Белорусский Национальный Технический Университет

Кафедра «Инновационные процессы»

Лабораторная работа № 3

 «Компиляция и отладка простейшего приложения в Linux. Автоматическая сборка многофайловых проектов»

|  |
| --- |
| Выполнил: студент группы 42221  Мельник К.А.  Проверил:  Станкевич С.Н. |

МИНСК 2023

Цель работы:

Изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов gcc/g++, отладчика gdb и других для проектирования, компиляции, отладки и запуска приложений на языке программирования С/C++. Научиться грамотно проектировать и разрабатывать многофайловые проекты на языке программирования С/C++ с использованием архитектурного шаблона проектирования и разработки гибких и масштабируемых приложений Model-View-Controller (MVC). Научиться эффективно использовать специальные средства для автоматизации процесса компиляции, сборки и запуска многофайловых проектов.

Задания:

**Задание 1:**

Изучите и выполните требования, представленные в лабораторной работе №2 «Компиляция и отладка простейшего приложения в Linux». Разработайте исходные коды приложений в соответствии с заданиями. Компиляцию приложений проведите с помощью консольных команд.

Вар10.

В одномерном массиве, состоящем из n вещественных элементов, вычислить: минимальный элемент массива и сумму элементов массива, расположенных между первым и последним положительными элементами.

Листинг кода:

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

// Вводим размер массива

int n;

cout << "Введите размер массива: ";

cin >> n;

// Проверка на корректность размера массива

if (n <= 0) {

cout << "Размер массива должен быть положительным числом." << endl;

return 1;

}

// Создаем массив вещественных чисел

double\* arr = new double[n];

// Вводим элементы массива

cout << "Введите элементы массива:" << endl;

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

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

cin >> arr[i];

}

// Ищем первый положительный элемент

int firstPositiveIndex = -1;

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

if (arr[i] > 0) {

firstPositiveIndex = i;

break;

}

}

// Если нет положительных элементов

if (firstPositiveIndex == -1) {

cout << "В массиве нет положительных элементов." << endl;

delete[] arr; // Освобождаем память

return 0;

}

// Ищем последний положительный элемент

int lastPositiveIndex = -1;

for (int i = n - 1; i >= 0; i--) {

if (arr[i] > 0) {

lastPositiveIndex = i;

break;

}

}

// Если первый положительный элемент идет после последнего

if (firstPositiveIndex >= lastPositiveIndex) {

cout << "В массиве нет положительных элементов, расположенных друг за другом." << endl;

delete[] arr; // Освобождаем память

return 0;

}

// Находим минимальный элемент массива

double minElement = arr[0];

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

if (arr[i] < minElement) {

minElement = arr[i];

}

}

// Вычисляем сумму элементов между первым и последним положительными элементами

double sumBetweenPositives = 0;

for (int i = firstPositiveIndex + 1; i < lastPositiveIndex; i++) {

sumBetweenPositives += arr[i];

}

// Выводим результаты

cout << "Минимальный элемент массива: " << minElement << endl;

cout << "Сумма элементов между первым и последним положительными: " << sumBetweenPositives << endl;

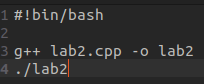
// Освобождаем память

delete[] arr;

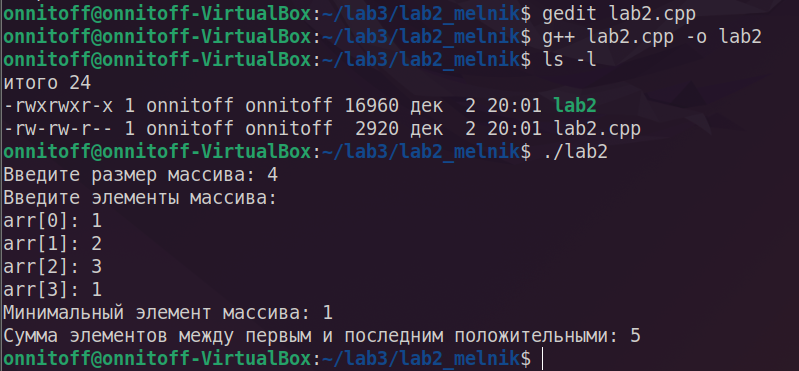
return 0;

}

Скрипт компиляции и выполнения:



Скриншот выполнения программы:



**Задание 2:**

Разработать интерактивную программу «Try to Guess the Number» («Попробуй угадать число»), которая эмулирует классическую игру на отгадывание числа. Суть программы (игры) сводиться к следующему: компьютер генерирует случайное число из диапазона, к примеру, от 1 до 100, а пользователь (далее игрок) пытается отгадать число за минимальное количество попыток. При каждой очередной попытке компьютер «подсказывает» игроку, как соизмеряется предложенный вариант игрока с действительным загаданным компьютером числом: загаданное число больше или меньше указанного (higher/lower). Как только игрок отгадывает число, компьютер должен «поздравить» его с выводом на экран угаданного числа и количество затраченных игроком попыток. Далее компьютер может «предложить» повторно сыграть игру или выйти из программы. Для универсальности предложенной программы можно добавить возможность выбора диапазона генерирования компьютером случайных чисел, а также задания ограничения на количество попыток. В случае, если игрок не укладывается в заданное количество попыток (т.е. проигрывает), программа должна выводить суровую надпись «Game Over».

Листниг кода:

Файл: main.cpp

#include "controller.h"

#include <locale.h>

int main() {

setlocale(LC\_ALL, "Russian");

const int minRange = 1;

const int maxRange = 100;

const int maxAttempts = 10;

GameController gameController(minRange, maxRange, maxAttempts);

gameController.startGame();

return 0;

}

Файл: model.cpp

#include <cstdlib>

#include <ctime>

#include "model.h"

NumberGame::NumberGame(int minRange, int maxRange, int maxAttempts)

: maxAttempts(maxAttempts), attempts(0) {

srand(static\_cast<unsigned>(time(nullptr)));

targetNumber = rand() % (maxRange - minRange + 1) + minRange;

}

int NumberGame::getTargetNumber() const {

return targetNumber;

}

int NumberGame::getAttempts() const {

return attempts;

}

bool NumberGame::isGameOver() const {

return attempts >= maxAttempts;

}

bool NumberGame::guessNumber(int guess) {

attempts++;

return guess == targetNumber;

}

Файл: view.cpp

#include "view.h"

#include <iostream>

using namespace std;

void GameView::showWelcomeMessage() const {

cout << "Добро пожаловать в игру «Попробуй угадать число»!" << std::endl;

}

void GameView::showGuessPrompt() const {

cout << "Введите свое предположение: ";

}

void GameView::showResult(bool isCorrect, int targetNumber, int attempts) const {

if (isCorrect) {

cout << "Поздравляем! Вы угадали число " << targetNumber

<< " за " << attempts << " попыток" << endl;

}

else {

cout << "Игра закончена! Правильный номер был " << targetNumber << "." << endl;

}

}

void GameView::showTryAgainPrompt() const {

cout << "Хотите сыграть еще раз? (y/n): ";

}

void GameView::showGoodbyeMessage() const {

cout << "Спасибо за игру! До свидания." << endl;

}

void GameView::showHigherLowerHint(bool isHigher) const {

if (isHigher) {

cout << "Попробуйте большее число." << endl;

}

else {

cout << "Попробуйте меньшее число." << endl;

}

}

Файл: controller.cpp

#include <iostream>

#include <limits>

#include "controller.h"

using namespace std;

GameController::GameController(int minRange, int maxRange, int maxAttempts)

: model(minRange, maxRange, maxAttempts) {}

void GameController::startGame() {

view.showWelcomeMessage();

do {

playRound();

} while (playAgain());

view.showGoodbyeMessage();

}

void GameController::playRound() {

while (!model.isGameOver()) {

view.showGuessPrompt();

int guess;

cin >> guess;

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Неверный Ввод. Пожалуйста введите правильное число." << endl;

continue;

}

if (model.guessNumber(guess)) {

view.showResult(true, model.getTargetNumber(), model.getAttempts());

return;

}

else {

view.showHigherLowerHint(guess < model.getTargetNumber());

}

}

view.showResult(false, model.getTargetNumber(), model.getAttempts());

}

bool GameController::playAgain() {

char choice;

do {

view.showTryAgainPrompt();

cin >> choice;

} while (choice != 'y' && choice != 'n');

return choice == 'y';

}

Файл: model.h

#pragma once

class NumberGame {

private:

int targetNumber;

int attempts;

int maxAttempts;

public:

NumberGame(int minRange, int maxRange, int maxAttempts);

int getTargetNumber() const;

int getAttempts() const;

bool isGameOver() const;

bool guessNumber(int guess);

};

Файл: view.h

#pragma once

class GameView {

public:

void showWelcomeMessage() const;

void showGuessPrompt() const;

void showResult(bool isCorrect, int targetNumber, int attempts) const;

void showTryAgainPrompt() const;

void showGoodbyeMessage() const;

void showHigherLowerHint(bool isHigher) const;

};

Файл: controller.h

#pragma once

#include "model.h"

#include "view.h"

class GameController {

private:

NumberGame model;

GameView view;

public:

GameController(int minRange, int maxRange, int maxAttempts);

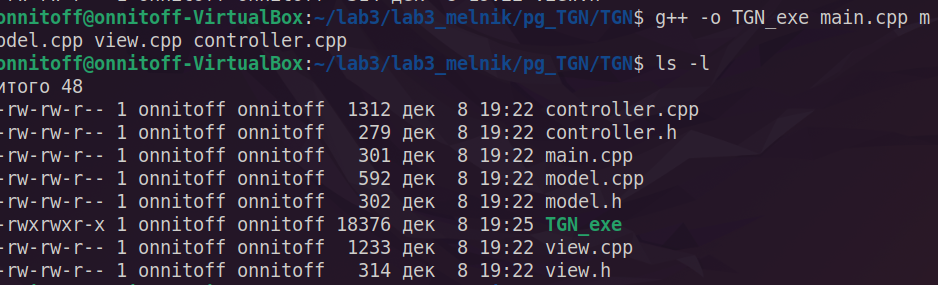
void startGame();

private:

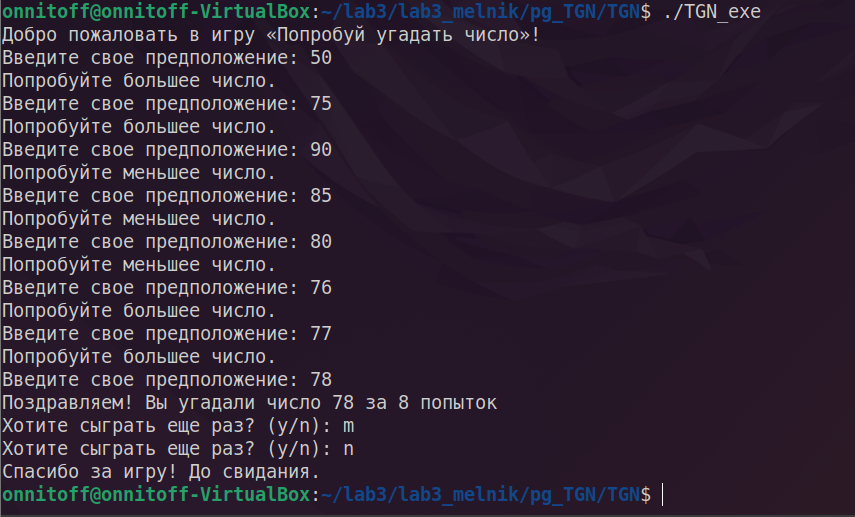
void playRound();

bool playAgain();

};



Работоспособность:



Задание 2.а:

Найти максимальное из чисел, встречающихся в заданной матрице более одного раза.

Листинг кода:

#include <iostream>

using namespace std;

const int MAX\_ROWS = 4;

const int MAX\_COLS = 4;

int findMaxRepeatedNumber(int matrix[MAX\_ROWS][MAX\_COLS]) {

const int maxNumber = 1000; // Предполагаем максимальное значение числа в матрице

// Используем массивы для подсчета встреч каждого числа в матрице

int countMap[maxNumber + 1] = { 0 };

// Подсчитываем количество встреч каждого числа в матрице

for (int i = 0; i < MAX\_ROWS; ++i) {

for (int j = 0; j < MAX\_COLS; ++j) {

countMap[matrix[i][j]]++;

}

}

int maxRepeatedNumber = -1;

int maxCount = 1;

// Находим максимальное число, встречающееся более одного раза

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

if (countMap[i] > 1 && countMap[i] > maxCount) {

maxCount = countMap[i];

maxRepeatedNumber = i;

}

}

return maxRepeatedNumber;

}

int main() {

setlocale(LC\_ALL, "Russian");

// Пример матрицы

int matrix[MAX\_ROWS][MAX\_COLS] = {

{4, 4, 3, 4},

{5, 6, 3, 1},

{6, 7, 6, 9},

{1, 2, 3, 6}

};

int maxRepeatedNumber = findMaxRepeatedNumber(matrix);

if (maxRepeatedNumber != -1) {

cout << "Максимальное число, встречающееся более одного раза: " << maxRepeatedNumber << endl;

}

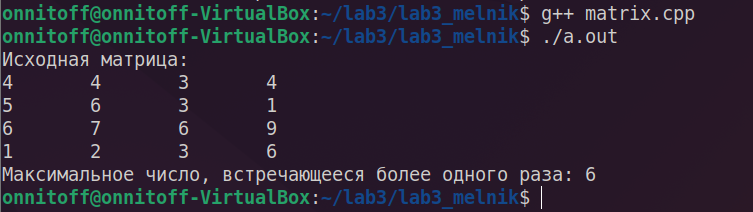
else {

cout << "В матрице нет чисел, встречающихся более одного раза." << endl;

}

return 0;

}



**Задание 3:**

Необходимо переработать основное задание из предыдущей лабораторной работы таким образом, чтобы пользователь (игрок) загадывал число, а компьютер, используя оптимальный и эффективный алгоритм, его отгадывал.

Листинг кода:

Файл: main.cpp

#include "controller.h"

int main() {

Controller controller;

controller.runGame();

return 0;

}

Файл: model.cpp

#include <iostream>

#include <limits>

#include "model.h"

using namespace std;

Model::Model(int minRange, int maxRange)

: attempts(0) {

cout << "Подумайте о числе между " << minRange << " и " << maxRange << "." << endl;

cout << "Нажмите Enter, когда будете готовы." << endl;

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Очистка буфера ввода

targetNumber = findOptimalGuess(minRange, maxRange);

}

int Model::getTargetNumber() const {

return targetNumber;

}

int Model::getAttempts() const {

return attempts;

}

bool Model::guessNumber(int guess) {

attempts++;

return guess == targetNumber;

}

int Model::findOptimalGuess(int low, int high) {

while (low <= high) {

int mid = low + (high - low) / 2;

cout << "Компьютер предполагает: " << mid << endl;

cout << "Это правильно? (y/n): ";

char response = NULL;

cin >> response;

if (response == 'y') {

attempts++;

return mid;

}

else if (response == 'n') {

cout << "Число больше или меньше, чем " << mid << "? (h/l): ";

cin >> response;

attempts++;

if (response == 'h') {

low = mid + 1;

}

else if (response == 'l') {

high = mid - 1;

}

else {

cout << "Неверный ответ. Пожалуйста, введите «h» или «l»." << endl;

}

}

else {

cout << "Неверный ответ. Пожалуйста, введите «y» или «n»." << endl;

}

}

return -1;

}

Файл: view.cpp

#include <iostream>

#include "view.h"

using namespace std;

void View::congratulate(int targetNumber, int attempts) const {

cout << "Поздравляем! Компьютер угадал ваш номер " << targetNumber

<< " за " << attempts << " попытки." << endl;

}

Файл: controller.cpp

#include <iostream>

#include "controller.h"

using namespace std;

void Controller::runGame() {

setlocale(LC\_ALL, "Russian");

const int minRange = 1;

const int maxRange = 100;

Model game(minRange, maxRange);

View view;

view.congratulate(game.getTargetNumber(), game.getAttempts());

}

Файл: model.h

#pragma once

class Model {

private:

int targetNumber;

int attempts;

public:

Model(int minRange, int maxRange);

int getTargetNumber() const;

int getAttempts() const;

bool guessNumber(int guess);

private:

int findOptimalGuess(int low, int high);

};

Файл: view.h

#pragma once

class View {

public:

void congratulate(int targetNumber, int attempts) const;

};

Файл: controller.h

#pragma once

#include "model.h"

#include "view.h"

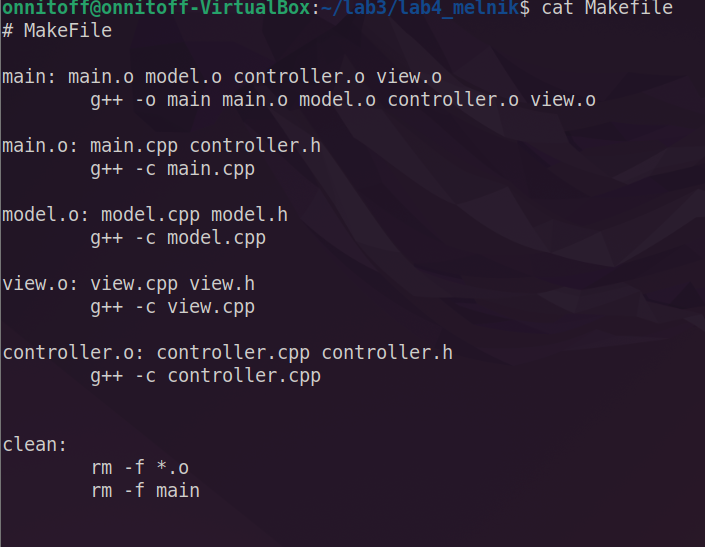
class Controller {

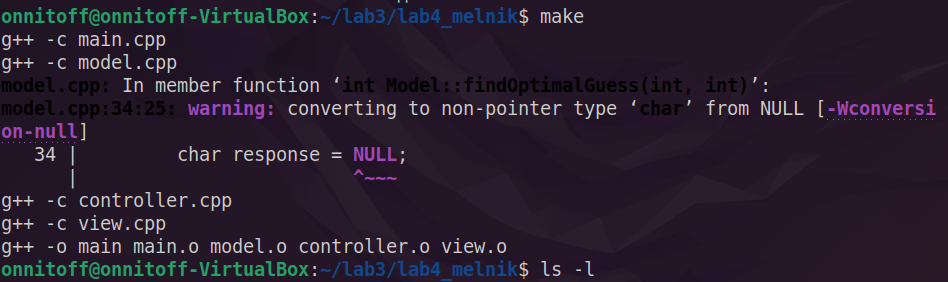
public:

void runGame();

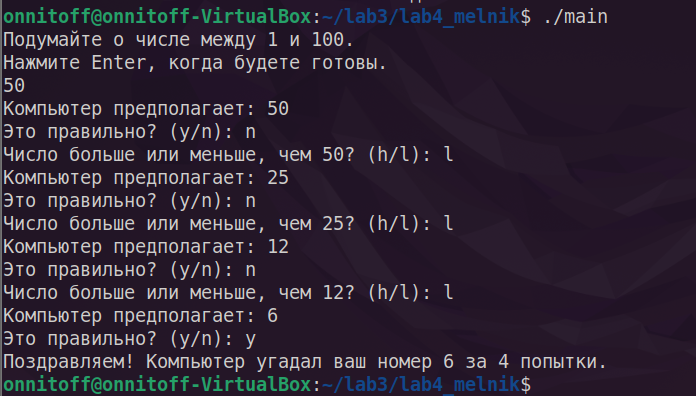
};

MakeFile:



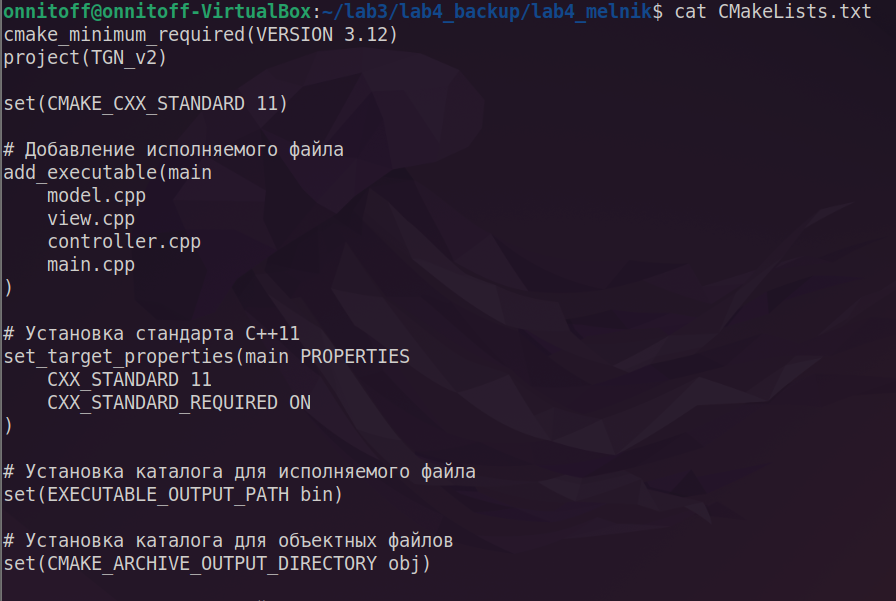
Результат выполнения команды make: 

Рабоспособность:

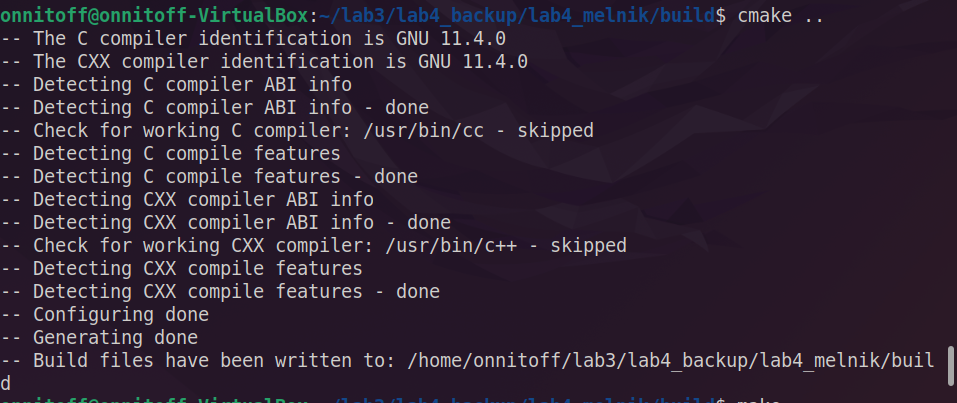


Использование cmake:

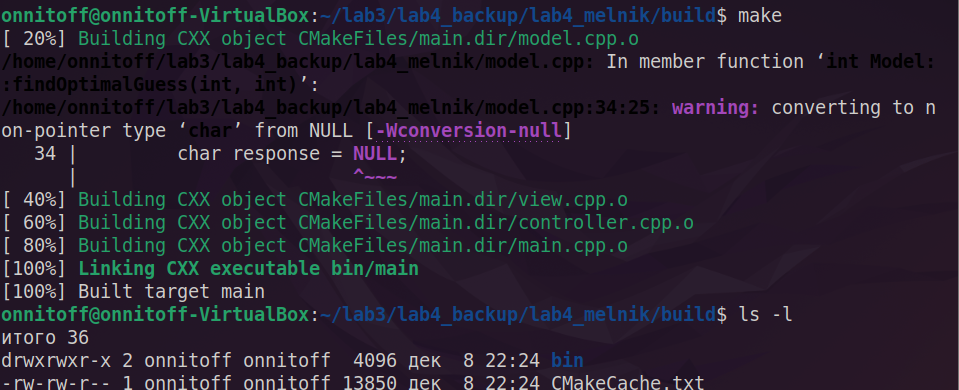
Текстовый файл CMakeList.txt:



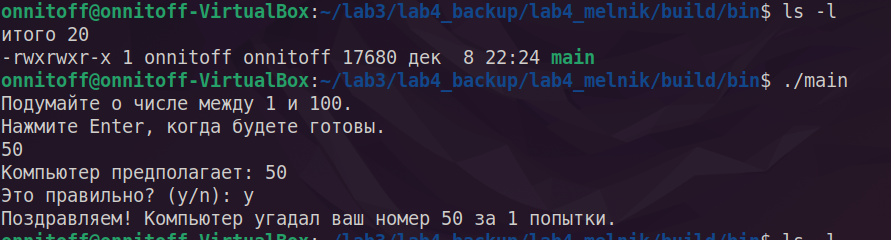
Настройка проекта с использованием cmake:



Сборка с помощью make:

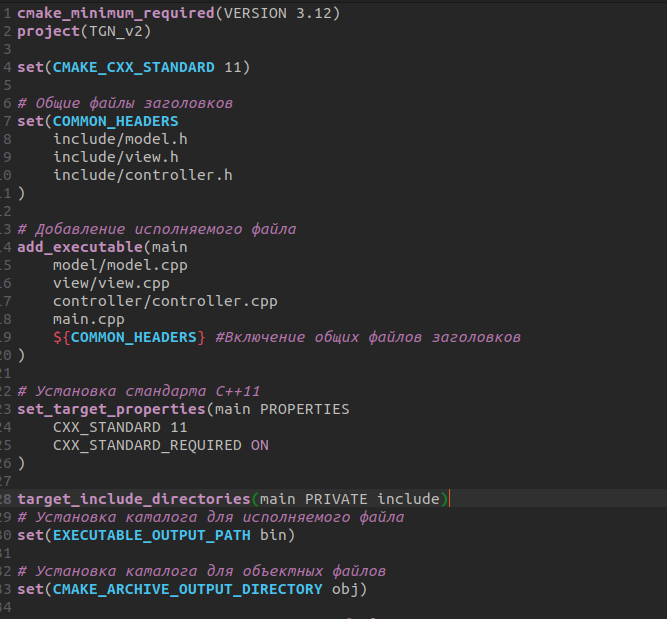


Исполняемый файл находится в папке bin согласено ранее указаному пути в файле CMakeList.txt:

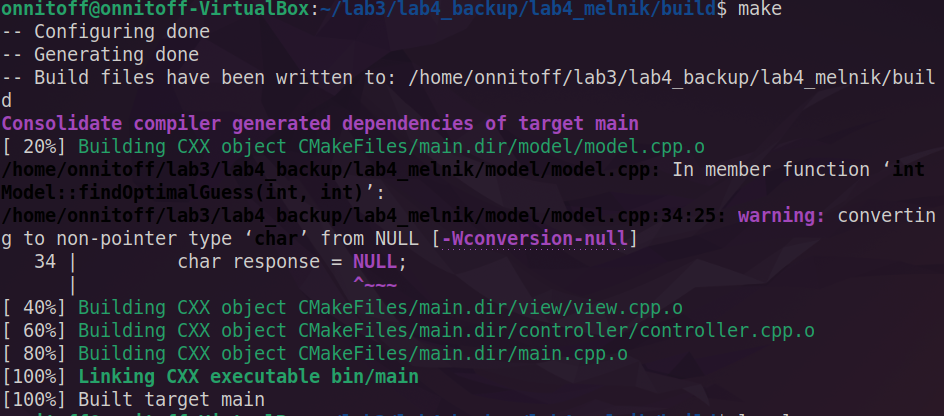


Рекурсивная сборка с использованием cmake:

Файл CMakeFile.txt:



Выполнение команды make:



Работоспособность:

