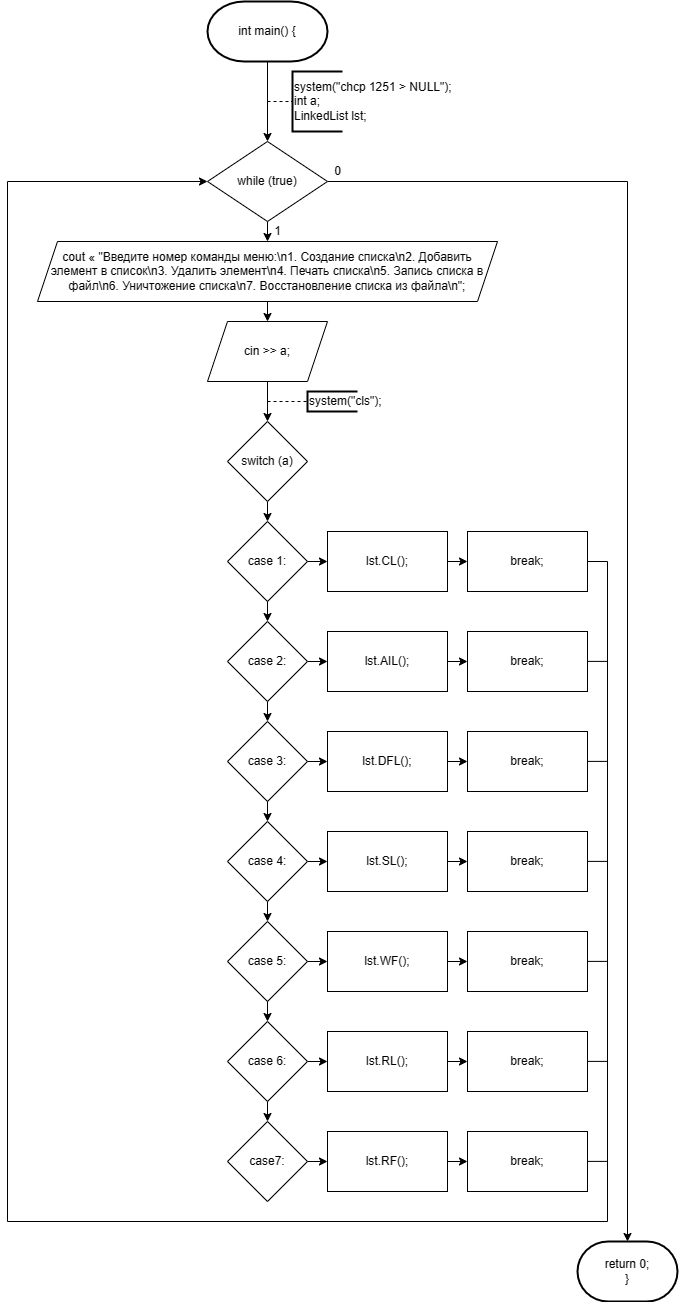
****

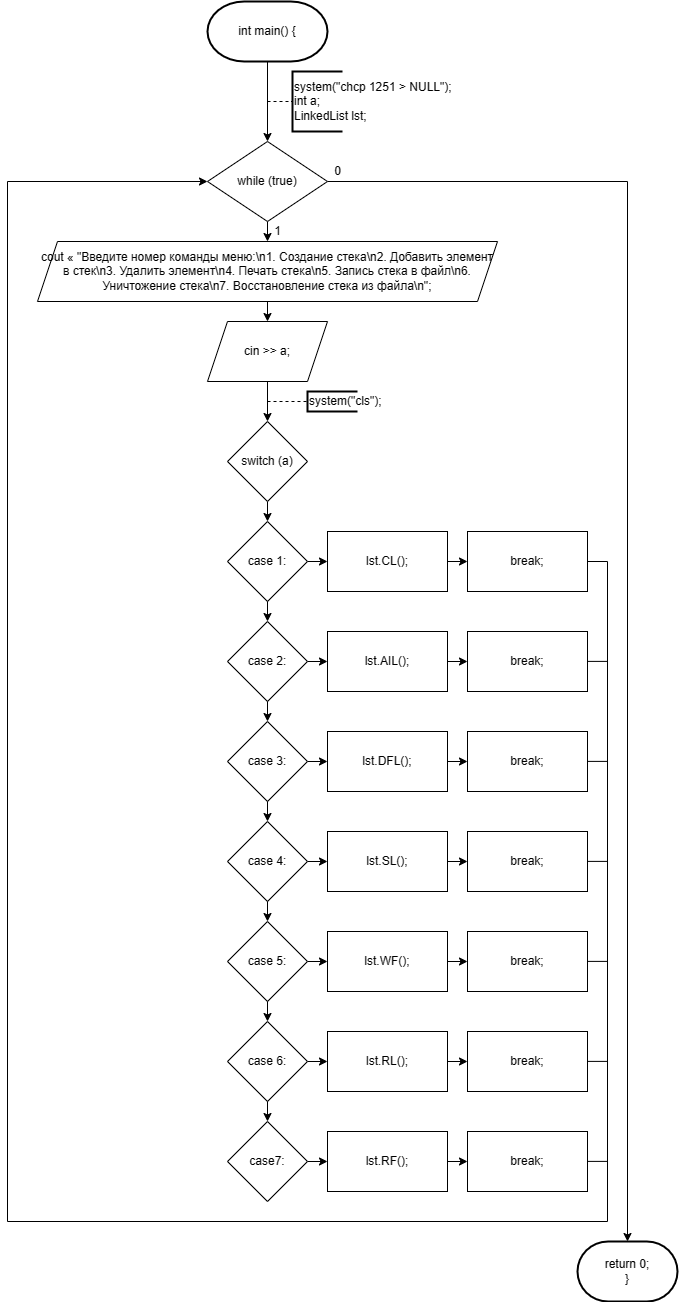
**Результат работы компилятора для задачи № 3:** ****

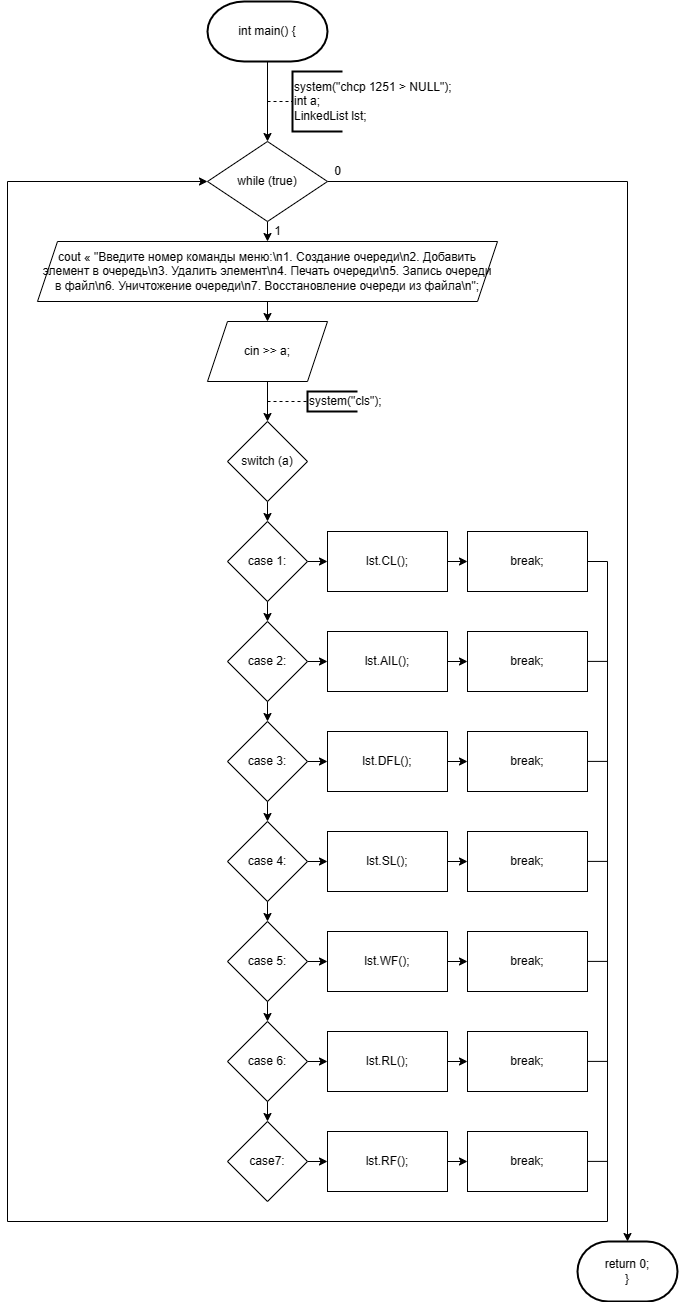
**Результат работы компилятора для задачи № 4:** ****

****

**Блок-схема алгоритма для задачи № 1:**

**Блок-схема алгоритма для задачи № 2: **

**Блок-схема алгоритма для задачи № 3: **

**Блок-схема алгоритма для задачи № 4:** 

**Ссылка на гит-хаб:**