

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Лингвистическое и программное обеспечение

автоматизированных систем»

Отчёт по лабораторной работе № 2. Этюды для программистов

**Вариант № 10**

**Выполнил студент группы №475:**

Овчинников Роман Сергеевич

**Проверили преподаватели:**

доцент, к.т.н. Корниенко И.Г.

ст. преп., Федин А.К.

Санкт-Петербург

2020

*Постановка задачи*

Напишите программу, которая отображает целые числа на плоскость некоторым регулярным образом и отмечает на рисунке места, где находятся простые числа.

Выведите формулы, описывающие прямые линии на вашем рисунке, и напечатайте те из них, которые особенно изобилуют простыми числами; печатайте также долю простых чисел на этих прямых.

Обеспечьте высокую эффективность ваших программ проверки целых чисел на простоту, так чтобы вам хватило времени для анализа весьма отдаленных отрезков натурального ряда

*Исходные данные*

В качестве исходных данных программа принимает от пользователя размер квадратной матрицы и способ заполнения. Пользователь может заполнить массив самостоятельно с клавиатуры, сгенерировать случайным образом, автоматически заполнить последовательными цифрами или взять его из файла. В случае автоматического заполнения пользователь должен ввести начальный символ отсчёта.

*Особые ситуации*

Необходимо рассмотреть следующие особые ситуации.

* Везде, где программой ожидаются числа, если пользователь ввёл символ вместо числа, будет отображено уведомление с просьбой ввести число.
* Программа выводит сообщение, если выбран несуществующий пункт меню
* При вводе пользователем пути к несуществующему файлу, будет отображено уведомление о том, что не удалось найти файл и предложено повторить ввод.
* При сохранении строки в файл, если такой файл уже имеется – спрашивать у пользователя разрешение на перезапись. Путь к нему не должен содержать недопустимых символов.
* После выполнения программы пользователю будет предложено сохранить результат.
* Пользователю предоставляется выбор способа ввода: заполнить вручную, сгенерировать случайным образом, автоматически заполнить последовательными цифрами или взять массив из файла.
* Перед выходом из программы будет предложено вернуться в главное меню.

*Алгоритмы решения задачи:*

Для выполнения задачи был выбран следующий алгоритм:

При заполнении массива проверять каждое число на простоту и присваивать соответствующий индекс. Последовательно проверять каждое число на прямой линии и записывать количество простых чисел на линии по следующим формулам:

* Вертикально: y=a (y = const).
* Горизонтально: x = b (х = const).
* Диагонально (/): y = kх + b, где k = 1, b от –size – 1 до size – 1

(где size - размер квадратной матрицы).

* Диагонально (\): y = kх + b, где k = -1, b от 0 до size \* size – 2

(где size - размер квадратной матрицы).

*Форматы представления данных*

Таблица 1 – Переменные и массивы, используемые в программе.

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| sizeMatrix | int | Размер квадратной матрицы |
| fileName | string | Путь к файлу |
| text | string | Текст, который будет печататься другим цветом |
| counter | int | Количество цифр в квадратной матрице |
| arr | int\*\*\* | Массив содержащий квадратную матрицу и значение, определяющие простоту. |
| first | int | Переменная для смены выводимого сообщения |
| output | string | Сбор информации для вывода на экран |
| quantity | bool | Проверка на наличие простых чисел в линиях |
| color | int | Номер цвета |
| сhoice | int | Выбор пункта меню пользователем |

*Структура программы*

Программа разделена на 3 модулей:

* Main – модуль, осуществляющий запуск программы.
* Menu – модуль, осуществляющий пользовательский интерфейс.
* FileWork –– модуль, осуществляющий работу с файлами.
* PrimeNumber – модуль, осуществляющий проверку числа на простоту.
* Matrix – модуль в котором выполняются основные алгоритмы программы.

Таблица 2 – Функции модуля Menu

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Greeting | Вывод на экран данных о авторе программы и текста задачи. |
| IntegerCheck | Проверка на целые числа |
| MainMenu | Отображение и выбор пункта меню с последующим переходом к нему |
| YesNo | Отображение вопроса на который можно ответить “Да” и “Нет”, и возвращение ответа |
| AskSave | Скрашивает нужно ли пользователю сохранять |
| ColorChange | Изменение цвета вывода |
| ChoiceOfFilling | Ввод размера матрицы и обработка выбора ввода с вызовом основных методов |
| MainMenu | Вывод на экран главное меню программы с возможностью завершения или продолжения работы (Начать/Выход) |
| MainProgram | Вывод на экран второго меню программы с возможностью выбора способа ввода (Ручной/Случайный и др) |
| UserСhoice | Вывод на экран третьего меню программы с возможностью выбора метода ввода (Змейка/Улитка) |

Таблица 3 – Функции модуля FileWork

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FileInputFun | Заполнение массива из файла и вызов методов |
| FileCheck | Проверка на существование и корректность заполнения данных |

Таблица 4 – Функции модуля PrimeNumber

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| CheckingTheNumber | Проверка числа на простоту |

Таблица 5 – Методы модуля Matrix

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Matrix() | Конструктор по умолчанию |
| Matrix(int) | Конструктор выделяющий память под массив |
| Matrix(int\*\*\*&, int) | Конструктор для файлового ввода |
| ~Matrix() | Удаляет выделенный массив |
| GetArray | Возвращает массив |
| OutputScreen | Выводит массив на экран |
| SaveToFile | Сохраняет массив в файл |
| ManualInputFun | Ручной ввод массива |
| RandomInputFun | Случайное заполнение массива |
| AutoInputFun | Автоматическое заполнение массива со стартового значения |
| HorizontallyChecking | Проверка чисел по горизонтали |
| VerticallyChecking | Проверка чисел по вертикали |
| DiagonalCheckingDownLeft | Проверка чисел по диагонали снизу налево |
| DiagonalCheckingUpRight | Проверка чисел по диагонали сверху направо |

Модульные тесты программы в модуле UnitTest1.cpp:

• TestMethodCheckingTheNumber;

• TestMethodMatrixSnake;

• TestMethodMatrixSnail;

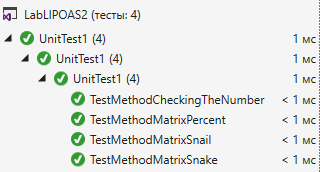
 • TestMethodMatrixPercent.

Рисунок 1 – Пройденные тесты

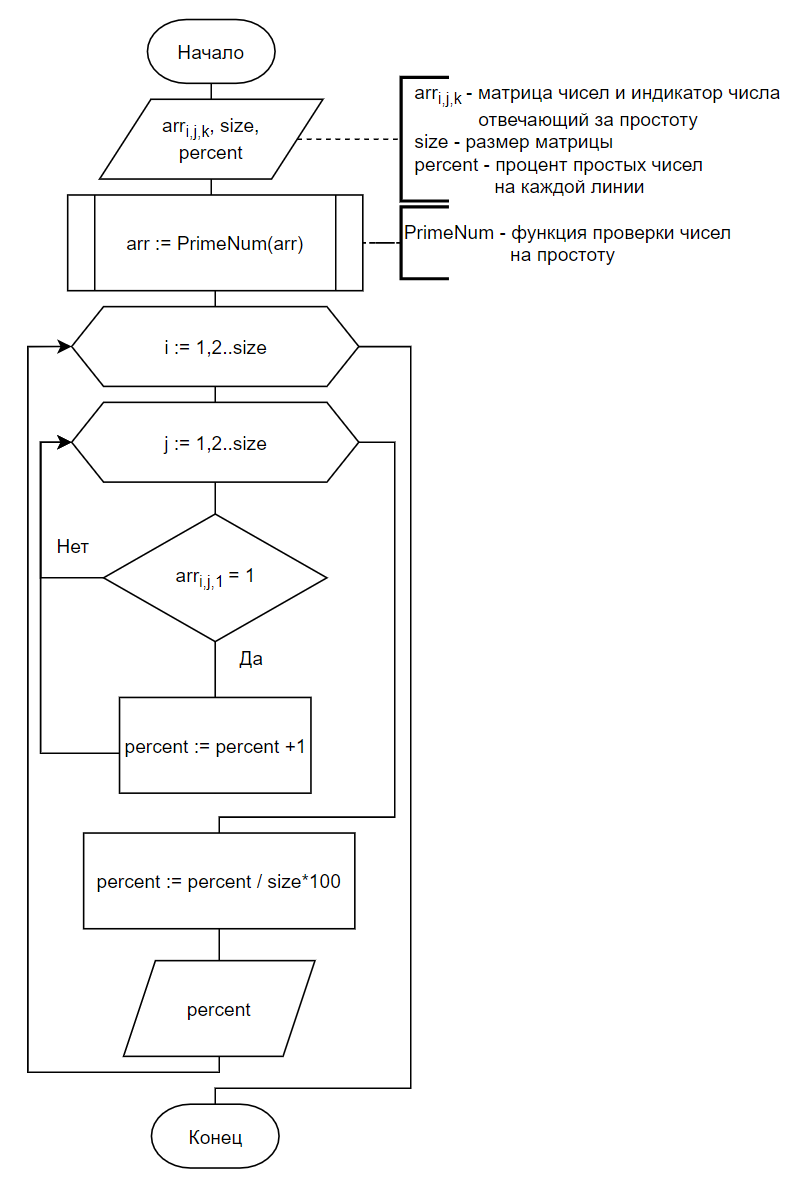
*Блок-схемы алгоритма программы*

Рисунок 2 – Блок-схема подсчёта по горизонтали или вертикали

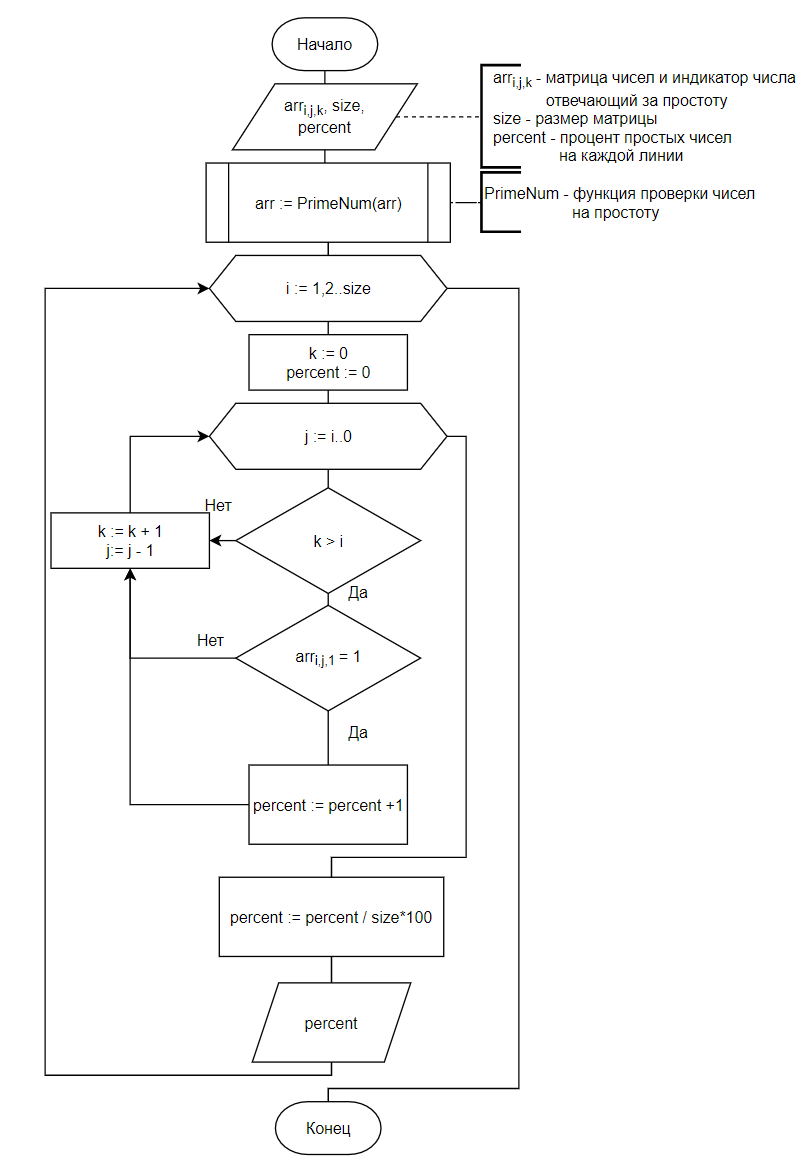
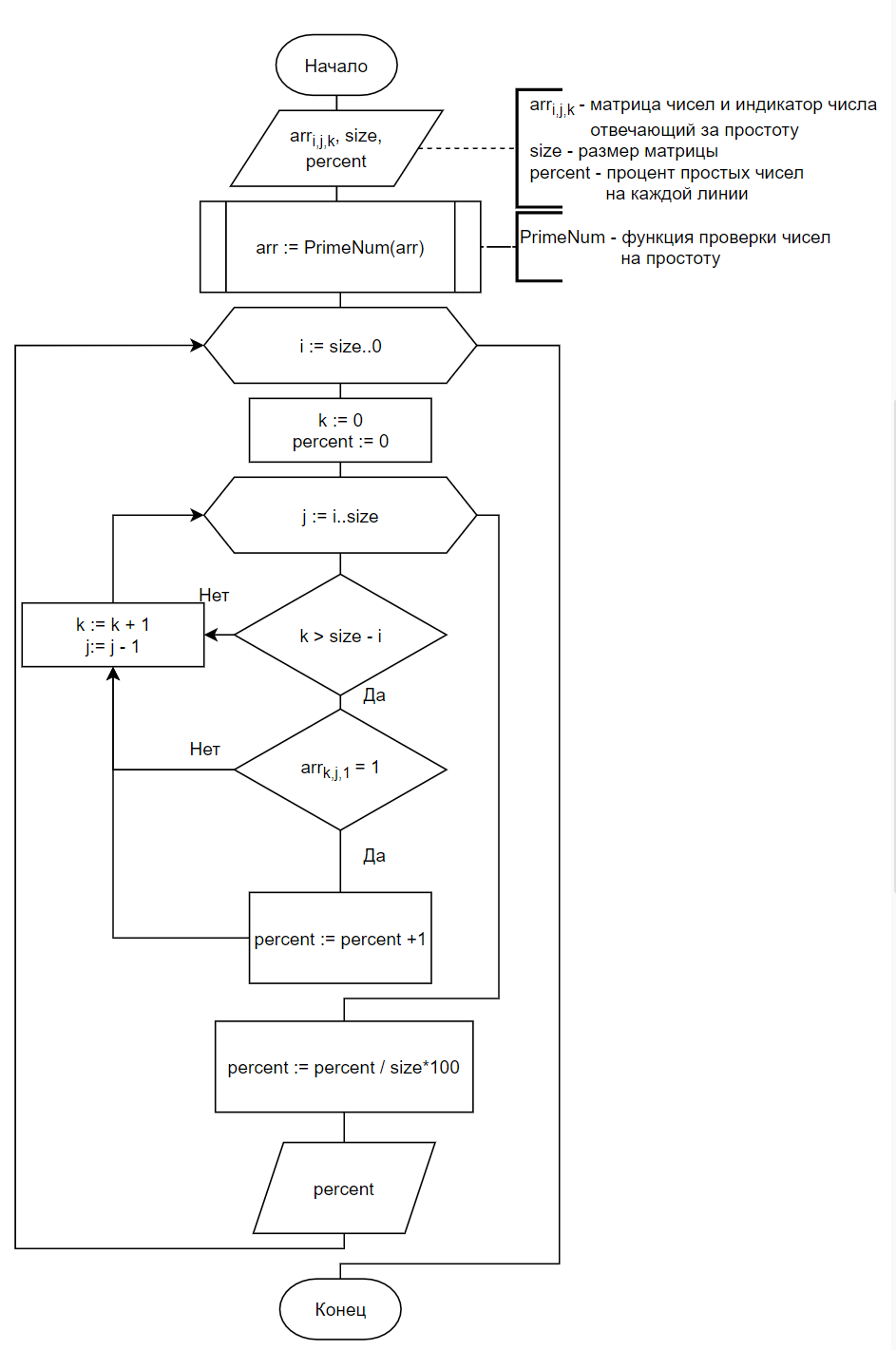
Рисунок 3 – Блок-схема подсчёта по диагонали снизу налево

Рисунок 3 – Блок-схема подсчёта по диагонали сверху направо 

*Описание хода выполнения лабораторной работы*

1. В ходе лабораторной работы было создано решение в интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был создан проект.
2. После набора теста программы выяснилось, что вывод текста на экран консольного приложения работает не правильно из-за различия кодировок консольного приложения и среды разработки.  
   Для решения этой проблемы была использована функция

setlocale(LC\_ALL, "Russian");

которая обеспечивает работу приложению с символами языка системы, в моём случае кириллицы.

1. При сохранении в файл, при вводе имён файлов с символами “/”, “\”, “|”, “\*”, “>” и т.д., а также некоторых слов, например, “aux”, “lpt1”, “com1”, “con”, выводилось сообщение о том, что файл сохранён, но в папке проекта никаких файлов не создавалось.  
   Для решения этой проблемы, я воспользовался функцией

is\_regular\_file(); которая расположена в библиотеке filesystem и содержит в себе все запрещённые имена.

*Результаты работы программы*

В результате работы программа выводит в консоль целые числа на некоторым регулярным образом и отмечает те, который являются простыми числами.

Также выведит формулы, описывающие прямые линии из простых чисел на матрице, и напечатает долю простых чисел на этих прямых.

Программа обеспечивает высокую эффективность проверки целых чисел на простоту.

*Исходный текст программы*

[Блок Main.cpp ---]

#include <iostream>

#include "Menu.h"

#include <Windows.h>

using namespace Menu;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

Greeting();

do {

} while (MainProgram(MainMenu(true)));

return 0;

}

[--- Конец блока Main.cpp]

[Блок Menu.cpp ---]

#include <Windows.h>

#include <iostream>

#include <string>

#include <fstream>

#include <filesystem>

#include <sstream>

#include "FileWork.h"

#include "Matrix.h"

using namespace std;

using namespace matrix;

namespace Menu {

// Пункты меню.

enum TypesSaving {

SaveArray = 1,

SaveResult,

};

enum InternalMenuMenu {

Start = 1,

Exit = 2,

};

enum Menu {

ManualInput = 1,

AutoInput,

RandomInput,

FileInput,

};

// Цвета

enum ConsoleColor {

LightGray = 7,

LightGreen = 10,

};

// Способ заполнения матрицы.

enum InputMethod {

Snake = 1,

Snail,

};

// Пункты меню.

enum InternalMenu {

Yes = 1,

No,

Again,

};

bool MainProgram(bool check);

// Приветствие

void Greeting() {

cout << "Лабораторная работа №2" << endl;

cout << "Студент группы № 475: Овчинников Роман" << endl << endl;

cout << " Напишите программу, которая отображает целые числа на плоскость" << endl;

cout << "некоторым регулярным образом и отмечает на рисунке места, где " << endl;

cout << "находятся простые числа. Выведите формулы, описывающие прямые " << endl;

cout << "линии на вашем рисунке, и напечатайте те из них, которые особенно" << endl;

cout << "изобилуют простыми числами; печатайте также долю простых чисел на " << endl;

cout << "этих прямых. Обеспечьте высокую эффективность ваших программ" << endl;

cout << "проверки целых чисел на простоту, так чтобы вам хватило времени для " << endl;

cout << "анализа весьма отдаленных отрезков натурального ряда (с. 88)." << endl << endl;

}

// Проверка на целочисленность. Принимает минимальное и максимальное значение.

int IntegerCheck(const int& min, const int& max) {

int tmp = 0;

cin >> tmp;

while (cin.fail()) {

cout << "Неверный ввод. Вводите только цифры. Повторите ввод:";

cin.clear();

cin.ignore(INT\_MAX, '\n');

cin >> tmp;

}

if ((tmp >= min) && (tmp <= max)) {

cin.ignore(INT\_MAX, '\n');

return tmp;

}

else {

cin.ignore(INT\_MAX, '\n');

cout << "Неверный ввод. Вводите цифры в пределах диапазона. Повторите ввод:";

tmp = IntegerCheck(min, max);

}

return tmp;

}

int YesNo() {

cout << "1. Да." << endl;

cout << "2. Нет." << endl;

cout << "Ваш выбор: ";

return IntegerCheck(Yes, No);

}

// Спросить нужно ли сохранять

int AskSave() {

cout << endl;

cout << "Хотите сохранить в файл?" << endl;

return YesNo();

}

// Изменение цвета

void ColorChange(string text, int color) {

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

SetConsoleTextAttribute(hConsoleHandle, (WORD)color);

cout << text;

SetConsoleTextAttribute(hConsoleHandle, LightGray);

}

void ChoiceOfFilling(int inputMethod, int choiceFilling = 0) {

string buf;

int sizeMatrix = 0;

if (inputMethod == ManualInput) {

cout << "Введите размер квадратной матрицы (от 2 до 5): ";

sizeMatrix = IntegerCheck(2, 5);

}

else {

cout << "Введите размер квадратной матрицы (от 2 до 50): ";

sizeMatrix = IntegerCheck(2, 50);

}

Matrix arr = Matrix(sizeMatrix);

if (inputMethod == ManualInput) {

cout << "Вводите матрицу (Каждый символ с новой строки от 1 до 90000): " << endl;

arr.ManualInputFun(choiceFilling);

cout << "Вы ввели: " << endl;

arr.OutputScreen();

}

if (inputMethod == AutoInput) {

cout << "Введите начальный символ (от 1 до 90000): " << endl;

int startNum = IntegerCheck(1, 90000);

arr.AutoInputFun(startNum, choiceFilling);

cout << "Вы ввели: " << endl;

arr.OutputScreen();

}

if (inputMethod == RandomInput) {

arr.RandomInputFun();

cout << "Зелёным цветом выделены натуральные числа" << endl;

cout << "Вы ввели: " << endl;

arr.OutputScreen();

}

string output = "";

if (AskSave() == Yes) {

arr.SaveToFile(SaveArray, output);

}

cout << endl << "Результат: " << endl;

arr.HorizontallyChecking(output);

arr.VerticallyChecking(output);

arr.DiagonalCheckingDownLeft(output);

arr.DiagonalCheckingUpRight(output);

cout << output << endl;

if (AskSave() == Yes) {

arr.SaveToFile(SaveResult, output);

}

}

// Пользовательский выбор способа запонения матрицы

void UserСhoice(int inputMethod) {

cout << "Выберите способ заполнения матрицы:" << endl;

cout << "1. Змейка." << endl;

cout << "2. Улитка." << endl;

cout << endl << "Ваш выбор: ";

int const NumberItemsMin = 1;

int const NumberItemsMax = 2;

switch (IntegerCheck(NumberItemsMin, NumberItemsMax)) {

case(Snake):

ChoiceOfFilling(inputMethod, Snake);

break;

case(Snail):

ChoiceOfFilling(inputMethod, Snail);

break;

}

}

// Главное меню

bool MainMenu(bool сhoice) {

if (сhoice) {

cout << "1. Начать работу." << endl;

}

else {

cout << endl << "Попробовать еще раз?" << endl;

cout << "1. Начать заново" << endl;

}

cout << "2. Выход." << endl << endl;

cout << "Ваш выбор: ";

int const NumberItemsMin = 1;

int const NumberItemsMax = 2;

switch (IntegerCheck(NumberItemsMin, NumberItemsMax)) {

case(Start):

cout << endl;

MainProgram(true);

break;

case(Exit):

return false;

}

return false;

}

bool MainProgram(bool check) {

if (check == true) {

cout << "1. Ручной ввод." << endl;

cout << "2. Автоматическое заполнение." << endl;

cout << "3. Случайное заполнение." << endl;

cout << "4. Ввод из файла." << endl;

cout << endl << "Ваш выбор: ";

int const NumberItemsMin = 1;

int const NumberItemsMax = 4;

switch (IntegerCheck(NumberItemsMin, NumberItemsMax)) {

case(ManualInput):

UserСhoice(ManualInput);

break;

case(AutoInput):

UserСhoice(AutoInput);

break;

case(RandomInput):

ChoiceOfFilling(RandomInput);

break;

case(FileInput):

FileWork::FileInputFun();

break;

}

MainMenu(false);

}

return false;

}

}

[--- Конец блока Menu.cpp]

[---Блок FileWork.cpp---]

#include <Windows.h>

#include <iostream>

#include <string>

#include <fstream>

#include <filesystem>

#include "Menu.h"

#include "PrimeNumber.h"

#include "Matrix.h"

using namespace std;

using namespace Menu;

using namespace PrimeNumber;

using namespace matrix;

namespace FileWork {

// Цвета

enum ConsoleColors {

LightGray = 7,

LightGreen = 10,

LightRed = 12,

};

// Пункты меню.

enum IntMenu {

Yes = 1,

No,

};

// Проверка файла

bool FileCheck(string fileName) {

ifstream file;

file.open(fileName);

int sizeMatrix, check;

int counter = 0;

file >> sizeMatrix;

const int MaxSize = 50;

const int MinSizeLines = 2;

const int NumberMax = 1000000;

const int NumberMin = 0;

if ((sizeMatrix <= MaxSize) && (sizeMatrix >= MinSizeLines)) {

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

if (!file.eof()) {

if (file >> check) {

if ((check <= NumberMax) && (check >= NumberMin)) {

counter += 1;

if (counter == sizeMatrix \* sizeMatrix) {

return true;

break;

}

}

else {

return false;

break;

}

}

}

}

}

else {

return 0;

}

if (counter < sizeMatrix\*sizeMatrix) {

return false;

}

file.close();

return false;

}

void FileInputFun() {

int sizeMatrix = 0;

string fileName = "";

cout << "Введите полный путь к файлу: ";

getline(cin, fileName);

ifstream file;

file.open(fileName);

if (std::experimental::filesystem::is\_regular\_file(fileName) && file.good() && (FileCheck(fileName) == true)) {

ColorChange("Файл открыт!", LightGreen);

cout << endl;

if (!file.eof()) {

file >> sizeMatrix;

int \*\*\*arrFile = new int\*\*[sizeMatrix];

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

{

arrFile[i] = new int\*[sizeMatrix];

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

{

arrFile[i][j] = new int[2];

arrFile[i][j][1] = 0;

file >> arrFile[i][j][0];

arrFile[i][j][1] = CheckingTheNumber(arrFile[i][j][0]);

}

}

Matrix arr = Matrix(arrFile, sizeMatrix);

for (size\_t i = 0; i < sizeMatrix; i++)

{

for (size\_t j = 0; j < sizeMatrix; j++)

delete[] arrFile[i][j];

delete[] arrFile[i];

}

delete[] arrFile;

cout << "Введённая матрица:" << endl;

arr.OutputScreen();

cout << endl << "Результат: " << endl;

string output = "";

arr.HorizontallyChecking(output);

arr.VerticallyChecking(output);

arr.DiagonalCheckingDownLeft(output);

arr.DiagonalCheckingUpRight(output);

cout << output << endl;

}

else {

ColorChange("Формат файла не поддерживается!", LightRed);

cout << endl;

}

}

else {

ColorChange("Ошибка открытия файла!", LightRed);

cout << endl;

}

}

}

[--- Конец блока Sourse.cpp---]

[--- Начало блока Matrix.cpp---]

#include "Matrix.h"

#include "Menu.h"

#include <string>

#include <iostream>

#include "PrimeNumber.h"

#include <ctime>

#include <tuple>

using std::cout; // пространство имен std для cout

using std::cin; // пространство имен std для cin

using std::endl; // пространство имен std для endl

using std::to\_string;

using std::string;

using std::ifstream;

using namespace std;

using Menu::IntegerCheck;

using Menu::ColorChange;

using Menu::YesNo;

using namespace PrimeNumber;

// Способ заполнения матрицы.

namespace matrix {

enum InputMethods {

Snake = 1,

Snail,

};

// Пункты меню.

enum TypeSaving {

SaveArray = 1,

SaveResult,

};

// Цвета

enum ConsoleColors {

LightGray = 7,

LightGreen = 10,

LightRed = 12,

Yellow = 14,

};

// Пункты меню.

enum IntMenu {

Yes = 1,

No,

};

Matrix::Matrix()

{

for (size\_t i = 0; i < sizeMatrix; i++)

{

arr[i] = new int\*[sizeMatrix];

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

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

}

}

}

Matrix::Matrix(int\*\*\* &arrFile, int size) {

this->sizeMatrix = size;

arr = new int\*\*[sizeMatrix]();

for (size\_t i = 0; i < sizeMatrix; i++)

{

arr[i] = new int\*[sizeMatrix];

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = arrFile[i][j][1];

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

}

}

}

Matrix::Matrix(int sizeMatrix)

{

this->sizeMatrix = sizeMatrix;

arr = new int\*\*[sizeMatrix](); // выделить место в памяти для множества

}

int\*\*\* Matrix::GetArray() {

return arr;

}

void Matrix::OutputScreen() {

cout << " ^" << endl;

for (size\_t i = 0; i < sizeMatrix; i++)

{

cout << sizeMatrix - (i + 1) << "|\t";

for (size\_t j = 0; j < sizeMatrix; j++)

{

if (arr[i][j][1] == 1) {

ColorChange(to\_string(arr[i][j][0]), LightGreen);

cout << '\t';

}

else

cout << arr[i][j][0] << '\t';

if (j == sizeMatrix - 1)

cout << endl << endl;

}

}

cout << " |" << endl;

for (size\_t i = 0; i < sizeMatrix + 1; i++)

{

cout << " —\t";

}

cout << ">" << endl;

for (size\_t i = 0; i < sizeMatrix; i++)

{

cout << "\t " << i;

}

}

void Matrix::SaveToFile(int choiceTypeSave, const string &output) {

bool is\_file\_save\_successful = false;

string path;

bool first = true;

do {

if (first) {

cout << endl << "Если вы введёте только имя файла, то он будет в папке с проектом." << endl;

cout << "Для сохрарения файла в другом месте укажите полный путь к файлу." << endl;

cout << "(По умолчанию в конец добавляется .txt)" << endl << endl;

first = false;

}

cout << endl << "Введите путь к файлу: ";

cin >> path;

path += ".txt";

bool is\_file\_exist = false;

if (ifstream(path)) // проверяем на сущ.

is\_file\_exist = true;

ofstream file;

file.open(path, ios::app);

if (file.good()) {

if (!std::experimental::filesystem::is\_regular\_file(path)) {

ColorChange("Недопустимое имя файла.", LightRed);

file.close();

continue;

}

if (is\_file\_exist) {

cout << "Данный файл существует! Перезаписать?" << endl;

if (YesNo() == No) {

file.close();

continue;

}

}

file.close();

file.open(path, ios::trunc);

file << sizeMatrix << '\n';

for (size\_t i = 0; i < sizeMatrix; i++) {

for (size\_t j = 0; j < sizeMatrix; j++) {

file << " " << arr[i][j][0];

}

file << '\n';

}

if (choiceTypeSave == SaveResult) {

file << output;

}

file.close();

is\_file\_save\_successful = true;

ColorChange("Файл успешно записан!", LightGreen);

cout << endl;

}

else {

ColorChange("Ошибка при открытии файла.", LightRed);

file.close();

}

} while (!is\_file\_save\_successful);

}

void Matrix::ManualInputFun(int choiceFilling) {

const int min = 1;

const int max = 100000;

if (choiceFilling == Snake) {

for (size\_t i = 0; i < sizeMatrix; i++)

{

arr[i] = new int\*[sizeMatrix]; // не понимаю зачем я опять выделяю память если выделил в конструкторе?

if (i % 2 == 0) {

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2]; // Однако без этого не робит. Спросить.

arr[i][j][1] = 0;

arr[i][j][0] = IntegerCheck(min, max);

arr[i][j][1] = PrimeNumber::CheckingTheNumber(arr[i][j][0]);

}

}

else {

for (int j = static\_cast<int>(sizeMatrix) - 1; j >= 0; j--)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

arr[i][j][0] = IntegerCheck(min, max);

arr[i][j][1] = PrimeNumber::CheckingTheNumber(arr[i][j][0]);

}

}

}

}

if (choiceFilling == Snail) {

for (size\_t i = 0; i < sizeMatrix; i++)

arr[i] = new int\*[sizeMatrix];

for (size\_t i = 0; i < sizeMatrix; i++)

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

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

}

int count = 0;

for (int y = 0; y < sizeMatrix; y++) {

count++;

arr[0][y][0] = IntegerCheck(min, max);

arr[0][y][1] = PrimeNumber::CheckingTheNumber(arr[0][y][0]);

}

for (int x = 1; x < sizeMatrix; x++) {

count++;

arr[x][sizeMatrix - 1][0] = IntegerCheck(min, max);

arr[x][sizeMatrix - 1][1] = PrimeNumber::CheckingTheNumber(arr[x][sizeMatrix - 1][0]);

}

for (int y = static\_cast<int>(sizeMatrix) - 2; y >= 0; y--) {

count++;

arr[static\_cast<int>(sizeMatrix) - 1][y][0] = IntegerCheck(min, max);

arr[static\_cast<int>(sizeMatrix) - 1][y][1] = PrimeNumber::CheckingTheNumber(arr[static\_cast<int>(sizeMatrix) - 1][y][0]);

}

for (int x = static\_cast<int>(sizeMatrix) - 2; x > 0; x--) {

count++;

arr[x][0][0] = IntegerCheck(min, max);

arr[x][0][1] = PrimeNumber::CheckingTheNumber(arr[x][0][0]);

}

int c = 1;

int d = 1;

for (size\_t i = count; i < sizeMatrix \* sizeMatrix; i++)

{

//Движемся вправо.

while (arr[c][d + 1][0] == 0) {

arr[c][d][0] = IntegerCheck(min, max);

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

d++;

}

//Движемся вниз.

while (arr[c + 1][d][0] == 0) {

arr[c][d][0] = IntegerCheck(min, max);

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

c++;

}

//Движемся влево.

while (arr[c][d - 1][0] == 0) {

arr[c][d][0] = IntegerCheck(min, max);

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

d--;

}

//Движемся вверх.

while (arr[c - 1][d][0] == 0) {

arr[c][d][0] = IntegerCheck(min, max);

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

c--;

}

}

for (int x = 0; x < sizeMatrix; x++) {

for (int y = 0; y < sizeMatrix; y++) {

if (arr[x][y][0] == 0) {

arr[x][y][0] = IntegerCheck(min, max);

arr[x][y][1] = PrimeNumber::CheckingTheNumber(arr[x][y][0]);

}

}

}

}

}

void Matrix::RandomInputFun() {

srand((unsigned)time(0));

for (size\_t i = 0; i < sizeMatrix; i++)

{

arr[i] = new int\*[sizeMatrix];

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

arr[i][j][0] = rand() % 100000;

arr[i][j][1] = PrimeNumber::CheckingTheNumber(arr[i][j][0]);

}

}

}

void Matrix::AutoInputFun(int startNum, int choiceFilling) {

if (choiceFilling == Snake) {

for (size\_t i = 0; i < sizeMatrix; i++)

{

arr[i] = new int\*[sizeMatrix];

if (i % 2 == 0) {

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

arr[i][j][0] = startNum++;

arr[i][j][1] = PrimeNumber::CheckingTheNumber(arr[i][j][0]);

}

}

else {

for (int j = static\_cast<int>(sizeMatrix) - 1; j >= 0; j--)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

arr[i][j][0] = startNum++;

arr[i][j][1] = PrimeNumber::CheckingTheNumber(arr[i][j][0]);

}

}

}

}

if (choiceFilling == Snail) {

for (size\_t i = 0; i < sizeMatrix; i++)

arr[i] = new int\*[sizeMatrix];

for (size\_t i = 0; i < sizeMatrix; i++)

for (size\_t j = 0; j < sizeMatrix; j++)

{

arr[i][j] = new int[2];

arr[i][j][1] = 0;

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

}

int count = 0;

for (int y = 0; y < sizeMatrix; y++) {

count++;

arr[0][y][0] = startNum++;

arr[0][y][1] = PrimeNumber::CheckingTheNumber(arr[0][y][0]);

}

for (int x = 1; x < sizeMatrix; x++) {

count++;

arr[x][sizeMatrix - 1][0] = startNum++;

arr[x][sizeMatrix - 1][1] = PrimeNumber::CheckingTheNumber(arr[x][sizeMatrix - 1][0]);

}

for (int y = static\_cast<int>(sizeMatrix) - 2; y >= 0; y--) {

count++;

arr[static\_cast<int>(sizeMatrix) - 1][y][0] = startNum++;

arr[static\_cast<int>(sizeMatrix) - 1][y][1] = PrimeNumber::CheckingTheNumber(arr[static\_cast<int>(sizeMatrix) - 1][y][0]);

}

for (int x = static\_cast<int>(sizeMatrix) - 2; x > 0; x--) {

count++;

arr[x][0][0] = startNum++;

arr[x][0][1] = PrimeNumber::CheckingTheNumber(arr[x][0][0]);

}

int c = 1;

int d = 1;

for (size\_t i = count; i < sizeMatrix \* sizeMatrix; i++)

{

//Движемся вправо.

while (arr[c][d + 1][0] == 0) {

arr[c][d][0] = startNum++;

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

d++;

}

//Движемся вниз.

while (arr[c + 1][d][0] == 0) {

arr[c][d][0] = startNum++;

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

c++;

}

//Движемся влево.

while (arr[c][d - 1][0] == 0) {

arr[c][d][0] = startNum++;

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

d--;

}

//Движемся вверх.

while (arr[c - 1][d][0] == 0) {

arr[c][d][0] = startNum++;

arr[c][d][1] = PrimeNumber::CheckingTheNumber(arr[c][d][0]);

c--;

}

}

for (int x = 0; x < sizeMatrix; x++) {

for (int y = 0; y < sizeMatrix; y++) {

if (arr[x][y][0] == 0) {

arr[x][y][0] = startNum;

arr[x][y][1] = PrimeNumber::CheckingTheNumber(arr[x][y][0]);

}

}

}

}

}

void Matrix::HorizontallyChecking(string &output) {

bool quantity = false;

output += "Горизонтально(—):\n";

output += "y = a, (y = const)\n";

for (size\_t i = 0; i < sizeMatrix; i++)

{

int count = 0;

size\_t j = 0;

for (; j < sizeMatrix; j++)

{

if (arr[i][j][1] == 1)

count++;

}

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При а: " + to\_string(sizeMatrix - i - 1);

output += " = ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

}

if (!quantity) {

output += "Все значения меньше ";

output += to\_string(percent);

output += "%\n";

}

}

void Matrix::VerticallyChecking(string &output) {

bool quantity = false;

output += "Вертикально(|):\n";

output += "x = b, х = const\n";

for (size\_t i = 0; i < sizeMatrix; i++)

{

int count = 0;

size\_t j = 0;

for (; j < sizeMatrix; j++)

{

if (arr[j][i][1] == 1)

count++;

}

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При b: " + to\_string(i);

output += " = ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

}

if (!quantity) {

output += "Все значения меньше ";

output += to\_string(percent);

output += "%\n";

}

}

void Matrix::DiagonalCheckingDownLeft(string &output) {

bool quantity = false;

output += "Диагонально(/):\n";

output += "y = х + b\n";

int countPrintX = 0;

int b = sizeMatrix - 1;

for (int i = 0; i < static\_cast<int>(sizeMatrix); i++) // переделывал формат, чтобы убрать предупреждение компилятора.

{

countPrintX++;

size\_t k = 0;

int count = 0;

for (int j = i; j >= 0; j--)

{

for (; k <= i;)

{

if (arr[k][j][1] == 1)

count++;

k++; // я не знаю почему, но если вынести k сюды, ошибка "warning C4702: недостижимый код" не возникает

break;

}

}

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При b = ";

output += to\_string(b);

output += ": ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

b--;

}

size\_t l = 1;

for (size\_t i = sizeMatrix - 1; i > 0; i--)

{

countPrintX++;

size\_t k = l;

int count = 0;

for (size\_t j = sizeMatrix - 1; j >= l; j--)

{

for (; k <= sizeMatrix - 1;)

{

if (arr[k][j][1] == 1)

count++;

k++; // я не знаю почему, но если вынести k сюды, ошибка "warning C4702: недостижимый код" не возникает

break;

}

}

l++;

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При b = ";

output += to\_string(b);

output += ": ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

b--;

}

if (!quantity) {

output += "Все значения меньше ";

output += to\_string(percent);

output += "%\n";

}

}

void Matrix::DiagonalCheckingUpRight(string &output) {

bool quantity = false;

output += "Диагонально(\\):\n";

output += "y = -1 \* х + b\n";

int b = sizeMatrix + sizeMatrix - 2;

int countPrintX = 0;

for (int i = static\_cast<int>(sizeMatrix) - 1; i >= 0; i--)

{

int count = 0;

size\_t k = 0;

countPrintX++;

for (int j = i; j < static\_cast<int>(sizeMatrix); j++)

{

for (; k < sizeMatrix - i;)

{

if (arr[k][j][1] == 1)

count++;

k++; // я не знаю почему, но если вынести k сюды, ошибка "warning C4702: недостижимый код" не возникает

break;

}

}

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При b = ";

output += to\_string(b);

output += ": ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

b--;

}

size\_t l = 1;

for (size\_t i = 0; i < sizeMatrix - 1; i++) {

int count = 0;

size\_t k = l;

countPrintX++;

for (size\_t j = 0; j < sizeMatrix - i; j++) {

for (; k < sizeMatrix;)

{

if (arr[k][j][1] == 1)

count++;

k++; // я не знаю почему, но если вынести k сюды, ошибка "warning C4702: недостижимый код" не возникает

break;

}

}

l++;

if (static\_cast<int>((double)count / (double)sizeMatrix\*100.0) > percent) {

quantity = true;

output += "При b = ";

output += to\_string(b);

output += ": ";

output += to\_string(static\_cast<int>((double)count / (double)sizeMatrix\*100.0));

output += "% (" + to\_string(count);

output += "шт)\n";

}

b--;

}

if (!quantity) {

output += "Все значения меньше ";

output += to\_string(percent);

output += "%\n";

}

}

Matrix::~Matrix()

{

for (size\_t i = 0; i < sizeMatrix; i++)

{

for (size\_t j = 0; j < sizeMatrix; j++)

delete[] arr[i][j];

delete[] arr[i];

}

delete[] arr;

}

}

[--- Конец блока Matrix.cpp---]

[--- Начало блока PrimeNumber.cpp ---]

#include <iostream>

namespace PrimeNumber {

int CheckingTheNumber(int num) {

if(num <= 3)

return 1;

for (int i = 2; i <= sqrt(num); i++) if ((num%i) == 0) return 0;

return 1;

}

}

[--- Конец блока PrimeNumber.cpp ---]

[--- Начало блока Unittest1.cpp]

#include "stdafx.h"

#include "CppUnitTest.h"

#include "../LabLIPOAS2/PrimeNumber.h"

#include "../LabLIPOAS2/PrimeNumber.Cpp"

#include "../LabLIPOAS2/Matrix.h"

#include "../LabLIPOAS2/Matrix.cpp"

#include "../LabLIPOAS2/Menu.cpp"

#include "../LabLIPOAS2/Menu.h"

#include "../LabLIPOAS2/FileWork.h"

#include "../LabLIPOAS2/FileWork.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

using namespace matrix;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethodCheckingTheNumber)

{

int number = 701;

int result = 1;

Assert::AreEqual(PrimeNumber::CheckingTheNumber(number), result);

}

TEST\_METHOD(TestMethodMatrixSnake)

{

int sizeMatrix = 2; // Размер матрицы

int startNum = 1; // Стартовое число заполнения

int choiceFilling = 1; // Выбор заполнения -- змейка

int resultArray[2][2][2] = { {{1,1}, {2,1}}, {{4,0}, {3,1}} }; // Размер sizeMatrix\*sizeMatrix. Третье измерение для проверки на натуральность

Matrix arr = Matrix(sizeMatrix);

arr.AutoInputFun(startNum, choiceFilling);

for (size\_t i = 0; i < sizeMatrix; i++)

{

for (size\_t j = 0; j < sizeMatrix; j++)

{

for (size\_t k = 0; k < 2; k++)

{

Assert::AreEqual(arr.GetArray()[i][j][k], resultArray[i][j][k]);

}

}

}

}

TEST\_METHOD(TestMethodMatrixSnail)

{

int sizeMatrix = 3; // Размер матрицы

int startNum = 101; // Стартовое число заполнения

int choiceFilling = 2; // Выбор заполнения -- улитка

int resultArray[3][3][2] = { {{101,1}, {102,0}, {103,1}}, {{108,0}, {109,1}, {104,0}}, {{107,1}, {106,0}, {105,0}} }; // Размер sizeMatrix\*sizeMatrix. Третье измерение для проверки на натуральность

Matrix arr = Matrix(sizeMatrix);

arr.AutoInputFun(startNum, choiceFilling);

for (size\_t i = 0; i < sizeMatrix; i++)

{

for (size\_t j = 0; j < sizeMatrix; j++)

{

for (size\_t k = 0; k < 2; k++)

{

Assert::AreEqual(arr.GetArray()[i][j][k], resultArray[i][j][k]);

}

}

}

}

TEST\_METHOD(TestMethodMatrixPercent)

{

int sizeMatrix = 2; // Размер матрицы

int startNum = 1; // Стартовое число заполнения

int choiceFilling = 1; // Выбор заполнения -- улитка

string resultOutput = "Горизонтально(—):\ny = a, (y = const)\nПри а: 1 = 100% (2шт)\nПри а: 0 = 50% (1шт)\nВертикально(|):\nx = b, х = const\nПри b: 0 = 50% (1шт)\nПри b: 1 = 100% (2шт)\nДиагонально(/):\ny = х + b\nПри b = 1: 50% (1шт)\nПри b = 0: 50% (1шт)\nПри b = -1: 50% (1шт)\nДиагонально(\\):\ny = -1 \* х + b\nПри b = 2: 50% (1шт)\nПри b = 1: 100% (2шт)\n";

string output{};

Matrix arr = Matrix(sizeMatrix);

arr.AutoInputFun(startNum, choiceFilling);

arr.HorizontallyChecking(output);

arr.VerticallyChecking(output);

arr.DiagonalCheckingDownLeft(output);

arr.DiagonalCheckingUpRight(output);

Assert::AreEqual(resultOutput, output);

}

};

}

[--- Конец блока Unittest1.cpp]