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.**

**“Функции и массивы”  
Вариант 5**

Выполнил студент гр. РИС-24-2б

Трофимов Степан Степанович

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

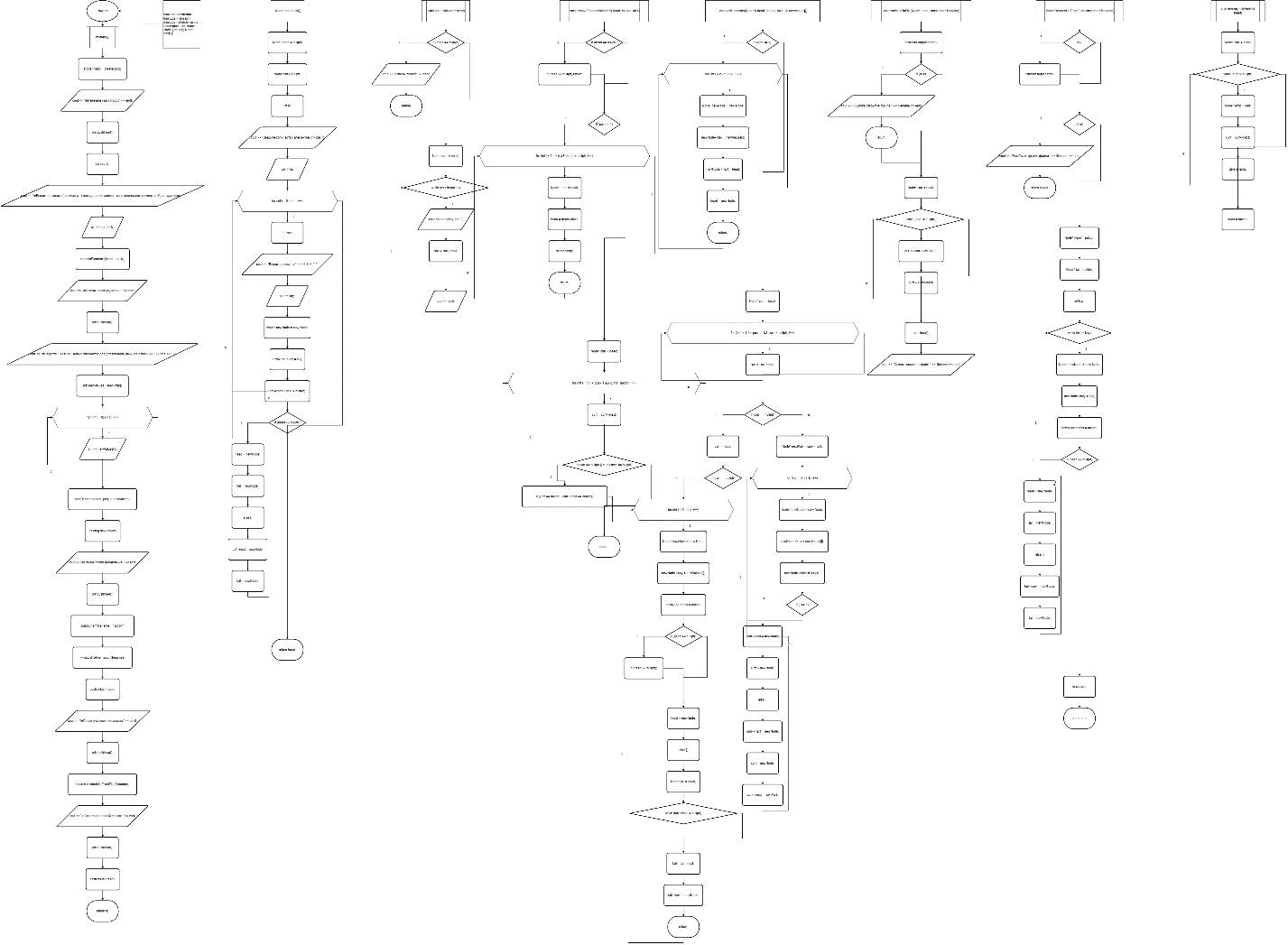
1. **Постановка задачи:**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом. Для каждого вариант 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 однонаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить К элементов, начиная с заданного номера;

**Односвязные списки  
Блок-схема**

****

**Код**

#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

struct Node {

int key;

Node\* next;

};

Node\* createList() {

Node\* head = nullptr;

Node\* tail = nullptr;

int n;

cout << "Введите количество элементов списка: ";

cin >> n;

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

int key;

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

cin >> key;

Node\* newNode = new Node;

newNode->key = key;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

}

return head;

}

void printList(Node\* head) {

if (head == nullptr) {

cout << "Список пустой" << endl;

return;

}

Node\* curr = head;

while (curr != nullptr) {

cout << curr->key << " ";

curr = curr->next;

}

cout << endl;

}

void removeElements(Node\*& head, int pos, int k) {

if (head == nullptr) return;

if (pos <= 1) {

for (int i = 0; i < k && head != nullptr; i++) {

Node\* temp = head;

head = head->next;

delete temp;

}

return;

}

Node\* curr = head;

for (int i = 1; i < pos - 1 && curr != nullptr; i++) {

curr = curr->next;

}

if (curr == nullptr || curr->next == nullptr)

return;

Node\* temp = curr->next;

for (int i = 0; i < k && temp != nullptr; i++) {

Node\* toDelete = temp;

temp = temp->next;

delete toDelete;

}

curr->next = temp;

}

void addElements(Node\*& head, int pos, int k, int newValues[]) {

if (pos <= 1) {

for (int i = k - 1; i >= 0; i--) {

Node\* newNode = new Node;

newNode->key = newValues[i];

newNode->next = head;

head = newNode;

}

return;

}

Node\* curr = head;

for (int i = 1; i < pos - 1 && curr != nullptr; i++) {

curr = curr->next;

}

if (curr == nullptr) {

curr = head;

if (curr == nullptr) {

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

Node\* newNode = new Node;

newNode->key = newValues[i];

newNode->next = nullptr;

if (head == nullptr)

head = newNode;

else {

Node\* tail = head;

while (tail->next != nullptr)

tail = tail->next;

tail->next = newNode;

}

}

return;

}

else {

while (curr->next != nullptr)

curr = curr->next;

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

Node\* newNode = new Node;

newNode->key = newValues[i];

newNode->next = nullptr;

curr->next = newNode;

curr = newNode;

}

return;

}

}

else {

Node\* nextPart = curr->next;

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

Node\* newNode = new Node;

newNode->key = newValues[i];

newNode->next = nullptr;

if (i == 0) {

curr->next = newNode;

curr = newNode;

}

else {

curr->next = newNode;

curr = newNode;

}

}

curr->next = nextPart;

}

}

void writeListToFile(Node\* head, const char\* filename) {

ofstream out(filename);

if (!out) {

cout << "Ошибка открытия файла " << filename << endl;

return;

}

Node\* curr = head;

while (curr != nullptr) {

out << curr->key << " ";

curr = curr->next;

}

out.close();

cout << "Список записан в файл " << filename << endl;

}

Node\* restoreListFromFile(const char\* filename) {

ifstream in(filename);

if (!in) {

cout << "Ошибка открытия файла " << filename << endl;

return nullptr;

}

Node\* head = nullptr;

Node\* tail = nullptr;

int key;

while (in >> key) {

Node\* newNode = new Node;

newNode->key = key;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

}

in.close();

return head;

}

void destroyList(Node\*& head) {

Node\* curr = head;

while (curr != nullptr) {

Node\* temp = curr;

curr = curr->next;

delete temp;

}

head = nullptr;

}

int main() {

Node\* head = createList();

cout << "\nИзначальный список:" << endl;

printList(head);

int pos, k;

cout << "\nВведите позицию (начиная с 1) для удаления элементов и количество элементов K для удаления: ";

cin >> pos >> k;

removeElements(head, pos, k);

cout << "\nСписок после удаления:" << endl;

printList(head);

cout << "\nВведите " << k << " новых элементов для добавления, начиная с позиции " << pos << ": ";

int\* newValues = new int[k];

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

cin >> newValues[i];

}

addElements(head, pos, k, newValues);

delete[] newValues;

cout << "\nСписок после добавления:" << endl;

printList(head);

const char\* filename = "list.txt";

writeListToFile(head, filename);

destroyList(head);

cout << "\nПосле уничтожения списка:" << endl;

printList(head);

head = restoreListFromFile(filename);

cout << "\nВосстановленный список:" << endl;

printList(head);

destroyList(head);

return 0;

}