Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

Факультет «Робототехника и комплексная автоматизация»

Кафедра «Системы автоматизированного проектирования»

**Отчет по практической работу**

По курсу «Проектно-технологическая практика»

Выполнил: Студент Петраков С.А.

Группа РК6-36Б

Проверил: \_\_Кузьмина И.А.\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2020 г.

**Вариант G16**

**Задание:**

Разработать ООП для компьютерной реализации игры с перестановкой односторонней ладьи на шахматной доске по следующим правилам. Сначала ладья стоит в поле a1. Два игрока должны по очереди переставлять ее по горизонтали или вертикали на любое число полей в направлении поля h8. Проигрывает тот, кто не может сделать свой очередной ход. Эта игра должна быть реализована в варианте, когда противниками являются компьютер и человек. Человек может делать произвольные ходы по запросу стандартного ввода. При этом любое нарушение правил должно приводить к аварийному прерыванию игры. Компьютер должен отвечать ортогональными ходами, возвращая ладью на диагональ a1-h8, со стандартным выводом положения ладьи на шахматной доске. Игра должна завершаться диагностикой ее итогового результата.

**Алгоритм:**

Выводим на экран поле с доской.

Вводим клетку, на которую мы хотим сходить.

Компьютер ходит, следуя такой логике: Если находится ниже диагонали доски, то необходимо сдвинуть его вверх на разницу между его буквенной и цифровой координатой, если ты то необходимо сдвинуть вправо, на ту же разницу.

Использовались классы фигуры, базовый класс игрока и два наследуемых от него класса человека и компьютера. В базовом классе хранится указатель на фигуру, и виртуальные функции хода игроков. В наследуемых классах описаны эти самые функции.

**Входные данные:**

Отсутствуют

**Выходные данные:**

Клетки шахматной доски. Горизонтали и вертикали изображения шахматной доски должны маркироваться цифрами 8−1 и буквами a−h по ее краям. Лог ходов игрока и компьютера. Вывод кто победил.

**Текст программы:**

***main.cpp***

#include "Figure.h"

#include "Man.h"

#include "Pen.h"

#include <iostream>

void fastPrint(int inputs, Figure& f)

{

std::cout << "\x1b[14D";

for (int i = 0; i < (inputs + 10); i++)

std::cout << "\x1b[A";

f.printBoard();

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

std::cout << "\n\r";

}

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

{

//Initialization start position

char\* pos = (char\*)"a1";

char\* endPos = (char\*)"h8";

//Init Game

Figure f(pos);

f.printBoard();

//Player\* p = new Player(std::string("First"));

Gambler\* players[] = {

new Man(f, "First"),

new Pen(f, "Second")

};

//Start Game

int i = 0;

int inputs = 0;

while (f != endPos)

{

players[i%2]->query();

if (!players[i%2]->move())

{

std::cout << "Invalid input. Terminate.\n";

return 0;

}

inputs++;

fastPrint(inputs,f);

i++;

}

std::cout << std::endl << players[(i+1) % 2]->getName()<<" Wins"<<std::endl;

delete players[0];

delete players[1];

return 0;

}

***Gambler.h***

#ifndef GAMBLERH

#define GAMBLERH

#include "Figure.h"

#include <iostream>

class Gambler

{

protected:

Figure\* \_figure;

const char\* \_name;

public:

const char\* getName();

Gambler(Figure&);

void query();

virtual bool move() = 0;//True if all good, false if game is broken

};

#endif

***Gambler.cpp***

#include "Gambler.h"

const char\* Gambler::getName()

{

return \_name;

}

Gambler::Gambler(Figure& f)

{

\_figure = &f;

}

void Gambler::query()

{

std::cout << \_name <<": "<< (\*\_figure)<<"-"<<\_figure->isA();

return;

}

***Man.h***

#ifndef MANH

#define MANH

#include "Gambler.h"

#include <iostream>

class Man : public Gambler

{

public:

Man(Figure&, const char\*);

virtual bool move();

};

#endif

***Man.cpp***

#include "Man.h"

Man::Man(Figure& f,const char\* n) :Gambler(f)

{

\_name = n;

}

bool Man::move()

{

char s[3];

std::cin.unsetf(std::ios::skipws);

std::cin >> s[0] >> s[1];

s[2] = '\0';

std::cin.ignore(64, '\n');

if (Figure::deskout(s) || (\_figure->attack(s) == 0) || (\*\_figure) == s)

return false;

(\*\_figure) = s;

return true;

}

***Pen.h***

#ifndef PENH

#define PENH

#include "Gambler.h"

class Pen : public Gambler

{

public:

Pen(Figure&, const char\*);

virtual bool move();

};

#endif

***Pen.cpp***

#include "Pen.h"

Pen::Pen(Figure& f, const char\* n) :Gambler(f)

{

\_name = n;

}

bool Pen::move()

{

static char s[3];

s[0] = \_figure->getPosAlp();

s[1] = \_figure->getPosNum();

s[2] = '\0';

if (s[1] - '1' < s[0] - 'a')

s[1] = s[1] + ((s[0] - 'a') - (s[1] - '0')) + 1;

else

s[0] = s[0] - ((s[0] - 'a') - (s[1] - '0')) - 1;

(\*\_figure) = s;

std::cout << s;

std::cin.ignore(64,'\n');

return true;

}

***Figure.h***

#ifndef FIGUREH

#define FIGUREH

#include <iostream>

#include <string.h>

class Figure

{

protected:

char \_position[2];

public:

Figure(const char\*);//Construcor with string-position

Figure();

char getPosAlp();

char getPosNum();

char isA();//will return what is figure

int attack(char\*);

static int deskout(char\*);

void printBoard();

Figure& operator=(char\*);

int operator!=(char\*);

int operator==(char\*);

friend std::ostream& operator<<(std::ostream&, Figure&);

};

#endif

***Figure.cpp***

#include "Figure.h"

Figure::Figure(const char\* inp)

{

\_position[0] = inp[0];

\_position[1] = inp[1];

}

Figure::Figure()

{

\_position[0] = 'a';

\_position[1] = '1';

}

char Figure::getPosAlp()

{

return \_position[0];

}

char Figure::getPosNum()

{

return \_position[1];

}

char Figure::isA()

{

return 'C';

}

int Figure::attack(char\* p)

{

if (deskout(p) > 0) {

return 0;

}

int x = p[0] - \_position[0];

int y = p[1] - \_position[1];

if ((x==0 &&y>0)|| (y==0 &&x>0)) {

return 1;

}

return 0;

}

int Figure::deskout(char\* pos)

{

if (strlen(pos) == 2)

return (pos[0] > 'h') || (pos[0] < 'a') || (pos[1] < '1') || (pos[1] > '8');

return 1;

}

void Figure::printBoard()

{

char s[3];

s[2] = '\0';

const char\* mark = ".+";

std::cout << " a b c d e f g h\n";

for (int i = 8; i > 0; i--) {

std::cout << i << ' ';

s[1] = '0' + i;

for (int j = 0; j < 8; j++) {

s[0] = 'a' + j;

char m = (\*this != s) ? mark[attack(s)] : isA();

std::cout << m << ' ';

}

std::cout << i << "\n";

}

std::cout << " a b c d e f g h\n";

}

Figure& Figure::operator=(char\* p)

{

\_position[0] = p[0];

\_position[1] = p[1];

return \*this;

}

int Figure::operator!=(char\* p)

{

return (\_position[0] != p[0]) || (\_position[1] != p[1]);

}

int Figure::operator==(char\* p)

{

return((\_position[0] == p[0]) && (\_position[1] == p[1]));

}

std::ostream& operator<<(std::ostream& out, Figure& f)

{

return out << f.isA() << f.\_position[0] << f.\_position[1];

}

**Тесты:**

|  |  |
| --- | --- |
| **Номер** | **Выход** |
| 1 |  |

**Список использованной литературы:**

* Волосатова Т.М., Родионов С.В. Лекции по курсу «Объектно-ориентированное программирование»
* bigor.bmstu.ru