МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание игрового поля

Студент гр. 9381 _____ Камакин Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Изучить синтаксис языка программирования C++, ознакомиться с основными принципами объектно-ориентированного программирования. Реализовать класс игрового поля и каждого элемента, также написать GUI.

Задание.

Написать класс игрового поля, которое представляет из себя прямоугольник (двумерный массив). Для каждого элемента поля должен быть создан класс клетки. Клетка должна отображать, является ли она проходимой, а также информацию о том, что на ней находится. Также, на поле должны быть две особые клетки: вход и выход.

При реализации поля запрещено использовать контейнеры из stl

Обязательные требования:

- Реализован класс поля
- Реализован класс клетки
- Для класса поля написаны конструкторы копирования и перемещения, а также операторы присваивания и перемещения
- Поле сохраняет инвариант - из любой клетки можно провести путь до любой другой
- Гарантированно отсутствует утечки памяти

Дополнительные требования:

- Поле создается с использованием паттерна Синглтон
- Для обхода по полю используется паттерн Итератор. Итератор должен быть совместим со стандартной библиотекой.

Выполнение работы.

Написание работы производилось на базе операционной системы Ubuntu 20.04 в среде разработки QtCreator с использованием фреймворка Qt.

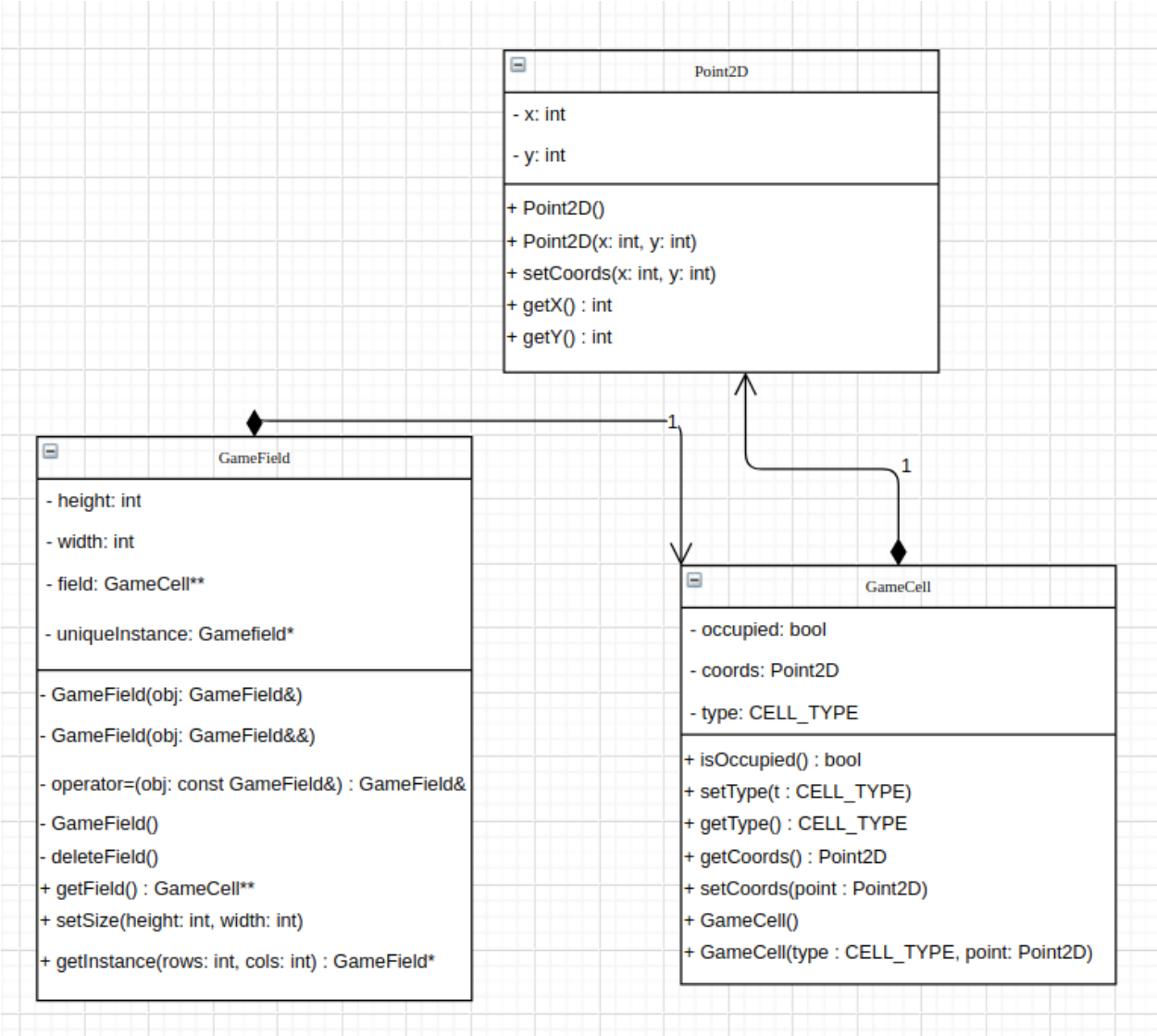
Для игрового поля был реализован класс GameField. Написаны конструкторы копирования и перемещения, а также операторы присваивания и перемещения. Класс содержит двумерный массив клеток собственного класса GameCell, а также переменную и соответствующие функции для реализации паттерна «Одиночка».

Класс GameCell служит для реализации элемента на игровом поле. Каждая клетка содержит поля для хранения её типа, информацию о положении и занятости объектом. Реализованы функции для получения текущего состояния клетки.

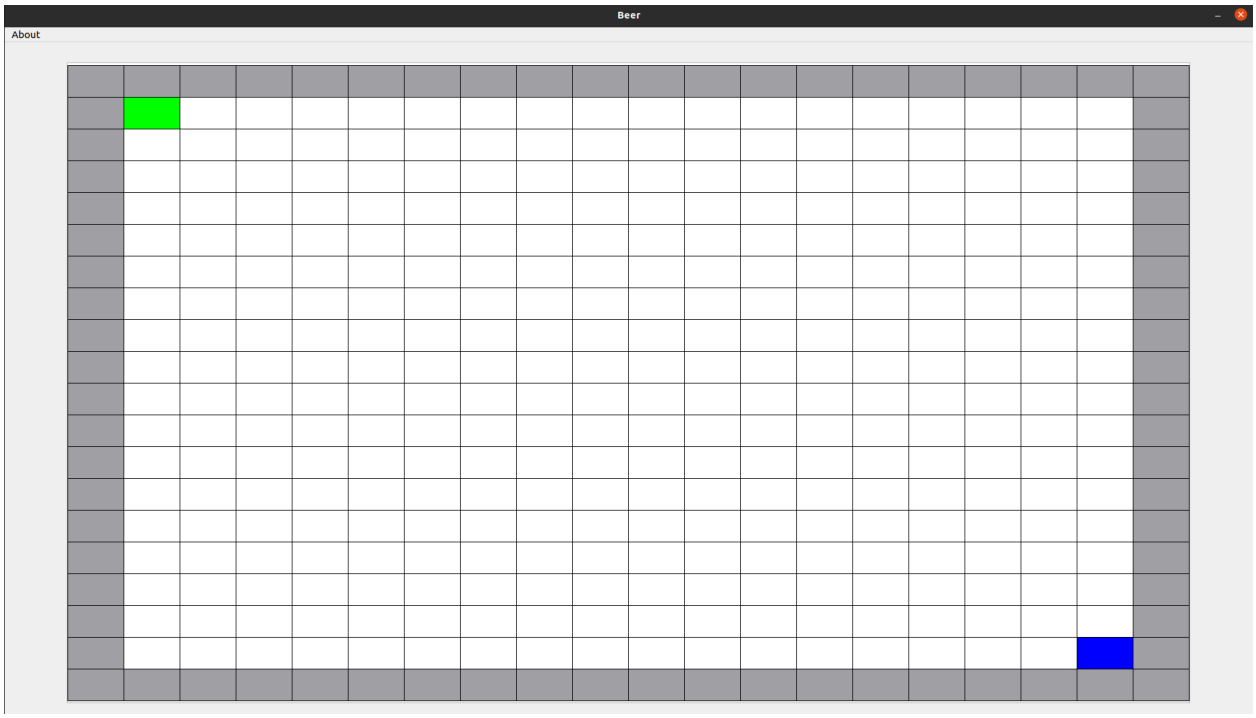
Для создания GUI был использован фреймворк Qt, написан класс MainWindow, объект которого отображается в функции main(). В menuBar был добавлен пункт информации об авторе программы. Поле отображается при помощи QGraphicsView, сцена QGraphicsScene которого содержит клетки QGraphicsRectItem, внешний вид которых зависит от типа элемента на игровом поле GameField. Размер игрового поля 20X20.

Кроме того, для удобства работы с координатами был написан класс Point2D, в котором определены функции для получения и установки координат.

UML диаграмма:



Тестирование.



Программа успешно запускается, поле корректно отображается.

Выводы.

Изучен синтаксис языка программирования C++, реализован класс игрового поля, все необходимые конструкторы и операторы, для каждого элемента создан класс клетки. Также, при помощи фреймворка Qt, был написан GUI для отображения информации об игровом поле.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
    MainWindow game;
    game.show();
    return app.exec();
}
```

Название файла: gamecell.cpp

```
#include "gamecell.h"

GameCell::GameCell() {
    coords = Point2D(0, 0);
}

GameCell::GameCell(CELL_TYPE t, Point2D point) {
    type = t;
    coords = point;
}

void GameCell::setType(CELL_TYPE t) {
    type = t;
}

CELL_TYPE GameCell::getType() {
    return type;
}

bool GameCell::isOccupied() {
    return occupied;
}
```

Название файла: gamefield.cpp

```
#include "gamefield.h"

GameField *GameField::uniqueInstance = nullptr;

GameField *GameField::getInstance(int rows, int cols) {
    if (!uniqueInstance)
        uniqueInstance = new GameField(rows, cols);
    return uniqueInstance;
}

GameField::GameField() {
    field = nullptr;
```

```
    }

    GameField::GameField(int rows, int cols) : height(rows),
width(cols), field (new GameCell*[height]) {
        for (auto y = 0; y < rows; y++) {
            field[y] = new GameCell[width];
            for (auto x = 0; x < cols; x++) {

                if (!y || x == cols - 1 || y == rows - 1 || !x)
                    field[y][x] = GameCell(WALL, Point2D(x, y));

                else
                    field[y][x] = GameCell(EMPTY, Point2D(x, y));
            }
        }

        field[1][1].setType(ENTER);
        field[rows - 2][cols - 2].setType(EXIT);
    }

    GameCell** GameField::getField() {
        return field;
    }

    GameField& GameField::operator=(const GameField &obj) {
        if (&obj == this)
            return *this;

        deleteField();
        height = obj.height;
        width = obj.width;

        field = new GameCell*[height];
        for (auto y = 0; y < height; y++) {
            field[y] = new GameCell[width];
            for (auto x = 0; x < width; x++) {
                field[y][x] = obj.field[y][x];
            }
        }

        return *this;
    }

    GameField::GameField(GameField &&obj) : height(obj.height),
width(obj.width),
                                            field(obj.field) {
        obj.field = nullptr;
        obj.height = 0;
        obj.width = 0;
    }

    GameField::GameField(GameField &obj) : height(obj.height),
width(obj.width),
                                                                   field(new
GameCell*[height]) {
        for (auto y = 0; y < height; y++) {
```

```
            field[y] = new GameCell[width];
            for (auto x = 0; x < width; x++) {
                field[y][x] = obj.field[y][x];
            }
        }
    }

    void GameField::deleteField() {
        for (auto y = 0; y < height; y++)
            delete [] field[y];

        delete [] field;
    }

    void GameField::setSize(int rows, int cols) {
        if (field)
            deleteField();

        field = new GameCell *[rows];
        for (auto y = 0; y < height; y++)
            field[y] = new GameCell[cols];
    }
```

Название файла: mainwindow.cpp

```
    #include "mainwindow.h"
    #include "ui_mainwindow.h"

    MainWindow::MainWindow(QWidget *parent)
        : QMainWindow(parent)
        , ui(new Ui::MainWindow) {
        ui->setupUi(this);

                                        QSize        screenSize        =
QDesktopWidget().availableGeometry(this).size();
        setFixedSize(screenSize);
        ui->view->setFixedSize(screenSize * FIELD_COEFFICIENT);

        gameField = GameField::getInstance(ROWS, COLS);
        ui->view->setScene(getScene(gameField, screenSize));
    }

    QBrush MainWindow::getBrush(CELL_TYPE type) {
        switch (type) {
            case EMPTY:
                return QBrush(Qt::white);
            case WALL:
                return QBrush(Qt::gray);
            case ENTER:
                return QBrush(Qt::green);
            case EXIT:
                return QBrush(Qt::blue);
            default:
                return QBrush(Qt::transparent);
        }
    }
```

```
    QGraphicsScene*   MainWindow::getScene(GameField   *field,   QSize
&size) {
        int rectWidth = (size.width() * FIELD_COEFFICIENT) / COLS;
        int rectHeight = (size.height() * FIELD_COEFFICIENT) / ROWS;
        GameCell **array = field->getField();

        QGraphicsScene *scene = new QGraphicsScene();

        for (auto y = 0; y < ROWS; y++) {
            for (auto x = 0; x < COLS; x++) {
                  QGraphicsRectItem *rect = new QGraphicsRectItem(x *
rectWidth, y * rectHeight,
                                                                    rectW
idth, rectHeight, nullptr);
                rect->setBrush(getBrush(array[y][x].getType()));
                scene->addItem(rect);
            }
        }

        return scene;
    }

    void MainWindow::on_actionAuthor_triggered() {
          QMessageBox::information(this,  "Author",  "Made  by  Kamakin
Daniil, 9381");
    }

    MainWindow::~MainWindow() {
        delete ui;
    }
```

Название файла: point2d.cpp

```
    #include "point2d.h"

    Point2D::Point2D() {
        x = 0;
        y = 0;
    }

    Point2D::Point2D(int x, int y) : x(x), y(y) {}

    void Point2D::setCoords(int x, int y) {
        this->x = x;
        this->y = y;
    }

    int Point2D::getX() {
        return x;
    }

    int Point2D::getY() {
        return y;
    }
```

Название файла: gamecell.h

```
#ifndef GAMECELL_H
#define GAMECELL_H

#include "point2d.h"

enum CELL_TYPE {
    EMPTY,
    ENTER,
    EXIT,
    WALL
};

class GameCell {
public:
    GameCell();
    GameCell(CELL_TYPE type, Point2D point);

    bool isOccupied();
    void setType(CELL_TYPE t);
    CELL_TYPE getType();
    Point2D getCoords();
    void setCoords(Point2D point);

private:
    bool occupied;
    Point2D coords;
    CELL_TYPE type;
};

#endif // GAMECELL_H
```

Название файла: gamefield.h

```
#ifndef GAMEFIELD_H
#define GAMEFIELD_H

#include "gamecell.h"

class GameField {
public:
    GameCell** getField();
    static GameField* getInstance(int rows, int cols);
    void setSize(int height, int width);

private:
    GameField(GameField &obj);
    GameField(GameField &&obj);
    GameField& operator=(const GameField &obj);
    GameField();
    GameField(int rows, int cols);
    void deleteField();

    static GameField *uniqueInstance;
    int height;
    int width;
```

```
    GameCell **field;
};

#endif // GAMEFIELD_H
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QDesktopWidget>
#include <QGraphicsScene>
#include <QGraphicsRectItem>
#include "gamefield.h"

#define ROWS 20
#define COLS 20
#define FIELD_COEFFICIENT 0.9

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow {
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    QGraphicsScene* getScene(GameField *field, QSize &size);
    QBrush getBrush(CELL_TYPE type);

private slots:
    void on_actionAuthor_triggered();

private:
    Ui::MainWindow *ui;
    GameField *gameField;
};

#endif // MAINWINDOW_H
```

Название файла: point2d.h

```
#ifndef POINT2D_H
#define POINT2D_H

class Point2D {
public:
    Point2D();
    Point2D(int x, int y);
    void setCoords(int x, int y);
    int getX();
    int getY();

private:
    int x;
```

```
    int y;
};

#endif // POINT2D_H
```