**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4372 |  | Кубрина А.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Соединить все 4 лабораторные работы в одну

**Постановка задачи.**

Необходимо объединить все 4 лабораторные работы в единый проект. Нужно добавить инфраструктуру переключения между заданиями (интерактивное меню).

**Выполнение работы.**

Код программы представлен в приложении А.

| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| --- | --- |
| Загрузка программы | |
| При запуске программы пользователь вводит номер работы | Ввод текста: |
| Лабораторная работа 1 | |
| Запуск первой лабораторной работы |  |
| Лабораторная работа 2 | |
| Запуск второй лабораторной работы |  |
| Лабораторная работа 3 | |
| Запуск третий лабораторной работы |  |
| Лабораторная работа 4 | |
| Запуск четвертый лабораторной работы |  |

**Выводы.**

Все 4 лабораторной работы были объединены в одну работу.

Приложение А

рабочий код

#include<iostream>

#include <windows.h>

#include <fstream>

#include <thread>

#include <chrono>

#include <ctime>

#include <bitset>

#include <cstdio>

using namespace std;

using namespace chrono;

int main\_lab1() {

//задание 1

setlocale(0, "");

cout << "Задание 1" << endl;

cout << "int: " << sizeof(int) << endl;

cout << "short int: " << sizeof(short int) << endl;

cout << "long int: " << sizeof(long int) << endl;

cout << "float: " << sizeof(float) << endl;

cout << "long double: " << sizeof(long double) << endl;

cout << "char: " << sizeof(char) << endl;

cout << "bool: " << sizeof(bool) << endl;

cout << "\n";

//задание 2

cout << "Задание 2" << endl;

cout << "Введите число: ";

int value;

cin >> value;

unsigned int order = sizeof(int) \* 8;

unsigned int mask = 1 << order - 1;

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

{

if (value & mask) {

cout << '1';

}

else {

cout << '0';

}

mask >>= 1;

if (i % 8 == 0 || i % order - 1 == 0)

{

cout << ' ';

}

}

cout << "\n";

//задание 3

cout << "Задание 3" << endl;

cout << "Введите число: ";

union {

float y;

int x2;

};

order = sizeof(float) \* 8;

mask = 1 << order - 1;

cin >> y;

for (int i = 1; i <= order; i++) {

if (x2 & mask) {

cout << '1';

}

else {

cout << '0';

}

mask >>= 1;

if (i % 8 == 0 || i % order - 1 == 0)

{

cout << ' ';

}

}

cout << endl;

//Задание 4

cout << "Задание 4" << endl;

union {

double y2;

long long int x;

};

unsigned long long newmask = 1LL << sizeof(double) \* 8 - 1;

cin >> y2;

for (int j = 1; j <= sizeof(double) \* 8; j++, newmask >>= 1) {

if (x & newmask) {

cout << '1';

}

else {

cout << '0';

}

if (j % 8 == 0 || j % sizeof(double) \* 8 - 1 == 0)

{

cout << ' ';

}

}

cout << endl;

return 0;

}

void quicksort(int\* arr, int end, int begin) //сортировка quicksort

{

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 binarySearch(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 binarySearch(arr, value, start, mid - 1);

}

return binarySearch(arr, value, mid + 1, end);

}

return -1;

}

int main\_lab2() {

// Задание 1

cout << "Задание 1" << endl;

const int N = 100; // заполнили массив случайными числами в диапозоне от -99 до 99

int randomNum[N] = {};

cout << "Наш массив: " << endl;

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

{

randomNum[i] = rand() % 199 - 99;

cout << randomNum[i] << " ";

}

cout << endl << endl;

// Задание 2

cout << "Задание 2" << endl;

int end = N - 1;

int begin = 0;

auto t1 = chrono::steady\_clock::now();

quicksort(randomNum, end, begin);

auto t2 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration1 = t2 - t1;

cout << "Время сортировки массива: " << duration1.count() << "" << endl;

cout << "Отсортированный массив: " << endl;

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

{

cout << randomNum[i] << " ";

}

cout << endl << endl;;

// Задание 3

cout << "Задание 3" << endl;

cout << "Новый массив: " << endl;

int randomNum2[N] = {}; // новый массив

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

{

randomNum2[i] = rand() % 199 - 99;

cout << randomNum2[i] << " ";

}

cout << endl;

auto start2 = chrono::steady\_clock::now();

int minValue = randomNum2[0];

int maxValue = randomNum2[0];

for (int j = 1; j < N; ++j) {

if (randomNum2[j] < minValue) {

minValue = randomNum2[j];

}

if (randomNum2[j] > maxValue) {

maxValue = randomNum2[j];

}

}

auto end2 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration2 = end2 - start2;

cout << "Минимальное значение: " << minValue << endl << "Максимальное: " << maxValue << endl << "Время нахождения мин и макс значения в неотсортированом массиве: " << duration2.count() << "" << endl;

cout << endl << "Теперь в отсортированном массиве: " << endl;

quicksort(randomNum2, end, begin);

auto start3 = chrono::steady\_clock::now();

int minValue2 = randomNum2[0];

int maxValue2 = randomNum2[N - 1];

auto end3 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration3 = end3 - start3;

cout << "Минимальное значение: " << minValue2 << endl << "Максимальное: " << maxValue2 << endl << "Время нахождения мин и макс значения в отсортированом массиве: " << duration3.count() << "" << endl;

cout << "Отсортированный массив: " << endl;

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

{

cout << randomNum2[i] << " ";

}

cout << endl << endl;;

cout << endl << endl;;

// Задание 4

cout << "Задание 4" << endl;

int middle1 = (minValue + maxValue) / 2;

int middle2 = (minValue2 + maxValue2) / 2;

cout << "Среднее значение в неотсортированном массиве: " << middle1 << endl << "Среднее значение в отсортированом массиве: " << middle2 << endl;

cout << "Индексы всех элементов которые равны " << middle1 << ": ";

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

if (randomNum2[k] == middle1) {

cout << k << " ";

}

}

auto start4 = chrono::steady\_clock::now();

int counter = 0;

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

if (randomNum2[k] == middle1) {

counter++;

}

}

auto end4 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration4 = end4 - start4;

cout << endl;

cout << "Количество элементов которые равны " << middle1 << ": " << counter << endl << "Время поиска: " << duration4.count() << endl << endl;

//Задание 5

cout << "Задание 5" << endl;

int a;

int counterNew = 0;

cout << "Введите число a ";

cin >> a;

for (int g = 0; g < N; g++) {

if (randomNum2[g] < a) {

counterNew++;

}

}

cout << "Количество элементов в отсортированном массиве, которые меньше числа " << a << ": " << counterNew << endl << endl;

//Задание 6

cout << "Задание 6" << endl;

int b;

int counterNew2 = 0;

cout << "Введите число b ";

cin >> b;

for (int g = 0; g < N; g++) {

if (randomNum2[g] > b) {

counterNew2++;

}

}

cout << "Количество элементов в отсортированном массиве, которые больше числа " << b << ": " << counterNew2 << endl << endl;

//Задание 7

cout << "Задание 7" << endl;

int randomNum3[N] = {};

bool arg = false;

cout << "Введите число, которое нужно найти: ";

int Num;

cin >> Num;

cout << endl;

cout << "Новый массив: " << endl;

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

{

randomNum3[i] = rand() % 199 - 99;

cout << randomNum3[i] << " ";

}

cout << endl << endl;

auto start5 = chrono::steady\_clock::now();

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

if (randomNum3[i] == Num) {

arg = true;

break;

}

}

auto end5 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration5 = end5 - start5;

if (arg) {

cout << "Число " << Num << " существует" << endl;

}

else {

cout << "Элемент " << Num << " не найден" << endl;

}

cout << "Время затраченное на перебор массива: " << duration5.count() << endl;

quicksort(randomNum3, end, begin);

auto start6 = chrono::steady\_clock::now();

int size = sizeof(randomNum3) / sizeof(randomNum3[0]);

int result = binarySearch(randomNum3, Num, 0, size - 1);

auto end6 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration6 = end6 - start6;

cout << endl;

if (result == -1) {

cout << "Элемент " << Num << " не найден" << endl;

}

else {

cout << "Число " << Num << " существует" << endl;

}

cout << "Время затраченное на бинарный поиск: " << duration6.count() << endl;

cout << endl;

//Задание 8

int index1, index2;

cout << "Введите первый индекс: ";

cin >> index1;

cout << endl;

cout << "Введите второй индекс: ";

cin >> index2;

cout << endl;

cout << "Массив был таким: " << endl;

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

{

cout << randomNum3[i] << " ";

}

cout << endl;

auto start7 = chrono::steady\_clock::now();

swap(randomNum3[index1], randomNum3[index2]);

auto end7 = chrono::steady\_clock::now();

chrono::duration<double, milli>duration7 = end7 - start7;

cout << endl;

cout << "Массив стал таким: " << endl;

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

{

cout << randomNum3[i] << " ";

}

cout << endl;

cout << "Время, за которое переставили элементы: " << duration7.count() << endl << endl << endl;

//ИДЗ2

cout << "ИДЗ2" << endl;

int Part1[50] = {};

int Part2[50] = {};

for (int i1 = 0; i1 < N / 2; i1++) {

Part1[i1] = randomNum[i1];

Part2[i1] = randomNum[50 + i1];

}

cout << "Первая часть массива: " << endl;

for (int i = 0; i < N / 2; i++)

{

cout << Part1[i] << " ";

}

cout << endl;

cout << "Вторая часть массива: " << endl;

for (int i = 0; i < N / 2; i++)

{

cout << Part2[i] << " ";

}

cout << endl << endl;

cout << "Первый и второй массив после того, как поменяли значения нечетных элементов 1-го массива с четными элементами 2-го массива" << endl << endl;

int n;

for (int t = 0; t < N / 2; t += 2) {

n = Part1[t + 1];

Part1[t + 1] = Part2[t];

Part2[t] = n;

}

cout << "Первый массив: " << endl;

for (int i = 0; i < N / 2; i++)

{

cout << Part1[i] << " ";

}

cout << endl;

cout << "Второй массив: " << endl;

for (int i = 0; i < N / 2; i++)

{

cout << Part2[i] << " ";

}

cout << endl << endl;

cout << "Второй массив, в котором все значения стали четными: " << endl;

for (int h = 0; h < N / 2; h++) {

if (Part2[h] % 2 != 0 && Part2[h] < 99) {

Part2[h]++;

}

else if (Part2[h] % 2 != 0 && Part2[h] == 99) {

Part2[h]--;

}

}

for (int i = 0; i < N / 2; i++)

{

cout << Part2[i] << " ";

}

int counter5 = 0;

for (int r = 0; r < N / 2; r++) {

if (Part1[r] > Part2[r]) {

counter5++;

}

}

cout << endl << endl << "Количество элементов, которые больше в 1-м массиве, чем во 2-м: " << counter5 << endl;

return 0;

}

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 mid;

int f = begin;

int l = end;

int index = (f + l) / 2;

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);

}

}

}

void printMatrix(int\* matrix, int n, int helper) {

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD destCoord;

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

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

destCoord.X = j \* 4 + 50;

destCoord.Y = i + helper;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(matrix + i \* n + j) << " ";

}

cout << endl;

}

}

void idz() {

int N;

cout << "Введите число для ИДЗ(от 2 до 10)" << endl;

cin >> N;

int\* matrix1 = new int[N \* N];

int\* matrix2 = new int[N \* N];

int\* matrix3 = new int[N \* N];

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

for (int j = 0; j < N; ++j) {

\*(matrix1 + i \* N + j) = rand() % (N \* N) + 1;

\*(matrix2 + i \* N + j) = rand() % (N \* N) + 1;

\*(matrix3 + i \* N + j) = 1;

}

}

cout << "Первая матрица" << endl;

printMatrix(matrix1, N, 0);

cout << "Вторая матрица" << endl;

printMatrix(matrix2, N, N + 1);

cout << endl;

int top = 0, bottom = N - 1, left = 0, right = N - 1;

while (top <= bottom && left <= right) {

for (int i = left; i <= right; i++) {

int\* num = matrix3 + i + top \* N;

\*num = \*(matrix2 + i + top \* N) - \*(matrix1 + i + top \* N);

printMatrix(matrix3, N, N + 12);

Sleep(50);

}

top++;

}

printMatrix(matrix3, N, N + 12);

}

int main\_lab3() {

int matrixA[N][N], matrixB[N][N];

int 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 N3" << endl;

idz();

return 0;

}

unsigned lengthStroki(char\* Arr) {

int length = 0;

while (Arr[length]) {

length++;

}

return length;

}

int deleteZnak(char\* arr, char znak, char\* newarr) {

int count = 0;

int index = 0;

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == znak) {

count++;

if (count <= 1) {

newarr[index] = arr[i];

index++;

}

}

else {

newarr[index] = arr[i];

index++;

count = 0;

}

newarr[index] = '\0';

}

return 1;

}

int deleteTochka(char\* arr, char znak, char\* newarr, int newlength) {

int count = 0;

int index = 0;

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

if (arr[i] == znak) {

count++;

if (count <= 3) {

newarr[index] = arr[i];

index++;

}

}

else {

newarr[index] = arr[i];

index++;

count = 0;

}

newarr[index + 1] = '\0';

}

return 1;

}

int BookvaMal(char\* arr, int newlength) {

const int bookva = 32;

if (arr[0] >= 97 and arr[0] <= 122)

arr[0] -= bookva;

for (int i = 1; i < newlength - 2; i++) {

if (arr[i] == ' ' and arr[i - 1] == '.' and arr[i + 1] >= 97 and arr[i + 1] <= 122) {

arr[i + 1] -= bookva;

}

if (arr[i] == ' ' and arr[i - 1] != '.' and arr[i + 1] >= 65 and arr[i + 1] <= 90) {

arr[i + 1] += bookva;

}

if (arr[i] != ' ' and (arr[i - 2] != '.' and arr[i - 1] != '.') and arr[i] >= 65 and arr[i] <= 90)

arr[i] += bookva;

}

return 1;

}

int Number(char\* arr, char\* newarr, int newlength) {

int index = 0;

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

if (arr[i] < 48 or arr[i]>57) {

newarr[index] = arr[i];

index++;

}

newarr[index + 1] = '\0';

}

return 1;

}

int Number2(char\* arr, char\* podstroka, char\* newarr) {

int length1 = lengthStroki(arr);

int length2 = lengthStroki(podstroka);

int count = 0;

int index = 0;

for (int i = 0; i <= length1; i++) {

int j;

for (j = 0; j < length2; j++) {

if (arr[i + j] != podstroka[j])

break;

}

if (j != length2) {

newarr[index] = arr[i];

index++;

}

else {

i = i + length2 - 1;

}

newarr[index + 1] = '\0';

}

return 1;

}

int linearSearch(char\* arr, char\* podstroka) {

int length1 = lengthStroki(arr);

int length2 = lengthStroki(podstroka);

int count = 0;

for (int i = 0; i <= length1 - length2; i++) {

int j;

for (j = 0; j < length2; j++) {

if (arr[i + j] != podstroka[j])

break;

}

if (j == length2)

count++;

}

return count;

}

int main\_lab4() {

int const length = 100;

char stroka[length];

char newstroka[length];

int newlength;

cout << "Введите вашу строку: " << endl;

cin.ignore();

cin.getline(stroka, length);

cin.sync();

char Znak1[length];

deleteZnak(stroka, ' ', Znak1);

char Znak2[length];

deleteZnak(Znak1, ',', Znak2);

char Znak3[length];

deleteZnak(Znak2, '!', Znak3);

char Znak4[length];

deleteZnak(Znak3, '?', Znak4);

char Znak5[length];

deleteZnak(Znak4, '/', Znak5);

char Znak6[length];

deleteZnak(Znak5, ':', Znak6);

char Znak7[length];

deleteZnak(Znak6, ';', Znak7);

char Znak8[length];

deleteZnak(Znak7, '(', Znak8);

char Znak9[length];

deleteZnak(Znak8, ')', Znak9);

char Znak10[length];

deleteZnak(Znak9, '[', Znak10);

char Znak11[length];

deleteZnak(Znak10, ']', Znak11);

char Znak12[length];

deleteZnak(Znak11, '-', Znak12);

char Znak13[length];

deleteZnak(Znak12, '\"', Znak13);

int newlength2 = lengthStroki(Znak5);

deleteTochka(Znak5, '.', newstroka, newlength2);

BookvaMal(newstroka, newlength2);

cout << "Ваша строка: " << endl << newstroka << endl;

char newstroka2[length];

Number(newstroka, newstroka2, newlength2);

cout << "Ваша строка без чисел: " << endl << newstroka2 << endl;

cout << "Введите последовательность которую нужно удалить: " << endl;

char posled[length];

cin.getline(posled, length);

cin.sync();

int posledlength = lengthStroki(posled);

char newstroka3[length];

Number2(newstroka, posled, newstroka3);

cout << "Ваша строка: " << endl << newstroka3 << endl;

cout << "Линейный поиск: " << endl;

char podstroka[length];

cin.getline(podstroka, length);

cin.sync();

int count = linearSearch(newstroka, podstroka);

cout << count << " подстрок было найдено " << endl;

return 0;

}

int main()

{

int i;

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (true) {

system("cls");

cout << "Практическая работа 1" << endl;

cout << "Практическая работа 2" << endl;

cout << "Практическая работа 3" << endl;

cout << "Практическая работа 4" << endl;

cout << "Выберите номер работы: " << endl;

cin >> i;

switch (i) {

case 1:

system("cls");

main\_lab1();

break;

case 2:

system("cls");

main\_lab2();

break;

case 3:

system("cls");

main\_lab3();

break;

case 4:

system("cls");

main\_lab4();

break;

case 5:

cout << "Выход!" << endl;

return 0;

default:

cout << "Некорректный номер работы." << endl;

break;

}

cout << "Работа выполнена. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}