Prototypes

22 décembre 2007

1. compter le nombre de nombres positifs et le nombre de nombres négatifs dans un tableau de 10 entiers.

```
prototype procédure prComptage(tableau entiers iTableau,
                               adresse entière iPositif,
                               adresse entière iNegatif)
```

   Les deux paramètres passés par adresses serviront à renvoyer le résultat des calculs.

2. déterminer la position de la valeur maximale contenue dans un tableau de nombres flottants.

```
prototype fonction entière fnPosition(tableau flottants fTableau,
                                      valeur entière iTaille)
```

   La taille sert à parcourir le tableau, elle n'est pas modifiée.

3. afficher un tableau d'entiers de dimension 2 sous la forme d'un tableau rectangulaire. On supposera que les deux dimensions sont variables.

```
prototype procédure prAffichage(tableau entiers iTableau :MAX,
                                valeur entière iLignes,
                                valeur entière iColonnes)
```

   C exige une taille pour le nombre de colonnes, on a mis ici MAX.

4. effectuer une rotation entre les valeurs de 3 variables de manière que la première prenne la valeur de la deuxième, la deuxième celle de la troisième et la troisième celle de la première.

```
prototype procédure prEchange(adresse entière iVar1,
                              adresse entière iVar2,
                              adresse entière iVar3)
```

   Chaque variable doit être modifiée -> argument passé par adresse.

5. remplir un tableau de 20 chaînes de caractères avec des données d'un fichier dont on connaît le nom.

```
prototype procédure prRemplir(tableau chaînes sTableau,
                              valeur chaîne sNomFichier)
```

   Attention, on ne vous demande pas de manipuler le fichier, mais de passer son nom.

6. rechercher la position d'un nom donné dans un tableau de chaînes de caractères.

```
prototype fonction entière fnPosition(tableau chaînes sTableau,
                                      valeur entière iTaille,
                                      valeur chaîne sNom)
```

7. imprimer une étiquette comportant le titre (c-à-d M. ou Mme), le prénom, le nom, l'adresse, le code postal et la localité d'un client.

```
definition structure eClient
   champ chaîne sTitre
   champ chaîne sPrenom
   champ chaîne sNom
   champ chaîne sAdresse
   champ chaîne sCodePostal
   champ chaîne sLocalite
fin de structure

prototype procédure prImprimer(valeur eClient uUnClient)
```

   Comme on précise les différents composants de l'étiquette, on peut envisager de donner une définition de la structure qui contiendra toutes les données. On aurait pu se limiter à signaler l'existence de la structure sans nécessairement en donner la définition exhaustive.

8. remplacer une valeur donnée par une autre dans un tableau d'entiers qu'on peut choisir.

```
prototype procédure prRemplacer(tableau entiers iTableau,
                                valeur entière iTaille,
                                valeur entière iAncienne,
                                valeur entière iNouvelle)
```

   Attention les deux derniers paramètres ne seront pas modifiés, ils serviront à modifier le tableau.

9. déterminer si deux tableaux d'entiers de même taille contiennent les mêmes nombres, éventuellement mélangés.

```
prototype fonction logique flIdentiques(tableau entiers iTableau1,
                                        tableau entiers iTableau2,
                                        valeur entière iTaille)
```

10. chercher les coordonnées d'un étudiant dans une base de données en se basant sur son numéro matricule.

```
définition structure eEtudiant
   champ ...
fin de structure
prototype fonction eEtudiant fnCoordonnees(valeur entière iNumero)
```

   Nous n'avons aucune idée précise du contenu de la structure `eEtudiant`. Nous nous limitons à postuler son existence.

11. afficher les nombres compris entre deux limites.

```
prototype procédure prAfficher(valeur entière iMinimum,
                               valeur entière iMaximum)
```

12. effectuer une recherche dans un fichier de factures : au départ d'un numéro de facture unique, on désire récupérer le nom du client et calculer le total des articles mentionnés dans cette facture.

```
prototype fonction flottante fnTotal(valeur entière iNumeroFacture,
                                     adresse chaine sNomClient)
/* ou */
```

```
prototype procédure prDonneesClient(valeur entière iNumeroFacture,
                                    adresse flottante fTotal,
                                    adresse chaine sNomClient)
```

13. multiplier les données contenues dans n'importe quel tableau de 10 flottants par un coefficient modifié chaque jour.

```
prototype procédure prMultiplie(tableau flottants fTableau,
                                valeur flottante fCoefficient)
```

14. déterminer le nombre de valeurs supérieures à un nombre donné dans un tableau d'entiers (taille variable).

```
prototype fonction entière fnCompterSuperieur(tableau entiers iTableau,
                                              valeur entière iTaille,
                                              valeur entière iNombre)
```

15. rechercher les coordonnées d'un client dont on donne le numéro.

```
definition structure eClient
   champ chaîne sNom
   ...
fin structure

prototype fonction eClient fnCoordonnees(valeur entière iNumero)
```

16. examiner le fichier des clients d'une succursale donnée de l'entreprise, afficher les coordonnées de ceux qui n'ont pas effectué d'achat pour un mois donné et en déterminer le nombre.

```
prototype fonction entière fnMauvaisClients(valeur entier iMois,
                                            valeur chaîne sSuccursale)
```

17. déterminer si un nombre entier est parfait, c'est-à -dire égal à la somme de ses diviseurs.

```
prototype fonction logique flParfait(valeur entière iNombre)
```

18. aligner une adresse mémoire pour qu'elle soit égale à un multiple de 4. On suppose que l'adresse mémoire est représentée par un entier.

```
prototype procédure prAlignement(adresse entière iAdresseMemoire)
```

19. afficher les positions des éléments d'un tableau de flottants dont la partie décimale est nulle. Le nombre d'éléments du tableau peut varier d'un appel à l'autre.

```
prototype procédure prAfficher(tableau flottants fTableau,
                               valeur entière iTaille)
```

20. déterminer le nombre d'éléments d'un tableau de 10 entiers qui sont supérieurs à la moyenne.

```
prototype fonction entier fnSupMoyenne(tableau entiers iTableau)
```

21. déterminer si deux nombres sont premiers entre eux (admettent un comme seul diviseur commun).

```
prototype fonction logique fnPremiers(valeur entière iNombre1,
                                      valeur entière iNombre2)
```

22. simplifier une fraction dont on connaît le numérateur et le dénominateur.

```
prototype procédure prSimplifier(adresse entière iNumerateur,
                                 adresse entière iDenominateur)
```

On suppose qu'on possède deux variables `iN` et `iD` qui contiennent, par exemple 2 et 4. Le but est d'obtenir 1et 2. Pour modifier nos variables, il faut les passer par adresse.

23. rechercher le plus petit nombre d'un tableau de flottants dont la taille est donnée par l'utilisateur.

```
prototype fonction flottante fnPlusPetit(tableau flottant fTableau,
                                         valeur entière iTaille)
```

24. copier un tableau d'entiers dans un autre, la taille des tableaux pouvant varier.

```
prototype procédure prCopier(tableau entiers iTableau1,
                             tableau entiers iTableau2,
                             valeur entière iTaille)
```