Java

wprowadzenie

instrukcja iteracyjna - DO-WHILE

Czego się dowiesz?

- Czym do-while różni się od while?
- Jak ją stworzyć?
- Jakie jest jej zastosowanie?

While vs do-while

Instrukcja do-while, różni się od while jednym elementem. Do-while sprawdza warunek na końcu instrukcji. W konsekwencji pętla do-while wykona się zawsze przynajmniej jeden raz.

Pętla do-while jest rzadziej wykorzystywana niż pętla while.

Definicja

```
do {
      instrukcje;
} while(warunek);
```

Pętla wykonuje instrukcje, dopóki warunek jest prawdziwy.

Przykład 1

Program losujący liczbę 10, spośród liczb od 1 do 10. W takiej sytuacji najpierw losujemy, później sprawdzamy.

```
do {
      // wylosuj liczbę od od 1 do 10
} while (liczba_nie_jest_równa_10);
```

Przykład 2

Gra w której, należy odgadnąć liczbę wymyśloną przez program.

```
// program zapamiętuje liczbę
do {
      // program pyta o liczbę
} while (wprowadzona liczba jest różna od zapamiętanej);
```

Przykład 3

Przejście przez ulicę.

```
do{
      // idź
} while(nie jesteś na drugiej stronie ulicy);
```

Zadanie 1

Dla dowolnego zbioru liczb wyświetlaj jego wartości tak długo, aż znajdziesz liczbę ujemną.

dla numbers = {72, 5, -6, 22,-9}
wyświetl:
72
5
-6

dla numbers = {9, -1, 55, 12, 9}
wyświetl:
9
-1

Zadanie 2*

Dla dowolnej liczby całkowitej podziel ją przez 2, wynik dzielenia (bez reszty) przypisz do tej liczby, resztę wyświetl. Wykonuj, dopóki liczba będzie równa 0.

np. dla **21**
21 / 2 = **10** reszta 1
10 / 2 = **5** reszta 0
5 / 2 = **2** reszta 1
2 / 2 = **1** reszta 0
1 / 2 = **0** reszta 1
Wyświetl: **10101**

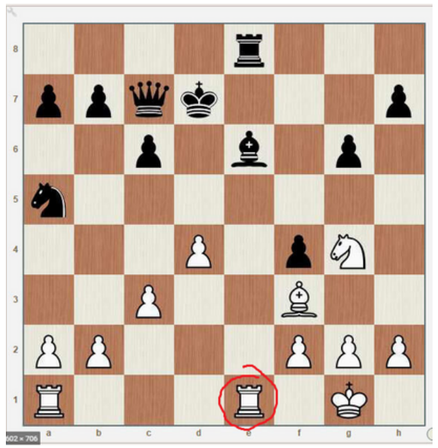
np. dla **7**
7 / 2 = **3** reszta 1
3 / 2 = **1** reszta 1
1 / 2 = **0** reszta 1
Wyświetl: **111**

np. dla **25**
25 / 2 = **12** reszta 1
12 / 2 = **6** reszta 0
6 / 2 = **3** reszta 0
3 / 2 = **1** reszta 1
1 / 2 = **0** reszta 1
Wyświetl: **10011**

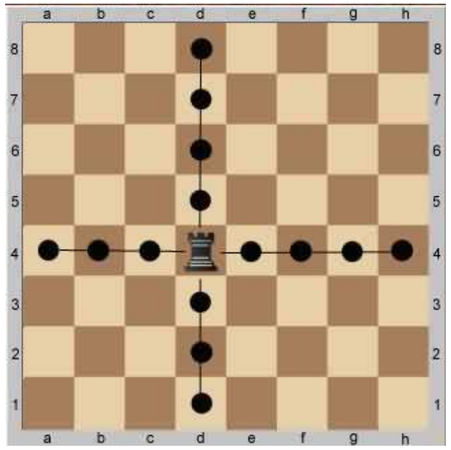
* wykonaj samodzielnie

Zadanie 3*

Gramy w szachy. Mamy stan taki jak poniżej (po lewej). Napisz algorytm, który sprawdzi jakie możliwości zbicia przeciwnika ma wieża (zanaczona na czerowono).



plansza gry



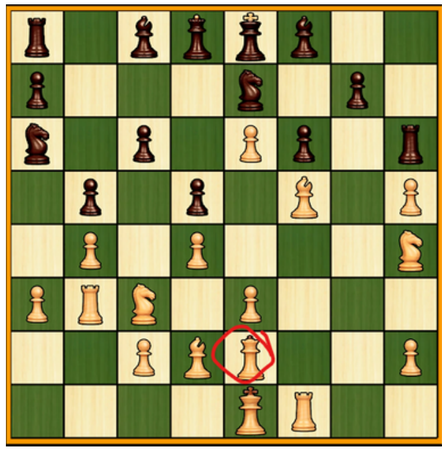
ruchy wieży

- wprowadź pionki przeciwnika (nie musisz rozróżniać figur)
- wprowadź swoje pionki (bez rozróżniania)
- poszukaj, jakich przeciwników ma w zasięgu wieża i wyświetl ich pozycje
- sprawdź, czy algorytm działa poprawnie gdy pionki inne niż wieża, zmienią położenie

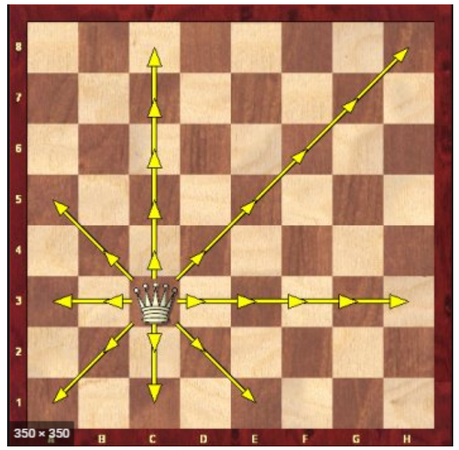
* wykonaj samodzielnie

Zadanie 4**

Gramy w szachy. Mamy stan taki jak poniżej (po lewej). Napisz algorytm, który sprawdzi jakie możliwości zbicia przeciwnika ma hetman (zanaczony na czerowono).



plansza gry



ruchy hetmana

- wprowadź pionki przeciwnika (nie musisz rozróżniać figur)
- wprowadź swoje pionki (bez rozróżniania)
- poszukaj, jakich przeciwników ma w zasięgu hetman i wyświetl ich pozycje
- sprawdź, czy algorytm działa poprawnie gdy pionki inne niż hetman, zmienią położenie

* wykonaj samodzielnie
** dodatkowe