**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: Объединение всех лабораторных работ

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 | Кравцов К.А. |  |
| Преподаватель | Глущенко А.Г. |  |

Санкт-Петербург

2021

**Цель работы.**

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

**Основные теоретические положения.**

Оператор sizeof(“тип\_данных”) позволяет узнать размер типов данных(int, float, double, char и т.д.).

Объединения(union) – это две или более переменных расположенных по одному адресу (они разделяют одну и ту же память). Объединения определяются с использованием ключевого слова union. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

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.

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.

Функции внутри своего тела могут вызывать сами себя. Такой вызов называется рекурсией.

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

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.

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.

Двумерный массив — это “массив массивов”, т.е. в каждом элементе хранится ещё один массив.

Массив же представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов.

Наглядно двумерный массив можно представить, как список одномерных масивов.

Зная, что массив в C++ это указатель на его первый элемент, мы можем получить элемент 2го столбца и 3й строки, воспользовавшись арифметикой указателей:

int element = \*(\*a + 11);

Т.к. это двумерный массив, то a – указатель на первый элемент двумерного массива, \*a – это указатель на первый элемент первого подмассива. Разыменовываем указатель *a* и прибавляем к нему 11. Таким образом, получаем нужный элемент.

Так же существуют и другие способы получить этот же элемент:

int element = \*((int\*)arr + 11);

В примере выше мы явно приводим arr к указателю (int\*), затем используем арифметику указателей.

int element = \*(\*(arr + 2) + 1);

Пример выше практически не отличается от первого, но в разыменовывании указателя arr мы прибавляем 2, тем самым выбирая указатель на 3ю строку, после чего получаем нужный элемент. Эта запись эквивалентна следующей:

int element = arr[2][1];

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

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

Текстовые строки представляются с помощью одномерных массивов символов. В языке **C++** текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). Поэтому, если вы хотите создать текстовый массив для хранения 10 (N) символов, нужно выделить память под 11 (N+1) символов.

Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.

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:

**Постановка задачи.**

Необходимо написать программу, которая:

Содержит все 4 лабораторные работ, есть возможность переключения между заданиями (интерактивное меню).

**Выполнение работы.**

Код программы представлен в приложении A.

1. Пользователь вводит номер лабораторной работы.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1 – Меню программы

2. Результат выполнения первой лабораторной работы

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 – Лабораторная работа №1

3. Результат выполнения 2 лабораторной работы

Изображение выглядит как текст

Автоматически созданное описание

Рис 3.1 – Лабораторная работа №2

Изображение выглядит как текст

Автоматически созданное описание

Рис 3.2 – Лабораторная работа №2

4. Результат выполнения 3 лабораторной работы:

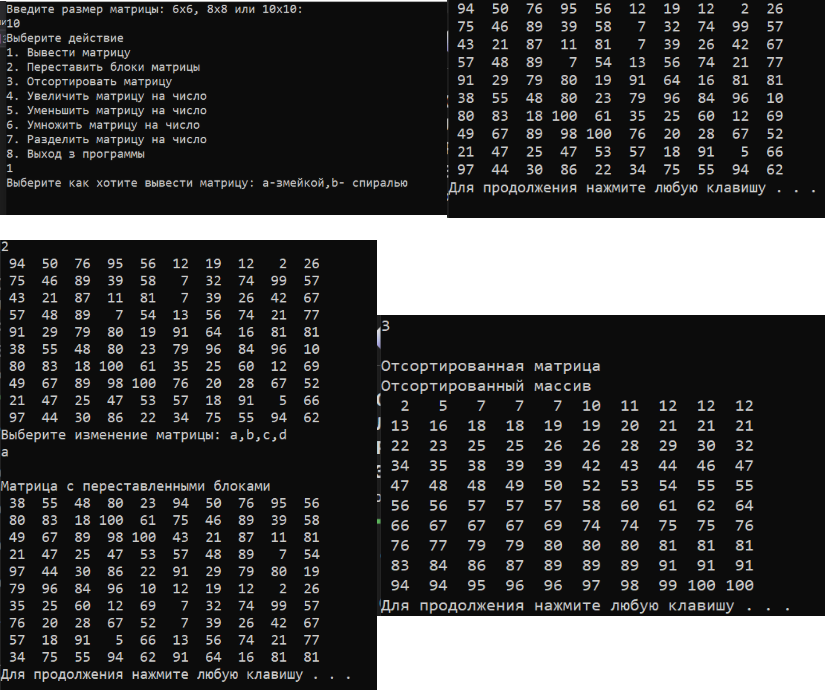
****

Рис. 4.1 – Лабораторная работа №3

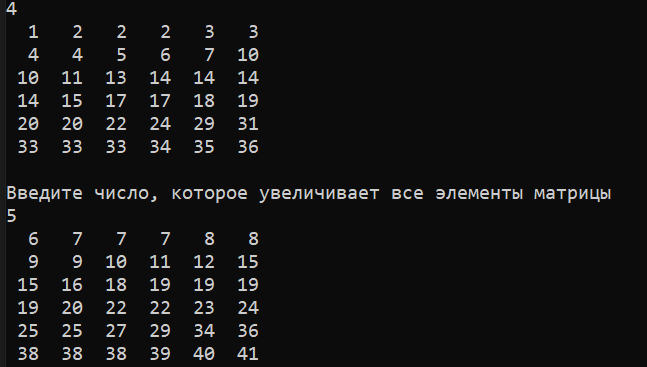
****

Рис. 4.2 – Лабораторная работа №3

5. Результат выполнения 4 лабораторной работы

Изображение выглядит как текст

Автоматически созданное описание

Рис.5.1 – Лабораторная работа №4

Изображение выглядит как текст

Автоматически созданное описание

Рис.5.2 – Лабораторная работа №4

Изображение выглядит как текст

Автоматически созданное описание

Рис.5.3 – Лабораторная работа №4

Изображение выглядит как текст

Автоматически созданное описание

Рис.5.3 – Лабораторная работа №4

**Выводы**

В результате выполнения курсовой работы были объединены все лабораторные работы, а также добавлено интерактивное меню для многоразового вызова любой работы пользователем.

**ПРИЛОЖЕНИЕ А**

**РАБОЧИЙ КОД**

**#include <iostream>**

**#include <string>**

**#include <algorithm>**

**#include <sstream>**

**#include <cctype>**

**#include <fstream>**

**#include <ctime>**

**#include <chrono>**

**#include <conio.h>**

**#include <iomanip>**

**#include <windows.h>**

**#include <string>**

**using namespace std;**

**using namespace chrono;**

**int\* maxSort = 0;**

**int\* minSort = 0;**

**int\* maxNoSort = 0;**

**int\* minNoSort = 0;**

**void Printarr(int\*\* arr, int N);**

**void zapolnenie(int\*\* arr, int N);**

**void SpiralArr(int\*\* arr, int N);**

**void SnakeArr(int\*\* arr, int N);**

**void modifiedArr(int\*\* arr, int N);**

**void SortOfArr(int\*\* arr, int N);**

**void PlusArr(int\*\* arr, int N);**

**void MinusArr(int\*\* arr, int N);**

**void MultiplyArr(int\*\* arr, int N);**

**void DivideArr(int\*\* arr, int N);**

**bool is\_sign(char value);**

**void lower(string& s);**

**void func2(string& s);**

**void func3(string s);**

**void func4(string& s1);**

**void linsearh(string& s);**

**void SInput(string& s);**

**int main()**

**{**

**setlocale(0, "");**

**int cursach;**

**do**

**{**

**cout << "Введите номер лабораторной работы:" << endl;**

**cout << "1 - Типы данных и их внутреннее представление в памяти" << endl;**

**cout << "2 - Одномерные статические массивы" << endl;**

**cout << "3 - Двумерные статические массивы.Указатели" << endl;**

**cout << "4 - Строки и символьные массивы,файлы" << endl;**

**cout << "5 - Выход из программы" << endl;**

**cin >> cursach;**

**/\*while (cursach != 1 && cursach != 2 && cursach != 3 && cursach != 4)**

**{**

**cout << "Повторите попытку! " << endl;**

**cin >> cursach;**

**}\*/**

**switch (cursach)**

**{**

**case 1:**

**{**

**cout << "Сколько памяти (в байтах) отводится под различные типы данных со спецификаторами и без" << endl;**

**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;**

**//2)**

**int value; // обьявил переменную**

**unsigned int order = sizeof(int) \* 8; // Количество разрядов**

**unsigned int mask = 1 << (order - 1); // Маска побитового сравнения**

**cout << "Введите число типа(int): " << endl;**

**cin >> value;//ввожу значение**

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

**{**

**putchar(value & mask ? '1' : '0');**

**value <<= 1; // Побитовый сдвиг числа**

**if (i % 8 == 0)**

**{**

**putchar(' ');**

**}**

**if (i % order - 1 == 0)**

**{**

**putchar(' ');**

**}**

**}**

**cout << endl;**

**//3)**

**cout << "Введите число типа (float): " << endl;**

**union {//обьединение указывает на 1 адрес**

**float value1;**

**int numberofvalue;**

**};**

**cin >> value1;**

**mask = 1 << order - 1;//обновил маску от предыдущих значений**

**cout << "\n" << " |'-----Мантисса-----'|'Порядок'|\n";**

**for (int i = 0; i < order; i++)**

**{**

**if (i == 23 || i == 1)//пробелы после значка и мантиссы**

**{**

**cout << " ";**

**}**

**if (numberofvalue & mask)**

**{**

**cout << "1";**

**}**

**else**

**{**

**cout << "0";**

**}**

**numberofvalue <<= 1;**

**}**

**cout << endl;**

**//4)**

**cout << "Введите число типа (double): " << endl;**

**union {**

**double doublevalue;**

**int number[2];**

**};**

**cin >> doublevalue;**

**unsigned order2 = sizeof(double) \* 8;**

**unsigned mask2 = 1 << order2 - 1;**

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

**for (int i = 0; i < order2 / 2; i++)//первый массив это половина**

**//дабла(работа с первыми 32 битами)**

**{**

**putchar(number[1] & mask2 ? '1' : '0');**

**number[1] <<= 1;**

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

**{**

**putchar(' ');**

**}**

**}**

**for (int j = 1; j < order2 / 2; j++)//вторые 32 бита**

**{**

**putchar(number[0] & mask2 ? '1' : '0');**

**number[0] <<= 1;**

**if (j % 8 == 0)**

**{**

**putchar(' ');**

**}**

**}**

**cout << "\n" << endl;**

**}**

**break;**

**case 2:**

**{**

**const int N = 100;**

**int arr[N];**

**int\* pNoSort[N] = { 0 };**

**int\* pSort[N] = { 0 };**

**cout << "Доброго времени суток,выберите действие от 1 до 9" << endl;**

**cout << "1 " << "создать массив длины 100" << endl;**

**cout << "2 " << "отсортировать массив длины 100" << endl;**

**cout << "3 " << "найти максимальный и минимальный элемент массива" << endl;**

**cout << "4 " << "найти количество элементов массива, которые равны среднему значению максимального и минимального элементов" << endl;**

**cout << "5 " << "найти кол-во элементов меньше числа а" << endl;**

**cout << "6 " << "найти кол-во элементов больше числа b" << endl;**

**cout << "7 " << "есть ли ваше число в массиве" << endl;**

**cout << "8 " << "поменять местами элементы массива" << endl;**

**cout << "9 " << "выход из программы" << endl;**

**for (;;)**

**{**

**int choose = 0;**

**cout << "выберите действие от 1 до 9" << endl;**

**cin >> choose;**

**switch (choose)**

**{**

**case 1:**

**{**

**cout << "======================================================================================================================" << endl;**

**cout << "неотсортированный массив" << endl;**

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

**{**

**arr[i] = rand() % 199 + -99;**

**}**

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

**{**

**cout << arr[i] << " ";**

**\*pNoSort = &arr[i];**

**}**

**cout << endl;**

**cout << "======================================================================================================================" << endl;**

**}**

**break;**

**case 2:**

**{**

**\*pNoSort;**

**int k;//bufer peremennaya**

**cout << "======================================================================================================================" << endl;**

**cout << "отсортированный массив" << endl;**

**steady\_clock::time\_point start = steady\_clock::now();**

**for (int j = 1; j <= (N - 1); j++)**

**for (int i = 0; i < N - j; i++)**

**if (arr[i] > arr[i + 1])**

**{**

**k = arr[i + 1];**

**arr[i + 1] = arr[i];**

**arr[i] = k;**

**}**

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

**{**

**cout << arr[i] << " ";**

**pSort[i] = &arr[i];**

**}**

**steady\_clock::time\_point end = steady\_clock::now();**

**int t = duration\_cast<nanoseconds>(end - start).count(); //Приводит объект duration к указанному int типу.**

**cout << endl;**

**cout << "время сортирровки массива " << t << " наносекунд \n";**

**cout << "======================================================================================================================" << endl;**

**}**

**break;**

**case 3:**

**{**

**\*pNoSort;**

**int max1 = arr[0];**

**int min1 = arr[0];**

**steady\_clock::time\_point startNoSort = steady\_clock::now();**

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

**{**

**if (max1 < arr[i]) max1 = arr[i]; //если значение элемента больше значения переменной max, то записываем это значение в переменную**

**if (min1 > arr[i]) min1 = arr[i]; //аналогично и для min**

**}**

**steady\_clock::time\_point endNoSort = steady\_clock::now();**

**int t = duration\_cast<nanoseconds>(endNoSort - startNoSort).count();**

**cout << "время нахождения min и max неотсортированного массива " << t << " наносекунд \n";**

**cout << "минимальный элемент неотсортированного массива: " << min1 << endl;**

**cout << "макисимальный элемент неотсортированного массива: " << max1 << endl;**

**minNoSort = &min1;**

**maxNoSort = &max1;**

**\*pSort;**

**steady\_clock::time\_point start = steady\_clock::now();**

**int min = arr[0];**

**int max = arr[N - 1];**

**steady\_clock::time\_point end = steady\_clock::now();**

**int time = duration\_cast<nanoseconds>(end - start).count();**

**cout << "время нахождения min и max отсортированного массива " << time << " наносекунд \n";**

**cout << "минимальный элемент отсортированного массива: " << min << endl;**

**cout << "макисимальный элемент отсортированного массива: " << max << endl;**

**minSort = &min;**

**maxSort = &max;**

**}**

**break;**

**case 4:**

**{**

**//Подсчитайте время поиска.**

**\*pNoSort;**

**int sumNoSort = \*minNoSort + \*maxNoSort;**

**double srednNoSort = (double)(sumNoSort) / 2.0;**

**if (srednNoSort < 0.0)**

**{**

**if (srednNoSort <= (int)srednNoSort - 0.5)**

**{**

**cout << "среднее значение неотсортированного массива " << int(srednNoSort) - 1 << endl;**

**}**

**else**

**{**

**cout << "среднее значение неотсортированного массива " << int(srednNoSort) << endl;**

**}**

**}**

**else**

**{**

**if (srednNoSort >= (int)srednNoSort + 0.5)**

**{**

**cout << "среднее значение неотсортированного массива " << int(srednNoSort) + 1 << endl;**

**}**

**else**

**{**

**cout << "среднее значение неотсортированного массива " << int(srednNoSort) << endl;**

**}**

**}**

**\*pSort;**

**int sum = \*minSort + \*maxSort;**

**double sredn = (double)(sum) / 2.0;**

**if (sredn < 0.0)**

**{**

**if (sredn <= (int)sredn - 0.5)**

**{**

**cout << "среднее значение отсортированного массива " << int(sredn) - 1 << endl;**

**}**

**else**

**{**

**cout << "среднее значение отсортированного массива " << int(sredn) << endl;**

**}**

**}**

**else**

**{**

**if (sredn >= (int)sredn + 0.5)**

**{**

**cout << "среднее значение отсортированного массива " << int(sredn) + 1 << endl;**

**}**

**else**

**{**

**cout << "среднее значение отсортированного массива " << int(sredn) << endl;**

**}**

**}**

**int kolvo = 0; //нужнен ли подсчет времени выполнения для неотсортированного массива?**

**steady\_clock::time\_point start = steady\_clock::now();**

**for (int i = 0; (i < N) && (arr[i] <= sredn); i++)**

**{**

**if (arr[i] == sredn)**

**{**

**cout << " " << "Индекс элемента " << i << " = среднему значению : " << endl;**

**kolvo++;**

**steady\_clock::time\_point end = steady\_clock::now();**

**int t = duration\_cast<nanoseconds>(end - start).count();**

**cout << " Количество элементов отсортированного массива,которые = среднему значению: " << kolvo << endl;**

**cout << "время поиска " << t << " наносекунд \n";**

**}**

**}**

**if (kolvo == 0)**

**{**

**cout << "Индексы элемента = среднему значению не найдено" << endl;**

**}**

**}**

**break;**

**case 5:**

**{**

**int LessNumber;**

**cout << "Введите целое число для вывода елементов массива,которые меньше данного числа" << endl;**

**cin >> LessNumber;**

**cout << "Элементы массива,которые меньше данного числа" << endl;**

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

**{**

**if (arr[i] < LessNumber)**

**{**

**cout << arr[i] << " ";**

**}**

**}**

**cout << endl;**

**}**

**break;**

**case 6:**

**{**

**int MoreNumber;**

**cout << endl << "Введите целое число для вывода елементов массива,которые больше данного числа" << endl;**

**cin >> MoreNumber;**

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

**{**

**if (arr[i] > MoreNumber)**

**{**

**cout << arr[i] << " ";**

**}**

**}**

**cout << endl;**

**}**

**break;**

**case 7:**

**cout << "Извините за предоставленные неудобства,автор не смог выполнить данное задание(";**

**break;**

**case 8:**

**{**

**cout << endl << "введите индексы тех элементов масива,которые вы хотите поменять" << endl;**

**int firstelement, secondelement;**

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

**{**

**cout << arr[i] << " ";**

**}**

**cout << endl;**

**cout << endl;**

**cout << "введите первый индекс элемента" << endl;**

**cin >> firstelement;**

**cout << "введите второй индекс элемента" << endl;**

**cin >> secondelement;**

**steady\_clock::time\_point start = steady\_clock::now();**

**swap(arr[secondelement], arr[firstelement]);**

**steady\_clock::time\_point exit = steady\_clock::now();**

**int t1 = duration\_cast<nanoseconds>(exit - start).count();**

**cout << t1 << " наносекунд - скорость обмена элементов массива \n";**

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

**{**

**cout << arr[i] << " ";**

**}**

**cout << endl;**

**}**

**break;**

**case 9:**

**{**

**cout << "До новых встреч!" << endl;**

**goto exit;**

**break;**

**}**

**default:**

**{**

**cout << "уупс... вы ввели неправильное значение,повторите попытку еще раз" << endl;**

**}**

**break;**

**}**

**}**

**exit:**

**break;**

**}**

**case 3:**

**{**

**srand((unsigned)time(NULL));**

**int N;**

**cout << "Введите размер матрицы: 6x6, 8x8 или 10x10:" << "\n";**

**cin >> N;**

**int\* pN = &N;**

**while (\*pN != 6 && \*pN != 8 && \*pN != 10) {**

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

**cin >> N;**

**}**

**signed int\*\* arr = new int\* [N];**

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

**arr[i] = new int[N];**

**zapolnenie(arr, N);**

**int number = 0;**

**do {**

**cout << "Выберите действие" << '\n';**

**cout << "1. Вывести матрицу" << '\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. Разделить матрицу на число" << '\n';**

**cout << "8. Выход из программы" << '\n';**

**cin >> number;**

**switch (number) {**

**case 1:**

**char snakeSpiral;**

**cout << "Выберите как хотите вывести матрицу: a-змейкой,b- спиралью" << endl;**

**cin >> snakeSpiral;**

**while (snakeSpiral != 'a' && snakeSpiral != 'b') {**

**cout << "Неверно введён номер, попробуйте снова: ";**

**cin >> snakeSpiral;**

**}**

**if (snakeSpiral == 'a')**

**{**

**cout << "Матрица змейкой" << endl;**

**SnakeArr(arr, N);**

**}**

**else**

**{**

**cout << "Спиральная матрица" << endl;**

**SpiralArr(arr, N);**

**}**

**break;**

**case 2:**

**Printarr(arr, N);**

**modifiedArr(arr, N);**

**break;**

**case 3:**

**cout << '\n' << "Отсортированная матрица" << '\n';**

**SortOfArr(arr, N);**

**break;**

**case 4:**

**Printarr(arr, N);**

**PlusArr(arr, N);**

**break;**

**case 5:**

**Printarr(arr, N);**

**MinusArr(arr, N);**

**break;**

**case 6:**

**Printarr(arr, N);**

**MultiplyArr(arr, N);**

**break;**

**case 7:**

**Printarr(arr, N);**

**DivideArr(arr, N);**

**break;**

**}**

**if (number != 8)**

**system("pause");**

**} while (number != 8);**

**break;**

**}**

**case 4:**

**{**

**string s, s1;**

**string symb;**

**cout << "Выберите способ ввода: a-файл,b- с клавиатуры" << endl;**

**getline(cin, symb);**

**while (symb != "a" && symb != "b") {**

**cout << "Неверно введён номер, попробуйте снова: ";**

**getline(cin, symb);**

**}**

**if (symb == "a")**

**{**

**SInput(s);**

**}**

**if (symb == "b")**

**{**

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

**getline(cin, s);**

**}**

**s1 = s;**

**int number;**

**do {**

**cout << endl;**

**cout << "Выберите действие" << '\n';**

**cout << "2. Отредактировать строку" << '\n';**

**cout << "3. Вывести на экран слова последовательности в алфавитном порядке" << '\n';**

**cout << "4. Вывести все слова исходной последовательности на экран вертикально" << '\n';**

**cout << "5. Линейный поиск подстроки в строке" << '\n';**

**cout << "6. Выход из программы" << '\n';**

**cin >> number;**

**switch (number) {**

**case 2:**

**func2(s);**

**break;**

**case 3:**

**func3(s);**

**break;**

**case 4:**

**func4(s1);**

**break;**

**case 5:**

**linsearh(s);**

**break;**

**}**

**} while (number != 6);**

**break;**

**}**

**}**

**} while (cursach != 5);**

**return 0;**

**}**

**void Printarr(int\*\* arr, int N)**

**{**

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

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

**{**

**cout << setw(3) << setprecision(3) << arr[i][j];//все элементы в поток вывода**

**cout.flush();**

**if (j != N)**

**cout << ' ';**

**}**

**cout << endl;**

**}**

**Sleep(10);**

**}**

**void zapolnenie(int\*\* arr, int N)**

**{**

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

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

**arr[i][j] = 0;**

**}**

**void SnakeArr(int\*\* arr, int N)**

**{**

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

**if ((i % 2) == 0)**

**{**

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

**{**

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

**system("cls");**

**Printarr(arr, N);**

**}**

**}**

**else {**

**for (int j = N - 1; j >= 0; j--)**

**{**

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

**system("cls");**

**Printarr(arr, N);**

**}**

**}**

**}**

**}**

**void SpiralArr(int\*\* arr, int N) {**

**zapolnenie(arr, N);**

**int FirstLine = 0;**

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

**for (int i = FirstLine; i < N - FirstLine; i++) {**

**arr[FirstLine][i] = rand() % (N \* N) + 1;**

**system("cls");**

**Printarr(arr, N);**

**}**

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

**arr[i][N - FirstLine - 1] = rand() % (N \* N) + 1;**

**system("cls");**

**Printarr(arr, N);**

**}**

**for (int i = N - FirstLine - 2; i >= FirstLine; i--) {**

**arr[N - FirstLine - 1][i] = rand() % (N \* N) + 1;**

**system("cls");**

**Printarr(arr, N);**

**}**

**for (int i = N - FirstLine - 2; i > FirstLine; i--) {**

**arr[i][FirstLine] = rand() % (N \* N) + 1;**

**system("cls");**

**Printarr(arr, N);**

**}**

**FirstLine++;**

**}**

**}**

**void modifiedArr(int\*\* arr, int N)**

**{**

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

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

**modarr[i] = new int[N];**

**char choose;**

**cout << "Выберите изменение матрицы: a,b,c,d" << endl;**

**cin >> choose;**

**while (choose != 'a' && choose != 'b' && choose != 'c' && choose != 'd')**

**{**

**cout << "Повторите попытку " << endl;**

**cin >> choose;**

**}**

**switch (choose)**

**{**

**case 'a':**

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

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

**\*(\*(modarr + i) + j) = arr[i + N / 2][j]; //**

**modarr[i][j + N / 2] = arr[i][j];**

**modarr[i + N / 2][j + N / 2] = arr[i][j + N / 2];**

**modarr[i + N / 2][j] = arr[i + N / 2][j + N / 2];**

**}**

**arr = modarr;**

**cout << '\n' << "Матрица с переставленными блоками" << '\n';**

**Printarr(arr, N);**

**break;**

**case 'b':**

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

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

**\*(\*(modarr + i) + j) = arr[i + N / 2][j + N / 2];**

**modarr[i][j + N / 2] = arr[i + N / 2][j];**

**modarr[i + N / 2][j + N / 2] = arr[i][j];**

**modarr[i + N / 2][j] = arr[i][j + N / 2];**

**}**

**arr = modarr;**

**cout << '\n' << "Матрица с переставленными блоками" << '\n';**

**Printarr(arr, N);**

**break;**

**case 'c':**

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

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

**\*(\*(modarr + i) + j) = arr[i + N / 2][j];**

**modarr[i][j + N / 2] = arr[i + N / 2][j + N / 2];**

**modarr[i + N / 2][j + N / 2] = arr[i][j + N / 2];**

**modarr[i + N / 2][j] = arr[i][j];**

**}**

**arr = modarr;**

**cout << '\n' << "Матрица с переставленными блоками" << '\n';**

**Printarr(arr, N);**

**break;**

**case 'd':**

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

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

**\*(\*(modarr + i) + j) = arr[i][j + N / 2];**

**modarr[i][j + N / 2] = arr[i][j];**

**modarr[i + N / 2][j + N / 2] = arr[i + N / 2][j];**

**modarr[i + N / 2][j] = arr[i + N / 2][j + N / 2];**

**}**

**arr = modarr;**

**cout << '\n' << "Матрица с переставленными блоками" << '\n';**

**Printarr(arr, N);**

**break;**

**}**

**}**

**void PlusArr(int\*\* arr, int N) {**

**int Number;**

**cout << '\n' << "Введите число, которое увеличивает все элементы матрицы" << '\n';**

**cin >> Number;**

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

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

**\*(\*(arr + i) + j) += Number;**

**Printarr(arr, N);**

**}**

**void MinusArr(int\*\* arr, int N) {**

**int Number;**

**cout << '\n' << "Введите число, которое уменьшает все элементы матрицы" << '\n';**

**cin >> Number;**

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

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

**\*(\*(arr + i) + j) -= Number;**

**Printarr(arr, N);**

**}**

**void MultiplyArr(int\*\* arr, int N) {**

**int Number;**

**cout << '\n' << "Введите число, которое умножает все элементы матрицы" << '\n';**

**cin >> Number;**

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

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

**\*(\*(arr + i) + j) \*= Number;**

**Printarr(arr, N);**

**}**

**void DivideArr(int\*\* arr, int N) {**

**int Number;**

**cout << '\n' << "Введите число, которое делит все элементы матрицы" << '\n';**

**cin >> Number;**

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

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

**\*(\*(arr + i) + j) = arr[i][j] / Number;**

**Printarr(arr, N);**

**}**

**void SortOfArr(int\*\* arr, int N)**

**{**

**cout << "Отсортированный массив" << endl;**

**int Asize = N \* N;**

**int\* Arry = new int[Asize];**

**int iOfArry = 0;**

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

**{**

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

**{**

**(Arry[iOfArry] = arr[i][j]);**

**iOfArry++;**

**}**

**}**

**bool Change = 1;**

**while (Change)**

**{**

**Change = 0;**

**for (int\* i = Arry, \*End\_i = i + Asize - 2; i <= End\_i; i++)**

**{**

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

**Change = 1;**

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

**}**

**}**

**}**

**iOfArry = 0;**

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

**{**

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

**{**

**arr[i][j] = Arry[iOfArry];**

**iOfArry++;**

**}**

**}**

**Printarr(arr, N);**

**}**

**bool is\_sign(char value)**

**{**

**return ispunct(value);**

**}**

**void lower(string& s)**

**{**

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

**{**

**s[i] = tolower(s[i]);**

**}**

**cout << s;**

**}**

**void func2(string& s) {**

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

**if ((s[i] == ' ') && (s[i + 1] == ' ')) {**

**while (!((s[i] == ' ') && (s[i + 1] != ' '))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == '!') && (s[i + 1] == '!')) {**

**while (!((s[i] == '!') && (s[i + 1] != '!'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == '?') && (s[i + 1] == '?')) {**

**while (!((s[i] == '?') && (s[i + 1] != '?'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == ':') && (s[i + 1] == ':')) {**

**while (!((s[i] == ':') && (s[i + 1] != ':'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == ';') && (s[i + 1] == ';')) {**

**while (!((s[i] == ';') && (s[i + 1] != ';'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == '"') && (s[i + 1] == '"')) {**

**while (!((s[i] == '"') && (s[i + 1] != '"'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if ((s[i] == ',') && (s[i + 1] == ',')) {**

**while (!((s[i] == ',') && (s[i + 1] != ','))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**if (s[i] == '.') {**

**string buff;**

**buff = s[i];**

**int j = i;**

**j++;**

**while (s[j] == '.') {**

**buff += s[j];**

**j++;**

**}**

**if (buff.length() < 3) {**

**while (!((s[i] == '.') && (s[i + 1] != '.'))) {**

**s.erase(s.begin() + i);**

**}**

**}**

**else {**

**while (s[i + 3] == '.') {**

**s.erase(s.begin() + i);**

**}**

**i += 3;**

**}**

**}**

**}**

**cout << "Исправленная строка" << endl;**

**lower(s);**

**}**

**void func3(string s)**

**{**

**//Вывести на экран слова последовательности в алфавитном порядке.**

**int i, j, Temp;**

**int Flag;**

**int Number;**

**int Words[150];**

**replace\_if(s.begin(), s.end(), is\_sign, ' ');//замена всех знаков на пробелы**

**for (Number = 0, Flag = 1, i = 0; s[i]; i++)**

**{**

**if (s[i] == ' ')**

**{**

**s[i] = 0;**

**Flag = 1;**

**}**

**else if (Flag)**

**{**

**Words[Number++] = i;**

**Flag = 0;**

**}**

**}**

**for (j = Number - 1; j > 0; j--)**

**for (i = 0; i < j; i++)**

**if (strcmp(&s[Words[i]], &s[Words[i + 1]]) > 0)**

**{**

**Temp = Words[i];**

**Words[i] = Words[i + 1];**

**Words[i + 1] = Temp;**

**}**

**cout << "Слова cтроки в алфавитном порядке" << '\n';**

**for (i = 0; i < Number; i++)**

**cout << &s[Words[i]] << " ";**

**}**

**void func4(string& s1)**

**{**

**// Вывести все слова исходной последовательности на экран вертикально.**

**int i;**

**int Flag;**

**int Number;**

**int Words[150];**

**replace\_if(s1.begin(), s1.end(), is\_sign, ' ');**

**for (Number = 0, Flag = 1, i = 0; s1[i]; i++)**

**{**

**if (s1[i] == ' ')**

**{**

**s1[i] = 0;**

**Flag = 1;**

**}**

**else if (Flag)**

**{**

**Words[Number++] = i;**

**Flag = 0;**

**}**

**}**

**cout << "Слова изначальной строки вертикально:" << '\n';**

**for (i = 0; i < Number; i++)**

**printf("%s\n", &s1[Words[i]]);**

**}**

**void linsearh(string& s)**

**{**

**string ch;**

**cout << s << endl;**

**cout << "Введите подстроку(итоговой строки): " << endl;**

**cin >> ch;**

**cout << "Индексы ключевого слова в строке" << endl;**

**unsigned int i, j;**

**for (int i = 0; i < s.length() - ch.length() + 1; i++)**

**{**

**j = 0;**

**while ((j < ch.length()) && (ch[j] == s[i + j]))**

**{**

**j = j + 1;**

**}**

**if (j == ch.length())**

**{**

**cout << i << " ";**

**}**

**}**

**}**

**void SInput(string& s) {**

**//hI,, HOw arE yOu???? HaVE a niCE.... day**

**ifstream File(R"(C:\Users\ACER\Desktop\nure\APrPract7\_1\4lab.txt)");**

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

**cout << "Открыть файл не удалось! \n";**

**}**

**while (!File.eof())**

**getline(File, s);**

**cout << "Введенная строка: " << '\n';**

**cout << s << '\n';**

**File.close();**

**}**