**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

Кафедра «Информационные системы»

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: **"ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ С++"**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Голубцов В.В. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**курсовая работа**

|  |  |  |
| --- | --- | --- |
| Студент: Голубцов В.В | | |
| Группа: 0323 | | |
| Тема проекта: "Основы алгоритмизации и программирования на языке С++" | | |
| Исходные данные: описание каждого подраздела курса и технические задания программ на информационно-обучающем сайте Stepik. | | |
| Содержание пояснительной записки:  «Аннотация», «Содержание», «Введение», «Проектные работы», «Заключение», «Список использованных источников», «Программный код». | | |
|  | | |
| Дата выдачи задания: 21.09.2020 | | |
| Дата сдачи реферата: 18.12.2020 | | |
| Дата защиты реферата: 18.12.2020 | | |
| Студент |  | Голубцов В.В. |
| Преподаватель |  | Глущенко А.Г. |

**Аннотация**

Курсовой проект состоял из обучающих разделов "Основы алгоритмизации и программирования на языке С++". В каждом разделе были подразделы с описанием главной задачи и тестовыми промежутками в виде написания небольшой программы или теста. Так всего было 8 обучающих подразделов, курсовая работа и 3 контрольные точки. Проект содержал в себе 4 проектных работы, в которых требовалось написать 4 программы по техническому заданию каждой из работы. По итогу были написаны все 4 проектные работы, выполнено >80% курса степика и выполнена курсовая работа в виде объединения всех проектных работ в один.

**Содержание**

1. Введение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5
2. Первая практическая работа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 6
3. Вторая практическая работа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9
4. Третья практическая работа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 18
5. Четвертая практическая работа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 24
6. Курсовая работа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 31
7. Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 34
8. Список использованных источников \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 35
9. Программный код практических работ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 36

**введение**

Целью работы заключалось обучение программирования и понимая работы программ на языке C++.

Требовалось по ходу курса выполнять различные тестовые промежуточные точки и выполнять проектные работы по пройденному разделу.

**1. ПервАЯ ПРАКТИЧЕСКАЯ РАБОТА**

**(ТИПЫ ДАННЫХ И ИХ ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ В ПАМЯТИ)**

**Цель работы.**

Разработать алгоритм и написать программу, которая позволяет:

1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

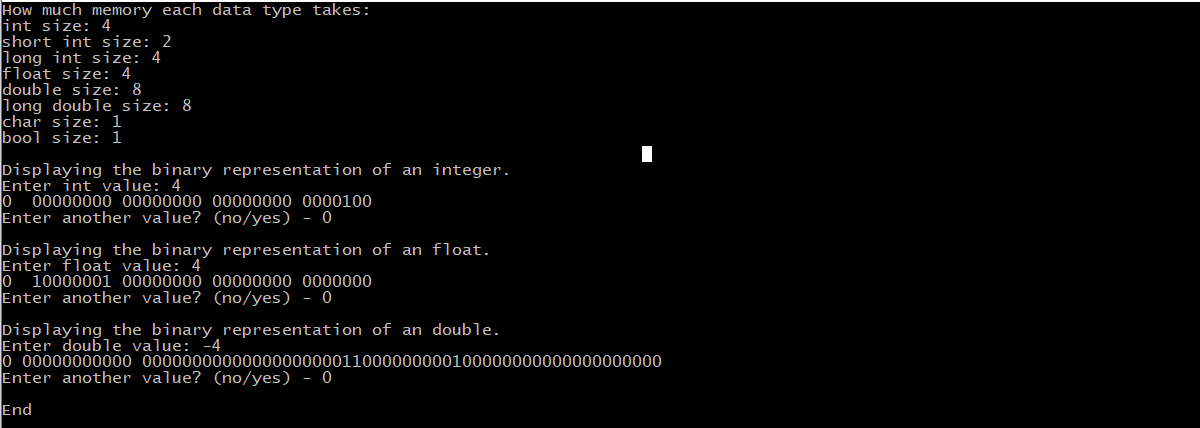
2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

**Экспериментальные результаты.**

Программа выводит на экран, сколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

Затем по порядку считываются, обрабатываются и выводятся двоичные представления в памяти типов int, float, double.

После этого программа завершает работу (см. рис. 1)

Рисунок 1 – Пример работы программы

Программа выводит корректные размеры выделяемой памяти для разных типов данных и представление чисел в памяти компьютера.

**Выводы.**

В ходе данной лабораторной работы я изучил внутреннее представление различных типов данных в памяти компьютера.

**2. ВТОРАЯ ПРАКТИЧЕСКАЯ РАБОТА**

**(ОДНОМЕРНЫЕ СТАТИЧЕСКИЕ МАССИВЫ)**

Цель работы.

Познакомиться с массивами, действиями над ними и различными типами их сортировки.

Постановка задачи.

Разработать алгоритм и написать программу, которая позволяет:

Необходимо написать программу, которая:

1)    Создает целочисленный массив размерности *N* = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

2)    Отсортировать заданный в пункте 1 элементы массива […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

3)    Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

4)    Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения. Выводит индексы всех элементов, которые равны этому значению, и их количество.

5)    Выводит количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

6)    Выводит количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

7)    Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

8)     Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

Должна присутствовать возможность запуска каждого пункта многократно.

Выполнение работы.

Для удобства и читаемости кода все выполняемые части программы я разделил на отдельные исполняемые методы.

В начале я создаю массив из 100 элементов которые принимают случайное число в диапазоне от -99 до 99 включительно используя функцию rand().

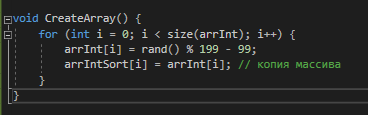


Рисунок : создание массива

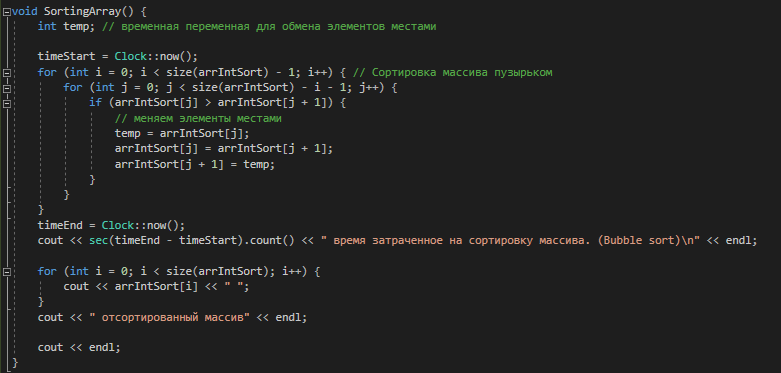
Далее нужно было отсортировать массив, я выбрал метод сортировки пузырьком.  


Рисунок : сортировка массива

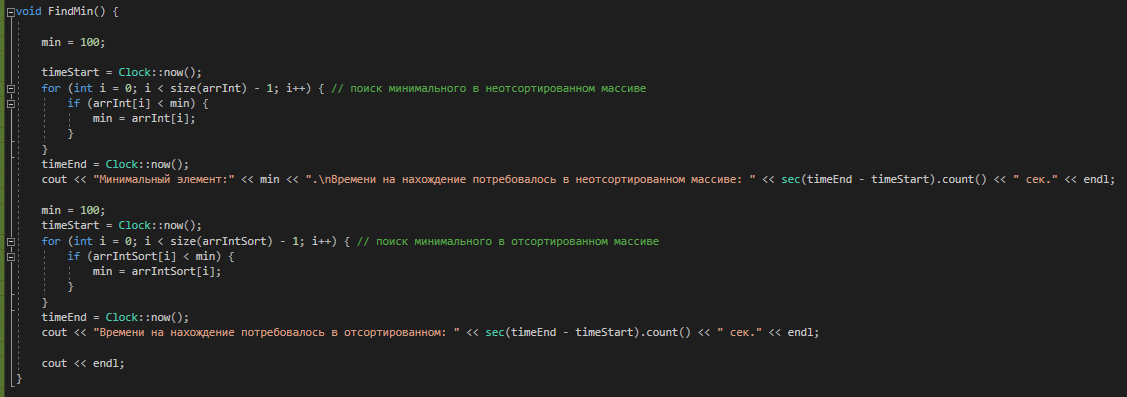
Для подсчета времени использовал две переменные типа Clock используя библиотеку "hrono". Первая принимает время до начала сортировки, вторая после, в выводе выводится их разность. Данная сортировка занимает в среднем 0,00025 сек.

Рисунок : нахождение минимальное элемента

В этой части кода я нахожу минимальный элемент массива с помощью перебора в двух массивах, отсортированном и обычном. Время требуемое на нахождение таких элементов занимает в среднем 1,2е-6 и 4,6е-6 соответственно.

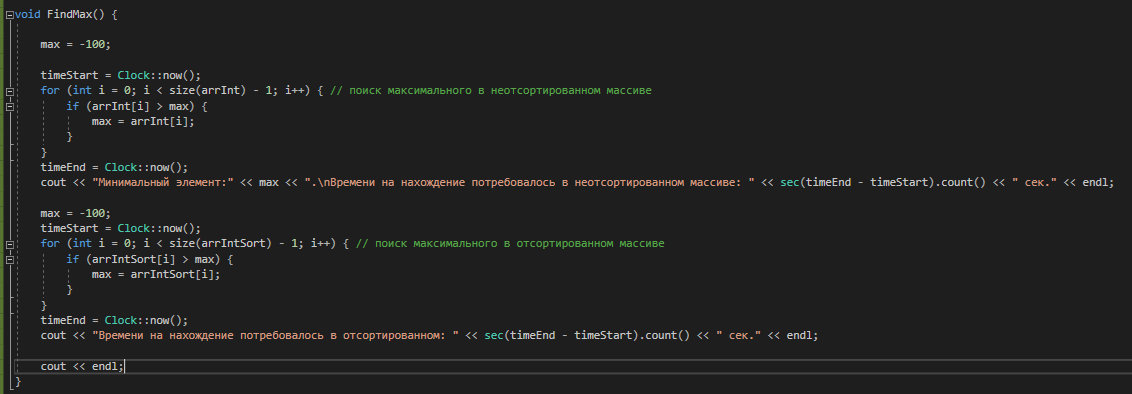
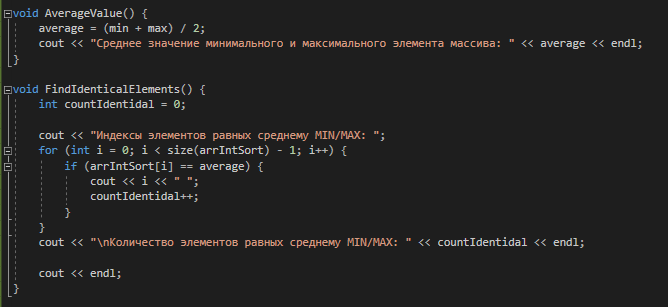
Аналогично методу поиска минимального элемента я нашел их максимальные. Времени занимает примерно как и с минимальными элементами.

Рисунок : поиск кол-ва элем. равынх среднему минимума и максимума

Рисунок : нахождение максимального значения

Далее я посчитал среднее значение минимального и максимального элемента и нашел сколько элементов в массиве соответствует данному среднему.

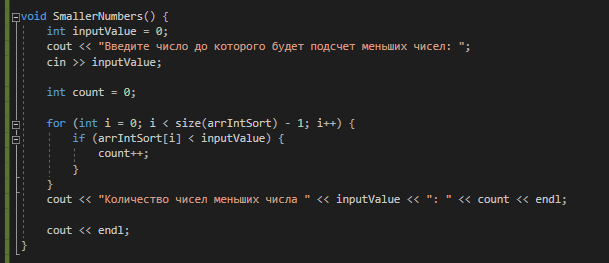


Рисунок : кол-во элементов меньших введенного пользователем элемента

Нахождение количества элементов до значения введенного пользователем выполняется простым счетчиком и перебором всех элементов.

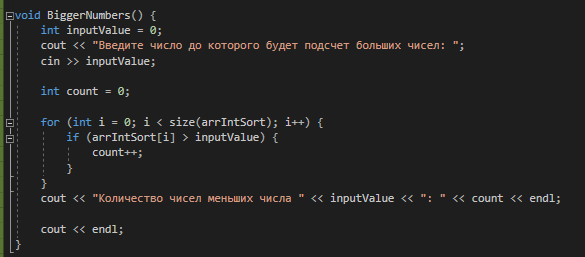
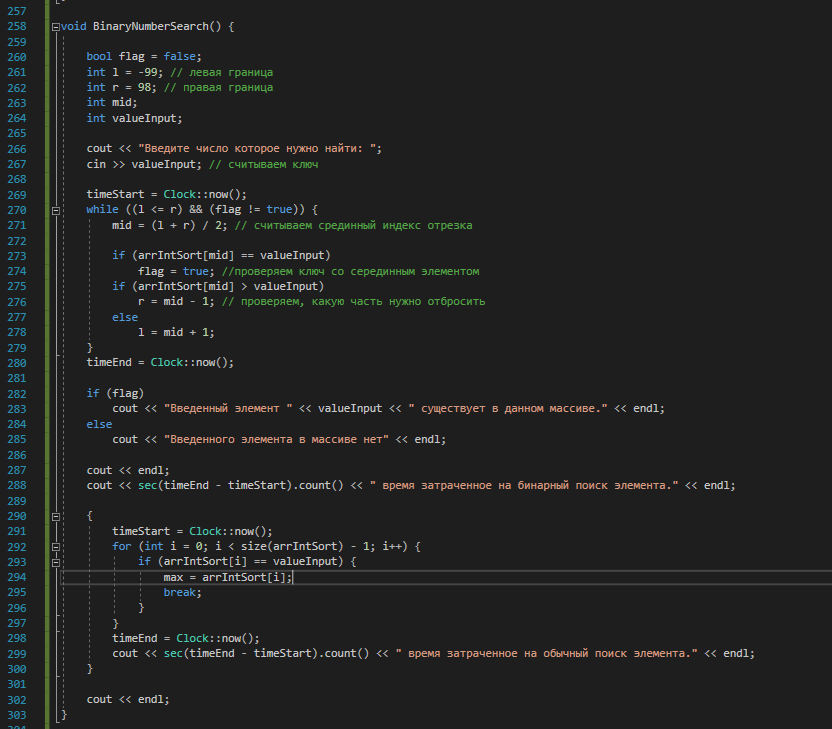


Рисунок : кол-во элементов больших введенного пользователем элемента

Поиск количества элементов больших введенного пользователем аналогичен предыдущему способу.

 Бинарный поиск. Он достаточно быстрый и не сложный в написании, что несоменный плюс его использования. Бинарный поиск находится в строках 270 — 279. В строке 263 мы создали переменную mid, в которой будет храниться индекс среднего элемента (из отрезка [l, r]). В строке 271 считываем средний элемент отрезка [l, r] в переменную mid, по формуле: (l + r) / 2 (в которой l — левая граница, r — правая граница). В строке 29 проверяем условие arrIntSort[mid] == valueInput:

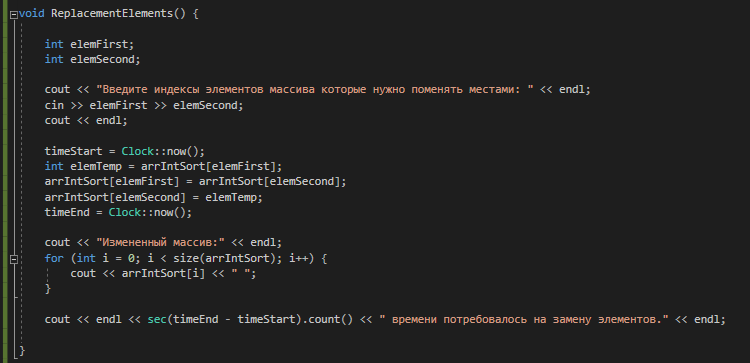
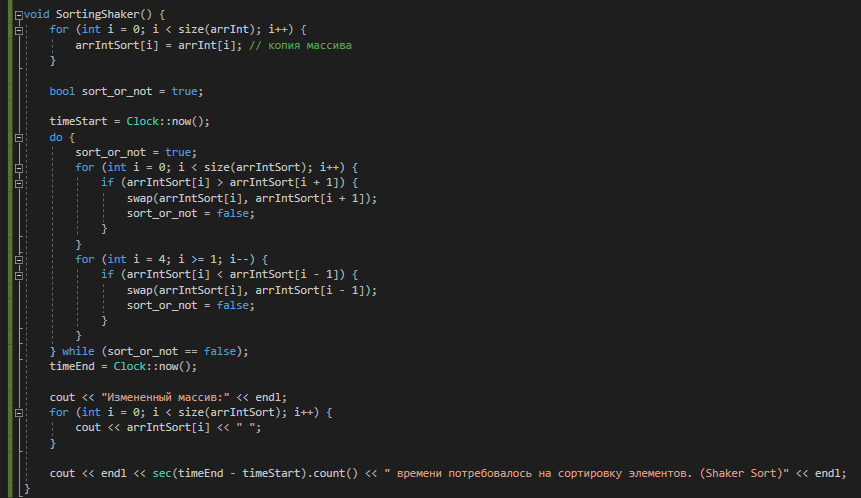
Замена двух элементов индекс которых вводится пользователем. Сначала запоминаем первый элемент в отдельную переменную, после присваиваем ему значение второго элемента и наконец второму присваиваем первый из переменной.

Рисунок : сортировка шейкером

Рисунок : замена местами 2-х элементов

В конце программы выполняется индивидуальное задание (рисунок 8), я выбрал 7-ое.

Реализовал шейкер сортировку. Принцип ее следующий: Перебирает до самого большого или минимального элемента, а далее перемещает его вправо или влево.

Плюсы: простота исполнения.

Минусы: медленное время выполнения, около 0,0007 с.

Выводы.

Массивы очень удобные и потому часто используются в программах. Познакомился с массивами на языке С++ и методами работы с ними.

**3. ТРЕТЬЯ ПРАКТИЧЕСКАЯ РАБОТА**

**(УКАЗАТЕЛИ)**

Цель работы.

Познакомиться на практике с указателями и арифметикой работы с ними.

Постановка задачи.

Необходимо написать программу, которая:

1)    Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка *N* (6,8,10) случайными числами от 1 до  N\*N согласно схемам, приведенным на рисунках. Пользователь должен видеть процесс заполнения квадратичной матрицы.



2)    Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами:



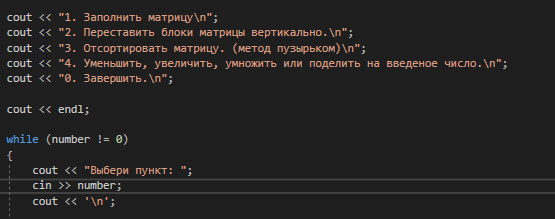
3)    Используя арифметику указателей, сортирует элементы любой сортировкой.

4)    Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

Выполнение работы.

Программа разделена на функции выполняющие операции над массивом данных.

Вначале пользователю необходимо выбрать операцию выполнения, всего их 4.



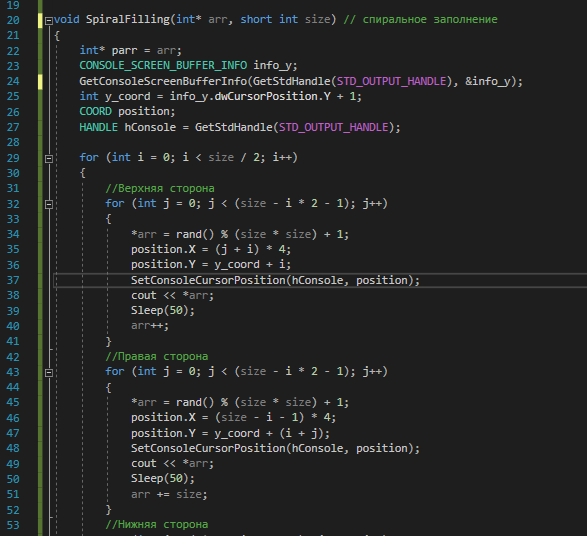
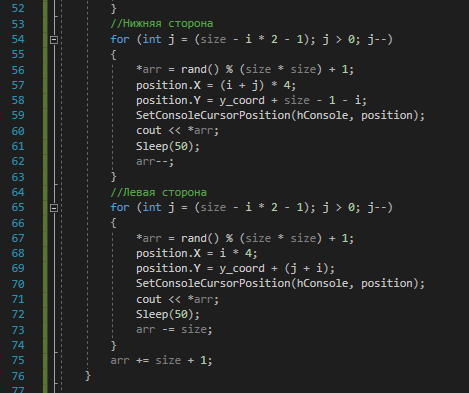
Первым пунктом мы заполняем массив элементов по спирали с поэтапным выводом их на экран с помощью задержки Sleep(50).

Рисунок : заполнение матрицы и вывод ее на экран



П

Рисунок : заполнение матрицы и вывод ее на экран

Принцип работы: от 0 до size/2 выполняются циклы. Вначале алгоритм проходит по верхней стороне, там указатель увеличивается на единицу, далее по правой, указатель увеличивается на значение, равное стороне матрицы, после по нижней стороне указатель уменьшается на единицу, и в конце - по левой стороне, где указатель снова уменьшается на значение, равное стороне матрицы. По завершению круга указатель смещается на единицу вниз и вперед.

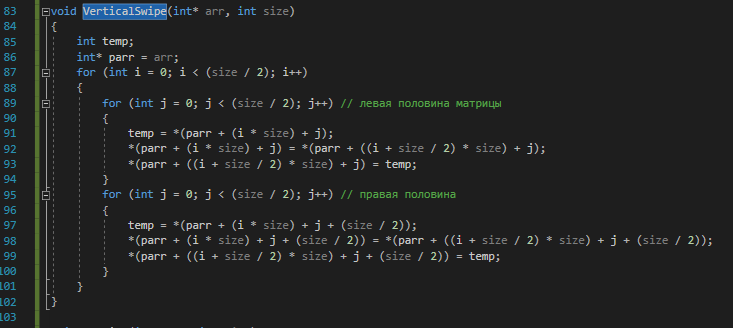
Следующий пункт это перестановка блоков матрицы. Я использовал перестановку по вертикали.

Рисунок : перестановка блоков

Здесь циклы разделены на правую и левую часть матрицы. Каждый выполняет следующие действия: запоминает значение элемента верхней половины и замещает его элементом из нижней половины, позиция которого соответствует ему в нижнем блоке.

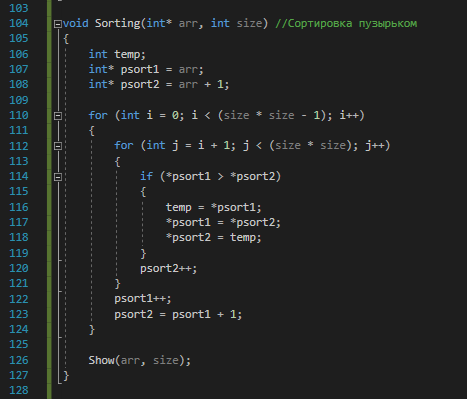
Далее идет сортировка массива элементов. Я выбрал сортировку пузырьком.

Рисунок : сортировка массива

Принцип работы это сравнение текущего элемента с следующим и если текущий меньше, то заменяем его на следующий. После завершения сортировки снова выводим матрицу на экран.

Финальный пункт это операции над каждым из элементов матрицы. Всего есть 4 типа операции, это: сложение, вычитание, умножение, деление.

Реализованы они перебором каждого элемента матрицы и выполнение действия с ним с помощью операций (+=, -=, \*=, /=). В конце снова выводим массив на экран.

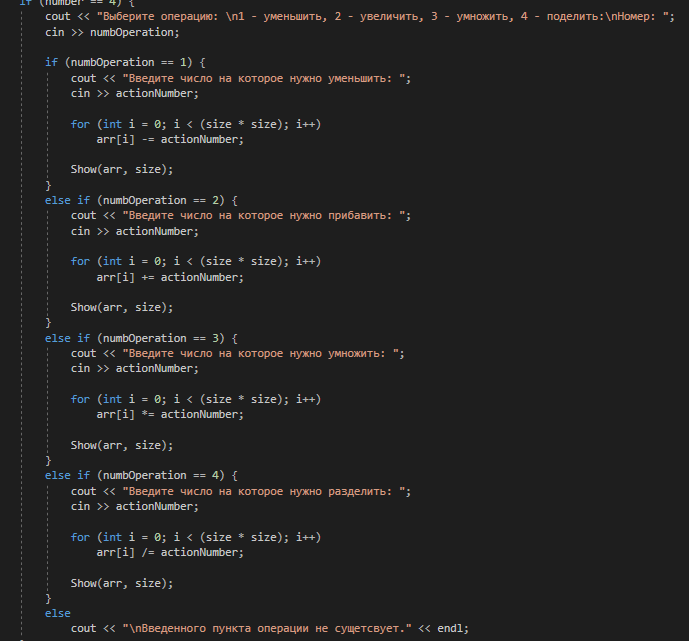


Рисунок : операции над элементами матрицы

Выводы.

Была написана программа реализующая способы работы с массивами посредством арифметики указателей, что дало понять и закрепить на практике их удобство работы с данными.

**4. ЧЕТВЕРТАЯ ПРАКТИЧЕСКАЯ РАБОТА**

**(ТЕКСТОВЫЕ СТРОКИ КАК МАССИВЫ СИМВОЛОВ)**

Цель работы.

Практические навыки работы с строками как массивом символов.

Постановка задачи.

Необходимо написать программу, которая реализует поставленную задачу:

1)    С клавиатуры или с файла (\*) (пользователь сам может выбрать способ ввода) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.

2)    Необходимо отредактировать входной текст:

·        удалить лишние пробелы;

·        удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков (обратите внимание, что «…» - корректное использование знака) в тексте);

·        исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр);

3) Выполнить задание по варианту:

|  |  |
| --- | --- |
| 1 | Вывести на экран слова последовательности в обратном порядке. |
| 2 | Вывести на экран слова последовательности в алфавитном порядке. |
| 3 | Вывести на экран слова последовательности, не содержащие цифр. |
| 4 | Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы. |
| 5 | Вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз. |
| 6 | После окончания ввода последовательности вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры. |

4) Выполнить задание по варианту:

|  |  |
| --- | --- |
| 1 | Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой. |
| 2 | Вывести на экран количество символов в каждом слове исходной последовательности. |
| 3 | Вывести на экран ту же последовательность, удалив из всех слов заданный набор букв и (или) цифр. |
| 4 | Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре. |
| 5 | Вывести на экран ту же последовательность, переместив все цифры, содержащиеся в словах, в конец соответствующих слов. |
| 6 | Вывести все слова исходной последовательности на экран вертикально. |
| 7 | Вывести на экран  все слова последовательности в две или три колонки (в зависимости от количества слов) с выравниванием слов по правой границе колонки. |

5)  Необходимо найти подстроку, которую введёт пользователь в имеющейся строке. Реализуйте два алгоритма: первый алгоритма – Линейный поиск, а второй алгоритм согласно вашему номеру в списке. Четные номера должны реализовать алгоритм КНМ, а нечетные – Бойера-Мура. (\*)

Выполнение работы.

В начале программы пользователю дается выбор чтения текста для его последующего редактирования. По задаче должно быть реализовано два способа: ввод текста вручную и чтение текста из файла.

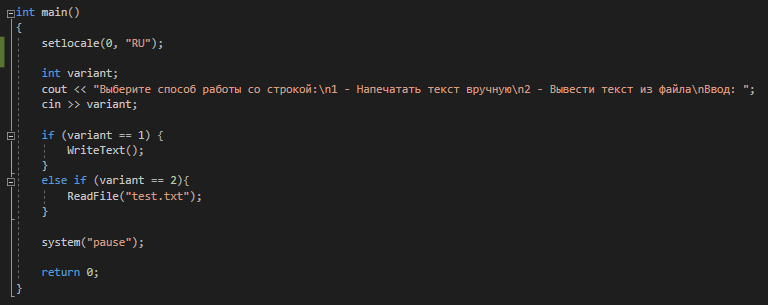
В первом варианте пользователю дается возможность ввести текст вручную с помощью cin.getline().

Рисунок : выбор чтения текста

Во втором варианте программа создает поток файла, открывает файл по его имени, делает проверку на то удается ли открыть и записывает в char наш текст.

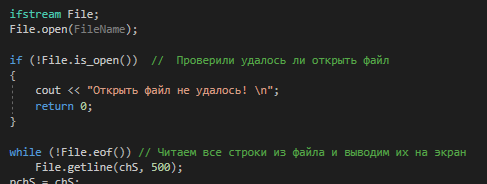


Рисунок :чтение из файла

Далее операции форматирования строки одинаковы для двух способов введенного текста.

Первым делом мы удаляем из строки все повторяющиеся символы (за исключением букв) с помощью функции *RemovingDuplicateCharacters*.

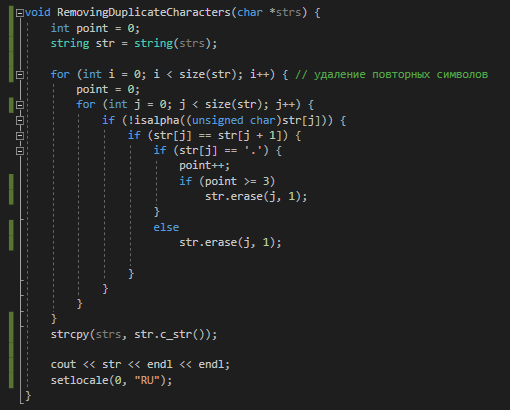


Рисунок 17: удаление повторояющихся элементов

В функцию передается строка. Преобразовываем ее в string. В цикле идет проверка на то является ли элемент буквой, если нет, значит сравниваем его с следующим элементом и если они равны, мы удаляем текущий элемент.

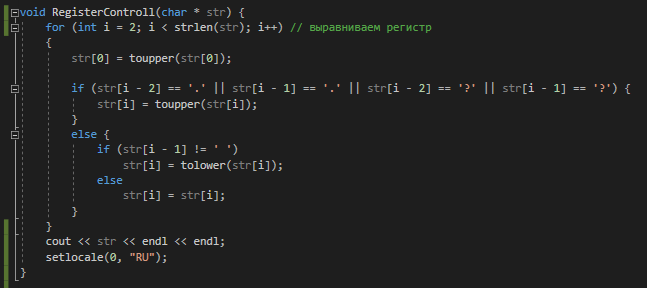
Далее вызывается функция исправления регистра строки, в которую так же передается текст.

Рисунок : исправление регистра букв

Все операции выполняются с помощью встроенной функции *toupper/tolower.* Первый символ всегда является заглавной буквой. Далее в цикле идет проверка предыдущих символов на то является ли текущий символ новым предложением. Если да, то делаем букву заглавной. Если нет, то идет проверка на то, является ли слово изначально с большой буквы, допустим чтобы не допустить ошибку в словах имени, городов и т.д. Все остальные буквы изменяем на строчные.

Т.к. у меня 5 вариант мне необходимо было сделать функцию, которая выводит на экран только те слова, в которых первая буква слова встречается в нем еще минимум раз.

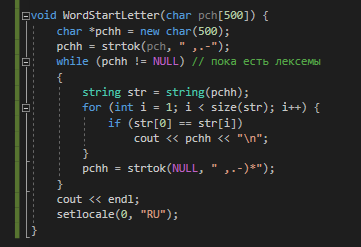


Рисунок : Вывод слов в которых первая буква встречается еще раз

В начале форматируем строку в массив слов(лексемы). Далее для каждого слова идет проверка на то, есть ли в слове повторяющиеся буквы. После завершения проверки удаляем слово из массива и идет новая проверка следующего слова.

Следующая функция ставит цифры в конец слова, если таковые есть и они стоят не в конце.

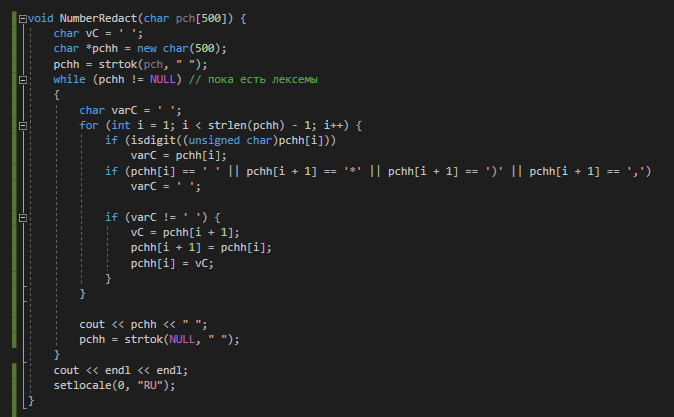
В начале преобразует строку в массив слов. В цикле проверка: если элемент цифра, то varC равна этой цифре. Пока след. элемент не символ ( \*),), элемент не перезаписывается, иначе мы его удаляем. Пока в varC не пустой, заменяем след. элемент на текущий и таким образом доходим до конца слова.

Рисунок : перемещение цифр в конец слова

*В программе используется частая смена кодировки из-за проблем с выводом русского текста в редактор, с английским текстом и без таких присвоений разных кодировок работает хорошо*.

Выводы.

С текстовыми строками, представленными в виде массива символов, достаточно легко выполнять различные преобразования.

**КУРСОВАЯ работа**

Цель работы.

Совместить все программы практических работ в одну и реализовать интерфейс переключения между ними.

Постановка задачи.

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

Выполнение работы.

Все методы программ помещены в один проект, при чем выше главного метода main, который является агрегатором всех программ.

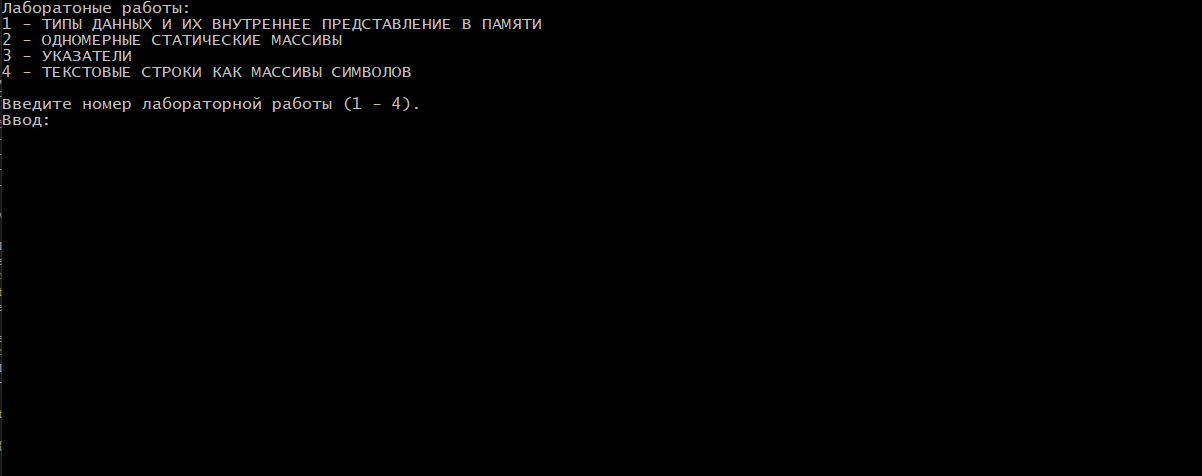
Если мы завершаем подпрограмму, то у нас снова появляется выбор запуска программы.

Рисунок : предлагаемы выбор программ

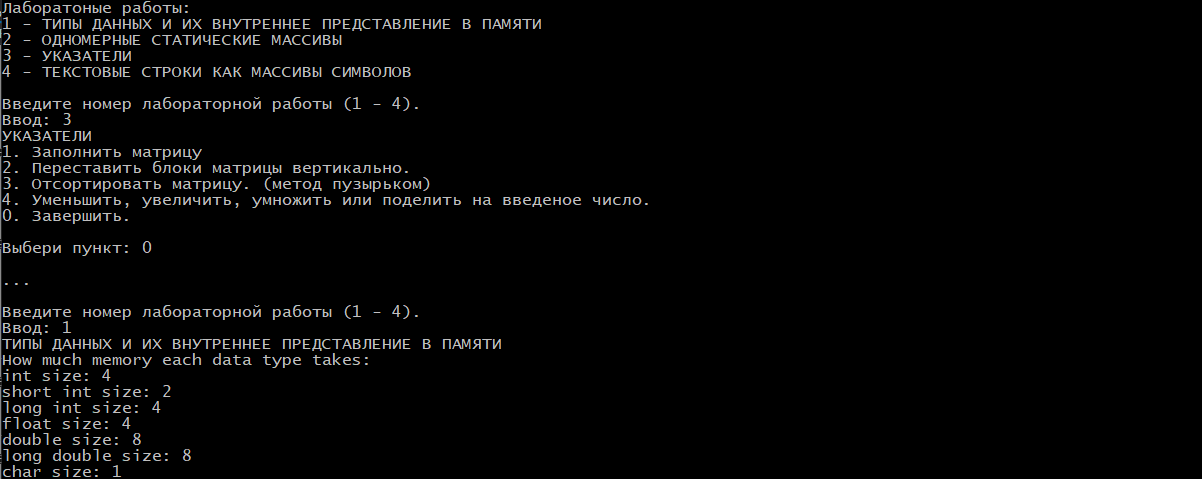


Рисунок : выбор следующей программы

**Выводы.**

Такое решение удобнее, чем закрывать одну программу и открывать другую, чтобы воспроизвести ее.

**заключение**

Курс был довольно насыщен практической частью закрепления пройденного материала, что хорошо сказалось на обучении и усвоении алгоритмов программирования на языке С++. Было изучено 8 основых разделов в которых выполнены все тестовые точки. Написано 4 программы для практических точек курса и в заключении собран курсовой проект состоящий из одной программы, выполняющая роль агрегатора всех практических программ, в которой можно переключаться между ними.

**список использованных источников**

•https://stepik.org/ **приложение А**

**Программный код**

**Первая практическая работа:**

#include <iostream>

#include <string>

using namespace std;

int main()

{

cout << "How much memory each data type takes: \n";

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

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

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

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

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

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

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

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

cout << "\n";

cout << "Displaying the binary representation of an integer.\n";

string repeat = "no";

do

{

cout << "Enter int value: ";

int value = 0;

cin >> value;

int mask = 1 << 32 - 1;

for (int i = 0; i < 32; i++) {

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

value <<= 1;

if (i % 31 == 0)

{

putchar(' ');

}

if (i % 8 == 0)

{

putchar(' ');

}

}

cout << endl;

cout << "Enter another value? (no/yes) - ";

cin >> repeat;

cout << endl;

} while (repeat == "yes");

cout << "Displaying the binary representation of an float.\n";

repeat = "no";

do

{

cout << "Enter float value: ";

union {

float f;

int i;

} u;

cin >> u.f;

int mask = 1 << 32 - 1;

for (int i = 0; i < 32; i++)

{

putchar(u.i & mask ? '1' : '0');

u.i <<= 1;

if (i % 31 == 0)

{

putchar(' ');

}

if (i % 8 == 0)

{

putchar(' ');

}

}

cout << endl;

cout << "Enter another value? (no/yes) - ";

cin >> repeat;

cout << endl;

} while (repeat == "yes");

cout << "Displaying the binary representation of an double.\n";

repeat = "no";

do

{

cout << "Enter double value: ";

int discharge = 0;

union {

double d;

int i[2];

} u;

cin >> u.d;

int mask = 1 << 31;

for (int x = 0; x < 2;) {

for (int i = 0; i < 32; i++)

{

putchar(u.i[x] & mask ? '1' : '0');

u.i[x] <<= 1;

discharge++;

if (discharge == 1)

{

putchar(' ');

}

if (discharge == 12)

{

putchar(' ');

}

}

x++;

}

cout << endl;

cout << "Enter another value? (no/yes) - ";

cin >> repeat;

cout << endl;

} while (repeat == "yes");

cout << "End\n";

return 0;

}

**Вторая практическая работа:**

#include <iostream>

#include <chrono>

#include <ctime>

using namespace std;

typedef chrono::high\_resolution\_clock Clock;

typedef chrono::duration<double> sec;

int arrInt[100]; // изначальный массив

int arrIntSort[100]; // отсортированный массив

int min;

int max;

int repeat = 0;

float average = 0;

Clock::time\_point timeStart;

Clock::time\_point timeEnd;

void CreateArray(); // создаем массив

void SortingArray(); // сортируем массив

void FindMin(); // находим минимальные элементы массивов

void FindMax(); // находим максимальные элементы массивов

void AverageValue(); // среднее значение мин/мах

void FindIdenticalElements(); // поиск одинаковых элементов равных среднему значению мин/мах

void SmallerNumbers(); // кол-во чисел меньших числа А

void BiggerNumbers(); // кол-во чисел больших числа Б

void BinaryNumberSearch(); // бинарный поиск введенного числа в массиве

void ReplacementElements(); // замена двух элементов местами

void SortingShaker(); // для индивидуального задания

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

do {

CreateArray();

SortingArray();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

FindMin();

FindMax();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

AverageValue();

FindIdenticalElements();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

SmallerNumbers();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

BiggerNumbers();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

BinaryNumberSearch();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

ReplacementElements();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

do {

SortingShaker();

repeat = 0;

cout << "Повторить? (0 - нет / 1 - да): ";

cin >> repeat;

cout << endl;

} while (repeat == 1);

cout << "End.";

}

void CreateArray() {

for (int i = 0; i < size(arrInt); i++) {

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

arrIntSort[i] = arrInt[i]; // копия массива

}

}

void SortingArray() {

int temp; // временная переменная для обмена элементов местами

timeStart = Clock::now();

for (int i = 0; i < size(arrIntSort) - 1; i++) { // Сортировка массива пузырьком

for (int j = 0; j < size(arrIntSort) - i - 1; j++) {

if (arrIntSort[j] > arrIntSort[j + 1]) {

// меняем элементы местами

temp = arrIntSort[j];

arrIntSort[j] = arrIntSort[j + 1];

arrIntSort[j + 1] = temp;

}

}

}

timeEnd = Clock::now();

cout << sec(timeEnd - timeStart).count() << " время затраченное на сортировку массива. (Bubble sort)\n" << endl;

for (int i = 0; i < size(arrIntSort); i++) {

cout << arrIntSort[i] << " ";

}

cout << " отсортированный массив" << endl;

cout << endl;

}

void FindMin() {

min = 100;

timeStart = Clock::now();

for (int i = 0; i < size(arrInt) - 1; i++) { // поиск минимального в неотсортированном массиве

if (arrInt[i] < min) {

min = arrInt[i];

}

}

timeEnd = Clock::now();

cout << "Минимальный элемент:" << min << ".\nВремени на нахождение потребовалось в неотсортированном массиве: " << sec(timeEnd - timeStart).count() << " сек." << endl;

min = 100;

timeStart = Clock::now();

for (int i = 0; i < size(arrIntSort) - 1; i++) { // поиск минимального в отсортированном массиве

if (arrIntSort[i] < min) {

min = arrIntSort[i];

}

}

timeEnd = Clock::now();

cout << "Времени на нахождение потребовалось в отсортированном: " << sec(timeEnd - timeStart).count() << " сек." << endl;

cout << endl;

}

void FindMax() {

max = -100;

timeStart = Clock::now();

for (int i = 0; i < size(arrInt) - 1; i++) { // поиск максимального в неотсортированном массиве

if (arrInt[i] > max) {

max = arrInt[i];

}

}

timeEnd = Clock::now();

cout << "Минимальный элемент:" << max << ".\nВремени на нахождение потребовалось в неотсортированном массиве: " << sec(timeEnd - timeStart).count() << " сек." << endl;

max = -100;

timeStart = Clock::now();

for (int i = 0; i < size(arrIntSort) - 1; i++) { // поиск максимального в отсортированном массиве

if (arrIntSort[i] > max) {

max = arrIntSort[i];

}

}

timeEnd = Clock::now();

cout << "Времени на нахождение потребовалось в отсортированном: " << sec(timeEnd - timeStart).count() << " сек." << endl;

cout << endl;

}

void AverageValue() {

average = (min + max) / 2;

cout << "Среднее значение минимального и максимального элемента массива: " << average << endl;

}

void FindIdenticalElements() {

int countIdentidal = 0;

cout << "Индексы элементов равных среднему MIN/MAX: ";

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

if (arrIntSort[i] == average) {

cout << i << " ";

countIdentidal++;

}

}

cout << "\nКоличество элементов равных среднему MIN/MAX: " << countIdentidal << endl;

cout << endl;

}

void SmallerNumbers() {

int inputValue = 0;

cout << "Введите число до которого будет подсчет меньших чисел: ";

cin >> inputValue;

int count = 0;

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

if (arrIntSort[i] < inputValue) {

count++;

}

}

cout << "Количество чисел меньших числа " << inputValue << ": " << count << endl;

cout << endl;

}

void BiggerNumbers() {

int inputValue = 0;

cout << "Введите число до которого будет подсчет больших чисел: ";

cin >> inputValue;

int count = 0;

for (int i = 0; i < size(arrIntSort); i++) {

if (arrIntSort[i] > inputValue) {

count++;

}

}

cout << "Количество чисел меньших числа " << inputValue << ": " << count << endl;

cout << endl;

}

void BinaryNumberSearch() {

bool flag = false;

int l = -99; // левая граница

int r = 98; // правая граница

int mid;

int valueInput;

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

cin >> valueInput; // считываем ключ

timeStart = Clock::now();

while ((l <= r) && (flag != true)) {

mid = (l + r) / 2; // считываем срединный индекс отрезка

if (arrIntSort[mid] == valueInput)

flag = true; //проверяем ключ со серединным элементом

if (arrIntSort[mid] > valueInput)

r = mid - 1; // проверяем, какую часть нужно отбросить

else

l = mid + 1;

}

timeEnd = Clock::now();

if (flag)

cout << "Введенный элемент " << valueInput << " существует в данном массиве." << endl;

else

cout << "Введенного элемента в массиве нет" << endl;

cout << endl;

cout << sec(timeEnd - timeStart).count() << " время затраченное на бинарный поиск элемента." << endl;

{

timeStart = Clock::now();

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

if (arrIntSort[i] == valueInput) {

max = arrIntSort[i];

break;

}

}

timeEnd = Clock::now();

cout << sec(timeEnd - timeStart).count() << " время затраченное на обычный поиск элемента." << endl;

}

cout << endl;

}

void ReplacementElements() {

int elemFirst;

int elemSecond;

cout << "Введите индексы элементов массива которые нужно поменять местами: " << endl;

cin >> elemFirst >> elemSecond;

cout << endl;

timeStart = Clock::now();

int elemTemp = arrIntSort[elemFirst];

arrIntSort[elemFirst] = arrIntSort[elemSecond];

arrIntSort[elemSecond] = elemTemp;

timeEnd = Clock::now();

cout << "Измененный массив:" << endl;

for (int i = 0; i < size(arrIntSort); i++) {

cout << arrIntSort[i] << " ";

}

cout << endl << sec(timeEnd - timeStart).count() << " времени потребовалось на замену элементов." << endl;

}

void SortingShaker() {

for (int i = 0; i < size(arrInt); i++) {

arrIntSort[i] = arrInt[i]; // копия массива

}

bool sort\_or\_not = true;

timeStart = Clock::now();

do {

sort\_or\_not = true;

for (int i = 0; i < size(arrIntSort); i++) {

if (arrIntSort[i] > arrIntSort[i + 1]) {

swap(arrIntSort[i], arrIntSort[i + 1]);

sort\_or\_not = false;

}

}

for (int i = 4; i >= 1; i--) {

if (arrIntSort[i] < arrIntSort[i - 1]) {

swap(arrIntSort[i], arrIntSort[i - 1]);

sort\_or\_not = false;

}

}

} while (sort\_or\_not == false);

timeEnd = Clock::now();

cout << "Измененный массив:" << endl;

for (int i = 0; i < size(arrIntSort); i++) {

cout << arrIntSort[i] << " ";

}

cout << endl << sec(timeEnd - timeStart).count() << " времени потребовалось на сортировку элементов. (Shaker Sort)" << endl;

}

**Третья практическая работа:**

#include <iostream>

#include <stdio.h>

#include <Windows.h>

#include <time.h>

#include <iostream>

using namespace std;

void Show(int\* arr, int size)

{

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

{

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

cout << arr[(i \* size) + j] << ' ';

cout << '\n';

}

}

void SpiralFilling(int\* arr, short int size)

{

int\* parr = arr;

CONSOLE\_SCREEN\_BUFFER\_INFO info\_y;

GetConsoleScreenBufferInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &info\_y);

int y\_coord = info\_y.dwCursorPosition.Y + 1;

COORD position;

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

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

{

//Верхняя сторона

for (int j = 0; j < (size - i \* 2 - 1); j++)

{

\*arr = rand() % (size \* size) + 1;

position.X = (j + i) \* 4;

position.Y = y\_coord + i;

SetConsoleCursorPosition(hConsole, position);

cout << \*arr;

Sleep(50);

arr++;

}

//Правая сторона

for (int j = 0; j < (size - i \* 2 - 1); j++)

{

\*arr = rand() % (size \* size) + 1;

position.X = (size - i - 1) \* 4;

position.Y = y\_coord + (i + j);

SetConsoleCursorPosition(hConsole, position);

cout << \*arr;

Sleep(50);

arr += size;

}

//Нижняя сторона

for (int j = (size - i \* 2 - 1); j > 0; j--)

{

\*arr = rand() % (size \* size) + 1;

position.X = (i + j) \* 4;

position.Y = y\_coord + size - 1 - i;

SetConsoleCursorPosition(hConsole, position);

cout << \*arr;

Sleep(50);

arr--;

}

//Левая сторона

for (int j = (size - i \* 2 - 1); j > 0; j--)

{

\*arr = rand() % (size \* size) + 1;

position.X = i \* 4;

position.Y = y\_coord + (j + i);

SetConsoleCursorPosition(hConsole, position);

cout << \*arr;

Sleep(50);

arr -= size;

}

arr += size + 1;

}

position.X = 0;

position.Y = y\_coord + size;

SetConsoleCursorPosition(hConsole, position);

}

void VerticalSwipe(int\* arr, int size)

{

int temp;

int\* parr = arr;

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

{

for (int j = 0; j < (size / 2); j++) // левая половина матрицы

{

temp = \*(parr + (i \* size) + j);

\*(parr + (i \* size) + j) = \*(parr + ((i + size / 2) \* size) + j);

\*(parr + ((i + size / 2) \* size) + j) = temp;

}

for (int j = 0; j < (size / 2); j++) // правая половина

{

temp = \*(parr + (i \* size) + j + (size / 2));

\*(parr + (i \* size) + j + (size / 2)) = \*(parr + ((i + size / 2) \* size) + j + (size / 2));

\*(parr + ((i + size / 2) \* size) + j + (size / 2)) = temp;

}

}

}

void Sorting(int\* arr, int size) //Сортировка пузырьком

{

int temp;

int\* psort1 = arr;

int\* psort2 = arr + 1;

for (int i = 0; i < (size \* size - 1); i++)

{

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

{

if (\*psort1 > \*psort2)

{

temp = \*psort1;

\*psort1 = \*psort2;

\*psort2 = temp;

}

psort2++;

}

psort1++;

psort2 = psort1 + 1;

}

Show(arr, size);

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

int number = 5;

int arr[100];

int size, numbOperation, actionNumber;

cout << "1. Заполнить матрицу\n";

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

cout << "3. Отсортировать матрицу. (метод пузырьком)\n";

cout << "4. Уменьшить, увеличить, умножить или поделить на введеное число.\n";

cout << "0. Завершить.\n";

cout << endl;

while (number != 0)

{

cout << "Выбери пункт: ";

cin >> number;

cout << '\n';

if (number == 1) { // создание матрицы

cout << "Выберите число: 6, 8, 10: ";

cin >> size;

if ((size == 6) || (size == 8) || (size == 10))

SpiralFilling(arr, size);

else

cout << "Вы ввели некорректное число\n";

}

if (number == 2) { // перемещение по вертикали

VerticalSwipe(arr, size);

Show(arr, size);

}

if (number == 3) { // сортировка

Sorting(arr, size);

}

if (number == 4) {

cout << "Выберите операцию: \n1 - уменьшить, 2 - увеличить, 3 - умножить, 4 - поделить:\nНомер: ";

cin >> numbOperation;

if (numbOperation == 1) {

cout << "Введите число на которое нужно уменьшить: ";

cin >> actionNumber;

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

arr[i] -= actionNumber;

Show(arr, size);

}

else if (numbOperation == 2) {

cout << "Введите число на которое нужно прибавить: ";

cin >> actionNumber;

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

arr[i] += actionNumber;

Show(arr, size);

}

else if (numbOperation == 3) {

cout << "Введите число на которое нужно умножить: ";

cin >> actionNumber;

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

arr[i] \*= actionNumber;

Show(arr, size);

}

else if (numbOperation == 4) {

cout << "Введите число на которое нужно разделить: ";

cin >> actionNumber;

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

arr[i] /= actionNumber;

Show(arr, size);

}

else

cout << "\nВведенного пункта операции не сущетсвует." << endl;

}

}

cout << "..." << endl;

}

**Четвертая практическая работа:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include <vector>

using namespace std;

void RemovingDuplicateCharacters(char \*strs) {

int point = 0;

string str = string(strs);

for (int i = 0; i < size(str); i++) { // удаление повторных символов

point = 0;

for (int j = 0; j < size(str); j++) {

if (!isalpha((unsigned char)str[j])) {

if (str[j] == str[j + 1]) {

if (str[j] == '.') {

point++;

if (point >= 3)

str.erase(j, 1);

}

else

str.erase(j, 1);

}

}

}

}

strcpy(strs, str.c\_str());

cout << str << endl << endl;

setlocale(0, "RU");

}

void RegisterControll(char \* str) {

for (int i = 2; i < strlen(str); i++) // выравниваем регистр

{

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

if (str[i - 2] == '.' || str[i - 1] == '.' || str[i - 2] == '?' || str[i - 1] == '?') {

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

}

else {

if (str[i - 1] != ' ')

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

else

str[i] = str[i];

}

}

cout << str << endl << endl;

setlocale(0, "RU");

}

void WordStartLetter(char pch[500]) {

char \*pchh = new char(500);

pchh = strtok(pch, " ,.-");

while (pchh != NULL) // пока есть лексемы

{

string str = string(pchh);

for (int i = 1; i < size(str); i++) {

if (str[0] == str[i])

cout << pchh << "\n";

}

pchh = strtok(NULL, " ,.-)\*");

}

cout << endl;

setlocale(0, "RU");

}

void NumberRedact(char pch[500]) {

char vC = ' ';

char \*pchh = new char(500);

pchh = strtok(pch, " ");

while (pchh != NULL) // пока есть лексемы

{

char varC = ' ';

for (int i = 1; i < strlen(pchh) - 1; i++) {

if (isdigit((unsigned char)pchh[i]))

varC = pchh[i];

if (pchh[i] == ' ' || pchh[i + 1] == '\*' || pchh[i + 1] == ')' || pchh[i + 1] == ',')

varC = ' ';

if (varC != ' ') {

vC = pchh[i + 1];

pchh[i + 1] = pchh[i];

pchh[i] = vC;

}

}

cout << pchh << " ";

pchh = strtok(NULL, " ");

}

cout << endl << endl;

setlocale(0, "RU");

}

bool ReadFile(const char \*FileName)

{

char chS[400];

char copyStr[500];

char \*pchS;

ifstream File;

File.open(FileName);

if (!File.is\_open()) // Проверили удалось ли открыть файл

{

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

return 0;

}

while (!File.eof()) // Читаем все строки из файла и выводим их на экран

File.getline(chS, 500);

pchS = chS;

cout << "Исходный текст:\n" << pchS << endl << endl;

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

cout << "\n\*\*\*В этой строке удалены все повторные символы\*\*\*\n";

RemovingDuplicateCharacters(pchS); // удаление повторных символов

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

cout << "\n\*\*\*В этой строке выравнен регистр букв\*\*\*\n";

RegisterControll(pchS); // выравниваем регистр

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

cout << "\n\*\*\*Слова, в которых есть буквы такие же как и первая буква слова\*\*\*\n";

strcpy(copyStr, pchS);

WordStartLetter(copyStr); // выводим слова, в которых есть буквы такие же как и первая буква слова

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

cout << "\n\*\*\*В этой все буквы в словах передвинуты в конец слова\*\*\*\n";

NumberRedact(pchS); // перемещаем буквы в словах в конец слова

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

File.close(); // Закрываем файл

return 1;

}

void WriteText() {

cin.get();

char chS[500];

char copyStr[500];

char \*pchS;

cout << "Введите текст содержащий от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.\nВвод: ";

cin.getline(chS, 500);

pchS = chS;

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

cout << "\n\*\*\*В этой строке удалены все повторные символы\*\*\*\n";

setlocale(0, ".866");

RemovingDuplicateCharacters(pchS); // удаление повторных символов

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

cout << "\n\*\*\*В этой строке выравнен регистр букв\*\*\*\n";

setlocale(0, ".866");

RegisterControll(pchS); // выравниваем регистр

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

cout << "\n\*\*\*Слова, в которых есть буквы такие же как и первая буква слова\*\*\*\n";

strcpy(copyStr, pchS);

setlocale(0, ".866");

WordStartLetter(copyStr); // выводим слова, в которых есть буквы такие же как и первая буква слова

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

cout << "\n\*\*\*В этой строке все буквы в словах передвинуты в конец слова\*\*\*\n";

setlocale(0, ".866");

NumberRedact(pchS); // перемещаем буквы в словах в конец слова

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

}

int main()

{

setlocale(0, "RU");

int variant;

cout << "Выберите способ работы со строкой:\n1 - Напечатать текст вручную\n2 - Вывести текст из файла\nВвод: ";

cin >> variant;

if (variant == 1) {

WriteText();

}

else if (variant == 2){

ReadFile("test.txt");

}

system("pause");

return 0;

}