**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: **"Единый проект по 4 лабам"**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4373 |  | Андреева П.Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение работы с двумерными массивами и сортировками, перестановками и указателями.

**Основные теоретические положения.**

### **Определение функции**

Функция определяет действия, которые выполняет программа. Функции позволяют выделить набор инструкций и придать ему имя. А затем многократно по присвоенному имени вызывать в различных частях программы. По сути функция - это именованный блок кода.

Формальное определение функции выглядит следующим образом:

тип имя\_функции(параметры)  
{  
 инструкции  
}

Первая строка представляет заголовок функции. Вначале указывается возвращаемый тип функции. Если функция не возвращает никакого значения, то используется тип void.

Затем идет имя функции, которое представляет произвольный идентификатор. К именованию функции применяются те же правила, что и к именованию переменных.

После имени функции в скобках идет перечисление параметров. Функция может не иметь параметров, в этом случае указываются пустые скобки.

После заголовка функции в фигурных скобках идет тело функции, которое содержит выполняемые инструкции.

Для возвращения результата функция применяет оператор return. Если функция имеет в качестве возвращаемого типа любой тип, кроме void, то она должна обязательно с помощью оператора return возвращать какое-либо значение.

### **Понятие массива**

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.

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

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

Аналогом одномерного массива из математики может служить последовательность некоторых элементов с одним индексом: *aiai* при i = 0, 1, 2, … n – одномерный вектор. Каждый элемент такой последовательности представляет собой некоторое значение определенного типа данных.

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.

При определении строковых переменных их можно инициализировать конкретными значениями с помощью строковых литералов:

char S1[15] = “This is text”;  
  
char S2[] = “Пример текста”;

Последние два элемента переменной просто не используются, а строка автоматически подстраивается под длину инициализирующего текста.

При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов:

cout << S1[0]; // На экране будет выведен символ ‘T’

Если строка формируется при помощи цикла (или иного способа), то необходимо в ее конец обязательно записать нулевой символ '\0'.

При выводе строк можно использовать форматирование (манипуляторы или функции потока вывода). Вывод текстовых строк на экран крайне простая задача.

Предпочтительно использование функции потока ввода cin.getline:

const int N = 21;  
  
char Str [N];  
  
cin.getline (Str, N); // Пусть введена строка “Это пример текста”  
  
cout << Str << endl; // На экран будет выведено “ Это пример текста”

Если длина введенного с клавиатуры текста превышает максимальную длину массива , в него будет записано (в нашем примере) 20 символов вводимого текста и нулевой символ. Остальные символы введенного текста остаются во входном потоке и могут быть взяты из него следующими инструкциями ввода.

Функция cin.getline может иметь третий параметр, задающий символ, при встрече которого чтение строки из потока прекращается:

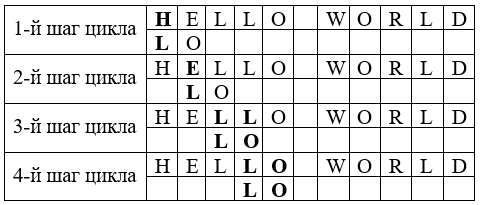
cin.getline (Str, N, ‘.’);

Иногда чтение из потока невозможно (например, попытка считать слишком длинный текст). Для того чтобы продолжить чтение из потока, необходимо восстановить его нормальное состояние. Этого можно достигнуть с помощью функции потока cin.clear(), которая сбрасывает состояние потока в нормальное. Если забирать остатки данных из потока ввода не надо, то следует очистить его с помощью функции cin.sync().

### **Прямой поиск**

При работе со строками часто будет возникать потребность в поиске набора символа или слов (поиска подстроки в строке). При условии, что текст может быть крайне большим, хочется, чтобы алгоритм поиска подстроки работал быстро.

Самый простой способ подстроки в строке – Линейный поиск – циклическое сравнение всех символов строки с подстрокой. Действительно, этот способ первый приходит в голову, но очевидно, что он будет самым долгим.



На первых двух итерациях цикла сравниваемые буквы не будут совпадать. На третьей же итерации, совпал символ ‘L’, это означает, что теперь нужно сравнивать следующий символ подстроки со следующим символом строки. Видно, что символы отличаются, поэтому алгоритм продолжает свою работу. На четвертой же итерации подстрока была найдена.

Если представить, что исходная строка непорядок больше и подстрока находится в конце строки (или вовсе отсутствует), то сразу видны минусы данного алгоритма.

Одной из вариаций алгоритма полного перебора является реализация таблицы включений. Суть алгоритма: дана подстрока S и строка T. Требуется определить индекс, начиная с которого образец S содержится в строке T. Если S не содержится в T, необходимо вернуть индекс, который не может быть интерпретирован как позиция в строке.

### **Бойера-Мура**

Алгоритм Бойера-Мура считается наиболее быстрым среди алгоритмов общего назначения, предназначенных для поиска подстроки в строке.

Преимущество этого алгоритма в том, что ценной некоторого количества предварительных вычислений над подстрокой (но не над исходной строкой, в которой ведётся поиск), подстрока сравнивается с исходным текстом не во всех позициях (пропускаются позиции, которые точно не дадут положительный результат).

Поиск подстроки ускоряется благодаря созданию таблиц сдвигов. Сравнение подстроки со строки начинается с последнего символа подстроки, а затем происходит прыжок, длина которого определяется по таблице сдвигов. Таблица сдвигов строится по подстроке так чтобы перепрыгнуть максимальное количество символов строки и не пропустить вхождение подстроки в строку.

Правила построения таблицы сдвигов:

1) Значение элемента таблицы равно удаленности соответствующего символа от конца шаблона (подстроки).

2) Если символ встречается более одного раза, то применятся значение, соответствующее символу, наиболее близкому к концу шаблона.

3) Если символ в конце шаблона встречается 1 раз, ему соответствует значение, равное длине образа; если более одного раза – значение, соответствующее символу, наиболее близкому к концу образа.

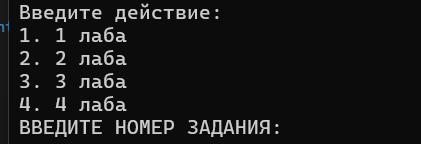
4) Для символов, отсутствующих в образе, применяется значение, равное длине шаблона.

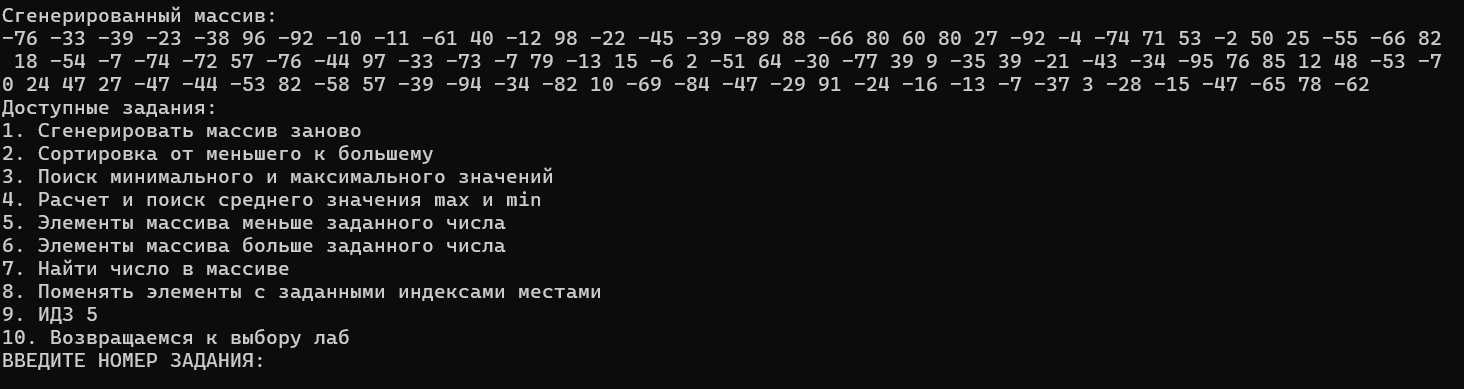
**Постановка задачи.**

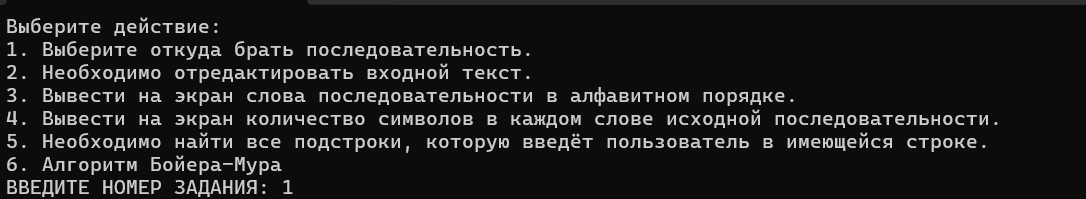
Необходимо объединить все 4 лабораторные работы в единый проект. Нужно добавить инфраструктуру переключения между заданиями (интерактивное меню).   
  
Добавьте сюда свой код, ссылку на git с проектом и отчётом (в отчете должны присутствовать скриншоты работы программы). Если в отчёте нет примеров работы программы, добавьте их здесь.

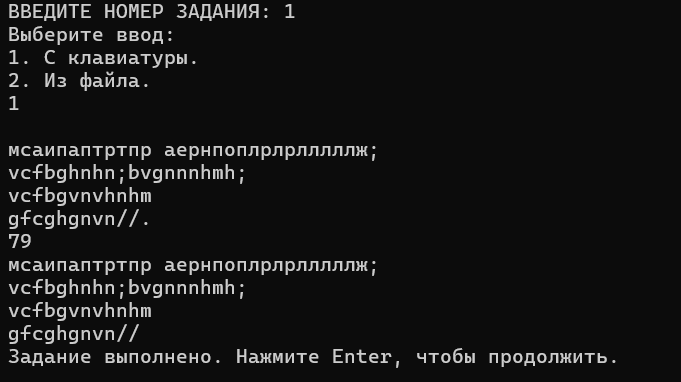
**Выполнение работы.**

Код программы представлен в приложении А









**Выводы.**

Я узнала, как работать с файлами, массивами, текстом, и многим другим. Изучила тему указателей и другие темы.

Приложение А

рабочий код

#include <iostream>

#include <windows.h>

#include <fstream>

#include <thread>

#include <chrono>

#include <ctime>

#include <bitset>

using namespace std;

using namespace chrono;

// лаба1

void replacing\_bit\_divisibility\_by\_2(int& number) {

cout << "ИДЗ(замена битов исходя из их четности):";

cout << endl;

unsigned int mask = 1 << (sizeof(int) \* 8) - 1;

if (number >= 0) {

for (int i = 0; i <= sizeof(int) \* 8 - 1; ++i) {

if (i % 2 != 0) {

number |= (1 << i);

};

mask >>= 1;

};

}

else {

for (int i = 0; i <= sizeof(int) \* 8 - 1; ++i) {

putchar((i % 2 == 0) ? '0' : ((number & mask) ? '1' : '0'));

if (i % 2 == 0) {

number &= (~(1 << i));

};

mask >>= 1;

};

};

cout << endl;

cout << number << endl;

}

void display\_menu\_for\_laba1() {

cout << "Выберите действие: " << endl;

cout << "1) Сколько памяти (в байтах)отводится под различные типы данных со спецификаторами и без:int, short int, long int, float, double, long double, char и bool." << endl;

cout << "2) Вывести на экран двоичное представление в памяти (все разряды) целого числа." << endl;

cout << "3) Вывести на экран двоичное представление в памяти (все разряды) типа float." << endl;

cout << "4) Вывести на экран двоичное представление в памяти (все разряды) типа double." << endl;

cout << "5) Возвращаемся к лабам." << endl;

}

int main\_of\_laba1() {

setlocale(0, "");

int i, part, j;

int int\_num;

unsigned int mask = 1 << (sizeof(int) \* 8) - 1;

union {

float float\_num;

int float\_to\_int;

};

unsigned int mask\_float = 1 << (sizeof(int) \* 8) - 1;

union {

double double\_num;

unsigned int double\_divided\_on\_two\_parts[2];

};

unsigned int mask\_double = 1 << ((sizeof(int) \* 8) - 1);

while (true) {

display\_menu\_for\_laba1();

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

cin >> i;

switch (i) {

case 1:

//1) Сколько памяти (в байтах)отводится под различные типы данных со спецификаторами и без:

// int, short int, long int, float, double, long double, char и bool.

cout << "Тип данных int - " << sizeof(int) << "байта" << endl;

cout << "Тип данных short int - " << sizeof(short int) << "байта" << endl;

cout << "Тип данных long int - " << sizeof(long int) << "байта" << endl;

cout << "Тип данных float - " << sizeof(float) << "байта" << endl;

cout << "Тип данных double - " << sizeof(double) << "байта" << endl;

cout << "Тип данных long double - " << sizeof(long double) << "байта" << endl;

cout << "Тип данных char - " << sizeof(char) << "байта" << endl;

cout << "Тип данных bool - " << sizeof(bool) << "байта" << endl;

break;

case 2:

//2) Вывести на экран двоичное представление в памяти (все разряды) целого числа.

cout << "Введите число типа int: ";

cin >> int\_num;

cout << "двоичное представление в памяти(все разряды) целого числа: " << endl;

for (i = 0; i <= sizeof(int) \* 8 - 1; ++i) {

putchar(int\_num & mask ? '1' : '0');

mask >>= 1;

if (i == 0 || i % 8 == 7) {

putchar(' ');

}

};

cout << endl;

replacing\_bit\_divisibility\_by\_2(int\_num);//ИДЗ

cout << endl;

cout << int\_num;

cout << endl;

mask = 1 << (sizeof(int) \* 8) - 1;

for (int i = 0; i <= sizeof(int) \* 8 - 1; ++i) {

putchar(int\_num & mask ? '1' : '0');

mask >>= 1;

if (i == 0 || i % 8 == 7) {

putchar(' ');

}

};

cout << endl;

break;

case 3:

//3) Вывести на экран двоичное представление в памяти (все разряды) типа float.

cout << "Введите число типа float: ";

cin >> float\_num;

cout << "двоичное представление в памяти (все разряды) типа float" << endl;

for (i = 0; i <= (sizeof(int) \* 8) - 1; ++i) {

putchar(float\_to\_int & mask\_float ? '1' : '0');

mask\_float >>= 1;

if (i == 0 || i == 8 || i == 12) { // Знак (1 бит) и Экспонента (8 бит)

putchar(' ');

}

}

cout << endl;

break;

case 4:

//4) Вывести на экран двоичное представление в памяти (все разряды) типа double.

cout << "Введите число типа double: ";

cin >> double\_num;

cout << "двоичное представление в памяти (все разряды) типа double: ";

cout << endl;

for (part = 1; part >= 0; --part) {

for (j = 0; j <= (sizeof(int) \* 8 - 1); ++j) {

putchar(double\_divided\_on\_two\_parts[part] & mask\_double ? '1' : '0');

mask\_double >>= 1;

};

mask\_double = 1 << ((sizeof(int) \* 8) - 1);

};

break;

case 5:

cout << "Возвращаемся к выбору лабы!!!" << endl;

return 0;

default:

cout << "Некорректный номер задания" << endl;

break;

}

cout << "Задание выполнено. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}

//лаба2

void idz\_for\_laba2(int arr[], int N) {

int value\_to\_decrease;

int counter\_even = 0;

int counter\_odd = 0;

int start\_val = 1;

int end\_val = 9;

srand(time(0));

cout << "Введите значение для уменьшения четных элементов ";

cin >> value\_to\_decrease;

cout << "Выводим четные элементы, уменьшенные в процессе ";

int random\_value = (rand() % (end\_val - start\_val + 1) + start\_val);

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

if (i % 2 == 0) {

// уменьшение на заданное значение

arr[i] = arr[i] - value\_to\_decrease;

cout << arr[i] << ' ';

// умножение четного на рандомное значение

arr[i] = arr[i] \* random\_value;

if (arr[i] % 2 == 0) {

counter\_even += 1;

}

}

else if (i % 2 != 0 and arr[i] % 2 != 0) {

counter\_odd += 1;

};

};

cout << endl;

cout << "Выводим массив с четными элементами, умноженными на значение " << random\_value << " : ";

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

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

};

cout << endl;

cout << "Количество четных элементов имеющих четные значения: " << counter\_even << endl;

cout << "Количество нечетных элементов имеющих нечетные значения: " << counter\_odd << endl;

for (int i = 1; i < 10; i++) {

int counter\_del = 0;

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

if (arr[j] % i == 0) {

counter\_del += 1;

}

};

cout << "Количество чисел делящихся на " << i << " равно " << counter\_del << endl;

};

};

void shaker\_sort(int arr[], int n) {

bool swapped = true;

int start = 0;

int end = n - 1;

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

nanoseconds result;

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

for (int a = 0; a < n / 2; a++) {

swapped = false;

// Проход слева направо

for (int i = start; i < end; ++i) {

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

std::swap(arr[i], arr[i + 1]);

swapped = true;

}

}

if (!swapped) {

break;

}

// Проход справа налево

for (int i = end - 1; i >= start; --i) {

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

std::swap(arr[i], arr[i + 1]);

swapped = true;

}

}

}

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

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

cout << result.count() << " наносекунд - время сортировки с помощью shaker sort" << endl;

}

void bubble\_sort(int arr[], int N) {

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

nanoseconds result;

start = steady\_clock::now();

for (int j = 0; j < N - 1; j++) {

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

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

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

};

};

};

end = steady\_clock::now();

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

cout << result.count() << " наносекунд - время сортировки с помощью bubble sort" << endl;

}

void insert\_sort(int arr[], int n) {

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

nanoseconds result;

start = steady\_clock::now();

int i, key, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

// Сдвигаем элементы массива, которые больше key, на одну позицию вперед

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

};

end = steady\_clock::now();

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

cout << result.count() << " наносекунд - время сортировки с помощью insert sort" << endl;

}

void quick\_sort(int arr[], int end, int begin)

{

int mid;

int f = begin;

int l = end;

mid = arr[(f + l) / 2];

while (f < l)

{

while (arr[f] < mid) f++;

while (arr[l] > mid) l--;

if (f <= l)

{

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

f++;

l--;

}

}

if (begin < l) quick\_sort(arr, l, begin);

if (f < end) quick\_sort(arr, end, f);

}

void comb\_sort(int arr[], int N) {

int swap;

float k = 1.247, S = N - 1;

int count = 0;

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

nanoseconds result;

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

while (S >= 1)

{

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

{

if (arr[i] > arr[int(i + S)])

{

swap = arr[int(i + S)];

arr[int(i + S)] = arr[i];

arr[i] = swap;

}

}

S /= k;

}

while (true)

{

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

{

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

{

swap = arr[i + 1];

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

arr[i] = swap;

}

else count++;

}

if (count == N - 1)

break;

else

count = 0;

}

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

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

cout << result.count() << " наносекунд - время сортировки с помощью comb sort" << endl;

}

void from\_small\_to\_big\_2(int arr[], int N) {

shaker\_sort(arr, N);

insert\_sort(arr, N);

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

nanoseconds result;

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

quick\_sort(arr, N - 1, 0);

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

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

cout << result.count() << " наносекунд - время сортировки с помощью quick sort" << endl;

comb\_sort(arr, N);

bubble\_sort(arr, N);

cout << "Отсортированный массив: ";

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

cout << arr[a] << ' ';

};

cout << endl;

};

void max\_min\_elems\_of\_arr\_3(int arr[], int not\_sort[], int N) {

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

nanoseconds result;

start = steady\_clock::now();

// несортированный массив

int min = not\_sort[0];

int max = not\_sort[0];

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

if (not\_sort[i] < min) {

min = not\_sort[i];

}

if (not\_sort[i] > max) {

max = not\_sort[i];

}

};

end = steady\_clock::now();

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

cout << result.count() << " наносекунд - ВРЕМЯ ВЫПОЛНЕНИЯ С НЕСОРТИРОВАННЫМ МАССИВОМ" << endl;

cout << "Максимальный и минимальный элементы для неотсортированного массива: " << min << ' ' << max << endl;

// для отсортированного массива

start = steady\_clock::now();

min = arr[0];

max = arr[N - 1];

end = steady\_clock::now();

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

cout << result.count() << " наносекунд - ВРЕМЯ ВЫПОЛНЕНИЯ С СОРТИРОВАННЫМ МАССИВОМ" << endl;

cout << "Максимальный и минимальный элементы для отсортированного массива: " << min << ' ' << max << endl;

};

void median\_of\_arr\_4(int arr[], int N) {

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

nanoseconds result;

int srednee\_znachenie = (arr[0] + arr[N - 1]) / 2;

cout << "среднее от min и max " << srednee\_znachenie << endl;

int counter = 0;

cout << "Под каким номером есть такое же число как среднее: ";

start = steady\_clock::now();

for (int i = 0; i < N && arr[i] <= srednee\_znachenie; i++) {

if (arr[i] == srednee\_znachenie) {

cout << i << endl;

counter += 1;

}

};

end = steady\_clock::now();

if (counter == 0) {

cout << "нет таких элементов массива" << endl;

}

else {

cout << "Всего найдено чисел: " << counter << endl;

};

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

cout << "Нахождение заданных элементов заняло: " << result.count() << " наносекунд" << endl;

}

void smaller\_num\_counter\_5(int arr[], int N) {

int num;

int count = 0;

cout << "ВВЕДИТЕ ЧИСЛО: ";

cin >> num;

for (int i = 0; i < N && arr[i] < num; i++, count++);

cout << endl;

cout << "КОЛИЧЕСТВО ЧИСЕЛ МЕНЬШЕ ЧЕМ " << num << " РАВНО " << count << endl;

};

void bigger\_num\_counter\_6(int arr[], int N) {

int num;

int count = 0;

cout << "ВВЕДИТЕ ЧИСЛО: ";

cin >> num;

for (int i = N - 1; i >= 0 && arr[i] > num; i--, count++) {};

cout << endl;

cout << "КОЛИЧЕСТВО ЧИСЕЛ БОЛЬШЕ ЧЕМ " << num << " РАВНО " << count << endl;

};

int binarySearch(int arr[], int value, int start, int end) {

if (end >= start) {

int mid = start + (end - start) / 2;

if (arr[mid] == value) {

return mid;

}

if (arr[mid] > value) {

return binarySearch(arr, value, start, mid - 1);

}

return binarySearch(arr, value, mid + 1, end);

}

return -1;

};

void find\_number\_in\_arr\_7(int arr[], int N) {

int num;

int num\_in\_arr = 0;

cout << "ВВЕДИТЕ ЧИСЛО ДЛЯ ЗАДАНИЯ 7: ";

cin >> num;

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

nanoseconds result;

cout << "Полным перебором: ";

// полный перебор

start = steady\_clock::now();

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

if (num == arr[a]) {

num\_in\_arr = 1;

break;

}

};

end = steady\_clock::now();

if (num\_in\_arr == 1) {

cout << "Введенное число есть в массиве!" << endl;

}

else {

cout << "Введенного числа нет в массиве!" << endl;

}

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

cout << result.count() << " наносекунд - ВРЕМЯ ВЫПОЛНЕНИЯ ОБЫЧНЫМ ПЕРЕБОРОМ" << endl;

cout << "Бинарным поиском: ";

start = steady\_clock::now();

// бинарный поиск

int answer = binarySearch(arr, num, 0, N - 1);

end = steady\_clock::now();

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

if (answer == -1) {

cout << "Введенного числа нет в массиве!" << endl;

}

else {

cout << "Введенное число есть в массиве!" << endl;

}

cout << result.count() << " наносекунд - ВРЕМЯ ВЫПОЛНЕНИЯ БИНАРНЫМ ПОИСКОМ" << endl;

};

void change\_elems\_of\_arr\_8(int arr[], int N) {

int elem\_index\_1;

int elem\_index\_2;

cout << "Введи индексы, которые хочешь поменять в массиве:";

cin >> elem\_index\_1;

cout << endl;

cin >> elem\_index\_2;

swap(arr[elem\_index\_1], arr[elem\_index\_2]);

cout << "МАССИВ С ИЗМЕННЕННЫМИ ЗНАЧЕНИЯМИ " << endl;

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

cout << arr[i] << ' ';

};

cout << endl;

};

int main\_of\_laba2()

{

setlocale(0, "");

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

const int N = 100;

unsigned identificator;

int Array[N];

int not\_sorted\_array[N];

unsigned int start\_val = -99;

int end\_val = 99;

srand(time(0));

cout << "Сгенерированный массив: " << endl;

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

Array[i] = rand() % (end\_val - start\_val + 1) + start\_val;

not\_sorted\_array[i] = Array[i];

cout << Array[i] << ' ';

};

cout << endl;

cout << "Доступные задания: " << endl;

cout << "1. Сгенерировать массив заново" << endl;

cout << "2. Сортировка от меньшего к большему" << endl;

cout << "3. Поиск минимального и максимального значений" << endl;

cout << "4. Расчет и поиск среднего значения max и min" << endl;

cout << "5. Элементы массива меньше заданного числа" << endl;

cout << "6. Элементы массива больше заданного числа" << endl;

cout << "7. Найти число в массиве" << endl;

cout << "8. Поменять элементы с заданными индексами местами" << endl;

cout << "9. ИДЗ 5" << endl;

cout << "10. Возвращаемся к выбору лаб" << endl;

while (true) {

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

if (!(cin >> identificator)) {

cin.clear();

continue;

}

switch (identificator) {

case 1:

cout << "Новый массив" << endl;

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

Array[i] = rand() % (end\_val - start\_val + 1) + start\_val;

not\_sorted\_array[i] = Array[i];

cout << Array[i] << ' ';

}

cout << endl;

break;

case 2:

cout << "Сортировка от меньшего к большему" << endl;

from\_small\_to\_big\_2(Array, N);

break;

case 3:

max\_min\_elems\_of\_arr\_3(Array, not\_sorted\_array, N);

break;

case 4:

median\_of\_arr\_4(Array, N);

break;

case 5:

smaller\_num\_counter\_5(Array, N);

break;

case 6:

bigger\_num\_counter\_6(Array, N);

break;

case 7:

find\_number\_in\_arr\_7(Array, N);

break;

case 8:

change\_elems\_of\_arr\_8(Array, N);

break;

case 9:

idz\_for\_laba2(Array, N);

break;

case 10:

cout << "Возвращаемся к выбору лабы!!" << endl;

return 0;

default:

cout << "Некорректный номер задания" << endl;

break;

}

}

}

// лаба 3

void print\_matrix(int\* matrix, int N) {

int\* end = matrix + (N \* N) - 1;

for (int\* i = matrix; i <= end; ++i) {

if ((i - matrix) % N == 0) {

cout << "\n";

}

cout << \*i << " ";

}

}

void print\_Matrix(int\* matrix, int n, int helper) {

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

COORD destCoord;

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

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

destCoord.X = j \* 4 + 50;

destCoord.Y = i + helper;

SetConsoleCursorPosition(hStdout, destCoord);

cout << \*(matrix + i \* n + j) << " ";

}

cout << endl;

}

}

void idz\_for\_3() {

int N;

cout << "Введите число для ИДЗ(от 2 до 10)" << endl;

cin >> N;

int\* matrix1 = new int[N \* N];

int\* matrix2 = new int[N \* N];

int\* matrix3 = new int[N \* N];

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

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

\*(matrix1 + i \* N + j) = rand() % (N \* N) + 1;

\*(matrix2 + i \* N + j) = rand() % (N \* N) + 1;

\*(matrix3 + i \* N + j) = 1;

}

}

cout << "Первая матрица" << endl;

print\_Matrix(matrix1, N, 0);

cout << "Вторая матрица" << endl;

print\_Matrix(matrix2, N, N + 1);

cout << endl;

int top = 0, bottom = N - 1, left = 0, right = N - 1;

while (top <= bottom && left <= right) {

for (int i = left; i <= right; i++) {

int\* num = matrix3 + i + top \* N;

\*num = \*(matrix1 + i + top \* N) + \*(matrix2 + i + top \* N);

print\_Matrix(matrix3, N, N + 12);

Sleep(50);

}

top++;

}

print\_Matrix(matrix3, N, N + 12);

}

void shaker\_sort\_for\_3(int\* matrix, int N) {

bool swapped = true;

while (swapped) {

swapped = false;

for (int\* i = matrix + 1, \*end = matrix + N \* N - 1; i <= end; i++) {

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

int temp = \*(i - 1);

\*(i - 1) = \*i;

\*i = temp;

swapped = true;

}

}

if (!swapped) {

break;

}

swapped = false;

for (int\* i = matrix + N \* N - 2; i >= matrix; i--) {

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

int temp = \*(i - 1);

\*(i - 1) = \*i;

\*i = temp;

swapped = true;

}

}

}

}

void comb\_sort\_for\_3(int\* matrix, int N) {

float k = 1.247;

int S = N \* N;

bool swapped = true;

while (S > 1 || swapped) {

if (S > 1) {

S /= k;

}

swapped = false;

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

if (\*(matrix + i) > \*(matrix + i + S)) {

swap(\*(matrix + i), \*(matrix + i + S));

swapped = true;

}

}

}

}

void quick\_sort\_for\_3(int\* matrix, int\* begin, int\* end) {

int\* f = begin;

int\* l = end;

int mid = \*(begin + (end - begin) / 2);

while (f <= l) {

while (\*f < mid) f++;

while (\*l > mid) l--;

if (f <= l) {

swap(\*f, \*l);

f++;

l--;

}

}

if (begin < l) quick\_sort\_for\_3(matrix, begin, l);

if (f < end) quick\_sort\_for\_3(matrix, f, end);

}

void insert\_sort\_for\_3(int\* matrix, int N) {

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

int key = \*(matrix + i);

int j = i - 1;

while (j >= 0 && \*(matrix + j) > key) {

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

j--;

}

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

}

}

void fill\_snake(int\* matrix, int n) {

int top = 0, bottom = n - 1, left = 0, right = n - 1;

while (top <= bottom && left <= right) {

for (int i = top; i <= bottom; i++) {

int\* num = matrix + i \* n + left;

\*num = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(400);

}

left++;

for (int i = bottom; i >= top; i--) {

int\* num = matrix + i \* n + left;

\*num = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(400);

}

left++;

}

}

void fill\_spiral(int\* matrix, int n) {

int top = 0, bottom = n - 1, left = 0, right = n - 1;

while (top <= bottom && left <= right) {

for (int i = left; i <= right; i++) {

\*(matrix + top \* n + i) = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(100);

}

top++;

for (int i = top; i <= bottom; i++) {

\*(matrix + i \* n + right) = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(100);

}

right--;

for (int i = right; i >= left; i--) {

\*(matrix + bottom \* n + i) = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(100);

}

bottom--;

for (int i = bottom; i >= top; i--) {

\*(matrix + i \* n + left) = rand() % (n \* n) + 1;

print\_Matrix(matrix, n, 0);

Sleep(100);

}

left++;

}

print\_Matrix(matrix, n, 0);

}

void swap\_quarters\_a(int\* matrix, int N) {

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

int middle = N / 2;

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

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

\*(temp + i \* N + j) = \*(matrix + i \* N + j);

}

}

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

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

if (i < middle && j < middle) {

\*(matrix + i \* N + j + middle) = \*(temp + i \* N + j);

}

else if (i < middle && j >= middle) {

\*(matrix + (i + middle) \* N + j) = \*(temp + i \* N + j);

}

else if (i >= middle && j >= middle) {

\*(matrix + i \* N + (j - middle)) = \*(temp + i \* N + j);

}

else {

\*(matrix + (i - middle) \* N + j) = \*(temp + i \* N + j);

}

}

}

print\_Matrix(matrix, N, N + 10);

}

void swap\_quarters\_b(int\* matrix, int N) {

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

int middle = N / 2;

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

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

\*(temp + i \* N + j) = \*(matrix + i \* N + j);

}

}

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

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

if (i < middle && j < middle) {

\*(matrix + (i + middle) \* N + j + middle) = \*(temp + i \* N + j);

}

else if (i < middle && j >= middle) {

\*(matrix + (i + middle) \* N + j - middle) = \*(temp + i \* N + j);

}

else if (i >= middle && j >= middle) {

\*(matrix + (i - middle) \* N + j - middle) = \*(temp + i \* N + j);

}

else {

\*(matrix + (i - middle) \* N + j + middle) = \*(temp + i \* N + j);

}

}

}

print\_Matrix(matrix, N, N + 10);

}

void swap\_quarters\_c(int\* matrix, int N) {

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

int middle = N / 2;

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

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

\*(temp + i \* N + j) = \*(matrix + i \* N + j);

}

}

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

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

if (i < middle) {

\*(matrix + (i + middle) \* N + j) = \*(temp + i \* N + j);

}

else {

\*(matrix + (i - middle) \* N + j) = \*(temp + i \* N + j);

}

}

}

print\_Matrix(matrix, N, N + 10);

}

void swap\_quarters\_d(int\* matrix, int N) {

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

int middle = N / 2;

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

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

\*(temp + i \* N + j) = \*(matrix + i \* N + j);

}

}

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

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

if (j < middle) {

\*(matrix + i \* N + j + middle) = \*(temp + i \* N + j);

}

else {

\*(matrix + i \* N + j - middle) = \*(temp + i \* N + j);

}

}

}

print\_Matrix(matrix, N, N + 10);

}

void enlarge\_each\_element(int\* matrix, int N) {

int enlarge\_num;

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

cin >> enlarge\_num;

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

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

\*(matrix + i \* N + j) += enlarge\_num;

}

}

print\_Matrix(matrix, N, N + 10);

}

void reduce\_each\_element(int\* matrix, int N) {

int reduce\_num;

cout << "Введите число, которое нужно отнять: ";

cin >> reduce\_num;

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

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

\*(matrix + i \* N + j) -= reduce\_num;

}

}

print\_Matrix(matrix, N, N + 10);

}

void divide\_each\_element(int\* matrix, int N) {

int divide\_num;

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

cin >> divide\_num;

if (divide\_num == 0) {

cout << "Деление на ноль!" << endl;

return;

}

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

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

\*(matrix + i \* N + j) /= divide\_num;

}

}

print\_Matrix(matrix, N, N + 10);

}

void multiply\_each\_element(int\* matrix, int N) {

int multiply\_num;

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

cin >> multiply\_num;

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

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

\*(matrix + i \* N + j) \*= multiply\_num;

}

}

print\_Matrix(matrix, N, N + 10);

}

void display\_menu\_for\_laba3() {

cout << "\nДоступные задания:" << endl;

cout << "1. Задание 1. Заполняет квадратичную целочисленную матрицу порядка N (6,8,10) случайными числами от 1 до N\*N согласно схемам" << endl;

cout << "2. Задание 2. Получение новой матрицы, из матрицы п. 1, переставляя ее блоки в соответствии со схемами" << endl;

cout << "3. Задание 3. Используя арифметику указателей, сортирует элементы любой сортировкой из списка ниже" << endl;

cout << "4. Задание 4. Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число" << endl;

cout << "5. Задание 5. ИДЗ" << endl;

cout << "6. Задание 6. Вернемся к выбору лабы." << endl;

}

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

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

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

\*(matrix + i \* N + j) = 1;

}

}

}

int main\_of\_laba3() {

setlocale(0, "");

srand(time(NULL));

int N, identificator, num1, num2, num3, num4;

cout << "Введите размер матрицы (6, 8, 10): ";

cin >> N;

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

while (true) {

display\_menu\_for\_laba3();

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

cin >> identificator;

system("cls"); // Для Windows

switch (identificator) {

case 1:

start(matrix, N);

cout << "1. Массив спиралью" << endl;

cout << "2. Массив змейкой" << endl;

cin >> num1;

switch (num1) {

case 1:

fill\_spiral(matrix, N);

break;

case 2:

fill\_snake(matrix, N);

break;

}

break;

case 2:

print\_Matrix(matrix, N, 0);

cout << "Меняем четверти" << endl;

cout << "1. по часовой стрелке" << endl;

cout << "2. по диагонали " << endl;

cout << "3. сверху вниз" << endl;

cout << "4. справа налево" << endl;

cin >> num2;

cout << "Измененная матрица" << endl;

switch (num2) {

case 1:

swap\_quarters\_a(matrix, N);

break;

case 2:

swap\_quarters\_b(matrix, N);

break;

case 3:

swap\_quarters\_c(matrix, N);

break;

case 4:

swap\_quarters\_d(matrix, N);

break;

}

break;

case 3:

print\_Matrix(matrix, N, 0);

cout << "Введите сортировку:" << endl;

cout << "1. Shaker sort" << endl;

cout << "2. Comb sort" << endl;

cout << "3. Insert sort" << endl;

cout << "4. Quick sort" << endl;

cin >> num3;

switch (num3) {

case 1:

shaker\_sort\_for\_3(matrix, N);

print\_Matrix(matrix, N, N + 10);

break;

case 2:

comb\_sort\_for\_3(matrix, N);

print\_Matrix(matrix, N, N + 10);

break;

case 3:

insert\_sort\_for\_3(matrix, N);

print\_Matrix(matrix, N, N + 10);

break;

case 4:

quick\_sort\_for\_3(matrix, matrix, matrix + N \* N - 1);

print\_Matrix(matrix, N, N + 10);

break;

}

cout << "Измененная матрица" << endl;

break;

case 4:

print\_Matrix(matrix, N, 0);

cout << "Введите номер операции с элементами массива:" << endl;

cout << "1. Увеличить элементы массива" << endl;

cout << "2. Уменьшить элементы массива " << endl;

cout << "3. Умножить элементы массива" << endl;

cout << "4. Разделить элементы массива" << endl;

cout << "Измененная матрица" << endl;

cin >> num4;

switch (num4) {

case 1:

enlarge\_each\_element(matrix, N);

break;

case 2:

reduce\_each\_element(matrix, N);

break;

case 3:

multiply\_each\_element(matrix, N);

break;

case 4:

divide\_each\_element(matrix, N);

break;

}

break;

case 5:

cout << "ИДЗ" << endl;

idz\_for\_3();

break;

case 6:

cout << "Обратно к выбору лабы!!" << endl;

return 0;

default:

cout << "Некорректный номер задания." << endl;

break;

}

cout << "Задание выполнено. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}

// 4 лаба

unsigned my\_str\_len(char\* S)

{

unsigned L = 0;

while (S[L])

++L;

return L;

}

void keyboard\_input(char\* text, int N) {

cin.getline(text, N, '.');

int len\_of\_sequence = my\_str\_len(text);

for (int j = 0; j < len\_of\_sequence; j++) {

text[j] = text[j + 1];

}

--len\_of\_sequence;

cin.sync();

}

void input\_from\_file(char\* text) {

ifstream File;

File.open("text.txt");

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

cout << "" << endl;

}

File.read((char\*)text, sizeof(text));

File.close();

}

void print\_result(char\* text, int N) {

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

cout << text[i];

cout << endl;

}

void delete\_space(char\* text, int N) {

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

if (text[i] == ' ' and text[i + 1] == ' ') {

while (text[i + 1] == ' ') {

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

text[j] = text[j + 1];

}

--N;

}

}

}

}

void delete\_punctuation(char\* text, int N) {

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

if (ispunct(text[i]) and ispunct(text[i + 1])) {

if (text[i] == '.' and text[i + 1] == '.' and text[i + 2] == '.' and not ispunct(text[i + 3])) {

i += 3;

}

else {

while (ispunct(text[i + 1])) {

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

text[j] = text[j + 1];

}

--N;

}

}

}

}

}

void make\_lower(char\* text, int N) {

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

if (i == 0) {

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

}

else if (text[i - 1] == ' ' and isalpha(text[i])) {

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

}

else {

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

}

}

}

void correctLetterCase(char\* text) {

for (int i = 0; i < my\_str\_len(text); i++) {

if (isalpha(text[i]) && (i == 0 || text[i - 1] == ' ')) {

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

}

else {

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

}

}

}

void how\_many\_symbols\_in\_word(char\* text, int N) {

int counter = 0;

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

if (text[i] != ' ' && text[i] != '\0') {

cout << text[i];

counter += 1;

}

else {

cout << " - " << counter << endl;

counter = 0;

}

}

}

int linear\_search(char\* text, char pattern[], int N) {

int pattern\_len = my\_str\_len(pattern);

for (int i = 0; i <= N - pattern\_len; ++i) {

int j;

for (j = 0; j <= pattern\_len; ++j) {

if (text[i + j] != pattern[j]) {

break;

}

}

if (j == pattern\_len) {

return i;

}

}

return -1;

}

void prepare\_table(const char\* pattern, int pattern\_len, int table[256]) {

for (int i = 0; i < 256; i++) {

table[i] = pattern\_len;

}

for (int i = 0; i < pattern\_len - 1; i++) {

table[(unsigned char)pattern[i]] = pattern\_len - i - 1;

}

}

int Boyer\_Moore\_algorithm(char\* text, char pattern[], int N, int\* table) {

int pattern\_len = my\_str\_len(pattern);

int text\_len = my\_str\_len(text);

prepare\_table(pattern, pattern\_len, table);

int j, k;

int i = pattern\_len - 1;

while (i < text\_len) {

int j = pattern\_len - 1;

int k = i;

while (j >= 0 && k >= 0 && text[k] == pattern[j]) {

k--;

j--;

}

if (j < 0) {

return k + 1;

}

i += table[(unsigned char)text[i]];

}

return -1;

}

int organize\_words(char\* text, char words[][10]) {

int i = 0, word\_len = 0, count = 0;

while (text[i] != '\0' && count < 50) {

if (text[i] == ' ' || text[i + 1] == '\0') {

if (word\_len > 0) {

words[count][word\_len] = '\0';

count++;

word\_len = 0;

}

}

else {

words[count][word\_len] = text[i];

word\_len++;

}

i++;

}

return count;

}

void swap\_words(char words[][10], int j) {

char temp[10];

int k = 0;

while (words[j][k] != '\0' && k < 10) {

temp[k] = words[j][k];

k++;

}

temp[k] = '\0';

k = 0;

while (words[j - 1][k] != '\0' && k < 10) {

words[j][k] = words[j - 1][k];

k++;

}

words[j][k] = '\0';

k = 0;

while (temp[k] != '\0' && k < 10) {

words[j - 1][k] = temp[k];

k++;

}

words[j - 1][k] = '\0';

}

void sort\_alphabetically(char words[][10], int count) {

int c = 0;

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

for (int j = count - 1; j > i; j--) {

if (words[j][0] < words[j - 1][0]) {

swap\_words(words, j);

}

else if (words[j][0] == words[j - 1][0]) {

while (words[j][c] == words[j - 1][c]) {

c++;

}

if (words[j][c] < words[j - 1][c]) {

swap\_words(words, j);

}

}

}

}

}

void display\_menu\_of\_laba4() {

cout << "Выберите действие:" << endl;

cout << "1. Выберите откуда брать последовательность." << endl;

cout << "2. Необходимо отредактировать входной текст." << endl;

cout << "3. Вывести на экран слова последовательности в алфавитном порядке." << endl;

cout << "4. Вывести на экран количество символов в каждом слове исходной последовательности." << endl;

cout << "5. Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке." << endl;

cout << "6. Алгоритм Бойера-Мура" << endl;

cout << "7. Возвращаемся в основное меню." << endl;

}

int main\_of\_laba4()

{

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int identificator, num1, num2, n, len\_of\_sequence, result, count, index;

const int N = 600;

char sequence[N] = { ' ' }, pattern[30], pattern1[30], words[50][10];

int table[256];

while (true) {

display\_menu\_of\_laba4();

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

cin >> identificator;

switch (identificator) {

case 1:

cout << "Выберите ввод:" << endl;

cout << "1. С клавиатуры." << endl;

cout << "2. Из файла." << endl;

cin >> num1;

cout << endl;

switch (num1) {

case 1:

keyboard\_input(sequence, N);

sequence[my\_str\_len(sequence)] = ' ';

sequence[my\_str\_len(sequence) + 1] = '\0';

break;

case 2:

const char\* filePath = "C:/laba4/ladna.txt";

ifstream file(filePath, ios::in);

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

cout << "Файл не открывается!!!!! \n";

return -1;

}

file.read((char\*)sequence, sizeof(sequence));

file.close();

break;

}

cout << my\_str\_len(sequence) << endl;

print\_result(sequence, my\_str\_len(sequence));

break;

case 2:

cout << "Измененный текст, очищенный от пробелов, знаков препинания и неправильного регистра" << endl;

delete\_space(sequence, my\_str\_len(sequence));

delete\_punctuation(sequence, my\_str\_len(sequence));

make\_lower(sequence, my\_str\_len(sequence));

cout << my\_str\_len(sequence) << endl;

print\_result(sequence, my\_str\_len(sequence));

break;

case 3:

cout << "Отсортированные по алфавиту слова:" << endl;

count = organize\_words(sequence, words);

sort\_alphabetically(words, count);

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

cout << words[i] << " ";

}

break;

case 4:

cout << "Вывести на экран количество символов в каждом слове исходной последовательности." << endl;

how\_many\_symbols\_in\_word(sequence, my\_str\_len(sequence));

break;

case 5:

system("cls");

cout << "Введите слово для линейного поиска" << endl;

cin.clear();

cin.sync();

keyboard\_input(pattern, my\_str\_len(pattern));

result = linear\_search(sequence, pattern, my\_str\_len(sequence));

if (result == -1) {

cout << "Нет таких слов в последовательности" << endl;

}

else {

cout << "Есть такая подстрока,начиная с индекса " << result << endl;

}

break;

case 6:

system("cls");

cout << "Введите слово для поиска" << endl;

cin.clear();

cin.sync();

keyboard\_input(pattern1, my\_str\_len(pattern1));

index = Boyer\_Moore\_algorithm(sequence, pattern1, my\_str\_len(sequence), table);

if (index != -1) {

cout << "Первое вхождение найдено на индексе: " << index << endl;

}

else {

cout << "Шаблон не найден." << endl;

}

break;

case 7:

cout << "Возвращаемся в основное меню!" << endl;

return 0;

default:

cout << "Некорректный номер задания." << endl;

break;

}

cout << "Задание выполнено. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}

void display\_1\_menu() {

cout << "Введите действие: " << endl;

cout << "1. 1 лаба" << endl;

cout << "2. 2 лаба" << endl;

cout << "3. 3 лаба" << endl;

cout << "4. 4 лаба" << endl;

}

int main() {

int i;

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (true) {

display\_1\_menu();

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

cin >> i;

switch (i) {

case 1:

system("cls");

main\_of\_laba1();

break;

case 2:

system("cls");

main\_of\_laba2();

break;

case 3:

system("cls");

main\_of\_laba3();

break;

case 4:

system("cls");

main\_of\_laba4();

break;

case 5:

cout << "Выход!" << endl;

return 0;

default:

cout << "Некорректный номер работы." << endl;

break;

}

cout << "Работа выполнена. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}