R101 : Initiation au développement

TD n° 7 – Boucle "pour" - Tableaux

Exercice 1 (Boucles "pour")

1. Question Proposer une procédure (ou une fonction ?) écrite en langage algorithmique qui affiche la table de multiplication (de 0 à 9) de l'entier n, n étant fourni en paramètre. Que vaut l'indice de boucle en fin de procédure/fonction ?
2. Question Utiliser cette procédure (ou fonction ?) dans un programme.

Exercice 2

On considère les définitions d'une constante et d'une variable :

```
constante entier MAX := 100 ;
tab : tableau[MAX] de entier ;
```

1. Question

a. Proposer en langage algorithmique la définition d'une procédure permettant de saisir au clavier toutes les valeurs de la variable tableau tab. On utilisera pour cela une boucle "pour".

```
procédure remplirTableau(sortF tab : tableau[] de entier)c'est ...
```

b. Modifier la procédure pour saisir une partie seulement des valeurs du tableau ; le nombre nbElt d'éléments qui seront saisis sera un paramètre de la procédure (nbElt <= MAX).

c. Proposer un programme pour tester la procédure remplirTableau().

2. Question

a. Écrire une procédure afficheTableau() utilisant une boucle "pour", afin d'afficher à l'écran toutes les valeurs existantes du tableau tab.

```
procédure afficheTableau(entF tab : tableau[] de entier, entF nbElt : entier) c'est...
```

b. Compléter le programme principal pour tester cette procédure.

3. Question

a. Écrire la fonction fNbFois() comptant, dans le tableau tab donné en paramètre avec son nombre d'éléments réellement remplis, le nombre de valeurs égales à une valeur n également donnée en paramètre.

b. Compléter le programme principal pour tester cette fonction.

4. Question

a. Écrire la procédure remplace() permettant de remplacer dans le tableau tab, donné en paramètre avec son nombre d'éléments, toutes les valeurs 0 par des 1 et de transmettre le tableau modifié au programme appelant.

b. Compléter le programme principal pour tester cette procédure remplace().

5. Question

a. On propose la procédure incrEntier() permettant d'incrémenter un entier de 1 unité. Appeler cette procédure dans le programme principal de manière à incrémenter le troisième élément du tableau tab.

```
procédure incrEntier(entF/sortF n : entier) c'est
début
    n := n+1;
fin
```

6. Question

a. Écrire en langage algorithmique la fonction booléenne existe() qui , à partir d'une valeur n, du tableau tab et de son nombre d'éléments tous trois donnés en paramètre, délivre vrai si n est présente dans tab et faux sinon.

b. Compléter le programme principal précédent pour tester cette fonction existe().

7. Question

Écrire la fonction fDouble() indiquant si le tableau tab transmis en paramètre contient au moins deux fois la même valeur.

8. Question

Proposer une procédure pour rechercher la plus grande valeur du tableau tab, ainsi que sa plus petite valeur et la moyenne de l'ensemble des valeurs.

9. Question

Proposer une procédure pour rechercher et effacer toutes les valeurs 0 du tableau tab, déplacer les éléments restants pour ne pas laisser de cases vides et mettre à jour le nombre d'éléments effectifs dans le tableau.

Exercice 3

Soient les déclarations de constante et de type :

```
constante entier MAX := 100 ;
type typeTab = tableau[MAX] de entier ;
```

On souhaite développer un programme qui :

- déclare un tableau de type typeTab,
- initialise toutes les valeurs de ce tableau à 0, à l'aide d'une procédure initTab(),
- effectue successivement MAX fois :
  - la saisie d'une valeur entière positive, à l'aide d'une fonction saisieEntPos() (la saisie est redemandée à l'utilisateur tant qu'il n'a pas rentré une valeur positive)
  - l'insertion de cet entier dans le tableau, à l'aide d'une procédure insereEntTab() permettant de ranger les entiers par ordre décroissant. Pour cela cette procédure pourra procéder si nécessaire à un décalage de certains éléments du tableau (cf. exemple ci-dessous).
  - affiche à l'écran le contenu du tableau, à l'aide d'une procédure afficheTab().

Exemple :

a. Tableau après initialisation :

T =										
	0	0	0	0	0	0	0	0	0	0

b. Tableau après la saisie de la valeur 5 :

T =										
	5	0	0	0	0	0	0	0	0	0

c. Tableau après la saisie de la valeur 19 :

T =										
	19	5	0	0	0	0	0	0	0	0

d. Tableau après la saisie de la valeur 2:

T =										
	19	5	2	0	0	0	0	0	0	0

e. Tableau après la saisie de la valeur 15 :

T =										
	19	15	5	2	0	0	0	0	0	0

f. Etc …

2. Question1/

a. Proposer en langage algorithmique le programme principal, avec l'appel des différentes fonctions et procédures.

b. Proposer en langage algorithmique la procédure initTab().

c. Proposer en langage algorithmique la procédure insereEntTab().

d. Proposer en langage algorithmique la fonction saisieEntPos().

e. Proposer en langage algorithmique la procédure afficheTab().

Exercice complémentaire 1

Ecrire un programme simulant le jeu de bataille entre 2 joueurs, avec un jeu de cartes de 32 cartes.
Le principe du jeu est le suivant : tout d'abord le jeu est mélangé puis chaque joueur reçoit la moitié des cartes. Les tours de jeu peuvent alors commencer. À chaque tour, chaque joueur retourne la carte du haut de son tas. Celui qui a la carte la plus haute gagne l'ensemble des cartes mises en jeu qui sont alors placées en dernier dans son tas. En cas d'égalité des valeurs des cartes - cas appelé bataille - les joueurs commencent par placer une première carte face cachée puis une seconde face visible pour décider qui gagnera le tour de jeu. En cas de nouvelle égalité, la procédure est répétée jusqu'à ce que le coup soit remporté.

Le joueur qui a emporté toutes les cartes est déclaré gagnant en fin de jeu.
Un tableau d'entiers représentera le jeu de cartes, ainsi que le jeu de chaque joueur. Chaque tableau contiendra 8 éléments, la valeur 1 correspondant à la carte 7, celle de 8 correspondant à un as. Dans cet exercice, les couleurs des cartes ne seront pas modélisées.
Le programme devra afficher le déroulement du jeu (cartes jouées et vainqueur de chaque tour) puis donner le vainqueur final. La structure de programme suivant est proposée :

```
Programme jeu_de_bataille c'est
constante entier N := 32 ;
type typeJeu = tableau [N] de entier;
Début
    jeuCartes : typeJeu ;       //jeu de cartes complet
    jeuJoueur1 : typeJeu ;      //cartes du joueur 1
    jeuJoueur2 : typeJeu ;      //cartes du joueur 2
    n1 : entier   //indice de la dernière carte du joueur 1
    n2 : entier   //indice de la dernière carte du joueur 2

    // on remplit le jeu avec les N cartes
    initJeu (sortF jeuCartes) ;
    // on mélange les N cartes
    mélangeJeu (entF/sortF jeuCartes) ;
    // on distribue la moitié des cartes aux 2 joueurs
    distribueJeu (entF/sortF jeuCartes, sortF jeuJoueur1, sortF jeuJoueur2) ;
    // on joue les plis
    tant que (//la partie n'est pas finie) faire
        jouer1pli (entF/sortF jeuJoueur1, entF/sortF n1, entF/sortF jeuJoueur2,entF/sortF n2) ;
    fin_tant_que
    // on affiche le gagnant
    afficheGagnant (entF n1, entF n2) ;
Fin
```

Informations complémentaires :

- Procédure mélangeJeu():
  - Principe : pour mélanger les N cartes (valeurs d'un tableau), on parcours le tableau et à chaque pas, on échange la valeur courante avec une autre valeur du tableau pointée au hasard.
  - Utilisation d'une fonction générant un entier aléatoire compris entre 0 et N :

```
aleaN(entF N) délivre entier
```

- Procédure jouer1pli() :
  - gère le résultat d'un tour de jeu, avec ou sans bataille (s)
  - affiche les cartes jouées et le résultat obtenu

Exercice complémentaire 2

On vous propose le programme suivant.

a. *À quoi correspond typeTab ?*

b. *Que réalise ce programme ?*

```
/* Programme principal */
programme Principal
        constante entier CMAX := 100000;
        TYPE typeTab = tableau [CMAX] de entier;
debut
        tabDeTravail : typeTab;
        sonMaximum : entier;
        lecture(entF/sortF tabDeTravail, entF/sortF sonMaximum);
        Calc(entF/sortF tabDeTravail, entF sonMaximum);
        affiche(entF tabDeTravail, entF sonMaximum);
fin// du programme principal

/* Procédure de lecture */
procedure lecture(entF/sortF tab : typeTab, entF/sortF longueur : entier) c'est
debut
        uneValeur: entier;
        EcrireEcran("donner des nombres positifs dont le dernier est zéro");
        longueur:= 0;
        LireClavier(uneValeur);
        tantque ((uneValeur != 0) et (longueur < CMAX)) faire
                longueur:= longueur + 1;
                tab [longueur] := uneValeur;
                lireClavier(uneValeur);
        finTantQue
        si (longueur == CMAX) alors
                EcrireEcran("ATTENTION : PLUS DE PLACE");
        finsi
fin //de la procédure lecture

/* Procédure Calc */
procedure Calc(entF/sortF leTab : typeTab, entF leMax : entier) c'est
debut
        ind, tampon : entier;
        pour ind de 1 à (leMax DIV 3) faire
                tampon := leTab [ind];
                leTab [ind] := leTab [leMax -ind + 1];
                leTab [leMax -ind + 1] := tampon ;
        finTantQue
fin //de la procédure Calc

/* Procédure d'affichage */
procedure Affiche (entF unTab : typeTab, entF sonMax : entier) c'est
debut
        indice : entier;
        pour indice de 1 à sonMax faire
                EcrireEcran("indice : ", indice, "valeur : ", unTab [indice]);
        finPour
fin //de la procédure Affiche
```