**Санкт-Петербургский государственный политехнический университет**

**Факультет технической кибернетики**

**Кафедра компьютерных систем и программных технологий**

**Курсовая работа по курсу «Программирование»**

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

**Выполнил: студент Киселёв Антон,**

**Группа 1081/3**

**Преподаватель: доцент Пышкин Е.В.**

**Санкт-Петербург, 2012**

1. **Описание и задачи проекта:**

**В качестве практического задания я решил взять игру «Судоку». Смысл игры в заполнении пустых клеток квадратной матрицы чисел 9 на 9. В игре предусмотрено несколько различных уровней сложностей и различные варианты игрового поля. Разрабатывая данный проект, передо мной стоит несколько задач.**

**Во-первых, сгенерировать игровое поле; для этого я формировал двумерный массив 9 на 9, который заполнял случайными числами, с учетом необходимых требований строки, столбца и квадрата – объектов поля, которые и определяют условия заполнения поля случайными числами. Данной генерации мне не удалось создать. Поэтому это одна из первых главных задач, стоящих передо мной.**

**Во-вторых, разработать игровую механику. Я создал небольшое меню и в игре установил порядок заполнения поля игроком и проверку поля на ошибки, которые может совершать игрок. Но сейчас мне необходимо будет реализовать игровую механику до более высокого уровня взаимодействия с пользователем, а именно приблизить игру к реальной.**

**В третьих, сделать игру в какой-то степени играбельной. Я предусмотрел возможность игрока самому проверять свои действия на ошибки в любой момент, но данной операции мне не удалось реализовать, так как игровое поле мне не удалось сделать соответствующим для требований настоящей игры.**

1. **Текст начатой в первом семестре программы:  
   а) Основной модуль.**

#include <iostream>

#include <fstream>

#include <locale.h>

#include "Field.h"

using namespace std;

void main(void)

{

setlocale(LC\_ALL, "Russian");

cout << "Игровое меню:\n";

cout << "1. Играть\n";

cout << "2. Выйти\n";

int choice;

cin >> choice;

switch(choice)

{

case 1:

{

Field GameField;

GameField.OperationField();

GameField.DefineVictory();

GameField.OutOfCorrectLocation();

}

case 2:

return;

}

return;

}  
**б) Объект поле.**  
 #pragma once

class Field

{

public:

int gamefield[ 9 ][ 9 ];//Поле 9х9

bool CorrectLocation[ 9 ][ 9 ];//Поле 9х9, в нем будут записываться ошибки по тем значениям, которые вставил игрок

Field(void);

void OperationField();

void OutOfField();

void DefineVictory();

void OutOfCorrectLocation();

bool ControlChislo(int& );

~Field(void);

};

#include <iostream>

#include <fstream>

#include <stdio.h>

#include "Field.h"

#include "Line.h"

#include "Column.h"

#include "Square.h"

using namespace std;

Field::Field(void)

{

int ixRow = 0;

int ixCol = 0;

ifstream input("input.txt");

while (!input.eof())

{

if (ixCol == 9)

{

cout << "(" << ixRow+1 << ")\n";

ixRow++;

ixCol = 0;

cout << endl;

}

input >> gamefield[ ixRow ][ ixCol ];

cout << " " << gamefield[ ixRow ][ ixCol ] << " ";

ixCol++;

}

cout << "\n";

for (ixRow = 0; ixRow < 9; ixRow++)

cout << "(" << ixRow+1 << ")";

cout << "\n";

}

void Field::OperationField()

{

for (int ixRow = 0; ixRow < 9; ixRow++)

for (int ixCol = 0; ixCol < 9; ixCol++)

{

m1:

if (gamefield[ ixRow ][ ixCol ] == 0)

{

int x = 0;

int y = 0;

int chislo = 0;

cout << "Введите координаты и число, которое вы хотите вставить\n";

cin >> x >> y >> chislo;

x = x-1;

y = y-1;

gamefield[ x ][ y ] = chislo;

OutOfField();

goto m1;

}

}

return;

}

void Field::OutOfField()

{

for (int ixRow = 0; ixRow < 9; ixRow++)

{

for (int ixCol = 0; ixCol < 9; ixCol++)

cout << " " << gamefield[ ixRow ][ ixCol ] << " ";

cout << "(" << ixRow+1 << ")";

cout << "\n";

}

for (int ixRow = 0; ixRow < 9; ixRow++)

cout << "(" << ixRow+1 << ")";

cout << "\n";

}

void Field::DefineVictory()

{

int sum = 0;

int count = 0;

for (int ixRow = 0; ixRow < 9; ixRow++)

{

for (int ixCol = 0; ixCol < 9; ixCol++)

{

if (CorrectLocation[ ixRow ][ ixCol ] == 1)

count++;

sum = sum + count;

}

}

if (sum == 81)

cout << "Вы выиграли!";

}

void Field::OutOfCorrectLocation()

{

for (int ixRow = 0; ixRow < 9; ixRow++)

{

for (int ixCol = 0; ixCol < 9; ixCol++)

{

cout << " " << CorrectLocation[ ixRow ][ ixCol ] << " ";

}

cout << "(" << ixRow+1 << ")\n\n";

}

for (int ixRow = 0; ixRow < 9; ixRow++)

cout << "(" << ixRow+1 << ") ";

cout << "\n";

}

bool Field::ControlChislo(int& chislo)

{

int ixRow = 0;

int ixCol = 0;

bool A, B, C;

Line Stroka(gamefield, ixRow);

Column Stolb(gamefield, ixCol);

Square Square(gamefield, ixRow, ixCol);

A = Stroka.ControlLine(chislo);

B = Stolb.ControlColumn(chislo);

C = Square.ControlSquare(chislo);

if (A == 1 && B == 1 && C == 1)

{

CorrectLocation[ ixRow ][ ixCol ] = 1;

return true;

}

else if (A == 0 || B == 0 || C == 0)

{

CorrectLocation[ ixRow ][ ixCol ] = 0;

cout << "Введённое значение неверно\n";

return false;

}

}

Field::~Field(void)

{

}

**в) Объект строка.**#pragma once

#include "Field.h"

class Line

{

static const int width = 9;//Ширина поля

public:

int line[ width ];//Объект строка, выводится из объекта поле

int NumberOfLine;//Номер строки, определяется из поля

Line(int gamefield[ 9 ][ 9 ], int& NumberOfLine);//Конструктор строки

bool ControlLine(int& variable);//Функция, определяющая правильность вставленного значения в строку

~Line();//Деструктор

};

#include "Line.h"

#include "Field.h"

Line::Line(int gamefield[ 9 ][ 9 ], int& NumberOfLine)

{

for(int k = 0; k < width; k++)

line[ k ] = gamefield[ NumberOfLine ][ k ];

}

bool Line::ControlLine(int& variable)

{

int index = 0;

for (index = 0; index < 9; index++)

{

if (line[ index ] == variable) return false;

}

return true;

}

Line::~Line()

{

}  
**г) Объект столбец.**  
#pragma once

#include "Field.h"

class Column

{

static const int high = 9;//Высота поля

public:

int column[ high ];//Объект столбец, выделяется из объекта поля

int NumberOfColumn;//Номер столбца, определяется из поля

Column(int gamefield[ 9 ][ 9 ], int& NumberOfColumn);//Конструктор столбца

bool ControlColumn(int& variable);//Функция, определяющая правильность вставленного значения в столбец

~Column();//Деструктор

};

#include "Column.h"

#include "Field.h"

Column::Column(int gamefield[ 9 ][ 9 ], int& NumberOfColumn)

{

for(int k = 0; k < high; k++)

column[ k ] = gamefield[ k ][ NumberOfColumn ];

}

bool Column::ControlColumn(int& variable)

{

int index = 0;

for (index = 0; index < 9; index++)

{

if (column[ index ] == variable) return false;

}

return true;

}

Column::~Column()

{

}  
**д) Объект квадрат.**

#pragma once

#include "Field.h"

class Square

{

public:

int square[ 3 ][ 3 ];//Объект квадрат, выводится из объекта поле

//Координаты точки поля, служат для выделения определенного объекта квадрат из объекта поля

int x;

int y;

Square(int gamefield[ 9 ][ 9 ], int& ixRow, int& ixCol);//Конструктор квадрата

bool ControlSquare(int& variable);//Функция, определяющая правильность вставленного значения в квадрат

~Square();//Деструктор

};

#include "Square.h"

#include "Field.h"

Square::Square(int gamefield[ 9 ][ 9 ], int& ixRow, int& ixCol)//Определяем квадрат 3х3 из игрового поля

{

int xmax = 0;

int ymax = 0;

if(ixRow < 3 && ixCol < 3)

{

ymax = 0;

xmax = 0;

}

else if(ixRow < 3 && ixCol < 6 && ixCol > 2)

{

ymax = 3;

xmax = 0;

}

else if(ixRow < 3 && ixCol < 9 && ixCol > 5)

{

ymax = 6;

xmax = 0;

}

else if(ixRow > 2 && ixRow < 6 && ixCol < 3)

{

ymax = 0;

xmax = 3;

}

else if(ixRow > 2 && ixRow < 6 && ixCol < 6 && ixCol > 2)

{

ymax = 3;

xmax = 3;

}

else if(ixRow > 2 && ixRow < 6 && ixCol < 9 && ixCol > 5)

{

ymax = 6;

xmax = 3;

}

else if(ixRow > 5 && ixRow < 9 && ixCol < 3)

{

ymax = 0;

xmax = 6;

}

else if(ixRow > 5 && ixRow < 9 && ixCol < 6 && ixCol > 2)

{

ymax = 3;

xmax = 6;

}

else if(ixRow > 5 && ixRow < 9 && ixCol < 9 && ixCol > 5)

{

ymax = 6;

xmax = 6;

}

this->x = xmax;

this->y = ymax;

int CurrentIxRow = 0;

int CurrentIxCol = 0;

for(CurrentIxRow = xmax; CurrentIxRow < xmax+3; CurrentIxRow++)

for(CurrentIxCol = ymax; CurrentIxCol < ymax+3;CurrentIxCol++)

square[ CurrentIxRow-xmax ][ CurrentIxCol-ymax ] = gamefield[ xmax ][ ymax ];

}

bool Square::ControlSquare(int& variable)

{

int CurrentIxRow = 0;

int CurrentIxCol = 0;

for (CurrentIxRow = 0; CurrentIxRow < 3; CurrentIxRow++)

for (CurrentIxCol = 0; CurrentIxCol < 3; CurrentIxCol++)

if (square[ CurrentIxRow ][ CurrentIxCol ] == variable) return false;

return true;

}

Square::~Square()

{

}

1. **Описание игры:**

**Игра состоит из нескольких модулей, 4 класса на 2 модулях каждый, главный модуль, объединяющий все, и текстового файла, предназначенный в данный момент для хранения готового поля игры. Сначала формируется объект поле, сразу заполняющийся значениями из текстового файла. Затем игрок вправе заменять пустые клетки (заполненные нулями) на цифры, которые он считает правильными. В момент заполнения для каждого значения выделяется соответствующие объекты строки, столбец и квадрат, для которых проверяется правильность вставленного значения. Если значение вставлено верно, то в массив ошибок – CorrectLocation – помещается значение 1, если неверно, то – 0. В конце программа выведет данный массив ошибок и попросит их исправить.**

1. **Планируемые нововведения и улучшения в программе на втором семестре:**
2. **Сформировать игровое поле для различных типов генерации с учетом возможных уровней сложности – поле останется прежним 9х9.**
3. **Доработать игровую механику, учитывая прежние недостатки и приблизив игру к реальной. Создать игровое меню, меню выбора уровня сложности, методы взаимодействия с пользователем (кнопки, панель и т.д.). Возможность игрока сразу проверять свои ошибки, сделанные в поле игры.**
4. **Разработать графический интерфейс к программе, чтобы сделать её более воспринимаемой пользователем.  
   Для этого я обдумываю создание дополнительного объекта – страница, на которое будет помещаться игровое меню, поле и другие нововведения (кнопки, курсоры). Для реализации графики я буду использовать библиотеку Qt.**
5. **Перспективные нововведения:**

**Игра судоку не является однотипной, существует множество различных её модификаций (супер судоку, судоку мишень, судоку блоки, японское судоку и другие), мне хотелось бы реализовать и некоторые другие её типы.**