

Algorithmes et structures de données : TD 3corrigé

Types - Tableau 2D - Occupations de la mémoire

Exercice 3.1 *Types*

Déclarer des types qui permettent de stocker :

1. Les états de la matière : liquide, solide, gaz.

```
type t_etat = (liquide, solide, gaz);
```

2. Les 5 dés d'un jeu de yahtzee.

```
type t_yahtzee = array[1..5] of Byte;
```

Remarque: Etant donné que les dés peuvent uniquement porter les nombres de un à six, on aurait pu définir un "subrange type". Néanmoins, ce type occupe aussi un octet par dés de mémoire.

```
type t_des = 1..6;
```

```
type t_yatzee = array[1..5] of t_des;
```

3. Une image noir et blanc de 512x512 pixels.

```
type t_pixel = (noir, blanc);
```

```
type t_image = array[1..512] of array[1..512] of t_pixel;
```

Remarque: Etant donné que la plus petite unité d'un type de Pascal à stocker est de 1 octet, cette image occupe 512*512 octets de mémoire. Par conséquent, il est courant de stocker 8 pixel de la même ligne dans un octet, et l'image n'occupera uniquement 64*512 octets de mémoire :

```
type t_image = array[1..64] of array[1..512] of Byte;
```

4. Un damier d'échec.

```
type t_figure = (vide, roi, dame, tour, fou, cavalier, pion);
```

```
type t_couleur = (vide, blanc, noir);
```

```
type t_champ = RECORD
```

```
    couleur : t_couleur;
```

```
    figure : t_figure;
```

```
END;
```

```
type t_damier = array[1..8] of array[1..8] of t_champ;
```

Remarque: Au moment du TD, les structures hétérogènes de type RECORD n'ont pas encore été introduites en cours. Une solution possible sans l'utilisation des structures hétérogènes est de déclarer deux variables d'un même type `t_damier` :

```

type t_figure = (vide, roi, dame, tour, fou, cavalier, pion);
type t_damier = array[1..8] of array[1..8] of t_figure;
var damier_blanc, damier_noir : t_damier;

```

Exercice 3.2 *Tableau 1D*

Considérez le tableau 1D suivant :

```

type t_champ = (vide,vert,rouge);
var tableau : array[1..6] of t_champ;

```

1. Ecrire un algorithme qui compte le nombre d'entrées de couleur verte dans le tableau.

```

verte := 0;
pour i de 1 à 6 faire
    si tableau[i] = vert alors
        verte := verte + 1;
    fin si
fin pour
afficher "Nombre d'entrées de couleur verte : "
afficher verte;

```

2. Ecrire un algorithme qui affiche à l'écran la couleur qui apparaît plus souvent dans le tableau ou "égalité" sinon. Faites tourner votre algorithme dans un tableau.

```

verte := 0;
rouge := 0;
pour i de 1 à 6 faire
    si tableau[i] = vert alors
        verte := verte + 1;
    sinon si tableau[i] = rouge alors
        rouge := rouge + 1;
    fin si
fin pour
si verte > rouge alors
    afficher "Il y a plus d'entrée de couleur verte."
sinon si rouge > verte alors
    afficher "Il y a plus d'entrée de couleur rouge."
sinon
    afficher "Il y a égalité"
fin si

```

3. Ecrire un algorithme qui parcourt le `tableau` et qui identifie la longueur maximale d'une suite d'entrées de `champ` rouge consécutive. Utilisez trois variables `i`, `longueur`, `max_longueur`. Faites tourner votre algorithme dans un tableau.

```

longueur := 0;
max_longueur := 0;

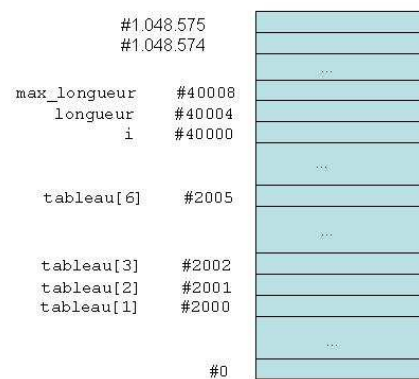
```

```

pour i de 1 à 6 faire
    si tableau[i] = rouge alors
        longueur := longueur + 1;
        si longueur > max_longueur alors
            max_longueur := longueur;
        fin si
    sinon
        longueur := 0;
    fin si
fin pour

```

4. Ebaucher l'occupation de la mémoire d'un ordinateur avec 1 Mo de mémoire vive.



Exercice 3.3 *Tableau 2D*

Considérez le damier d'un jeu de dame :



1. Définissez les types pour stocker le damier.

```

type t_champ = (vide, noir, blanc);
var damier : array[1..8] of array[1..8] of t_champ;

```

2. Ebaucher l'occupation de la mémoire d'un ordinateur avec 1 Mo de mémoire vive.

	#1 048 575	
	#1 048 574	
		...
		
		...
damier[8][8]	#2063	
...		
damier[2][1]	#2008	
...		
damier[1][3]	#2002	
damier[1][2]	#2001	
damier[1][1]	#2000	
		
	#0	

3. Ecrire un algorithme qui initialise le damier de dame en posant les pions noirs et blanc sur le damier conforme à la Figure.

```

pour i de 1 à 3 faire
    pour j de 1 à 8 faire
        si (i+j) MOD 2 = 0 alors
            damier[i][j] := blanc;
        else
            damier[9-i][j] := noir;
        fin si
    fin pour
fin pour

```

4. Ecrire un algorithme qui répond à l'écran à la question "qui est en train de gagner", c'est-à-dire de quelle couleur il y a plus de pions sur le damier.

```

blanc := 0;
noir := 0;
pour i de 1 à 8 faire
    pour j de 1 à 8 faire
        si damier[i][j] = blanc alors
            blanc := blanc + 1;
        sinon si damier[i][j] = noir alors
            noir := noir + 1;
        fin si
    fin pour
fin pour
si blanc>noir alors
    afficher "Blanc est en train de gagner."
sinon si noir>blanc alors
    afficher "Noir est en train de gagner."
sinon
    afficher "Il y a égalité"
fin si

```

Exercice 3.4 *Enregistrements*

1. Déclarer un type hétérogène `t_entree` qui est composé d'une année, d'un mois, d'un jour, d'une heure et d'une minute.

```
type t_entree = RECORD
    annee      : integer;
    mois       : byte;
    jour       : byte;
    heure      : byte;
    minutes    : byte;
END;
```

2. Considérer la déclaration des deux variables suivantes :

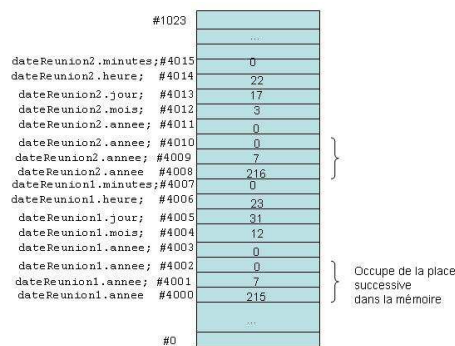
```
var dateReunion1 : t_entree
var dateReunion2 : t_entree
```

Affecter les deux variables avec la date du 31 décembre 2007 à 23h00 ainsi que la date du 17 mars 2008 à 22h00.

```
dateReunion1.jour = 31;
dateReunion1.mois = 12;
dateReunion1.annee = 2007;
dateReunion1.heure = 23;
dateReunion1.minute = 0;
```

```
dateReunion2.jour = 17;
dateReunion2.mois = 3;
dateReunion2.annee = 2008;
dateReunion2.heure = 22;
dateReunion2.minute = 0;
```

3. Ebaucher l'occupation de la mémoire d'un ordinateur avec 1024 octets de mémoire vive.



Exercice 3.5 *Fonctions*

1. Déclarer une fonction s'appelant `carre` qui calcule et renvoie le carré d'un nombre entier qui est passé en paramètre.
2. Déclarer 2 variables `x` et `res` du type nombre entier. Affecter la variable `res` avec le carré de `x` en appelant la fonction `carre`.

```
var x,res : integer;

function carre(a : integer) : integer;
début
    result := a * a; { la fonction renvoie le carré de son paramètre }
fin

debut
    res := carre(x);
fin
```

Les exercices sont à rendre dans le prochain TD.
--