Санкт-Петербургский Политехнический Государственный Университет

факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

## Курсовая работа по курсу «Программирование»

## Тема работы: «Разработка приложения с графическим интерфейсом на языке C++ с использованием библиотеки QT»

Работу выполнил:

студентка 1 курса

группы 1081/3

Назарова К. Е.

Работу принял:

Пышкин Е. В.

Санкт-Петербург 2012

## Образ и границы проекта

Целью данного проекта является разработка проекта с графическим интерфейсом на языке C++ на примере игры «*Sunny Seeds*».

Суть игры заключается в том, чтобы очистить поле от всех чисел.

Правила игры:

1. На входе таблица случайных чисел от 1 до 9;
2. Требуется найти пары одинаковых чисел или чисел, сумма которых равна 10, в случае чего они исчезнут;
3. Числа, которые можно удалить, должны располагаться на одном уровне по горизонтали или вертикали, при этом между ними не должно быть других чисел;
4. При необходимости можно добавить новую группу чисел, состоящую из оставшихся на поле цифр (это дает гарантию выигрыша, но для того, чтобы игра не была бесконечной, доступное количество добавлений будет ограничено).

## Техническое задание

Разработать приложение на языке C++ с использованием библиотеки Qt для реализации графического интерфейса в двух режимах:

1. Консольное приложение, предназначенное для тестирования и проверки правильности работы приложения;
2. Интеграция кода программы для консоли с графической библиотекой Qt, и создание графического интерфейса.

Основные требования

Меню приложения:

1. Новая игра;
2. Продолжить сохраненную игру;
3. Выход.

Сценарии выполнения программы

1. Новая игра

При выборе этого пункта перед пользователем появляется игровое поле;

1. Продолжить сохраненную игру

При нажатии на эту кнопку меню загрузится последняя сохраненная игра (данные хранятся в текстовом формате в документе Save.txt);

1. Выход

Выход из приложения. При этом все данные незаконченной игры будут потеряны.

Требования к графическому интерфейсу

1. На входе стартовое меню в виде кнопок;
2. Удаление пар чисел и выбор других действий производится посредством нажатия левой кнопки мыши.

## Системная архитектура

В консольном приложении используются следующие классы:

1. Игровое поле. Этот класс создает вектор, состоящий из элементов класса *Row*. Размер поля зависит от выбранного разработчиком числа. По мере необходимости добавляет строки с требуемыми значениями.

class Field{

int rows;

bool saveField;

public:

Field();

Field(int a);

~Field();

vector<Row>field;

bool deleteCell(int, int, int, int);

void deleteEmptyRow();

void addCells();

void getState(bool);

void saveGame();

void loadGame();

int getRows();

};

1. Строка. Этот класс хранит в себе строку из 7 элементов, которая заполняется случайными числами.

class Row{

vector<int> arr;

vector<bool> isPressed;

public:

Row();

int getData(int);

void setData(int number, int value);

void clearCell(int a);

void setStateIsPressed(int, bool);

bool getStateIsPressed(int);

};

1. Проверка хода. С помощью методов этого класса в ходе игры проверяются все ходы, сделанные игроком, а так же проверяется конец игры.

class Check{

Field f;

bool checkSameValues(int, int, int, int);

bool checkSumValues(int, int, int, int);

bool checkPlace(int, int, int, int);

bool nullValue(int, int, int, int);

bool checkSamePlace(int, int, int, int);

bool checkLoss();

bool checkWin();

public:

Check();

Check(Field&);

bool checkMove(int, int, int, int);

bool getLoss();

bool getWin();

};

1. Тест. Класс, тестирующий некоторые функции других классов.

class Test{

public:

Test();

bool nearSameNumber();

bool emptyOrFilledCell();

bool sameNumThroughEmptyCell();

};

В графической реализации программы добавляется еще один класс, отвечающий за представление игры:

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

protected:

bool firstClick;

int startX, startY, finishX, finishY;

virtual void *paintEvent*(QPaintEvent \*);

void *mousePressEvent*(QMouseEvent \*event);

void leftButtonPressEvent(QMouseEvent\* event);

void *closeEvent*(QCloseEvent \*event);

bool outLoss();

bool outWin();

private slots:

void on\_actionNew\_game\_activated();

void on\_pushButton\_clicked();

void on\_actionExit\_activated();

void on\_verticalScrollBar\_valueChanged(int value);

void on\_actionContinued\_activated();

private:

Ui::MainWindow \*ui;

Field f;

int myVar;

};

Тестирование работы приложения

Файл test.h

#ifndef TEST\_H

#define TEST\_H

#include "field.h"

class Test{

public:

Test();

bool nearSameNumber();

bool emptyOrFilledCell();

bool sameNumThroughEmptyCell();

};

#endif // TEST\_H

Файл test.cpp

#include "test.h"

Test::Test()

{

if (nearSameNumber())

cout << "Test #1 passed."<<endl;

else cout << "Test #1 didn't pass."<<endl;

cout<<endl;

if (emptyOrFilledCell())

cout << "Test #2 passed."<<endl;

else cout << "Test #2 didn't pass."<<endl;

cout<<endl;

if (sameNumThroughEmptyCell())

cout << "Test #3 passed."<<endl;

else cout << "Test #3 didn't pass."<<endl;

cout<<endl;

}

bool Test::nearSameNumber(){

cout<<"Test #1. The choice of identical adjacent numbers."<<endl;

Field f(3);

cout<<f<<endl;

f.field[0].setData(0, 1);

f.field[0].setData(1, 1);

cout<<f<<endl;

return f.deleteCell(1, 1, 1, 2);

}

bool Test::emptyOrFilledCell(){

cout<<"Test #2. The choice of empty and filled cell."<<endl;

Field c(3);

cout<<c<<endl;

c.field[0].setData(0, 0);

c.field[0].setData(1, 1);

cout<<c<<endl;

return c.deleteCell(1, 1, 1, 2);

}

bool Test::sameNumThroughEmptyCell(){

cout<<"Test #3. The choice of identical numbers, among which is an empty cell."<<endl;

Field f(3);

cout<<f<<endl;

f.field[2].setData(4, 1);

f.field[1].setData(4, 0);

f.field[0].setData(4, 1);

cout<<f<<endl;

return f.deleteCell(3, 5, 1, 5);

}

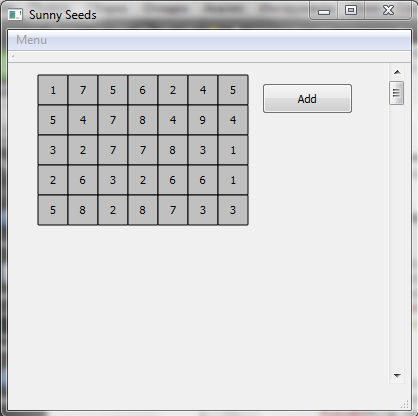
Первый тест заключается в том, чтобы проверить рядом стоящие одинаковые числа. Создается поле 3х7, в котором в ячейках с координатами [0, 0] и [0, 1] стоят единицы. Выбирается эта пара цифр, затем удаляется. Следовательно, тест пройден, так как соблюдены все правила.

Второй тест заключается в том, чтобы проверить рядом стоящие не одинаковые числа. Создается поле 3х7, в котором ячейках с координатами [0, 0] пустая, а в ячейке [0, 1] стоит единица. Выбирается эта пара ячеек, затем выходит сообщение, что тест не пройден, так как выбраны неверные значения.

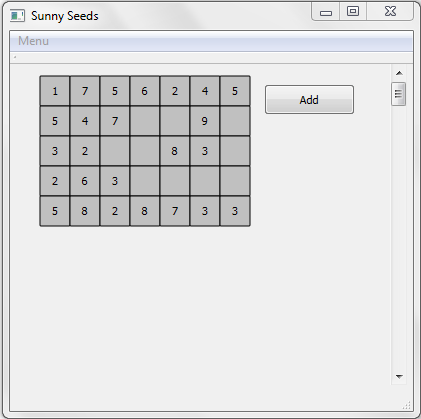
Суть третьего теста в том, чтобы проверить одинаковые числа, между которыми стоят пустые ячейки. Создается поле 3х7, в котором в ячейках с координатами [2, 4] и [0, 4] стоят единицы, а ячейка с координатами [1, 4] пустая. Выбирается пара, затем удаляется. Следовательно, тест пройден, так как соблюдены все правила.

## Описание испытаний графического интерфейса

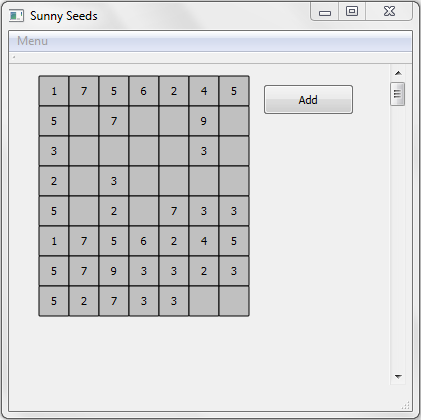
Начальное окно приложения



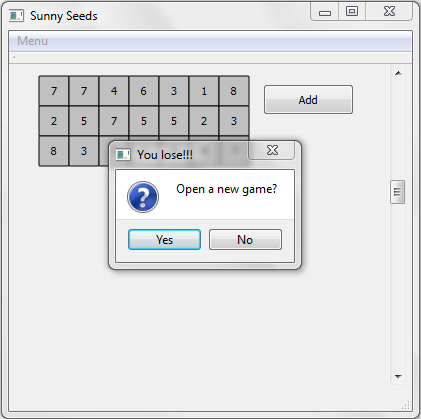
Поле после удаления некоторых ячеек



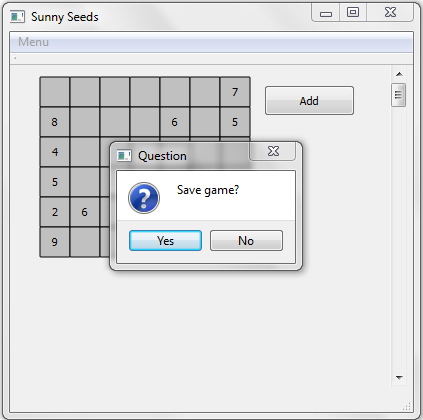
Поле после добавления не нулевых ячеек (нажатие на кнопку “Add”)



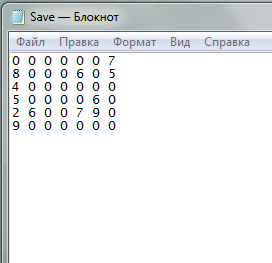
Состояние проигрыша



Выход из игры и предложение сохранить текущее состояние



Данные сохраняются в текстовом документе Save.txt в виде двумерного массива



## Вывод

Благодаря выполнению этого курсового проекта, я узнала, какие возможности представляет QtCreator, освоила некоторые функции и библиотеки, использованные для написания программы с графическим интерфейсом. В ходе работы возникали проблемы с созданием игрового поля, но они были решены посредством использования контейнерного класса *vector*. В дальнейшем планируется некоторая доработка программы, в частности, добавление возможности подсчета заработанных очков, следовательно, и рейтинговой таблицы.