Algorithmique et structures de données 2017-18

Contenu du cours 3 du 5.10.2017

1. Tableaux non contraints à une dimension en Ada
   - Un tableau de nombres flottants :
     ```
     type T_Tab_Flt is array (Integer range <>) of Float;
     X : T_Tab_Flt(-7..7);
     ```
   - Recherche d'un indice de la valeur minimale d'un tableau de `Float` et permutation avec l'élément en 1ère position
   - Répétition de l'opération précédente sur des tranches du tableau
   - Tri par sélection avec illustration sur un exemple
   - Exemple de type tableau non-contraint prédéfini :
     ```
     type String is array(Positive range <>) of Character;
     ```
   - Algorithme de vérification que deux mots sont des anagrammes en utilisant un tri
2. Tableaux non contraints à deux dimensions en Ada
   - Image noir et blanc
     ```
     type T_Image_NB
          is array(Natural range <>,Natural range <>) of Boolean;
     ```
   - Image en niveaux de gris
     ```
     type T_Octet is mod 256;
     type T_Image_Gris
          is array(Natural range <>,Natural range <>) of T_Octet;
     ```
     - Négatif d'une image
   - Initialisation et/ou afffectation avec un agrégat
     ```
     type T_Image_NB
          is array(Natural range <>,Natural range <>) of Boolean;
     X : T_Image_NB :(0..7,0..7):=(0 => (others => true),
                                   others => (others => false));
     ```
   - Travail pratique de la « couverture de la reine »
     ```
     type T_Case is (Vide,Prise,Reine);
     type T_Echiquier is array(Positive range <>,
                               Character range <>) of T_Case;
     Aire_Jeu : T_Echiquier(1..8,'a'..'h')
                     := (3 => ('d' => Reine, others => Vide),
                         others => (others => Vide));
     ```
3. Les fonctions
   - Exemples : cube d'un nombre réel $f(x) = x^3$, parité d'un entier
   - Appel de fonction
   - Schéma (boîte noire) des paramètres en entrée et de la valeur de retour
     - notions de variable (nom, emplacement, contenu)
     - paramètre formel (représente une variable, constante, expression, comme une variable locale)
     - paramètre effectif (c'est une variable, constante, expression ; remplace le paramètre formel dans la définition de la fonction à l'appel de celle-ci)
     - Exemples
       ```
       type T_Stat is array(Integer range <>) of Float;
       function Random_Tab return T_Stat;
       function Minimum(X : T_Stat) return T_Stat;
       ```

- Syntaxe

```
function Nom_Fonction(Param_1 : Type_1;
                      Param_2 : Type_2;
                      …
                      Param_K : Type_K) return Type_Retour is
   -- Déclarations
begin
   -- Instructions
   return Expression;
end Nom_Fonction;
```

4. Les procédures
   - Schéma (boîte noire) des paramètres en entrée `in`, en sortie `out` et en entrée-sortie `in out`
   - Exemples : `Get(N);  Put(N);  Get_Line(Nom, Taille);`
   - Syntaxe

```
procedure Nom_Procedure(Param_1 : in     Type_1;
                        Param_2 : in     Type_2;
                        Param_3 :    out Type_3;
                        Param_4 : in out Type_4) is
   -- Déclarations
begin
   -- Instructions
end Nom_Procedure;
```

   - Exemples: affichage des valeurs d'un tableau, remplissage d'un tableau avec des nombres aléatoires
   - Exemple complet: incrément d'un nombre entier
   - Un paramètre effectif "`out`" doit être une variable, pas une expression, ni une constante:

```
Put(3*N*N); -- correct, car paramètre en "in"
Get(3*N); -- FAUX! car paramètre en "out"
```

   - Une procédure qui échange 2 valeurs
   - Pseudo-code du tri par sélection

```
procédure tri_sélection(tableau t(1 à n))

début
   pour i de 1 à n-1
      min := index_min(t(i à n))
      si min ≠ i alors
         échanger(t[i],t[min])
      fin si
   fin pour
fin tri_sélection
```

5. Différence entre fonctions et procédures, syntaxe sans paramètre (pas de parenthèses)