Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

Курсовая работа по курсу «Программирование».

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

Студентка группы 1081/3 Никитина Анна

Преподаватель Пышкин Е.В.

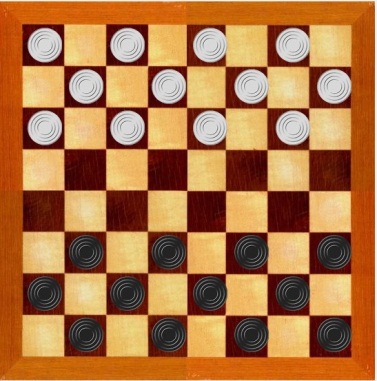
Санкт-Петербург 2012

**Образ и границы проекта.**

Целью данного проекта является написание программы, реализующую игру «шашки».

В игре должны быть реализованы стандартные правила:

## Игровое поле. Начальная позиция



Игровое поле представляет собой квадратную доску, подобную шахматной. В отличие от шахмат игровыми считаются не все, а только темные поля. Каждая из сторон в начале игры имеет одинаковое количество шашек. Шашки расставляются на ближайших к игрокам, горизонталях. Два центральных ряда полей остаются свободными. Здесь, на этих полях, происходит сближение и первое соприкосновение противостоящих сил. Противники ходят поочередно, перемещая шашки своего цвета по игровым полям. Первыми начинают белые. Возникающие на доске ситуации называют позициями, или положениями. Проведенная от начала и до завершения игра называется партией, а передвижения шашек - ходами. Выигрывает та сторона, которой удалось уничтожить все шашки противника.

## Простые шашки.

В начале игры все шашки – простые.

D:\practice 1081\trunk\NikitinaAnna\source\QT\blackshashka.png D:\practice 1081\trunk\NikitinaAnna\source\QT\whiteshashka.png

Простая шашка может совершать, в зависимости от позиции, ходы типов: тихие и ударные.

* Тихий ход - перемещение на одно поле вперед по диагонали. Тихий ход возможен, если соответствующее поле свободно.
* Ударный ход (бой шашки противника) - перемещение на два поля вперед по диагонали, через шашку противника. Шашка при своем ходе может побить (взять) шашку противника, если та находится на соседнем (по диагонали) поле и если следующее за ней поле свободно. Берущая шашка становится на это свободное поле, перескакивая через вражескую шашку, которая снимается с доски.

Если после взятия одной шашки оказывается возможным побить еще одну шашку противника, взятие продолжается.

Дамки.

D:\practice 1081\trunk\NikitinaAnna\source\QT\blackking.png D:\practice 1081\trunk\NikitinaAnna\source\QT\whiteking.png

Если простая шашка достигает последней горизонтали, она становится дамкой. Дамки, в отличие от обычных шашек, способны перемещаться на произвольное количество полей по выбранной диагонали, причем как вперед, так и назад. Дамка при своем ходе может взять шашку соперника (как вперед, так и назад) независимо от количества свободных полей до нее, если только эта шашка находится   
на одной диагонали с дамкой и за этой шашкой имеется одно или несколько свободных полей.

# Техническое задание.

Приложение должно быть реализовано на языке программирования C++.В дальнейшем предстоит реализовать графический интерфейс с использованием библиотеки QT. Игроку, для того чтобы сделать ход, надо будет просто нажать мышкой сначала на шашку, которая должна ходить, а потом на пустую клетку (т. е. игрок сможет играть только с помощью мыши, без клавиатуры)*.*

## Режимы работы приложения.

Реализуется два режима работы этапа:

1. Консольное приложение без графического интерфейса. В этом режиме обеспечивается проверка бизнес-логики.
2. Интеграция кода программы для консоли с графической библиотекой Qt, и создание графического интерфейса.

## Основной сценарий функционирования приложения.

1. Определяются переменные и константы. Создается игровое поле, заполняется оно строго определенным образом.
2. Выполняется цикл, который продолжается до завершения игры.
3. Игрок ходит. Проверка правильности хода.
4. Проверка продолжения игры. Если игра продолжается, определяется: чей ход.
5. Определяется: можно ли компьютеру сходить так, чтобы «съесть» шашку игрока. Если нет, компьютер делает такой ход, чтобы его фигуру нельзя было «съесть». Если при любом ходе компьютера его фигуру нельзя съесть или, наоборот, при любом ходе фигура компьютера будет съедена, то ход совершается случайным образом.
6. Проверка продолжения игры. Если игра продолжается, определяется: чей ход.
7. Если определен победитель или все шашки съедены (ничья) выводится информация о победителе. Игра закончена.

## Требования к графическому пользовательскому интерфейсу.

1. Экран с пунктами меню в виде кнопок.
2. После нажатия пользователем кнопки «Начать игру» на экране появляется игровое поле с расставленными в нужном порядке шашками. Для совершения хода пользователю надо будет нажать на шашку, которую он хочет переместить, а потом на ячейку, на которую пользователь хочет поставить шашку. Так до завершения игры.
3. После завершении игры появляется окошко, которое сообщает о конце игры и либо о победителе, либо о ничье.
4. Предполагается обработка ошибок. Если игрок попытается сделать неверный ход, выйдет окно, в котором будет написано о неверности данного хода. Для этого используется специальный алгоритм обработки хода,который устроен следующим образом:

* Считываются координаты первого и второго щелчка мыши.
* Координаты курсора преобразовываются в координаты клетки и вызывается метод класса Game «result», который в свою очередь вызывает метод «permutation» или «eatenChecker»(в зависимости от того совершает фигура обычный ход или ход, сопровождающийся съедением фигуры противника) класса Player.
* Проверяет корректно ли указаны начальные и конечные координаты, точнее не выходят ли они за рамки игрового поля.
* Затем проверяется находится ли на стартовой и на конечной клетке фигура
* Далее проверяется нужно ли превращать какую-либо шашку в дамку.
* Проверка того, может ли шашка совершать второй ход подряд, сопровождающийся съедением шашки противника
* Если это надо меняем игрока, который должен совершать ход
* Возвращается значение true
* Если ход не может быть совершён возвращается значение false

Этот цикл из ходов продолжается до тех пор, пока не будет определен победитель(пока winner==empty).

После совершения каждого хода производится перерисовка доски, пока игра не будет закончена.

Когда это произойдет, выйдет окошко, в котором будем написано, кто победил.

## Системная архитектура.

В программе предполагается использовать следующие классы:

1. Класс ячейка.

Ячейка может находиться в пяти состояниях (на ней стоит белая фигура (дамка или обычная шашка), черная (дамка или обычная шашка), или она пустая), для этого предполагается использовать перечисления. Методы: для каждого вида фигуры определяются правила, по которым фигура должна делать ход.

1. Класс игровое поле.

Методы: заполнение игрового поля, превращение шашки в дамку, проверка окончания игры.

1. Класс Игрок.

Методы: осуществление хода, проверка того, возможен ли подряд второй ход того же игрока.

1. Class Player

Заголовочный фаил:

#ifndef \_player\_h\_

#define \_player\_h\_

#include"field.h"

class Player

{

int opponentX;

int opponentY;

public:

Player();

~Player();

bool permutation(int Px,int Py, int x, int y,Field& field);//Реализация обычного хода

bool testEatenChecker(int Px,int Py, int x, int y,Field& field);//Проверка:возможно ли данной шашке или дамке "съесть" шашку противника

void eatenChecker(int Px,int Py, int x, int y,Field& field);//Реализация хода при съедении шашки врага

bool secondCourse(int Px,int Py,Field& field);//Реализация следующего хода той же шашки после того, как она "съела" шашку противника

};

#endif

void Player::eatenChecker(int Px,int Py, int x, int y,Field& field )//Реализация хода при съедении шашки врага

{

int i,j;

if(field.gameField[x][y]==blackShashka || field.gameField[x][y]==whiteShashka)

{

i=x;j=y;

if((field.gameField[x][y]==whiteShashka && ((Py-y==2 && (field.gameField[++i][++j]==blackShashka || field.gameField[++i][++j]==blackKing))

||(y-Py==2 && (field.gameField[++i][--j]==blackShashka || field.gameField[++i][--j]==blackKing))))

||(field.gameField[x][y]==blackShashka && ((Py-y==2 && (field.gameField[--i][++j]==whiteShashka || field.gameField[--i][++j]==whiteKing)) ||

(y-Py==2 && (field.gameField[--i][--j]==whiteShashka || field.gameField[--i][--j]==whiteKing)))))//если на заданной ячейки белая шашка,а на той,которой мы хотим "съесть",стоит черная шашка

{

field.gameField[Px][Py]=field.gameField[x][y];

field.gameField[i][j]=empty;

field.gameField[x][y]=empty;//Совершаем заданные преобразования поля

}

}

if(field.gameField[x][y]==blackKing || field.gameField[x][y]==whiteKing)//если на заданной ячейке какая-либо дамка

{

field.gameField[opponentX][opponentY]=empty;

field.gameField[Px][Py]=field.gameField[x][y];

field.gameField[x][y]=empty;//совершаем заданные преобразования поля

}

}

bool Player::secondCourse(int Px,int Py,Field& field )//Реализация следующего хода той же шашки после того, как она "съела" шашку противника

{

int i=Px; int j=Py;

if(i<size-2 && j<size-2 && j>1)

{

if(field.gameField[Px][Py]==whiteShashka//если предыдущий ход был совершен белой шашкой

&&(((field.gameField[++i][++j]==blackShashka || field.gameField[i][j]==blackKing) && field.gameField[++i][++j]==empty)

|| ((field.gameField[--i][j=j-3]==blackShashka || field.gameField[i][j]==blackKing) && field.gameField[++i][--j]==empty)))

return true;

}

i=Px; j=Py;

if(i>1 && j<size-2 && j>1)

{

if(field.gameField[Px][Py]==blackShashka//если предыдущий ход был совершен черной шашкой

&&(((field.gameField[--i][++j]==whiteShashka || field.gameField[i][j]==whiteKing) && field.gameField[--i][++j]==empty)

|| ((field.gameField[++i][j=j-3]==whiteShashka || field.gameField[i][j]==whiteKing) && field.gameField[--i][--j]==empty)))

return true;

}

if(field.gameField[Px][Py]==whiteKing || field.gameField[Px][Py]==blackKing)

while(i<size && i>0 && j<size && j>0)

{

i=Px+1; j=Py+1;//проверка всевозможных направлений

while(field.gameField[i][j]==empty)//"идем" по диагоналям

{

i++;j++;

}

if(((field.gameField[i][j]==blackShashka || field.gameField[i][j]==blackKing ) && field.gameField[++i][++j]==empty && field.gameField[Px][Py]==whiteKing)//если на диагонали будет стоять черная фигура,а следующая после нее ячейка пустая

||(field.gameField[--i][--j]==whiteShashka || field.gameField[i][j]==whiteKing ) && field.gameField[++i][++j]==empty && field.gameField[Px][Py]==blackKing)

return true;//второй ход возможен

i=Px+1;j=Py-1;//проверка другой диагонали

while(field.gameField[i][j]==empty)

{

i++;j--;

}

if(((field.gameField[i][j]==blackShashka || field.gameField[i][j]==blackKing ) && field.gameField[++i][--j]==empty && field.gameField[Px][Py]==whiteKing)

||(field.gameField[--i][++j]==whiteShashka || field.gameField[i][j]==whiteKing ) && field.gameField[++i][--j]==empty && field.gameField[Px][Py]==blackKing)

return true;

i=Px-1;j=Py-1;

while(field.gameField[i][j]==empty)

{

i--;j--;

}

if(((field.gameField[i][j]==blackShashka || field.gameField[i][j]==blackKing ) && field.gameField[--i][--j]==empty && field.gameField[Px][Py]==whiteKing)

||(field.gameField[++i][++j]==whiteShashka || field.gameField[i][j]==whiteKing ) && field.gameField[--i][--j]==empty && field.gameField[Px][Py]==blackKing)

return true;

i=Px-1;j=Py+1;

while(field.gameField[i][j]==empty)

{

i--;j++;

}

if(((field.gameField[i][j]==blackShashka || field.gameField[i][j]==blackKing ) && field.gameField[--i][++j]==empty && field.gameField[Px][Py]==whiteKing)

||(field.gameField[++i][--j]==whiteShashka || field.gameField[i][j]==whiteKing ) && field.gameField[--i][++j]==empty && field.gameField[Px][Py]==blackKing)

return true;

}

return false;

}

bool Player::permutation(int Px,int Py,int x, int y,Field& field)//Реализация обычного хода

{

if((Px-x==1 && field.gameField[x][y]==whiteShashka) || (x-Px==1 && field.gameField[x][y]==blackShashka))//Ход на соседнюю ячейку шашки

{

field.gameField[Px][Py]=field.gameField[x][y];

field.gameField[x][y]=empty;

return true;

}

if((Px-x==y-Py || x-Px==y-Py )&& (field.gameField[x][y]==whiteKing || field.gameField[x][y]==blackKing))//Ход дамки не на соседнюю ячейку

{

int i=x;int j=y;

do

{

if(Px>x) i++;

else i--;

if(Py>y) j++;

else j--;

}

while(field.gameField[i][j]==empty && i!=Px && j!=Py);//"Идем" по диагонали и проверяем все ячейки на пустоту

if(i==Px && j==Py)//если все ячейки пустые

{

field.gameField[Px][Py]=field.gameField[x][y];//совершаем нужные преобразования поля

field.gameField[x][y]=empty;

return true;

}

}

return false;

}

1. Класс Игра.

Используется перечисление: выиграть может игрок, играющий белыми, черными, или игра может закончиться ничьей. Методы: определение победителя, чей ход.

Class Game

Заголовочный фаил:

#ifndef \_game\_h\_

#define \_game\_h\_

#include <QtGui>

#include"player.h"

#include"field.h"

#include"shashki.h"

enum View

{

black,//Игрок, играющий черными

white,//Игрок, играющий белыми

blank//Ничья

};

class Game/\*:public QObject\*/

{

// Q\_OBJECT

public:

Game();

~ Game();

void findWinner(Field& field);//проверка завершения игры и определение победителя

View winner;//Кто выиграл

bool isWhiteTurn;//Кто на данный момент ходит

bool result(int Px,int Py,int x, int y,int variable,Field& field,Checker& checker, King &king,Player &player);

/\*signals:

void end();

void printFalseTurn();\*/

};

#endif

#include"game.h"

#include"field.h"

#include"player.h"

#include"shashki.h"

Game::Game()

{

View winner=blank;//Первоначально состояние-ничья

isWhiteTurn=true;//Первый ход совершает игрок, играющий белыми

}

Game::~Game()

{

}

bool Game::result(int Px,int Py,int x, int y,int variable,Field& field,Checker& checker, King &king,Player &player)

{

if(((isWhiteTurn==true && field.gameField[x][y]==whiteShashka) ||

(isWhiteTurn==false && field.gameField[x][y]==blackShashka))//если сейчас должен ходить игрок, играющий белыми(черными), и шашка выбрана так же белая(черная)

&&(field.gameField[Px][Py]==empty))

{

checker.setx(x);

checker.sety(y);

if ((field.gameField[x][y]==whiteShashka && checker.*makeMove1*(Px,Py)==true)

|| (field.gameField[x][y]==blackShashka && checker.*makeMove2*(Px,Py)==true))//если на ячейке белая(черная) шашка и возможен ход ее на заданную ячейку

{

if( player.permutation(Px,Py,x,y,field)==true)

{

field.transformKing();

isWhiteTurn=!isWhiteTurn;//передача хода другому игроку

return true;

}

if(player.testEatenChecker(Px,Py,x,y,field)==true)//если шашка может "съесть" шашку противника

{

player.eatenChecker(Px,Py,x,y,field);

field.transformKing();

if(player.secondCourse(Px,Py,field)==true)//если возможен второй ход

variable=1;//в обратном случае присваиваем переменной значение 1

else

isWhiteTurn=!isWhiteTurn;//передача хода другому игроку

return true;

}

// emit printFalseTurn();

}

// emit printFalseTurn();

}

if(((isWhiteTurn==true && field.gameField[x][y]==whiteKing) ||

(isWhiteTurn==false && field.gameField[x][y]==blackKing))//если сейчас должен ходить игрок, играющий белыми(черными), и шашка выбрана так же белая(черная)

&&(field.gameField[Px][Py]==empty))

{

king.setx(x);

king.sety(y);

if ((field.gameField[x][y]==whiteKing && king.*makeMove1*(Px,Py)==true)

|| (field.gameField[x][y]==blackKing && king.*makeMove2*(Px,Py)==true))//если на ячейке белая(черная) шашка и возможен ход ее на заданную ячейку

{

if( player.permutation(Px,Py,x,y,field)==true)

{

isWhiteTurn=!isWhiteTurn;//передача хода другому игроку

return true;

}

if(player.testEatenChecker(Px,Py,x,y,field)==true)//если шашка может "съесть" шашку противника

{

player.eatenChecker(Px,Py,x,y,field);

if(player.secondCourse(Px,Py,field)==true)//если возможен второй ход

variable=1;//в обратном случае присваиваем переменной значение 1

else

isWhiteTurn=!isWhiteTurn;//передача хода другому игроку

return true;

}

}

emit printFalseTurn();

}

return false;

}

1. Тестовый класс.

Класс предназначен для проверки правильности выполнения игры и корректности результатов.

1. Класс MainWindow.

Он осуществляет взаимодействие пользователя с программой и отображает игровое поле на экран.

## Графический интерфейс.

Взаимодействие программы с пользователем происходит при помощи графического интерфейса. MainWindow – главное диалоговое окно, через которое осуществляется взаимодействие с пользователем. Внешний вид программы:



В левой части окна располагается шахматная доска с фигурами. В правой – индикатор, показывающий, какие фигуры ходят на данный момент. В левом верхнем углу кнопка «New game», при нажатии на которую выходит диалоговое окно, если нажимаем «Yes», игра начинается заново, если «No»-прежняя игра продолжается.



При попытке сделать неправильный ход выводится сообщение о его неверности:

При победе выводится сообщение о том, какая сторона победила:

