**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2373 |  | Суднищиков М.К |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Разработать алгоритм и написать программу, которая позволяет, понять сколько памяти занимают основные типы данных, понять, как работает двоичное представление числа в языке C++.

Изучение понятия массива и их внутреннего представления в памяти, получение практических навыков работы с массивами. Знакомство и методами сортировки массива.

Познакомиться с двумерными статическими массивами, их объявлением, научимся обрабатывать данные двумерных статических массивов с помощью арифметики указателей, применим сортировку массива, реализованную с указателями. Написать программу, создающую матрицу размера N = (6, 8, 10) и заполняющую её рандомными значениями по заданным схемам, меняющую блоки матрицы местами, увеличивающую, уменьшающую, умножающую и делящею элементы матрицы на введенное пользователем число.

Проведение работы над типом данных char, действий над ним, и поиска подстрок.

**Основные теоретические положения.**

Бит информации – символ или сигнал, который может принимать два значения: включено или выключено, да или нет, высокий или низкий, заряженный или незаряженный; в двоичной системе исчисления это 1 (единица) или 0 (ноль).

Тип данных – тип никогда не изменяется (не следует путать со статическими объектами данных). При объявлении переменной в коде необходимо явно указать его тип или использовать ключевое слово, чтобы указать компилятору выводить тип из инициализатора. При объявлении функции в коде необходимо указать тип каждого аргумента и его возвращаемое значение. В отличие от некоторых других языков, в C++ нет универсального базового типа, от которого наследуются все остальные типы. Язык включает множество фундаментальных типов, также известных как встроенные типы. Сюда входят числовые типы, такие как int, double, longbool, а также char типы и wchar\_t типы для символов ASCII и ЮНИКОД соответственно. Большинство целочисленных фундаментальных типов (за исключением bool, doublewchar\_tи связанных типов) имеют unsigned версии, которые изменяют диапазон значений, которые может хранить переменная. Например, int32-разрядное целое число со знаком может представлять значение от -2 147 483 648 до 2 147 483 647. Значение unsigned int, которое также хранится как 32-разрядные, может хранить значение от 0 до 4294 967 295. Общее количество возможных значений в каждом случае одинаково, отличается только диапазон.

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнить однообразные действия, им дают одно имя, а различают по порядковому номеру (индексу). Это дает возможность компактно записать множество операций с использованием циклов.

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.

Аналогом одномерного массива из математики может служить последовательность некоторых элементов с одним индексом: *ai* при *i* = 0, 1, 2,…n – одномерный вектор. Каждый элемент такой последовательности представляет собой некоторое значение определенного типа данных. Наглядно одномерный массив можно представить как набор пронумерованных ячеек, в каждой из которых содержится определенное значение.

Массив – это индексированная последовательность однотипных элементов с заранее определенным количеством элементов. Массивы используются для обработки большого количества однотипных данных.

Имя массива является указателем.

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

Многомерный массив – это массив, характеризованный двумя или более измерениями, причем доступ к каждому элементу такого массива осуществляется с помощью определенной комбинации двух или более индексов.

Двумерный массив – форма многомерного массива, элементы определяются двумя индексами.

Указатель – переменная служащая для хранения адреса.

Указатель определяется:

<тип данных> \*<имя переменной>

Тип данных указателя должен совпадать с типом данных объекта, на который он указывает. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

К указателям можно применять некоторые арифметические операции. К таким операциям относятся: сложение, вычитание, инкремент и декремент (+, -, ++, --). Результат применения этих операций к указателям отличается от результата применения их к разыменованным указателям или числам, так как изменяет не значение, а адрес.

Текстовые строки представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов.

Ввод текста с клавиатуры можно осуществлять разными способами, каждый из которых имеет определенные особенности:

1. Непосредственное чтение текстовых строк из потока вывода

Осуществляется до первого знака пробела. Такой способ чтения обеспечивает ввод символов до первого пробельного символа, а не до конца строки. Остальные символы введенного с клавиатуры остаются в потоке ввода и могут быть прочитаны из него следующими операторами .

1. Функции gets / gets\_s

Используются для того, чтобы прочесть всю строку полностью. Функция gets имеет один параметр, соответствующий массиву символов, в который осуществляется чтение. Вторая функция (gets\_s) имеет второй параметр, задающий максимальную длину массива символов.

Ввод текста, длина которого (вместе с нулевым символом) превышает значение второго параметра (то есть длины символьного массива) приводит к возникновению ошибки при выполнении программы

Предпочтительно использование функции потока ввода cin.getline.

Функция strlen(char \*s) – возвращает фактическую длину текстовой строки, хранящейся в символьном массиве s (см. аналог my\_StrLen(char \*S)).

Функция strcpy(char \*dest, char \*source) – копирует содержимое строки source в строку dest.

Функция strcat(char \*s1, char \*s2) – добавляет содержимое строки s2 в конец строки s1.

Функция strcmp(char \*s1, char \*s2) – осуществляет лексикографическое сравнение строк s1 и s2. Возвращает значение 0, если строки одинаковы (равны), значение большее 0 при s1 > s2 и отрицательное значение при s1 < s2.

Широкое применение в программировании получила специальная форма двумерных массивов символов, которые можно рассматривать как одномерный массив строк, заканчивающихся нулевым символом. Определяются такие массивы ка char arr\_str [30] [21] (30 строк, каждая строка может содержать текст из максимум 20 символов + нулевой символ).

Алгоритм Бойера и Мура – это алгоритм поиска подстроки в строке, при котором первоначально строится таблица смещений для искомой подстроки, проверка начинается с последнего символа подстроки после совмещения начала строки и подстроки. Сначала строится таблица смещений для каждого символа. Затем исходная строка и шаблон совмещаются по началу, сравнение ведется по последнему символу. Если последние символы совпадают, то сравнение идет по предпоследнему символу и так далее. Если же символы не совпали, то шаблон смещается вправо, на число позиций взятое из таблицы смещений для символа из исходной строки, и тогда снова сравниваются последние символы исходной строки и шаблона. И так далее, пока не шаблон полностью не совпадет с подстрокой исходной строки, или не будет достигнут конец строки.

**Постановка задачи.**

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.

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 массив […] 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.

Необходимо разработать алгоритм и написать программу, которая:

1) Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка N случайными числами от 1 до N\*N по столбцам попеременно по направлению вниз вверх и по спирали от левого верхнего края к центру. Пользователь должен видеть процесс заполнения квадратичной матрицы.

2) Получает новую матрицу, переставляя блоки исходной матрицы.

3) Используя арифметику указателей, сортирует элементы матрицы.

4) Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число. Необходимо создать код, который будет:

1) С клавиатуры или с файла 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.

4) Выполнить задание по варианту: (1) Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.

5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. Реализуйте алгоритм Бойера-Мура.

**Выполнение работы.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

В моем коде используются циклы и простейшие типы данных.

Блок скриншотов работы программы

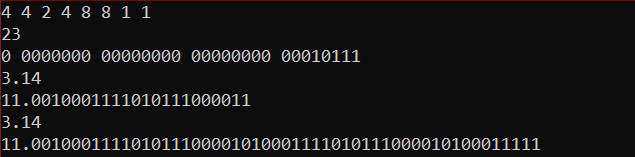


Рисунок 1.1. Результат программы .

**Результаты работы программы.**

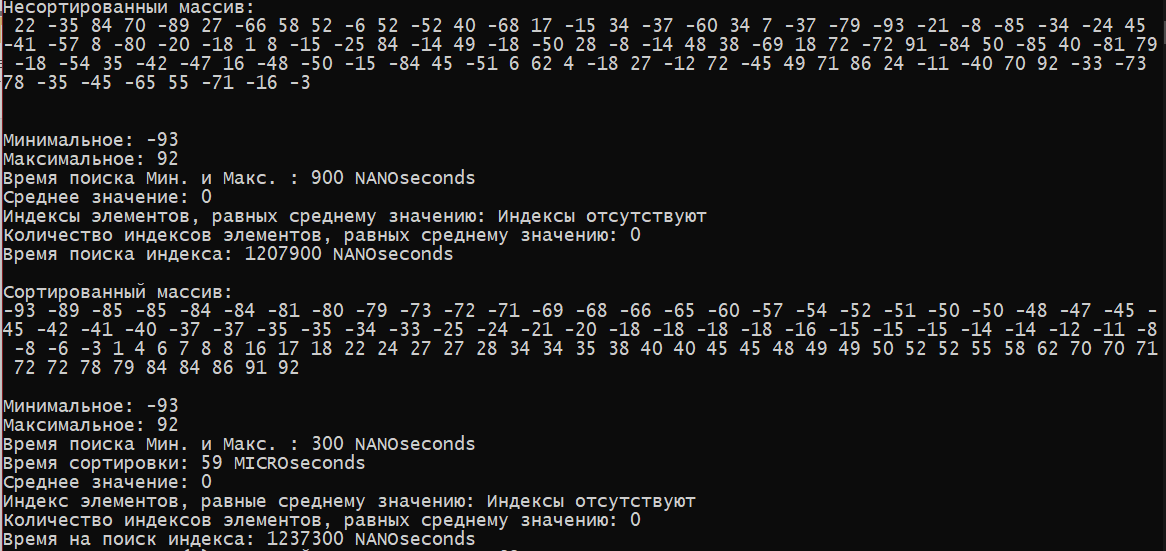


Рисунок 2.1. Результат программы (ч.1).

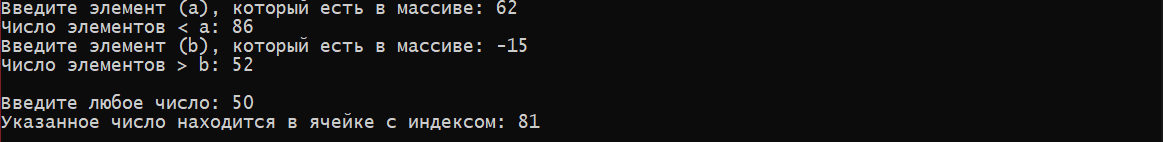


Рисунок 2.2. Результат программы (ч.2).

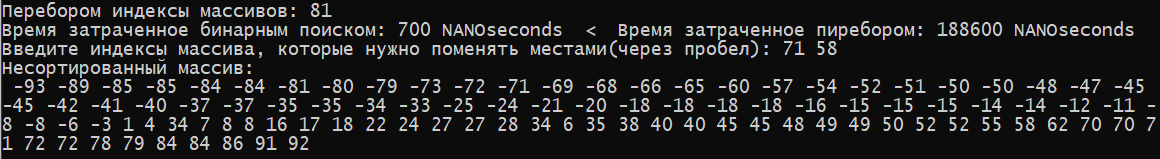


Рисунок 2.3. Результат программы (ч.3).

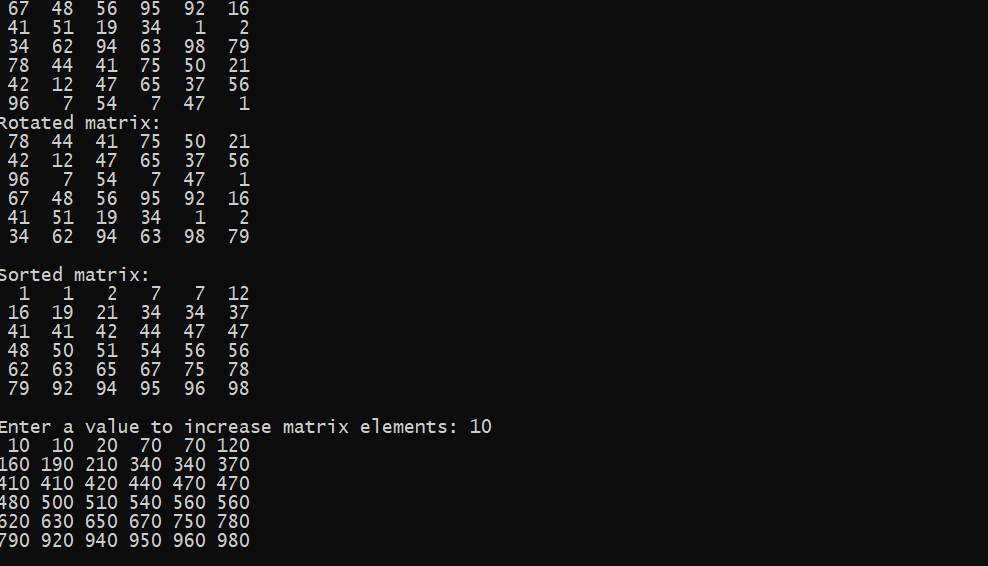
Для выполнения поставленной задачи работы использован язык программирования С++. Итоговый код программы представлен в приложении.

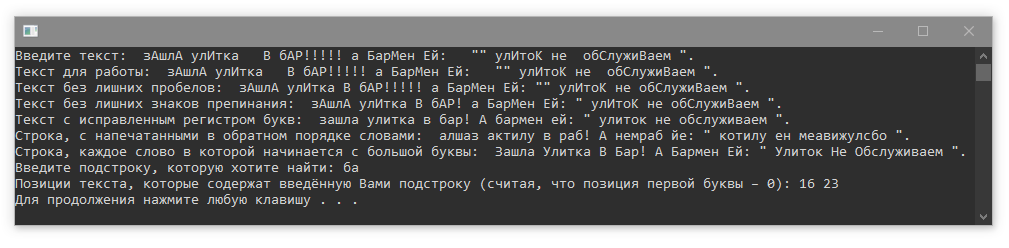
Для создания массива случайных значений в заданном диапазоне используем функцию srand().

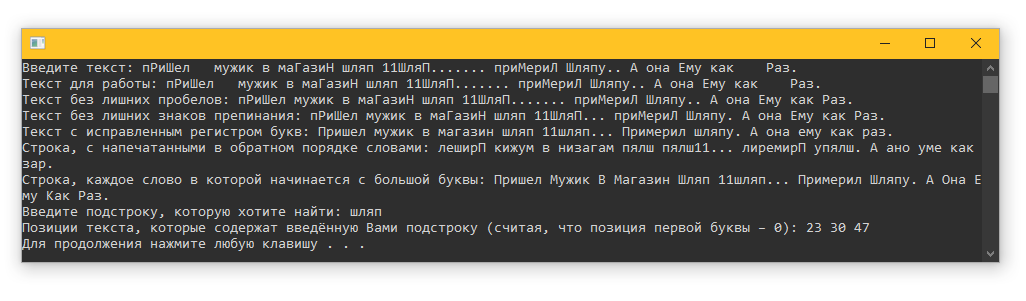
Для вывода массива в консоль соблюдая правильный порядок значений подключаем библиотеку <Windows.h>, используем функцию, управляющую курсором.

Для выполнения заданий используем арифметику указателей, передаём указатель на первый элемент массива и остальные необходимые в конкретном задании значения.









**Выводы.**

Мы вывели двоичное значение типа данных int и float. Понял как устроено двоичное представление числа в языке С++.

Во время выполнения данной работы были изучены массивы и их внутреннее представление в памяти. Также был получен навык работы с массивами и получено представление о работе алгоритмов сортировки. Полученная программа способна выводить на экран массив, сортировать его и производить другие операции над массивом, такие как: поиск, бинарный поиск, нахождение медианы и т.п.

В процессе работы было получено представление о работе с двумерными статическими массивами, сортировками, с использованием арифметики указателей. Написана программа, заполняющая двумя способами случайными элементами двумерный статический массив, отображающая в консоли процесс заполнения массива поэтапно, изменяющая этот массив.

В ходе лабораторной работы была проведена работа над типом данных char и произведён поиск подстрок из него.

Приложение А

рабочий код

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

#include <algorithm>

#include <cctype>

#include <cstdlib>

#include <time.h>

#include <chrono>

#include <vector>

#include <thread>

#include <iomanip>

#include <ctime>

using namespace std;

using namespace std::chrono;

int rrand(int range\_min, int range\_max)

{

return rand() % (range\_max - range\_min + 1) + range\_min;

}

const unsigned int arrSIZE = 100;

const int arrLIMIT = 99;

int arr[arrSIZE];

int Search\_Binary(int arr[], int left, int right, int key)

{

int midd = 0;

while (1)

{

midd = (left + right) / 2;

if (key < arr[midd])

right = midd - 1;

else if (key > arr[midd])

left = midd + 1;

else

return midd;

if (left > right)

return -1;

}

}

void bubbleSort(int arr[], int n)

{

int i, j;

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

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

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

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

}

void printArray(int arr[], int size)

{

cout << "Несортированный массив:\n ";

int i;

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

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

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

}

void printArrayS(int arr[], int size)

{

cout << "Сортированный массив:\n";

int i;

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

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

cout << "\n\n";

}

void printMatrix(int\*\*, int, bool = 1);

void printMatrix(int\*\* matrix, int N, bool zero)

{

if (zero) system("cls");

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

cout << setw(3) << \*elemI;

if (elemI != lastI) cout << " ";

}

cout << "\n";

}

std::this\_thread::sleep\_for(0.1s);

}

void zeroMatrix(int\*\* matrix, int N) {

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI = 0;

}

}

}

void snakeMatrix(int\*\* matrix, int N) {

printMatrix(matrix, N);

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

for (int\*\* elemJ = matrix, \*\*lastJ = matrix + N - 1; elemJ <= lastJ; elemJ++)

{

\*(\*elemJ + lineNum \* 2) = rand() % 101;

printMatrix(matrix, N);

}

for (int\*\* elemJ = matrix + N - 1, \*\*lastJ = matrix; elemJ >= lastJ; elemJ--)

{

\*(\*elemJ + lineNum \* 2 + 1) = rand() % 101;

printMatrix(matrix, N);

}

}

}

void rotateMatrix(int\*\* matrix, int N) {

int\*\* a = new int\* [N / 2], \*\* b = new int\* [N / 2], \*\* c = new int\* [N / 2], \*\* d = new int\* [N / 2], \*\*\* arr = new int\*\* [4];

arr[0] = a; arr[1] = b; arr[2] = c; arr[3] = d;

for (int\*\*\* i = arr, \*\*\*ilast = arr + 3; i <= ilast; i++)

{

for (int\*\* elem = \*i, \*\*last = elem + N / 2 - 1; elem <= last; elem++)

{

\*elem = new int[N / 2];

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = a; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = b; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = c; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = d; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

int\*\* pa, \*\* pb, \*\* pc, \*\* pd;

pa = c; pb = d; pc = a; pd = b;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pa; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pb; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pc; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pd; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

printMatrix(matrix, N, 0);

}

void sortMatrix(int\*\* matrix, int N) {

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

int i = 0;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*(arr + i) = \*elemI;

i++;

}

}

bool reverse = 1;

while (reverse) {

reverse = 0;

for (int\* j = arr, \*jlast = j + (N \* N) - 2; j <= jlast; j++)

{

if (\*j > \*(j + 1))

{

reverse = 1;

int t = \*(j + 1);

\*(j + 1) = \*j;

\*j = t;

}

}

}

i = 0;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI = \*(arr + i);

i++;

}

}

printMatrix(matrix, N, 0);

}

void increaseMatrix(int\*\* matrix, int N, int k) {

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI \*= k;

}

}

printMatrix(matrix, N, 0);

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int main()

{

setlocale(LC\_ALL, "Rus");

cout << "Выберите лабораторную работу: \n" << "1. Типы данных и их внутреннее представление в паняти. \n"

<< "2. Одномерные статические массивы. \n" << "3. Указатели. \n" << "4. Текстовые строки.\n";

int a=0;

cin >> a;

switch (a)

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////

case 1:

{

setlocale(0, "");

cout << "~~~~~ 1 задание ~~~~~ \n";

cout << "В памяти компьютера: \n";

cout << "int занимает " << sizeof(int) << " байта \n";

cout << "short int занимает " << sizeof(short int) << " байта \n";

cout << "long int занимает " << sizeof(long int) << " байта \n";

cout << "float занимает " << sizeof(float) << " байта \n";

cout << "double занимает " << sizeof(double) << " байт \n";

cout << "long double занимает " << sizeof(long double) << " байт \n";

cout << "char занимает " << sizeof(char) << " байт \n";

cout << "bool занимает " << sizeof(bool) << " байт \n";

cout << "\n";

cout << "~~~~~ 2 задание ~~~~~ \n";

int imput;

cout << "Введите любое целое число \n";

cin >> imput;

int idz = imput;

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

int a = sizeof(imput) \* 8;

int mask = 1 << (a - 1);

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

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

putchar(' ');

}

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

imput <<= 1;

}

cout << "\n";

union {

float value;

int b;

};

cout << "\n";

cout << "~~~~~ 3 задание ~~~~~\n";

cout << "Введите любое значение типа float \n";

cin >> value;

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

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

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

putchar(' ');

}

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

b <<= 1;

}

cout << "\n |Экспонента|--------Мантисса-------|" << endl;

cout << "\n";

cout << "~~~~~ 4 задание ~~~~~\n";

cout << "Введите значение типа double \n";

union {

double num;

int num3[2];

};

cin >> num;

int a2 = 32;

int mask2 = 1 << (a2 - 1);

for (int i = 0; i < a2; i++) {

if (i == 1 || i % 8 == 0 || i == 12)

cout << " ";

cout << (num3[1] & mask2 ? '1' : '0');

num3[1] <<= 1;

}

for (int i = 0; i < a2; i++) {

if (i % 8 == 0)

cout << " ";

cout << (num3[0] & mask2 ? '1' : '0');

num3[0] <<= 1;

}

cout << "\n | Экспонента |-------------------------Мантисса-------------------------| \n";

}

break;

///////////////////////////////////////////////////////////////////////////////////////////////////////

case 2: {

setlocale(LC\_ALL, "rus");

int a, b, c, d, e, f, g, j, l, w, buf;

c = 0; d = 0; e = 0; f = 0;

srand(static\_cast<unsigned int>(time(NULL)));

for (unsigned int i = 0; i < arrSIZE; i++)

{

arr[i] = rrand(-arrLIMIT, arrLIMIT);

}

printArray(arr, arrLIMIT);

auto start = steady\_clock::now();

int min = arr[0];

for (int i = 0; i < 100; ++i)

{

if (arr[i] < min)

{

min = arr[i];

}

}

int max = arr[0];

for (int i = 0; i < 100; ++i)

{

if (arr[i] > max)

{

max = arr[i];

}

}

auto finish = steady\_clock::now();

auto Elapsed\_1 = duration\_cast<nanoseconds>(finish - start);

cout << "Минимальное: " << min << "\n" << "Максимальное: " << max << " \n" << "Время поиска Мин. и Макс. : " << Elapsed\_1.count() << " " << "NANOseconds\n";

a = (min + max) / 2;

cout << "Среднее значение: " << a << "\n";

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

auto Start3 = steady\_clock::now();

for (int i = 0; i < 100; ++i)

{

if (arr[i] == a)

{

c = c++;

cout << i << " ";

}

if (c == 0 && i == 99) cout << "Индексы отсутствуют";

}

auto Finish3 = steady\_clock::now();

cout << "\n";

cout << "Количество индексов элементов, равных среднему значению: " << c << "\n";

auto ELapsed5 = duration\_cast<nanoseconds>(Finish3 - Start3);

cout << "Время поиска индекса: " << ELapsed5.count() << " " << "NANOseconds\n\n";

auto begin = steady\_clock::now();

bubbleSort(arr, arrLIMIT);

auto end = steady\_clock::now();

printArrayS(arr, arrLIMIT);

auto Start = steady\_clock::now();

int min1 = arr[0];

int max1 = arr[98];

auto Finish = steady\_clock::now();

cout << "Минимальное: " << min1 << "\n" << "Максимальное: " << max1 << " \n";

auto ELapsed2 = duration\_cast<nanoseconds>(Finish - Start);

cout << "Время поиска Мин. и Макс. : " << ELapsed2.count() << " " << "NANOseconds\n";

auto elapsed\_3 = duration\_cast<microseconds>(end - begin);

cout << "Время сортировки: " << elapsed\_3.count() << " " << "MICROseconds\n";

b = (min1 + max1) / 2;

cout << "Среднее значение: " << b << "\n";

cout << "Индекс элементов, равные среднему значению: ";

auto Start2 = steady\_clock::now();

for (int i = 0; i < 100; ++i)

{

if (arr[i] == b)

{

d = d++;

cout << i << " ";

}

if (b == 0 && i == 99) cout << "Индексы отсутствуют";

}

auto Finish2 = steady\_clock::now();

cout << "\n";

cout << "Количество индексов элементов, равных среднему значению: " << d << "\n";

auto ELapsed4 = duration\_cast<nanoseconds>(Finish2 - Start2);

cout << "Время на поиск индекса: " << ELapsed4.count() << " " << "NANOseconds\n";

cout << "Введите элемент (a), который есть в массиве: ";

cin >> g;

for (int i = 0; i < 100; ++i)

{

if (arr[i] < g)

{

e = e++;

}

}

cout << "Число элементов < a: " << e << "\n";

cout << "Введите элемент (b), который есть в массиве: ";

cin >> j;

for (int i = 0; i < 100; ++i)

{

if (arr[i] > j)

{

f = f++;

}

}

cout << "Число элементов > b: " << f << "\n";

int key = 0;

int index = 0;

cout << "\nВведите любое число: ";

cin >> key;

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

index = Search\_Binary(arr, 0, arrSIZE, key);

auto finish3 = steady\_clock::now();

auto eLapsed5 = duration\_cast<nanoseconds>(finish3 - start3);

if (index >= 0)

cout << "Указанное число находится в ячейке с индексом: " << index << "\n\n";

else

cout << "В массиве нет такого числа!\n\n";

cout << "Перебором индексы массивов: ";

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

for (int i = 0; i < 100; ++i)

{

if (arr[i] == key)

{

cout << i << " ";

}

}

auto finish4 = steady\_clock::now();

auto eLapsed6 = duration\_cast<nanoseconds>(finish4 - start4);

cout << "\n";

cout << "Время затраченное бинарным поиском: " << eLapsed5.count() << " " << "NANOseconds" << " < " << "Время затраченное пиребором: " << eLapsed6.count() << " " << "NANOseconds\n";

cout << "Введите индексы массива, которые нужно поменять местами(через пробел): ";

cin >> l >> w;

swap(arr[l], arr[w]);

printArray(arr, arrLIMIT);

}

break;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

case 3: {

srand(time(NULL));

int N;

cout << "Enter N(6, 8 or 10):" << "\n";

cin >> N;

while (N != 6 && N != 8 && N != 10) {

cout << "Wrong N value, try again" << "\n";

cin >> N;

}

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

for (int\*\* elem = matrix, \*\*last = elem + N - 1; elem <= last; elem++)

{

\*elem = new int[N];

}

zeroMatrix(matrix, N);

snakeMatrix(matrix, N);

cout << "Rotated matrix: " << '\n';

rotateMatrix(matrix, N);

cout << '\n' << "Sorted matrix: " << '\n';

sortMatrix(matrix, N);

cout << '\n' << "Enter a value to increase matrix elements: ";

int k;

cin >> k;

increaseMatrix(matrix, N, k);

}

break;

case 4: {

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string line;

cout << "Введите текст с клавиатуры:" << "\n";

cin.get();

getline(cin, line);

string symbol = " ";

size\_t pos;

while ((pos = line.find(symbol)) != line.npos)

line.replace(pos, symbol.length(), " ");

string symbol2 = ",,";

size\_t pos2;

while ((pos2 = line.find(symbol2)) != line.npos)

line.replace(pos2, symbol2.length(), ",");

string symbol3 = "..";

size\_t pos3;

while ((pos3 = line.find(symbol3)) != line.npos)

line.replace(pos3, symbol3.length(), ".");

string symbol4 = "!!";

size\_t pos4;

while ((pos4 = line.find(symbol4)) != line.npos)

line.replace(pos4, symbol4.length(), "!");

string symbol5 = "??";

size\_t pos5;

while ((pos5 = line.find(symbol5)) != line.npos)

line.replace(pos5, symbol5.length(), "?");

string symbol6 = "--";

size\_t pos6;

while ((pos6 = line.find(symbol6)) != line.npos)

line.replace(pos6, symbol6.length(), "-");

string symbol7 = "\"\"";

size\_t pos7;

while ((pos7 = line.find(symbol7)) != line.npos)

line.replace(pos7, symbol7.length(), "\"");

transform(line.begin(), line.end(), line.begin(), tolower);

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

{

if (i != line.length() - 2)

{

if ((line[i] == '.' || line[i] == '!' || line[i] == '?') && line[i + 1] == ' ')

{

line[i + 2] = toupper(line[i + 2]);

}

}

}

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

cout << endl;

cout << "Изменённая строка:" << "\n";

cout << line << "\n";

cout << endl;

string linesecond = (line + " ");

int lengthWord = 0;

cout << "Строка, с напечатанными в обратном порядке словами: " << "\n";

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

{

if (int(linesecond[i]) < 0 || (47 < int(linesecond[i]) && int(linesecond[i]) < 58))

{

lengthWord += 1;

}

else

{

if (lengthWord > 0)

{

for (int j = i - 1; j > i - lengthWord - 1; j--)

{

cout << linesecond[j];

}

}

cout << linesecond[i];

lengthWord = 0;

}

}

cout << endl;

cout << endl;

cout << "Введите то, что вы хотите удалить из строки(нужен пробел перед началом):" << "\n";

string linethird = line, part;

cin.get();

getline(cin, part);

size\_t position;

while ((position = linethird.find(part)) != linethird.npos)

linethird.replace(position, part.length(), "");

cout << linethird << "\n";

string text1, part2;

cout << "Введите подстроку, которую хотите найти (нужен пробел перед началом): ";

cin.get();

getline(cin, text1);

cout << "Позиции текста, которые содержат введённую Вами подстроку (считая, что позиция первой буквы – 0): " << "\n";

if (text1.length() == 1) {

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

if (line[i] == text1[0]) {

cout << i << " ";

}

}

}

if (text1.length() >= 2) {

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

{

if (line[i] == text1[0])

{

for (int j = i; j < (i + text1.length()); j++)

{

for (int k = j; k < (j + text1.length()); k++) {

part2 += line[k - 1];

if (k == (j + text1.length()))

part2 += "\0";

}

if (part2 == text1) {

cout << i << " - " << (j + text1.length() - 2) << "\t";

part2.clear();

}

if (part2 != text1) {

part2.clear();

}

}

}

}

}

cout << endl;

}

break;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////

default:

{

cout << "Выберите от 1-4 : \n";

}

}

return 0;

}