# 

# Локальная разработка. Линейные алгоритмы

*Выполнила: Романчук Олеся 3ИБ3*

**Содержание**

[**Локальная разработка. Линейные алгоритмы 1**](#_hg50dknlsxtq)

[**Задача 2**](#_eqrsjc6esygm)

**Вербальная модель 2**

[**Математическая модель 3**](#_ejgsxj6ygn14)

[Угроза от ладьи возникает при условии: 3](#_jfaswue0kkot)

[Угроза от слона возникает при условии: 3](#_c15pv94vjgv0)

[Условия перекрытия: 3](#_70oevlbrp7ix)

[**Блок схема 4**](#_cqqoet3e4hgu)

[**Программа на языке высокого уровня 4**](#_rd6g91tjfuh3)

# Задача

На шахматной доске стоят черный король и белые ладья и слон (ладья бьет по горизонтали и вертикали, слон – по диагоналям). Проверить, есть ли угроза королю и если есть, то от кого именно. Учесть возможность защиты (например, ладья не бьет через слона)

# Вербальная модель

Дана шахматная доска 8х8, с координатами по х - {a-h} и по y - {1-8}.

На доске находятся 3 фигуры: слон, король, ладья, расположение (координаты) которые задаются случайным образом.

Слон может совершать движение по диагоналям, ладья по вертикали и горизонтали, король в нашем случае всегда находится на своей позиции.

Следовательно, мы можем рассмотреть исходы, где будет поставлен шах:

1. Если король и ладья принадлежат одной координате по горизонтали или по вертикале
2. Если слон и король принадлежат одной и той же диагонали

Также в работе мы будем учитывать возможность перекрытия фигур, т.е.

1. Ладья перекрывает ход, находясь на диагонали слона
2. Слон перекрывает ход, находясь на горизонтали и вертикали хода ладьи

В случае выполнения одного из исходов и отсутствия перекрытия фигур в ответе мы должны получить вывод шаха, в ином случае, сообщение об его отсутвии.

# Математическая модель

Представим координаты каждой из фигур, как

(Кор.г; Кор.в)

(Лад.г; Лад.в)

(Сл.г; Сл.в)

## Угроза от ладьи возникает при условии:

Если Кор.г = Лад.г или если Кор.в = Лад.в

## Угроза от слона возникает при условии:

## Условия перекрытия:

обозначим фигура, как

k - король

l - ладья

s - слон

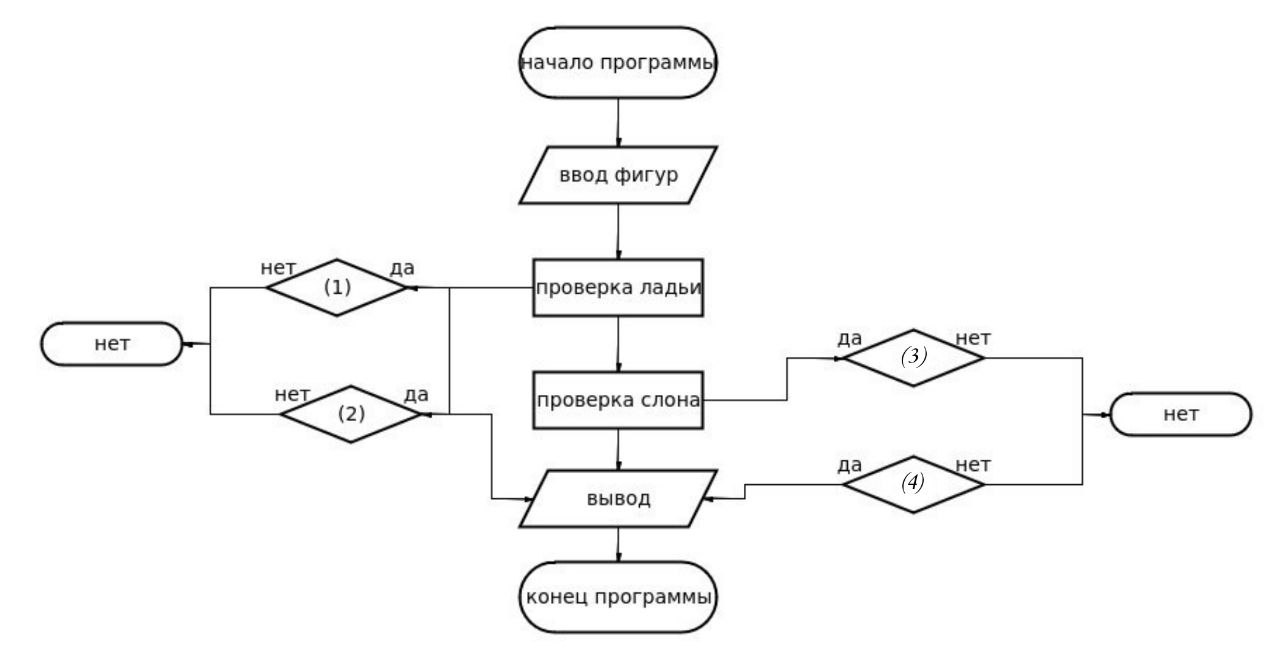
Ладья перекрывает слона, если

Слон перекрывает ладью, при условии

]

Эти условия представляют собой математическое описание задачи с учетом угрозы и возможности перекрытия между ладьей и слоном.

# Блок схема



1. K.x = L.x or K.y = L.y

(2)

(3)

(4)

# Программа на языке высокого уровня

Фрагмент кода отвечает за случайное распределение фигур на шахматной доске.

*Создание объекта Random:*

Создается объект Random, который будет использоваться для генерации случайных чисел.

*Размещение короля ('K'):*

Генерируются случайные координаты kingX и kingY для короля.

*Размещение ладьи ('L') и Размещение слона ('S')::*

Генерируются случайные координаты rookX и rookY для ладьи.

Вводится цикл while, чтобы убедиться, что координаты ладьи не совпадают с координатами короля. Если они совпадают, генерируются новые координаты.

Фигура ладьи ('L') помещается на доску в соответствующую ячейку.

Те же действия выполняются с координатами слона

print 'hello world!' **private** **void** placeRandomPieces() {  
 Random random = **new** Random();  
  
 kingX = random.nextInt(**8**);  
 kingY = random.nextInt(**8**);  
 board[kingX][kingY] = 'K';  
  
 rookX = random.nextInt(**8**);  
 rookY = random.nextInt(**8**);  
 **while** (rookX == kingX && rookY == kingY) {  
 rookX = random.nextInt(**8**);  
 rookY = random.nextInt(**8**);  
 }  
 board[rookX][rookY] = 'L';  
  
 bishopX = random.nextInt(**8**);  
 bishopY = random.nextInt(**8**);  
 **while** ((bishopX == kingX && bishopY == kingY) || (bishopX == rookX && bishopY == rookY)) {  
 bishopX = random.nextInt(**8**);  
 bishopY = random.nextInt(**8**);  
 }  
 board[bishopX][bishopY] = 'S';  
 }

Фрагмент кода отвечает за проверку, находится ли король под угрозой шаха.

*Проверка по горизонтали или вертикали для ладьи:*

**if (kingX == rookX || kingY == rookY):** Проверяется, находятся ли король и ладья в одной строке или столбце. Если координаты короля совпадают с координатами ладьи по горизонтали или вертикали, то метод возвращает **true**, что означает, что король находится под угрозой шаха.

*Проверка по диагонали для слона:*

**if (Math.abs(kingX - bishopX) == Math.abs(kingY - bishopY)):** Здесь используется Math.abs для определения расстояния по диагонали между королём и слоном. Если расстояния по горизонтали и вертикали равны (что означает, что король и слон находятся на одной диагонали), то метод также возвращает **true**, указывая на угрозу шаха.

*Возврат значения:* Если ни одно из условий не выполнено, метод возвращает **false**, что говорит о том, что король в текущей позиции не находится под угрозой шаха.

**public** **boolean** **isCheck**() {  
 // проверяем, находятся ли король и ладья в одной строке или столбце  
 **if** (kingX == rookX || kingY == rookY) {  
 **return** **true**;  
 }  
  
 // проверяем, находятся ли король и слон на одной диагонали  
 **if** (Math.abs(kingX - bishopX) == Math.abs(kingY - bishopY)) {  
 **return** **true**;  
 }  
  
 **return** **false**;  
 }

Проверка

(ТУТ ПРОВЕРКА)

Заключение

В данной работе, мы разработали программу для решения шахматной задачи на обнаружение шаха для короля. Разработка программы началась с постановки задачи и создания вербальной модели, в которой мы словесно описали будущую работу программы и на ее основе создали математическую модель, которая дополнила вышесказанное описанием с помощью математических формул. После была составлена блок-схема, отражающая основной алгоритм программы. В конечном итоге мы написали код на высокоуровневом языке Java (полный код см. “Приложение”).

Вывод

(тут написать чет в кратце и объяснить результаты тестов)

Список использованных источников

* <https://tproger.ru/translations/java-intro-for-beginners>
* <https://multiurok.ru/files/pravila-nakhozhdeniia-koeffitsientov-kvadratichnoi.html>
* https://javarush.com/groups/posts/1949-znakomstvo-s-klassami-napisanie-sobstvennihkh-klassov-konstruktorih
* <http://bikmeyev-at.ugatu.su/students/CPP/Handbook/blocks.html>
* <https://obzorposudy.ru/polezno/verbalnye-modeli-sushhnost-i-znacenie>

Приложение

**import** **java.util.Random**;  
  
**class** **ChessBoard** {  
 **char**[][] board;  
 **int** kingX, kingY;  
 **int** rookX, rookY;  
 **int** bishopX, bishopY;  
  
 **public** **ChessBoard**() {  
 board = **new** **char**[**8**][**8**];  
 initializeBoard();  
 placeRandomPieces();  
 }  
  
 **private** **void** **initializeBoard**() {  
 **for** (**int** i = **0**; i < **8**; i++) {  
 **for** (**int** j = **0**; j < **8**; j++) {  
 board[i][j] = '-';  
 }  
 }  
 }  
  
 **private** **void** **placeRandomPieces**() {  
 Random random = **new** Random();  
  
 kingX = random.nextInt(**8**);  
 kingY = random.nextInt(**8**);  
 board[kingX][kingY] = 'K';  
  
 rookX = random.nextInt(**8**);  
 rookY = random.nextInt(**8**);  
 **while** (rookX == kingX && rookY == kingY) {  
 rookX = random.nextInt(**8**);  
 rookY = random.nextInt(**8**);  
 }  
 board[rookX][rookY] = 'L';  
  
 bishopX = random.nextInt(**8**);  
 bishopY = random.nextInt(**8**);  
 **while** ((bishopX == kingX && bishopY == kingY) || (bishopX == rookX && bishopY == rookY)) {  
 bishopX = random.nextInt(**8**);  
 bishopY = random.nextInt(**8**);  
 }  
 board[bishopX][bishopY] = 'S';  
 }  
  
 **public** **boolean** **isCheck**() {  
 // проверяем, находятся ли король и ладья в одной строке или столбце  
 **if** (kingX == rookX || kingY == rookY) {  
 **return** **true**;  
 }  
  
 // проверяем, находятся ли король и слон на одной диагонали  
 **if** (Math.abs(kingX - bishopX) == Math.abs(kingY - bishopY)) {  
 **return** **true**;  
 }  
  
 **return** **false**;  
 }  
  
 **public** **void** **printBoard**() {  
 **for** (**int** i = **0**; i < **8**; i++) {  
 **for** (**int** j = **0**; j < **8**; j++) {  
 System.out.print(board[i][j] + " ");  
 }  
 System.out.println();  
 }  
 }  
}  
  
**public** **class** **chess** {  
 **public** **static** **void** **main**(String[] args) {  
 ChessBoard chessBoard = **new** ChessBoard();  
  
 System.out.println("Исходная шахматная доска:");  
 chessBoard.printBoard();  
  
 **if** (chessBoard.isCheck()) {  
 System.out.println("Шах! Король находится под угрозой.");  
 } **else** {  
 System.out.println("Шаха нет. Король в безопасности.");  
 }  
 }  
}