|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИКИ**

Студент Камалов Антон Павлович

Группа РК6-35Б

Модуль 1

Вариант G16

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Камалов А.П.**

*подпись, дата фамилия, и.о.*

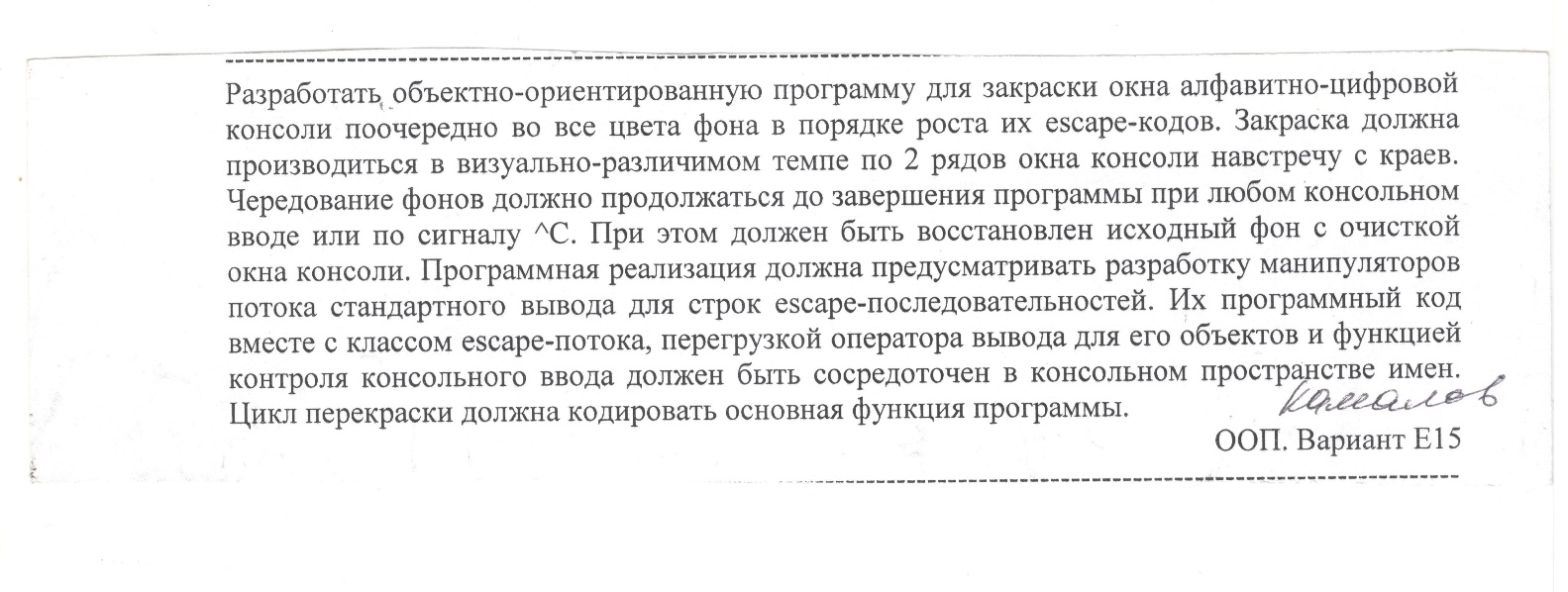
Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмина И.А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

**Задание**

****

**Описание программы**

**Входные данные:** отсутствуют в данной программе.

**Выходные данные:** расположение фигуры на шахматной доске.

**Класс Figure**

*Информационные поля:*

pos (char []) – координаты фигуры на шахматной доске;

*Конструкторы и деструкторы:*

Figure () – конструктор по умолчанию;

Figure (char\*) – конструктор инициализации;

*Методы:*

operator== (string) – публичный метод перегрузки оператора равенства;

Figure& operator=(char\*) – публичный метод перегрузки оператора присвоения переменной;

attack () – публичный виртуальный метод атаки фигуры;

isa () – публичный виртуальный метод маркировки фигуры;

desk () – публичный метод отображения шахматной доски;

deskout (char\*) – публичный статический метод проверки положения фигуры на доске;

get\_pos () – публичный метод получения координат фигуры на шахматной доске

**Класс Rook**

*Конструкторы и деструкторы:*

Rook (char\*) – конструктор инициализации;

*Методы:*

attack () – публичный метод атаки ладьи;

isa () – публичный метод маркировки ладьи;

desk () – публичный метод отображения шахматной доски.

**Класс Player**

*Информационные поля:*

rook (Rook\*) – указатель на тип фигуры «Ладья»;

name (const char\*) – имя игрока;

*Конструкторы и деструкторы:*

Player (char\*) – конструктор инициализации;

*Методы:*

strategy () – публичный виртуальный метод хода игрока;

request () – публичный метод проверки положения фигуры на шахматной доске;

get\_name () – публичный метод получения имени игрока.

**Класс Human**

*Конструкторы и деструкторы:*

Human (Rook, char\*) – конструктор инициализации;

*Методы:*

strategy () – публичный метод хода игрока-человека.

**Класс AI**

*Конструкторы и деструкторы:*

AI (Rook, char\*) – конструктор инициализации;

*Методы:*

strategy () – публичный метод хода игрока-компьютера.

**Описание алгоритма:**

Для обеспечения работы с изменением цвета фона терминала будем работать с escape-последовательностью (набором символов, который выталкивается в поток для его форматирования). Написание программы начнем с написания манипуляторов. Для написания манипулятора цвета, перемещения курсора необходимо создадим класс estream с приватным строковым полем escape, так как все они имеют параметры. Конструктор класса будет инициализировать это поле строкой, переданной аргументом. Для этого класса будет перегружен оператор вывода. Он будет направит строку в поток. Манипулятор flush будет выталкивать содержимое буфера в поток. Для escape-последовательности манипулятор необходимо явно прописать. Манипулятор цвета принимает в качестве аргумента код цвета, который имеет вид – 40+x, где х ∈ [0,7]. Манипуляторный код сосредоточен в пространстве имён con, куда также входят функции comax() и romax(), хранящие информацию о размерности терминала. Функция kbin() обеспечивает проверку нажатия клавиши ENTER. В буфер заносятся полученные из потока стандартного ввода символы. В переменную flags занесется результат вызова функции fcntrl(0, F\_GETFL). Здесь 0 означает, что действие идет с потоком стандартного ввода. Результат будет использован в функции read, которая будет просматривать буфер до тех пор, пока не будет нажата клавиша ENTER. Kbin() вернет значение большее единицы, когда будет нажата данная клавиша, и после программа завершит свою работу.

Необходимо раскрасить экран терминала по 2 ряда навстречу с краев во все цвета фона в порядке роста их escape-кодов (черный-красный-зеленый-желтый-синий-пурпурный-голубой-белый). Закраска продолжается до ввода с консоли, либо сигнала ^C. Цикл перекраски содержится в основной функции. С помощью функций определения ширины и длины экрана, а также манипуляторов во вложенных циклах меняется положение курсора и цвет, которым красить: курсор пробегает с каждого края по 2 строки слева-направо по каждой строчке, а затем меняет цвет и вновь меняет положение на строчку ниже/выше. Если пользователь изменяет размер терминала, то выполняется функция resize(), которая задаёт новое положение середины окна терминала.

**Список литературы**

1. Пол Айра. Объектно-ориентированное программирование на C++: пер. с англ. 2-е изд. СПБ.: Невский Диалект; М.: Издательство БИНОМ, 2001. 462 с. [Pohl Ira. Object-Oriented Programming Using C++. 2nd ed. Addison-Wesley, 1996. 576 p.].
2. Страуструп Б. Язык программирования C++: пер. с англ / под ред. Н.Н. Мартынова. Специальное изд. М.: Бином, 2011. 1035 c. [Stroustrup B. The C++ Programming Language. Special ed. Addison-Wesley, 2000. 1029p.].
3. Шилдт Г. Самоучитель C++. 3-е изд. СПБ.: БХВ-Петербург, 2002. 688 с. [Schildt H. Teach Yourself C++. 3d ed. McGraw-Hill, 1998. 768 p.].
4. Волосатова Т.М., Родионов С.В. Объектно-ориентированное программирование на С++. Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou (дата обращения 23.05.2023).

**Приложение 1**

Текст программы

**chman.h**

#ifndef CHESSMAN

#define CHESSMAN

#include <iostream>

using std::cout;

class Figure {

protected:

char pos[2];

public:

Figure() {}; // нужен для реализации мн. наследования

Figure(char \* p) {

pos[0] = p[0];

pos[1] = p[1];

};

bool operator==(char \* p) { return (pos[0] == p[0]) && (pos[1] == p[1]); };

Figure& operator=(char\*);

virtual int attack(char \*) = 0;

virtual char isa() = 0;

void desk();

static bool deskout(char \*);

char\* get\_pos() {return pos;};

};

class Rook: public Figure{

public:

Rook(char\* p) : Figure(p){};

int attack(char\*);

char isa() {return 'R';};

};

#endif

**chman.cc**

#include "chman.h"

Figure& Figure::operator=(char\* s){

pos[0] = s[0];

pos[1] = s[1];

return (\*this);

}

bool Figure::deskout(char\* p) {

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

}

void Figure::desk() {

int i, j;

char s[2];

const char \* mark = ".+x";

char m;

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

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

cout << i << ' ';

s[1] = '0' + i;

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

s[0] = 'a' + j;

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

cout << m << " ";

}

cout << i << '\n';

}

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

return;

}

**chlib.h**

#ifndef CHESSLIB

#define CHESSLIB

#include "chman.h"

class Player{

protected:

Rook\* rook;

const char\* name;

public:

Player(Rook\* p) : rook(p){};

virtual int strategy() = 0;

int request();

const char\* get\_name();

};

class Human : public Player{

public:

Human(Rook\* rook, const char\*p) : Player(rook){name = p;};

int strategy();

};

class AI : public Player{

public:

AI(Rook\* rook, const char\* p) : Player(rook){name = p;};

int strategy();

};

#endif

**chlib.cc**

#include "chlib.h"

const char\* Player::get\_name(){

return name;

}

int Player::request(){

if (\*rook == (char\*)"a1"){

return 0;

} else{

return 1;

}

}

int Human::strategy(){

cout << "Ход игрока: ";

char s[3];

char\* pos = rook->get\_pos();

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

s[0] = tolower(s[0]);

if (rook->deskout(s) || (pos[0] == s[0] && pos[1] == s[1]) || (pos[0] != s[0] && pos[1] != s[1]) || (pos[0] == s[0] && s[1] > pos[1]) || (pos[1] == s[1] && s[0] > pos[0])){

return 0;

}

\*rook = s;

return 1;

}

int AI::strategy(){

char s[2];

char \*pos = rook->get\_pos();

if (int(pos[0]) - 96 > int(pos[1]) - 48){

s[0] = char('a' + int(pos[1])-49);

s[1] = pos[1];

} else{

s[0] = pos[0];

s[1] = char('1' + (int(pos[0]) - 97));

}

cout << "Ход компьютера: " << s[0] << s[1] << std::endl;

\*rook = s;

return 1;

}

int Rook::attack(char \*p) {

if (deskout(p) || (p[1] == pos[1] && p[0] > pos[0]) || (p[0] == pos[0] && p[1] > pos[1]))

return 0;

int x = abs(p[0] - pos[0]);

int y = abs(p[1] - pos[1]);

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

return 1;

return 0;

}

**chess.cpp**

#include "chlib.h"

int main() {

int i = 1;

Rook k((char\*)"h8");

Player\* player[] = {new Human(&k, "Human"), new AI(&k, "Computer")};

while (player[i]->request()){

i = (i == 1) ? 0 : 1;

k.desk();

if (!(player[i]->strategy())){

cout << player[i]->get\_name() << " made mistake. " << player[(i+1 == 1) ? 1 : 0]->get\_name() << " has won!\n";

exit(-1);

}

}

k.desk();

if (i == 0){

cout << player[0]->get\_name() << " WINS!" << '\n';

} else{

cout << player[1]->get\_name() << " WINS!" << '\n';

}

delete player[0];

delete player[1];

}

**Приложение 2**

Результат работы программы

