Exemples Cours n°6 - Les types en Ada

- Spécifier et écrire un sous-programme permettant de saisir les éléments d'un tableau

```
Bmax : constante entier <- 50
Type T_Tab Est Tableau(1..Bmax) de Entier

Procedure Lect_Tab ( FT : OUT T_Tab ; Fnb : OUT entier ) Est
        nb : entier -- nombre d'éléments ou taille effective du tableau
Debut
        -- lire nb de manière fiable et conviviale
        Repeter
                Ecrire("Saisir le nombre d'éléments du tableau")
                Lire(nb)
        Jusqu'a nb >= 1 ET nb =< Bmax
        -- lecture des éléments
        Pour i de 1 à nb Faire
                Ecrire("Saisir un nombre")
                Lire(FT(i))
        Fin Pour
        Fnb <- nb
Fin Lect_Tab
```

- Spécifier et écrire un sous-programme permettant d'afficher les éléments d'un tableau

```
Bmax : constante entier <- 50
Type T_Tab Est Tableau(1..Bmax) de Entier

Procedure Affich_Tab ( FT : IN T_Tab ; Fnb : IN entier ) Est
Debut
        Pour i de 1 à Fnb Faire
                Ecrire(FT(i))
        Fin Pour
Fin Affich_Tab
```

- Spécifier, définir une fonction qui recherche la 1ère occurence d'un entier e dans un tableau T d'entiers à une seule dimension, initialisé avec *nb_elements*.

```
Bmax : constante entier <- 50
Type T_Tab Est enregistrement
        Le_Tab : Tableau(1..Bmax) de Entier
        Nb_el : entier ∈ [1..Bmax]
Fin enregistrement

Nom : Prem_Occ
Sémantique : Recherche 1è occurence d'un élément dans un tableau
Paramètres : FT : T_Tab (IN) -- le tableau au type record
                Fe : Entier (IN) -- élément recherché
Type retour : Entier
Pré-condition : Tableau rempli jusqu'à FT.Nb_el >= 1
Post-condition : ( FT.Le_Tab(i) = Fe et FT.Le_Tab(i) 1è occ et i ∈ [1..FT.Nb_el] => resultat = i ) OU
                ( Fe ∉ FT.Le_Tab[1..FT.Nb_el] => resultat = 0 )
```

- Ecrire les instructions permettant de déterminer si un mot rangé dans un tableau, à 1 dimension, de caractères est un palindrome. Le début du mot n'est pas nécessairement dans la 1è case du tableau et la fin du mot n'est pas nécessairement dans la dernière case du tableau.

```
Bmax : constante entier <- 50
Type T_Tab Est enregistrement
        Le_Tab : Tableau(1..Bmax) de Entier
        Nb_el : entier ∈ [1..Bmax]
Fin enregistrement

Nom : Is_Palindrome
Sémantique : Indique si un mot m = T(d..f) rangé dans un tableau T est un palindrome
Paramètres : FT : T_Tab (IN) -- tableau contenant le mot
                Fdeb : entier (IN) -- indice du début du mot
                Ffin : entier (IN) -- indice de fin du mot
Type retour : booléen
Pré-condition : Fdeb < Ffin et Fdeb >= 1 et Ffin =< Bmax
Post-condition : Resultat si palindrome Non Resultat sinon
Tests

Fonction Is_Palindrome ( FT : IN T_Tab ; Fdeb : IN entier ; Ffin : IN entier ) Retourne booleen Est
        fin : entier -- borne supérieure du sous-tableau
        deb : entier -- borne inférieure du sous-tableau
        palin : booleen -- resultat indiquant si palindrome ou non
Debut
        (*R0 Indique si un mot m = T(d..f) rangé dans un tableau T est un palindrome *)
        palin <- Vrai
```

```
        deb <- Fdeb
        fin <- Ffin
        Repeter
                (*R1 Comparer les 2 bouts du sous-tableau non exploré *)
                Si FT.Le_Tab(deb) /= FT.Le_Tab(fin) Alors
                        palin <- Faux
                Sinon
                        fin <- fin-1
                        deb <- deb+1
                Fin Si
        Jusqu'a Non palin OU fin =< deb
        Retourne palin
Fin Is_Palindrome
```

- Ecrire les instructions permettant de rechercher l'indice d'un minimum d'un tableau *T* d'entiers à une seule dimension, initialisé avec *nb_elements*.

Nom : Minimum
Sémantique : Recherche de minimum dans un tableau de borne inf à borne sup
Paramètres : FT : T_Tab (IN/OUT) -- tableau où chercher minimum
Fdeb : ENTIER (IN) -- borne inf
Ffin : ENTIER (IN) -- borne sup
Type retour : Entier
Pré-condition : Fdeb'Avant <= Ffin'Avant ET
Ffin'Avant <= Capacite ET
Fdeb'Avant >= 1
Post-condition : Resultat = Minimum pour chaque terme du tableau avant de j=Fdeb'Avant à Ffin'Avant

Permuter(FT,Fi,Fm)
Post-condition : (∀k ≠ Fi ET k ≠ Fm : FT(k) = FT'Avant(k)) ET
(FT(Fi'Avant) = FT'Avant(Fm'Avant)) ET
(FT(Fm'Avant) = FT'Avant(Fi'Avant))

- Ecrire les instructions permettant de trier un tableau *T* d'entiers à une seule dimension, initialisé avec *nb_elements*. On utilisera pour cela le tri par sélection.

Capacite : constante ENTIER <- 50
Type T_Tab est TABLEAU(1..Capacite) de ENTIER

Nom : Tri_Selection
Sémantique : Tri un tableau par ordre croissant à partir d'une borne inf à une borne sup
Paramètres : FT : T_Tab (IN/OUT) -- tableau à trier
Fdeb : ENTIER (IN) -- borne inf de FT
Ffin : ENTIER (IN) -- borne sup de FT
Pré-condition : Fdeb'Avant <= Ffin'Avant ET
Ffin'Avant <= Capacite ET
Fdeb'Avant >= 1

Post-condition : $\forall i \in$ [Fdeb'Avant .. Ffin'Avant-1], FT(i) <= FT(i+1) ET
FT = Perm(FT'Avant) -- le tableau après est une permutation du tableau
Tests : T'Avant = [4 5 2]
T = [2 4 5]

```
Procedure Tri_Selection ( FT : IN/OUT T_Tab ) Est
    Min : ENTIER
Debut
_____(*R0 Tri un tableau par ordre croissant à partir d'une borne inf à une borne sup *)
    Pour i de Fdeb à Ffin-1 Faire
        (*R1 rechercher minimum de FT(i .. Ffin) *)
        -- i >= Fdeb
        Min <- Minimum(FT,i,Ffin)
        (*R1 permuter minimum avec FT(i) *)
        Permuter(FT,i,Min)
    Fin pour
Fin Tri_Selection
```

La méthode des raffinages successifs est la suivante :

R0	Tri un tableau par ordre croissant à partir d'une borne inf à une borne sup
R1	(*R0 Tri un tableau par ordre croissant à partir d'une borne inf à une borne sup *) Pour i de Fdeb à Ffin-1 Faire rechercher minimum de FT(i .. Ffin) permuter minimum avec FT(i) Fin pour
R2	(*R0 Tri un tableau par ordre croissant à partir d'une borne inf à une borne sup *) Pour i de Fdeb à Ffin-1 Faire (*R1 rechercher minimum de FT(i .. Ffin) *) -- i >= Fdeb Min <- Minimum(FT,i,Ffin) (*R1 permuter minimum avec FT(i) *) Permuter(FT,i,Min) Fin pour