МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №1-2

# По дисциплине:

# « Операционные Системы и Системное Программирование»

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Кныш К.О.

Проверила:

Дряпко А.В.

Брест 2021

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

Вариант 18

Ход работы:

Задание: Создание игры “Судоку”

Для всех приложений создать пункт главного меню «О программе», в котором дать краткую характеристику разработанной программы.

Для игровых приложений предусмотреть пункт главного меню «Новая игра», по нажатию на который происходит инициализация новой игры.

Код программы:

**Sudoku.h**

#ifndef SUDOKU\_H

#define SUDOKU\_H

#include <QMainWindow>

namespace **Ui** {

class **Sudoku**;

}

class **Sudoku** : public QMainWindow {

Q\_OBJECT

public:

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

virtual ~***Sudoku***();

public slots:

void **changeMode**(QAction\* action);

void **showAbout**();

void **showProgress**(double progress);

void **changeAppFont**(QFont font);

private:

Ui::Sudoku \*ui;

};

#endif // SUDOKU\_H

**DefaultBoards.h**

#ifndef DEFAULTBOARDS\_H

#define DEFAULTBOARDS\_H

#include <QMap>

#include <QList>

#include <QMutex>

#include <QObject>

#include <QSettings>

class **DefaultBoards** : public QObject {

Q\_OBJECT

Q\_ENUMS(**Mode**)

public:

enum **Mode** { Test, Easy, Medium, Hard, Evil };

struct **SimpleBoard** {

int board[9][9];

};

virtual ~***DefaultBoards***();

SimpleBoard **getBoard**(Mode mode);

static DefaultBoards\* **instance**();

static QString **mode2str**(DefaultBoards::Mode mode);

signals:

public slots:

private:

static const int BOARDS = 100;

static DefaultBoards\* m\_instance;

static QMutex& m\_mutex;

QSettings m\_settings;

**DefaultBoards**();

};

#endif // DEFAULTBOARDS\_H

**Position.h**

#ifndef POSITION\_H

#define POSITION\_H

class **Position** {

public:

explicit **Position**();

explicit **Position**(int row, int col);

virtual ~***Position***();

int **row**() const { return m\_row; }

int **col**() const { return m\_col; }

private:

int m\_row;

int m\_col;

};

#endif // POSITION\_H

**Board.h**

#ifndef BOARD\_H

#define BOARD\_H

#include <QWidget>

#include "DefaultBoards.h"

namespace **Ui** {

class **Board**;

}

class **Cell**;

class **Board** : public QWidget {

Q\_OBJECT

Q\_PROPERTY(DefaultBoards::Mode mode READ mode WRITE setMode NOTIFY modeChanged)

public:

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

virtual ~***Board***();

DefaultBoards::Mode **mode**() const { return m\_mode; }

signals:

void **modeChanged**(DefaultBoards::Mode mode);

void **completed**(double stats);

void **gameOver**();

public slots:

void **renew**();

void **clear**();

void **setMode**(DefaultBoards::Mode mode);

private:

DefaultBoards::Mode m\_mode;

Cell\* m\_cells[9][9];

QList<Cell\*> m\_all;

Ui::Board \*ui;

bool **checkLine**(Cell\* cell) const;

bool **checkColumn**(Cell\* cell) const;

bool **checkBox**(Cell\* cell) const;

bool **checkCell**(Cell\* orig, Cell\* other) const;

private slots:

void **updateValue**();

void **checkBoard**();

void **showCongratulations**();

};

#endif // BOARD\_H

**Cell.h**

#ifndef BOARD\_H

#define BOARD\_H

#include <QWidget>

#include "DefaultBoards.h"

namespace **Ui** {

class **Board**;

}

class **Cell**;

class **Board** : public QWidget {

Q\_OBJECT

Q\_PROPERTY(DefaultBoards::Mode mode READ mode WRITE setMode NOTIFY modeChanged)

public:

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

virtual ~***Board***();

DefaultBoards::Mode **mode**() const { return m\_mode; }

signals:

void **modeChanged**(DefaultBoards::Mode mode);

void **completed**(double stats);

void **gameOver**();

public slots:

void **renew**();

void **clear**();

void **setMode**(DefaultBoards::Mode mode);

private:

DefaultBoards::Mode m\_mode;

Cell\* m\_cells[9][9];

QList<Cell\*> m\_all;

Ui::Board \*ui;

bool **checkLine**(Cell\* cell) const;

bool **checkColumn**(Cell\* cell) const;

bool **checkBox**(Cell\* cell) const;

bool **checkCell**(Cell\* orig, Cell\* other) const;

private slots:

void **updateValue**();

void **checkBoard**();

void **showCongratulations**();

};

#endif // BOARD\_H

**Main.cpp**

#include "Sudoku.h"

#include <QApplication>

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

{

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

Sudoku w;

w.show();

return a.exec();

}

**Sudoku.cpp**

#include "Sudoku.h"

#include "ui\_Sudoku.h"

#include <QActionGroup>

#include <QMessageBox>

#include "Board.h"

Sudoku::**Sudoku**(QWidget \*parent) :

QMainWindow(*parent*),

ui(new Ui::Sudoku) {

ui->setupUi(this);

QActionGroup\* group = new QActionGroup(this);

group->setExclusive(true);

group->addAction(*ui->actionEasy*);

group->addAction(*ui->actionMedium*);

group->addAction(*ui->actionHard*);

QObject::connect(ui->actionNew, SIGNAL(triggered()), ui->board, SLOT(renew()));

QObject::connect(ui->actionClean, SIGNAL(triggered()), ui->board, SLOT(clear()));

QObject::connect(ui->actionExit, SIGNAL(triggered()), qApp, SLOT(quit()));

QObject::connect(group, SIGNAL(triggered(QAction\*)), this, SLOT(changeMode(QAction\*)));

QObject::connect(ui->board, SIGNAL(completed(double)), this, SLOT(showProgress(double)));

QObject::connect(ui->actionAbout, SIGNAL(triggered()), this, SLOT(showAbout()));

loadPlugins();

this->showProgress(0);

this->adjustSize();

this->setFixedSize(this->size());

}

Sudoku::~***Sudoku***() {

delete ui;

}

void Sudoku::**changeMode**(QAction\* action) {

DefaultBoards::Mode mode;

if (action == ui->actionEasy)

mode = DefaultBoards::Easy;

else if (action == ui->actionMedium)

mode = DefaultBoards::Medium;

else if (action == ui->actionHard)

mode = DefaultBoards::Hard;

else {

Q\_UNREACHABLE();

qApp->exit();

}

ui->board->setMode(mode);

}

void Sudoku::**showAbout**() {

QMessageBox::information(this, tr("О программе"), tr("Судоку\n\nПрокопчик Егор ПО-5"));

}

void Sudoku::**showProgress**(double progress) {

ui->statusBar->showMessage(QString("Прогресс: %1%").arg(QString::number(progress, 'f', 2)));

}

DefaultBoards.cpp

#include "DefaultBoards.h"

#include <ctime>

DefaultBoards\* DefaultBoards::m\_instance = 0;

QMutex& DefaultBoards::m\_mutex = \*(new QMutex());

DefaultBoards::**DefaultBoards**()

: QObject(0), m\_settings(":/boards", QSettings::IniFormat) {

std::srand(std::time(0));

}

DefaultBoards::~***DefaultBoards***() {

}

DefaultBoards::SimpleBoard DefaultBoards::**getBoard**(DefaultBoards::Mode mode) {

m\_settings.beginGroup(DefaultBoards::mode2str(mode));

QString boardStr = (mode == DefaultBoards::Test ?

m\_settings.value("board") :

m\_settings.value(QString("board%1").arg(std::rand() % BOARDS))).toString();

m\_settings.endGroup();

Q\_ASSERT\_X(boardStr.count() == (9\*9), "getBoard", "invalid board");

SimpleBoard board;

for (int l = 0; l < 9; l++) {

for (int c = 0; c < 9; c++) {

board.board[l][c] = boardStr.at((l \* 9) + c).digitValue();

}

}

return board;

}

DefaultBoards\* DefaultBoards::**instance**() {

if (!m\_instance) {

QMutexLocker locker(*&m\_mutex*);

Q\_UNUSED(locker);

if (!m\_instance)

m\_instance = new DefaultBoards();

}

return m\_instance;

}

QString DefaultBoards::**mode2str**(DefaultBoards::Mode mode) {

switch (mode) {

// case DefaultBoards::Test:

// return "Test";

case DefaultBoards::Easy:

return "Easy";

case DefaultBoards::Medium:

return "Medium";

case DefaultBoards::Hard:

return "Hard";

default:

Q\_UNREACHABLE();

}

}

**Position.cpp**

#include "Position.h"

Position::**Position**() : m\_row(0), m\_col(0) {

}

Position::**Position**(int row, int col) : m\_row(row), m\_col(col) {

}

Position::~***Position***() {

}

**Board.cpp**

#include "Board.h"

#include "ui\_Board.h"

#include <QMessageBox>

#include <algorithm>

#include "Cell.h"

Board::**Board**(QWidget \*parent) :

QWidget(*parent*), m\_mode(DefaultBoards::Easy), ui(new Ui::Board) {

ui->setupUi(this);

for (int r = 0; r < 9; r++) {

for (int c = 0; c < 9; c++) {

Cell\* cell = this->findChild<Cell\*>(QString("cell%1%2").arg(r+1).arg(c+1));

Q\_ASSERT\_X(cell, "board", "invalid cell");

Q\_ASSERT\_X(!m\_all.contains(cell), "board", "duplicated cell");

cell->setPosition(Position(r, c));

QObject::connect(cell, SIGNAL(valueChanged(int)), this, SLOT(updateValue()));

QObject::connect(cell, SIGNAL(valueChanged(int)), this, SLOT(checkBoard()));

m\_cells[r][c] = cell;

m\_all << cell;

}

}

QObject::connect(this, SIGNAL(modeChanged(DefaultBoards::Mode)), this, SLOT(renew()));

QObject::connect(this, SIGNAL(gameOver()), this, SLOT(showCongratulations()));

QMetaObject::invokeMethod(this, "renew");

}

Board::~***Board***() {

delete ui;

}

void Board::**renew**() {

this->clear();

DefaultBoards::SimpleBoard board =

DefaultBoards::instance()->getBoard(m\_mode/\*DefaultBoards::Test\*/);

for (int r = 0; r < 9; r++) {

for (int c = 0; c < 9; c++) {

int value = board.board[r][c];

m\_cells[r][c]->setValue(value);

m\_cells[r][c]->setEnabled(value == 0);

}

}

}

void Board::**clear**() {

for (int r = 0; r < 9; r++) {

for (int c = 0; c < 9; c++) {

Cell\* tmp = m\_cells[r][c];

if (tmp->isEnabled())

m\_cells[r][c]->setValue(0);

}

}

}

void Board::**setMode**(DefaultBoards::Mode mode) {

if (mode != m\_mode) {

m\_mode = mode;

emit modeChanged(mode);

}

}

bool Board::**checkLine**(Cell\* cell) const {

int r = cell->row();

for (int c = 0; c < 9; c++) {

Cell\* tmp = m\_cells[r][c];

if (this->checkCell(*cell*, *tmp*))

return false;

}

return true;

}

bool Board::**checkColumn**(Cell\* cell) const {

int c = cell->col();

for (int r = 0; r < 9; r++) {

Cell\* tmp = m\_cells[r][c];

if (this->checkCell(*cell*, *tmp*))

return false;

}

return true;

}

bool Board::**checkBox**(Cell\* cell) const {

int i = (cell->row() / 3) \* 3;

int j = (cell->col() / 3) \* 3;

for (int r = 0; r < 3; r++) {

for (int c = 0; c < 3; c++) {

Cell\* tmp = m\_cells[i+r][j+c];

if (this->checkCell(*cell*, *tmp*))

return false;

}

}

return true;

}

bool Board::**checkCell**(Cell\* orig, Cell\* other) const {

return (orig != other &&

!other->isInvalid() &&

orig->value() == other->value());

}

void Board::**checkBoard**() {

struct {

int valid;

int invalid;

} count = { 0, 0 };

for (int r = 0; r < 9; r++) {

for (int c = 0; c < 9; c++) {

if (m\_cells[r][c]->isInvalid())

count.invalid++;

else if (m\_cells[r][c]->isEnabled())

count.valid++;

}

}

emit completed((count.valid \* 100.0) / (count.valid + count.invalid));

if (count.invalid == 0)

emit gameOver();

}

void Board::**updateValue**() {

Cell\* cell = qobject\_cast<Cell\*>(QObject*::sender()*);

if (!cell)

return;

if (cell->value() == 0) {

cell->setInvalid(true);

return;

}

bool ok = checkLine(*cell*) && checkColumn(*cell*) && checkBox(*cell*);

cell->setInvalid(!ok);

}

void Board::**showCongratulations**() {

QMessageBox::information(

this,

tr("Конец игры"),

tr("Поздравляю, вы собрали доску"));

}

**Cell.cpp**

#include <QStyle>

#include "Cell.h"

Cell::**Cell**(QWidget \*parent) : QSpinBox(*parent*), m\_invalid(true) {

this->setSpecialValueText(" ");

QObject::connect(this, SIGNAL(invalidChanged(bool)), this, SLOT(repolish()));

}

Cell::**Cell**(const Position& position, QWidget \*parent)

: QSpinBox(*parent*), m\_position(position), m\_invalid(true) {

this->setSpecialValueText(" ");

QObject::connect(this, SIGNAL(invalidChanged(bool)), this, SLOT(repolish()));

}

Cell::~***Cell***() {

}

void Cell::**setPosition**(const Position position) {

m\_position = position;

}

void Cell::**setInvalid**(bool invalid) {

if (m\_invalid != invalid) {

m\_invalid = invalid;

emit invalidChanged(invalid);

}

}

void Cell::**repolish**() {

this->style()->*unpolish*(this);

this->style()->*polish*(this);

this->update();

}

**Результат:**

****

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