**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. | Минин М. А. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Объединить все 4 лабораторные работы в единый проект.

**Основные теоретические положения.**

В зависимости от используемого компилятора встроенные типы данных могут занимать разное количество байт в памяти. Чтобы узнать сколько байт весит тот или иной тип данных можно использовать встроенную функцию sizeof(type\_name), которая возвращает размер типа. Для побитовых операций есть специальные операторы (>>, <<, |, &, ^), которые позволяют делать некие поразрядные операции. Ключевое слово union позволяет создавать объединения объектов, после которого они будут занимать одну область в памяти.

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.

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

Сортировка – процесс размещения элементов заданного множества объектов в определенном порядке. Когда элементы отсортированы, их проще найти, производить с ними различные операции. Сортировка напрямую влияет на скорость алгоритма, в котором нужно обратиться к определенному элементу массива.

Простейшая из сортировок – сортировка обменом (пузырьковая сортировка). 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.

Быстрая сортировка (quick sort) – одна из самых быстрых сортировок. Эта сортировка по сути является существенно улучшенной версией алгоритма пузырьковой сортировки.

Общая идея алгоритма состоит в том, что сначала выбирается из массива элемент, который называется опорным. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность. Затем необходимо сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующие друг за другом: меньше опорного, раны опорному и больше опорного. Для меньших и больших значений необходимо выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

Алгоритм бинарного поиска – классический алгоритм поиска 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.

Все массивы можно разделить на две группы: одномерные и многомерные. 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.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Так же, как и в одномерном массиве, элементы многомерных массивов располагаются друг за другом в непрерывном участке памяти.

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.

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 целыезначения 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.

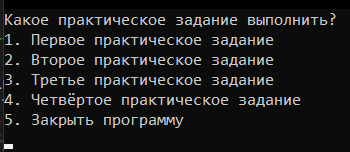
1. При запуске программы пользователю выводится меню доступных команд и ожидается ввод команды с клавиатуры.

Рис. 1 – Меню программы

1. Практическая работа №1

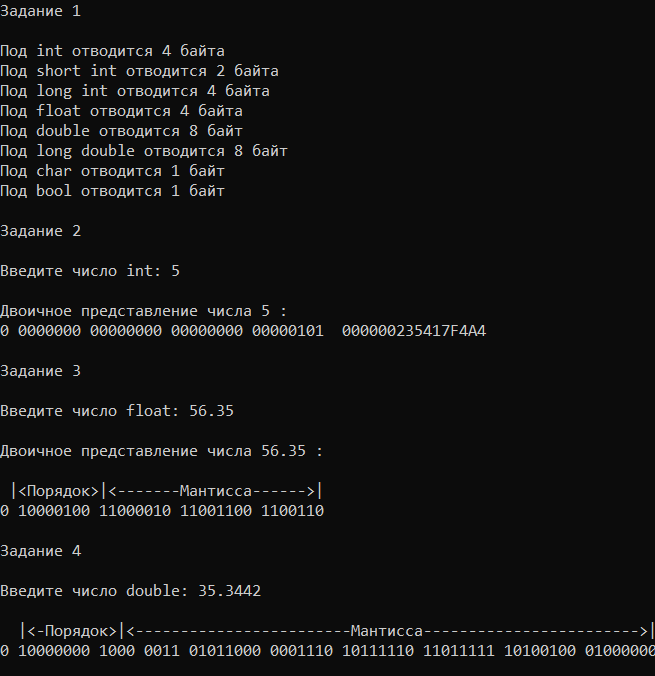


Рис. 2 – Практическая работа №1

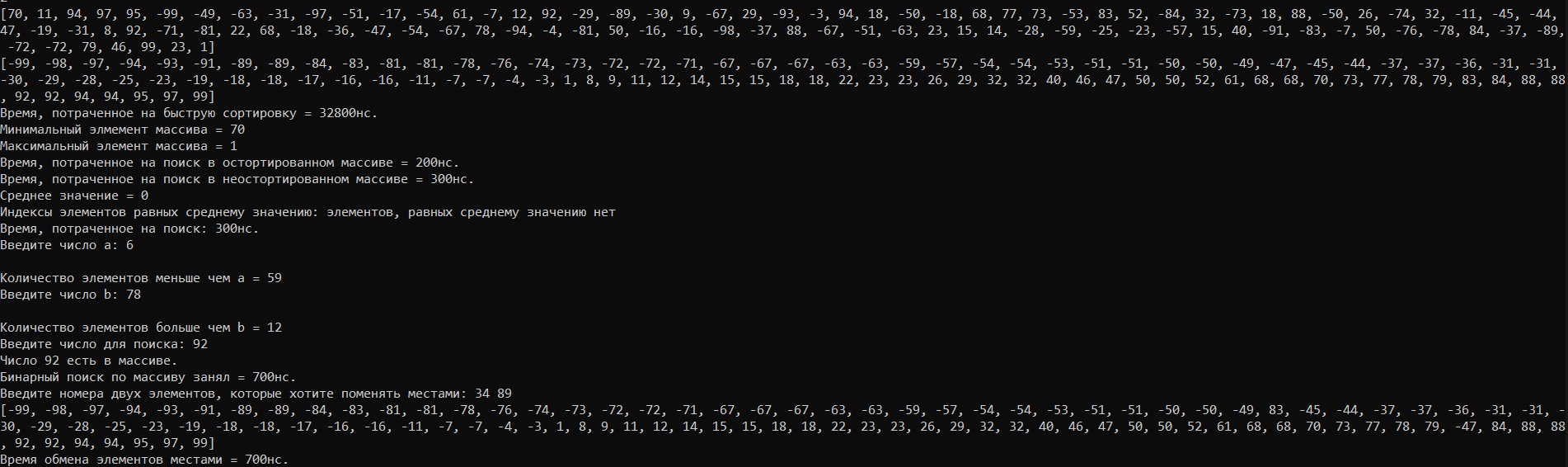
1. Практическая работа №2

Рис. 3 – Практическая работа №2

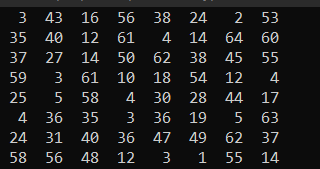
1. Практическая работа №3

Рис. 4 – Практическая работа №3

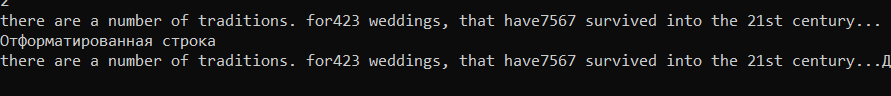
1. Практическая работа №4

Рис. 5 – Практическая работа №4

**Вывод.**

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 <iomanip>

#include <windows.h>

#include <ctime>

#include <string>

#include <chrono>

#include <fstream>

using namespace std;

void Task1() {

cout << "Задание 1 \n\n";

cout << "Под int отводится " \

<< sizeof(int) << " байта \n";

cout << "Под short int отводится " \

<< sizeof(short int) << " байта \n";

cout << "Под long int отводится " \

<< sizeof(long int) << " байта \n";

cout << "Под float отводится " \

<< sizeof(float) << " байта \n";

cout << "Под double отводится " \

<< sizeof(double) << " байт \n";

cout << "Под long double отводится " \

<< sizeof(long double) << " байт \n";

cout << "Под char отводится " \

<< sizeof(char) << " байт \n";

cout << "Под bool отводится " \

<< sizeof(bool) << " байт \n\n";

int chislo, i;

unsigned int kol\_razryadov = sizeof(chislo) \* 8;

unsigned int maska = 1 << (kol\_razryadov - 1);

cout << "Задание 2 \n\n";

cout << "Введите число int: ";

cin >> chislo;

cout << "\n" << "Двоичное представление числа " << chislo << " : \n";

for (i = kol\_razryadov - 1; i >= 0; i -= 1)

{

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

maska >>= 1;

if (i % 8 == 0)

putchar(' ');

if (i % (kol\_razryadov - 1) == 0)

putchar(' ');

}

cout << &chislo;

cout << "\n\n" << "Задание 3" << "\n\n";

cout << "Введите число float: ";

union

{

float chislo\_float;

int chislo\_3;

};

cin >> chislo\_float;

cout << "\n" << "Двоичное представление числа " << chislo\_float << " : \n";

kol\_razryadov = sizeof(chislo\_float) \* 8; // Обновляем количество разрядов

maska = 1 << kol\_razryadov - 1; // Обновляем маску

cout << "\n" << " |<Порядок>|<-------Мантисса------>|\n";

for (i = 0; i < kol\_razryadov; i += 1)

{

putchar(chislo\_3 & maska ? '1' : '0');

chislo\_3 <<= 1;

if (i % 8 == 0)

putchar(' ');

}

cout << "\n\n" << "Задание 4" << "\n\n";

cout << "Введите число double: ";

union {

double chislo\_double;

int chislo\_4[2];

};

cin >> chislo\_double;

kol\_razryadov = sizeof(double) \* 8; //Обновляем количество разрядов для работы

maska = 1 << kol\_razryadov - 1; // Обновялем маску

cout << "\n" << " |<-Порядок>|<------------------------Мантисса------------------------>|\n";

for (int i = 0; i < kol\_razryadov / 2; i++)

{

putchar(chislo\_4[1] & maska ? '1' : '0');

chislo\_4[1] <<= 1;

if (i % 8 == 0 || i == 12)

putchar(' ');

} // Работает с первыми 4 битами

for (int i = 0; i < kol\_razryadov / 2; i++) {

if (i % 8 == 0)

putchar(' ');

putchar(chislo\_4[0] & maska ? '1' : '0');

chislo\_4[0] <<= 1;

} // Работает со вторыми 4 битами

}

//-----------------------------------------------------------------------------------------------------------------------------------------

void Output(int\* array, int N) {

cout << '[';

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

if (i != N - 1)

cout << array[i] << ", ";

else

cout << array[i];

}

cout << ']' << '\n';

}

void Fill(int N, int\* array, int\* arraySorted) {

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

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

arraySorted[i] = array[i];

}

Output(array, N);

}

void Quicksort(int\* array, int begin, int end) {

int mid;

int start = begin;

int finish = end;

mid = array[(start + finish) / 2];

while (start < finish)

{

while (array[start] < mid) start++;

while (array[finish] > mid) finish--;

if (start <= finish)

{

swap(array[start], array[finish]);

start++;

finish--;

}

}

if (begin < finish) Quicksort(array, begin, finish);

if (start < end) Quicksort(array, start, end);

}

void OutSort(int\* array, int begin, int end, int N) {

auto StartTime = chrono::high\_resolution\_clock::now();

Quicksort(array, begin, end);

auto EndTime = chrono::high\_resolution\_clock::now();

Output(array, N);

cout << "Время, потраченное на быструю сортировку = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime - StartTime).count() << "нс." << '\n';

}

void SearchMinMax(int\* arraySorted, int\* array, int N) {

auto StartTime = chrono::high\_resolution\_clock::now();

int minimal = array[0];

int maximal = array[N - 1];

auto EndTime = chrono::high\_resolution\_clock::now();

cout << "Минимальный элмемент массива = " << minimal << '\n';

cout << "Максимальный элемент массива = " << maximal << '\n';

cout << "Время, потраченное на поиск в остортированном массиве = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime - StartTime).count() << "нс." << '\n';

int min = array[0];

int max = array[0];

auto StartTime1 = chrono::high\_resolution\_clock::now();

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

if (array[i] < min)

min = array[i];

if (array[i] > max)

max = array[i];

}

auto EndTime1 = chrono::high\_resolution\_clock::now();

cout << "Время, потраченное на поиск в неостортированном массиве = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime1 - StartTime1).count() << "нс." << '\n';

}

void SearchMiddle(int\* array, int N) {

int middle = round(array[0] + array[N - 1]);

cout << "Среднее значение = " << middle << '\n';

cout << "Индексы элементов равных среднему значению: ";

int cnt = 0;

auto StartTime = chrono::high\_resolution\_clock::now();

for (int i = 0; ((array[i] <= middle) && (i < N)); i++) {

if (array[i] == middle)

cnt++;

}

auto EndTime = chrono::high\_resolution\_clock::now();

if (cnt == 0)

cout << "элементов, равных среднему значению нет" << '\n';

else {

for (int i = 0; array[i] <= middle; i++) {

if (array[i] == middle)

cout << i << ' ';

}

cout << '\n' << "Количество элементов равных среднему = " << cnt << '\n';

}

cout << "Время, потраченное на поиск: " << chrono::duration\_cast<chrono::nanoseconds>(EndTime - StartTime).count() << "нс." << '\n';

}

void SearchLessA(int\* array, int N) {

int a, cnt = 0;

cout << "Введите число а: ";

cin >> a;

for (int i = 0; ((array[i] < a) && (i < N)); i++) {

cnt++;

}

cout << '\n' << "Количество элементов меньше чем a = " << cnt << '\n';

}

void SearchMoreB(int\* array, int N) {

int b, cnt = 0;

cout << "Введите число b: ";

cin >> b;

for (int i = N - 1; (array[i] > b && i >= 0); i--) {

cnt++;

}

cout << '\n' << "Количество элементов больше чем b = " << cnt << '\n';

}

int BinSearch(int\* array, int left, int right, int N) {

bool flag = false;

int chislo;

cout << "Введите число для поиска: ";

cin >> chislo;

int medium;

auto StartTime = chrono::high\_resolution\_clock::now();

while ((left <= right) && (flag != true)) {

medium = (left + right) / 2;

if (chislo == array[medium]) {

flag = true;

}

if (array[medium] > chislo)

right = medium - 1;

else

left = medium + 1;

}

auto EndTime = chrono::high\_resolution\_clock::now();

if (flag)

cout << "Число " << chislo << " есть в массиве." << '\n';

else

cout << "Числа " << chislo << " нет в массиве." << '\n';

cout << "Бинарный поиск по массиву занял = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime - StartTime).count() << "нс." << '\n';

return N + 1;

flag = false;

auto StartTime1 = chrono::high\_resolution\_clock::now();

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

if (array[i] == chislo)

flag = true;

if (flag)

break;

}

auto EndTime1 = chrono::high\_resolution\_clock::now();

cout << "Поиск перебором занял = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime1 - StartTime1).count() << "нс." << '\n';

}

void ChangeLoc(int\* array, int N) {

int First, Second;

cout << "Введите номера двух элементов, которые хотите поменять местами: ";

cin >> First >> Second;

auto StartTime = chrono::high\_resolution\_clock::now();

if (First >= 0 && Second >= 0 && First < N && Second < N) {

swap(array[First], array[Second]);

auto EndTime = chrono::high\_resolution\_clock::now();

Output(array, N);

cout << "Время обмена элементов местами = " << chrono::duration\_cast<chrono::nanoseconds>(EndTime - StartTime).count() << "нс." << '\n';

}

else {

cout << "Вы неверно ввели номера элементов, попройте снова" << '\n';

return ChangeLoc(array, N);

}

}

void PrintMenu2() {

system("cls");

cout << "Что выхотите сделать?" << '\n';

cout << "1. Выполнить 1 задание" << '\n';

cout << "2. Выполнить 2 задание" << '\n';

cout << "3. Выполнить 3 задание" << '\n';

cout << "4. Выполнить 4 задание" << '\n';

cout << "5. Выполнить 5 задание" << '\n';

cout << "6. Выполнить 6 задание" << '\n';

cout << "7. Выполнить 7 задание" << '\n';

cout << "8. Выполнить 8 задание" << '\n';

cout << "9. Выйти из программы" << '\n';

}

void Menu2(int\* array, int\* arraySorted, int N) {

int Variant;

system("pause");

do {

PrintMenu2();

cin >> Variant;

switch (Variant) {

case 1:

Fill(N, array, arraySorted);

break;

case 2:

Output(array, N);

OutSort(arraySorted, 0, N - 1, N);

break;

case 3:

Output(arraySorted, N);

SearchMinMax(arraySorted, array, N);

break;

case 4:

Output(arraySorted, N);

SearchMiddle(arraySorted, N);

break;

case 5:

Output(arraySorted, N);

SearchLessA(arraySorted, N);

break;

case 6:

Output(arraySorted, N);

SearchMoreB(arraySorted, N);

break;

case 7:

Output(arraySorted, N);

BinSearch(arraySorted, 0, N - 1, N);

break;

case 8:

Output(arraySorted, N);

ChangeLoc(arraySorted, N);

Quicksort(arraySorted, 0, N - 1);

break;

}

if (Variant != 9)

system("pause");

} while (Variant != 9);

}

void Task2() {

const int N = 100;

setlocale(0, "");

int array[N];

int arraySorted[N];

srand(time(0));

Fill(N, array, arraySorted);

OutSort(arraySorted, 0, N - 1, N);

SearchMinMax(arraySorted, array, N);

SearchMiddle(arraySorted, N);

SearchLessA(arraySorted, N);

SearchMoreB(arraySorted, N);

BinSearch(arraySorted, 0, N - 1, N);

ChangeLoc(arraySorted, N);

Quicksort(arraySorted, 0, N - 1);

Menu2(array, arraySorted, N);

}

//------------------------------------------------------------------------------------------------------------------------------------------

void PrintMatrix(int\*\* Matrix, int Size) {

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

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

cout << setw(3) << setprecision(3) << Matrix[i][j];

cout.flush();

if (j != Size)

cout << ' ';

}

cout << "\n";

}

Sleep(50);

}

void EmptyMatrix(int\*\* Matrix, int Size) {

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

for (int j = 0; j < Size; j++)

Matrix[i][j] = 0;

}

void SpiralMatrix(int\*\* Matrix, int Size) {

EmptyMatrix(Matrix, Size);

int LineNumber = 0;

for (int LineSize = 0; LineSize < Size / 2; LineSize++) {

for (int\* i = \*(Matrix + LineNumber) + LineNumber, \*End = i + Size - 2 \* LineNumber; i < End; i++) {

\*i = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

for (int\*\* i = Matrix + LineNumber + 1, \*\*End = Matrix + Size - 2 - LineNumber; i <= End; i++) {

\*(\*i + Size - LineNumber - 1) = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

for (int\* i = \*(Matrix + Size - LineNumber - 1) + Size - LineNumber - 1, \*End = i - Size + 1 + 2 \* LineNumber; i > End; i--) {

\*i = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

for (int\*\* i = Matrix + Size - 1 - LineNumber, \*\*End = Matrix + LineNumber + 1; i >= End; i--) {

\*(\*i + LineNumber) = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

LineNumber++;

}

}

void SnakeMatrix(int\*\* Matrix, int Size) {

EmptyMatrix(Matrix, Size);

for (int LineNumber = 0; LineNumber < Size / 2; LineNumber++) {

for (int\*\* i = Matrix; i <= (Matrix + Size - 1); i++) {

\*(\*i + LineNumber \* 2) = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

for (int\*\* i = Matrix + Size - 1; i >= Matrix; i--) {

\*(\*i + LineNumber \* 2 + 1) = rand() % (Size \* Size) + 1;

system("cls");

PrintMatrix(Matrix, Size);

}

}

}

void RotateMatrix(int\*\* Matrix, int Size) {

int\*\* New\_Matrix = new int\* [Size];

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

New\_Matrix[i] = new int[Size];

char permutation;

cout << "Какую перестановвку выполнить ?(a или b или c или d)" << '\n';

cin >> permutation;

while (permutation != 'a' and permutation != 'b' and permutation != 'c' and permutation != 'd') {

cout << "Неверно введён вид перестановки, попробуйте снова: ";

cin >> permutation;

}

if (permutation == 'a') {

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

for (int j = 0; j < Size / 2; j++) {

New\_Matrix[i][j] = Matrix[i + Size / 2][j];

New\_Matrix[i][j + Size / 2] = Matrix[i][j];

New\_Matrix[i + Size / 2][j + Size / 2] = Matrix[i][j + Size / 2];

New\_Matrix[i + Size / 2][j] = Matrix[i + Size / 2][j + Size / 2];

}

Matrix = New\_Matrix;

PrintMatrix(Matrix, Size);

}

else if (permutation == 'b') {

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

for (int j = 0; j < Size / 2; j++) {

New\_Matrix[i][j] = Matrix[i + Size / 2][j + Size / 2];

New\_Matrix[i][j + Size / 2] = Matrix[i + Size / 2][j];

New\_Matrix[i + Size / 2][j + Size / 2] = Matrix[i][j];

New\_Matrix[i + Size / 2][j] = Matrix[i][j + Size / 2];

}

Matrix = New\_Matrix;

PrintMatrix(Matrix, Size);

}

else if (permutation == 'c') {

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

for (int j = 0; j < Size / 2; j++) {

New\_Matrix[i][j] = Matrix[i + Size / 2][j];

New\_Matrix[i][j + Size / 2] = Matrix[i + Size / 2][j + Size / 2];

New\_Matrix[i + Size / 2][j + Size / 2] = Matrix[i][j + Size / 2];

New\_Matrix[i + Size / 2][j] = Matrix[i][j];

}

Matrix = New\_Matrix;

PrintMatrix(Matrix, Size);

}

else {

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

for (int j = 0; j < Size / 2; j++) {

New\_Matrix[i][j] = Matrix[i][j + Size / 2];

New\_Matrix[i][j + Size / 2] = Matrix[i][j];

New\_Matrix[i + Size / 2][j + Size / 2] = Matrix[i + Size / 2][j];

New\_Matrix[i + Size / 2][j] = Matrix[i + Size / 2][j + Size / 2];

}

Matrix = New\_Matrix;

PrintMatrix(Matrix, Size);

}

}

void SortMatrix(int\*\* Matrix, int Size) {

int ArrSize = Size \* Size;

int\* Array = new int[ArrSize];

int Array\_i = 0;

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

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

Array[Array\_i] = Matrix[i][j];

Array\_i++;

}

}

bool Change = TRUE;

while (Change) {

Change = FALSE;

for (int\* i = Array, \*End\_i = i + ArrSize - 2; i <= End\_i; i++) {

if (\*i > \*(i + 1)) {

Change = TRUE;

swap(\*i, \*(i + 1));

}

}

}

Array\_i = 0;

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

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

Matrix[i][j] = Array[Array\_i];

Array\_i++;

}

}

PrintMatrix(Matrix, Size);

}

void MagnificationMatrix(int\*\* Matrix, int Size) {

int Number;

cout << '\n' << "Введите число" << '\n';

cin >> Number;

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

for (int j = 0; j < Size; j++)

Matrix[i][j] += Number;

PrintMatrix(Matrix, Size);

}

void ReducationMatrix(int\*\* Matrix, int Size) {

int Number;

cout << '\n' << "Введите число" << '\n';

cin >> Number;

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

for (int j = 0; j < Size; j++)

Matrix[i][j] -= Number;

PrintMatrix(Matrix, Size);

}

void MultiplicationMatrix(int\*\* Matrix, int Size) {

int Number;

cout << '\n' << "Введите число" << '\n';

cin >> Number;

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

for (int j = 0; j < Size; j++)

Matrix[i][j] \*= Number;

PrintMatrix(Matrix, Size);

}

void DivisionMatrix(int\*\* Matrix, int Size) {

int Number;

cout << '\n' << "Введите число" << '\n';

cin >> Number;

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

for (int j = 0; j < Size; j++)

Matrix[i][j] /= Number;

PrintMatrix(Matrix, Size);

}

void PrintMenu3() {

system("cls");

cout << "Что выхотите сделать?" << '\n';

cout << "1. Заполнить матрицу новыми числами" << '\n';

cout << "2. Переставить блоки матрицы" << '\n';

cout << "3. Отсортировать матрицу" << '\n';

cout << "4. Увеличить матрицу на число" << '\n';

cout << "5. Уменьшить матрицу на число" << '\n';

cout << "6. Умножить матрицу на число" << '\n';

cout << "7. Поделить матрицу на число" << '\n';

cout << "8. Выйти из программы" << '\n';

}

void Menu3(int\*\* Matrix, int Size) {

int Variant;

system("pause");

do {

PrintMenu3();

cin >> Variant;

switch (Variant) {

case 1:

char perm;

cout << "В каком виде вывести матрицу?(a или b)" << '\n';

cin >> perm;

while (perm != 'a' and perm != 'b') {

cout << "Неверно введён вид вывода, попробуйте снова: ";

cin >> perm;

}

if (perm == 'a')

SpiralMatrix(Matrix, Size);

else

SnakeMatrix(Matrix, Size);

break;

case 2:

PrintMatrix(Matrix, Size);

cout << '\n' << "Смена блоков" << '\n';

RotateMatrix(Matrix, Size);

break;

case 3:

cout << '\n' << "Отсортированный массив" << '\n';

SortMatrix(Matrix, Size);

break;

case 4:

PrintMatrix(Matrix, Size);

MagnificationMatrix(Matrix, Size);

break;

case 5:

PrintMatrix(Matrix, Size);

ReducationMatrix(Matrix, Size);

break;

case 6:

PrintMatrix(Matrix, Size);

MultiplicationMatrix(Matrix, Size);

break;

case 7:

PrintMatrix(Matrix, Size);

DivisionMatrix(Matrix, Size);

break;

}

if (Variant != 8)

system("pause");

} while (Variant != 8);

}

void Task3() {

setlocale(0, "");

srand(time(0));

int Size;

cout << "Введите размер матрицы: 6 или 8 или 10:" << "\n";

cin >> Size;

while (Size != 6 and Size != 8 and Size != 10) {

cout << "Неверно введён размер матрицы, попробуйте снова: ";

cin >> Size;

}

int\*\* Matrix = new int\* [Size];

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

Matrix[i] = new int[Size];

EmptyMatrix(Matrix, Size);

Menu3(Matrix, Size);

}

//------------------------------------------------------------------------------------------------------------------------------------------

void StrInput(string& str) {

ifstream File(R"(C:\Программирование\4\_прак\Test.txt)");

if (!File.is\_open()) {

cout << "Файл не открыт";

}

string str2;

while (!File.eof())

getline(File, str);

cout << "Введённая строка:" << '\n';

cout << str << '\n';

File.close();

}

void DeleteSymbols(string& str, char Symbol) {

int Len = str.length();

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

{

int cnt = 0;

if (str[i] == Symbol)

{

while (str[i + cnt] == Symbol)

{

++cnt;

--Len;

}

if (cnt > 0)

str = str.erase(i + 1, cnt - 1);

}

}

}

void DeletePoints(string& str) {

char Symbol = '.';

int Len = str.length();

if (str.find(Symbol) >= 0) {

for (int i = str.find(Symbol); i < Len; i++) {

int cntPoints = 0;

int cntEllipsis = 0;

while (str[i + cntPoints] == Symbol)

++cntPoints;

cntEllipsis = cntPoints / 3;

if (cntEllipsis != 0) {

str = str.erase(i + 3, cntPoints - 3);

Len -= cntPoints - 3;

i += 3;

}

if (cntPoints == 2) {

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

Len--;

}

if (cntEllipsis > 1) {

str = str.erase(i, 3);

Len -= 3;

}

}

}

}

void DeleteNumbers(string& str, int& i) {

int Len = str.length();

string str2;

bool Number = false;

while (((str[i] != ' ') and (str[i] != ',')

and (str[i] != '!') and (str[i] != '?')

and (str[i] != ';') and (str[i] != ':')

and (str[i] != ')') and (str[i] != '.')

and (str[i] != '/')) and (i != Len))

{

if (str[i] >= 48 and str[i] <= 57)

Number = true;

str2 += str[i];

++i;

}

if (Number == false)

cout << str2 << ' ';

}

void UppercaseLetters(string str) {

int Len = str.length();

bool space = false;

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

if (str[i] == ' ')

str[i + 1] = toupper(str[i + 1]);

}

str[0] = toupper(str[0]);

cout << str << '\n';

}

void LinearSearch(string str) {

string Substring;

cout << "Исходная строка:" << '\n';

cout << str << '\n';

getline(cin, Substring);

cout << '\n' << "Введите подстроку для её поиска:" << '\n';

cin >> Substring;

int Len = str.length();

int LenSub = Substring.length();

string str2;

bool flag = false;

for (int i = 0; i < Len - LenSub + 1; i++) {

int j = i, cnt = 0;

str2.clear();

while (cnt != LenSub) {

str2 += str[j];

j++;

cnt++;

}

if (str2 == Substring) {

for (int k = i; k < j; k++)

str[k] = toupper(str[k]);

flag = true;

}

}

if (flag == false)

cout << '\n' << "Данная подстрака не найдена" << '\n';

else

cout << '\n' << str << '\n';

}

void Task2(string& str) {

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

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

{

char Symbol = Symbols[i];

DeleteSymbols(str, Symbol);

}

DeletePoints(str);

int Len = str.length();

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

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

cout << str;

}

void Task3(string& str) {

cout << "Исходная строка:" << '\n';

cout << str << '\n';

cout << "Слова из строки, не содержащие цифры:" << '\n';

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

DeleteNumbers(str, i);

}

cout << '\n';

}

void Task4(string& str) {

cout << "Исходная строчка:" << '\n';

cout << str << '\n';

cout << "Строчка со словами с заглавной буквы:" << '\n';

str[0] = tolower(str[0]);

UppercaseLetters(str);

}

void PrintMenu4() {

system("cls");

cout << "Что выхотите сделать?" << '\n';

cout << "1. Заполнить строку" << '\n';

cout << "2. Отформатировать строку" << '\n';

cout << "3. Слова не содержащие цифры" << '\n';

cout << "4. Строка со словами с заглавной буквы" << '\n';

cout << "5. Найти подстроку" << '\n';

cout << "6. Выйти из программы" << '\n';

}

void Menu4(string& str) {

int Variant;

do {

PrintMenu4();

cin >> Variant;

switch (Variant) {

case 1:

char perm;

cout << "Ввести строку с файла или с клавиатуры(a или b)" << '\n';

cin >> perm;

while (perm != 'a' and perm != 'b') {

cout << "Неверно введён вид вывода, попробуйте снова: ";

cin >> perm;

}

if (perm == 'a')

StrInput(str);

else {

cout << "Введите строку:" << '\n';

getline(cin, str);

}

break;

case 2:

Task2(str);

cout << '\n' << "Отформатированная строка" << '\n';

cout << str;

break;

case 3:

Task3(str);

break;

case 4:

Task4(str);

break;

case 5:

LinearSearch(str);

break;

}

if (Variant != 6)

system("pause");

} while (Variant != 6);

}

void TaskPrac4() {

setlocale(0, "");

string str;

Menu4(str);

}

void PrintMainMenu() {

cout << "1. Первое практическое задание" << '\n';

cout << "2. Второе практическое задание" << '\n';

cout << "3. Третье практическое задание" << '\n';

cout << "4. Четвёртое практическое задание" << '\n';

cout << "5. Закрыть программу" << '\n';

}

//------------------------------------------------------------------------------------------------------------------------------------------

void MainMenu() {

int Task;

do {

system("cls");

cout << "Какое практическое задание выполнить?" << '\n';

PrintMainMenu();

cin >> Task;

switch (Task) {

case 1:

Task1();

break;

case 2:

Task2();

break;

case 3:

Task3();

break;

case 4:

TaskPrac4();

break;

}

if (Task != 5)

system("pause");

} while (Task != 5);

}

int main() {

setlocale(LC\_ALL, "Russian");

MainMenu();

}