TP4

Procédures sans paramètres ou avec paramètres en entrée uniquement

Procédures sans paramètres

Exercice 1

Proposez, et testez avec un `main()` approprié, une procédure qui affiche le texte suivant à l'écran (vous remplacerez les valeurs de l'exemple par vos données personnelles) :

```
Auteur : Julien Le Bihan                          Groupe : X1
Matiere : Algorithme et Programmation
Exercice: Introduction aux procedures
```

Exercice 2

Question 1

Écrivez une procédure *menu*() qui affiche le texte suivant à l'écran :

```
0 pour arrêter
1 pour l'action n°1
2 pour l'action n°2
```

Question 2

Écrivez et testez un programme principal qui :

- affiche le menu en appelant la procédure `menu()`,
- demande la réponse à l'utilisateur,
- affiche le message « `action 1` », « `action 2` », ou « `choix errone` » suivant le cas, et cela tant que l'utilisateur n'a pas tapé 0.

Procédures avec paramètres d'entrée seulement

Exercice 3

Proposez et testez une procédure qui admet en paramètre d'entrée deux nombres entiers et qui **affiche à l'écran** le plus petit d'entre eux.

Exercice 4

1) Écrivez un programme qui affiche à l'écran la table de multiplication à double entrée de 0 à 9 sous cette forme :

```
0      0       0      …       0
0      1       2      …       9
0      2       4      …      18
0      3       6      …      27
…
0      9      18      …      81
```

Pour cela, votre programme devra faire appel (10 fois) à une procédure :

`ligne(entrée m : entier)`

qui affiche sur une ligne la table de multiplication de `m` (en base dix).

2) Idem que (1) mais avec uniquement le triangle inférieur gauche.

Exercice 5

Écrivez un programme qui dessine un rectangle de **hauteur** et de **largeur** données par l'utilisateur avec la lettre 'I' et le signe '-'.

Exemple avec hauteur = 4 et largeur = 8 :

```
--------
I      I
I      I
I      I
I      I
--------
```

Pour cela, écrire et utiliser :
- une procédure `ligneTirets(entree n : entier)` qui trace une ligne de *n* tirets,
- une procédure `ligneCourante(entrée n : entier)` qui affiche une ligne constituée d'un 'I' puis de n-2 espaces puis d'un 'I',
- une procédure `corps(entrée n : entier, entrée larg : entier)` qui affiche *n* lignes courantes de largeur *larg*.

Exercice 6

Écrivez un programme qui dessine un triangle rectangle dont la hauteur h est donnée au clavier et dont la base est égale à la h+1, avec la lettre 'I' et les signes '-' et '\'.

Exemple avec h=4 :

```
I\
I \
I  \
I   \
-----
```

Pour cela, écrire et utiliser :
- une procédure `ligneTirets(entree n : entier)` qui trace une ligne de *n* tirets.
- une procédure `ligneCourante(entrée nl : entier)` qui affiche la ligne n° *nl*, constituée d'un 'I' puis de *nl*-1 espaces puis d'un '\'
- une procédure `corps(entrée n : entier)` qui affiche *n* lignes courantes.

Exercice 7 : une calculette

Nous souhaitons réaliser une calculette en nombre entiers. Le fonctionnement du programme est le suivant (les saisies utilisateur sont en gras) :

```
Ecrivez une suite d'operations sous la forme:
<operande> <operateur> <operande>
Pour terminer utilisez ":" comme operateur.
Par exemple "0 : 0" stoppe le programme
9 + 6
9 + 6 = 15
9 - 6
9 - 6 = 3
9 * 6
9 * 6 = 54
9 > 6
operateur inconnu: >
9 / 6
9 / 6 = 1
900 / 6
900 / 6 = 150
0 : 0
Au revoir....
```

Question 1

Écrivez la procédure `usage()`, qui écrit le texte de présentation, et testez-la avec un `main()` approprié.

Question 2

Écrivez la procédure `calcul(...)` qui accepte en paramètres d'entrée deux entiers et un caractère. Si ce caractère est +, −, ∗ ou / la procédure écrit l'opération correspondante, ses deux paramètres, et son résultat. Si le caractère est autre elle écrit un message d'erreur.

Question 3

Intégrez vos procédures avec un programme principal, qui effectue les actions suivantes :

– Il présente le programme.

– Il lit les entrées tant que la marque de fin n'a pas été rencontrée.

– Si l'entrée est composée d'une suite d'un entier, un caractère et un entier le programme principal appelle la procédure calcul

– Si l'entrée n'est pas d'un format correct, il traite l'erreur en recommençant l'interrogation.

– À la fin du programme un message de clôture est affiché.

Exercices complémentaires :

Exercice 8 (procédure sans paramètres)

Proposez et testez une procédure qui demande une suite de nombres entiers positifs à l'utilisateur et qui affiche à l'écran le plus petit et le plus grand d'entre eux.
La suite de nombres se termine par la valeur -1 (qui ne doit pas être prise en compte pour déterminer les extrema).
Vous devez vérifier que les nombres entrés sont positifs ; si la suite est vide, c'est à dire quand l'utilisateur rentre tout de suite -1, vous signalerez que le minimum et le maximum sont indéterminés.

Exercice 9
De la même façon que dans l'exercice 6 (*mutatis mutandis*), écrivez et testez une procédure qui dessine un triangle isocèle dont la hauteur est donnée au clavier et dont la base est le double de la hauteur.

Exemple : hauteur=3

Exercice 10 (procédure avec paramètre)

Reprendre l'exercice 4 mais en n'affichant cette fois que le **triangle supérieur droit**.

Exercice 11
Proposez et testez une procédure qui admet en paramètre d'entrée une valeur drapeau entière et demande une suite de nombres entiers à l'utilisateur. Cette suite doit se terminer par la valeur drapeau qui ne peut pas faire partie des valeurs traitées.
La procédure affiche la chaîne de caractères "constante" si la suite est constante : "croissante" si la suite de nombres est croissante ; "décroissante" si elle est décroissante ; et "non monotone" sinon.

Exercice 12 : une calculette postfixée

Réécrivez le programme de l'exercice 7 de manière à effectuer des opérations postfixées, c'est à dire que l'opérateur est situé après les opérandes.
Le schéma du fonctionnement est le suivant :

```
Ecrivez une suite d'operations sous la forme:
<operande> <operande> <operateur>
Pour terminer utilisez ":" comme operateur.
Par exemple "0 0 :" stoppe le programme
9 8 -
9 8 - => 1
7 2 *
7 2 * => 14
13 2 /
13 2 / => 6
6 3 @
erreur d'opérateur '@'
1 1 +
1 1 + => 2
0 0 :
Au revoir...
```