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 домашнему заданию

По курсу «Объектно-ориентированное программирование»

Выполнил: Студент Петраков С.А.

Группа РК6-26Б

Проверил: _____

Дата _____

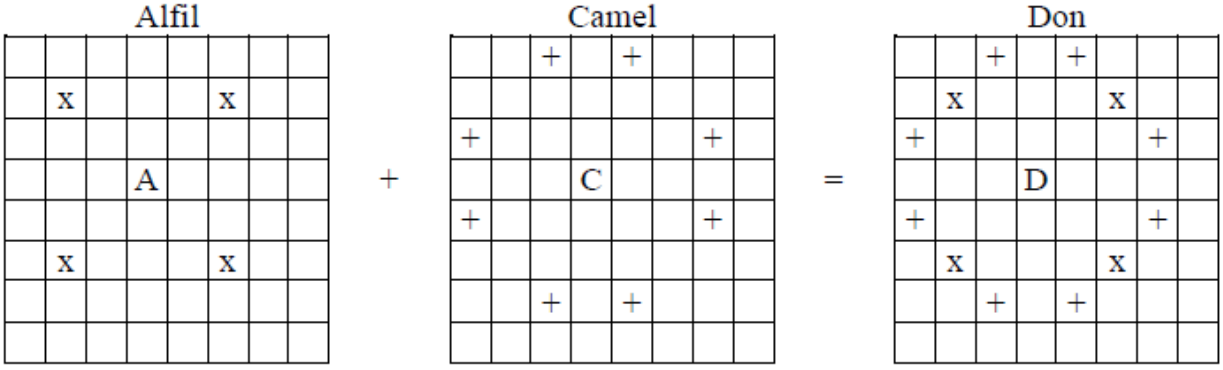
Подпись_____

Москва, 2020 г.

ООП С16+

Задание:

Разработать объектно-ориентированную программу ходов комплексной шахматной фигуры Don, которая объединяет свойства фигур Alfil и Camel со следующей схемой угроз. Программная реализация должна предусматривать множественное наследование базового суперкласса абстрактной фигуры и производных составляющих подклассов с перегрузкой метода виртуальной атаки для них и в их общем подклассе комплексной фигуры.



Начальная позиция фигуры (по умолчанию a1) должна задаваться аргументом командной строки вызова программы. Перестановка фигуры должна производиться по запросам новой позиции записями строк потока стандартного ввода в шахматной нотации. Стандартный вывод программы должен отображать клетки шахматной доски, которым угрожает заданная фигура из любой позиции, куда она может быть установлена. Клетки угроз должны быть обозначены знаками + и x по атакам составляющих фигур. Клетка, занятая самой фигурой должна маркироваться ее литерой, а остальные клетки для наглядности должны заполняться точками. Горизонтали и вертикали изображения шахматной доски должны маркироваться цифрами 8−1 и буквами a−h по ее краям. Шахматные координаты текущей позиции и запрос их для новой позиции фигуры должны отображаться под отображением схемы угроз. Выход из программы должен происходить при перестановке фигуры с нарушением правил ее хода. Программный код должен предусматривать виртуальное множественное наследование базового суперкласса абстрактной фигуры, где специфицированы защищенное позиционное поле с чистой атакой из него и идентификацией типа. Кроме того, в базовом классе должен быть реализован публичный метод индикации дифференциальных угроз по клеткам доски, а также публичная перегрузка операций адресации, сравнения, перестановки и потокового ввода-вывода позиции фигуры. Позиционный контроль должен обеспечивать статический метод. Производные публичные подклассы базового суперкласса должны специфицировать публичную перегрузку методов виртуальных атак и идентификации типа для составляющих фигур. Производный подкласс комплексной фигуры должен быть образован множественным наследованием подклассов составляющих фигур с публичной перегрузкой виртуальной атаки и идентификации типа. Конструкторы всех подклассов должны обеспечивать вызов конструктора базового суперкласса для инициализации позиции фигуры, по его адресному

аргументу. Для множественного наследования должны быть предусмотрены конструкторы без аргументов. Декларации базового и производных классов должны быть специфицированы в отдельных заголовочных файлах. Исходный код компонентных методов этих классов должен быть (ра)сосредоточен в соответствии с их заголовками. Визуальный интерфейс отображения схемы комплексных угроз из различных позиций шахматной доски должен быть реализован в основной функции программы.

Алгоритм:

Реализуем суперкласс абстрактной фигуры с чистыми виртуальными функциями, подклассы нашего базового класса и производный подкласс комплексной фигуры.

Проверяем на корректность введённую позицию на шахматной доске, если позиция удовлетворяет правилам шахмат, то ставим на эту позицию нашу фигуру и с помощью специальных функций помечаем поля шахматной доски, которым угрожает наша комплексная фигура, иначе прекращаем обработку ввода.

Входные данные:

Позиция поля на шахматной доске, куда перемещается фигура.

Выходные данные:

Клетки шахматной доски. Клетки угроз должны быть обозначены знаками + и x по атакам составляющих фигур. Клетка, занятая самой фигурой должна маркироваться ее литерой, а остальные клетки для наглядности должны заполняться точками. Горизонтали и вертикали изображения шахматной доски должны маркироваться цифрами 8−1 и буквами a−h по ее краям.

Текст программы:

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();
	virtual char isA() = 0;//will return what is figure
	virtual int attack(char*) = 0;
	static int deskout(char*);
	void printBoard();
	Figure& operator=(char*);
	int operator!=(char*);
	int operator==(char*);
	friend std::ostream& operator<<(std::ostream&, Figure&);
	friend std::istream & operator>>(std::istream&, Figure&);
};
#endif
```

Figure.cpp

```
#include "Figure.h"


Figure::Figure(const char* inp)
{
	_position[0] = inp[0];
	_position[1] = inp[1];
}

Figure::Figure()
{
}

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 = ".+x";
	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];
}

std::istream& operator>>(std::istream& in, Figure& f)
{
	char s[3];
	std::cin.unsetf(std::ios::skipws);
	in >> s[0] >> s[1];
	s[2] = '\0';
	in.ignore(64, '\n');
	if (Figure::deskout(s) || (f.attack(s) == 0) || f == s)
		in.setstate(std::ios::failbit);
	f = s;
	return in;
}
```

Alfil.h

```
#ifndef ALFILH
#define ALFILH
#include "Figure.h"
class Alfil : virtual public Figure
{
public:
    Alfil(const char* p) : Figure(p) {};
    char isA();
    int attack(char*);
};

#endif
}
```

Alfil.cpp

```
#include "Alfil.h"

char Alfil::isA()
{
    return 'A';
}

int Alfil::attack(char* p) {
    if (deskout(p) > 0) {
        return 0;
    }
    int x = p[0] - _position[0];
    int y = p[1] - _position[1];
    if (x < 0) {
        x = -x;
    }
    if (y < 0) {
        y = -y;
    }
    if ((x + y) != 4 || x != y) {
        return 0;
```

```
    }
    return 2;
}
```

Camel.h

```
#ifndef CAMELH
#define CAMELH
#include "Figure.h"
class Camel : virtual public Figure
{
public:
    Camel(const char* p) : Figure(p) {};
    char isA();
    int attack(char*);
};

#endif
```

Camel.cpp

```
#include "Camel.h"

char Camel::isA()
{
	return 'C';
}

int Camel::attack(char* p) {
	if (deskout(p) > 0) {
		return 0;
	}
	int x = p[0] - _position[0];
	int y = p[1] - _position[1];
	if (x < 0) {
		x = -x;
	}
	if (y < 0) {
		y = -y;
	}
	if ((x + y) != 4 || x == y || x * y == 0) {
		return 0;
	}
	return 1;
}
```

Don.h

```
#ifndef DONH
#define DONH
#include "Figure.h"
#include "Camel.h"
#include "Alfil.h"

class Don : public Camel, public Alfil
{
```

```
public:
    Don(const char* p) : Camel(p), Alfil(p), Figure(p) {};
    char isA();
    int attack(char*);
};

#endif
```

Don.cpp

```
#include "Don.h"

char Don::isA()
{
    return 'D';
}

int Don::attack(char* s) {
    if (Camel::attack(s) > 0)
        return 1;
    if (Alfil::attack(s) > 0)
        return 2;
    return 0;
}
```

main.cpp

```
#include "Don.h"
#include "Camel.h"
#include "Alfil.h"
#include <iostream>
int main(int argc, char* argv[])
{
	//Initialization start position
	const char* pos = "a1";
	if (argc < 2) {
		std::cout << "Incorrect count arguments. Using default: " << pos << std::endl;
	}
	else
	{
		if (Figure::deskout(argv[1]))
		{
			std::cout << "Incorrect position: " << argv[1];
			std::cout << " Using default: " << pos << std::endl;
		}
		else
			pos = argv[1];
	}
	//Init figure
	Don f(pos);
	int inputs = 0;//for nice output
	do
	{
		//from this for nice output
		if (inputs != 0)
		{
			for (int i = 0; i < (inputs + 10); i++)
```

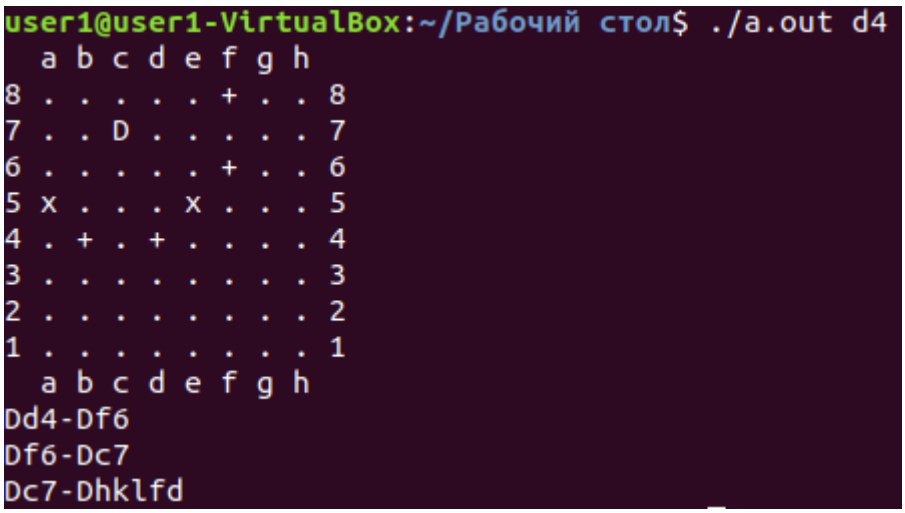
```
                        std::cout << "\x1b[A";
                f.printBoard();
                for (int i = 0; i < inputs; i++)
                        std::cout << "\n\r";
            }
            else//to this for nice output
                f.printBoard();
            std::cout << f << "-" << f.isA();
            inputs++;
        } while (std::cin >> f); //input new position

        return 0;
}
```

Тесты:

```
user1@user1-VirtualBox:~/Рабочий стол$ ./a.out d4
  a b c d e f g h
8 . . . . . . . . 8
7 . . + . + . . . 7
6 . x . . . x . . 6
5 + . . . . . + . 5
4 . . . D . . . . 4
3 + . . . . . + . 3
2 . x . . . x . . 2
1 . . + . + . . . 1
  a b c d e f g h
Dd4-D
```

```
user1@user1-VirtualBox:~/Рабочий стол$ ./a.out d4
  a b c d e f g h
8 . . . x . . . x 8
7 . . + . . . . . 7
6 . . . . . D . . 6
5 . . + . . . . . 5
4 . . . x . . . x 4
3 . . . . + . + . 3
2 . . . . . . . . 2
1 . . . . . . . . 1
  a b c d e f g h
Dd4-Df6
Df6-D
```

```
user1@user1-VirtualBox:~/Рабочий стол$ ./a.out d4
  a b c d e f g h
8 . . . . . + . . 8
7 . . D . . . . . 7
6 . . . . . + . . 6
5 x . . . x . . . 5
4 . + . + . . . . 4
3 . . . . . . . . 3
2 . . . . . . . . 2
1 . . . . . . . . 1
  a b c d e f g h
Dd4-Df6
Df6-Dc7
Dc7-D
```



Список использованной литературы:

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