**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: **Многофайловая программа и интерактивное меню.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4372 |  | Барабанов И. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Разработка программы для объединения в едином пользовательском интерфейсе ранее разработанных практических работ.

**Основные теоретические положения.**

**1. Многофайловые програмы**

Многофайловые программы в C++ представляют собой структуру, в которой код разбивается на несколько файлов. Это позволяет организовать код более эффективно, улучшает читаемость и упрощает его поддержку.

Основные компоненты многофайловой программы включают:

* Исходные файлы (.cpp), которые содержат реализацию функций и методов;
* Заголовочные файлы (.h или .hpp), которые содержат объявления функций, классов, структур и других элементов, которые могут использоваться в нескольких исходных файлах.

**2. Заголовочные файлы**

Заголовочные файлы играют ключевую роль в многофайловых программах. Они позволяют:

* Объявлять интерфейсы, так как заголовочные файлы содержат объявления функций и классов, что позволяет другим файлам использовать эти функции и классы без необходимости знать их реализацию.
* Избежать дублирования кода, потому что использование заголовочных файлов позволяет избежать повторного объявления одних и тех же функций и классов в разных исходных файлах.
* Упрощать управление зависимостями: заголовочные файлы могут включать другие заголовочные файлы, что позволяет организовать зависимости между различными частями программы.

**3. Директивы препроцессора**

Директивы препроцессора — это команды, которые обрабатываются до компиляции кода. Они начинаются с символа # и выполняют различные функции:

* #include: используется для включения содержимого одного файла в другой. Например:

#include <iostream> // Включает стандартную библиотеку ввода-вывода

#include "myheader.h" // Включает пользовательский заголовочный файл

* #define: используется для определения макросов, которые могут быть заменены на заданные значения или выражения. Например:

#define PI 3.14159

* #ifndef, #define, #endif: используются для предотвращения множественного включения одного и того же заголовочного файла. Это называется "защита от повторного включения". Пример:

#ifndef MYHEADER\_H

#define MYHEADER\_H

*// Содержимое заголовочного файла*

#endif *// MYHEADER\_H*

* #ifdef, #else, #endif: используются для условной компиляции кода. Это позволяет включать или исключать части кода в зависимости от того, определены ли определенные макросы. Например:

#ifdef DEBUG

std::cout << "Debug mode" << std::endl;

#endif

**Постановка задачи.**

Необходимо написать программу, которая содержит в себе инфраструктуру переключения между практическими работами (интерактивное меню). В меню должна быть реализована возможность многократно запускать разные лабораторные работы.

**Выполнение работы.**

Код программы представлен в приложении А.

| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| --- | --- |
| Меню | |
| При запуске программы пользователь видит меню. | Программа предоставляет пользователю выбор между запуском конкретной практической работы или выходом из программы: |
| Запуск практической работы | |
| Пользователь вводит номер практической работы для запуска. | Программа запрашивает ввод пользователя:    После ввода запускается соответствующая практическая работа:    После завершения выполненя практической работы интерфейс программы вновь предлагает пользовалю выбрать один из пунктов меню. |
| Выход из программы | |
| Пользователь вводит цифру «5» для завершения работы програмы. | Программа запрашивает ввод пользователя:    Интерфейс уведомляет пользователя о завершении работы программы: |

**Выводы.**

В ходе выполнения курсовой работы была изучена структура многофайловой программы на C++, что позволило понять принципы организации кода и использования заголовочных файлов для объявления функций из разных модулей. Также было реализовано интерактивное меню, позволяющее пользователю выбирать различные функции программы.

Приложение А

рабочий код

**Файл main.cpp**

#include <iostream>  
#include "lab1.h"  
#include "lab2.h"  
#include "lab3.h"  
#include "lab4.h"  
  
int main() {  
 int choice;  
  
 do {  
 system("cls");  
 std::cout << "Меню:" << std::endl;  
 std::cout << "1. Лабораторная работа #1" << std::endl;  
 std::cout << "2. Лабораторная работа #2" << std::endl;  
 std::cout << "3. Лабораторная работа #3" << std::endl;  
 std::cout << "4. Лабораторная работа #4" << std::endl;  
 std::cout << "5. Выход" << std::endl;  
  
 do {  
 std::cout << "Выберите пункт меню (1-5): ";  
 std::cin >> choice;  
 switch (choice) {  
 case 1:  
 system("cls");  
 run\_lab\_1();  
 break;  
 case 2:  
 system("cls");  
 run\_lab\_2();  
 break;  
 case 3:  
 system("cls");  
 run\_lab\_3();  
 break;  
 case 4:  
 system("cls");  
 run\_lab\_4();  
 break;  
 case 5:  
 std::cout << "Выход из программы." << std::endl;  
 break;  
 default:  
 std::cout << "Неверный выбор. Пожалуйста, попробуйте снова." << std::endl;  
 }  
 } while (choice > 5);  
  
 std::cout << std::endl;  
  
 } while (choice != 5);  
  
 return 0;  
}

**Файл lab1.cpp**

#include <iostream>  
  
int run\_lab\_1() {  
 std::cout << "Размеры различных типов данных: " << std::endl;  
 std::cout << "int: " << sizeof(int) << std::endl  
 << "short int: " << sizeof(short int) << std::endl  
 << "long int: " << sizeof(long int) << std::endl  
 << "float: " << sizeof(float) << std::endl  
 << "double: " << sizeof(double) << std::endl  
 << "long double: " << sizeof(long double) << std::endl  
 << "char: " << sizeof(char) << std::endl  
 << "bool: " << sizeof(bool) << std::endl;  
  
 std::cout << "Введите целое число: ";  
 int integer;  
 std::cin >> integer;  
 std::cout << integer << ": ";  
 unsigned int mask = 1U << 31;  
 for (int i = 0; i < 32; i++) {  
 std::cout << ((integer & mask) ? "1" : "0");  
 if (i == 0 || (i + 1) % 4 == 0) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 }  
 std::cout << std::endl;  
 mask = 1U << 31;  
  
 union {  
 float one;  
 unsigned int two;  
 } hack;  
  
 std::cout << "Введите вещественное число: ";  
 std::cin >> hack.one;  
 std::cout << hack.one << ": ";  
  
 for (int i = 0; i < 32; i++) {  
 std::cout << ((hack.two & mask) ? "1" : "0");  
 if (i == 0 || i == 8) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 }  
 mask = 1U << 31;  
 std::cout << std::endl;  
  
 union {  
 double one;  
 unsigned int two[2];  
 } double\_hack;  
  
 std::cout << "Введите вещественное число двойной точности: ";  
 std::cin >> double\_hack.one;  
 std::cout << double\_hack.one << ": ";  
 for (int i = 0; i < 64; i++) {  
 int index = i < 64 / 2;  
 std::cout << ((double\_hack.two[index] & mask) ? "1" : "0");  
 if (i == 0 || i == 11) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 if (mask == 0) {  
 mask = 1U << 31;  
 }  
 }  
 mask = 1U << 31;  
  
 std::cout << std::endl;  
 std::cout << "------" << std::endl;  
 std::cout << "ИДЗ №1: возможность изменять произвольные биты на 1 или 0 по указанию пользователя." << std::endl;  
 std::cout << "------" << std::endl;  
  
 char choice;  
 do {  
 std::cout << "Введите номер бита: ";  
 int number, value;  
 std::cin >> number;  
 while (true) {  
 std::cout << "Введите значение бита (0 или 1): ";  
 std::cin >> value;  
 if (value != 0 && value != 1) {  
 std::cout << "Ошибка: неверное значение бита." << std::endl;  
 continue;  
 }  
 break;  
 }  
  
 if (number >= 64) {  
 std::cout << "Слишком большой номер бита. Нет изменений.";  
 return 0;  
 }  
  
 if (number >= 32) {  
 std::cout << "Номер бита выходит за размер в 4 байта. Изменяется только double." << std::endl;  
 if (value) {  
 double\_hack.two[1] |= 1U << (number % 32);  
 } else {  
 double\_hack.two[1] &= ~(1U << (number % 32));  
 }  
 std::cout << "Новое значение double: " << std::endl;  
  
 std::cout << double\_hack.one << ": ";  
 for (int i = 0; i < 64; i++) {  
 int index = i < 64 / 2;  
 std::cout << ((double\_hack.two[index] & mask) ? "1" : "0");  
 if (i == 0 || i == 11) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 if (mask == 0) {  
 mask = 1U << 31;  
 }  
 }  
  
 return 0;  
 }  
  
  
 std::cout << "Бит №" << number << " устанавливается в значение " << value << " для всех типов данных."  
 << std::endl;  
 if (value) {  
 integer |= 1 << (number % 32);  
 hack.two |= 1U << (number % 32);  
 double\_hack.two[0] |= 1U << (number % 32);  
 } else {  
 integer &= ~(1 << (number % 32));  
 hack.two &= ~(1U << (number % 32));  
 double\_hack.two[0] &= ~(1U << (number % 32));  
 }  
  
 std::cout << std::endl;  
 std::cout << "Новое значение int: " << std::endl;  
 std::cout << integer << ": ";  
 for (int i = 0; i < 32; i++) {  
 std::cout << ((integer & mask) ? "1" : "0");  
 if (i == 0 || (i + 1) % 4 == 0) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 }  
 mask = 1U << 31;  
  
 std::cout << std::endl;  
 std::cout << "Новое значение float: " << std::endl;  
 std::cout << hack.one << ": ";  
 for (int i = 0; i < 32; i++) {  
 std::cout << ((hack.two & mask) ? "1" : "0");  
 if (i == 0 || i == 8) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 }  
 mask = 1U << 31;  
  
 std::cout << std::endl;  
 std::cout << "Новое значение double: " << std::endl;  
 std::cout << double\_hack.one << ": ";  
 for (int i = 0; i < 64; i++) {  
 int index = i < 64 / 2;  
 std::cout << ((double\_hack.two[index] & mask) ? "1" : "0");  
 if (i == 0 || i == 11) {  
 std::cout << " ";  
 }  
 mask >>= 1;  
 if (mask == 0) {  
 mask = 1U << 31;  
 }  
 }  
 mask = 1U << 31;  
  
 std::cout << std::endl;  
 std::cout << "Изменить ещё один бит? (y/n): ";  
 std::cin >> choice;  
  
 } while (choice == 'y');  
 return 0;  
}

**Файл lab1.h**

//  
// Created by Ilia on 12/5/2024.  
//  
  
#ifndef LAB1\_H  
#define LAB1\_H  
  
int run\_lab\_1();  
#endif //LAB1\_H

**Файл lab2.cpp**

#include <iostream>  
#include <chrono>  
#include <ctime>  
  
using namespace std;  
using namespace chrono;  
  
const int N = 1000;  
int unsorted\_array[N], sorted\_array[N];  
time\_point<steady\_clock, duration<\_\_int64, ratio<1, 1000000000>>> start\_time, end\_time;  
nanoseconds result\_time;  
  
void bubble\_sort(int \*arr, int n);  
  
void shaker\_sort(int \*arr, int n);  
  
void comb\_sort(int \*arr, int n);  
  
void insertion\_sort(int \*arr, int n);  
  
void quicksort(int \*arr, int end, int begin);  
  
int binary\_search(int \*arr, int value, int start, int end);  
  
void print\_array(int \*arr, int n);  
  
void copy\_array(const int \*src, int \*dest, int n);  
  
void task\_1();  
  
void task\_2();  
  
void task\_3();  
  
void task\_4();  
  
void task\_5();  
  
void task\_6();  
  
void task\_7();  
  
void task\_8();  
  
void task\_9();  
  
  
int run\_lab\_2() {  
 srand(time(nullptr));  
 task\_1();  
  
 int choice;  
  
 do {  
 cout << "Choose a task to execute (1-9) or 0 to exit: ";  
 cin >> choice;  
 system("cls");  
  
 switch (choice) {  
 case 1:  
 task\_1();  
 break;  
 case 2:  
 task\_2();  
 break;  
 case 3:  
 task\_3();  
 break;  
 case 4:  
 task\_4();  
 break;  
 case 5:  
 task\_5();  
 break;  
 case 6:  
 task\_6();  
 break;  
 case 7:  
 task\_7();  
 break;  
 case 8:  
 task\_8();  
 break;  
 case 9:  
 task\_9();  
 break;  
 case 0:  
 cout << "Exiting the program." << endl;  
 break;  
 default:  
 cout << "Invalid choice. Please select a number from 0 to 8." << endl;  
 break;  
 }  
  
 } while (choice != 0);  
 return 0;  
}  
  
void bubble\_sort(int \*arr, int n) {  
 bool swapped;  
  
 for (int i = 0; i < n - 1; i++) {  
 swapped = false;  
  
 for (int j = 0; j < n - 1 - i; j++) {  
 if (arr[j] > arr[j + 1]) {  
 swap(arr[j], arr[j + 1]);  
 swapped = true;  
 }  
 }  
  
 if (!swapped) {  
 break;  
 }  
 }  
}  
  
void shaker\_sort(int \*arr, int n) {  
 bool swapped = true;  
 int start = 0;  
 int end = n - 1;  
  
 while (swapped) {  
 swapped = false;  
  
 for (int i = start; i < end; ++i) {  
 if (arr[i] > arr[i + 1]) {  
 swap(arr[i], arr[i + 1]);  
 swapped = true;  
 }  
 }  
  
 if (!swapped) {  
 break;  
 }  
  
 swapped = false;  
 --end;  
  
 for (int i = end - 1; i >= start; --i) {  
 if (arr[i] > arr[i + 1]) {  
 swap(arr[i], arr[i + 1]);  
 swapped = true;  
 }  
 }  
  
 ++start;  
 }  
}  
  
void comb\_sort(int \*arr, int n) {  
 const double shrinkFactor = 1.247;  
 int gap = n;  
 bool swapped = true;  
  
 while (gap > 1 || swapped) {  
 gap = gap / shrinkFactor;  
 if (gap < 1) {  
 gap = 1;  
 }  
  
 swapped = false;  
  
 for (int i = 0; i + gap < n; i++) {  
 if (arr[i] > arr[i + gap]) {  
 swap(arr[i], arr[i + gap]);  
 swapped = true;  
 }  
 }  
 }  
}  
  
void insertion\_sort(int \*arr, int n) {  
 int i, key, j;  
 for (i = 1; i < n; i++) {  
 key = arr[i];  
 j = i - 1;  
  
 while (j >= 0 && arr[j] > key) {  
 arr[j + 1] = arr[j];  
 j = j - 1;  
 }  
 arr[j + 1] = key;  
 }  
}  
  
void quicksort(int \*arr, int end, int begin) {  
 int mid;  
 int f = begin;  
 int l = end;  
 mid = arr[(f + l) / 2];  
 while (f < l) {  
 while (arr[f] < mid) f++;  
 while (arr[l] > mid) l--;  
 if (f <= l) {  
 swap(arr[f], arr[l]);  
 f++;  
 l--;  
 }  
 }  
 if (begin < l) quicksort(arr, l, begin);  
 if (f < end) quicksort(arr, end, f);  
}  
  
int binary\_search(int \*arr, int value, int start, int end) {  
 if (end >= start) {  
 int mid = start + (end - start) / 2;  
  
 if (arr[mid] == value) {  
 return mid;  
 }  
  
 if (arr[mid] > value) {  
 return binary\_search(arr, value, start, mid - 1);  
 }  
  
 return binary\_search(arr, value, mid + 1, end);  
 }  
  
 return -1;  
}  
  
void print\_array(int \*arr, int n) {  
 for (int i = 0; i < n; ++i) {  
 cout << arr[i] << " ";  
 }  
 cout << endl;  
}  
  
void copy\_array(const int \*src, int \*dest, int n) {  
 for (int i = 0; i < n; ++i) {  
 dest[i] = src[i];  
 }  
}  
  
void task\_1() {  
 cout << "--- task #1 ----" << endl;  
 for (int i = 0; i < N; ++i) {  
 unsorted\_array[i] = rand() % 199 - 99;  
 }  
  
 cout << "Unsorted array:" << endl;  
 print\_array(unsorted\_array, N);  
  
 cout << "Sorted array:" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
 print\_array(sorted\_array, N);  
}  
  
void task\_2() {  
 cout << "--- task #2 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 start\_time = steady\_clock::*now*();  
 bubble\_sort(sorted\_array, N);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Bubble sort, ns:\t" << result\_time.count() << endl;  
  
 copy\_array(unsorted\_array, sorted\_array, N);  
 start\_time = steady\_clock::*now*();  
 shaker\_sort(sorted\_array, N);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Shaker sort, ns:\t" << result\_time.count() << endl;  
  
 copy\_array(unsorted\_array, sorted\_array, N);  
 start\_time = steady\_clock::*now*();  
 comb\_sort(sorted\_array, N);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Comb sort, ns :\t" << result\_time.count() << endl;  
  
 copy\_array(unsorted\_array, sorted\_array, N);  
 start\_time = steady\_clock::*now*();  
 insertion\_sort(sorted\_array, N);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Insertion sort, ns:\t" << result\_time.count() << endl;  
  
 copy\_array(unsorted\_array, sorted\_array, N);  
 start\_time = steady\_clock::*now*();  
 quicksort(sorted\_array, N - 1, 0);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Quick sort, ns: \t" << result\_time.count() << endl;  
}  
  
void task\_3() {  
 cout << "--- task #3 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
  
 int min = 100, max = -100;  
  
 start\_time = steady\_clock::*now*();  
 for (int i = 0; i < N; ++i) {  
 if (min > unsorted\_array[i]) {  
 min = unsorted\_array[i];  
 }  
 if (max < unsorted\_array[i]) {  
 max = unsorted\_array[i];  
 }  
 }  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Min: " << min << ", max: " << max << endl;  
 cout << "Unsorted search min and max time, ns: " << result\_time.count() << endl;  
  
 start\_time = steady\_clock::*now*();  
 min = sorted\_array[0];  
 max = sorted\_array[N - 1];  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Min: " << min << ", max: " << max << endl;  
 cout << "Sorted search min and max time, ns: " << result\_time.count() << endl;  
}  
  
void task\_4() {  
 cout << "--- task #4 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
 int min = sorted\_array[0], max = sorted\_array[N - 1], count = 0;  
 int average = (max + min) / 2;  
 cout << "Average min and max value: " << average << endl;  
 cout << "Indexes with same value, unsorted array: ";  
 start\_time = steady\_clock::*now*();  
 for (int i = 0; i < N; ++i) {  
 if (unsorted\_array[i] == average) {  
 cout << i << " ";  
 count++;  
 }  
 }  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << endl << "Indexes count: " << count << endl;  
 cout << "Time spent in unsorted array, ns: " << result\_time.count() << endl;  
  
 count = 0;  
 cout << "Indexes with same value, sorted array: ";  
 start\_time = steady\_clock::*now*();  
 for (int i = 0; sorted\_array[i] <= average; ++i) {  
 if (sorted\_array[i] == average) {  
 cout << i << " ";  
 count++;  
 }  
 }  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << endl << "Indexes count: " << count << endl;  
 cout << "Time spent in sorted array, ns: " << result\_time.count() << endl;  
}  
  
void task\_5() {  
 cout << "--- task #5 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
 int a, counter = 0;  
 print\_array(sorted\_array, N);  
 cout << "Enter number to count less elements: ";  
 cin >> a;  
 while (sorted\_array[counter++] < a);  
 cout << "Result: " << --counter << endl;  
}  
  
void task\_6() {  
 cout << "--- task #6 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
 int b, counter = N - 1;  
 print\_array(sorted\_array, N);  
 cout << "Enter number to count greater elements: ";  
 cin >> b;  
 while (sorted\_array[counter--] > b);  
 cout << "Result: " << N - counter - 2 << endl;  
}  
  
void task\_7() {  
 cout << "--- task #7 ----" << endl;  
 copy\_array(unsorted\_array, sorted\_array, N);  
 quicksort(sorted\_array, N - 1, 0);  
 int number, index = -1;  
 print\_array(sorted\_array, N);  
 cout << "Enter number to search in array: ";  
 cin >> number;  
  
 start\_time = steady\_clock::*now*();  
 for (int i = 0; i < N; ++i) {  
 if (sorted\_array[i] == number) {  
 index = i + 1;  
 break;  
 }  
 }  
 end\_time = steady\_clock::*now*();  
  
 if (index == -1) {  
 cout << "Not found" << endl;  
 } else {  
 cout << "Found at " << index << " position" << endl;  
 }  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Time spent, simple search, ns: " << result\_time.count() << endl;  
  
 start\_time = steady\_clock::*now*();  
 index = binary\_search(sorted\_array, number, 0, N - 1);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Time spent, binary search, ns: " << result\_time.count() << endl;  
}  
  
void task\_8() {  
 cout << "--- task #8 ----" << endl;  
 cout << "Array:" << endl;  
 print\_array(unsorted\_array, N);  
 cout << "Enter two indexes for swap: ";  
 int i1, i2;  
 cin >> i1 >> i2;  
 start\_time = steady\_clock::*now*();  
 swap(unsorted\_array[i1], unsorted\_array[i2]);  
 end\_time = steady\_clock::*now*();  
 result\_time = duration\_cast<nanoseconds>(end\_time - start\_time);  
 cout << "Time spent for swap, ns: " << result\_time.count() << endl;  
 cout << "New array:" << endl;  
 print\_array(unsorted\_array, N);  
}  
  
void task\_9() {  
 cout << "--- individual homework #9 ----" << endl;  
 cout << "Array:" << endl;  
 print\_array(unsorted\_array, N);  
 cout << "Enter number: ";  
 int number;  
 cin >> number;  
 for (int i = 1; i < N; i += 2) {  
 unsorted\_array[i] -= number;  
 unsorted\_array[i] \*= rand() % 9 + 1;  
 }  
 cout << "New array:" << endl;  
 print\_array(unsorted\_array, N);  
 cout << "Number of elements that are divisible by:" << endl;  
 for (int i = 1; i < 10; ++i) {  
 int counter = 0;  
 for (int j = 0; j < N; ++j) {  
 if (unsorted\_array[j] % i == 0) {  
 counter++;  
 }  
 }  
 cout << i << " - " << counter << endl;  
 }  
}

**Файл lab2.h**

//  
// Created by Ilia on 12/5/2024.  
//  
  
#ifndef LAB2\_H  
#define LAB2\_H  
  
int run\_lab\_2();  
#endif //LAB2\_H

**Файл lab3.cpp**

#include <iostream>  
#include <ctime>  
#include <Windows.h>  
  
using namespace std;  
  
const int N = 6;  
const int WIDTH = 3;  
  
HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);  
COORD destCoord;  
  
  
void set\_element(int matrix[][N], int x, int y, int offset\_x, int offset\_y) {  
 \*(\*(matrix + y) + x) = (rand() % (N \* N)) + 1;  
 destCoord.X = (x + offset\_x) \* WIDTH;  
 destCoord.Y = y + offset\_y;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + y) + x);  
 Sleep(10);  
}  
  
void print\_matrix(int matrix[][N], int offset\_x, int offset\_y) {  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < N; j++) {  
 destCoord.X = (i + offset\_x) \* WIDTH;  
 destCoord.Y = j + offset\_y;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j);  
 }  
 }  
}  
  
void quicksort(int arr[][N], int end, int begin) {  
 int f = begin;  
 int l = end;  
 int index = (f + l) / 2;  
 int mid = \*(\*(arr + index % N) + index / N);  
 while (f < l) {  
 while (\*(\*(arr + f % N) + f / N) < mid) f++;  
 while (\*(\*(arr + l % N) + l / N) > mid) l--;  
 if (f <= l) {  
 swap(\*(\*(arr + f % N) + f / N), \*(\*(arr + l % N) + l / N));  
 f++;  
 l--;  
 }  
 }  
 if (begin < l) quicksort(arr, l, begin);  
 if (f < end) quicksort(arr, end, f);  
}  
  
void show\_a(int matrix[][N], int offset\_x, int offset\_y) {  
 for (int i = 0; i < N / 2; i++) {  
 for (int j = 0; j < N / 2; j++) {  
 destCoord.X = offset\_x + j \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j);  
  
 destCoord.X = offset\_x + (j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j);  
  
 destCoord.X = offset\_x + j \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j + N / 2);  
  
 destCoord.X = offset\_x + (j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j + N / 2);  
 }  
 }  
}  
  
void show\_b(int matrix[][N], int offset\_x, int offset\_y) {  
 for (int i = 0; i < N / 2; i++) {  
 for (int j = 0; j < N / 2; j++) {  
 destCoord.X = (offset\_x + (j + N / 2)) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j);  
  
 destCoord.X = (offset\_x + j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j);  
  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j + N / 2);  
  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j + N / 2);  
 }  
 }  
}  
  
void show\_c(int matrix[][N], int offset\_x, int offset\_y) {  
 for (int i = 0; i < N / 2; i++) {  
 for (int j = 0; j < N / 2; j++) {  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j);  
  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j);  
  
 destCoord.X = (offset\_x + j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j + N / 2);  
  
 destCoord.X = (offset\_x + j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j + N / 2);  
 }  
 }  
}  
  
void show\_d(int matrix[][N], int offset\_x, int offset\_y) {  
 for (int i = 0; i < N / 2; i++) {  
 for (int j = 0; j < N / 2; j++) {  
 destCoord.X = (offset\_x + j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j);  
  
 destCoord.X = (offset\_x + j + N / 2) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j);  
  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i + N / 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i + N / 2) + j + N / 2);  
  
 destCoord.X = (offset\_x + j) \* WIDTH;  
 destCoord.Y = offset\_y + i;  
 SetConsoleCursorPosition(hStdout, destCoord);  
 cout << \*(\*(matrix + i) + j + N / 2);  
 }  
 }  
}  
  
int run\_lab\_3() {  
 srand(time(nullptr));  
 system("cls");  
  
 int matrixA[N][N], matrixB[N][N], y = 0, x = 0;  
  
 cout << "Исходные матрицы: ";  
 for (int i = 0; i < N \* 2 - 1; ++i) {  
 switch (i % 4) {  
 case 0:  
 y = i / 4;  
 for (x = i / 4; x < N - i / 4; ++x) {  
 set\_element(matrixA, x, y, 0, 1);  
 }  
 break;  
 case 1:  
 x = N - i / 4 - 1;  
 for (y = i / 4 + 1; y < N - i / 4; ++y) {  
 set\_element(matrixA, x, y, 0, 1);  
 }  
 break;  
 case 2:  
 y = N - i / 4 - 1;  
 for (x = N - i / 4 - 2; x >= i / 4; --x) {  
 set\_element(matrixA, x, y, 0, 1);  
 }  
 break;  
 case 3:  
 x = i / 4;  
 for (y = N - i / 4 - 2; y >= i / 4 + 1; --y) {  
 set\_element(matrixA, x, y, 0, 1);  
 }  
 break;  
 }  
 }  
  
 for (int i = 0; i < N; ++i) {  
 if (i % 2 == 0) {  
 for (int j = 0; j < N; ++j) {  
 set\_element(matrixB, i, j, N + 2, 1);  
 }  
 } else {  
 for (int j = N - 1; j >= 0; --j) {  
 set\_element(matrixB, i, j, N + 2, 1);  
 }  
 }  
 }  
  
 destCoord.X = 0;  
 destCoord.Y = N + 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 cout << "Перестановки первой матрицы: ";  
  
 show\_a(matrixA, 0, N + 3);  
 show\_b(matrixA, N + 1, N + 3);  
 show\_c(matrixA, (N + 1) \* 2, N + 3);  
 show\_d(matrixA, (N + 1) \* 3, N + 3);  
  
 destCoord.X = 0;  
 destCoord.Y = (N + 2) \* 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 cout << "Перестановки второй матрицы: ";  
  
 show\_a(matrixB, 0, (N + 2) \* 2 + 1);  
 show\_b(matrixB, N + 1, (N + 2) \* 2 + 1);  
 show\_c(matrixB, (N + 1) \* 2, (N + 2) \* 2 + 1);  
 show\_d(matrixB, (N + 1) \* 3, (N + 2) \* 2 + 1);  
  
 destCoord.X = 0;  
 destCoord.Y = (N + 2) \* 3;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 system("pause");  
 system("cls");  
  
 cout << "Исходные матрицы: ";  
 print\_matrix(matrixA, 0, 1);  
 print\_matrix(matrixB, N + 1, 1);  
  
 destCoord.X = 0;  
 destCoord.Y = (N + 2) \* 1;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 cout << "Отсортированные матрицы: ";  
 quicksort(matrixA, N \* N - 1, 0);  
 quicksort(matrixB, N \* N - 1, 0);  
 print\_matrix(matrixA, 0, (N + 2) \* 1 + 1);  
 print\_matrix(matrixB, N + 1, (N + 2) \* 1 + 1);  
  
 destCoord.X = 0;  
 destCoord.Y = (N + 2) \* 2;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 system("pause");  
 system("cls");  
  
 cout << "Модификация всех элементов матрицы" << endl;  
 cout << "Введите число: ";  
 int number;  
 cin >> number;  
  
 char operation;  
 do {  
 cout << "Введите знак операции (+, -, \*, /): ";  
 cin >> operation;  
 } while (operation != '+' && operation != '-' && operation != '\*' && operation != '/');  
  
 system("cls");  
 cout << "Исходные матрицы: ";  
 print\_matrix(matrixA, 0, 1);  
 print\_matrix(matrixB, N + 1, 1);  
  
 for (int i = 0; i < N; ++i) {  
 for (int j = 0; j < N; ++j) {  
 switch (operation) {  
 case '+':  
 \*(\*(matrixA + i) + j) += number;  
 \*(\*(matrixB + i) + j) += number;  
 break;  
 case '-':  
 \*(\*(matrixA + i) + j) -= number;  
 \*(\*(matrixB + i) + j) -= number;  
 break;  
 case '\*':  
 \*(\*(matrixA + i) + j) \*= number;  
 \*(\*(matrixB + i) + j) \*= number;  
 break;  
 case '/':  
 \*(\*(matrixA + i) + j) /= number;  
 \*(\*(matrixB + i) + j) /= number;  
 break;  
 }  
 }  
 }  
  
 destCoord.X = 0;  
 destCoord.Y = (N + 2) \* 1;  
 SetConsoleCursorPosition(hStdout, destCoord);  
  
 cout << "Результат: ";  
 print\_matrix(matrixA, 0, (N + 2) \* 1 + 1);  
 print\_matrix(matrixB, N + 1, (N + 2) \* 1 + 1);  
  
 cout << endl;  
 system("pause");  
 system("cls");  
  
 cout << "Индивидуальное домашнее задание №10" << endl;  
 cout << "Матрица 3x3 со случайными числами от -30 до 30: " << endl;  
  
 int matrixC[3][3];  
 for (int i = 0; i < 3; ++i) {  
 for (int j = 0; j < 3; ++j) {  
 \*(\*(matrixC + i) + j) = rand() % 61 - 30;  
 cout << \*(\*(matrixC + i) + j) << '\t';  
 }  
 cout << endl;  
 }  
 int a = \*(\*(matrixC + 0) + 0) \* \*(\*(matrixC + 1) + 1) \* \*(\*(matrixC + 2) + 2);  
 int b = \*(\*(matrixC + 0) + 1) \* \*(\*(matrixC + 1) + 2) \* \*(\*(matrixC + 2) + 0);  
 int c = \*(\*(matrixC + 0) + 2) \* \*(\*(matrixC + 1) + 0) \* \*(\*(matrixC + 2) + 1);  
 int d = \*(\*(matrixC + 0) + 2) \* \*(\*(matrixC + 1) + 1) \* \*(\*(matrixC + 2) + 0);  
 int e = \*(\*(matrixC + 0) + 0) \* \*(\*(matrixC + 1) + 2) \* \*(\*(matrixC + 2) + 1);  
 int f = \*(\*(matrixC + 0) + 1) \* \*(\*(matrixC + 1) + 0) \* \*(\*(matrixC + 2) + 2);  
 int det = a + b + c - (d + e + f);  
 cout << "Вычисление определителя матрицы..." << endl;  
 Sleep(50);  
 cout << "Произведение элементов главной диагонали: " << a << endl;  
 Sleep(50);  
 cout << "Произведение элементов первого треугольника параллельного главной диагонали: " << b << endl;  
 Sleep(50);  
 cout << "Произведение элементов второго треугольника параллельного главной диагонали: " << c << endl;  
 Sleep(50);  
 cout << "Произведение элементов побочной диагонали: " << d << endl;  
 Sleep(50);  
 cout << "Произведение элементов первого треугольника параллельного побочной диагонали: " << e << endl;  
 Sleep(50);  
 cout << "Произведение элементов второго треугольника параллельного побочной диагонали: " << f << endl;  
 Sleep(50);  
 cout << "Вычисленный определитель: " << det << endl;  
 system("pause");  
 return 0;  
}

**Файл lab3.h**

//  
// Created by Ilia on 12/5/2024.  
//  
  
#ifndef LAB3\_H  
#define LAB3\_H  
  
int run\_lab\_3();  
#endif //LAB3\_H

**Файл lab4.cpp**

#include <algorithm>  
#include <iostream>  
#include <string>  
#include <fstream>  
#include <limits>  
#include <vector>  
  
using namespace std;  
  
string words[50], lower\_words[50], punctuation[50];  
  
bool is\_letter\_or\_number(const char ch) {  
 return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9');  
}  
  
int linear\_search\_substring(const string &text, const string &pattern) {  
 const size\_t n = text.length();  
 const size\_t m = pattern.length();  
  
 for (int i = 0; i <= n - m; i++) {  
 int j;  
 for (j = 0; j < m; j++) {  
 if (tolower(text[i + j]) != tolower(pattern[j]))  
 break;  
 }  
  
 if (j == m)  
 return i;  
 }  
  
 return -1;  
}  
  
int search\_substrings\_count(string text, const string &pattern) {  
 int count = 0, index;  
 do {  
 index = linear\_search\_substring(text, pattern);  
 if (index != -1) {  
 count++;  
 text = text.substr(index + 1, text.length() - index - 1);  
 }  
 } while (index != -1);  
 return count;  
}  
  
vector<int> compute\_lps(string pattern) {  
 int n = pattern.length();  
 vector<int> lps(n, 0);  
 int len = 0;  
 int i = 1;  
  
 while (i < n) {  
 if (pattern[i] == pattern[len]) {  
 len++;  
 lps[i] = len;  
 i++;  
 } else {  
 if (len != 0) {  
 len = lps[len - 1];  
 } else {  
 lps[i] = 0;  
 i++;  
 }  
 }  
 }  
  
 return lps;  
}  
  
vector<int> kmp(string text, string pattern) {  
 vector<int> lps = compute\_lps(pattern);  
 vector<int> occurrences;  
 int n = text.length();  
 int m = pattern.length();  
 int i = 0;  
 int j = 0;  
  
 while (i < n) {  
 if (tolower(text[i]) == tolower(pattern[j])) {  
 i++;  
 j++;  
 }  
  
 if (j == m) {  
 occurrences.push\_back(i - j);  
 j = lps[j - 1];  
 } else if (i < n && text[i] != pattern[j]) {  
 if (j != 0) {  
 j = lps[j - 1];  
 } else {  
 i++;  
 }  
 }  
 }  
  
 return occurrences;  
}  
  
  
int run\_lab\_4() {  
 cout << "Выберите способ ввода текста:" << endl;  
 cout << "1. С клавиатуры" << endl;  
 cout << "2. Из файла" << endl;  
 int ans = -1;  
 do {  
 cout << "Ваш выбор: ";  
 cin >> ans;  
 } while (ans != 1 && ans != 2);  
  
 cin.ignore(numeric\_limits<streamsize>::*max*(), '\n');  
  
  
 string text;  
 if (ans == 1) {  
 cout << "Введите текст" << endl;  
 getline(cin, text);  
 } else {  
 cout << "Введите путь до файла с текстом: ";  
 string path;  
 getline(cin, path);  
 fstream file(path, ios\_base::*in*);  
 if (!file.is\_open()) {  
 cout << "Не удалось открыть файл, завершаю выполнение.";  
 return 1;  
 }  
 getline(file, text);  
 file.close();  
 }  
  
 int start = 0, ind1 = 0, ind2 = 0;  
 for (int i = 0; i < text.length() - 1; i++) {  
 if (is\_letter\_or\_number(text[i]) && !is\_letter\_or\_number(text[i + 1])) {  
 words[ind1++] = text.substr(start, i - start + 1);  
 start = i + 1;  
 } else if (!is\_letter\_or\_number(text[i]) && is\_letter\_or\_number(text[i + 1])) {  
 punctuation[ind2++] = text.substr(start, i - start + 1);  
 start = i + 1;  
 }  
 }  
 punctuation[ind2++] = ".";  
  
 for (int i = 0; i < ind1; i++) {  
 for (int j = 1; j < words[i].length(); j++) {  
 words[i][j] = tolower(words[i][j]);  
 }  
 }  
  
  
 for (int i = 0; i < ind2; i++) {  
 string buffer;  
 buffer.push\_back(punctuation[i][0]);  
 for (int j = 1; j < punctuation[i].length(); j++) {  
 if (buffer.back() == '.' &&  
 j + 1 < punctuation[i].length() &&  
 punctuation[i][j] == '.' &&  
 punctuation[i][j + 1] == '.') {  
 buffer.push\_back('.');  
 buffer.push\_back('.');  
 j++;  
 continue;  
 }  
  
 if (buffer.back() != punctuation[i][j]) {  
 buffer.push\_back(punctuation[i][j]);  
 }  
 }  
 punctuation[i] = buffer;  
 }  
 for (int i = 0; i < ind1; i++) {  
 cout << words[i] << punctuation[i];  
 lower\_words[i] = words[i];  
 lower\_words[i][0] = tolower(lower\_words[i][0]);  
 }  
  
 cout << endl;  
 sort(lower\_words, lower\_words + ind1);  
 cout << "Задание 3. Вариант 2. Слова последовательности в алфавитном порядке." << endl;  
 for (int i = 0; i < ind1; i++) {  
 cout << lower\_words[i] << " ";  
 }  
 cout << endl;  
 cout << "Задание 4. Вариант 2. Вывести на экран количество символов в каждом слове исходной последовательности." <<  
 endl;  
 for (int i = 0; i < ind1; i++) {  
 cout << words[i] << " - " << words[i].length() << " || ";  
 }  
 cout << endl;  
 cout << "Задание 5. Поиск подстроки в строке методом линейного поиска и КМП." << endl;  
 string long\_line;  
 for (int i = 0; i < ind1; i++) {  
 long\_line += words[i] + punctuation[i];  
 }  
 string pattern;  
 cout << "Введите подстроку для поиска: ";  
 getline(cin, pattern);  
 int result = search\_substrings\_count(long\_line, pattern);  
 auto kmp\_result = kmp(long\_line, pattern);  
 cout << "Найдено " << result << " подстрок." << endl;  
 cout << "Найдено " << kmp\_result.size() << " подстрок." << endl;  
 system("pause");  
 return 0;  
}

**Файл lab4.h**

//  
// Created by Ilia on 12/5/2024.  
//  
  
#ifndef LAB4\_H  
#define LAB4\_H  
  
int run\_lab\_4();  
#endif //LAB4\_H