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 №11.1**

**"Информационные динамические структуры"**

Выполнил студент гр. РИС-24-1б

Иванов Семен Сергеевич

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

**Постановка задачи**

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

Для каждого варианта 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.

Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Удалить из него К элементов перед элементом с заданным номером, добавить К элементов в конец списка.

**Анализ задачи**

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.

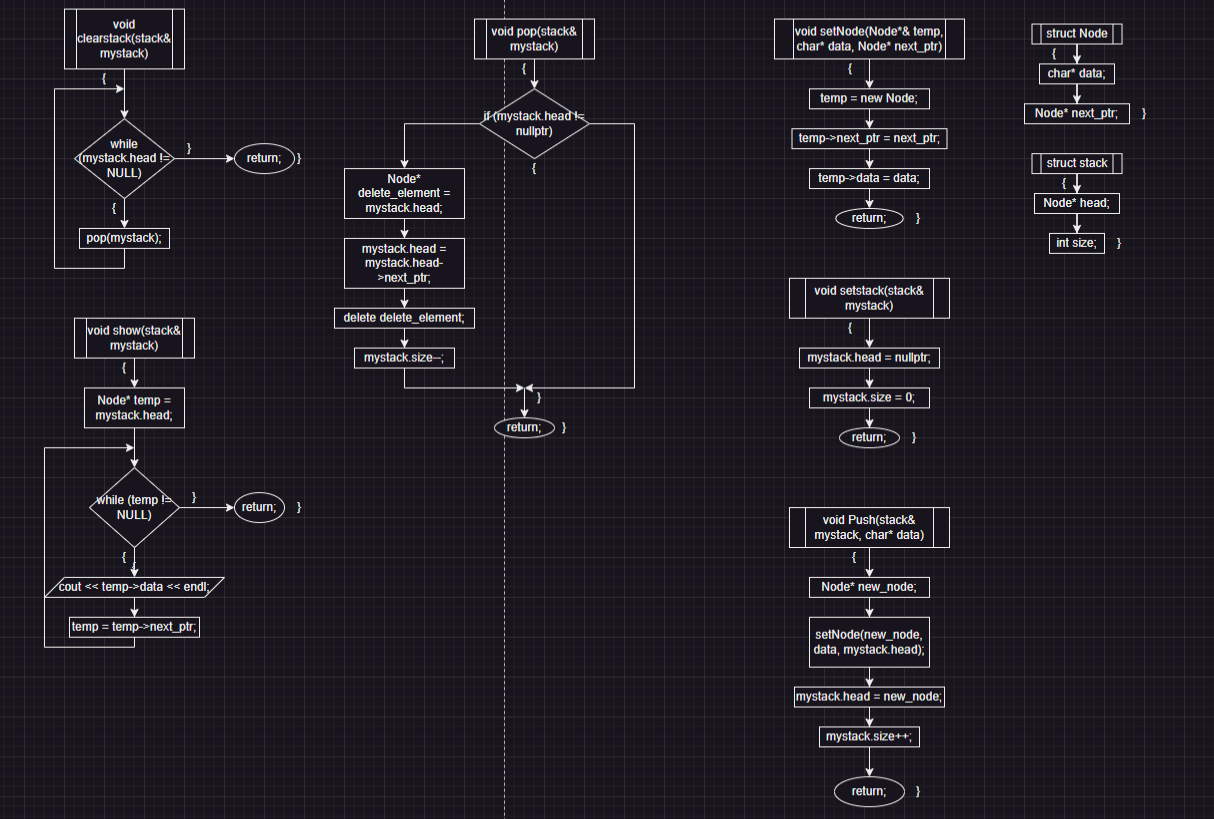
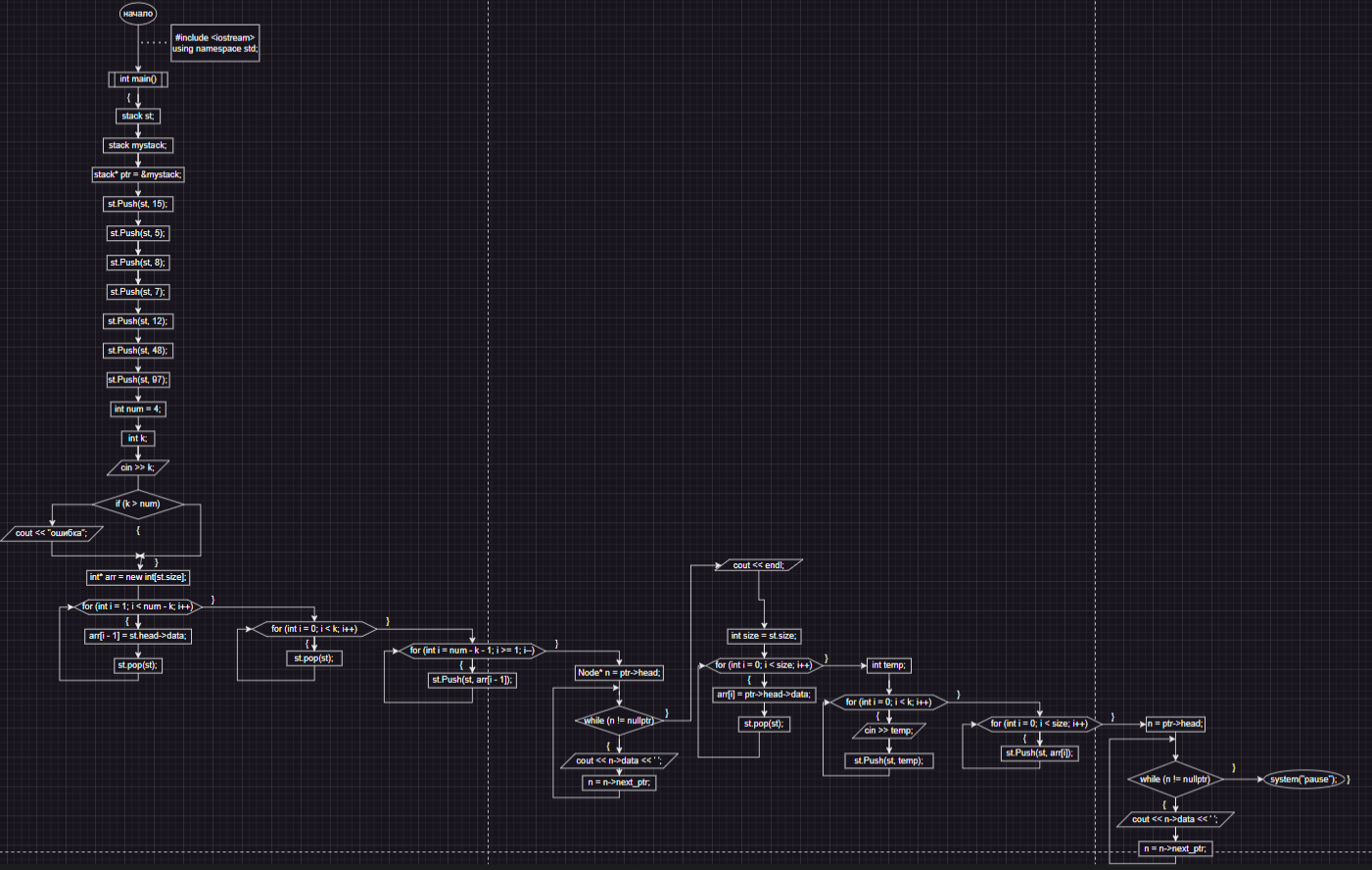
7. Записать список в файл, уничтожить его и выполнить печать (при печати должно быть выдано сообщение "Список пустой").

8. Написать функцию для восстановления списка из файла.

9. Восстановить список и распечатать его.

10. Уничтожить список.

**Блок-схема**

**** ****

**Код**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next\_ptr;

};

struct stack {

Node\* head;

int size;

void setNode(Node\*& temp, int data, Node\* next\_ptr) {

temp = new Node;

temp->next\_ptr = next\_ptr;

temp->data = data;

return;

}

void setstack(stack& mystack) {

mystack.head = nullptr;

mystack.size = 0;

return;

}

void Push(stack& mystack, int data) {

Node\* new\_node;

setNode(new\_node, data, mystack.head);

mystack.head = new\_node;

mystack.size++;

return;

}

void pop(stack& mystack) {

if (mystack.head != nullptr) {

Node\* delete\_element = mystack.head;

mystack.head = mystack.head->next\_ptr;

delete delete\_element;

mystack.size--;

}

return;

}

void clearstack(stack& mystack) {

while (mystack.head != NULL) {

pop(mystack);

}

}

void show(stack& mystack) {

Node\* temp = mystack.head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next\_ptr;

}

return;

}

};

int main() {

stack st;

st.setstack(st);

stack\* ptr = &st;

st.Push(st, 15);

st.Push(st, 5);

st.Push(st, 8);

st.Push(st, 7);

st.Push(st, 12);

st.Push(st, 48);

st.Push(st, 97);

int num = 4;

int k;

cin >> k;

if (k > num) cout << "ошибка";

int\* arr = new int[st.size];

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

arr[i - 1] = st.head->data;

st.pop(st);

}

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

st.pop(st);

}

for (int i = num - k - 1; i >= 1; i--) {

st.Push(st, arr[i - 1]);

}

Node\* n = ptr->head;

while (n != nullptr) {

cout << n->data << ' ';

n = n->next\_ptr;

}

cout << endl;

int size = st.size;

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

arr[i] = ptr->head->data;

st.pop(st);

}

int temp;

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

cin >> temp;

st.Push(st, temp);

}

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

st.Push(st, arr[i]);

}

n = ptr->head;

while (n != nullptr) {

cout << n->data << ' ';

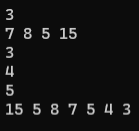
n = n->next\_ptr;

}

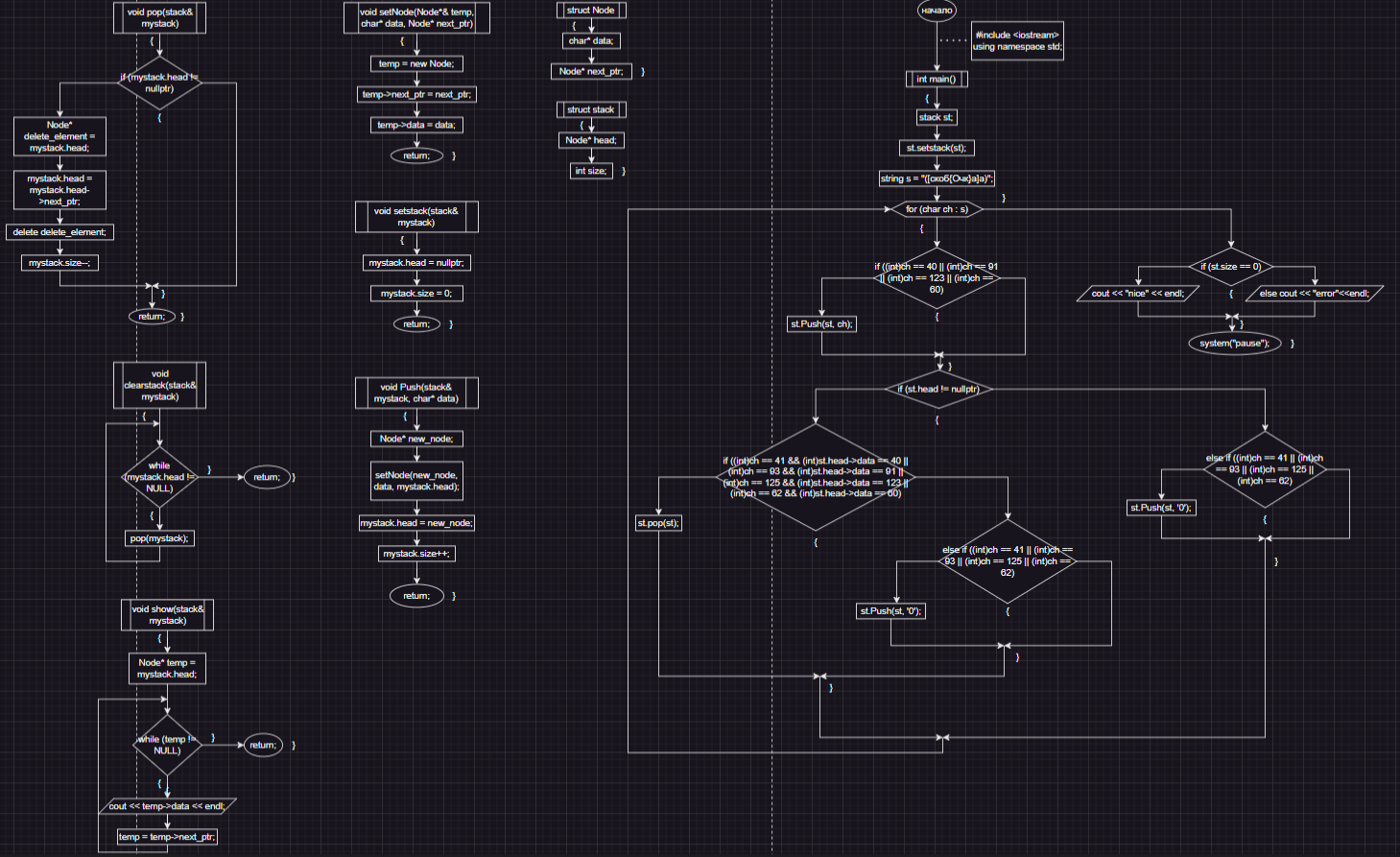
system("pause");

}

**Скриншот решения и github**

****

**Блок-схема(Проверка скобочек)**

****

**Код(проверка скобочек)**

#include <iostream>

using namespace std;

struct Node {

char data;

Node\* next\_ptr;

};

struct stack {

Node\* head;

int size;

void setNode(Node\*& temp, char data, Node\* next\_ptr) {

temp = new Node;

temp->next\_ptr = next\_ptr;

temp->data = data;

return;

}

void setstack(stack& mystack) {

mystack.head = nullptr;

mystack.size = 0;

return;

}

void Push(stack& mystack, char data) {

Node\* new\_node;

setNode(new\_node, data, mystack.head);

mystack.head = new\_node;

mystack.size++;

return;

}

void pop(stack& mystack) {

if (mystack.head != nullptr) {

Node\* delete\_element = mystack.head;

mystack.head = mystack.head->next\_ptr;

delete delete\_element;

mystack.size--;

}

return;

}

void clearstack(stack& mystack) {

while (mystack.head != NULL) {

pop(mystack);

}

}

void show(stack& mystack) {

Node\* temp = mystack.head;

while (temp != NULL) {

cout << temp->data << endl;

temp = temp->next\_ptr;

}

return;

}

};

int main() {

stack st;

stack\* ptr = &st;

st.setstack(st);

string s = "([скоб{Очк}а]а)";

for (char ch : s) {

if ((int)ch == 40 || (int)ch == 91 || (int)ch == 123 || (int)ch == 60) {

st.Push(st, ch);

}

if (st.head != nullptr) {

if ((int)ch == 41 && (int)st.head->data == 40 || (int)ch == 93 && (int)st.head->data == 91 || (int)ch == 125 && (int)st.head->data == 123 || (int)ch == 62 && (int)st.head->data == 60) {

st.pop(st);

}

else if ((int)ch == 41 || (int)ch == 93 || (int)ch == 125 || (int)ch == 62) {

st.Push(st, '0');

}

}

else if ((int)ch == 41 || (int)ch == 93 || (int)ch == 125 || (int)ch == 62) {

st.Push(st, '0');

}

}

if (st.size == 0) {

cout << "nice" << endl;

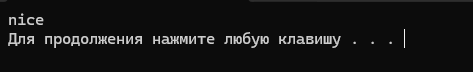
}

else cout << "error" << endl;

system("pause");

}

**Скриншот решения и github**

****