I. Entrainement récusivité

I.1. Maximum

Algorithme principal :

Variables :

tab : un tableau d'entiers

i : un entier

Pour $i \in [\![1, \text{len}(\text{tab})]\!]$ **faire :**

$\text{tab}[i] \leftarrow$ lire()

Fin pour

Afficher maximum(tab, 0)

Procédure maximum(tab, k) **:**

Sémantique :

Entrée : un tableau et un entier k,

Sortie : le maximum du tableau commençant à l'indice k

Si $\text{len}(\text{tab}) = 0$ **alors Renvoyer** $+\infty$

Sinon si $k = \text{len}(\text{tab}) - 1$ **alors Renvoyer** $\text{tab}[k]$

Sinon Renvoyer $\max(\text{tab}[k], \text{maximum}(\text{tab}, k+1))$

I.2. Inverse d'un tableau

Algorithme principal :

Variables :

tab : un tableau de caractères

i : un entier

Pour $i \in [\![1, \text{len}(\text{tab})]\!]$ **faire :**

$\text{tab}[i] \leftarrow$ lire()

Fin pour

affiche_inverse(tab, 0)

Procédure affiche_inverse(tab, k) **:**

> **Sémantique :**
>
> > *Entrée* : un tableau de caractères et un entier k,
> > *Sortie* : rien
>
> **Si** $k = \mathrm{len}(\mathrm{tab}) - 1$ **alors**
>
> > **Afficher** $\mathrm{tab}[k]$
>
> **Sinon**
>
> > affiche_inverse($\mathrm{tab}, k + 1$)
> > **Afficher** $\mathrm{tab}[k]$
>
> **Fin si**

I.3. Palindrome

Algorithme principal :

> **Variables :**
>
> > tab : un tableau de caractères
> > tab' : un tableau de caractères
> > i : un entier
>
> **Pour** $i \in [\![1, \mathrm{len}(\mathrm{tab})]\!]$ **faire :**
>
> > $\mathrm{tab}[i] \leftarrow$ lire()
> > $\mathrm{tab}'[i] \leftarrow \mathrm{tab}[i]$
>
> **Fin pour**
>
> inverse($\mathrm{tab}, 0$) $\quad \triangleright \quad$ *on modifie* tab *en place*
> **Afficher** $\mathrm{tab} = \mathrm{tab}'$

Procédure inverse(tab, k) **:**

Sémantique :

Entrée : un tableau (partagé) de caractères et un entier k,
Sortie : rien,

Variables :

a : un caractère
n : un entier

$n \leftarrow \text{len}(\text{tab})$

Si $k < n/2$ **alors**

$a \leftarrow \text{tab}[k]$
$\text{tab}[k] \leftarrow \text{tab}[n-k]$
$\text{tab}[n-k] \leftarrow a$
inverse(tab, $k+1$)

Fin si

Autre solution :

Sémantique :

Entrée : une chaîne de caractères,
Sortie : la chaîne inversée,

Variables :

n : un entier,
a : un caractère,
w : une chaîne de caractères

$n \leftarrow \text{len}(\text{tab})$

Si $n = 0$ **alors Renvoyer** ""
Sinon

$a + w \leftarrow \text{tab}$ $\quad \triangleright$ *opération réalisée avec « queue »*
Renvoyer inverse$(w) + a$

Fin si

II. Dame et pions

On stocke le damier sous forme de matrice d'entiers : `0` pour vide (noté `.`), `1` pour un pion (noté `P`) et `2` pour une dame (notée `D`).

Différence avec les échecs (TD1.2) :

- on ne se déplace *que* sur les diagonales.
- on ne peut prendre que s'il y a une case de libre derrière.

II.1. Sans prises multiples

Algorithme principal

> **Variables :**
>
> > mat : matrice d'entiers (0, 1, 2),
> > x, y : deux entiers
>
> $\mathrm{mat} \leftarrow$ lecture()
> $(x, y) \leftarrow$ trouve_reine(mat)
> $\mathrm{mat}[x, y] \leftarrow 0$
> **Affiche** attaque(mat, x, y)

Procédure trouve_reine(mat)

> **Sémantique :**
>
> > *Entrée* : une matrice d'entiers
> > *Sortie* : une couple d'entiers
>
> **Variables :**
>
> > i, j : deux entiers
>
> **Pour** $i \in [\![0, 9]\!]$ **faire :**
>
> > **Pour** $j \in [\![0, 9]\!]$ **faire :**
> >
> > > **Si** $\mathrm{mat}[i, j] = 2$ **alors Renvoyer** (i, j)
> >
> > **Fin pour**
>
> **Fin pour**
>
> **Renvoyer** $(0, 0)$

Procédure attaque(mat, x, y)

Sémantique :

Entrée : une matrice d'entiers et deux entiers
Sortie : un entier

Variables :

a, b, c, d : quatre entiers

$a \leftarrow$ prise_direction($\mathrm{mat}, -1, -1, x, y$)
$b \leftarrow$ prise_direction($\mathrm{mat}, +1, -1, x, y$)
$c \leftarrow$ prise_direction($\mathrm{mat}, -1, +1, x, y$)
$d \leftarrow$ prise_direction($\mathrm{mat}, +1, +1, x, y$)

Renvoyer $\max(a, b, c, d)$

Procédure prise_direction($\mathrm{mat}, \Delta x, \Delta y, x, y$)

Sémantique :

Entrée : une matrice d'entiers (non partagée) et quatre entiers
Sortie : un entier

Variables :

attaqués : un entier
groupe : un autre entier

$\mathrm{attaqués} \leftarrow 0$

Tant que $0 \leq x < 10$ et $0 \leq y < 10$ et $\mathrm{attaqués} \leq 1$ **faire :**

Si $\mathrm{mat}[x, y] = 0$ **alors**
$\mathrm{attaqués} \leftarrow \mathrm{attaqués} + \mathrm{groupe}$
$\mathrm{groupe} \leftarrow 0$
Sinon si $\mathrm{mat}[x, y] = 1$ **alors**
$\mathrm{groupe} \leftarrow \mathrm{groupe} + 1$
$\mathrm{mat}[x, y] \leftarrow 0$
$x \leftarrow x + \Delta x$
$y \leftarrow y + \Delta y$
Fin si
Fin tant que

Renvoyer attaqués

II.2. Avec prises multiples

Seule la procédure « prise_direction » est modifiée par rapport à la section précédente.

Algorithme principal

> **Variables :**
>
> > mat : matrice d'entiers (0, 1, 2),
> > x, y : deux entiers
>
> $\mathrm{mat} \leftarrow$ lecture()
> $(x, y) \leftarrow$ trouve_reine(mat)
> $\mathrm{mat}[x, y] \leftarrow 0$
> attaque(mat, x, y)

Procédure trouve_reine(mat)

> **Sémantique :**
>
> > *Entrée* : une matrice d'entiers
> > *Sortie* : une couple d'entiers
>
> **Variables :**
>
> > i, j : deux entiers
>
> **Pour** $i \in [\![0, 9]\!]$ **faire :**
>
> > **Pour** $j \in [\![0, 9]\!]$ **faire :**
> >
> > > **Si** $\mathrm{mat}[i, j] = 2$ **alors Renvoyer** (i, j)
> >
> > **Fin pour**
>
> **Fin pour**
>
> **Renvoyer** $(0, 0)$

Procédure attaque(mat, x, y)

Sémantique :

Entrée : une matrice d'entiers et deux entiers
Sortie : un entier

Variables :

a, b, c, d : quatre entiers

$a \leftarrow$ prise_direction($\mathrm{mat}, -1, -1, x, y$)
$b \leftarrow$ prise_direction($\mathrm{mat}, +1, -1, x, y$)
$c \leftarrow$ prise_direction($\mathrm{mat}, -1, +1, x, y$)
$d \leftarrow$ prise_direction($\mathrm{mat}, +1, +1, x, y$)

Renvoyer $\max(a, b, c, d)$

Procédure prise_direction($\mathrm{mat}, \Delta x, \Delta y, x, y$)

Sémantique :

Entrée : une matrice d'entiers (non partagée) et quatre entiers
Sortie : un entier

Variables :

attaqués : un entier $\mathrm{attaqués}_{\max}$: un entier

$\mathrm{attaqués} \leftarrow 0$

Tant que $0 \leq x < 10$ et $0 \leq y < 10$ et $\mathrm{attaqués} \leq 1$ **faire :**

Si $\mathrm{mat}[x, y] = 0$ et $\mathrm{attaqués} = 1$ **alors**

$\mathrm{attaqués}_{\max} \leftarrow \mathrm{attaqués} +$ attaque(mat, x, y)

Tant que $0 \leq x < 10$ et $0 \leq y < 10$ et $\mathrm{mat}[x, y] = 0$ **faire**

$x \leftarrow x + \Delta x$
$y \leftarrow y + \Delta y$
$\mathrm{attaqués}_{\max} \leftarrow \max(\mathrm{attaqués}_{\max}, \mathrm{attaqués} +$ attaque(mat, x, y

Renvoyer $\mathrm{attaqués}_{\max}$

Sinon si $\mathrm{mat}[x, y] = 1$ **alors**

$\mathrm{attaqués} \leftarrow \mathrm{attaqués} + 1$
$\mathrm{mat}[x, y] \leftarrow 3$
$x \leftarrow x + \Delta x$
$y \leftarrow y + \Delta y$

Sinon

Arrêter la boucle.

Fin si

Fin tant que

Renvoyer 0