Сибирский государственный университет

телекоммуникаций и информатики

(СибГУТИ)

Курсовая работа

по дисциплине «Технологии разработки программного обеспечения»

на тему «Игра Жизнь»

Выполнил студент:

Группы ЗП-021

Коновалов Владимир Владимирович

Шифр:73200073

Проверил:

ст. преподаватель Токмашева Е. И.

Новосибирск 2022

**Содержание:**

Введение и постановка задачи…………………………………………….3

Техническое задание……………………………………………………….4

* Функционал приложения…………………………………………...4
* Формат входных данных……………………………………………4
* Интерфейс приложения…………………………………………..…4

Описание выполненного проекта…………………………………………5

* Общие задачи проекта………………………………………………5
* Подзадачи……………………………………………………………5
* Командная работа…………………………………………………...5
* Выполненные задачи………………………………………………..6
* Примеры работы приложения……………………………………...7
* Тестирование………………………………………………………...8

Личный вклад в проект…………………………………………………….9

Приложение. Код программы……………………………………………10

# Введение и постановка задачи

Цель работы: освоение курса по предмету «технологии разработки программного обеспечения», включающего в себя знакомство с системами контроля версий, командную разработку с помощью Git, интеграция Git в Visual Studio Code, отслеживание и обработку ошибок, отладку программного кода и модульное тестирование.

Задача: на основании изученных материалов и полученных навыков разработать приложение «игра Жизнь» используя современные средства командной разработки: система контроля версий Git, хранилище удаленных репозиториев GitHub, механизмы непрерывной интеграции, автоматизированная сборка при помощи утилиты make, модульное тестирование. В качестве инструментов разработки был выбран язык программирования C++ стандарта C++17, текстовый редактор Visual Studio Code.

Целевая платформа приложения – Linux. Для успешной сборки и запуска приложения должны быть установлены пакеты G++ версии не ниже 9.4.0, make, Git.

# Техническое задание

Функционал приложения:

Место действия игры — клеточное поле на ограниченной плоскости. Каждая клетка может находиться в двух состояниях «живая» или «мертвая» (пустой). У каждой клетки есть 8 окружающих ее клеток.

Начальное состояние или первое поколение — это расположение клеток на старте игры. Будем заполнять его из файла формата \*.txt. Каждое новое поколение рассчитывается на основе предыдущего по определенным правилам.

Правила игры или логика работы программы:

* В пустой клетке, возле которой расположены три живые, зарождается жизнь.
* Клетка продолжает жизнь, если рядом есть 2 или 3 «соседки». В противном случае клетка умирает.
* Если на поле не остается ни одной живой клетки, либо на очередном шаге ни одна клетка не изменит свое состояние, то игра прекращается.

Проект покрывает следующие сценарии использования:

* Генерация поколений на основе начальной конфигурации игрового поля;
* Сохранение текущей конфигурации игрового поля;
* Загрузка сохраненных конфигураций игрового поля.

Формат входных данных.

TXT – файл с конфигурацией игрового поля, последовательность нажатий клавиш управления игровым процессом.

Интерфейс приложения.

Игра запускается в консольном режиме, в соответствии с содержанием \*.txt файла, который выбран из уже существующих, либо задан самим пользователем. Файлы хранятся в корневом каталоге в папке «presets». В качестве элементов интерфейса будет использовано меню, состоящее из следующих пунктов: «новое поколение», «загрузить», «сохранить» и «выход».

Содержание конфигурационного файла.

Первая строка отвечает за количество строк и столбцов на поле, а также за два значения для символа «живой» и «мертвой» (пустой) клетки в формате Unicode, соответственно. На следующих строках задаются «живые» и «мертвые» (пустые) клетки в бинарном формате «1» и «0», где 0 – мёртвая, а 1 – живая клетка.

Запущенный текстовый файл определяет начальное состояние игры, т.е. первое поколение.

# Описание выполненного проекта

**Общие задачи проекта:**

* Создание командного репозитория на GitHub;
* Написание ТЗ;
* Распределение задач по проекту;
* Написание основной функциональности;
* Начальное покрытие тестами;
* Отладочные работы;
* Окончательное тестирование;
* Завершение работ по проекту;
* Составление отчета;
* Представление проекта.

**Подзадачи процесса разработки:**

1. Реализация считывания конфигурации из текстового файла и вывод игрового поля на экран;
2. Реализация механизма генерации новых поколений;
3. Разработка класса, осуществляющего запуск игры и предоставление интерфейса взаимодействия с пользователем посредством обработки нажатия клавиш управления;
4. Параллельно основному процессу разработки будут осуществлены

* Создание файла Makefile для использования утилиты make осуществляющей автоматическую сборку приложения;
* Модульные тесты;
* Создание коммитов для системы контроля версий;
* Разработка документации.

**Командная работа:**

Выявленные задачи были распределены между участниками команды. Каждым участником были успешно выполнены поставленные задачи. Каждый принимал активное участие в разработке логики проекта и покрытии его тестами. Так же каждым участником вносились небольшие правки как в свои, так и в функции других участников.

**Детали реализации:**

Приложение состоит из двух классов:

Класс Life является объектно-ориентированных представлением игрового поля и конфигурации игрового процесса в настоящий момент. Основные методы: Print – выводит на экран игровое поле. MakeStep – производит смену поколений. Save – сохранение текущей конфигурации игрового поля в текстовый файл.

Класс App представляет собой среду для запуска и контроля игрового процесса. Основные методы: MainLoop – вывод меню и обработка нажатий клавиш управления.

Классы Life и App связаны отношением агрегации для того, чтобы иметь возможность вывести на экран игровые поля разных размеров.

Точка входа в приложение – функция main, которая очищает экран и запускает основной цикл.

В реализации приложения использована стандартная библиотека C++. Для просмотра списка файлов с конфигурациями используется модуль «filesystem». Для считывания нажатий клавиш были использованы заголовочные файлы «usistd.h» «termios.h». Работа с выводом консоли реализована с помощью библиотеки «iostream».

В проекте использована обработка следующих исключительных ситуаций:

* Запуск несуществующего файла;
* Неверная структура входного файла.

**Примеры работы приложения**

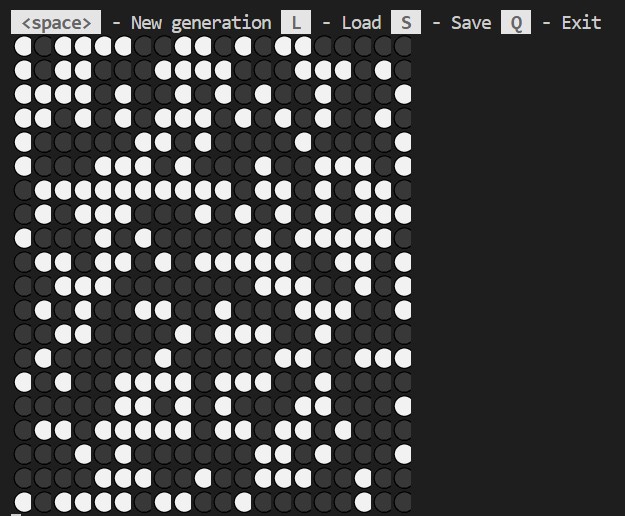


Рисунок 1 – Главное меню.

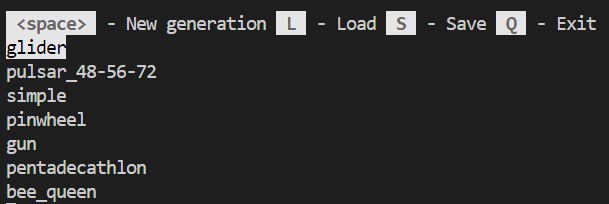


Рисунок 2 – Меню выбора конфигурации.

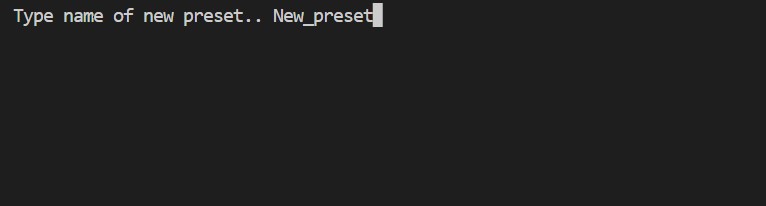


Рисунок 3 – Окно сохранения текущего пресета в файл

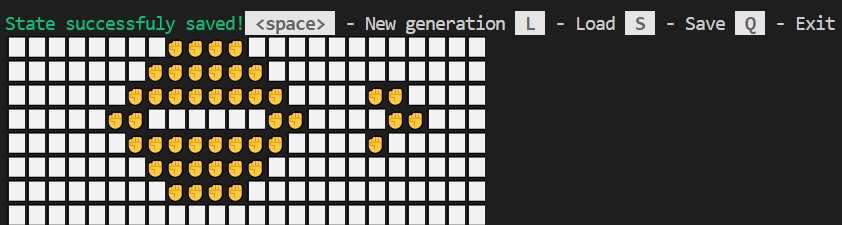


Рисунок 4 – Сообщение об успешном сохранении файла.

Тестирование

Для тестирования проекта используется библиотека [«**ctest.h**»](https://github.com/bvdberg/ctest) (github.com/bvdberg/ctest). Тестами покрыто следующее:

* Проверка на отсутствие файла (testNoFile);
* Проверка на наличие данных в файле (testEmptyfile);
* Проверка функции корректности вывода на экран (testPrint);
* Проверка функции «сделать шаг» (смена поколения) на выполнение (testMakeStep);
* Проверка алгоритма смены поколения граничных точках игрового поля (testMakeStepBorderPoints).

Сборка и запуск тестов осуществляется с помощью консоли командой «make runtests».

Результат тестирования:

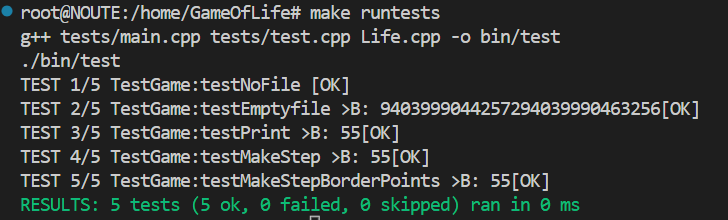


Рисунок 5 – Результаты тестирования.

# Личный вклад в проект

* ИМЯ:

Коновалов Владимир

* ИМЯ:

Кулеш Иван

# Приложение. Код программы