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 10**

Выполнил студент гр. РИС-24-2б

Бартов Игорь Сергеевич

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

1. **Постановка задачи:**

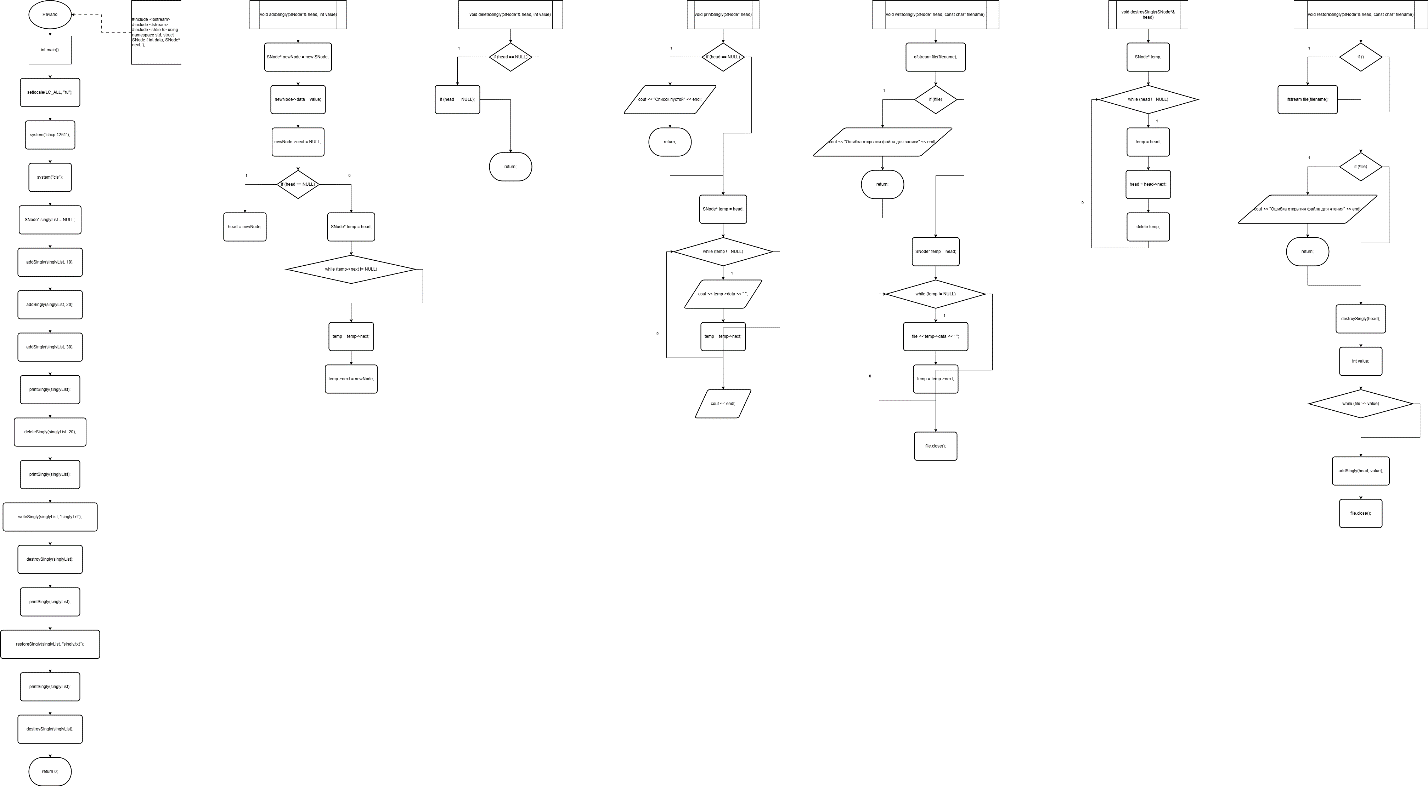
Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом. Для каждого вариант разработать следующие функции:

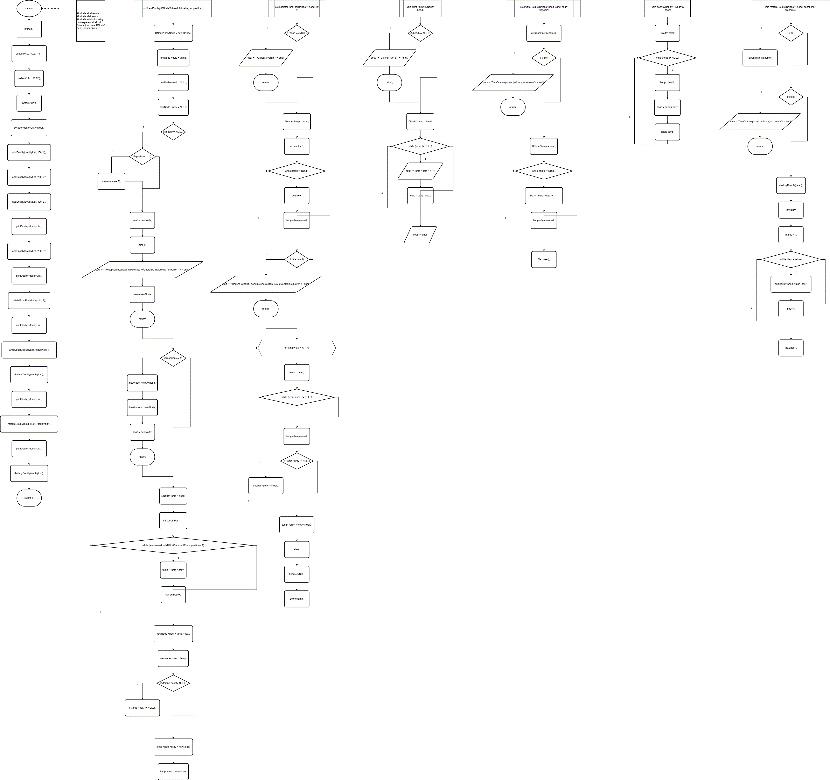
1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

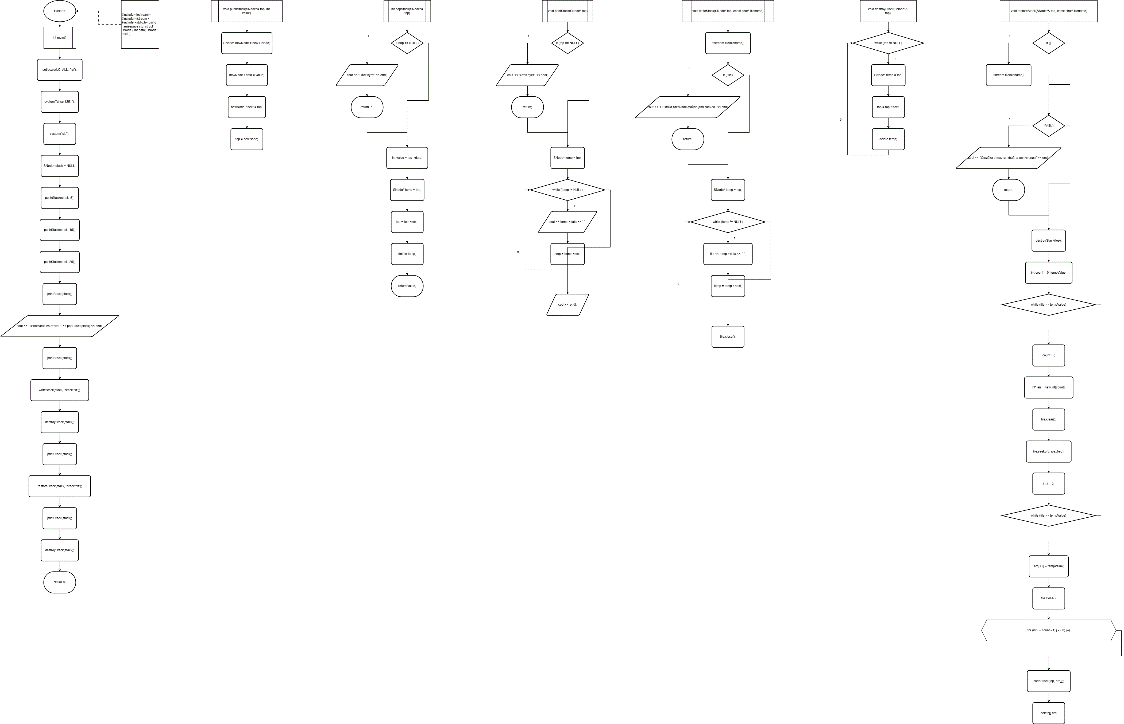
Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Добавить в него элемент с заданным номером, удалить К элементов из конца списка.

**2. Блок-схема:**

Односвязный список:

  
Двусвязный список:

  
Стек:



1. **Код:**

Односвязный список:

#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

struct SNode {

int data;

SNode\* next;

};

void addSingly(SNode\*& head, int value) {

SNode\* newNode = new SNode;

newNode->data = value;

newNode->next = NULL;

if (head == NULL) {

head = newNode;

}

else {

SNode\* temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

}

}

void deleteSingly(SNode\*& head, int value) {

if (head == NULL)

return;

SNode\* temp = head, \* prev = NULL;

while (temp != NULL && temp->data != value) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

cout << "Элемент не найден" << endl;

return;

}

if (prev == NULL)

head = head->next;

else

prev->next = temp->next;

delete temp;

}

void printSingly(SNode\* head) {

if (head == NULL) {

cout << "Список пустой" << endl;

return;

}

SNode\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

void writeSingly(SNode\* head, const char\* filename) {

ofstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для записи" << endl;

return;

}

SNode\* temp = head;

while (temp != NULL) {

file << temp->data << " ";

temp = temp->next;

}

file.close();

}

void destroySingly(SNode\*& head) {

SNode\* temp;

while (head != NULL) {

temp = head;

head = head->next;

delete temp;

}

}

void restoreSingly(SNode\*& head, const char\* filename) {

ifstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для чтения" << endl;

return;

}

destroySingly(head);

int value;

while (file >> value)

addSingly(head, value);

file.close();

}

int main() {

setlocale(LC\_ALL, "ru");

system("chcp 1251");

system("cls");

SNode\* singlyList = NULL;

addSingly(singlyList, 10);

addSingly(singlyList, 20);

addSingly(singlyList, 30);

printSingly(singlyList);

deleteSingly(singlyList, 20);

printSingly(singlyList);

writeSingly(singlyList, "singly.txt");

destroySingly(singlyList);

printSingly(singlyList);

restoreSingly(singlyList, "singly.txt");

printSingly(singlyList);

destroySingly(singlyList);

return 0;

}

Двусвязный список:

#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

struct DNode {

int data;

DNode\* next;

DNode\* prev;

};

void addDoubly(DNode\*& head, int value, int position) {

DNode\* newNode = new DNode;

newNode->data = value;

newNode->next = NULL;

newNode->prev = NULL;

if (head == NULL) {

if (position == 1)

head = newNode;

else {

cout << "Неверная позиция, список пустой, вставка только на позицию 1" << endl;

delete newNode;

}

return;

}

if (position == 1) {

newNode->next = head;

head->prev = newNode;

head = newNode;

return;

}

DNode\* temp = head;

int currentPos = 1;

while (temp->next != NULL && currentPos < position - 1) {

temp = temp->next;

currentPos++;

}

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != NULL)

temp->next->prev = newNode;

temp->next = newNode;

}

void deleteFromEnd(DNode\*& head, int K) {

if (head == NULL) {

cout << "Список пустой" << endl;

return;

}

DNode\* temp = head;

int count = 0;

while (temp != NULL) {

count++;

temp = temp->next;

}

if (K > count) {

cout << "Нельзя удалить больше элементов, чем имеется в списке" << endl;

return;

}

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

temp = head;

while (temp->next != NULL)

temp = temp->next;

if (temp->prev != NULL)

temp->prev->next = NULL;

else

head = NULL;

delete temp;

}

}

void printDoubly(DNode\* head) {

if (head == NULL) {

cout << "Список пустой" << endl;

return;

}

DNode\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

void writeDoubly(DNode\* head, const char\* filename) {

ofstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для записи" << endl;

return;

}

DNode\* temp = head;

while (temp != NULL) {

file << temp->data << " ";

temp = temp->next;

}

file.close();

}

void destroyDoubly(DNode\*& head) {

DNode\* temp;

while (head != NULL) {

temp = head;

head = head->next;

delete temp;

}

}

void restoreDoubly(DNode\*& head, const char\* filename) {

ifstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для чтения" << endl;

return;

}

destroyDoubly(head);

int value;

int pos = 1;

while (file >> value) {

addDoubly(head, value, pos);

pos++;

}

file.close();

}

int main() {

setlocale(LC\_ALL, "ru");

system("chcp 1251");

system("cls");

DNode\* doublyList = NULL;

addDoubly(doublyList, 100, 1);

addDoubly(doublyList, 200, 2);

addDoubly(doublyList, 300, 3);

printDoubly(doublyList);

addDoubly(doublyList, 250, 3);

printDoubly(doublyList);

deleteFromEnd(doublyList, 2);

printDoubly(doublyList);

writeDoubly(doublyList, "doubly.txt");

destroyDoubly(doublyList);

printDoubly(doublyList);

restoreDoubly(doublyList, "doubly.txt");

printDoubly(doublyList);

destroyDoubly(doublyList);

return 0;

}

Стек:

#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

struct SNode {

int data;

SNode\* next;

};

void pushStack(SNode\*& top, int value) {

SNode\* newNode = new SNode;

newNode->data = value;

newNode->next = top;

top = newNode;

}

int popStack(SNode\*& top) {

if (top == NULL) {

cout << "Стек пуст" << endl;

return -1;

}

int value = top->data;

SNode\* temp = top;

top = top->next;

delete temp;

return value;

}

void printStack(SNode\* top) {

if (top == NULL) {

cout << "Стек пуст" << endl;

return;

}

SNode\* temp = top;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

void writeStack(SNode\* top, const char\* filename) {

ofstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для записи" << endl;

return;

}

SNode\* temp = top;

while (temp != NULL) {

file << temp->data << " ";

temp = temp->next;

}

file.close();

}

void destroyStack(SNode\*& top) {

while (top != NULL) {

SNode\* temp = top;

top = top->next;

delete temp;

}

}

void restoreStack(SNode\*& top, const char\* filename) {

ifstream file(filename);

if (!file) {

cout << "Ошибка открытия файла для чтения" << endl;

return;

}

destroyStack(top);

int count = 0, tempValue;

while (file >> tempValue)

count++;

int\* arr = new int[count];

file.clear();

file.seekg(0, ios::beg);

int i = 0;

while (file >> tempValue)

arr[i++] = tempValue;

file.close();

for (int j = count - 1; j >= 0; j--)

pushStack(top, arr[j]);

delete[] arr;

}

int main() {

setlocale(LC\_ALL, "ru");

system("chcp 1251");

system("cls");

SNode\* stack = NULL;

pushStack(stack, 5);

pushStack(stack, 15);

pushStack(stack, 25);

printStack(stack);

cout << "Извлечено из стека: " << popStack(stack) << endl;

printStack(stack);

writeStack(stack, "stack.txt");

destroyStack(stack);

printStack(stack);

restoreStack(stack, "stack.txt");

printStack(stack);

destroyStack(stack);

return 0;

}

Очередь:

#include <iostream>

#include <fstream>

#include <cstdlib> // для system()

#include <clocale> // для setlocale()

using namespace std;

// Определение структуры узла двунаправленного списка

struct DNode {

int data;

DNode\* next;

DNode\* prev;

};

// Функция добавления элемента в конец двунаправленного списка

void addDNode(DNode\*& head, DNode\*& tail, int value) {

DNode\* newNode = new DNode;

newNode->data = value;

newNode->next = nullptr;

newNode->prev = tail; // предыдущий элемент – текущий tail

if(tail != nullptr) {

tail->next = newNode;

} else {

// если список пуст, новый узел становится и head

head = newNode;

}

tail = newNode;

}

// Функция вставки элемента в список по заданной позиции (номер начинается с 1)

// Если pos <= 1 или список пуст – вставка в начало,

// Если позиция больше размера списка – вставка в конец.

void insertAtPosition(DNode\*& head, DNode\*& tail, int pos, int value) {

DNode\* newNode = new DNode;

newNode->data = value;

newNode->next = nullptr;

newNode->prev = nullptr;

if(pos <= 1 || head == nullptr) { // вставка в начало

newNode->next = head;

if(head != nullptr)

head->prev = newNode;

else

tail = newNode;

head = newNode;

return;

}

DNode\* current = head;

int currentPos = 1;

// Идём до узла, после которого нужно вставить новый

while(current != nullptr && currentPos < pos - 1) {

current = current->next;

currentPos++;

}

if(current == nullptr) { // если позиция больше длины списка – добавляем в конец

addDNode(head, tail, value);

return;

}

newNode->next = current->next;

newNode->prev = current;

if(current->next != nullptr)

current->next->prev = newNode;

else

tail = newNode; // если вставляем в конец, обновляем tail

current->next = newNode;

}

// Функция удаления первого найденного элемента по значению

void deleteDNode(DNode\*& head, DNode\*& tail, int value) {

DNode\* current = head;

while(current != nullptr && current->data != value)

current = current->next;

if(current == nullptr) {

cout << "Элемент не найден" << endl;

return;

}

// Перенастройка ссылок соседних узлов

if(current->prev != nullptr)

current->prev->next = current->next;

else

head = current->next;

if(current->next != nullptr)

current->next->prev = current->prev;

else

tail = current->prev;

delete current;

}

// Функция печати двунаправленного списка

void printDList(DNode\* head) {

if(head == nullptr) {

cout << "Список пустой" << endl;

return;

}

DNode\* current = head;

while(current != nullptr) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

// Функция записи списка в файл (данные записываются через пробел)

void writeDList(DNode\* head, const char\* filename) {

ofstream file(filename);

if(!file) {

cout << "Ошибка открытия файла для записи" << endl;

return;

}

DNode\* current = head;

while(current != nullptr) {

file << current->data << " ";

current = current->next;

}

file.close();

}

// Функция уничтожения (освобождения памяти) списка

void destroyDList(DNode\*& head, DNode\*& tail) {

DNode\* current;

while(head != nullptr) {

current = head;

head = head->next;

delete current;

}

tail = nullptr;

}

// Функция восстановления списка из файла

void restoreDList(DNode\*& head, DNode\*& tail, const char\* filename) {

ifstream file(filename);

if(!file) {

cout << "Ошибка открытия файла для чтения" << endl;

return;

}

// Сначала уничтожаем текущий список

destroyDList(head, tail);

int value;

while(file >> value) {

addDNode(head, tail, value);

}

file.close();

}

// Функция удаления K элементов с конца списка

void deleteKFromEnd(DNode\*& head, DNode\*& tail, int k) {

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

if(tail == nullptr) break;

DNode\* temp = tail;

tail = tail->prev;

if(tail != nullptr)

tail->next = nullptr;

else

head = nullptr; // если список стал пустым

delete temp;

}

}

int main() {

// Установка русской локали и очистка экрана (для Windows)

setlocale(LC\_ALL, "ru");

system("chcp 1251");

system("cls");

DNode\* head = nullptr;

DNode\* tail = nullptr;

// Создаём список и добавляем элементы

addDNode(head, tail, 10);

addDNode(head, tail, 20);

addDNode(head, tail, 30);

cout << "Начальный список: ";

printDList(head);

// Вставка элемента со значением 15 на позицию 2

insertAtPosition(head, tail, 2, 15);

cout << "После вставки элемента 15 на позицию 2: ";

printDList(head);

// Удаление элемента по значению (удаляем 20)

deleteDNode(head, tail, 20);

cout << "После удаления элемента со значением 20: ";

printDList(head);

// Запись списка в файл "doubly.txt"

writeDList(head, "doubly.txt");

// Удаление 2 элементов с конца списка

deleteKFromEnd(head, tail, 2);

cout << "После удаления 2 элементов с конца: ";

printDList(head);

// Восстановление списка из файла

restoreDList(head, tail, "doubly.txt");

cout << "После восстановления из файла: ";

printDList(head);

// Уничтожение списка

destroyDList(head, tail);

return 0;

}

