**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Лабораторная работа №5**

**по дисциплине «Языки программирования»**

Рекурсия и поисковые задачи

Группа: **АВТ-615**

Студент: **Кузенков В.С.**

Преподаватель: **Балагуров М.В.**

НОВОСИБИРСК 2017

# Постановка задачи

На шахматной доске имеется N фигур противника. Разместить на ней минимальное количество слонов так, чтобы все фигуры противника находились «под боем

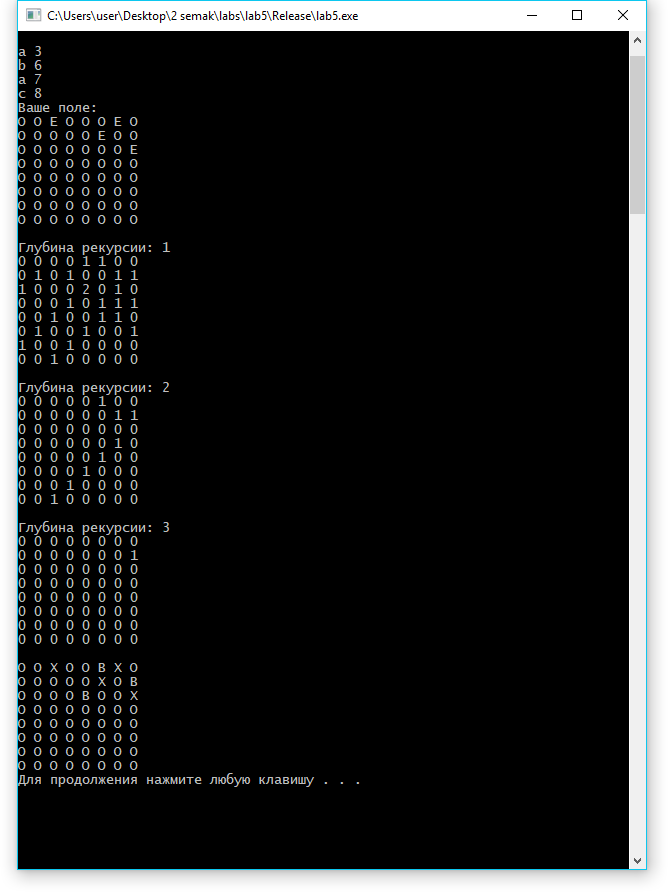
# Ограничения

Отсутствуют

# Тестирование

**Входные данные**

**Выходные данные**

****

# Текст программы:

#include <iostream>

using namespace std;

/\*7.На шахматной доске имеется N фигур противника. Разместить на ней минимальное количество слонов так, чтобы все фигуры противника находились «под боем».\*/

unsigned short chessboard[8][8];

unsigned short reqDeep = 0;

void fill\_chessboard()

{

for (int i = 0; i < 8;i++)

{

for (int j = 0; j < 8; j++) chessboard[i][j] = 0;

}

}

void put\_figure(char x, int y)

{

chessboard[x- 96 -1][y-1] = 1;

}

void print\_chessboard()

{

for (int i = 0; i < 8; i++)

{

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

if(chessboard[i][j] == 0) //empty

{

cout << 'O' << ' ';

}

if (chessboard[i][j] == 1) //enemy

{

cout << 'E' << ' ';

}

if (chessboard[i][j] == 2) //shooted enemy

{

cout << 'X' << ' ';

}

if (chessboard[i][j] == 3) //bishop

{

cout << 'B' << ' ';

}

}

cout << endl;

}

}

bool isNormal()

{

for (int i = 0; i < 8; ++i)

{

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

{

//проверяем фигуры в углу

if (chessboard[i][j] == 1 && i == 0 && j == 0 && chessboard[i + 1][j + 1] == 1) //левый верхний

{

return 0;

}

if (chessboard[i][j] == 1 && i == 0 && j == 7 && chessboard[i + 1][j - 1] == 1) //правый верхний

{

return 0;

}

if (chessboard[i][j] == 1 && i == 7 && j == 0 && chessboard[i - 1][j + 1] == 1) //левый нижний

{

return 0;

}

if (chessboard[i][j] == 1 && i == 7 && j == 7 && chessboard[i - 1][j - 1] == 1) //правый нижний

{

return 0;

}

//проверяем фигуры на 4 диагоналях

if(chessboard[i][j] == 1 && chessboard[i - 1][j - 1] == 1 && chessboard[i - 1][j + 1] == 1 && chessboard[i + 1][j - 1] == 1 && chessboard[i + 1][j + 1] == 1)

{

return 0;

}

}

}

return 1;

}

unsigned short put\_bishop(unsigned short bishopMatrix[8][8])

{

short figureCount = 0; //узнаем количество не убитых фигур

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++) if(chessboard[i][j] == 1) figureCount++;

}

if (figureCount <= 0)

{

return \*\*bishopMatrix;

}

//заполняем массив с диагональными "прострелами" фигур

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++) bishopMatrix[i][j] = 0;

}

//простреливаем диагонали

for (int i = 0; i < 8; i++)

{

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

{

if (chessboard[i][j] == 1)

{

for (int k = i + 1, t = j + 1; k < 8 && t < 8; k++, t++)

{

if (chessboard[k][t] == 1 || chessboard[k][t] == 2) break; //если на диагонали встречается еще одна фигура, то пропускаем

bishopMatrix[k][t]++;

}

for (int k = i - 1, t = j - 1; k >= 0 && t >= 0; k--, t--)

{

if (chessboard[k][t] == 1 || chessboard[k][t] == 2) break;

bishopMatrix[k][t]++;

}

for (int k = i - 1, t = j + 1; k >= 0 && t < 8; --k, ++t)

{

if (chessboard[k][t] == 1 || chessboard[k][t] == 2) break;

bishopMatrix[k][t]++;

}

for (int k = i + 1, t = j - 1; k < 8 && t >= 0; ++k, t--)

{

if (chessboard[k][t] == 1 || chessboard[k][t] == 2) break;

bishopMatrix[k][t]++;

}

}

}

}

reqDeep++;

cout << "Глубина рекурсии: " << reqDeep << endl;

for (int i = 0; i < 8; i++)

{

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

cout << bishopMatrix[i][j] << ' ';

}

cout << endl;

}

cout << endl;

//ищем точку с наибольшим прострелом

unsigned short max = 0;

unsigned short x = 0, y = 0;

for (int i = 0; i < 8; ++i)

{

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

{

if(bishopMatrix[i][j] > max)

{

max = bishopMatrix[i][j];

x = j;

y = i;

}

}

}

//ставим туда слона

chessboard[y][x] = 3;

//фигуры в ауте заменяем двойками

for (int k = y + 1, t = x + 1; k < 8 && t < 8; k++, t++)

{

if (chessboard[k][t] == 1) {

chessboard[k][t] = 2;

break;

}

}

for (int k = y - 1, t = x - 1; k >= 0 && t >= 0; k--, t--)

{

if (chessboard[k][t] == 1) {

chessboard[k][t] = 2;

break;

}

}

for (int k = y - 1, t = x + 1; k >= 0 && t < 8; --k, ++t)

{

if (chessboard[k][t] == 1){

chessboard[k][t] = 2;

break;

}

}

for (int k = y + 1, t = x - 1; k < 8 && t >= 0; ++k, t--)

{

if (chessboard[k][t] == 1) {

chessboard[k][t] = 2;

break;

}

}

return put\_bishop(bishopMatrix);

}

int main()

{

setlocale(0, "rus");

fill\_chessboard();

int count = 0;

cout << "Сколько фигур будем расставлять?" << endl;

cin >> count;

cout << "Введите по очереди, сначала строчную букву, затем цифру, расположения фигуры" << endl;

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

{

int n = 0;

char c = ' ';

cin >> c;

cin >> n;

if(n > 8 || (c - 96) > 8)

{

cout << "Введите корректные значения" << endl;

--i;

continue;

}

put\_figure(c, n);

}

cout << "Ваше поле:" << endl;

print\_chessboard();

cout << endl;

unsigned short bishopMatrix[8][8];

if (isNormal()){

put\_bishop(bishopMatrix);

print\_chessboard();

}

else cout << "Есть фигуры закрытые по углам, слонов поставить невозможно" << endl;

system("pause");

return 0;

}

**Вывод:** Основываясь на практическом опыте было выяснено, что рекурсивным называется способ построения объекта (понятия, системы, описание действия), в котором определение объекта включает аналогичные объекты (понятие, систему, действие) в виде составных  частей.