ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Наследование классов, базовый класс, производный класс» |
| по курсу: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель Работы**

Изучить механизм создания нового класса на основе уже существующего, варианты доступа к элементам базового класса из производного.

**Вариант 3**

Создать класс Board для описания шахматной доски. В нём предусмотреть массив 8х8 элементов и метод для перевода цифр 1-8 в буквы A-H и обратно. На основе класса Board создать класс Composition для составления шахматной композиции. В нём предусмотреть возможность многократного добавления/удаления фигур на доску, распечатку композиции.

**Листинг**

#include <iostream>

#include <string>

class Board

{

public:

Board(int x, int y);

~Board();

void render();

class tile

{

public:

tile(int x = 0, int y = 0) { this->hor = x; this->ver = y; };

void set(int x, int y) { this->hor = x; this->ver = y; ych = 64 + y; };

int hor, ver;

char ych;

char ent = '.';

};

protected:

tile\*\* tiles;

int size\_x, size\_y;

};

Board::Board(int x=8, int y=8)

{

tiles = new tile\*[x];

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

{

tiles[i] = new tile[y];

}

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

{

for (int j = 0; j < y; j++)

{

tiles[i][j].set(i + 1, j + 1);

}

}

size\_x = x;

size\_y = y;

}

Board::~Board()

{

}

void Board::render()

{

for (int i = 0; i < size\_x; i++)

{

for (int j = 0; j < size\_y; j++)

{

std::cout << tiles[i][j].ych << tiles[i][j].hor << '\t';

}

std::cout << std::endl;

}

}

class Composition : public Board

{

public:

Composition(int x , int y );

void show();

void place(char y, int x, char fig);

void compose();

void move(char y, int x, char v, int z);

private:

};

Composition::Composition(int x, int y)

{

tiles = new tile \* [x];

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

{

tiles[i] = new tile[y];

}

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

{

for (int j = 0; j < y; j++)

{

tiles[i][j].set(i + 1, j + 1);

}

}

size\_x = x;

size\_y = y;

};

void Composition::place(char y, int x, char fig)

{

tiles[x-1][y - 65].ent = fig;

}

void Composition::compose()

{

//Black

place('A', 1, 'T'); place('B', 1, 'H'); place('C', 1, 'E'); place('D', 1, 'Q'); place('E', 1, 'K'); place('F', 1, 'E'); place('G', 1, 'H'); place('H', 1, 'T');

place('A', 2, 'M'); place('B', 2, 'M'); place('C', 2, 'M'); place('D', 2, 'M'); place('E', 2, 'M'); place('F', 2, 'M'); place('G', 2, 'M'); place('H', 2, 'M');

//White

place('A', 8, 't'); place('B', 8, 'h'); place('C', 8, 'e'); place('D', 8, 'q'); place('E', 8, 'k'); place('F', 8, 'e'); place('G', 8, 'h'); place('H', 8, 't');

place('A', 7, 'm'); place('B', 7, 'm'); place('C', 7, 'm'); place('D', 7, 'm'); place('E', 7, 'm'); place('F', 7, 'm'); place('G', 7, 'm'); place('H', 7, 'm');

}

void Composition::move(char y, int x, char v, int z)

{

if (y<=57)

{

char fig = tiles[x - 1][y - 49].ent;

if (fig == '.') return;

tiles[x - 1][y -49].ent = '.';

tiles[z - 1][v -49].ent = fig;

}

else

{

char fig = tiles[x - 1][y - 65].ent;

if (fig == '.') return;

tiles[x - 1][y - 65].ent = '.';

tiles[z - 1][v - 65].ent = fig;

}

}

void Composition::show()

{ //оглавление сетки

std::cout << "\t";

for (int i = 0; i < size\_x; i++)

{

char symb = 65 + i;

std::cout << symb << "\t";

}

std::cout << "\n";

for (int i = 0; i < size\_x; i++)

{

std::cout << i + 1 << '\t';

for (int j = 0; j < size\_y; j++)

{

std::cout << tiles[i][j].ent << "\t";

}

std::cout << "\n";

}

std::cout << "Пример ввода A5 A7\n";

}

int main()

{

system("color f1");

Composition chess(8, 8);

chess.compose();

char y, v, chx, chz;

int x, z;

std::string from;

std::string to;

while (true)

{

chess.show();

std::cin >> from;

std::cin >> to;

y = from[0];

chx = from[1];

x = (int)chx -48;

v = to[0];

chz = to[1];

z = (int)chz -48;

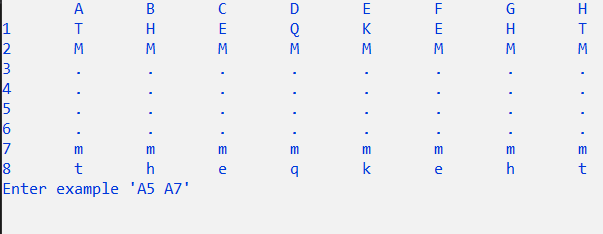
chess.move(y, x, v, z);

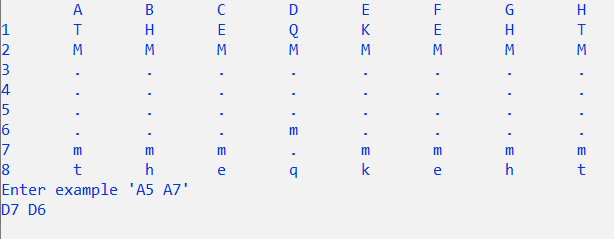
system("cls");

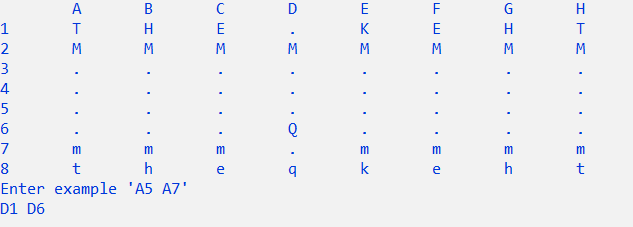
}

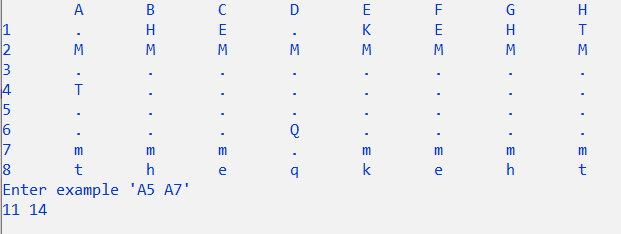
}

**Результат**









**Вывод**

Я изучил наследование классов и разработал на основе родительского класса Board класс Composition, который имитирует шахматную доску