МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ ЛАБОРАТОРНОЙ РАБОТЫ №1

ПО ДИСЦИПЛИНЕ «ОПЕРАЦИОННЫЕ СИСТЕМЫ И СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ»
ЗА II СЕМЕСТР

Выполнил:
студент 3-го курса
V-го семестр
факультета ЭИС
группы ПО-4(2)
зачётная книжка №190333
Галанин П. И.
«__» _________ 2021 г.

Проверил:
ассистент
кафедры ИИТ
Дряпко А. В.
«__» _________ 2021 г.

Брест 2021

Цель работы: приобрести практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.

Вариант 5

Игра «Сокобан». Один уровень игры. Общая идея: имеется комната-лабиринт (15X15 ячеек), в которой необходимо расставить ящики (5 штук) на указанные позиции. Главный герой может лишь толкать ящики вперед. Таким образом, возможны конфигурации, из которых не возможно построить желаемое решение (например, если ящик был задвинут в тупик).

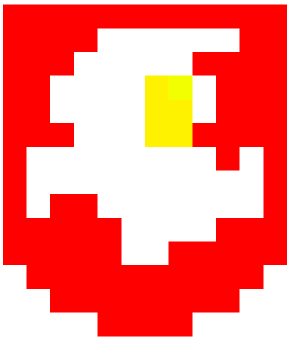

Рисунок 1 – Иконка игры

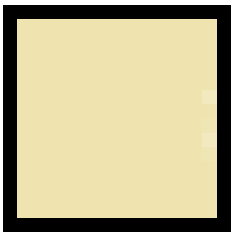

Рисунок 2 – Пустая ячейка

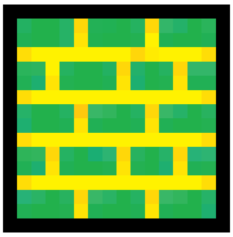

Рисунок 3 – Ячейка с коробкой

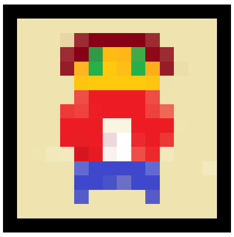

Рисунок 4 – Ячейка с игроком

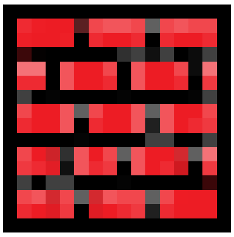

Рисунок 5 – Ячейка со стеной

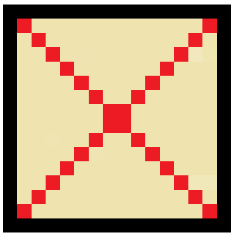

Рисунок 6 – Финиш для коробки

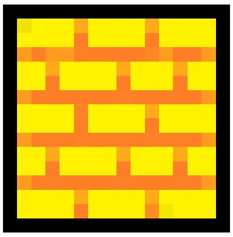

Рисунок 7 – Коробка на финише

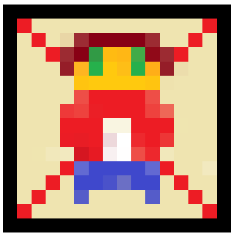

Рисунок 8 – Игрок на финише

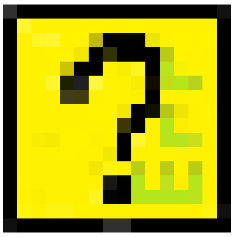

Рисунок 9 – Ячейка без функционала

					ОЛР.ПО4.190333.01 81 00			
Изм.	*Лист*	*№ докум.*	*Подп.*	*Дата*				
Разраб.		*Галанин*			*ОТЧЁТ ЛАБОРАТОРНОЙ РАБОТЫ №1*	*Лит.*	*Лист*	*Листов*
Пров.		*Дряпко*				*О Л Р*	*2*	*21*
						БрГТУ		
Н. контр.		*Дряпко*						
Утв.								

../gpi_osisp5_option5/gpi_osisp5_option5.pro

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    gpi_gamewindow.cpp \
    gpi_gamewindow__gpi_drawAxes.cpp \
    gpi_gamewindow__gpi_drawTextures.cpp \
    gpi_gamewindow__gpi_generateLevel1.cpp \
    gpi_gamewindow__gpi_goBottom.cpp \
    gpi_gamewindow__gpi_goLeft.cpp \
    gpi_gamewindow__gpi_goRight.cpp \
    gpi_gamewindow__gpi_goTop.cpp \
    gpi_gamewindow__gpi_sayWon.cpp \
    main.cpp \
    gpi_mainwindow.cpp

HEADERS += \
    gpi_gamewindow.hpp \
    gpi_mainwindow.hpp

FORMS += \
    gpi_gamewindow.ui \
    gpi_mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    gpi_gamewindow.qrc
```

../gpi_osisp5_option5/main.cpp

```
#include "gpi_mainwindow.hpp"

#include <QApplication>

int main (int argc, char *argv[])
{
    QApplication a (argc, argv);
    gpi_MainWindow w;
    w.show ();
    return a.exec ();
}
```

../gpi_osisp5_option5/gpi_mainwindow.hpp

```
#ifndef GPI_MAINWINDOW_HPP
#define GPI_MAINWINDOW_HPP
```

```cpp

#include <QMainWindow>
#include "gpi_gamewindow.hpp"

QT_BEGIN_NAMESPACE
namespace Ui { class gpi_MainWindow; }
QT_END_NAMESPACE

class gpi_MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    gpi_MainWindow (QWidget *parent = nullptr);
    ~gpi_MainWindow ();

private slots:
    void on_pushButton_ShowGameWindow_clicked ();
    void on_pushButton_CloseMainWindow_clicked ();

private:
    Ui::gpi_MainWindow *ui;
    gpi_GameWindow gpi_gw;
};
#endif // GPI_MAINWINDOW_HPP
```

../gpi_osisp5_option5/gpi_mainwindow.cpp

```cpp
#include "gpi_mainwindow.hpp"
#include "ui_gpi_mainwindow.h"

gpi_MainWindow::gpi_MainWindow (QWidget *parent)
    : QMainWindow (parent)
    , ui (new Ui::gpi_MainWindow)
{
    ui->setupUi(this);
    this->setWindowIcon (QIcon (":/@gpi@/_assets/gpi_MainWindow__favicon.png"));
}

gpi_MainWindow::~gpi_MainWindow ()
{
    delete ui;
}



void gpi_MainWindow::on_pushButton_ShowGameWindow_clicked ()
{
    gpi_gw.show ();
}


void gpi_MainWindow::on_pushButton_CloseMainWindow_clicked ()
{
    this->close ();
}
```

../gpi_osisp5_option5/gpi_gamewindow.hpp

```cpp
#ifndef GPI_GAMEWINDOW_HPP
```

```
#define GPI_GAMEWINDOW_HPP
#define LENGTH 15

#include <QWidget>
#include <QPainter>
#include <QMessageBox>
#include <QKeyEvent>
#include <QTimer>
#include <QIcon>

namespace Ui {
class gpi_GameWindow;
}

class gpi_GameWindow : public QWidget
{
    Q_OBJECT

public:
    explicit gpi_GameWindow (QWidget *parent = nullptr);
    ~gpi_GameWindow ();

public slots:
    void gpi_gamewindow_animate();

protected:
    void paintEvent (QPaintEvent *event) override;
    void keyPressEvent (QKeyEvent *e) override;

private:
    Ui::gpi_GameWindow *ui;

    const int    gpi_length              = LENGTH;
    const int    gpi_WinWidth            = 500;
    const int    gpi_WinHeight           = 500;
    int          gpi_xPlayer;
    int          gpi_yPlayer;
    //================

    enum gpi_MapChar {
        err,
        wall,
        floor,
        player,
        finPlayer,
        box,
        finBox,
        finish,
    } gpi_map[LENGTH][LENGTH] = {{err}};

    void gpi_drawAxes (QPainter* painter);
    void gpi_drawTextures (QPainter *gpi_painter);
    void gpi_generateLevel1 ();
    void gpi_goBottom ();
    void gpi_goLeft ();
    void gpi_goRight ();
    void gpi_goTop ();
    void gpi_sayWon ();
};
```

```
#endif // GPI_GAMEWINDOW_HPP
```

../gpi_osisp5_option5/gpi_gamewindow.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

gpi_GameWindow::gpi_GameWindow (QWidget *parent) :
    QWidget (parent),
    ui (new Ui::gpi_GameWindow)
{
    QTimer*            timer        = new QTimer (this);
    // ================

    ui->setupUi (this);
    this->setWindowIcon (QIcon (":/@gpi@/_assets/gpi_MainWindow__favicon.png"));

    connect (timer, SIGNAL (timeout ()), this, SLOT (gpi_gamewindow_animate ()));
    timer->start (100);

    this->setFixedSize (this->gpi_WinWidth, this->gpi_WinHeight);
    this->gpi_generateLevel1 ();
}

gpi_GameWindow::~gpi_GameWindow ()
{
    delete ui;
}

void gpi_GameWindow::paintEvent (QPaintEvent *event)
{
    QPainter           gpi_painter (this);
    // ================

    Q_UNUSED (event);
    this->gpi_sayWon ();
    this->gpi_drawAxes (&gpi_painter);
    this->gpi_drawTextures (&gpi_painter);
}

void gpi_GameWindow::gpi_gamewindow_animate ()
{
    repaint();
}

void gpi_GameWindow::keyPressEvent (QKeyEvent *e)
{
    switch(e->key ())
    {
        case Qt::Key_Up:
        case Qt::Key_W:
            this->gpi_goTop();
            break;
        case Qt::Key_Right:
        case Qt::Key_D:
            this->gpi_goRight();
            break;
        case Qt::Key_Down:
        case Qt::Key_S:
            this->gpi_goBottom();
            break;
        case Qt::Key_Left:
        case Qt::Key_A:
            this->gpi_goLeft();
            break;
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow.qrc

```
<RCC>
    <qresource prefix="@gpi@">
        <file>_assets/gpi_MainWindow__favicon.png</file>
        <file>_assets/gpi_GameWindow__err.png</file>
        <file>_assets/gpi_GameWindow__wall.png</file>
        <file>_assets/gpi_GameWindow__floor.png</file>
        <file>_assets/gpi_GameWindow__player.png</file>
        <file>_assets/gpi_GameWindow__finPlayer.png</file>
        <file>_assets/gpi_GameWindow__box.png</file>
        <file>_assets/gpi_GameWindow__finBox.png</file>
        <file>_assets/gpi_GameWindow__finish.png</file>
    </qresource>
</RCC>
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_drawAxes.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_drawAxes(QPainter *gpi_painter)
{
    int         gpi_x   = this->gpi_WinWidth / this->gpi_length;
    int         gpi_y   = this->gpi_WinHeight / this->gpi_length;
    const int   gpi_l   = this->gpi_length;
    int         gpi_i;
    int         gpi_j;
    //================

    gpi_i = 0;
    while(gpi_i < gpi_l)
    {
        gpi_j = 0;
        while(gpi_j < gpi_l)
        {
            QRect gpi_rect(gpi_x * gpi_i, gpi_y * gpi_j, gpi_x, gpi_y);
            gpi_painter->drawRect(gpi_rect);
            ++gpi_j;
        }
        ++gpi_i;
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_drawTextures.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_drawTextures(QPainter *gpi_painter)
{
    int             gpi_x       = this->gpi_WinWidth / this->gpi_length;
    int             gpi_y       = this->gpi_WinHeight / this->gpi_length;
    const int       gpi_l       = this->gpi_length;
    QString         gpi_path    = ":/img/_pics/err.png";
    int             gpi_i;
    int             gpi_j;
    // ================

    gpi_i = 0;
    while(gpi_i < gpi_l)
    {
        gpi_j = 0;
        while(gpi_j < gpi_l)
        {
            switch(this->gpi_map[gpi_i][gpi_j])
            {
                case finish:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__finish.png";
                    break;
                case floor:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__floor.png";
                    break;
                case wall:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__wall.png";
                    break;
                case player:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__player.png";
                    break;
                case finPlayer:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__finPlayer.png";
                    break;
                case box:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__box.png";
                    break;
                case finBox:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__finBox.png";
                    break;
                default:
                    gpi_path = ":/@gpi@/_assets/gpi_GameWindow__err.png";
                    break;
            }

            QPixmap gpi_pixmap(gpi_path);
            gpi_painter->drawPixmap(gpi_x * gpi_i, gpi_y * gpi_j, gpi_x, gpi_y, gpi_pixmap);
            ++gpi_j;
        }
        ++gpi_i;
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_generateLevel1.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_generateLevel1 ()
{
    int         gpi_iter;
    int         gpi_i;
    int         gpi_j;
    int         gpi_l           = this->gpi_length;
    char          gpi_strMap[]      = "\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxx...xxxxxx\n\
xxxxf@b..xxxxxx\n\
xxxxxx.bfxxxxxx\n\
xxxxfxxb.xxxxxx\n\
xxxx.x.f.xxxxxx\n\
xxxxb.Bbbfxxxxx\n\
xxxx...f..xxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
xxxxxxxxxxxxxxx\n\
";
    int           gpi_strMapLen     = strlen(gpi_strMap);
    // ================

    gpi_iter = 0;
    gpi_i = 0;
    gpi_j = 0;
    while (gpi_iter < gpi_strMapLen)
    {
        if (gpi_strMap[gpi_iter] == '\n')
        {
            ++gpi_j;
            gpi_i = 0;
        }
        else if (gpi_i > gpi_l)
        {
            continue;
        }
        else
        {
            switch(gpi_strMap[gpi_iter])
            {
                case '.':
                    this->gpi_map[gpi_i][gpi_j] = floor;
                    break;
                case 'x':
                    this->gpi_map[gpi_i][gpi_j] = wall;
                    break;
                case '@':
                    this->gpi_map[gpi_i][gpi_j] = player;
                    this->gpi_xPlayer = gpi_i;
                    this->gpi_yPlayer = gpi_j;
                    break;
                case 'f':
                    this->gpi_map[gpi_i][gpi_j] = finish;
                    break;
                case 'A':
                    this->gpi_map[gpi_i][gpi_j] = finPlayer;
                    this->gpi_xPlayer = gpi_i;
                    this->gpi_yPlayer = gpi_j;
                    break;
                case 'b':
                    this->gpi_map[gpi_i][gpi_j] = box;
                    break;
                case 'B':
                    this->gpi_map[gpi_i][gpi_j] = finBox;
                    break;
                default:
                    this->gpi_map[gpi_i][gpi_j] = err;
                    break;
            }
            ++gpi_i;
        }
        ++gpi_iter;
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_goBottom.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_goBottom ()
{
    int         gpi_l           = this->gpi_length;
    int         gpi_px          = this->gpi_xPlayer;
    int         gpi_py          = this->gpi_yPlayer;
    gpi_MapChar gpi_c0;
    gpi_MapChar gpi_c1;
    gpi_MapChar gpi_c2;
    // ================

    if (gpi_py != gpi_l - 1)
    {
        gpi_c0 = this->gpi_map[gpi_px][gpi_py];
        gpi_c1 = this->gpi_map[gpi_px][gpi_py + 1];
        gpi_c2 = this->gpi_map[gpi_px][gpi_py + 2];

        // player -> floor
        // floor  -> player
        if (gpi_c0 == player && gpi_c1 == floor)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            return;
        }

        // finPlayer -> floor
        // finish    -> player
        if (gpi_c0 == finPlayer && gpi_c1 == floor)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            return;
        }

        // player -> box    -> floor
        // floor  -> player -> box
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == floor)
        {
```

```
        gpi_py += 1;
        this->gpi_yPlayer = gpi_py;
        this->gpi_map[gpi_px][gpi_py - 1] = floor;
        this->gpi_map[gpi_px][gpi_py    ] = player;
        this->gpi_map[gpi_px][gpi_py + 1] = box;
        return;
    }

    // finPlayer -> box    -> floor
    // finish    -> player -> box
    if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == floor)
    {
        gpi_py += 1;
        this->gpi_yPlayer = gpi_py;
        this->gpi_map[gpi_px][gpi_py - 1] = finish;
        this->gpi_map[gpi_px][gpi_py    ] = player;
        this->gpi_map[gpi_px][gpi_py + 1] = box;
        return;
    }

    // player -> finBox    -> floor
    // floor  -> finPlayer -> box
    if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == floor)
    {
        gpi_py += 1;
        this->gpi_yPlayer = gpi_py;
        this->gpi_map[gpi_px][gpi_py - 1] = floor;
        this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
        this->gpi_map[gpi_px][gpi_py + 1] = box;
        return;
    }

    // finPlayer -> finBox    -> floor
    // finish    -> finPlayer -> box
    if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == floor)
    {
        gpi_py += 1;
        this->gpi_yPlayer = gpi_py;
        this->gpi_map[gpi_px][gpi_py - 1] = finish;
        this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
        this->gpi_map[gpi_px][gpi_py + 1] = box;
        return;
    }

    // player -> box    -> finish
    // floor  -> player -> finBox
    if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == finish)
    {
        gpi_py += 1;
        this->gpi_yPlayer = gpi_py;
        this->gpi_map[gpi_px][gpi_py - 1] = floor;
        this->gpi_map[gpi_px][gpi_py    ] = player;
        this->gpi_map[gpi_px][gpi_py + 1] = finBox;
        return;
    }

    // finPlayer -> box    -> finish
    // finish    -> player -> finBox
    if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == finish)
    {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            this->gpi_map[gpi_px][gpi_py + 1] = finBox;
            return;
        }

        // player -> finBox    -> finish
        // floor  -> finPlayer -> finBox
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py + 1] = finBox;
            return;
        }

        // finPlayer -> finBox    -> finish
        // finish    -> finPlayer -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py + 1] = finBox;
            return;
        }

        // player -> finish
        // floor  -> finPlayer
        if (gpi_c0 == player && gpi_c1 == finish)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            return;
        }

        // finPlayer -> finish
        // finish    -> finPlayer
        if (gpi_c0 == finPlayer && gpi_c1 == finish)
        {
            gpi_py += 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py - 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            return;
        }
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_goLeft.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"
```

```

void gpi_GameWindow::gpi_goLeft ()
{
    int         gpi_px          = this->gpi_xPlayer;
    int         gpi_py          = this->gpi_yPlayer;
    gpi_MapChar gpi_c0;
    gpi_MapChar gpi_c1;
    gpi_MapChar gpi_c2;
    //=================

    if (gpi_px != 0)
    {
        gpi_c0 = this->gpi_map[gpi_px    ][gpi_py];
        gpi_c1 = this->gpi_map[gpi_px - 1][gpi_py];
        gpi_c2 = this->gpi_map[gpi_px - 2][gpi_py];

        // player -> floor
        // floor  -> player
        if (gpi_c0 == player && gpi_c1 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            return;
        }

        // finPlayer -> floor
        // finish    -> player
        if (gpi_c0 == finPlayer && gpi_c1 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            return;
        }

        // player -> box    -> floor
        // floor  -> player -> box
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px - 1][gpi_py] = box;
            return;
        }

        // finPlayer -> box    -> floor
        // finish    -> player -> box
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px - 1][gpi_py] = box;
            return;
        }

        // player -> finBox    -> floor
        // floor  -> finPlayer -> box
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px - 1][gpi_py] = box;
            return;
        }

        // finPlayer -> finBox    -> floor
        // finish    -> finPlayer -> box
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px - 1][gpi_py] = box;
            return;
        }

        // player -> box    -> finish
        // floor  -> player -> finBox
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px - 1][gpi_py] = finBox;
            return;
        }

        // finPlayer -> box    -> finish
        // finish    -> player -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px - 1][gpi_py] = finBox;
            return;
        }

        // player -> finBox    -> finish
        // floor  -> finPlayer -> finBox
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px - 1][gpi_py] = finBox;
            return;
        }

        // finPlayer -> finBox    -> finish
        // finish    -> finPlayer -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px - 1][gpi_py] = finBox;
            return;
        }

        // player -> finish
        // floor  -> finPlayer
        if (gpi_c0 == player && gpi_c1 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            return;
        }

        // finPlayer -> finish
        // finish    -> finPlayer
        if (gpi_c0 == finPlayer && gpi_c1 == finish)
        {
            gpi_px -= 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px + 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            return;
        }
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_goRight.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_goRight ()
{
    int         gpi_l           = this->gpi_length;
    int         gpi_px          = this->gpi_xPlayer;
    int         gpi_py          = this->gpi_yPlayer;
    gpi_MapChar gpi_c0;
    gpi_MapChar gpi_c1;
    gpi_MapChar gpi_c2;
    // ================

    if (gpi_px != gpi_l - 1)
    {
        gpi_c0 = this->gpi_map[gpi_px    ][gpi_py];
        gpi_c1 = this->gpi_map[gpi_px + 1][gpi_py];
        gpi_c2 = this->gpi_map[gpi_px + 2][gpi_py];

        // player -> floor
        // floor  -> player
        if (gpi_c0 == player && gpi_c1 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            return;
        }

        // finPlayer -> floor
        // finish    -> player
        if (gpi_c0 == finPlayer && gpi_c1 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            return;
        }

        // player -> box    -> floor
        // floor  -> player -> box
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px + 1][gpi_py] = box;
            return;
        }

        // finPlayer -> box    -> floor
        // finish    -> player -> box
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px + 1][gpi_py] = box;
            return;
        }

        // finPlayer -> finBox -> floor
        // finish    -> finPlayer -> box
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px + 1][gpi_py] = box;
            return;
        }

        // player -> finBox    -> floor
        // floor  -> finPlayer -> box
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px + 1][gpi_py] = box;
            return;
        }

        // player -> box    -> finish
        // floor  -> player -> finBox
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px + 1][gpi_py] = finBox;
            return;
        }

        // finPlayer -> box    -> finish
        // finish    -> player -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = player;
            this->gpi_map[gpi_px + 1][gpi_py] = finBox;
            return;
        }

        // player -> finBox    -> finish
        // floor  -> finPlayer -> finBox
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px + 1][gpi_py] = finBox;
            return;
        }

        // finPlayer -> finBox    -> finish
        // finish    -> finPlayer -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            this->gpi_map[gpi_px + 1][gpi_py] = finBox;
            return;
        }

        // player -> finish
        // floor  -> finPlayer
        if (gpi_c0 == player && gpi_c1 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = floor;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            return;
        }

        // finPlayer -> finish
        // finish    -> finPlayer
        if (gpi_c0 == finPlayer && gpi_c1 == finish)
        {
            gpi_px += 1;
            this->gpi_xPlayer = gpi_px;
            this->gpi_map[gpi_px - 1][gpi_py] = finish;
            this->gpi_map[gpi_px    ][gpi_py] = finPlayer;
            return;
        }
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_goTop.cpp

```
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_goTop ()
{
    int         gpi_px          = this->gpi_xPlayer;
    int         gpi_py          = this->gpi_yPlayer;
    gpi_MapChar gpi_c0;
    gpi_MapChar gpi_c1;
    gpi_MapChar gpi_c2;
    //================

    if (gpi_py != 0)
    {
        gpi_c0 = this->gpi_map[gpi_px][gpi_py];
        gpi_c1 = this->gpi_map[gpi_px][gpi_py - 1];
        gpi_c2 = this->gpi_map[gpi_px][gpi_py - 2];

        // player -> floor
        // floor  -> player
        if (gpi_c0 == player && gpi_c1 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            return;
        }

        // finPlayer -> floor
        // finish    -> player
        if (gpi_c0 == finPlayer && gpi_c1 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            return;
        }

        // player -> box    -> floor
        // floor  -> player -> box
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            this->gpi_map[gpi_px][gpi_py - 1] = box;
            return;
        }

        // finPlayer -> box    -> floor
        // finish    -> player -> box
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            this->gpi_map[gpi_px][gpi_py - 1] = box;
            return;
        }

        // player -> box    -> finish
        // floor  -> player -> finBox
        if (gpi_c0 == player && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            this->gpi_map[gpi_px][gpi_py - 1] = finBox;
            return;
        }

        // finPlayer -> box    -> finish
        // finish    -> player -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == box && gpi_c2 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = player;
            this->gpi_map[gpi_px][gpi_py - 1] = finBox;
            return;
        }

        // player -> finBox    -> floor
        // floor  -> finPlayer -> box
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py - 1] = box;
            return;
        }

        // finPlayer -> finBox    -> floor
        // finish    -> finPlayer -> box
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == floor)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py - 1] = box;
            return;
        }

        // player -> finBox    -> finish
        // floor  -> finPlayer -> finBox
        if (gpi_c0 == player && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py - 1] = finBox;
            return;
        }

        // finPlayer -> finBox    -> finish
        // finish    -> finPlayer -> finBox
        if (gpi_c0 == finPlayer && gpi_c1 == finBox && gpi_c2 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            this->gpi_map[gpi_px][gpi_py - 1] = finBox;
            return;
        }

        // player -> finish
        // floor  -> finPlayer
        if (gpi_c0 == player && gpi_c1 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = floor;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            return;
        }

        // finPlayer -> finish
        // finish    -> finPlayer
        if (gpi_c0 == finPlayer && gpi_c1 == finish)
        {
            gpi_py -= 1;
            this->gpi_yPlayer = gpi_py;
            this->gpi_map[gpi_px][gpi_py + 1] = finish;
            this->gpi_map[gpi_px][gpi_py    ] = finPlayer;
            return;
        }
```

```
    }
}
```

../gpi_osisp5_option5/gpi_gamewindow__gpi_sayWon.cpp

```cpp
#include "gpi_gamewindow.hpp"
#include "ui_gpi_gamewindow.h"

void gpi_GameWindow::gpi_sayWon ()
{
    const int    gpi_w              = this->gpi_WinWidth;
    const int    gpi_l              = this->gpi_length;
    int          gpi_i;
    int          gpi_j;
    QMessageBox gpi_msgBox;
    // =================

    gpi_i = 0;
    while (gpi_i < gpi_l)
    {
        gpi_j = 0;
        while (gpi_j < gpi_l)
        {
            if (
                    this->gpi_map[gpi_i][gpi_j] == finish
                    || this->gpi_map[gpi_i][gpi_j] == finPlayer
            )
            {
                return;
            }
            ++gpi_j;
        }
        ++gpi_i;
    }

    this->close ();
    gpi_msgBox.setStyleSheet ("QLabel{min-width: " + QString::number (gpi_w) + "px; }");
    gpi_msgBox.setText ("You won");
    gpi_msgBox.setWindowTitle ("You won");
    gpi_msgBox.exec ();
}
```