Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №1-2

за 5 семестр

По дисциплине: «ОСиСП»

Выполнила:

студентка 3 курса

группы ПО-4(1)

Грицак A.В.

Проверила:

Дряпко А.В.

2021

Лабораторная работа №1-2

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

Вариант 7

Задание:

7) Игра «Сапер». Реализовать игру только для одного размера игрового поля (8Х8 клеток) с

фиксированным количеством случайно расставленных мин (10 штук).

**Текст программы:**

1. **game.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include "startwindow.h"

#include "mybutton.h"

#include "endwindow.h"

#include "settingswindow.h"

#include <QMainWindow>

#include <QtWidgets>

#include <qsizepolicy.h>

#include <QMouseEvent>

extern int sizeField;

extern int countMines;

class **Game** : public QMainWindow

{

Q\_OBJECT

public:

explicit **Game**(QWidget \*parent = 0);

~***Game***();

private:

QSizePolicy \*policy;

StartWindow \*startWindow;

EndWindow \*endWindow;

SettingsWindow \*settingsWindow;

MyButton \*\*buttons;

QWidget \*widget;

QMenu \*menu\_file, \*menu\_help;

QAction \*menu\_file\_newGame, \*menu\_file\_settings;

QLineEdit \*le\_timer, \*le\_countOfMines;

QLabel \*lbl\_timer, \*lbl\_countOfMines;

QTimer \*timer;

int \*\*mines;

bool \*\*flags;

bool firstCheck;

int timerCount;

int countMinesForLineEdit;

QIcon windowIcon;

QIcon zero, one, two, three, four, five, six, seven, eight;

QIcon bomb;

QIcon flag;

int **findAround**(int, int);

void **createField**(int, int);

void **openField**(int, int);

bool **isEmpty**(int, int);

void **start\_window**();

void **end\_window**();

void **minesOpen**(bool);

void **newGame**();

void **smartOpen**(int, int);

void **paintIcons**(int, int);

int **findFlags**(int, int);

void **newField**();

public slots:

void **left\_click**();

void **right\_click**();

void **double\_click**();

void **menu\_newGame**();

void **menu\_settings**();

void **timeOut**();

};

#endif // MAINWINDOW\_H

1. **game.cpp**

#include "game.h"

#include "startwindow.h"

#include <QtWidgets>

#include <ctime>

Game::**Game**(QWidget \*parent) : QMainWindow(*parent*)

{

// Initialize variables.

policy = new QSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);

menu\_file = new QMenu("&File");

menu\_help = new QMenu("&Help");

menu\_file\_newGame = new QAction("&New game", *menu\_file*);

menu\_file\_settings = new QAction("&Settings", *menu\_file*);

le\_timer = new QLineEdit("0");

le\_countOfMines = new QLineEdit;

lbl\_timer = new QLabel;

lbl\_countOfMines = new QLabel;

// Show start window.

start\_window();

// Set style and properties.

this->setStyleSheet(

" QMainWindow { "

" background-color: black; " // MediumPurple

" } "

);

le\_timer->setStyleSheet(

" QLineEdit {"

" width: 18px;"

" font-size: 20px;"

" border-width: 5px;"

" }"

);

le\_countOfMines->setStyleSheet(

" QLineEdit {"

" width: 18px;"

" font-size: 20px;"

" border-width: 5px;"

" }"

);

this->move(300, 0);

this->show();

// Create new field.

newField();

// Set menu bar.

menu\_file->addAction(*menu\_file\_newGame*);

menu\_file->addAction(*menu\_file\_settings*);

menu\_file->addSeparator();

menu\_file->addAction("&Exit", qApp, SLOT(quit()));

menu\_file\_newGame ->setShortcut(QKeySequence(Qt::Key\_F2));

menu\_file\_settings->setShortcut(QKeySequence(Qt::Key\_F5));

menu\_help->addAction("&About Qt", qApp, SLOT(aboutQt()), Qt::Key\_F1);

menuBar()->addMenu(*menu\_file*);

menuBar()->addMenu(*menu\_help*);

// Set widgets to form.

QGridLayout \*layout\_field = new QGridLayout;

layout\_field->*setSpacing*(0);

int k = 0;

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

layout\_field->addWidget(*buttons[k++]*, i, j);

QHBoxLayout \*layout\_lineEdits = new QHBoxLayout;

layout\_lineEdits->addWidget(*lbl\_timer*);

layout\_lineEdits->addWidget(*le\_timer*);

layout\_lineEdits->addStretch(10);

layout\_lineEdits->addWidget(*lbl\_countOfMines*);

layout\_lineEdits->addWidget(*le\_countOfMines*);

QVBoxLayout \*full = new QVBoxLayout;

full->addLayout(*layout\_field*);

full->addLayout(*layout\_lineEdits*);

widget = new QWidget;

widget->setLayout(*full*);

setCentralWidget(*widget*);

// Set images to timer and count of mines.

QPixmap pix;

pix.load(":/image/clock.ico");

lbl\_timer->resize(pix.size());

lbl\_timer->setPixmap(pix);

pix.load(":/image/bomb\_6231.ico");

lbl\_countOfMines->resize(pix.size());

lbl\_countOfMines->setPixmap(pix);

// Read icons.

firstCheck = false;

flag = QIcon(":/image/flag\_green.png");

bomb = QIcon(":/image/bomb.png");

one = QIcon(":/image/one.png");

two = QIcon(":/image/two.png");

three = QIcon(":/image/three.png");

four = QIcon(":/image/four.png");

five = QIcon(":/image/five.png");

six = QIcon(":/image/six.png");

seven = QIcon(":/image/seven.png");

eight = QIcon(":/image/eight.png");

le\_countOfMines->setReadOnly(true);

le\_timer->setReadOnly(true);

timerCount = 0;

le\_countOfMines->setText(QString::number(countMinesForLineEdit = countMines));

for (int i = 0; i < sizeField\*sizeField; i++)

{

connect(buttons[i], &MyButton::leftClicked, this, &Game::left\_click );

connect(buttons[i], &MyButton::rightClicked, this, &Game::right\_click );

connect(buttons[i], &MyButton::doubleClicked, this, &Game::double\_click);

}

connect(menu\_file\_newGame, &QAction::triggered, this, &Game::menu\_newGame );

connect(menu\_file\_settings, &QAction::triggered, this, &Game::menu\_settings);

}

Game::~***Game***()

{

}

int Game::**findAround**(int i, int j)

{

int count = 0;

if (i-1 >= 0 && j-1 >= 0 && mines[i-1][j-1] == 9) count++;

if (i-1 >= 0 && mines[i-1][j] == 9) count++;

if (i-1 >= 0 && j+1 < sizeField && mines[i-1][j+1] == 9) count++;

if (j-1 >= 0 && mines[i][j-1] == 9) count++;

if (j+1 < sizeField && mines[i][j+1] == 9) count++;

if (i+1 < sizeField && j-1 >= 0 && mines[i+1][j-1] == 9) count++;

if (i+1 < sizeField && mines[i+1][j] == 9) count++;

if (i+1 < sizeField && j+1 < sizeField && mines[i+1][j+1] == 9) count++;

return count;

}

void Game::**createField**(int i\_first, int j\_first)

{

while (mines[i\_first][j\_first] != 0)

{

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

mines[i][j] = 0;

srand(time(NULL));

int count = 0;

while (count != countMines)

{

int i = rand() % sizeField, j = rand() % sizeField;

if (mines[i][j] == 9)

continue;

if (mines[i][j] != 9)

mines[i][j] = 9;

if (findAround(i\_first, j\_first) == 0)

count++;

else

mines[i][j] = 0;

}

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

if (mines[i][j] == 0)

mines[i][j] = findAround(i, j);

}

}

void Game::**openField**(int i, int j)

{

if (i >= 0 && i < sizeField)

{

if (j >= 0 && j < sizeField)

{

paintIcons(i, j);

if (!buttons[i\*sizeField+j]->isChecked())

buttons[i\*sizeField+j]->setChecked(true);

while (mines[i][j] == 0)

if (mines[i][j] == 0)

{

mines[i][j] = -1;

openField(i - 1, j - 1);

openField(i - 1, j );

openField(i - 1, j + 1);

openField(i, j - 1);

openField(i, j + 1);

openField(i + 1, j - 1);

openField(i + 1, j );

openField(i + 1, j + 1);

}

else

{

if (isEmpty(i - 1, j - 1)) openField(i - 1, j - 1);

if (isEmpty(i - 1, j )) openField(i - 1, j );

if (isEmpty(i - 1, j + 1)) openField(i - 1, j + 1);

if (isEmpty(i, j - 1)) openField(i, j - 1);

if (isEmpty(i, j + 1)) openField(i, j + 1);

if (isEmpty(i + 1, j - 1)) openField(i + 1, j - 1);

if (isEmpty(i + 1, j )) openField(i + 1, j );

if (isEmpty(i + 1, j + 1)) openField(i + 1, j + 1);

}

}

}

}

bool Game::**isEmpty**(int i, int j)

{

// if cells is empty return true, else return false.

if (i >= 0 && i < sizeField && j >= 0 && j < sizeField && mines[i][j] == 0)

return true;

return false;

}

void Game::**start\_window**()

{

// Show the first window and choose complexity.

startWindow = new StartWindow;

startWindow->*exec*();

int complexity = startWindow->clicked\_start();

if (complexity == 1)

{

sizeField = 10;

countMines = 8;

}

else

std::exit(1);

}

void Game::**minesOpen**(bool isWin)

{

// if you lost then open all mines.

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

if (mines[i][j] == 9)

{

QString loseColor("QPushButton:checked {background-color: red;}");

QString winColor("QPushButton:checked {background-color: green;}");

buttons[i\*sizeField+j]->setStyleSheet(isWin ? winColor : loseColor);

buttons[i\*sizeField+j]->setIcon(bomb);

buttons[i\*sizeField+j]->setChecked(true);

}

}

void Game::**newGame**()

{

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

{

mines[i][j] = 1;

flags[i][j] = false;

}

for (int i = 0; i < sizeField\*sizeField; i++)

{

buttons[i]->setChecked(false);

buttons[i]->setText("");

buttons[i]->setStyleSheet(

" QPushButton { "

" background-color: Black; " // DarkOrchid

" font-size: 25px; "

" border: 1px solid MistyRose; "

" } "

" "

" QPushButton:hover { "

" background-color: Fuchsia; "

" } "

" "

" QPushButton:checked { "

" background-color: White; "

" } "

);

buttons[i]->setIcon(zero);

}

timerCount = 0;

le\_timer->setText("0");

timer->stop();

le\_countOfMines->setText(QString::number(countMines));

countMinesForLineEdit = countMines;

}

void Game::**smartOpen**(int i, int j)

{

if (i-1 >= 0 && j-1 >= 0 && !flags[i-1][j-1])

mines[i-1][j-1] == 0 ? openField(i-1, j-1) : paintIcons(i-1, j-1);

if (i-1 >= 0 && !flags[i-1][j])

mines[i-1][j] == 0 ? openField(i-1, j) : paintIcons(i-1, j);

if (i-1 >= 0 && j+1 < sizeField && !flags[i-1][j+1])

mines[i-1][j+1] == 0 ? openField(i-1, j+1) : paintIcons(i-1, j+1);

if (j-1 >= 0 && !flags[i][j-1])

mines[i][j-1] == 0 ? openField(i, j-1) : paintIcons(i, j-1);

if (j+1 < sizeField && !flags[i][j+1])

mines[i][j+1] == 0 ? openField(i, j+1) : paintIcons(i, j+1);

if (i+1 < sizeField && j-1 >= 0 && !flags[i+1][j-1])

mines[i+1][j-1] == 0 ? openField(i+1, j-1) : paintIcons(i+1, j-1);

if (i+1 < sizeField && !flags[i+1][j])

mines[i+1][j] == 0 ? openField(i+1, j) : paintIcons(i+1, j);

if (i+1 < sizeField && j+1 < sizeField && !flags[i+1][j+1])

mines[i+1][j+1] == 0 ? openField(i+1, j+1) : paintIcons(i+1, j+1);

}

void Game::**paintIcons**(int i, int j)

{

if (mines[i][j] == 0 || mines[i][j] == -1)

{

buttons[i\*sizeField+j]->setText("");

buttons[i\*sizeField+j]->setChecked(true);

}

else if (flags[i][j])

buttons[i\*sizeField+j]->setIcon(flag);

else

{

if (mines[i][j] == 1)

{

buttons[i\*sizeField+j]->setIcon(one);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 2)

{

buttons[i\*sizeField+j]->setIcon(two);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 3)

{

buttons[i\*sizeField+j]->setIcon(three);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 4)

{

buttons[i\*sizeField+j]->setIcon(four);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 5)

{

buttons[i\*sizeField+j]->setIcon(five);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 6)

{

buttons[i\*sizeField+j]->setIcon(six);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 7)

{

buttons[i\*sizeField+j]->setIcon(seven);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 8)

{

buttons[i\*sizeField+j]->setIcon(eight);

buttons[i\*sizeField+j]->setChecked(true);

}

else if (mines[i][j] == 9)

{

buttons[i\*sizeField+j]->setChecked(true);

buttons[i\*sizeField+j]->setIcon(bomb);

}

}

}

int Game::**findFlags**(int i, int j)

{

int count = 0;

if (i-1 >= 0 && j-1 >= 0 && flags[i-1][j-1]) count++;

if (i-1 >= 0 && flags[i-1][j] ) count++;

if (i-1 >= 0 && j+1 < sizeField && flags[i-1][j+1]) count++;

if (j-1 >= 0 && flags[i][j-1] ) count++;

if (j+1 < sizeField && flags[i][j+1] ) count++;

if (i+1 < sizeField && j-1 >= 0 && flags[i+1][j-1]) count++;

if (i+1 < sizeField && flags[i+1][j] ) count++;

if (i+1 < sizeField && j+1 < sizeField && flags[i+1][j+1]) count++;

return count;

}

void Game::**newField**()

{

// Create and initialize arrays.

buttons = new MyButton\* [sizeField\*sizeField];

mines = new int\* [sizeField];

flags = new bool\* [sizeField];

for (int i = 0; i < sizeField; i++)

{

mines[i] = new int [sizeField];

flags[i] = new bool [sizeField];

}

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

{

mines[i][j] = 1;

flags[i][j] = false;

}

// Set properties for buttons.

for (int i = 0; i < sizeField\*sizeField; i++)

{

buttons[i] = new MyButton;

buttons[i]->setMinimumSize(25, 25);

buttons[i]->setCheckable(true);

buttons[i]->setSizePolicy(policy->verticalPolicy(), policy->horizontalPolicy());

buttons[i]->setStyleSheet(

" QPushButton { "

" background-color: Black; " // DarkOrchid

" font-size: 25px; "

" border: 1px solid MistyRose; "

" } "

" "

" QPushButton:hover { "

" background-color: Fuchsia; "

" } "

" "

" QPushButton:checked { "

" background-color: White; "

" } "

);

buttons[i]->setIcon(zero);

}

}

void Game::**left\_click**()

{

int count = 0;

for (int i = 0; i < sizeField\*sizeField; i++, count++)

if (buttons[i] == sender())

break;

int ii = count / sizeField;

int jj = count - ii\*sizeField;

if (!flags[ii][jj])

{

// Open sheet with mine.

if (mines[ii][jj] == 9)

{

timer->stop();

minesOpen(false);

endWindow = new EndWindow;

endWindow->lose\_lbl();

endWindow->show();

endWindow->*exec*();

newGame();

firstCheck = false;

return;

}

// First click (create field).

if (!firstCheck)

{

timer = new QTimer;

timer->start(1000);

connect(timer, &QTimer::timeout, this, &Game::timeOut);

createField(ii, jj);

openField(ii, jj);

firstCheck = true;

}

openField(ii, jj);

bool isWin = true;

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

if (!buttons[i\*sizeField+j]->isChecked() && mines[i][j] != 9)

isWin = false;

// If you win.

if (isWin)

{

timer->stop();

minesOpen(true);

endWindow = new EndWindow;

endWindow->win\_lbl();

endWindow->show();

endWindow->*exec*();

newGame();

firstCheck = false;

return;

}

}

}

void Game::**right\_click**()

{

int count = 0;

for (int i = 0; i < sizeField\*sizeField; i++, count++)

if (buttons[i] == sender())

break;

int ii = count / sizeField;

int jj = count - ii\*sizeField;

if (!buttons[count]->isChecked() && !flags[ii][jj])

{

buttons[count]->setIcon(flag);

flags[ii][jj] = true;

le\_countOfMines->setText(QString::number(--countMinesForLineEdit));

return;

}

if (!buttons[count]->isChecked() && flags[ii][jj])

{

buttons[count]->setIcon(zero);

le\_countOfMines->setText(QString::number(++countMinesForLineEdit));

flags[ii][jj] = false;

}

}

void Game::**double\_click**()

{

int count = 0;

for (int i = 0; i < sizeField\*sizeField; i++, count++)

if (buttons[i] == sender())

break;

int ii = count / sizeField;

int jj = count - ii\*sizeField;

if (mines[ii][jj] == findFlags(ii, jj))

smartOpen(ii, jj);

bool isWin = true;

for (int i = 0; i < sizeField; i++)

for (int j = 0; j < sizeField; j++)

{

if (mines[i][j] == 9 && buttons[i\*sizeField+j]->isChecked())

{

minesOpen(false);

endWindow = new EndWindow;

endWindow->lose\_lbl();

endWindow->show();

endWindow->*exec*();

newGame();

firstCheck = false;

}

if (!buttons[i\*sizeField+j]->isChecked() && mines[i][j] != 9)

isWin = false;

}

if (isWin)

{

timer->stop();

minesOpen(true);

endWindow = new EndWindow;

endWindow->win\_lbl();

endWindow->show();

endWindow->*exec*();

newGame();

firstCheck = false;

return;

}

}

void Game::**menu\_newGame**()

{

newGame();

firstCheck = false;

}

void Game::**menu\_settings**()

{

settingsWindow = new SettingsWindow(*sizeField*, *countMines*);

settingsWindow->show();

settingsWindow->*exec*();

}

void Game::**timeOut**()

{

le\_timer->setText(QString::number(++timerCount));

}

1. **mybutton.h**

#ifndef MYBUTTON\_H

#define MYBUTTON\_H

#include <QPushButton>

#include <QMouseEvent>

class **MyButton** : public QPushButton

{

Q\_OBJECT

public:

explicit **MyButton**(QWidget \*parent = 0);

~***MyButton***();

signals:

void **rightClicked**();

void **leftClicked**();

void **doubleClicked**();

private slots:

void ***mousePressEvent***(QMouseEvent \*event);

void ***mouseDoubleClickEvent***(QMouseEvent \*event);

};

#endif // MYBUTTON\_H

1. **mybutton.cpp**

#include "mybutton.h"

#include <iostream>

MyButton::**MyButton**(QWidget \*parent) : QPushButton(*parent*)

{

}

MyButton::~***MyButton***()

{

}

void MyButton::***mousePressEvent***(QMouseEvent \*event)

{

if (event->button() == Qt::RightButton)

emit rightClicked();

if (event->button() == Qt::LeftButton)

emit leftClicked();

if (event->button() == Qt::MiddleButton)

emit doubleClicked();

}

void MyButton::***mouseDoubleClickEvent***(QMouseEvent \*event)

{

if (event->button() == Qt::LeftButton)

emit doubleClicked();

}

1. **startwindow.h**

#ifndef STARTWINDOW\_H

#define STARTWINDOW\_H

#include <QDialog>

#include <QtWidgets>

class **StartWindow** : public QDialog

{

public:

**StartWindow**(QWidget \*parent = 0);

~***StartWindow***();

QRadioButton \*newg;

QPushButton \*start, \*exit;

QVBoxLayout \*startgame, \*full;

QHBoxLayout \*buttons;

QIcon windowIcon;

public slots:

void **unlockingButton**();

int **clicked\_start**();

bool **clicked\_exit**();

};

#endif // STARTWINDOW\_H

1. **startwindow.cpp**

#include "startwindow.h"

StartWindow::**StartWindow**(QWidget \*parent) : QDialog(*parent*, Qt::WindowCloseButtonHint)

{

this->setStyleSheet(

" QDialog { "

" background-color: Lavender; "

" } "

" "

" QRadioButton { "

" font-size: 15px; "

" } "

" "

" QPushButton { "

" font-size: 20px; "

" } "

);

newg = new QRadioButton("&New Game" );

start = new QPushButton ("&Start" );

start->setEnabled(false);

exit = new QPushButton ("&Exit" );

startgame = new QVBoxLayout;

startgame->addWidget(*newg* );

buttons = new QHBoxLayout;

buttons->addWidget(*start*);

buttons->addWidget(*exit* );

full = new QVBoxLayout;

full->addLayout(*startgame*);

full->addLayout(*buttons*);

setLayout(*full*);

windowIcon = QIcon(":/image/bomb.png");

this->setWindowIcon(windowIcon);

connect(newg, &QRadioButton::clicked, this, &StartWindow::unlockingButton);

connect(start, &QPushButton ::clicked, this, &StartWindow::clicked\_start );

connect(exit, &QPushButton ::clicked, this, &StartWindow::clicked\_exit );

}

StartWindow::~***StartWindow***()

{

}

void StartWindow::**unlockingButton**()

{

start->setEnabled(true);

}

int StartWindow::**clicked\_start**()

{

this->close();

int complexity = newg->isChecked();

return complexity;

}

bool StartWindow::**clicked\_exit**()

{

std::exit(1);

}

1. **endwindow.h**

#ifndef ENDWINDOW\_H

#define ENDWINDOW\_H

#include <QDialog>

#include <QtWidgets>

class **EndWindow** : public QDialog

{

public:

explicit **EndWindow**(QWidget \*parent = 0);

~***EndWindow***();

QPushButton \*newGame, \*exit;

QLabel \*lbl;

QHBoxLayout \*down;

QVBoxLayout \*full;

void **win\_lbl**();

void **lose\_lbl**();

QIcon windowIcon;

public slots:

void **clicked\_exit**();

bool **clicked\_newGame**();

};

#endif // ENDWINDOW\_H

1. **endwindow.cpp**

#include "endwindow.h"

EndWindow::**EndWindow**(QWidget \*parent) : QDialog(*parent*, Qt::WindowSystemMenuHint)

{

this->setStyleSheet(

" QDialog { "

" background-color: Lavender; "

" } "

" "

" QPushButton { "

" font-size: 20px; "

" } "

);

this->setModal(true); // Dialog window is modal.

newGame = new QPushButton("&New Game");

exit = new QPushButton("&Exit" );

lbl = new QLabel;

down = new QHBoxLayout;

full = new QVBoxLayout;

down->addWidget(*newGame*);

down->addWidget(*exit* );

full->addWidget(*lbl*);

full->addLayout(*down*);

setLayout(*full*);

windowIcon = QIcon(":/image/Picture/bomb.png");

this->setWindowIcon(windowIcon);

connect(exit, &QPushButton::clicked, this, &EndWindow::clicked\_exit );

connect(newGame, &QPushButton::clicked, this, &EndWindow::clicked\_newGame);

}

EndWindow::~***EndWindow***()

{

}

void EndWindow::**win\_lbl**()

{

QString style\_lbl("QLabel {color: green;}");

lbl->setStyleSheet(style\_lbl);

lbl->setText("<h2 align=\"center\">You win!</h2>");

}

void EndWindow::**lose\_lbl**()

{

QString style\_lbl("QLabel {color: red;}");

lbl->setStyleSheet(style\_lbl);

lbl->setText("<h2 align=\"center\">You lose!</h2>");

}

void EndWindow::**clicked\_exit**()

{

std::exit(1);

}

bool EndWindow::**clicked\_newGame**()

{

this->close();

return true;

}

1. **main.cpp**

#include "game.h"

#include <QApplication>

int sizeField; // 9 16 25

int countMines; // 10 40 120

int main(int argc, char \*argv[])

{

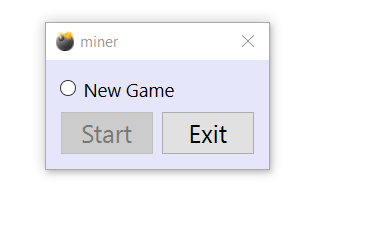
QApplication app(*argc*, *argv*);

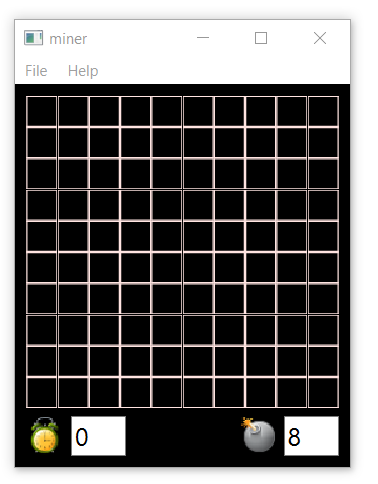
Game window;

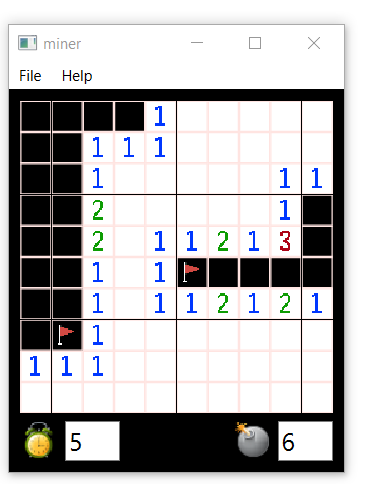
return app.exec();

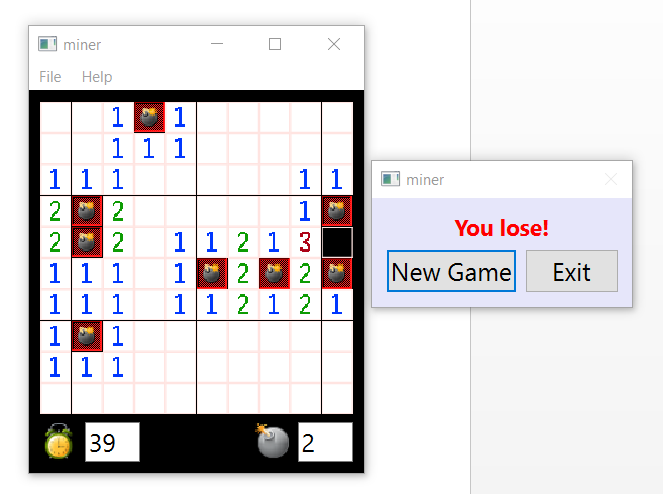
}

**Результаты тестирования программы:**









**Вывод:** приобрела практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.