МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 9**

по дисциплине: “Системное программирование”

на тему: ***“***Автоматическая сборка многофайловых проектов ***”***

Вариант 1

Выполнил**:** студент группы 10702121 Меркулова М. С.

Черняк П. С.

Принял**:** Хавитько Е. А.

Минск 2023

# Лабораторная работа №2\_1. Компиляция и отладка простейшего приложения в Linux

**Цель работы:** Научиться эффективно использовать специальные средства для автоматиза- ции процесса компиляции, сборки и запуска многофайловых проектов.

**Основное задание**

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

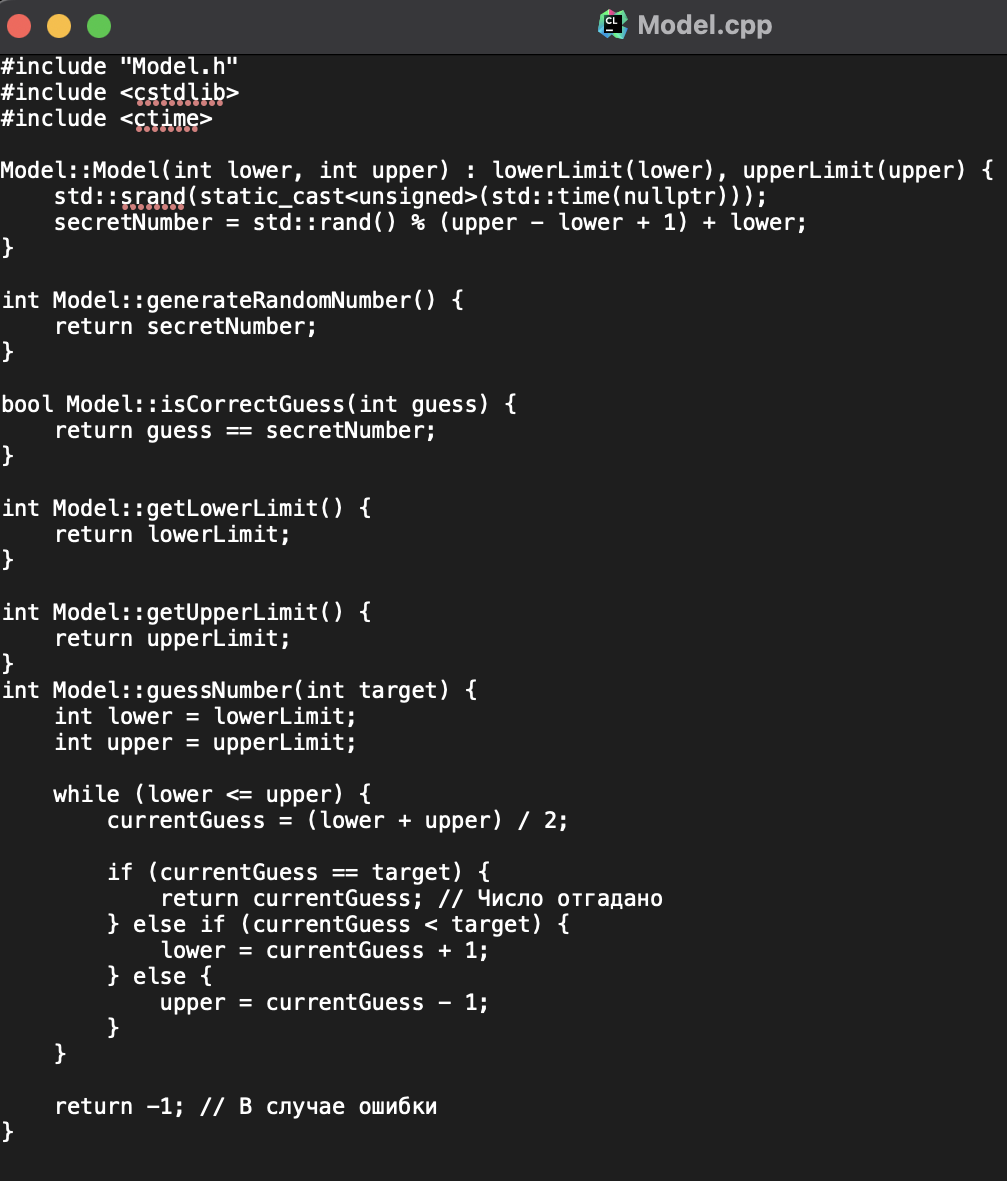
**Индивидуальное задание**

Для программ, которые были разработаны в двух предыдущих лабораторных работах No2 и No3 необходимо добавить автоматизацию сборки многофайло- вого проекта с использованием сценарных оболочек и автосборщиков.

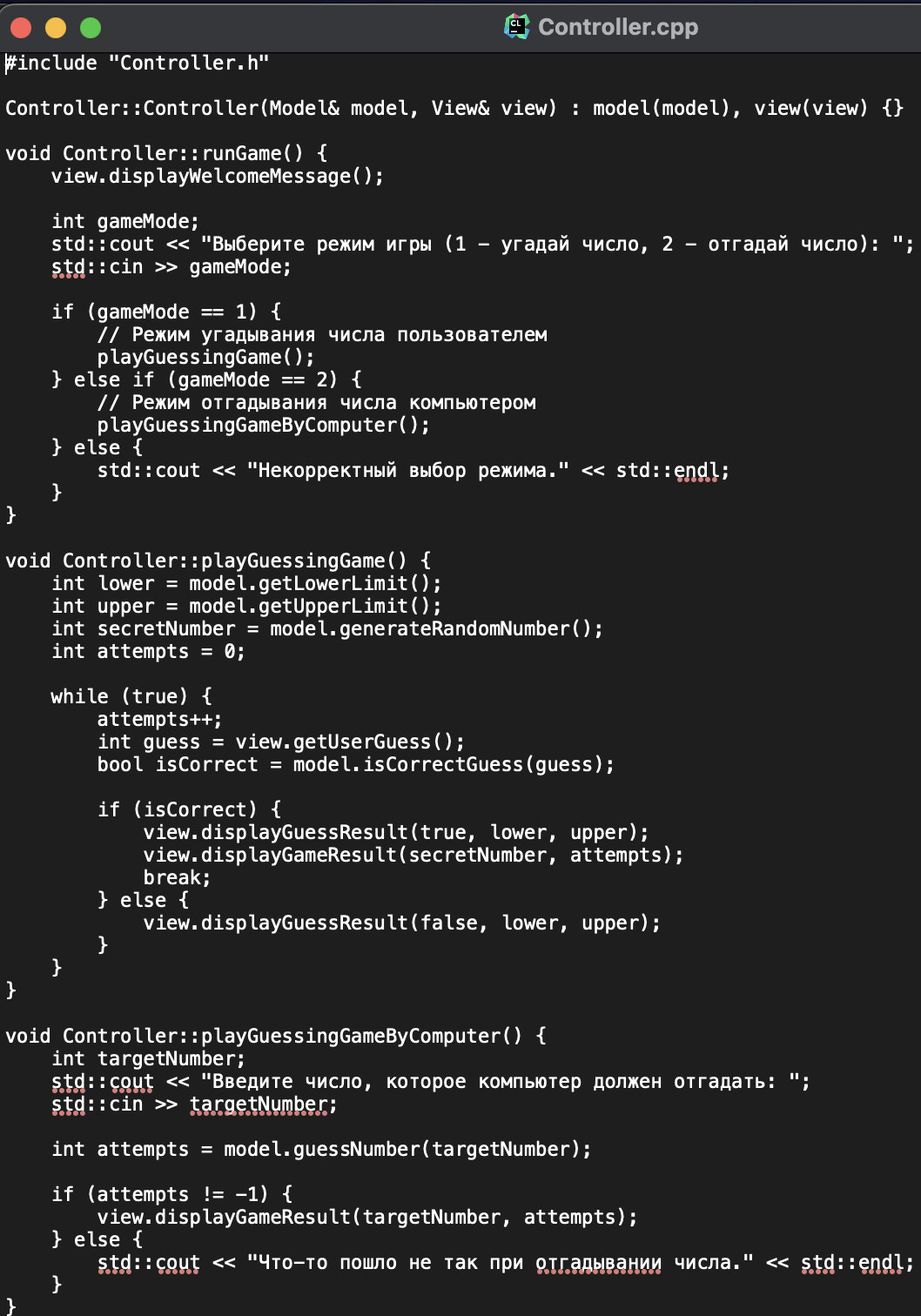
**Код программы**

Внесенные изменения в код лаб8:

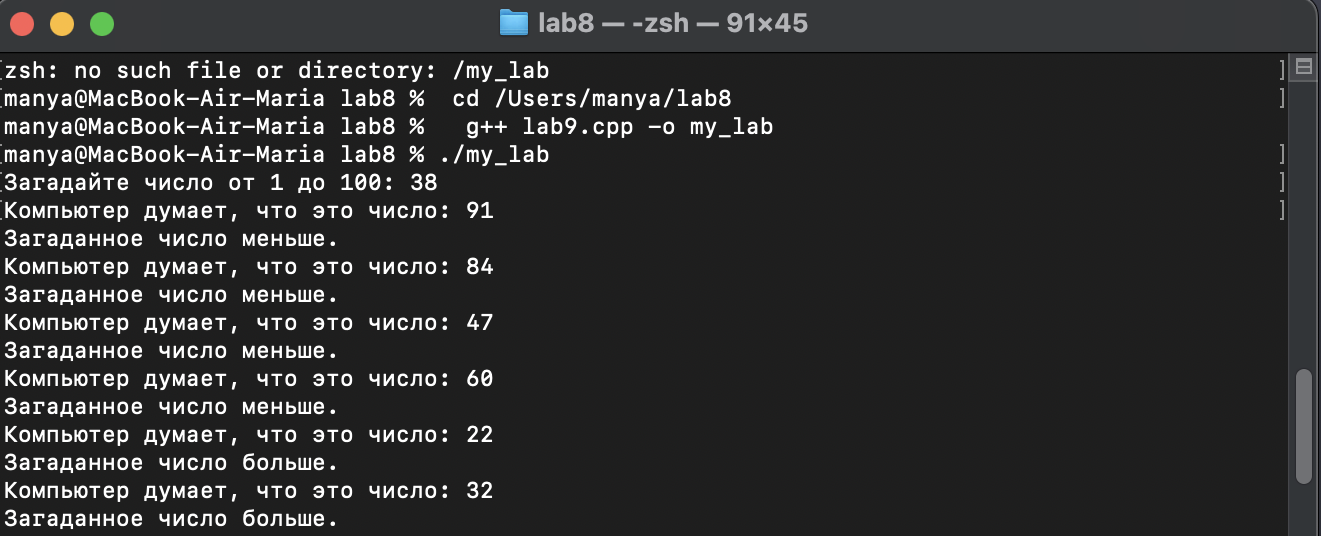
#include **<iostream>**#include **<cstdlib>**#include **<ctime>  
  
int** main() {  
 *// Инициализация генератора случайных чисел* std::srand(**static\_cast**<**unsigned**>(std::time(**nullptr**)));  
  
 **int** lower = 1; *// Нижний предел диапазона* **int** upper = 100; *// Верхний предел диапазона* **int** secretNumber;  
 std::cout << **"Загадайте число от "** << lower << **" до "** << upper << **": "**;  
 std::cin >> secretNumber;  
  
 **int** guess;  
 **int** attempts = 0;  
  
 **do** {  
 guess = std::rand() % (upper - lower + 1) + lower; *// Генерация случайной догадки* attempts++;  
  
 std::cout << **"Компьютер думает, что это число: "** << guess << std::endl;  
  
 **if** (guess < secretNumber) {  
 std::cout << **"Загаданное число больше."** << std::endl;  
 } **else if** (guess > secretNumber) {  
 std::cout << **"Загаданное число меньше."** << std::endl;  
 }  
 } **while** (guess != secretNumber);  
  
 std::cout << **"Компьютер угадал число "** << secretNumber << **" с "** << attempts << **" попыток."** << std::endl;  
  
 **return** 0;  
}

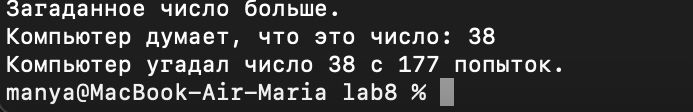


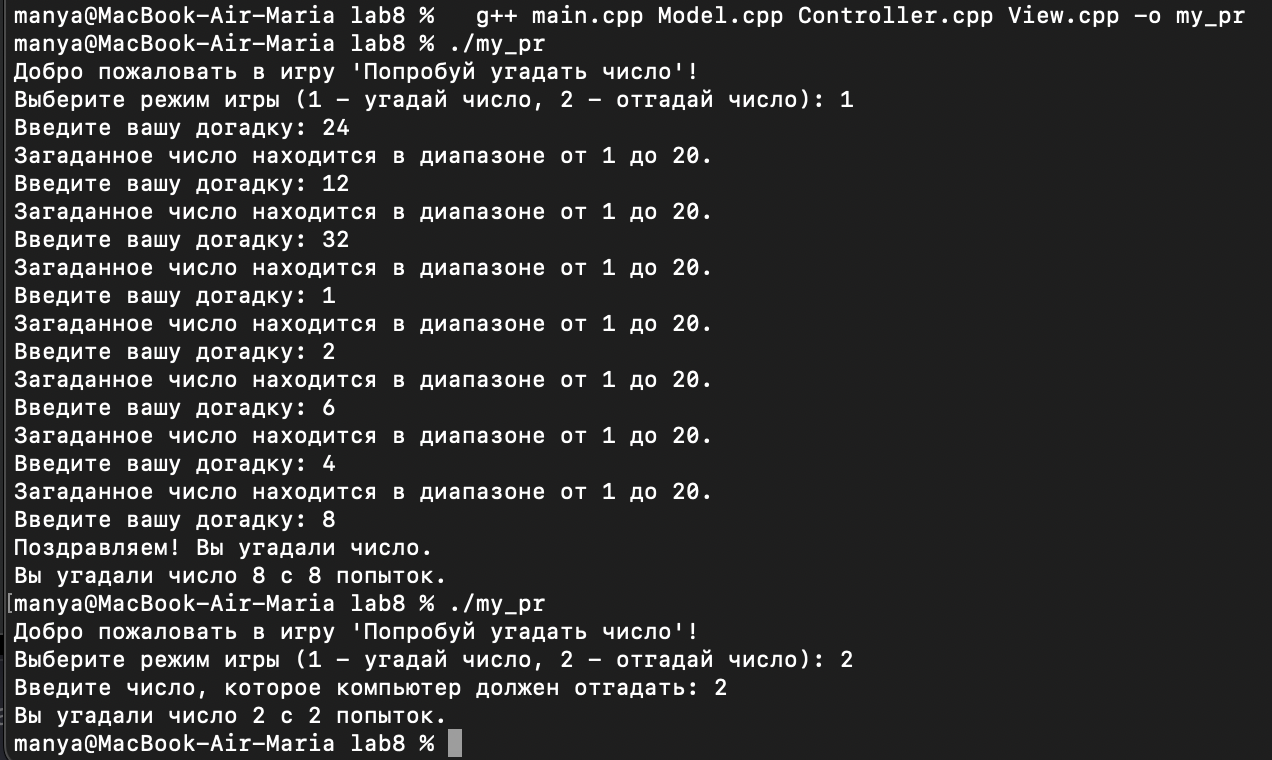


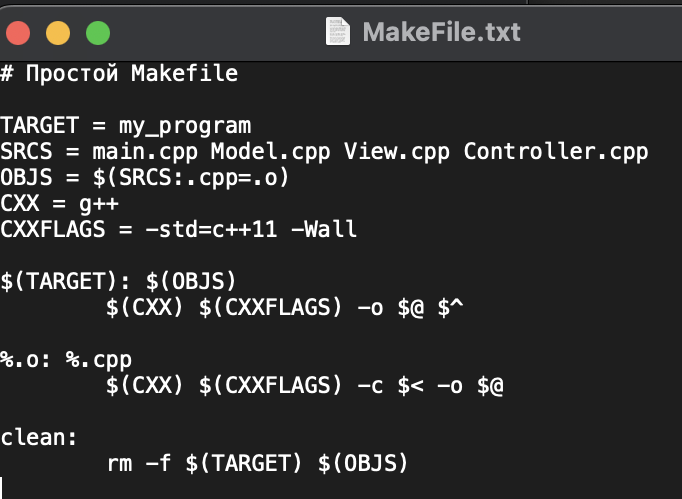


**Результат:**

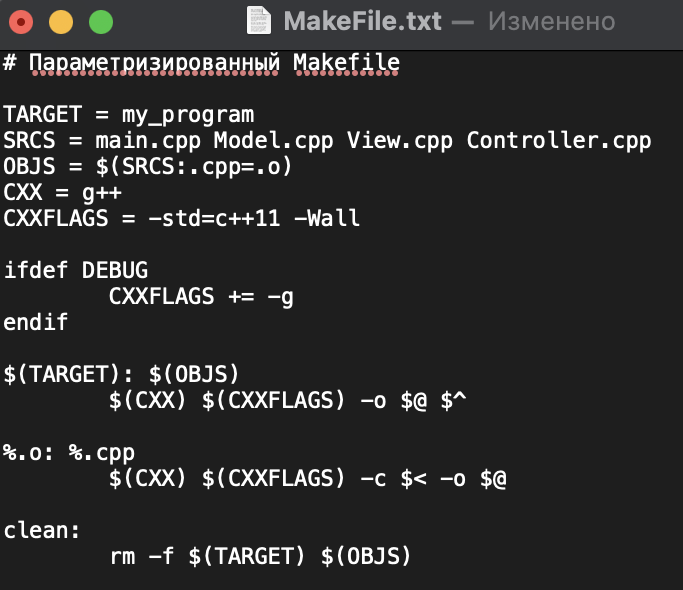
****

****

****

****

* **TARGET** - цель (исполняемый файл).
* **SRCS** - имена исходных файлов.
* **OBJS** - имена объектных файлов.
* **CXX** - компилятор (g++).
* **CXXFLAGS** - опции компиляции.



**Далее следует прописать следующие команды:**

make # для компиляции проекта

make DEBUG=1

make clean # для удаления скомпилированных файлов

**Контрольный вопрос:** Какие существуют способы и средства для автоматической сборки многофай- ловых проектов? Опишите категории утилит автосборок в Linux, а также их преимущества и недостатки?

В Linux существует несколько утилит и систем автоматической сборки для многофайловых проектов. Рассмотрим основные категории и их преимущества/недостатки:

**Makefile:**

• Преимущества:

• Прост в использовании и создании.

• Широко распространен и хорошо поддерживается.

• Недостатки:

• Возможно, требует больше строк кода для сложных проектов.

• Отслеживание зависимостей иногда может быть сложным.

**CMake:**

• Преимущества:

• Кроссплатформенность: может генерировать файлы сборки для различных систем (Makefiles, Visual Studio проекты и т.д.).

• Удобное описание зависимостей и конфигурации проекта.

• Встроенная поддержка для многих компиляторов и сред разработки.

• Недостатки:

• Изначально требует создания файла конфигурации (CMakeLists.txt), что может показаться избыточным для небольших проектов.

**Automake и Autoconf:**

• Преимущества:

• Сильная поддержка для проектов на языке C и C++.

• Система макросов позволяет легко настраивать сборку для различных систем.

• Недостатки:

• Некоторые считают, что это сложно для изучения и использования.

**Meson:**

• Преимущества:

• Прост в использовании и создании.

• Быстро сборает проекты.

• Удобное описание зависимостей.

• Недостатки:

• Меньшее распространение по сравнению с Make, CMake или Autotools.

**SCons:**

• Преимущества:

• Написан на Python, что может быть удобным для тех, кто знаком с этим языком.

• Простой синтаксис для описания сборки.

• Недостатки:

• Может быть медленнее в сравнении с другими инструментами.

Общие преимущества:

• Автоматизация: Утилиты автосборок позволяют автоматизировать процесс компиляции, упрощая сборку проектов.

• Поддержка зависимостей: Эти утилиты позволяют эффективно отслеживать зависимости между файлами, обеспечивая чистую и эффективную пересборку только необходимых частей проекта.

Общие недостатки:

• Изучение: Некоторые из этих инструментов могут потребовать времени для изучения, особенно для новичков.

• Конфигурация: Настройка может потребовать определенного времени, особенно при работе с более сложными проектами.

Выбор утилиты зависит от требований вашего проекта и вашего личного опыта. CMake широко используется в индустрии и предоставляет хороший баланс между гибкостью и удобством.

Начало формы