License Informatique L3
Module ABP
Devoirs 2
Etudiant Régis Azoulay

Exercice 1

Question 1 : Ecrire une procédure "RechercheMiniMaxi" qui permet de rechercher la plus petite valeur MINI et la plus grande valeur MAXI du tableau TAB. On n'écrira qu'une seule boucle pour effectuer cette recherche.

```
PROCEDURE RechercheMiniMaxi (VARIABLE TAB:TYPTAB;
                             DIM:ENTIER;
                             VARIABLE MINI:ENTIER;
                             VARIABLE MAXI : ENTIER );
DEBUT {RechercheMiniMaxi}
   MINI <- TAB[1] ;
   MAXI <- TAB[1] ;
   POUR I VARIANT DE 2 A DIM FAIRE
      SI (TAB[I] > MAXI)
      ALORS
         MAXI <- TAB[I] ;
      FIN SI;
      SI (TAB[I] < MINI)
      ALORS
         MINI <- TAB[I] ;
      FIN SI ;
   FAIT ;
   RENVOYER MINI, MAXI ;
FIN ; {RechercheMiniMaxi}
```

Remarque :

Cette procédure a été écrite selon l'exemple donné dans la consigne à savoir :
Exemple : Pour le tableau TAB défini précédemment, RechercheMiniMaxi (TAB, DIM, MINI, MAXI) va **renvoyer** les valeurs : MINI = 0 et MAXI = 23.

J'ai donc renvoyé les valeurs MINI et MAXI. Cependant on aurait pu écrire cet procédure sans l'instruction RENVOYER : en effet, puisque MINI et MAXI sont passées à la procédure par VARIABLE (par référence) elles comporteront donc leurs valeurs réspéctive après l'appel de la procédure.

On aurait pu aussi déclarer et initialiser les variables MINI et MAXI dans le corps de la procédure, en réduisant la portée de ces deux variables à leur minimums ce qui permetterait d' économiser sur la mémoire globale du programme. Dans ce cas la signature de la procédure se voirait réduite de ces 2 paramètres et l'instruction RENVOYER prendrait alors tout son sens.
Cependant dans un souci de respect de l'exemple j'ai donc écrit cette instruction et les paramètres tel qu'il était demandé.

Question 2 : Ecrire une procédure "RechercheIndiceMiniMaxi" qui recherche l'indice INDMINI de la plus petite valeur et l'indice INDMAXI de la plus grande valeur du tableau TAB entre les indices IND1 et IND2.

```
PROCEDURE RechercheIndiceMiniMaxi (VARIABLE TAB:TYPTAB;
                                   IND1:ENTIER;
                                   IND2:ENTIER;
                                   VARIABLE INDMINI:ENTIER;
                                   VARIABLE INDMAXI:ENTIER );

DEBUT {RechercheIndiceMiniMaxi}
   INDMINI <- IND1 ;
   INDMAXI <- IND1 ;
   POUR I VARIANT DE IND1 A IND2 FAIRE
      SI (TAB[I] > TAB[INDMAXI])
      ALORS
         INDMAXI <- I ;
      FIN SI;
      SI (TAB[I] < TAB[INDMINI])
      ALORS
         INDMINI <- I;
      FIN SI ;
   FAIT ;
   RENVOYER INDMINI, INDMAXI ;
FIN; {RechercheIndiceMiniMaxi}
```

Remarque :

De la meme manière qu'à la question précédente, dans un souci de respect de l'exemple fourni à savoir :

Exemple : Pour le tableau TAB défini précédemment, si l'on suppose que IND1 = 6 et IND2 = 9, RechercheIndiceMiniMaxi (TAB, DIM, IND1, IND2, INDMINI, INDMAXI) va renvoyer les valeurs : INDMINI = 6 et INDMAXI = 8.

J'ai renvoyé INDMINI, INDMAXI. Mais on aurait pu se passer de cette instruction ou initialiser ces variables dans le corps de la procédure (Cf. Remarque Question 1).

Question 3 : Adapter l'algorithme du tri par sélection dans une procédure "Tri" (qui prendra en paramètres le tableau TAB et sa taille DIM) pour qu'à chaque parcours du tableau, on mette la plus petite valeur à sa place et la plus grande valeur à sa place (au lieu de ne traiter que la plus petite). On utilisera la procédure "RechercheIndiceMiniMaxi".

```
PROCEDURE Tri (VARIABLE TAB:TYPTAB, DIM:ENTIER);
VARIABLE
   INDMINI:ENTIER;
   INDMAXI:ENTIER;
   IND1:ENTIER;
   IND2:ENTIER;
   TEMP:ENTIER;
DEBUT {Tri}

   IND1<-1;
   IND2<-DIM;

   POUR I VARIANT DE 1 A DIM FAIRE

      INDMINI, INDMAXI <- RechercheIndiceMiniMaxi(TAB, IND1, IND2, INDMINI,
INDMAXI);

      TEMP <- TAB[IND1];
      TAB[IND1] = TAB[INDMINI];
      TAB[INDMINI] = TEMP;

      TEMP <- TAB[IND2];
      TAB[IND2] = TAB[INDMAXI];
      TAB[INDMAXI] = TEMP;

      IND1 <- IND1 + 1;
      IND2 <- IND2 - 1;
   FAIT;
FIN; {Tri}
```

Exercice 2

Ecrire une procédure "ProduitMatrice" qui calculera le produit des deux matrices A et B dans la matrice C.

```
PROCEDURE ProduitMatrice (A:TYPEMAT;
                          B:TYPEMAT;
                          VARIABLE C:TYPEMAT;
                          DIM:ENTIER);
VARIABLE
   I,J, K : ENTIER;
DEBUT {ProduitMatrice}
   I <-0;
   J <-0;
   K <-0;

   POUR I VARIANT DE 1 A DIM FAIRE
      POUR J VARIANT DE 1 A DIM FAIRE
         C [I,J] <- 0;
         POUR I VARIANT DE K A DIM FAIRE
            C [I,J] <- C [I,J] + A[I,J] * B[I,J];
         FAIT;
      FAIT;
   FAIT;

FIN; {ProduitMatrice}
```

Exercice 3

Question 1 : Ecrire une fonction "CalculDizaine" qui calcule le numéro de la dizaine dans laquelle se trouve un nombre entier.

```
FONCTION CalculDizaine (NB:ENTIER) : ENTIER;
VARIABLE
   RESTE : ENTIER;
DEBUT {CalculDizaine}
   RESTE <- NB MOD 10;
   RENVOYER (NB - RESTE) / 10;
FIN; {CalculDizaine}
```

Question 2 : Ecrire une procédure "Transformation" qui permettra de remplir la matrice MAT à partir du tableau TAB. On sait que cette matrice possède cinq lignes, mais on ne connaît pas encore son nombre de colonnes NBCOL. Après avoir calculé NBCOL, il s'agira d'examiner chaque élément de TAB un par un et pour chaque élément, on procédera de la manière suivante :
- on regarde dans quelle dizaine il se situe (on utilisera pour cela la fonction "CalculDizaine"),
- on insère l'élément dans la matrice à sa place dans la ligne correspondant à sa dizaine. Pour cela, on parcourt la ligne, et dès que l'on a trouvé la place de l'élément, on décale tous les éléments plus grands vers la fin de la ligne afin de pouvoir insérer cet élément.

```
PROCEDURE Transformation (TAB:TYPTAB; DIM:ENTIER; VARIABLE MAT:TYPEMAT);
VARIABLE
   NBCOL : ENTIER;
   LIGNE_DIZAINE : ENTIER;
   TEMP_SUP : ENTIER;
   TEMP_INF : ENTIER;
   I,J : ENTIER;
   FIN_AFFECTATION_TRI : BOOLEEN;
DEBUT {Transformation}
   NBCOL <- DIM / 5;
   POUR I VARIANT DE 1 A DIM FAIRE
      LIGNE_DIZAINE <- CalculDizaine(TAB [I]);
      TEMP_SUP <- TAB[I];
      FIN_AFFECTATION_TRI <- FAUX;
      POUR J VARIANT DE 0 A NBCOL FAIRE
         SI (FIN_AFFECTATION_TRI = FAUX)
         ALORS
            SI (MAT[LIGNE_DIZAINE, J] != 50)
            ALORS
               SI (TEMP_SUP < MAT[LIGNE_DIZAINE, J])
               ALORS
                  TEMP_INF <- TEMP_SUP;
                  TEMP_SUP <- MAT[LIGNE_DIZAINE, J];
                  MAT[LIGNE_DIZAINE, J] <- TEMP_INF;
               FIN SI;
            SINON
               MAT[LIGNE_DIZAINE, J] <- TEMP_SUP;
               FIN_AFFECTATION_TRI = VRAI;
            FIN SI;
         FIN SI;
      FAIT;
   FAIT;
FIN; {Transformation}
```