Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра **«Информационные технологии и автоматизированные системы»**

направление подготовки: 09.03.04 - «Программная инженерия»

Лабораторная работа №11

По теме **«Информационный динамические структуры»**

**Вариант № 21**

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

Постановка задачи: написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Разработать следующие функции:

1. Создание списка (однонаправленного и двунаправленного), стека и очереди.
2. Добавление элемента в список (стек, очередь).
3. Удаление элемента из списка (стека, очереди).
4. Вывод списка (стека, очереди).
5. Запись списка (стека, очереди) в файл.
6. Уничтожение списка (стека, очереди).
7. Восстановление списка из файла.

Задание: записи в линейном списке содержат ключевое поле типа \*char(строка символов). Сформировать однонаправленный и двунаправленный список, стек и очередь. Удалить элементы перед и после элемента с заданным ключом. Добавить по К элементов в начало и в конец списка.

**Однонаправленный список**

Код

#include <iostream>

#include <Windows.h>

#include <fstream>

using namespace std;

struct Node

{

char data;

Node\* next;

void add(char Data)

{

data = Data;

next = nullptr;

}

};

struct list

{

Node\* head = nullptr;

Node\* tail = nullptr;

void push\_b(char data)

{

Node\* node = new Node; // новый узел

node->add(data); // добавление элемента

if (head == nullptr)

head = node;

if (tail != nullptr)

tail->next = node;

tail = node;

}

void push\_f(char data)

{

Node\* node = new Node;

node->add(data);

node->next = head;

head = node;

if (tail == nullptr)

tail = node;

}

void pop\_f()

{

if (head == nullptr) return;

if (head == tail)

{

delete tail;

head = tail = nullptr;

return;

}

Node\* node = head;

head = node->next;

delete node;

}

Node\* get(int k) // возвращение адреса элемента по его индексу

{

if (k < 0) return 0;

Node\* node = head;

int n = 0;

while (node && n != k && node->next)

{

node = node->next;

n++;

}

if (n == k) return node;

else return nullptr;

}

void erase(int k) // удаление по индексу

{

if (k < 0) return;

if (k == 0)

{

pop\_f();

return;

}

Node\* left = get(k - 1);

Node\* node = left->next;

if (node == nullptr) return;

Node\* right = node->next;

left->next = right;

if (node == tail)

tail = left;

delete node;

}

};

void print(list l) // вывод списка

{

cout << "Список: ";

Node\* node = l.head;

while (node != nullptr)

{

cout << node->data << ' ';

node = node->next;

}

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

list l;

char temp;

int lot;

cout << "Введите количество символов: ";

cin >> lot;

cout << "Вводите символы (через enter): " << endl;

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

{

cin >> temp;

l.push\_b(temp);

}

print(l);

int k;

cout << "Введите номер элемента, перед которым и после которого нужно удалить символы: ";

cin >> k;

l.erase(k);

l.erase(k - 2);

print(l);

cout << "Сколько элементов добавить в начало и конец списка? ";

cin >> k;

cout << "Введите значение этих элементов (через enter): " << endl;

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

{

cin >> temp;

l.push\_f(temp);

}

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

{

cin >> temp;

l.push\_b(temp);

}

print(l);

ofstream file1("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_список1\\list.txt");

if (!file1.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

Node\* node = l.head;

while (node != nullptr)

{

file1 << node->data << '\n';

node = node->next;

}

file1.close();

system("pause");

for (int i = 0; i < lot + 2 \* k - 2; i++)

l.pop\_f();

print(l);

ifstream file2("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_список1\\list.txt");

if (!file2.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

char s;

for (int i = 0; i < lot + 2 \* k - 2; i++)

{

file2 >> s;

l.push\_b(s);

}

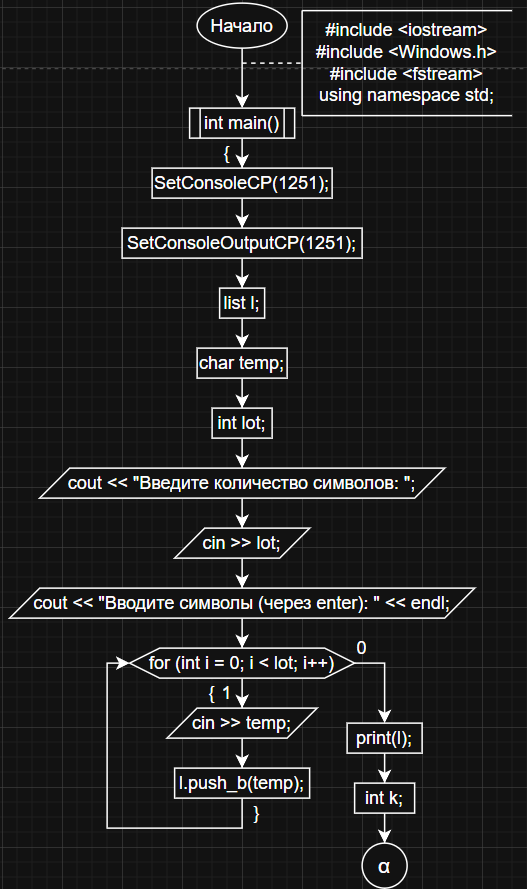
file2.close();

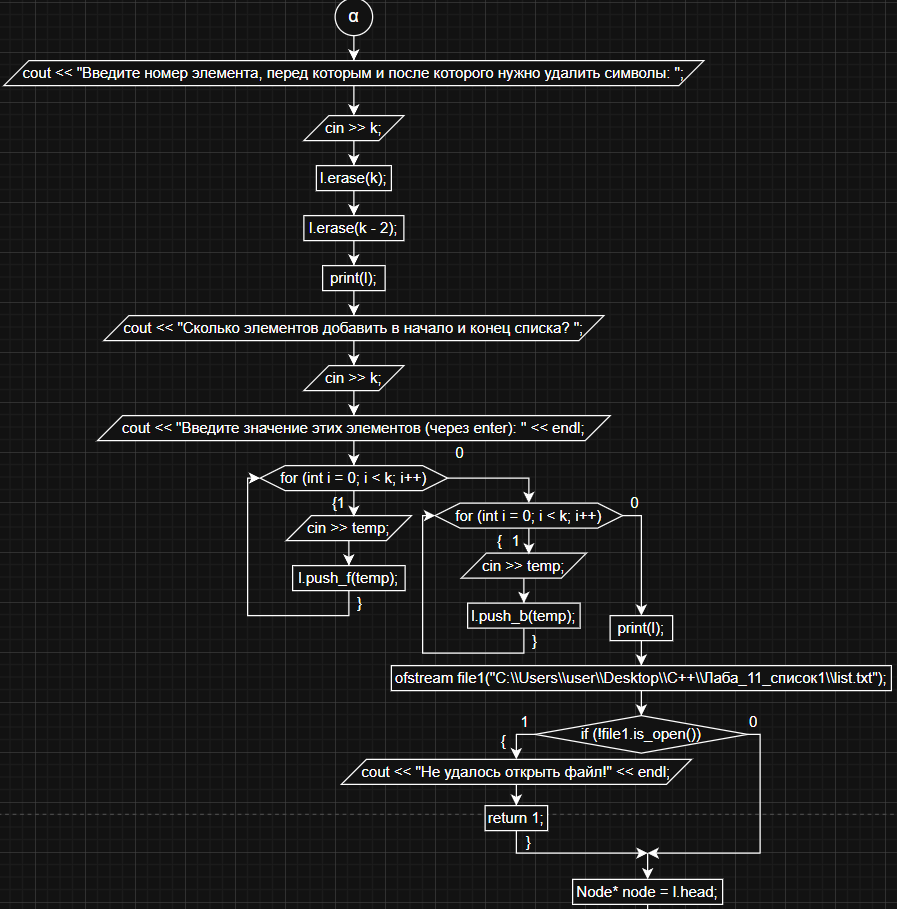
print(l);

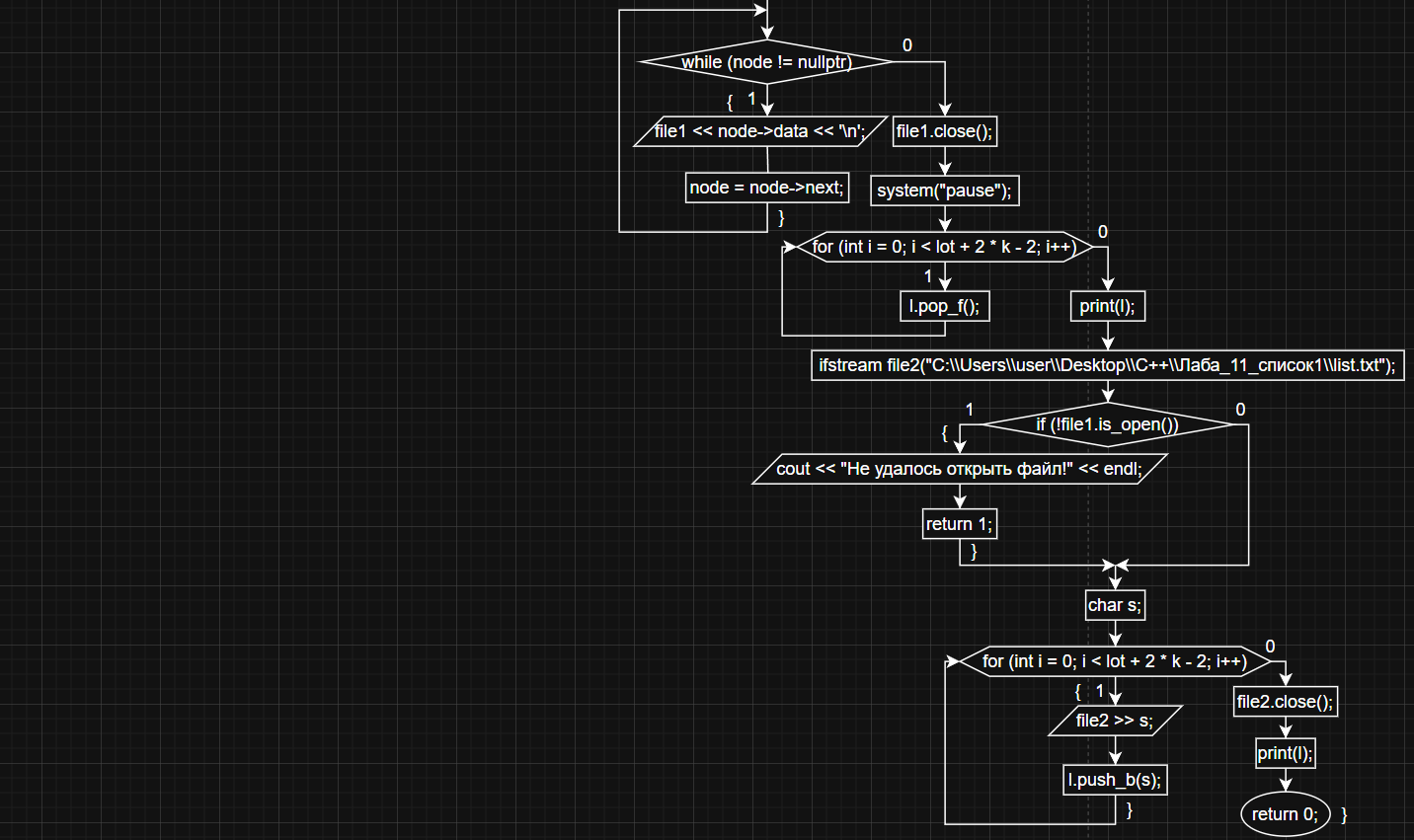
return 0;

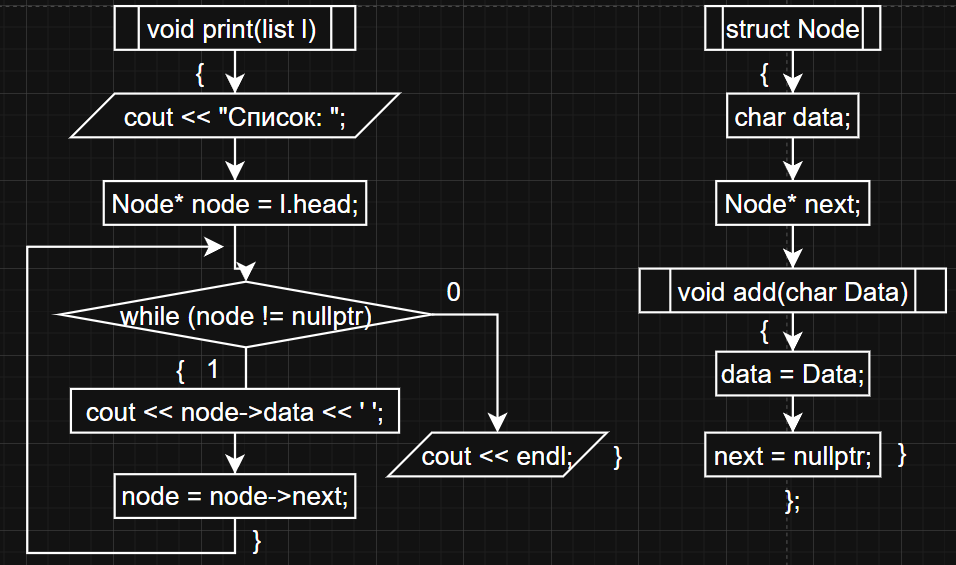
}

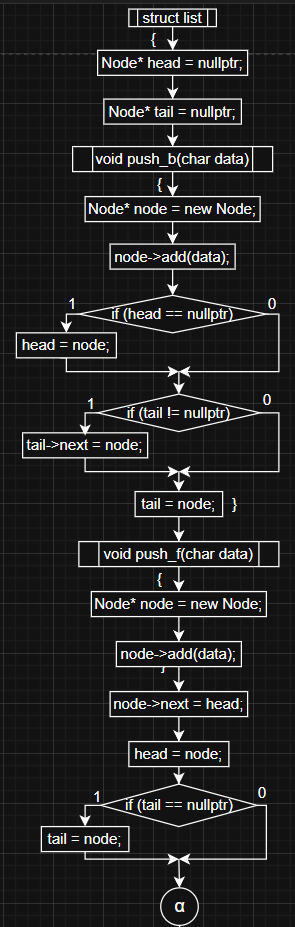
Блок-схема

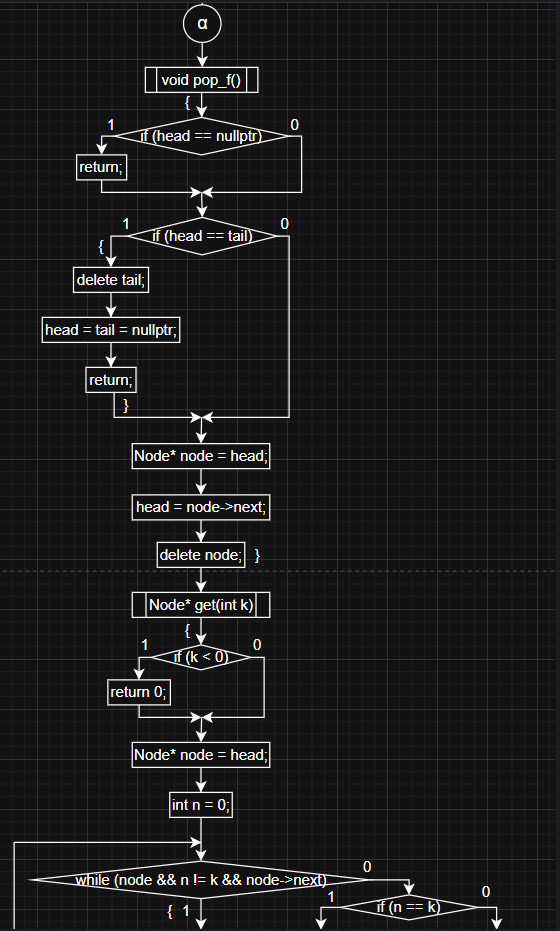


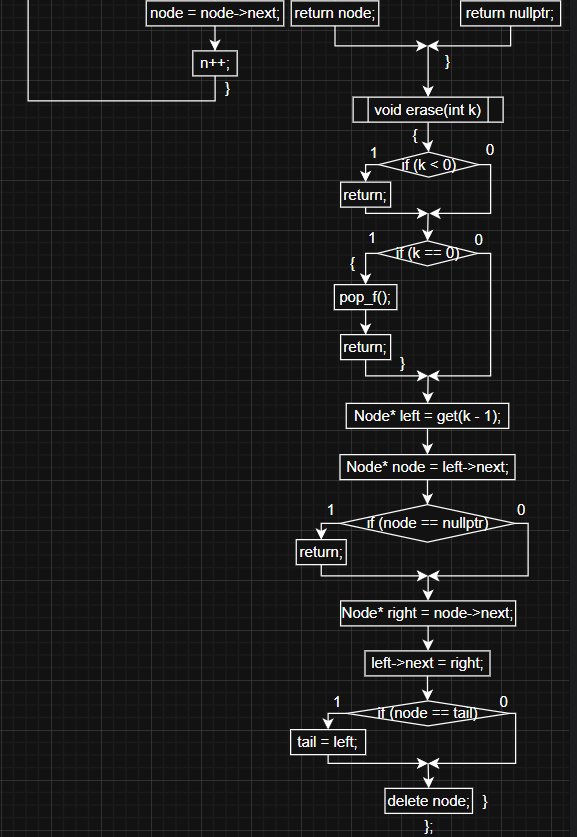




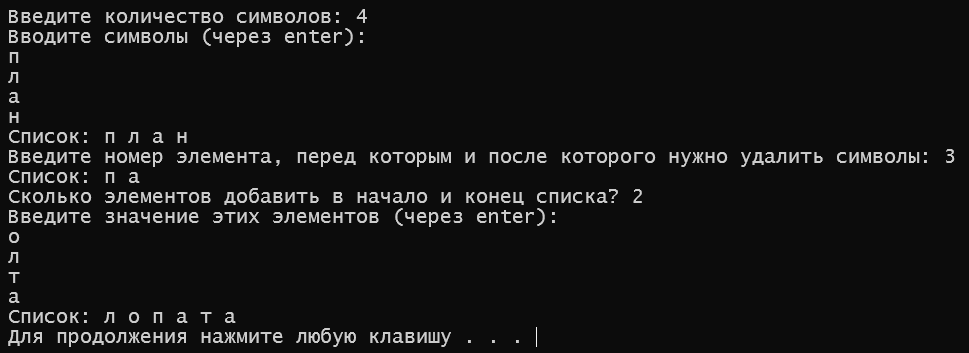


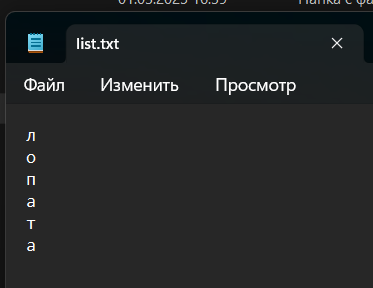


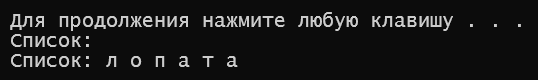




Результат работы программы:







**Двунаправленный список**

Программа отличается только функциями, а главная функция main() остаётся прежней (за исключением местонахождения файла).

Код

struct Node

{

char data;

Node\* prev;

Node\* next;

void add(char Data)

{

data = Data;

next = nullptr;

prev = nullptr;

}

};

struct list

{

Node\* head = nullptr;

Node\* tail = nullptr;

void push\_b(char data)

{

Node\* node = new Node;

node->add(data);

if (head == nullptr)

head = tail = node;

else

{

tail->next = node;

node->prev = tail;

tail = node;

}

}

void push\_f(char data)

{

Node\* node = new Node;

node->add(data);

if (head == nullptr)

head = tail = node;

else

{

node->next = head;

head->prev = node;

head = node;

}

}

void pop\_f()

{

if (head == nullptr) return;

if (head == tail)

{

delete head;

head = tail = nullptr;

return;

}

Node\* node = head;

head = head->next;

head->prev = nullptr;

delete node;

}

void pop\_b()

{

if (tail == nullptr) return;

if (head == tail)

{

delete tail;

head = tail = nullptr;

return;

}

Node\* node = tail;

tail = tail->prev;

tail->next = nullptr;

delete node;

}

Node\* get(int k)

{

if (k < 0) return nullptr;

Node\* node = head;

int n = 0;

while (node && n != k)

{

node = node->next;

n++;

}

return node;

}

void erase(int k)

{

if (k < 0) return;

if (k == 0)

{

pop\_f();

return;

}

Node\* node = get(k);

if (node == nullptr) return;

if (node == tail)

{

pop\_b();

return;

}

node->prev->next = node->next;

node->next->prev = node->prev;

delete node;

}

void insert(int k, char data) // добавление элемента по его индексу

{

if (k < 0) return;

if (k == 0)

{

push\_f(data);

return;

}

Node\* left = get(k - 1);

if (left == nullptr) return;

Node\* node = new Node;

node->add(data);

node->next = left->next;

node->prev = left;

if (left->next)

left->next->prev = node;

left->next = node;

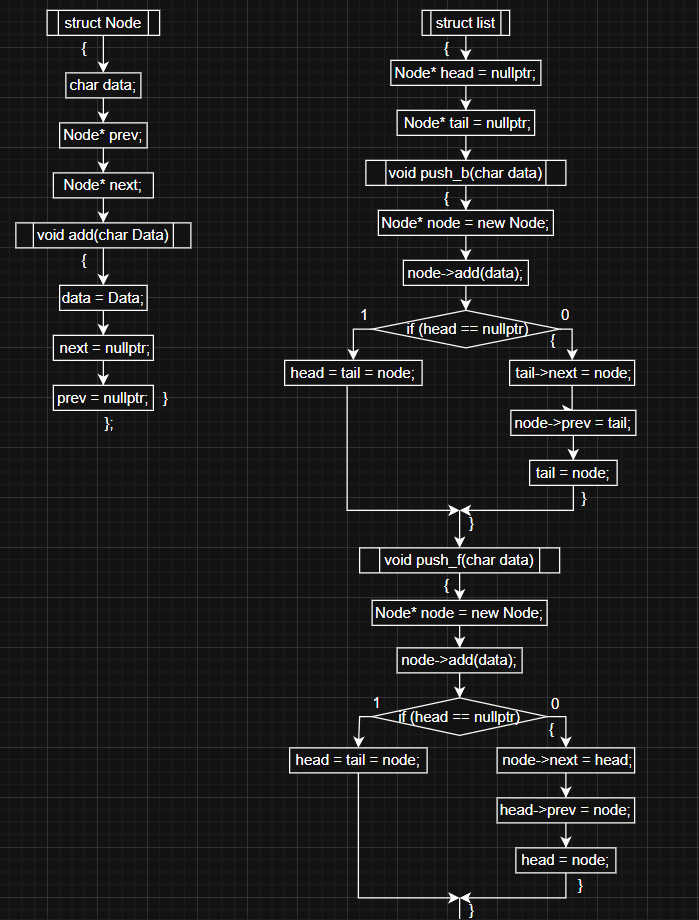
if (node->next == nullptr)

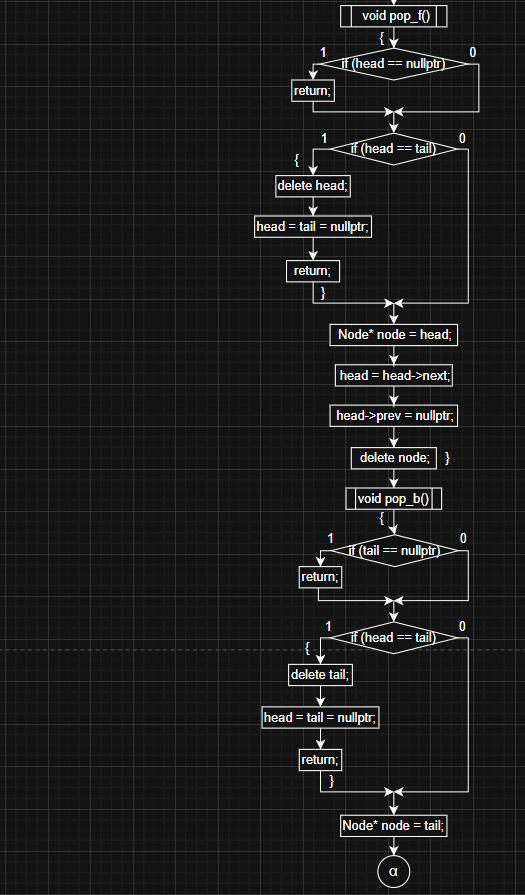
tail = node;

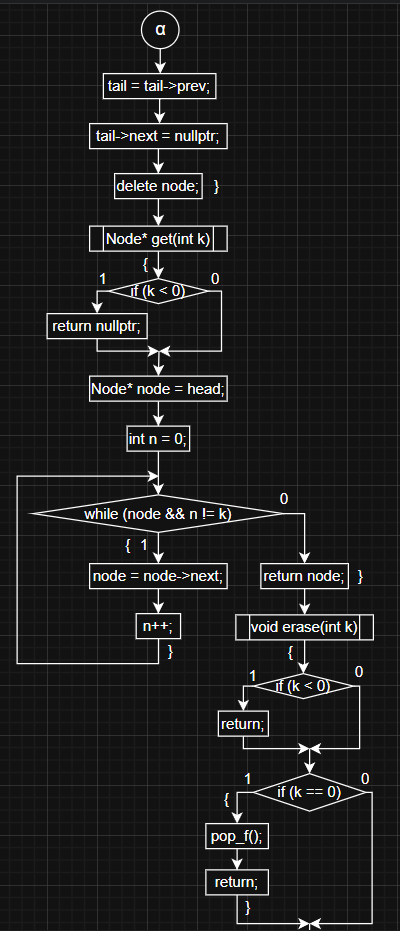
}

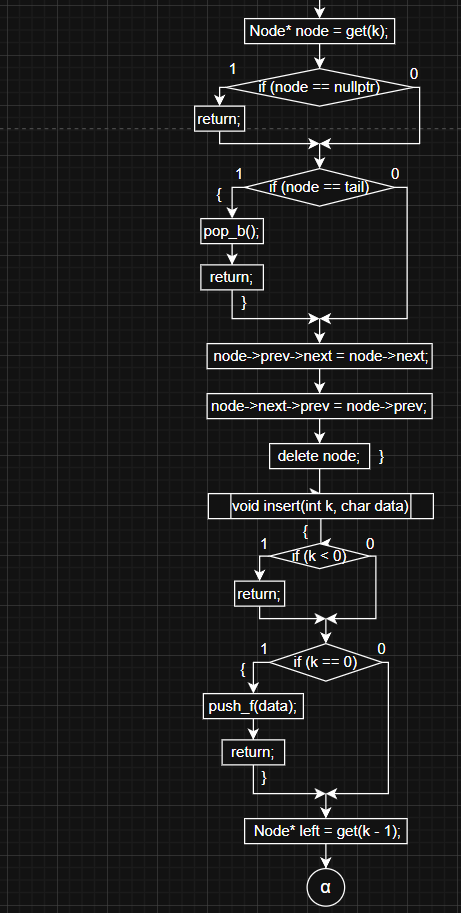
};

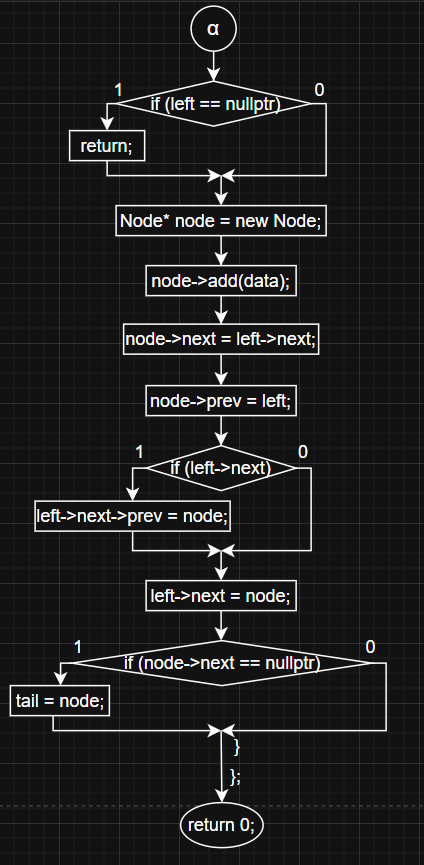
Блок-схема



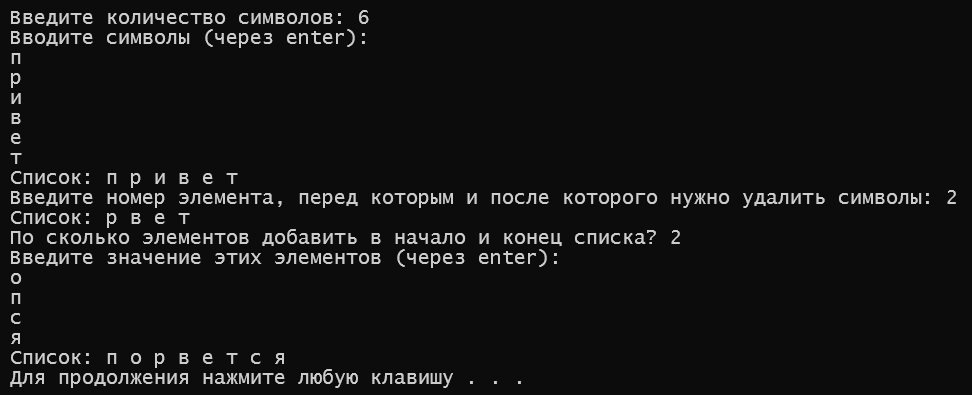


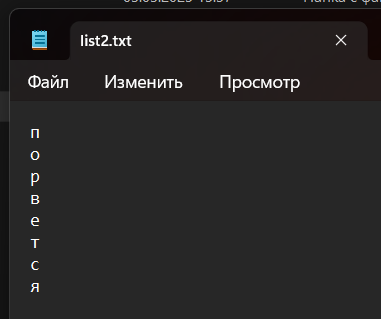


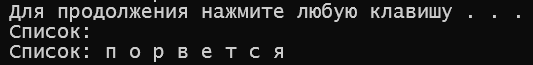




Результат работы программы:







**Стеки**

Код

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

struct Node

{

char data;

Node\* next;

};

struct Stack

{

Node\* head;

int size;

void setN(Node\*& temp, char data, Node\* next)

{

temp = new Node;

temp->next = next;

temp->data = data;

return;

}

void setS(Stack& myst)

{

myst.head = nullptr;

myst.size = 0;

return;

}

void push(Stack& myst, const char& data)

{

Node\* newN;

setN(newN, data, myst.head);

myst.head = newN;

myst.size++;

return;

}

void pop(Stack& myst)

{

if (myst.head != nullptr)

{

Node\* del\_el = myst.head;

myst.head = myst.head->next;

delete del\_el;

myst.size--;

}

return;

}

void clear(Stack& myst)

{

while (myst.head != NULL)

{

Node\* del\_el = myst.head;

myst.head = myst.head->next;

delete del\_el;

myst.size--;

}

return;

}

};

void print(Stack& myst)

{

Node\* tep = myst.head;

cout << "Стек: ";

while (tep != 0)

{

cout << tep->data << ' ';

tep = tep->next;

}

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stack stack;

stack.setS(stack);

char temp;

int n;

cout << "Введите количество символов: ";

cin >> n;

cout << "Вводите эти символы (через Enter)" << endl;

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

{

cin >> temp;

stack.push(stack, temp);

}

print(stack);

int k;

cout << "Введите номер ключа, до и после которого нужно удалить элементы: ";

cin >> k;

char buf[100];

int i = 0, j = 0;

while (stack.head != nullptr)

{

if (i == k or i == k - 2)

stack.pop(stack);

else

{

buf[j] = stack.head->data;

stack.pop(stack);

j++;

}

i++;

}

cout << "Стек: ";

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

cout << buf[i] << ' ';

cout << endl;

cout << "По сколько элементов нужно добавить в конец и начало стека? ";

cin >> k;

cout << "Вводите эти зачения (через Enter)" << endl;

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

{

cin >> temp;

stack.push(stack, temp);

}

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

stack.push(stack, buf[i]);

system("pause");

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

{

cin >> temp;

stack.push(stack, temp);

}

print(stack);

ofstream file1("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_стеки\\mystack.txt");

if (!file1.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

Node\* tp = stack.head;

while (tp != 0)

{

file1 << tp->data << '\n';

tp = tp->next;

}

file1.close();

system("pause");

stack.clear(stack);

print(stack);

ifstream file2("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_стеки\\mystack.txt");

if (!file2.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

char s;

for (int i = 0; i < n + 2; i++)

{

file2 >> s;

stack.push(stack, s);

}

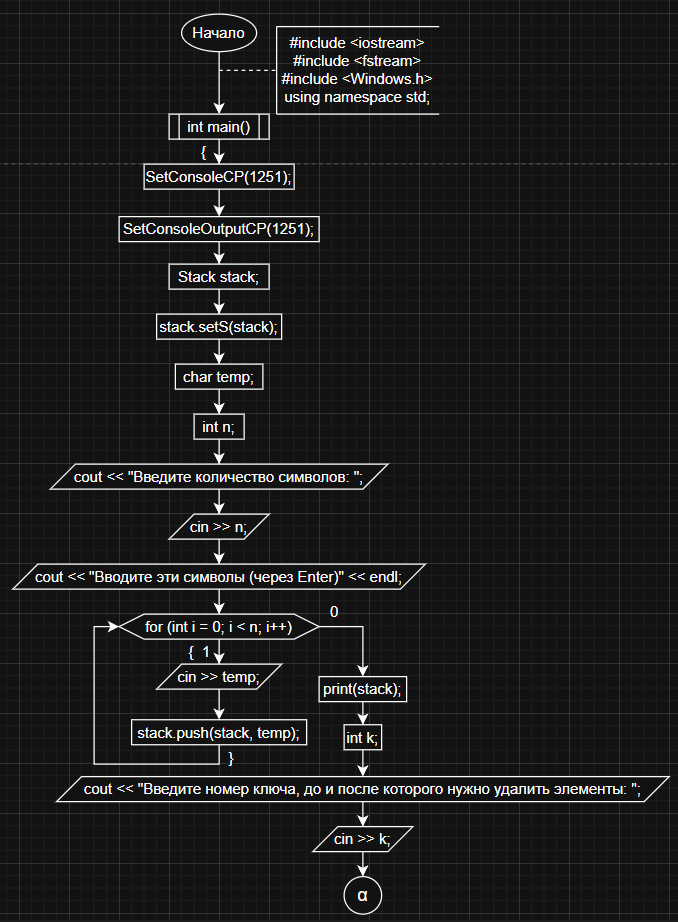
file2.close();

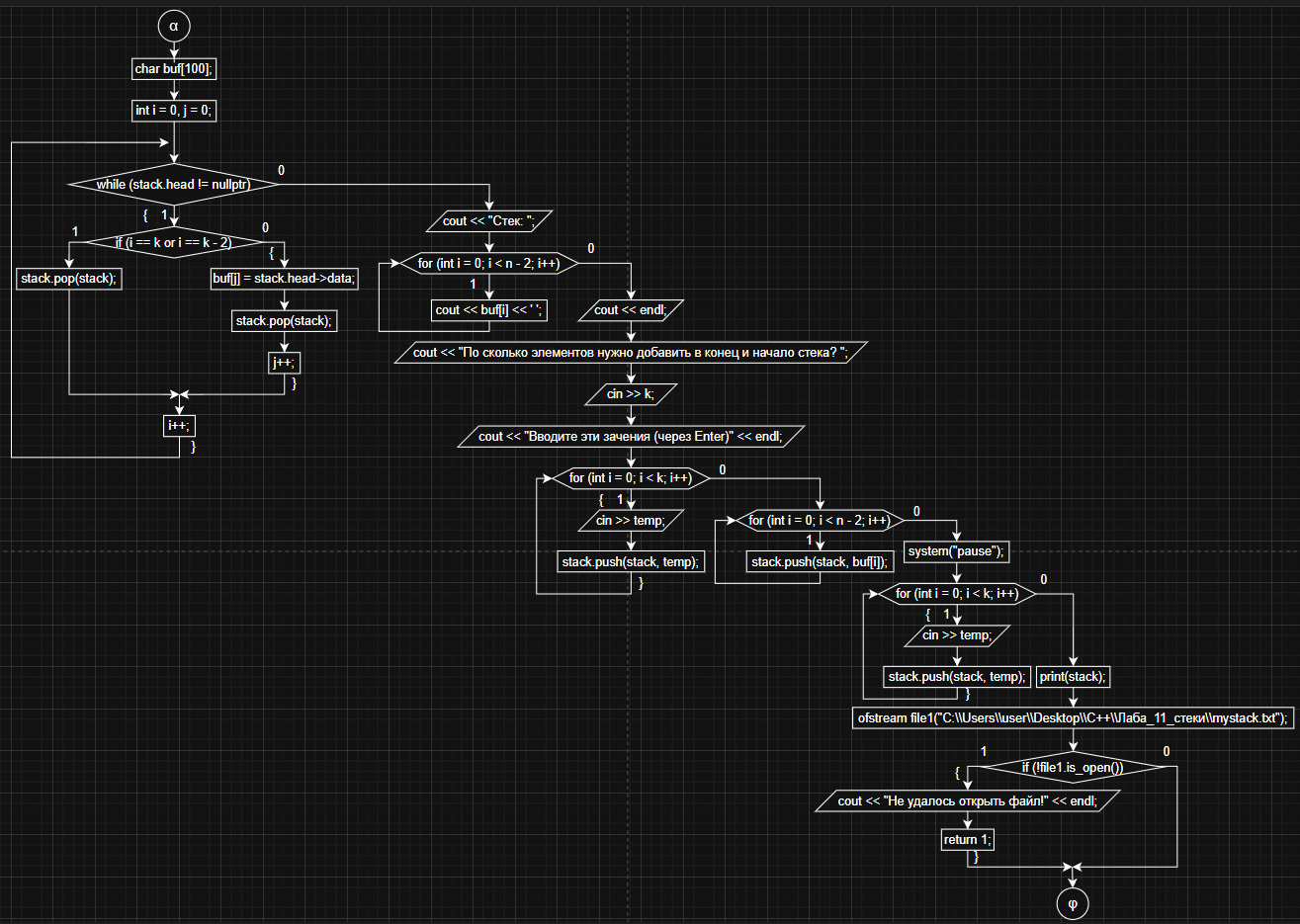
print(stack);

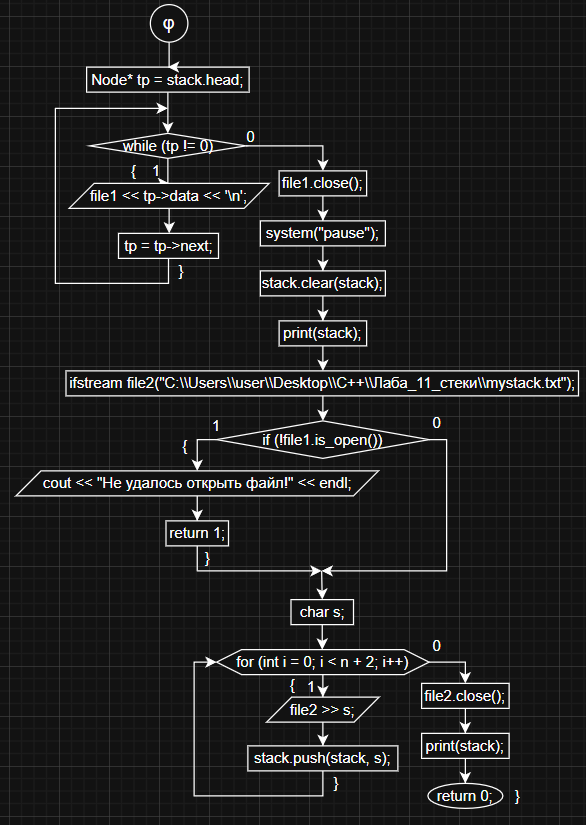
return 0;

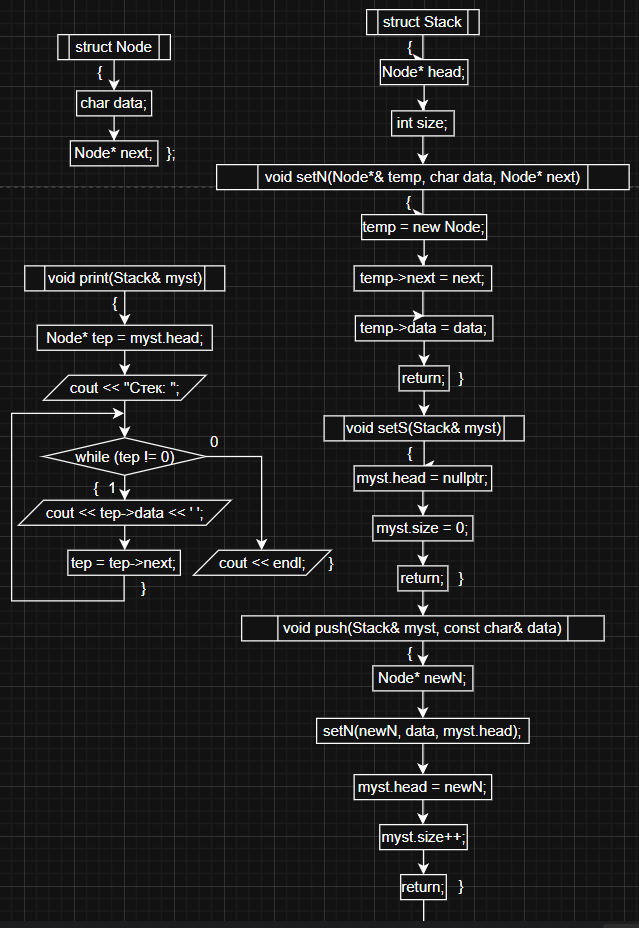
}

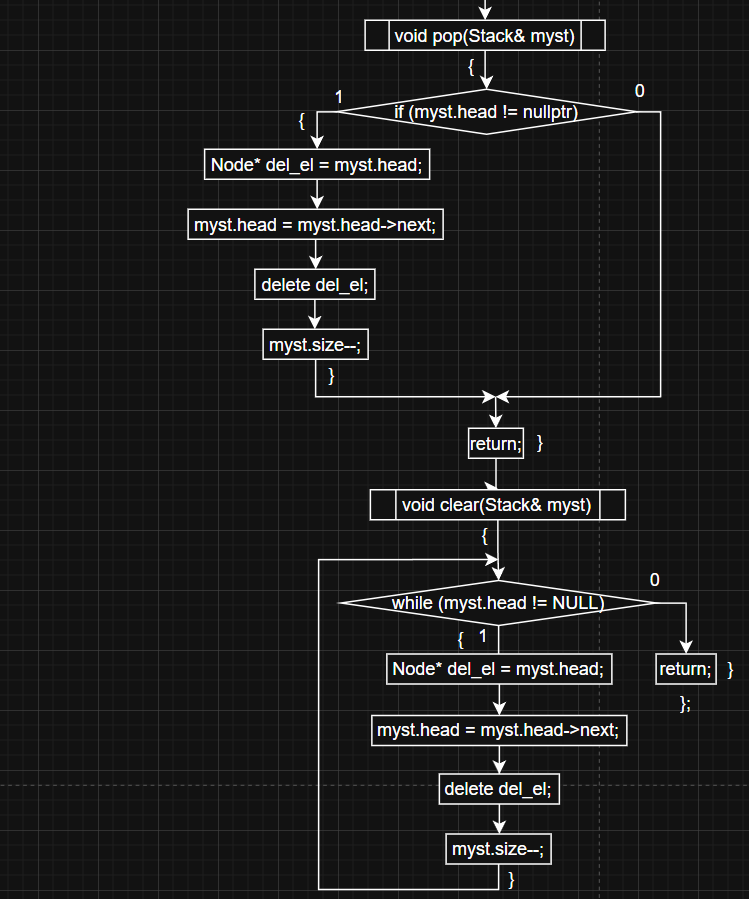
Блок-схема



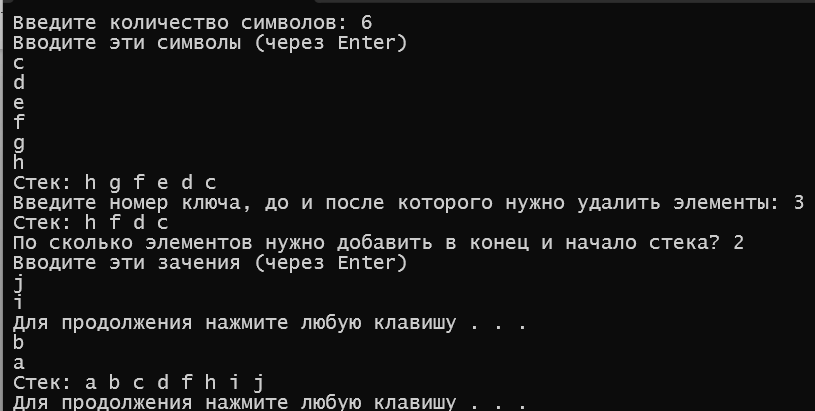


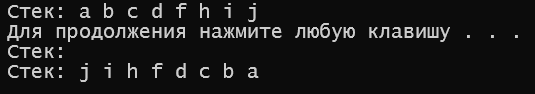
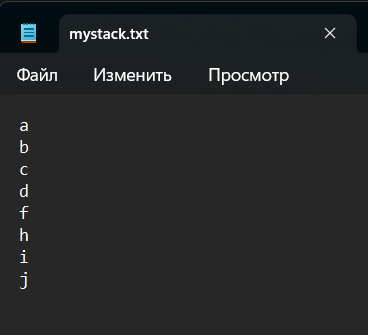






Результат работы программы:





**Очереди**

Код

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

struct Node

{

char data;

Node\* next;

};

struct Queue

{

int size;

Node\* head;

Node\* tail;

void initQ(Queue& q, char value) //инициализация очереди

{

Node\* newN = new Node;

newN->data = value;

newN->next = nullptr;

q.head = newN;

q.tail = newN;

q.size = 1;

return;

}

void pop(Queue& q) //удаление из начала очереди

{

if (q.head != nullptr)

{

Node\* temp = q.head;

q.head = q.head->next;

delete temp;

q.size--;

}

}

void push(Queue& q, char data) //добавление в конец очереди

{

Node\* newN = new Node{ data, nullptr };

q.tail->next = newN;

q.tail = newN;

q.size++;

}

void remove(Queue& q, int key) //удаление по индексу

{

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

{

if (i == key)

pop(q);

else

{

push(q, q.head->data);

pop(q);

}

}

}

void insert(Queue& q, int k, int n) //вставка элемента по номеру

{ //k - количество добавляемых элементов, n - номер

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

{

push(q, q.head->data);

pop(q);

}

char value;

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

{

cout << "Введите символ: ";

cin >> value;

push(q, value);

}

int size = q.size;

for (int i = n; i <= size - k; i++)

{

push(q, q.head->data);

pop(q);

}

}

void del\_q(Queue& q)

{

while (q.head->next != nullptr)

pop(q);

Node\* temp = q.head;

q.head = nullptr;

q.size--;

delete temp;

}

};

void print(Queue& q)

{

Node\* temp = q.head;

cout << "Очередь: ";

while (temp != nullptr)

{

cout << temp->data << ' ';

temp = temp->next;

}

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Queue qu;

int n, k;

char dat;

cout << "Введите количество симолов: ";

cin >> n;

cout << "Вводите эти симовлы (через enter)" << endl;

cin >> dat;

qu.initQ(qu, dat);

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

{

cin >> dat;

qu.push(qu, dat);

}

print(qu);

cout << "Введите ключ, перед которым и после которого нужно удалить символы: ";

cin >> k;

qu.remove(qu, k);

qu.remove(qu, k - 2);

print(qu);

cout << "По сколько элементов добавить в начало и в конец списка? ";

cin >> k;

qu.insert(qu, k, 1);

qu.insert(qu, k, n + k - 1);

print(qu);

ofstream file1("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_очереди\\queue.txt");

if (!file1.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

Node\* node = qu.head;

while (node != nullptr)

{

file1 << node->data << '\n';

node = node->next;

}

file1.close();

system("pause");

qu.del\_q(qu);

print(qu);

ifstream file2("C:\\Users\\user\\Desktop\\C++\\Лаба\_11\_очереди\\queue.txt");

if (!file2.is\_open())

{

cout << "Не удалось открыть файл!" << endl;

return 1;

}

char s;

file2 >> s;

qu.initQ(qu, s);

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

{

file2 >> s;

qu.push(qu, s);

}

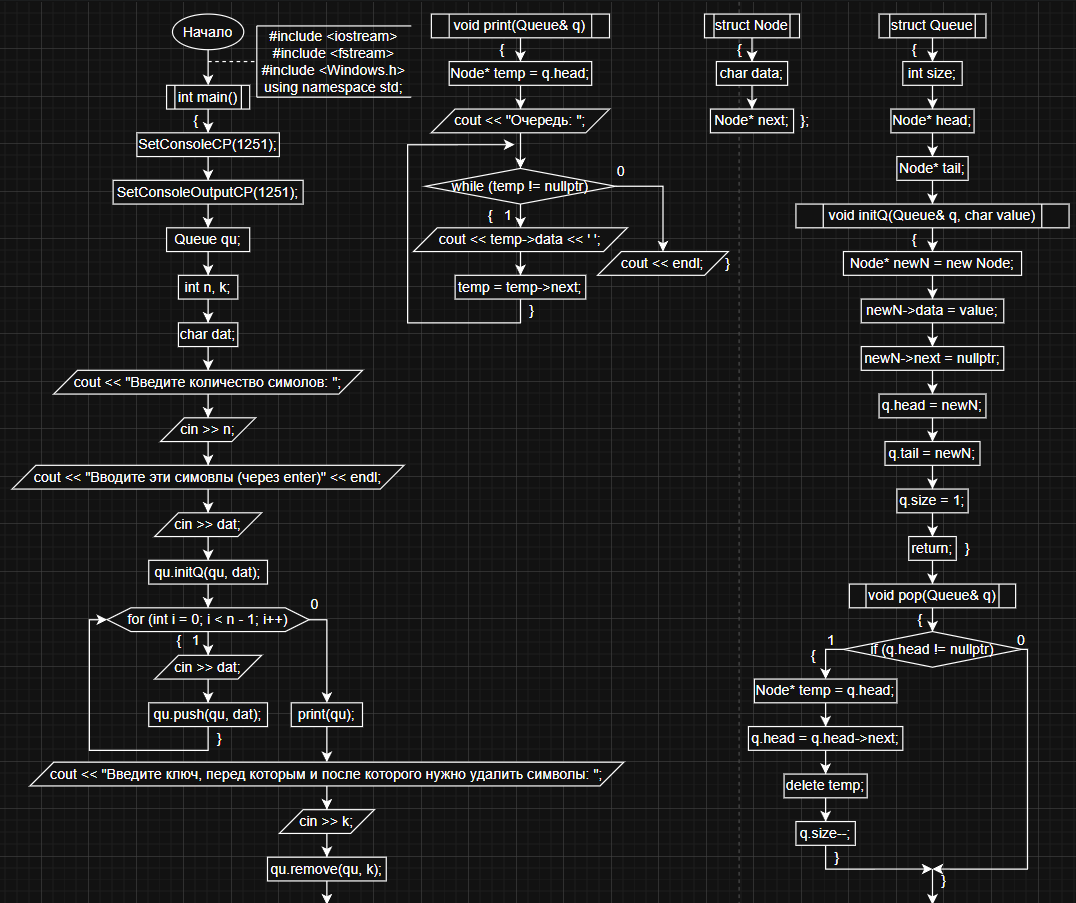
file2.close();

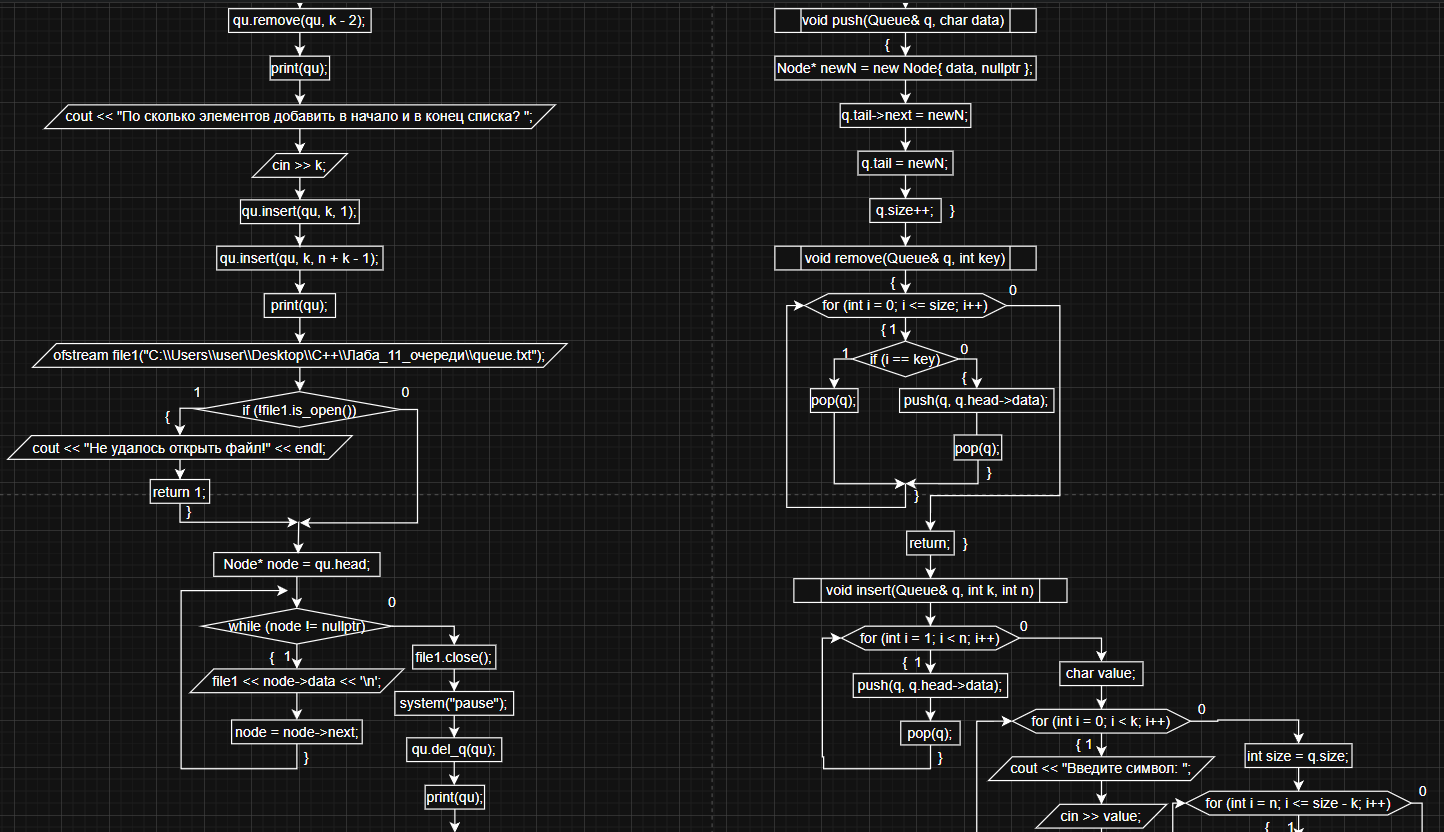
print(qu);

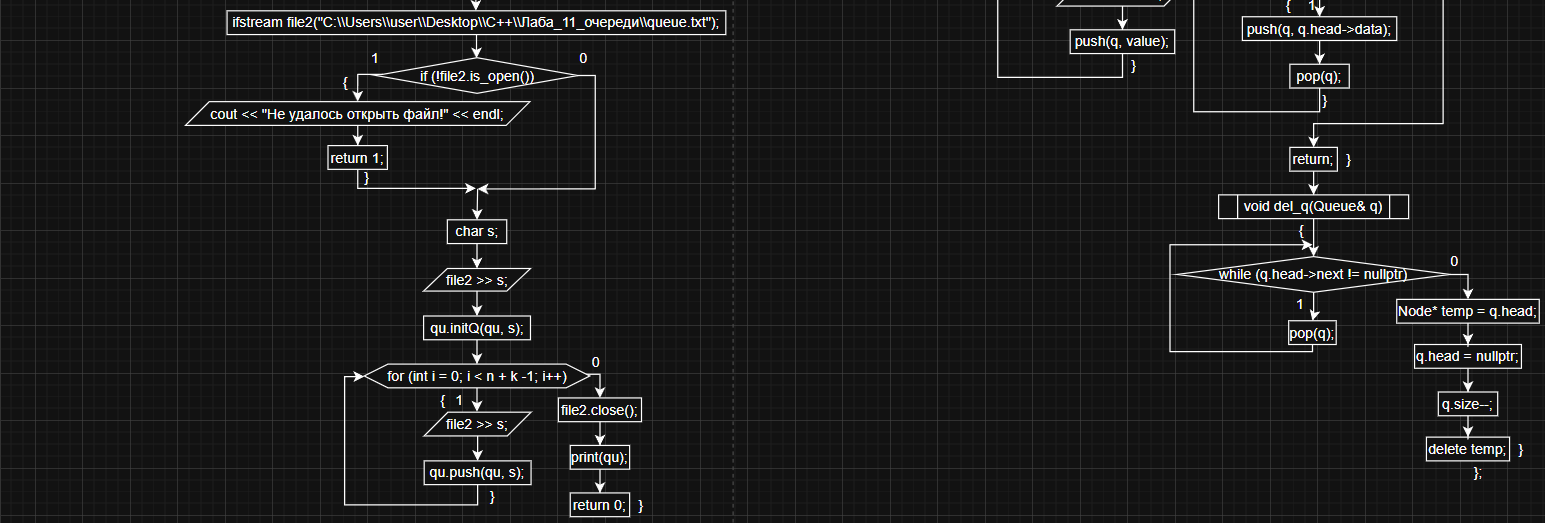
return 0;

}

Блок-схема







Результат работы программы:

