**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. 2372 |  | Астафьева У.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Объединить все 4 лабораторные работы в единый проект.

**Основные теоретические положения.**

Представление положительных и отрицательных числе в прямом, обратном и дополнительном кодах отличается. В прямом коде в знаковый разряд помещается цифра 1, а в разряды цифровой части числа – двоичный код его абсолютной величины.

Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака. Прямой код можно преобразовать в обратный, инвертировав все значения битов.

Дополнительный код получается образованием обратного кода с прибавлением единицы к его младшему разряду.

Увидеть, как тип данных представляется на компьютере, можно при помощи побитового сдвига и поразрядной конъюнкции.

При сдвиге вправо для положительных чисел освобожденные позиции битов, заполняются нулями. Для отрицательных – единицами. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка (экспоненты). Для 32-разрядного процессора для float под мантиссу отводится 23 бита, под экспоненту – 8, под знак – 1. Для double под мантиссу отводится 52 бита, под экспоненту – 11, под знак – 1.

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение. Каждый элемент в этих массивах определяется значением индекса элемента.

Все массивы можно разделить на две группы: *одномерные* и *многомерные*. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

<Базовый тип элементов> <Идентификатор массива> [<Количество элементов>]

Значения индексов элементов массивов всегда начинается с 0. Поэтому максимальное значение индекса элемента в массиве всегда на единицу меньше количества элементов в массиве.

Ввод и вывод массивов реализуются с помощью циклов.

Сортировка – процесс размещения элементов заданного множества объектов в определенном порядке.

Сортировка обменом (bubble sort). Вся суть метода заключается в попарном сравнении элементов и последующем обмене. Таким образом, если следующий элемент меньше текущего, то они меняются местами, максимальный элемент массива постепенно смещается в конец массива, а минимальный – в начало. Один полный проход по массиву может гарантировать, что в конце массива находится максимальный элемент. Затем процесс повторяется до тех пор, пока вся последовательность не будет упорядочена. Важно заметить, что после первого прохода по массиву, уже имеется один упорядоченный элемент, он стоит на своем месте, и менять его не надо. Таким образом на следующем шаге будут сравниваться *N*-1 элемент.

Шейкер-сортировка массива (shaker sort)– модификация пузырьковой сортировки. Принцип работы этой сортировки аналогичен bubble sort: попарное сравнение элементов и последующий обмен местами. Но имеется существенное отличие. Как только максимальный элемент становится на свое место, алгоритм не начинает новую итерацию с первого элемента, а запускает сортировку в обратную сторону. Алгоритм гарантирует, что после выполнения первой итерации, минимальный и максимальный элемент будут в начале и конце массива соответственно.Затем процесс повторяется до тех пор, пока массив не будет отсортирован.

Сортировка расческой (comb sort) – алгоритм был разработан специально для случаев, когда минимальные элементы стоят слишком далеко, или максимальные – слишком близко к началу массива. В сортировке расческой переставляются элементы, стоящие на расстоянии. Оптимально изначально взять расстояние равным длине массива, а далее уменьшать его на определенный коэффициент, который примерно равен 1.247. Когда расстояние станет равно 1, выполняется обычная сортировка пузырьком.

Сортировка вставками (insert sort) – алгоритм сортировки, в котором элементы массива просматриваются по одному, и каждый новый элемент размещается в подходящее место среди ранее упорядоченных элементов. Сортировка вставками делить массив на 2 части – отсортированную и неотсортированную. С каждым новым элементом отсортированная часть будет увеличиваться, а неотсортированная уменьшаться. Причем найти нужное место для очередного элемента в отсортированном массиве достаточно легко.

Алгоритм бинарного поиска – классический алгоритм поиска элемента в отсортированном массиве, который использует дробление массива на половины. Если элемент, который необходимо найти, присутствует в списке, то бинарный поиск возвращает ту позицию, в которой он был найден.

Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратных скобках. Для доступа к определенному элементу многомерного массива необходимо указать в квадратных скобках конкретные значения всех индексов этого элемента.

Указатели – это обычные переменные, но они служат для хранения адресов памяти.

Указатели определяются в программе следующим образом:

<тип данных>\* <имя переменной>

Формально указатели представляют собой обычные целые значения типа int и занимают в памяти 4 байта не зависимо от базового типа указателя. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

Класс string предназначен для работы со строками типа char, которые представляют собой строчку с завершающим нулем (символ ‘\0’). Класс string был введен как альтернативный вариант для работы со строками типа char. Чтобы использовать возможности класса string, нужно подключить библиотеку <string> и пространство имен std. Объявление же переменной типа string осуществляется схоже с обычной переменной.

Основными задачами при обработке текстовых строк являются следующие: определение фактической длины текста, копирование текста из одной строки в другую, объединение строк, сравнение (больше, меньше, равно) строк и т.п.

**Постановка задачи.**

Необходимо написать программу, которая:

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

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователю выводится меню доступных Практических работ и ожидается ввод команды с клавиатуры.

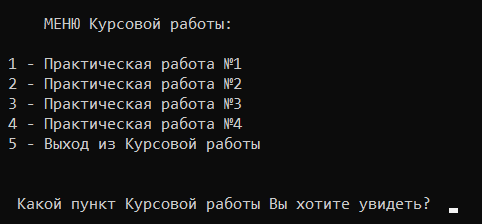


Рисунок 1. – Запуск программы, меню

1. Далее пользователю предлагается ввести интересующий пункт из выбранной работы согласно меню

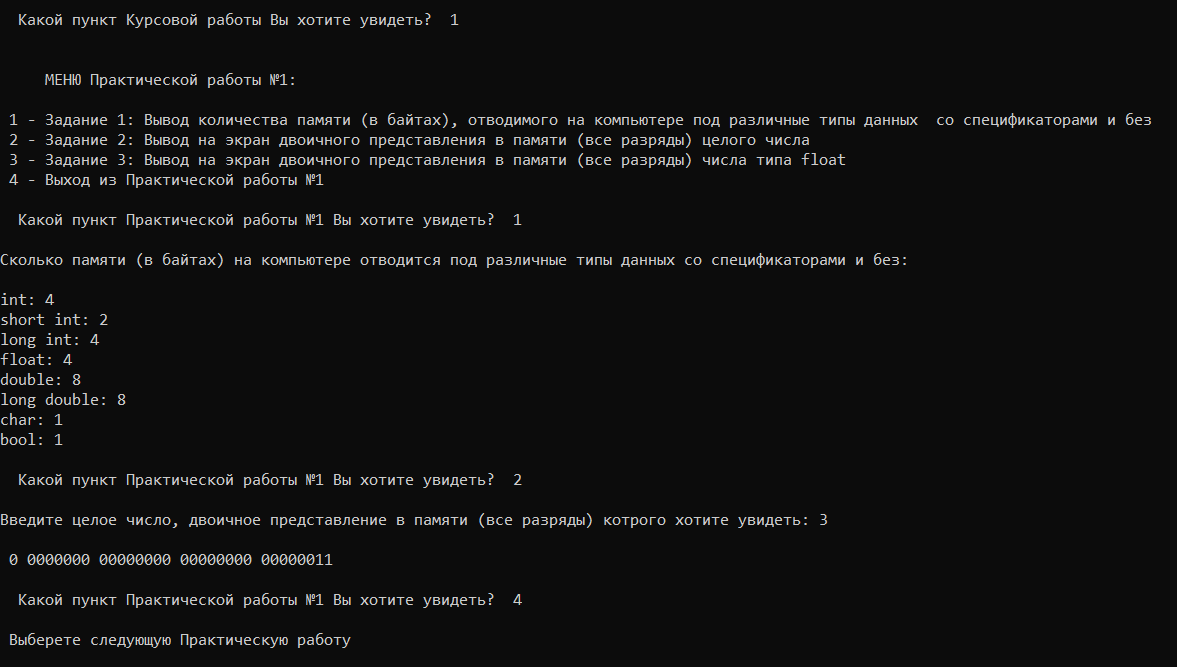


Рисунок . Практическая работа №1. Интерактивный выбор любого пункта согласно меню

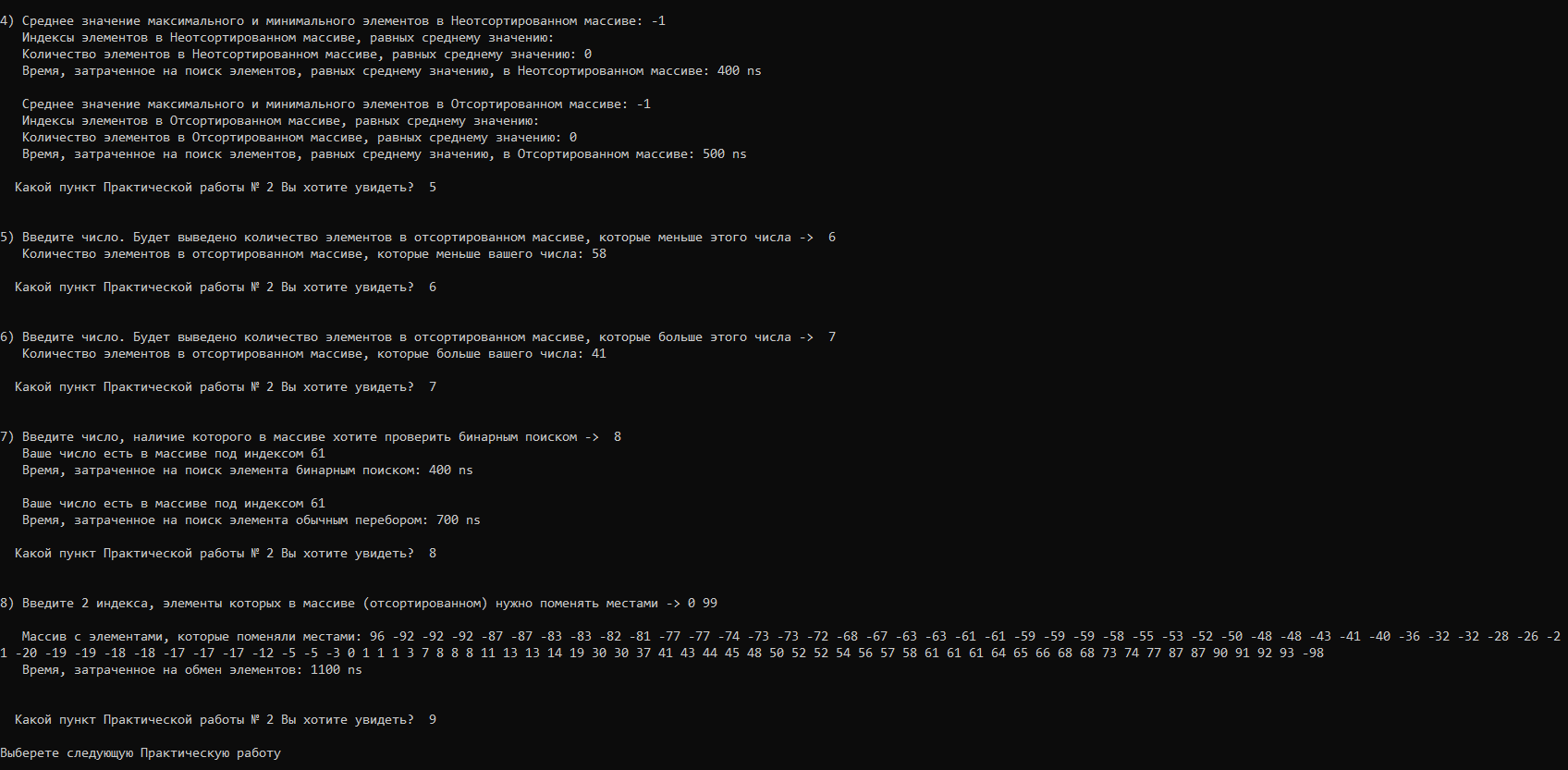
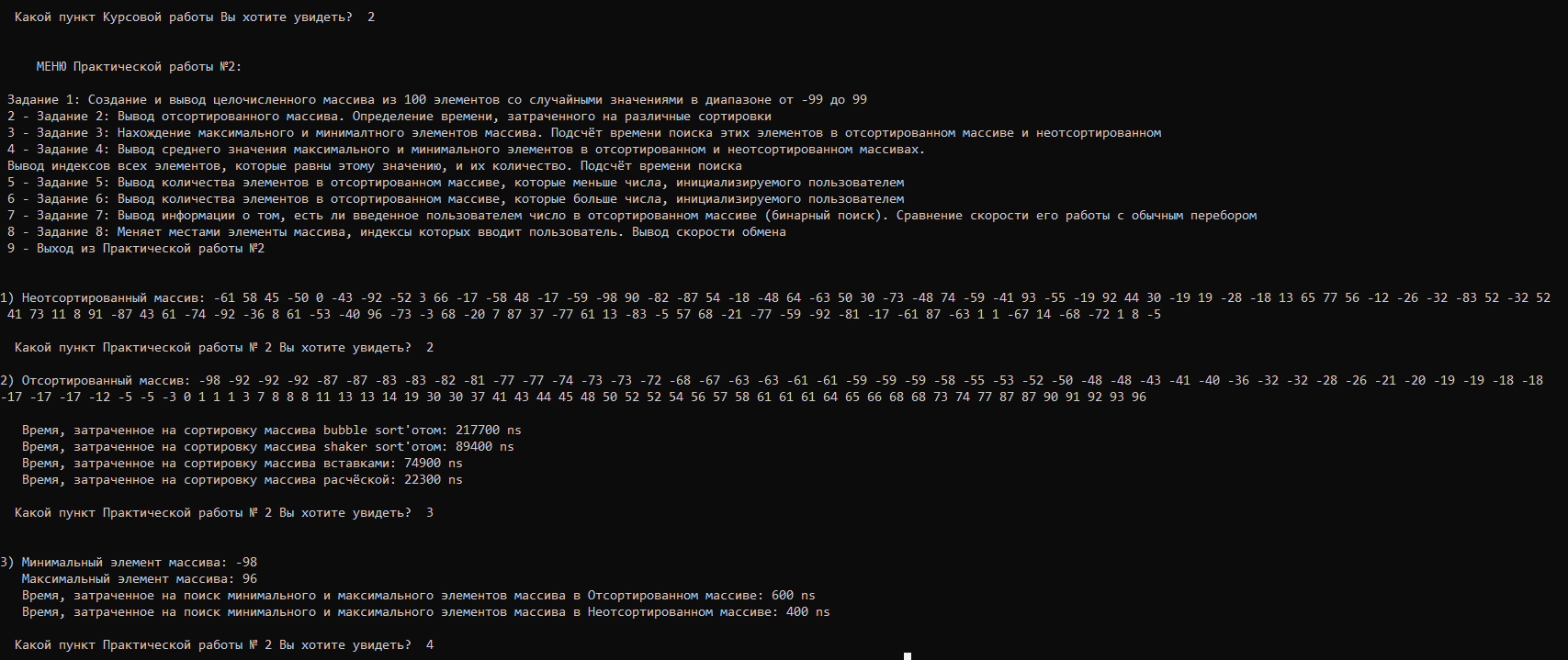
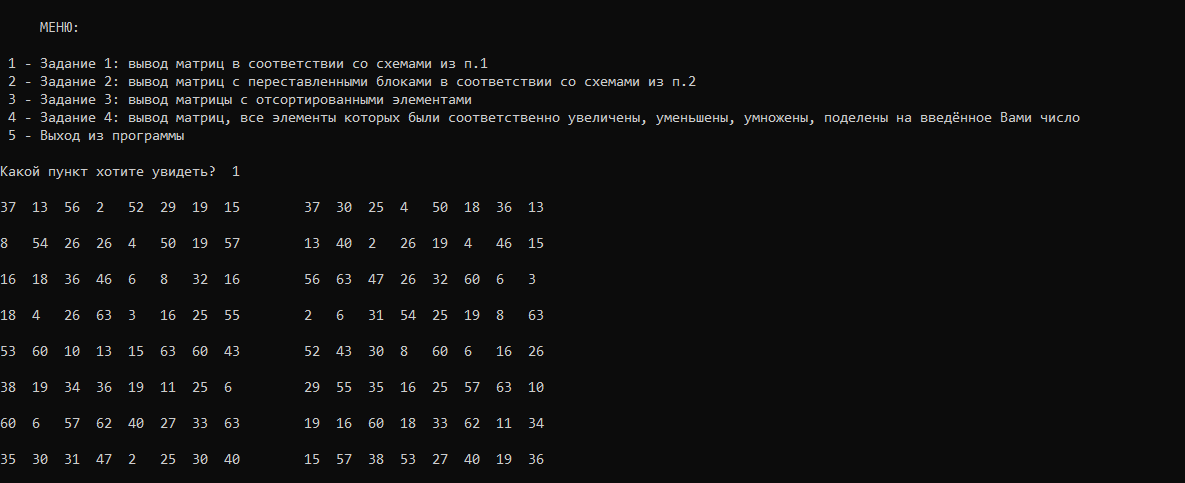


Рисунок 3. Практическая работа №2. Интерактивный выбор любого пункта согласно меню



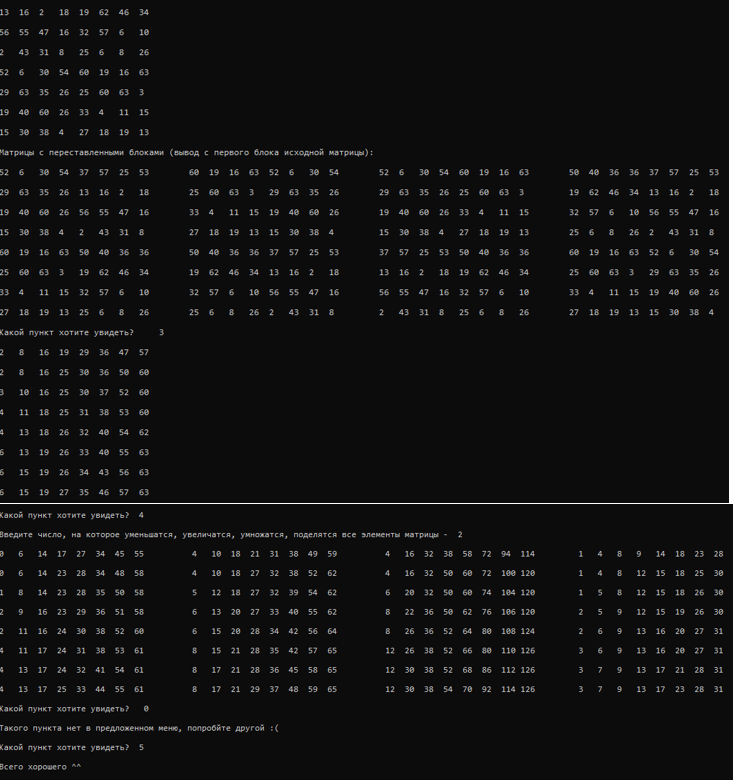


Рисунок 4. Практическая работа №3. Интерактивный выбор любого пункта согласно меню



Рисунок 5. Практическая работа №4. Интерактивный выбор любого пункта согласно меню

“0”, от “6” до любого другого числа, то пользователю предлагается ввести другое число, так как команды с таким номером нет.

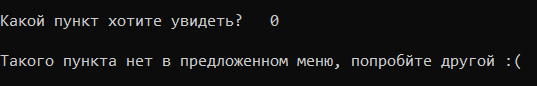


Рисунок 6 – Вывод пункта, не предложенного в меню

“5”, то происходит выход из Курсовой работы.

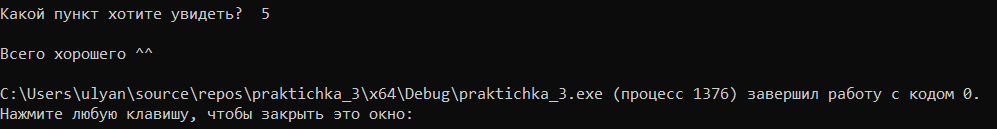


Рисунок 7 – Выход из программы

**Выводы.**

В ходе работы были объединены 4 практические работы. Добавлена инфраструктура переключения между практическими работами, заданиями в них (интерактивное меню).

Приложение А

рабочий код

#include <iostream>

#include <string>

#include <algorithm>

#include <chrono>

#include <ctime>

#include <Windows.h>

#include <algorithm>

using namespace std;

using namespace chrono;

void matrix(int arr[], int n, int& elemMass, int coordY)

{

int coord = coordY;

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0; i < (n \* 4); i += 4)

{

for (int j = coord; j < (n \* 2) + coord; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(70);

}

}

}

void block(int arr[], int n, int& elemMass, int coordY, int coordX)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + coordX; i < ((n / 2) \* 4) + coordX; i += 4)

{

for (int j = coordY; j < ((n / 2) \* 2) + coordY; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(70);

}

elemMass += (n / 2);

}

}

void spiral(int arr[], int n, int koffX, int koffY, int& elemMass, int coordY)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + koffX; i < (n \* 4); i += 4)

{

destCoord.X = i;

destCoord.Y = coordY + 0 + koffY;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

for (int i = coordY + 2 + koffY; i < (n \* 2) + coordY; i += 2)

{

destCoord.X = n \* 4 - 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

for (int i = (n \* 4 - 4 \* 2); i > (-1 + koffX); i -= 4)

{

destCoord.X = i;

destCoord.Y = coordY + n \* 2 - 2;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

for (int i = (coordY + n \* 2 - 2 \* 2); i > (1 + koffY) + coordY; i -= 2)

{

destCoord.X = koffX;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(90);

}

while (n > (n / 2))

return spiral(arr, n - 1, koffX + 4, koffY + 2, elemMass, coordY);

}

void snake(int arr[], int n, int koffX, int koffY, int& t, int coordY)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0 + coordY; i < (n \* 2) + coordY; i += 2)

{

destCoord.X = (n \* 4 + 6) + koffX \* 2;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + t);

t += 1;

cout.flush();

Sleep(90);

}

for (int i = (n \* 2) - 2 + coordY; i > (-1) + coordY; i -= 2)

{

destCoord.X = ((n \* 4 + 6) + 4) + koffX \* 2;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + t);

t += 1;

cout.flush();

Sleep(90);

}

while (t < (n \* n))

return snake(arr, n, koffX + 4, koffY + 2, t, coordY);

}

void blocks\_4(int arr[], int n, int coordY, int coordX)

{

int elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = (n \* n) / 2;

block(arr, n, elemMass, coordY, coordX);

coordY += (n / 2) \* 2;

elemMass = (n \* n) / 2 + 4;

block(arr, n, elemMass, coordY, coordX);

coordX += (n / 2) \* 4;

elemMass = n / 2;

block(arr, n, elemMass, coordY, coordX);

}

void matrix4(int arr[], int n, int& elemMass, int coordY, int coordX)

{

int\* p = arr;

HANDLE hStdout;

COORD destCoord{};

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

elemMass = 0;

for (int i = 0 + coordX; i < (n \* 4) + coordX; i += 4)

{

for (int j = coordY; j < (n \* 2) + coordY; j += 2)

{

destCoord.X = i;

destCoord.Y = j;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(p + elemMass);

elemMass += 1;

cout.flush();

Sleep(70);

}

}

}

void blocks\_1(int arr[], int n, int coordY, int coordX)

{

int elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordY += (n / 2) \* 2;

elemMass = (n \* n) / 2;

block(arr, n, elemMass, coordY, coordX);

coordX = 0;

elemMass = (n \* n) / 2 + 4;

block(arr, n, elemMass, coordY, coordX);

elemMass = n / 2;

coordY -= (n / 2) \* 2;

block(arr, n, elemMass, coordY, coordX);

}

void blocks\_2(int arr[], int n, int coordY, int coordX)

{

int elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = (n \* n) / 2;

block(arr, n, elemMass, coordY, coordX);

elemMass = (n \* n) / 2 + 4;

coordY -= (n / 2) \* 2;

block(arr, n, elemMass, coordY, coordX);

elemMass = n / 2;

coordX += (n / 2) \* 4;

block(arr, n, elemMass, coordY, coordX);

}

void sort(int arr[], int n)

{

int\* p = arr;

for (int i = 1; i < (n \* n); i++)

for (int j = i; \*(p + j - 1) > \*(p + j); j--)

swap(\*(p + j - 1), \*(p + j));

}

void Blinking(int off = 99, bool mood = 0)

{

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

CONSOLE\_CURSOR\_INFO cci{};

cci.dwSize = off;

cci.bVisible = mood;

SetConsoleCursorInfo(hStdout, &cci);

}

void blocks\_3(int arr[], int n, int coordY, int coordX)

{

int elemMass = 0;

block(arr, n, elemMass, coordY, coordX);

elemMass = (n \* n) / 2;

coordX += (n / 2) \* 4;

block(arr, n, elemMass, coordY, coordX);

coordY -= (n / 2) \* 2;

elemMass = (n \* n) / 2 + 4;

block(arr, n, elemMass, coordY, coordX);

coordX -= (n / 2) \* 4;

elemMass = n / 2;

block(arr, n, elemMass, coordY, coordX);

}

void delete\_simvols(string& text, char simvol, int& n)

{

n = text.length();

int index = text.find(simvol);

if (index > -1)

{

for (int i = index; i < n; i++)

{

int count = 0;

if (text[i] == simvol)

{

while (text[i + count] == simvol)

{

++count;

}

if (count > 1)

{

n -= count - 1;

text = text.erase(i + 1, count - 1);

}

}

}

}

}

void checking\_numbers(string text, string clearWords, int& i, bool flag\_number)

{

int n = text.length();

while (((text[i] != ' ') && (text[i] != ',') && (text[i] != '!') && (text[i] != '?') && (text[i] != ';') && (text[i] != ':') && (text[i] != ')') && (text[i] != '.') && (text[i] != '-') && (text[i] != '/')) && (i < n))

{

if (text[i] >= 48 && text[i] <= 57)

flag\_number = true;

clearWords += text[i];

++i;

}

if ((flag\_number == false) && (clearWords != ""))

cout << clearWords << ' ';

}

void line\_search(string text, string sub)

{

int count\_sub = 0;

cout << "\nИндексы: ";

string s1 = "";

for (int i = 0; i <= text.length() - sub.length(); i++)

{

s1 = "";

for (int j = i; j < sub.length() + i; j++)

{

if (text[j] == sub[j - i])

s1 += text[j];

else

continue;

}

if (sub == s1)

{

cout << i + 1 << " ";

count\_sub += 1;

}

}

cout << "\n\nНайдено: " << count\_sub << " подстроки\n";

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time\_t(NULL));

bool flag = true;

bool flag\_kur = true;

int point = 0;

int laba = 0;

cout << "\n МЕНЮ Курсовой работы:\n\n 1 - Практическая работа №1\n" \

" 2 - Практическая работа №2\n" \

" 3 - Практическая работа №3\n" \

" 4 - Практическая работа №4\n" \

" 5 - Выход из Курсовой работы\n\n";

while (flag\_kur)

{

cout << "\n Какой пункт Курсовой работы Вы хотите увидеть? ";

cin >> point;

switch (point)

{

case 1: //ПР №1

{

flag = true;

cout << "\n\n МЕНЮ Практической работы №1:\n\n 1 - Задание 1: Вывод количества памяти (в байтах), отводимого на компьютере под различные типы данных со спецификаторами и без\n" \

" 2 - Задание 2: Вывод на экран двоичного представления в памяти (все разряды) целого числа\n" \

" 3 - Задание 3: Вывод на экран двоичного представления в памяти (все разряды) числа типа float\n" \

" 4 - Выход из Практической работы №1\n";

while (flag)

{

int point = 0;

cout << "\n Какой пункт Практической работы №1 Вы хотите увидеть? ";

cin >> point;

switch (point)

{

case 1:

{

cout << "\nCколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без: \n\n";

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

<< "long int: " << sizeof(long int) << endl << "float: " << sizeof(float) << endl

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

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

break;

}

case 2:

{

int number = 0;

cout << "\nВведите целое число, двоичное представление в памяти (все разряды) котрого хотите увидеть: ";

cin >> number;

cout << endl;

int cdvig = sizeof(number) \* 8 - 1;

unsigned maska = 1 << cdvig;

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

{

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

{

putchar(' ');

}

putchar(number & maska ? '1' : '0');

maska = maska >> 1;

}

cout << endl;

break;

}

case 3:

{

cout << "\nВведите число типа float, двоичное представление в памяти (все разряды) котрого хотите получить: ";

union

{

int number\_int;

float number\_float;

};

cin >> number\_float;

cout << endl;

int bit = sizeof(number\_float) \* 8 - 1;

int maskaa = 1 << bit;

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

{

if ((i == 1) || ((i % 23) == 0))

{

putchar(' ');

}

putchar(number\_int & maskaa ? '1' : '0');

number\_int = number\_int << 1;

}

cout << endl;

break;

}

case 4:

{

cout << "\n Выберете следующую Практическую работу\n";

flag = false;

break;

}

default:

{

cout << "\nТакого пункта нет в предложенном меню, попробйте другой :(\n\n";

break;

}

}

}

break;

}

case 2: //ПР №2

{

srand(time\_t(NULL));

const int n = 100;

int arr[n]{};

int copyArr[n] = { };

int point(0);

flag = true;

time\_point<steady\_clock, duration<\_\_int64, ratio<1, 1000000000>>> start, end;

nanoseconds result;

cout << "\n\n МЕНЮ Практической работы №2:\n\n Задание 1: Создание и вывод целочисленного массива из 100 элементов со случайными значениями в диапазоне от -99 до 99\n" \

" 2 - Задание 2: Вывод отсортированного массива. Определение времени, затраченного на различные сортировки\n" \

" 3 - Задание 3: Нахождение максимального и минималтного элементов массива. Подсчёт времени поиска этих элементов в отсортированном массиве и неотсортированном\n" \

" 4 - Задание 4: Вывод среднего значения максимального и минимального элементов в отсортированном и неотсортированном массивах.\n Вывод индексов всех элементов, которые равны этому значению, и их количество. Подсчёт времени поиска \n" \

" 5 - Задание 5: Вывод количества элементов в отсортированном массиве, которые меньше числа, инициализируемого пользователем\n" \

" 6 - Задание 6: Вывод количества элементов в отсортированном массиве, которые больше числа, инициализируемого пользователем\n" \

" 7 - Задание 7: Вывод информации о том, есть ли введенное пользователем число в отсортированном массиве (бинарный поиск). Сравнение скорости его работы с обычным перебором\n" \

" 8 - Задание 8: Меняет местами элементы массива, индексы которых вводит пользователь. Вывод скорости обмена\n" \

" 9 - Выход из Практической работы №2\n\n";

cout << "\n1) Неотсортированный массив: ";

for (int v = 0; v < n; v++)

{

arr[v] = rand() % 199 - 99;

cout << arr[v] << ' ';

}

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

copyArr[i] = arr[i];

while (flag)

{

cout << "\n\n Какой пункт Практической работы № 2 Вы хотите увидеть? ";

cin >> laba;

switch (laba)

{

case 2: {

start = steady\_clock::now();

for (int i = 0; i < (n - 1); i++) // bubble sort

{

for (int j = 0; j < (n - 1 - i); j++)

{

if (copyArr[j] > copyArr[j + 1])

swap(copyArr[j], copyArr[j + 1]);

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n2) Отсортированный массив: ";

for (int w = 0; w < n; w++)

cout << copyArr[w] << ' ';

cout << "\n\n Время, затраченное на сортировку массива bubble sort'отом: " << result.count() << " ns";

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

copyArr[i] = arr[i];

start = steady\_clock::now();

for (int i = 0; i < n - 1; i++) // shaker sort

{

for (int j = i; j < n - 1 - i; j++)

{

if (copyArr[j] > copyArr[j + 1])

swap(copyArr[j], copyArr[j + 1]);

}

for (int l = n - 2 - i; l > i; l--)

{

if (copyArr[l] < copyArr[l - 1])

swap(copyArr[l], copyArr[l - 1]);

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Время, затраченное на сортировку массива shaker sort'отом: " << result.count() << " ns";

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

copyArr[i] = arr[i];

start = steady\_clock::now();

for (int i = 1; i < n; i++) // insert sort

for (int j = i; copyArr[j - 1] > copyArr[j]; j--) {

swap(copyArr[j - 1], copyArr[j]);

if (j == 1) break;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Время, затраченное на сортировку массива вставками: " << result.count() << " ns";

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

copyArr[i] = arr[i];

double k = 1.247;

int step = n - 1;

bool flag = true;

start = steady\_clock::now();

while (step >= 1) // comb sort

{

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

{

if (copyArr[i] > copyArr[int(i + step)])

swap(copyArr[i], copyArr[int(i + step)]);

}

step = int(step / k);

}

while (flag)

{

for (int i = 0; i < n - 1; i++)

{

flag = false;

for (int j = 0; j < n - 1 - i; j++)

{

if (copyArr[j] > copyArr[j + 1])

{

swap(copyArr[j], copyArr[j + 1]);

flag = true;

}

}

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Время, затраченное на сортировку массива расчёской: " << result.count() << " ns";

break;

}

case 3: {

start = steady\_clock::now();

int minOm = copyArr[0];

int maxOm = copyArr[n - 1];

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n\n3) Минимальный элемент массива: " << minOm;

cout << "\n Максимальный элемент массива: " << maxOm;

cout << "\n Время, затраченное на поиск минимального и максимального элементов массива в Oтсортированном массиве: " << result.count() << " ns";

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

copyArr[i] = arr[i];

int maxNm = -100;

int minNm = 100;

start = steady\_clock::now();

for (int l = 0; l < n; l++)

{

if (copyArr[l] > maxNm)

maxNm = copyArr[l];

if (copyArr[l] < minNm)

minNm = copyArr[l];

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Время, затраченное на поиск минимального и максимального элементов массива в Неотсортированном массиве: " << result.count() << " ns";

break;

}

case 4:

{

int midNm, count = 0;

int maxNm = -100;

int minNm = 100;

for (int l = 0; l < n; l++)

{

if (copyArr[l] > maxNm)

maxNm = copyArr[l];

if (copyArr[l] < minNm)

minNm = copyArr[l];

}

midNm = (minNm + maxNm) / 2;

cout << "\n\n4) Среднее значение максимального и минимального элементов в Неотсортированном массиве: " << midNm;

cout << "\n Индексы элементов в Неотсортированном массиве, равных среднему значению: ";

start = steady\_clock::now();

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

{

if (copyArr[i] == midNm)

{

count += 1;

cout << i << " ";

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Количество элементов в Неотсортированном массиве, равных среднему значению: " << count;

cout << "\n Время, затраченное на поиск элементов, равных среднему значению, в Неотсортированном массиве: " << result.count() << " ns";

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - 1 - i; j++)

{

if (copyArr[j] > copyArr[j + 1])

swap(copyArr[j], copyArr[j + 1]);

}

}

int midOm;

count = 0;

int in = 100;

midOm = (copyArr[0] + copyArr[n - 1]) / 2;

cout << "\n\n Среднее значение максимального и минимального элементов в Oтсортированном массиве: " << midOm;

cout << "\n Индексы элементов в Oтсортированном массиве, равных среднему значению: ";

for (int u = 0; u < n; u++)

if (copyArr[u] > midOm)

in = u - 1;

start = steady\_clock::now();

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

{

while (copyArr[i] <= midOm)

{

if (copyArr[i] == midOm)

{

count += 1;

cout << i << " ";

}

break;

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Количество элементов в Oтсортированном массиве, равных среднему значению: " << count;

cout << "\n Время, затраченное на поиск элементов, равных среднему значению, в Oтсортированном массиве: " << result.count() << " ns";

break;

}

case 5: {

int a;

int count = 0;

cout << "\n\n5) Введите число. Будет выведено количество элементов в отсортированном массиве, которые меньше этого числа -> ";

cin >> a;

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

{

if (copyArr[i] < a)

count += 1;

if (copyArr[i] > a)

break;

}

cout << " Kоличество элементов в отсортированном массиве, которые меньше вашего числа: " << count;

break;

}

case 6: {

int b;

int count = 0;

cout << "\n\n6) Введите число. Будет выведено количество элементов в отсортированном массиве, которые больше этого числа -> ";

cin >> b;

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

{

if (copyArr[i] > b)

count += 1;

}

cout << " Kоличество элементов в отсортированном массиве, которые больше вашего числа: " << count;

break;

}

case 7: {

int key;

cout << "\n\n7) Введите число, наличие которого в массиве хотите проверить бинарным поиском -> ";

cin >> key;

int midIndex;

int left = 0;

int right = n - 1;

bool flagBin = false;

start = steady\_clock::now();

while (left <= right) // бинарный поиск

{

midIndex = (left + right) / 2;

if (copyArr[midIndex] == key)

{

flagBin = true;

break;

}

if (copyArr[midIndex] > key)

right = midIndex - 1;

else

left = midIndex + 1;

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

if (flagBin == true)

cout << " Ваше число есть в массиве под индексом " << midIndex;

else

cout << " Вашего числа нет в массиве :(";

cout << "\n Время, затраченное на поиск элемента бинарным поиском: " << result.count() << " ns";

int countI = 0;

start = steady\_clock::now();

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

{

if (copyArr[i] == key)

{

countI += 1;

break;

}

}

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

if (countI > 0)

cout << "\n\n Ваше число есть в массиве под индексом " << midIndex;

else

cout << "\n Вашего числа нет в массиве :(";

cout << "\n Время, затраченное на поиск элемента обычным перебором: " << result.count() << " ns";

break;

}

case 8: {

int el1, el2;

cout << "\n\n8) Введите 2 индекса, элементы которых в массиве (отсортированном) нужно поменять местами -> ";

cin >> el1;

cin >> el2;

start = steady\_clock::now();

swap(copyArr[el1], copyArr[el2]);

end = steady\_clock::now();

result = duration\_cast<nanoseconds>(end - start);

cout << "\n Массив с элементами, которые поменяли местами: ";

for (int w = 0; w < n; w++)

cout << copyArr[w] << ' ';

cout << "\n Время, затраченное на обмен элементов: " << result.count() << " ns\n";

break;

}

case 9: {

cout << "\nВыберете следующую Практическую работу\n";

flag = false;

break;

}

default:

{

cout << "\nТакого пункта нет в предложенном меню, попробуйте другой :(\n\n";

break;

}

}

}

break;

}

case 3: // ПР №3

{

const int n = 8;

int arr[n \* n]{};

int koffX = 0, koffY = 0;

int elemMass = 0, t = 0, point, coordY = 0, coordX = 0;

flag = true;

for (int v = 0; v < (n \* n); v++)

arr[v] = rand() % (n \* n) + 1;

cout << "\n МЕНЮ:\n\n 1 - Задание 1: вывод матриц в соответствии со схемами из п.1\n" \

" 2 - Задание 2: вывод матриц с переставленными блоками в соответствии со схемами из п.2\n" \

" 3 - Задание 3: вывод матрицы с отсортированными элементами\n" \

" 4 - Задание 4: вывод матриц, все элементы которых были соответственно увеличены, уменьшены, умножены, поделены на введённое Вами число \n" \

" 5 - Выход из Практической работы №3\n\n";

while (flag)

{

Blinking(99, 1);

cout << "Какой пункт Практической работы №3 Вы хотите увидеть? ";

cin >> point;

Blinking();

switch (point)

{

case 1:

{

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

elemMass = 0;

t = 0;

spiral(arr, n, koffX, koffY, elemMass, coordY);

snake(arr, n, koffX, koffY, t, coordY);

for (int i = 0; i < (n \* 2); i++)

cout << "\n";

break;

}

case 2:

{

cout << "\nИсходная матрица: \n";

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

elemMass = 0;

matrix(arr, n, elemMass, coordY);

cout << "\n\nМатрицы с переставленными блоками (вывод с первого блока исходной матрицы): \n";

CONSOLE\_SCREEN\_BUFFER\_INFO fi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &fi);

coordY = fi.dwCursorPosition.Y + 1;

coordX = (n / 2) \* 4;

blocks\_1(arr, n, coordY, coordX);

coordX = (n \* 4) + 6 + (n / 2) \* 4;

coordY += (n / 2) \* 2;

blocks\_2(arr, n, coordY, coordX);

coordX = (n \* 4) \* 2 + 6 \* 2;

blocks\_3(arr, n, coordY, coordX);

coordX = (n \* 4) \* 3 + 6 \* 3 + (n / 2) \* 4;

coordY -= (n / 2) \* 2;

blocks\_4(arr, n, coordY, coordX);

cout << "\n\n";

break;

}

case 3:

{

elemMass = 0;

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

sort(arr, n);

matrix(arr, n, elemMass, coordY);

cout << "\n\n";

break;

}

case 4:

{

int num, coordX = 0;

int copyArr[n \* n]{};

elemMass = 0;

cout << "\nВведите число, на которое уменьшатся, увеличатся, умножатся, поделятся все элементы матрицы - ";

cin >> num;

CONSOLE\_SCREEN\_BUFFER\_INFO bi;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &bi);

coordY = bi.dwCursorPosition.Y + 1;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] - num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX = n \* 4 + 8;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] + num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX += n \* 4 + 8;

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] \* num;

matrix4(copyArr, n, elemMass, coordY, coordX);

coordX += n \* 4 + 8;

if (num != 0)

{

for (int i = 0; i < (n \* n); i++)

copyArr[i] = arr[i] / num;

matrix4(copyArr, n, elemMass, coordY, coordX);

}

cout << "\n\n";

break;

}

case 5:

{

cout << "\nВыберете следующую Практическую работу\n";

flag = false;

break;

}

default:

{

cout << "\nТакого пункта нет в предложенном меню, попробйте другой :(\n\n";

break;

}

}

}

break;

}

case 4: //ПР №4

{

cout << "\n\n МЕНЮ:\n\n Задание 1: Ввод последовательности с клавиатуры\n" \

" 2 - Задание 2: Вывод отредактированного текста (удаление лишних пробелов, знаков препинания, исправление регистра букв\n" \

" 3 - Задание 3: Вывод на экран слов последовательности, не содержащих цифр\n" \

" 4 - Задание 4: Вывод на экран этой же последовательности, при этом удалив из всех слов заданный пользователем набор букв и (или) цифр\n" \

" 5 - Задание 5: Линейный поиск подстроки\n" \

" 6 - Выход из Практической работы №4\n\n";

cout << "Задание 1: Введите последовательность, пожалуйста - ";

string text{};

cin.ignore();

getline(cin, text);

flag = true;

while (flag)

{

cout << "\n Какой пункт Практической работы №4 Вы хотите увидеть? ";

cin >> laba;

switch (laba)

{

case 2:

{

int n = text.length();

string simvols = " ,/)(!;:?\_";

for (int i = 0; i < simvols.length(); i++)

{

char simvol = simvols[i];

delete\_simvols(text, simvol, n);

}

char simvol = '.';

int index = text.find(simvol);

for (int i = index; i < n; i++)

{

int count\_points = 0;

int count\_threetochie = 0;

while (text[i + count\_points] == simvol)

++count\_points;

count\_threetochie = count\_points / 3;

if (count\_threetochie != 0)

{

text = text.erase(i + 3, count\_points - 3);

n -= count\_points - 3;

i += 3;

}

if (count\_points == 2)

{

text = text.erase(i + 1, 1);

n -= 1;

}

}

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

text[i] = tolower(text[i]);

cout << "\nОтредактированный текст: " << text;

cout << endl;

break;

}

case 3:

{

int n = text.length();

cout << "\nИсходный текст: " << text;

cout << "\n\nСлова из последовательности без цифр: ";

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

{

string clearWords{};

bool flag\_number = false;

if (i == 0)

checking\_numbers(text, clearWords, i, flag\_number);

if (text[i] == ' ')

{

++i;

checking\_numbers(text, clearWords, i, flag\_number);

--i;

}

}

cout << endl;

break;

}

case 4:

{

int n = text.length();

string mini\_text = "";

cout << "\nВведите набор букв и (или) цифр, который необходимо удалить из последовательности: ";

cin >> mini\_text;

int index = -1;

while ((index = text.find(mini\_text)) != -1)

{

text = text.erase(index, mini\_text.length());

n -= mini\_text.length();

}

n = text.length();

string simvols = " ,/)(!;:?";

for (int i = 0; i < simvols.length(); i++)

{

char simvol = simvols[i];

delete\_simvols(text, simvol, n);

}

cout << "\nПоследовательность без заданной подстроки: " << text;

cout << endl;

break;

}

case 5:

{

string sub{};

cout << "\nВведите подстроку, которую хотите найти: ";

cin >> sub;

line\_search(text, sub);

break;

}

case 6:

{

cout << "\nВыберете следующую Практическую работу\n";

flag = false;

break;

}

default:

{

cout << "\nТакого пункта нет в предложенном меню, попробуйте другой :(\n";

break;

}

}

}

break;

}

case 5:

{

cout << "\nВсего хорошего ^^\n";

flag\_kur = false;

break;

}

default:

{

cout << "\nТакого пункта нет в предложенном меню, попробуйте другой :(\n";

break;

}

}

}

return 0;

}