Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Системное программирование (СП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Игровое приложение «Сапер»

БГУИР КП 1-40 01 01  003  ПЗ

Студент: гр. 151003 Барановский Р.А.

Руководитель: Низовцов Д.В.

Минск 2023

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2023г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Барановскому Роману Алексеевичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Игровое приложение «Сапер»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи студентом законченной работы\_\_19.12.2023г.\_\_\_\_

3. Исходные данные к работе *Среда программирования Visual studio. Windows API для взаимодействия приложения с операционной системой Windows. Язык программирования c++. Наличие графической реализации интерфейса игры. Возможность игроку выбрать желаемую сложность.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по использованию программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованной литературы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе Низовцов Д.В.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.09.2023г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 25.09.2023г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.10.2023г. – 30% готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.11.2023г. – 60% готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Разделы 4, 5, Заключение к 15.12.2023 – 90 % готовности работы;

Оформление пояснительной записки и графического материала к 17.12.2023 – 100 % готовности работы.

Защита курсового проекта с 17.12.2023г. по 20.12.2023г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Низовцов Д.В.

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_Барановский Р.А. 16.09.2023г.*

*(дата и подпись студента)*

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc135084506)

[1 анализ литературных источников 7](#_Toc135084507)

[1.1 Анализ существующих аналогов 7](#_Toc135084508)

[1.1.1 «Hearthstone» 7](#_Toc135084509)

[1.1.2 «Гвинт»…… 8](#_Toc135084510)

[1.1.3 «Kards»…… 10](#_Toc135084511)

[1.2 Анализ методов и способов разработки 11](#_Toc135084512)

[1.2.1 Используемые библиотеки и технологии 11](#_Toc135084513)

[1.2.2 Используемые структуры данных 11](#_Toc135084514)

[2 Постановка задачи 12](#_Toc135084515)

[2.1 Назначение разработки 12](#_Toc135084516)

[2.2 Перечень функциональных требований 13](#_Toc135084517)

[2.3 Структура программы 14](#_Toc135084518)

[2.4 Состав и параметры технических и программных средств 15](#_Toc135084519)

[3 Разработка программного средства 16](#_Toc135084520)

[3.1 Описание алгоритмов решения задачи 16](#_Toc135084521)

[3.2 Структура типов 18](#_Toc135084522)

[3.3 Схема алгоритмов решения задач по ГОСТ 19.701-90 22](#_Toc135084523)

[3.3.1 Схема алгоритма GameController.startGame 22](#_Toc135084524)

[3.3.2 Схема алгоритма GameConroller.nextTurn 25](#_Toc135084525)

[3.3.3 Схема алгоритма Instance.setupServer 27](#_Toc135084526)

[3.3.4 Схема алгоритма Instance.setupClient 29](#_Toc135084527)

[3.3.5 Схема алгоритма Instance.run 31](#_Toc135084528)

[4 тестирование и проверка работоспособности программного средства 33](#_Toc135084529)

[4.1 Взаимодействие с главным меню 33](#_Toc135084530)

[4.1.1 Тест 1…….. 33](#_Toc135084531)

[4.1.2 Тест 2…….. 33](#_Toc135084532)

[4.1.3 Тест 3…….. 34](#_Toc135084533)

[4.2 Взаимодействие с мастером колод 35](#_Toc135084534)

[4.2.1 Тест 4…….. 35](#_Toc135084535)

[4.2.2 Тест 5…….. 35](#_Toc135084536)

[4.2.3 Тест 6…….. 36](#_Toc135084537)

[4.2.4 Тест 7…….. 37](#_Toc135084538)

[4.2.5 Тест 8…….. 38](#_Toc135084539)

[4.2.6 Тест 9…….. 38](#_Toc135084540)

[4.3 Взаимодействие с меню мультиплеера 39](#_Toc135084541)

[4.3.1 Тест 10…… 39](#_Toc135084542)

[4.3.2 Тест 11…… 39](#_Toc135084543)

[4.3.3 Тест 12…… 40](#_Toc135084544)

[4.4 Игровой процесс 41](#_Toc135084545)

[4.4.1 Тест 13…… 41](#_Toc135084546)

[4.4.2 Тест 14…… 41](#_Toc135084547)

[4.4.3 Тест 15…… 42](#_Toc135084548)

[4.4.4 Тест 16…… 43](#_Toc135084549)

[4.4.5 Тест 17…… 44](#_Toc135084550)

[5 Руководство по использованию программного средства 45](#_Toc135084551)

[5.1 Мастер колод 45](#_Toc135084552)

[5.2 Установление соединения 45](#_Toc135084553)

[5.3 Игра……. 45](#_Toc135084554)

[5.4 Завершение игры 45](#_Toc135084555)

[Заключение 46](#_Toc135084556)

[Список использованной литературы 47](#_Toc135084557)

[Приложение А 48](#_Toc135084558)

Введение

Данная работа посвящена созданию компьютерной игры «Сапер». Данная игра была создана в компании «Microsoft» Робертом Доннером и Куртом Джонсоном в 1992 году для операционной системы «Windows» с целью помочь пользователям отточить навыки использования компьютерной мыши. Поиск заминированных клеток призван был научить новичков свободно владеть как правой, так и левой кнопками мыши, и показать им, как соотносятся скорость и точность движения мышки. Принцип игры состоит в следующем: плоское или объёмное игровое поле разделено на смежные ячейки (квадраты, шестиугольники, кубы и т. п.), некоторые из которых «заминированы»; количество «заминированных» ячеек известно. Целью игры является открытие всех ячеек, не содержащих мины. Игрок открывает ячейки, стараясь не открыть ячейку с миной. Открыв ячейку с миной, он проигрывает. Для того, чтобы игрок открывал ячейки не «вслепую», используется следующий принцип: если под открытой ячейкой мины нет, то в ней появляется число, показывающее, сколько ячеек, соседствующих с только что открытой, «заминировано» (в каждом варианте игры соседство определяется по-своему); используя эти числа, игрок пытается рассчитать расположение мин, однако иногда даже в середине и в конце игры некоторые ячейки всё же приходится открывать наугад. Если под соседними ячейками тоже нет мин, то открывается некоторая «не заминированная» область до ячеек, в которых есть цифры. «Заминированные» ячейки игрок может пометить, чтобы случайно не открыть их. «Сапер» завоевал огромное популярность, существует множество различных версий игры и даже проводятся соревнования по скорости «разминирования». Особо примечательным рекордом является достижение Атто Юкисиро (Япония), который в 2016 году прошел поле «Сапера» размером 64х48 на ультрасложном уровне с 777 бомбами, на что у него ушло 12,5 часов, попытки пройти это поле он предпринимал в течение 10 лет.

# анализ литературных источников

## Анализ существующих аналогов

Современные разновидности «Сапера» отличаются огромным разнообразием. Они кардинально отличаются как графически, так и реализицией. Однако стандартные правила остаются везде практически без изменений.

### «Сапер» от google

«Сапер» от google – популярная версия игры «сапер» от компании google. Сыграть можно при наличии браузера и доступа к интернету. Игра обладает стандартными правилами, ведет статистику о времени в игре и количестве найденных мин, предоставляет возможность выбрать уровень сложности.

Достоинства игры:

* отличное графическое, музыкальное сопровождение;

В свою очередь к недостаткам можно отнести:

* доступность игры лишь в виде веб-приложения, что не позволяет установить ее на устройство и играть без доступа к интернету;
* отсутствие возможности выбрать желаемый размер поля.



Рисунок 1.1 – «Сапер» от google



Рисунок 1.2 – «Сапер» от google

### «Minesweeper» от Evkar games

«Minesweeper» от Evkar games – игра на мобильные устройства, позволяющая проникнуться духом классического «сапера».

Достоинства игры:

* отличное графическое сопровождение;
* наличие огромного количества различных настроек.

В свою очередь к недостаткам можно отнести:

* присутствие багов;
* большое количество рекламы в бесплатной версии.

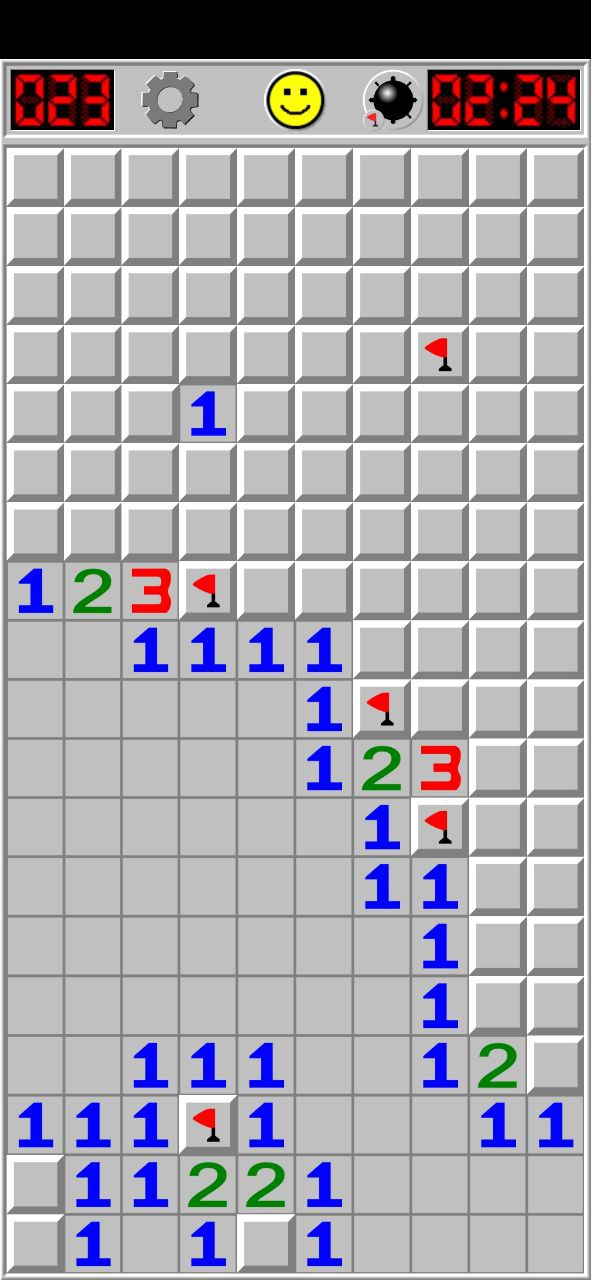


Рисунок 1.3 – «Minesweeper» от Evkar games



Рисунок 1.4 – «Minesweeper» от Evkar games

### «Microsoft Minesweeper»

«Microsoft Minesweeper» – «сапер» от компании Microsoft, разработчики которой придумали правила данной игры и впервые разработали ее.

Достоинства игры:

* присутствие различных разнообразных режимов игры;
* присутствие ежедневных заданий;
* отличное графическое, музыкальное сопровождение.

В свою очередь к недостаткам можно отнести:

* возрастное ограничение 16+ с пометкой «насилие».
* большое количество рекламы в бесплатной версии



Рисунок 1.5 – «Microsoft Minesweeper»



Рисунок 1.6 – «Microsoft Minesweeper»

## Анализ методов и способов разработки

### Используемые библиотеки и технологии

Предполагается, что разрабатываемая игра будет обладать графикой с возможностью взаимодействия с пользователем. Для этих целей будет использован Windows API.

Windows API – общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации Microsoft. Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой Windows.

### Используемые структуры данных

В данном проекте в качестве основной структуры данных будет использован vector.

Структура vector – структура данных в языке c++, предназначенная для хранения множества значений. Данная структура данных основана на массиве и является его усовершенствованной версией, так как динамична. Использование данной структуры облегчает работу за счет удаления необходимости вручную изменять размер массива, удалять элементы из него, вставлять элементы в массив. Структура vector представлена на рисунке 1.7.

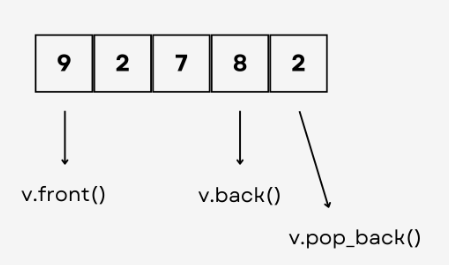


Рисунок 1.7 – vector

# Постановка задачи

## Назначение разработки

Назначением проектирования является разработка игры «Сапер». На основании произведенного обзора существующих аналогов, выявленных преимуществ и недостатков данных игр, сделан вывод, что для решения поставленной цели необходимо выполнить следующие задачи:

* проектирование архитектуры игры;
* проектирование графического сопровождения;
* разработка алгоритмов загрузки изображений из внешних файлов;
* разработка алгоритмов взаимодействия игрока с полем;
* разработка алгоритмов взаимодействия объектов между собой;
* разработка алгоритмов отображения сопровождающей информации;
* разработка меню для настроек игры;
* тестирование приложения.

## Перечень функциональных требований

Целью разработки игры «Сапер» является объединение основных достоинств рассмотренных существующих аналогов, а также компенсация недостатков этих игр. В результате разработки необходимо предоставить реализацию следующих функций:

* загрузка изображений из внешних файлов;
* настройка пользователем сложности и размера поля;
* взаимодействие между игроком и полем;
* отображение количества времени текущей игры;
* отображение количества оставшихся мин;
* просмотр информации о программе;
* завершение игры при попадании на мину.

## Структура программы

Основные модули программы:

* Program.cpp (Program.h) – модуль, обеспечивающий запуск игры и обработку сообщений;
* Type.cpp (Type.h) – модуль, содержащий информацию о возможных типах ячеек в игре(закрытая ячейка, бомба, флажок и т.д.);
* Cell.cpp (Cell.h) – модуль, обеспечивающий логику взаимодействия с ячейкой;
* Field.cpp (Field.h) – модуль, обеспечивающий логику взаимодействия с игровым полем;
* DrawableField.cpp (DrawableField.h) – модуль, обеспечивающий логику отрисовки игрового поля на экране пользователя.

## Состав и параметры технических и программных средств

Игра «Сапер» должна функционировать на персональных компьютерах со следующими характеристиками:

* Операционная система: Windows 10;
* RAM: 2 GB;
* Пространство на диске: 1 GB;
* Процессор: минимальное требование - Pentium 2 266 МГц;
* Монитор;
* Мышь;
* Клавиатура.

В данном разделе указаны минимальные технические требования для запуска игры. Для эксплуатации в реальных условиях могут потребоваться более мощные технические средства. Разработанная игра должна корректно функционировать на более мощном оборудовании.

# Разработка программного средства

## Описание алгоритмов решения задачи

Таблица 1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагае-мый тип реализации |
| 1 | Program::  wWinMain(  \_In\_ HINSTANCE hInstance, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPWSTR lpCmdLine, \_In\_ int nCmdShow) | Служит отправной точкой выполнения программы.  Создает главное окно программы. Запускает цикл обработки сообщений | hInstance – дескриптор для текущего экземпляра приложения.  hPrevInstance – дескриптор предыдущего экземпляра приложения.  lpCmdLine – командная строка приложения, за исключением имени программы.  nCmdShow – управляет тем, как должно отображаться окно | Функция.  Возвраща-емое значение –  выходное значение из параметра сообщения wParam или 0 в случае заверше-  ния до входа в цикл обработки сообщений |
| 2 | Program::  WndProc(  HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) | Обрабатывает сообщения, отправляемые в главное окно приложения | hWnd – дескриптор окна  message – сообщение  wParam – дополнитель- ные сведения о сообщении  lParam – дополнитель- | Функция.  Возвраща-емое значение – результат обработки сообщения |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | ные сведения о сообщении |  |
| 3 | Field::  reveal(int x,  int y) | Открывает одну или несколько ячеек на игровом поле | x – координата ячейки на оси абсцисс.  y – координата ячейки на оси ординат | Процедура |
| 4 | Field::  revealAll() | Открывает все ячейки на игровом поле | Формальные параметры отсутствуют | Процедура |
| 5 | Cell::  changeFlag() | Меняет состояние флага на ячейке | Формальные параметры отсутствуют | Функция.  Возвраща-емое значение – тип клетки в результате смены состояния флага |

## Структура типов

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Type | class Type  {  private:  static std::map<Types, std::string> typeImagePath;  Types m\_type;  std::string m\_imagePath;  public:  enum class Types;  Type(Types type);  operator std::string() const;  operator Types() const;  bool operator == (const Type& type);  bool operator == (const Types& type);  bool operator != (const Type& type);  bool operator != (const Types& type);  Type& operator = (const Types& type);  static std::map<Types, std::string> getTypesAndPaths();  }; | Класс, содержащий информацию о типе ячейки;  Поля данных:  typeImagePath – ассоциативный массив с ключами-типами и значениями-логическими путями типов;  m\_type – тип ячейки;  m\_imagePath – логический путь к изображению типа;  Types – перечисление, содержащее все возможные типы ячейки;  Методы класса:  Type(Types type) – конструктор класса;  operator std::string() – оператор приведения к строке(логическому пути к изображению типа);  operator Types() – оператор приведения к типу-перечислению;  operator == (const Type& type) – оператор сравнения с типом;  operator == (const Types& type) – оператор сравнения с типом-перечислением; |

Продолжение таблицы 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
|  |  | operator != (const Type& type) – оператор сравнения с типом;  operator != (const Types& type) – оператор сравнения с типом-перечислением;  operator = (const Types& type) – оператор присваивания с типом-перечислением;  getTypesAndPaths() – возвращает typeImagePath; |
| Cell | class Cell  {  private:  Type m\_inner;  Type m\_cover;  bool m\_isOpened;  public:  Cell();  Type getInner();  void setInner(Type inner);  Type getCover();  void setCover(Type cover);  bool isOpened();  Type getType();  Type changeFlag();  bool open();  }; | Класс, представляющий собой ячейку;  Поля данных:  m\_inner – внутренняя часть ячейки;  m\_cover – внешняя часть ячейки;  m\_isOpened – true, если ячейка открыта, false в противном случае;  Методы класса:  Cell() – конструктор класса;  getInner() – возвращает m\_inner;  setInner(Type inner) – устанавливает m\_inner;  getCover() – возвращает m\_cover;  setCover(Type cover) – устанавливает m\_cover;  isOpened() – возвращает m\_isOpened;  getType() – возвращает тип ячейки |

Продолжение таблицы 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
|  |  | в зависимости от ее состояния.  changeFlag() – устанавливает или убирает флаг на ячейке  open() – открывает ячейку |
| Field | class Field  {  private:  struct Coordinates;  static std::random\_device rd;  static std::mt19937 mersenne;  std::vector<std::vector  <Cell>> m\_field;  bool m\_isDeminingStarted;  bool m\_isGameOver;  int m\_numMines;  int m\_cellsRevealed;  int m\_width;  int m\_height;  bool outBounds(int x, int y);  public:  Field(int width, int height, int numMines);  bool isGameEnded();  bool isDemined();  bool isDeminingStarted();  int generateInt(int min, int max);  int calcBombsNear(int x, int y);  void countBombsNearEachCell();  void moveAwayBomb(int x, int y); | Класс, представляющий собой игровое поле;  Поля данных:  Coordinates – структура с координатами x, y;  rd – объект для инициализации mersenne;  mersenne – объект для генерации случайных целочисленных значений;  m\_field – вектор ячеек, представляющий поле;  m\_isDeminingStarted – true, если началось разминирование, false в противном случае;  m\_isGameOver – true, если игра завершилась, иначе false;  m\_numMines – количество мин;  m\_cellsRevealed – количество открытых ячеек;  m\_width – ширина поля;  m\_height – высота поля;  Методы класса: |

Продолжение таблицы 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
|  | void reveal(int x, int y);  void revealAll();  void generate(int width,  int height, int numMines);  std::vector<Cell>& operator[](const int index);  int getHeight() const;  int getWidth() const;  int getNumMines() const;  }; | outBounds(int x, int y) – возвращает true, если x или y за границами поля, false в противном случае;  Field(int width, int height, int numMines) – конструктор класса;  isGameEnded() – возвращает true, если игра завершилась, false в противном случае;  isDemined() – возвращает true, если всё поле разминировано(игрок победил), false в противном случае;  isDeminingStarted() – возвращает true, если разминирование началось(открыта хотя бы 1 клетка), false в противном случае;  generateInt(int min, int max) – генерирует случайное целое число от min до max включительно;  calcBombsNear(int x,  int y) – подсчитывает количество бомб рядом с ячейкой с координатами x, y;  countBombsNearEach-  Cell() – подсчитвает количество бомб рядом с каждой свободной клеткой.  moveAwayBomb(int x, int y) – убирает бомбу с ячейки с координатами x, y и переносит в |

Продолжение таблицы 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
|  |  | случайную свободную ячейку на поле;  reveal(int x, int y) – раскрывает ячейки, начиная с ячейки с координатами x, y в соответствии с правилами;  revealAll() – раскрывает все ячейки на поле;  generate(int width,  int height, int numMines)  – генерирует случайное поле в соответствии с переданными размерами и количеством мин;  operator[](const int index) – оператор индексации элементов массива для получения ячеек поля.  getHeight() – получение высоты поля;  getWidth() – получение ширины поля;  getNumMines() – получение количества мин на поле; |
| DrawableField | class DrawableField  {  private:  HWND& m\_hWnd;  HINSTANCE& m\_hInst;  Field& m\_field;  std::vector<HWND> m\_buttons; | Класс, служащий для отрисовки поля:  Поля данных:  m\_hWnd – дескриптор окна, в котором происходит отрисовка;  m\_hInst – сущность окна, в котором происходит отрисовка;  m\_field – ссылка на поле для отрисовки; |

Продолжение таблицы 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
|  | std::vector<HBITMAP> m\_images;  public:  DrawableField(Field& field, HWND& hWnd, HINSTANCE& hInst);  void update(bool isRecreateButtons, SUBCLASSPROC fieldActions);  }; | m\_buttons – кнопки, представляющие ячейки поля;  m\_images – изображения на кнопках(необходимо для избежания обновления изображений на кнопках, которые не изменились);    Методы класса:  DrawableField(Field& field, HWND& hWnd, HINSTANCE& hInst) – конструктор класса;  update(bool isRecreateButtons, SUBCLASSPROC fieldActions) – перерисовывает поле с пересозданием кнопок или без; |

## Схема алгоритмов решения задач по ГОСТ 19.701-90

### Схема алгоритма GameController.startGame

Схема алгоритма запуска игры. Порядок тасования колод важен, так как после синхронизации рандома, имея randSeed, сервер и клиент будут по отдельности генерировать одинаковые псевдорандомные значения. И для того, чтобы клиент мог знать о порядке карт в колоде сервера, а сервер о порядке карт в колоде клиента, необходимо, чтобы псевдорандомные значения, определяющие то, как будут перетасованы колоды, совпадали. Это необходимо, так как при взаимодействии сервера и клиента передаются индексы карт, миньонов и т.д., и подразумевается, что сторона-получатель сообщения знает, какая карта, миньон и т.д. находится под соответствующим индексом.



Рисунок 3.1 – Схема алгоритма startGame (часть 1)



Рисунок 3.1 – Схема алгоритма startGame (часть 2)

### Схема алгоритма GameConroller.nextTurn

Cхема алгоритма передачи хода.

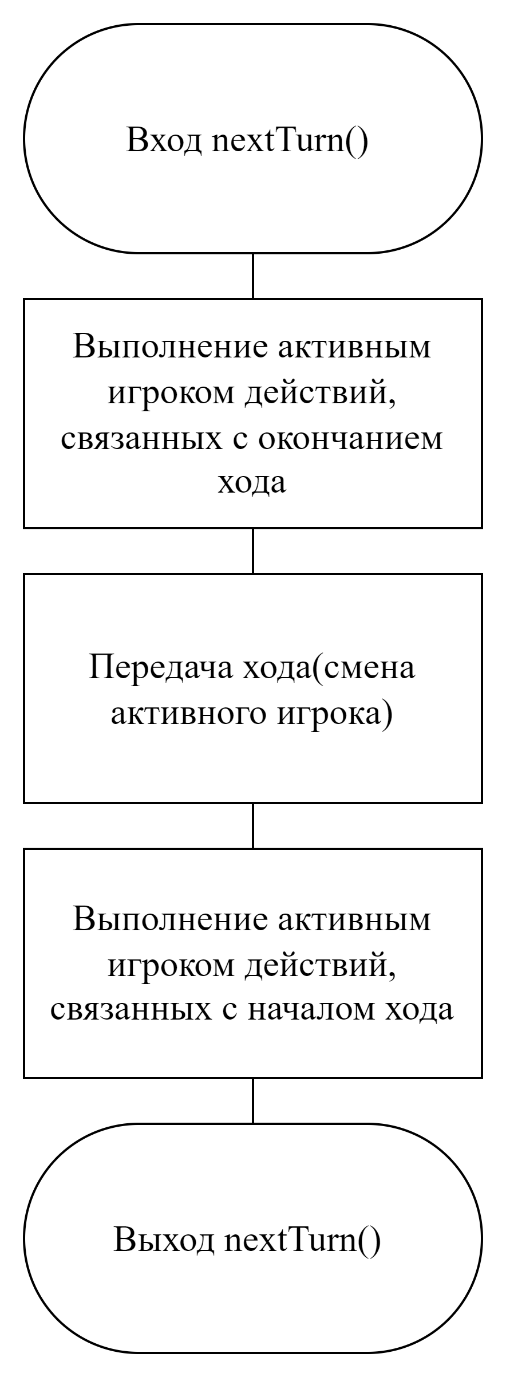


Рисунок 3.2 – Схема алгоритма nextTurn

### Схема алгоритма Instance.setupServer

Схема алгоритма настройки сервера перед началом сетевого взаимодействия.

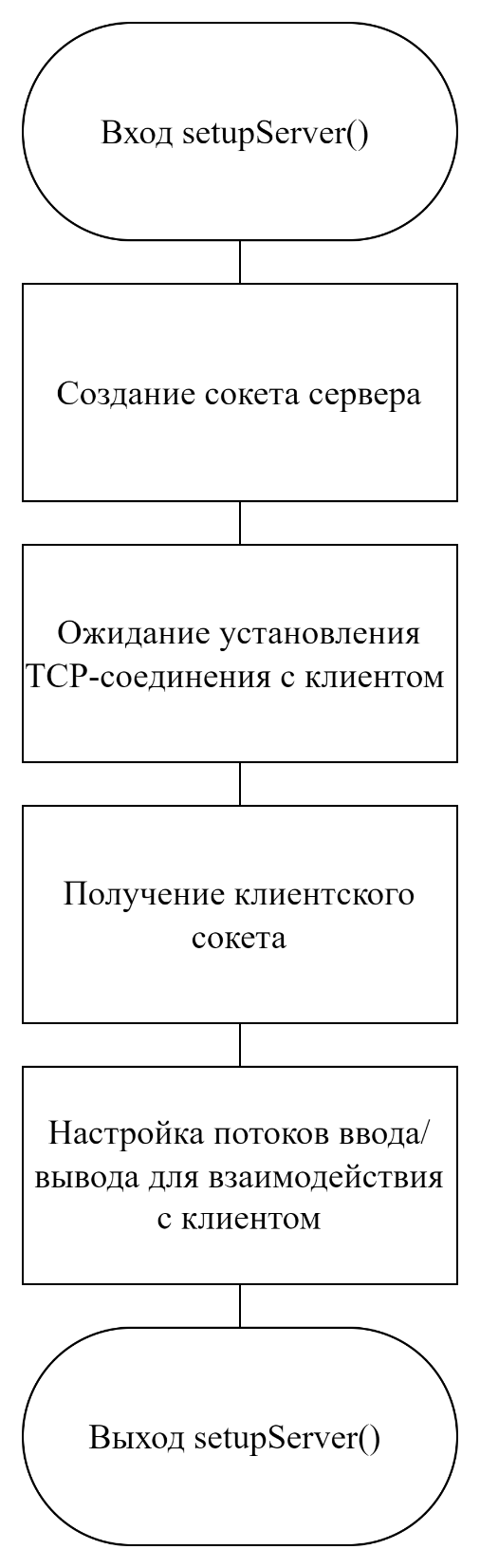


Рисунок 3.3 – Схема алгоритма setupServer

### Схема алгоритма Instance.setupClient

Схема алгоритма настройки клиента перед началом сетевого взаимодействия.

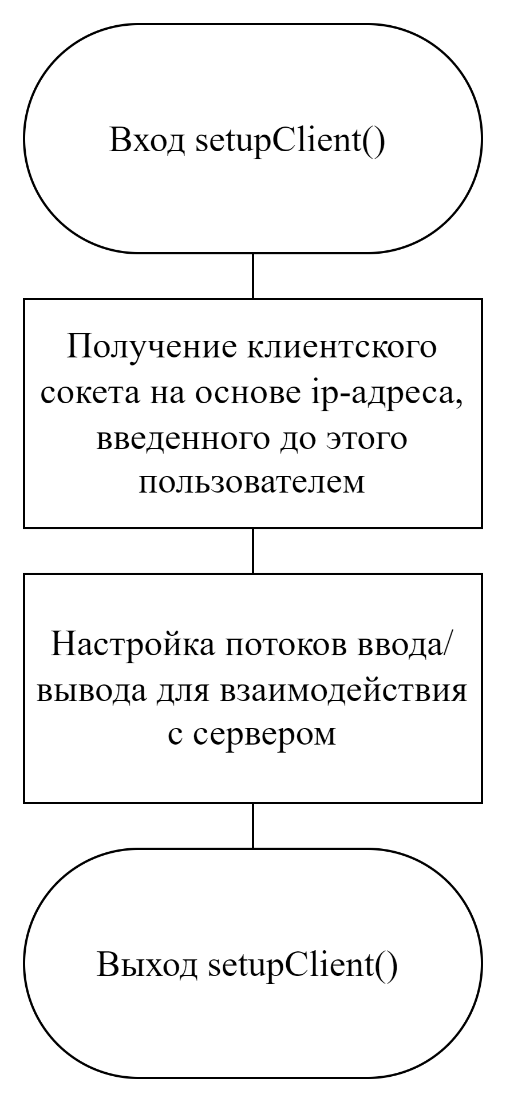


Рисунок 3.4 – Схема алгоритма setupClient

### Схема алгоритма Instance.run

Схема алгоритма работы потока сервера или клиента. Следует обратить внимание, что работа сервера и клиента по большому счету идентична, за исключением стадии подготовки к игре. Также стоит отметить, что функция отправки сообщения реализована также в классе Instance и вызывается в других методах, которые и формируют отправляемое сообщение.

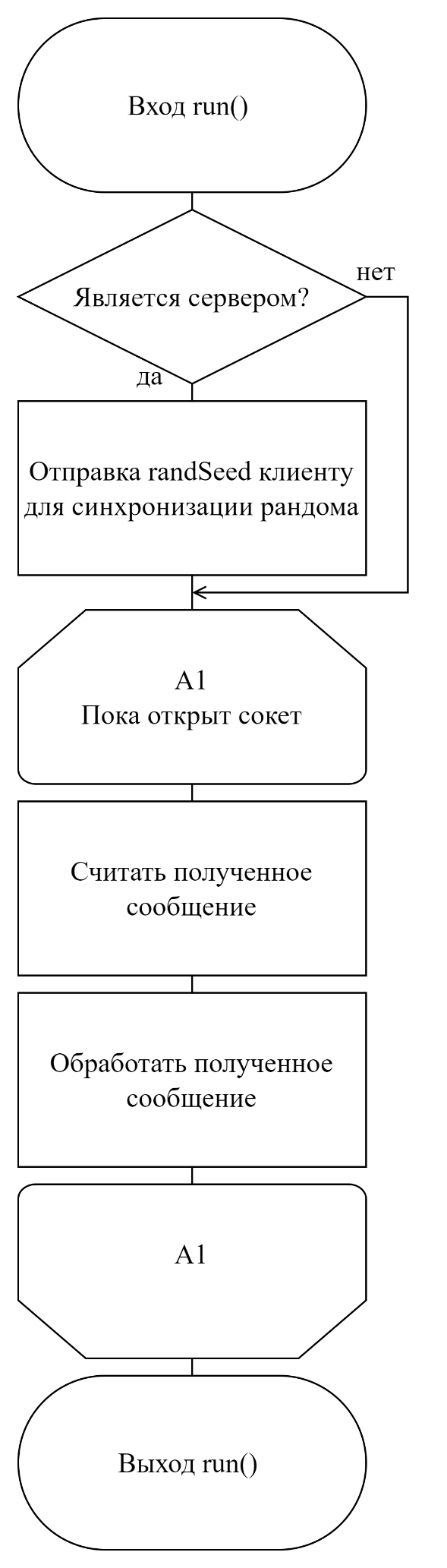


Рисунок 3.5 – Схема алгоритма run

# тестирование и проверка работоспособности программного средства

## Взаимодействие с главным меню

### Тест 1

Таблица 3 – Тест1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности отображения главного меню при запуске программы |
| Исходный набор данных: | Запуск программы |
| Ожидаемый результат: | Корректное открытие главного меню |
| Полученный результат: |  |

### Тест 2

Таблица 4 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности открытия мастера колод |

Продолжение таблицы 4 – Тест 2

|  |  |
| --- | --- |
| Исходный набор данных: | Нажатие на кнопку «Мастер колод» |
| Ожидаемый результат: | Корректное открытие мастера колод |
| Полученный результат: |  |

### Тест 3

Таблица 5 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности открытия мультиплеера |
| Исходный набор данных: | Нажатие на кнопку «Мультиплеер» |
| Ожидаемый результат: | Корректное открытие мультиплеера |
| Полученный результат: |  |

## Взаимодействие с мастером колод

### Тест 4

Таблица 6 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Смена фракции |
| Исходный набор данных: | Нажатие на выпадающий список фракциями |
| Ожидаемый результат: | Корректная смена текущей фракции |
| Полученный результат: | До:    После: |

### Тест 5

Таблица 7 – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация: | Добавление карты в колоду |
| Исходный набор данных: | Нажатие на карту существа в списке |
| Ожидаемый результат: | Добавление карты в колоду в правой части экрана |

Продолжение таблицы 7 – Тест 5

|  |  |
| --- | --- |
| Полученный результат: |  |

### Тест 6

Таблица 8 – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация: | Удаление карты из колоды |
| Исходный набор данных: | Нажатие на карту в списке или на название карты в списке карт в колоде |
| Ожидаемый результат: | Удаление карты из колоды в правой части экрана |
| Полученный результат: |  |

### Тест 7

Таблица 9 – Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка работоспособности кнопки «Очистить» |
| Исходный набор данных: | Нажатие на кнопку «Очистить» |
| Ожидаемый результат: | Удаление всех карт из колоды |
| Полученный результат: | До:    После: |

### Тест 8

Таблица 10 – Тест 8

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка работоспособности кнопки «Загрузить» |
| Исходный набор данных: | Нажатие на кнопку «Загрузить» |
| Ожидаемый результат: | Загрузка выбранной колоды |
| Полученный результат: |  |

### Тест 9

Таблица 11 – Тест 9

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка работоспособности кнопки «Сохранить» |
| Исходный набор данных: | Нажатие на кнопку «Сохранить» |
| Ожидаемый результат: | Сохранение созданной колоды |
| Полученный результат: |  |

## Взаимодействие с меню мультиплеера

### Тест 10

Таблица 12 – Тест 10

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка возможности загрузить колоду, созданную в мастере колод |
| Исходный набор данных: | Выбор в выпадающем списке пункта «Загрузить свою» |
| Ожидаемый результат: | Загрузка кастомной колоды |
| Полученный результат: |  |

### Тест 11

Таблица 13 – Тест 11

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности запуска сервера |
| Исходный набор данных: | Нажатие на кнопку «Сервер» |
| Ожидаемый результат: | Запуск сервера |
| Полученный результат: |  |

### Тест 12

Таблица 14 – Тест 12

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности подключения клиента |
| Исходный набор данных: | Нажатие на кнопку «Клиент» |
| Ожидаемый результат: | Подключение клиена, запуск игры |
| Полученный результат: |  |

## Игровой процесс

### Тест 13

Таблица 15 – Тест 13

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка корректности передачи хода и обновление количества маны у игрока, принимающего ход |
| Исходный набор данных: | Нажатие на кнопку «Передать» |
| Ожидаемый результат: | Передача хода другому игроку и обновление количества маны |
| Полученный результат: |  |

### Тест 14

Таблица 16 – Тест 14

|  |  |
| --- | --- |
| Тестовая ситуация: | Использование карты |
| Исходный набор данных: | Нажатие на кнопку на нужную карту |
| Ожидаемый результат: | Появление карты на столе у обоих игроков, обновление количества маны у сыгравшего игрока |
| Полученный результат: |  |

Продолжение таблицы 16 – Тест 14

|  |  |
| --- | --- |
| Тестовая ситуация: |  |

### Тест 15

Таблица 17 – Тест 15

|  |  |
| --- | --- |
| Тестовая ситуация: | Атака одним миньоном другого |
| Исходный набор данных: | Перетягивание курсора мыши с зажатой левой кнопкой с атакующего миньона на атакуемого |
| Ожидаемый результат: | Вычитание очков урона у обоих миньонов и отображение результата у обоих игроков |
| Полученный результат: |  |

### Тест 16

Таблица 18 – Тест 16

|  |  |
| --- | --- |
| Тестовая ситуация: | Атака миньоном героя |
| Исходный набор данных: | Перетягивание курсора мыши с зажатой левой кнопкой с атакующего миньона на героя противника |
| Ожидаемый результат: | Вычитание очков урона у героя противника и отображение результата у другого игрока |
| Полученный результат: |  |

### Тест 17

Таблица 19 – Тест 17

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка работоспособности свойства миньона |
| Исходный набор данных: | Нахождение миньона на поле. |
| Ожидаемый результат: | Другие миньона класса «Горгулья» получают прибавку к урону и/или здоровью и результат отображается у другого игрока |
| Полученный результат: |  |

# Руководство по использованию программного средства

## Мастер колод

В мастере колод у игрока имеется возможность собрать собственную колоду из имеющихся в игре карт. Для добавления карты необходимо нажать правой кнопкой мыши на карту, которую пользователь желает добавить. Для удаления необходимо нажать на имя карты в колоде или же на изображение карты в списке. Для сохранения/загрузки/очистки колоду есть соответствующие кнопки под списком доступных карт.

## Установление соединения

Для установления соединения первому игроку необходимо нажать кнопку «Сервер», на экране появится IP-адрес игрока и порт. Второму игроку необходимо нажать кнопку «Клиент», ввести IP-адрес сервера и нажать кнопку «Ок».

## Игра

После выполнения действий из пункта 5.2 начнется игра. Первым всегда ходит сервер. Во время хода игрок может разыграть карту или атаковать миньоном. В конце хода для передачи его другому игроку необходимо нажать кнопку «Передать» рядом с портретом героя.

## Завершение игры

При желании по нажатию кнопки «Escape» во время игры можно выйти в главное меню и при закрытии приложения соединение будет разорвано. Игра продолжается до тех пор, пока один из игроков не выйдет из нее или же пока здоровье одного из героев игроков не упадет до 0.

Заключение

В результате разработки игры «Карточная игра по мотивам Heroes 3» было получено приложение, позволяющее полноценно испытать достоинства и недостатки коллекционных карточных игр.

Данная игра является простой в освоении и интуитивно понятной, так как все функции реализованы с использованием максимально понятных механик.

Для успешного выполнено всех поставленных целей потребовалось ознакомиться со средой разработки Intelij Idea. Для создания графического интерфейса требовалось изучить различные аналоги игры в данном жанре. Также потребовалось изучить возможности библиотеки Swing.

Приложение прошло все этапы тестирования, в результате которых были устранены все неполадки. Приложение имеет относительно высокую скорость работы. Возможна дальнейшая доработка при выявлении ошибок.

Список использованной литературы

[1] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2004. – 1 ч.

[2] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2005. – 2 ч.

[3] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 3 ч.

[4] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2013. – 4 ч.

[5] Серебряная Л.В. Структуры и алгоритмы обработки данных: учеб.-метод. пособие / Л.В. Серебряная, И.М. Марина. – Минск: БГУИР, 2013. – 5 с.

[6] Вирт Н. Алгоритмы и структуры данных / Н. Вирт. – Москва: Мир 1989. – 90 с.

[7] Глухова Л.А. Основы алгоритмизации и программирования: учебное пособие / Л.А. Глухова. – Минск: БГУИР, 2006. – 1 ч.

[8] JavaRush[Электронный ресурс]. – Режим доступа: <https://javarush.com/> – Дата обращения: 10.05.2023.

[9] JavaOnline[Электронный ресурс]. – Режим доступа: <https://java-online.ru/libs-swing.xhtml> – Дата обращения: 05.05.2023.

[10] StackOverflow[Электронный ресурс]. – Режим доступа: <https://stackoverflow.com/> – Дата обращения: 12.05.2023.

[11] Metanit[Электронный ресурс]. – Режим доступа: <https://metanit.com/java/> – Дата обращения: 11.05.2023.

[12] Сурков Д.А. Сети ЭВМ: лабораторный практикум / Д.А. Сурков, С.В. Коростель, Е.В. Мельникова, И.М. Марина. – Минск: БГУИР, 2006. – 1 ч.

Приложение А

(обязательное)

Исходный код программы

Модуль Instance.java

package multiplayer;  
  
import cards.Card;  
import customDecks.CustomDeck;  
import customDecks.HeroClass;  
import logic.Board;  
import logic.Hero;  
import logic.Main;  
import logic.Sticker;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.net.SocketException;  
import java.util.ArrayList;  
import java.util.Random;  
import javax.swing.JOptionPane;  
  
public final class Instance implements Runnable {  
 public Hero hero;  
 public boolean isServer;  
 public static volatile boolean *isSyncedRandom* = false;  
 public boolean isReceivedDeck = false;  
 public static int *port* = 444;  
 static boolean *isConnected* = false;  
 private static final int *randSeed* = (int)

(Math.*random*() \* 9999);  
 public static String *connectionAddress* = "localhost";  
 public static Random *random* = new Random(*randSeed*);  
 public ServerSocket serverSocket;  
 public Socket socket;  
 public InputStreamReader input;  
 public PrintStream output;  
 public BufferedReader reader;  
 public static Instance *instance*;  
  
  
 public static void closeConnections() throws

IOException {  
 if (*instance* != null) {  
 if (*instance*.isServer) {  
 *instance*.serverSocket.close();  
 } else {  
 *instance*.socket.close();  
 }  
 }  
 *isSyncedRandom* = false;  
 }  
  
 //Creates instance to control the given hero,if client,

//automatically tries to connect  
 public Instance(Hero hero, boolean isServer) throws

Exception {  
 *closeConnections*();  
 *instance* = this;  
 this.hero = hero;  
 this.isServer = isServer;  
 if (isServer) {  
 setupServer();  
 } else {  
 setupClient();  
 }  
 Thread t = new Thread(this);  
 t.start();  
 }  
  
 public void setupServer() throws Exception {  
 serverSocket = new ServerSocket(*port*);  
 int timeoutDuration = 50000; //Time to connect  
 new TimeoutController(timeoutDuration,

serverSocket);  
 socket = serverSocket.accept();  
  
 //Reaching this point means connection is

//successful  
 *isConnected* = true;  
 input = new

InputStreamReader(socket.getInputStream());  
 reader = new BufferedReader(input);  
 output = new PrintStream(socket.getOutputStream());  
 }  
  
 public void setupClient() throws Exception {  
 socket = new Socket(Instance.*connectionAddress*,

444);  
 input = new

InputStreamReader(socket.getInputStream());  
 output = new PrintStream(socket.getOutputStream());  
 reader = new BufferedReader(input);  
 }  
  
  
 public synchronized void sendMessage(String message) {  
 output.println(message);  
 }  
  
 @Override  
 public void run() {  
 try {  
 if (isServer) {  
 output.println("randSeed: " + *randSeed*);  
 }  
 while (!socket.isClosed()) {  
 String message = reader.readLine();  
 parseMessage(message);  
 }  
 } catch (SocketException se) {  
 if (se.getMessage().equals("Connection reset")) {  
 JOptionPane.*showMessageDialog*(null, "Соединение разорвано");  
 System.*exit*(0);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public synchronized void sendDeck(ArrayList<Card> deck)

{  
 sendMessage("deckSend");  
 for (Card c : deck) {  
 sendMessage("card:" + c.name);  
 }  
 sendMessage("deckEnd");  
 }  
  
 /\* PLAY COMMANDS FORMAT: actorType-actorIndex-targetType-targetIndex  
 \* actorType: what we call to perform an action: c - card, m - minion  
 \* actorIndex: index of minion/card: 0-5 cards in hand, 0-3 minion on board  
 \* targetType: fm(friendly minion), em(enemy minion), fh(friendly hero), eh(enemy hero)  
 \* target index: index of target  
 \* "end": end turn  
 \* \*/  
 private boolean receivingDeck = false;  
  
 private synchronized void parseMessage(String message)

{  
 try {  
 if (message == null) return;  
 if (message.startsWith("heroPortrait")) {  
 hero.heroPortraitPath = message.split

(" ")[1];  
 switch (hero.heroPortraitPath) {  
 case "Assets\\hero.png" -> hero.picture

= HeroClass.*Dungeon*.getHeroPortrait();  
 case "Assets\\towerHero.png" ->

hero.picture =

HeroClass.*Tower*.getHeroPortrait();  
 }  
 }  
 if (message.equals("gotRandom")) {  
 *isSyncedRandom* = true;  
 return;  
 }  
 if (message.equals("deckSend")) {  
 hero.deck.clear();  
 receivingDeck = true;  
 return;  
 }  
 if (message.equals("deckEnd")) {  
 receivingDeck = false;  
 isReceivedDeck = true;  
 }  
 if (receivingDeck &&

message.startsWith("card:")) {  
 Card toAdd =

CustomDeck.*getCard*(message.substring(5));  
 toAdd.setHero(hero);  
 hero.deck.add(toAdd);  
 }  
 if (message.startsWith("randSeed: ")) {  
 if (*isSyncedRandom*) return;  
 int seed = Integer.*parseInt*(message.split

(" ")[1]);  
 *random* = new Random(seed);  
 *isSyncedRandom* = true;  
 sendMessage("gotRandom");  
 return;  
 }  
 if (message.equals("end")) {  
 Main.*wait*(300);  
 Board.*controller*.nextTurn();  
 return;  
 }  
 if (!hero.turn) return; //Messages are not

//executed if it's not your turn  
 String[] contents = message.split(" ");  
 int actorIndex = Integer.*parseInt*(contents[1]);  
 int targetIndex =

Integer.*parseInt*(contents[3]);  
 Card toUse = null;  
 switch (contents[0]) {  
 case "c": //Cast card  
 switch (contents[2]) {  
 case "fm" -> { //On friendly minion  
 toUse =

hero.hand.get(actorIndex);  
 displayCard(toUse);

toUse.cast(hero.minions.

get(targetIndex));  
 }  
 case "em" -> { //On enemy minion  
 toUse =

hero.hand.get(actorIndex);  
 displayCard(toUse);

toUse.cast(hero.opponent.minions.

get(targetIndex));  
 }  
 case "fh" -> { //On friendly hero  
 toUse =

hero.hand.get(actorIndex);  
 displayCard(toUse);  
 toUse.castOnHero(hero);  
 }  
 case "eh" -> { //On enemy hero  
 toUse =

hero.hand.get(actorIndex);  
 displayCard(toUse);

toUse.castOnHero(hero.opponent);  
 }  
 case "n" -> {  
 toUse =

hero.hand.get(actorIndex);  
 displayCard(toUse);  
 toUse.cast(null);  
 }  
 }  
 break;  
 case "m": //Minion attacks  
 switch (contents[2]) {  
 case "fm" -> //Friendly

//minion(forbidden)

hero.minions.get(actorIndex).

attack(hero.minions.

get(targetIndex));  
 case "em" -> //Enemy minion

hero.minions.get(actorIndex).

attack(hero.opponent.minions.

get(targetIndex));  
 case "fh" -> //Friendly

//hero(forbidden)

hero.minions.get(actorIndex).

attack(hero);  
 case "eh" -> //Enemy hero

hero.minions.get(actorIndex).

attack(hero.opponent);  
 }  
 break;  
 }  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 //Used to show user that card is being played  
 private void displayCard(Card card) {  
 new Sticker(card, 1700, 200, 1600);  
 Main.*wait*(1600);  
 }  
}

Модуль GameController.java

package cardgame;  
  
import GUI.MultiplayerFrame;  
import Multiplayer.Instance;  
  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class GameController {  
 public Hero activeHero;  
 public Hero inactiveHero;  
 public Board board;  
  
 public GameController(Board board) {  
 this.board = board;  
 }  
  
 //Actions to do on turn switch  
 public void nextTurn() {  
 activeHero.onTurnEnd();  
 Hero temp = activeHero;  
 activeHero = inactiveHero;  
 inactiveHero = temp;  
 activeHero.onTurnStart();  
 }  
  
 public void startGame() {  
  
 Board.*topHero*.shuffle();  
 Board.*botHero*.shuffle();  
 for (int i = 0; i < 4; i++) {  
 Board.*topHero*.draw();  
 Board.*botHero*.draw();  
 }  
 activeHero = Board.*botHero*;  
 activeHero.turn = true;  
 inactiveHero = Board.*topHero*;  
 inactiveHero.turn = false;  
 }  
  
 //Starts the game for multiplayer  
 public void startGame(boolean isServer) {  
 if (MultiplayerFrame.*mainMultiplayerFrame* != null)

MultiplayerFrame.*mainMultiplayerFrame*.dispose();  
 Instance.*instance*.sendDeck(Board.*playerHero*.deck);  
 while (!Instance.*isSyncedRandom* || !Instance.*instance*.isReceivedDeck) {  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }

if (isServer) {  
 Board.*botHero*.shuffle();  
 Board.*topHero*.shuffle();  
 } else {  
 Board.*topHero*.shuffle();  
 Board.*botHero*.shuffle();  
 }  
  
 for (int i = 0; i < 4; i++) {  
 Board.*topHero*.draw();  
 Board.*botHero*.draw();  
 }  
 if (isServer) {  
 activeHero = Board.*playerHero*;  
 activeHero.turn = true;  
 inactiveHero = Board.*enemyHero*;  
 inactiveHero.turn = false;  
 } else {  
 activeHero = Board.*enemyHero*;  
 activeHero.turn = true;  
 inactiveHero = Board.*playerHero*;  
 inactiveHero.turn = false;  
 }  
 activeHero.maxResource = 1;  
 activeHero.resource = 1;  
 inactiveHero.resource = 1;  
 inactiveHero.maxResource = 1;  
 }  
  
}