Travaux Pratiques N°2

R107

Avant-propos

Ce TP se déroulera sur deux séances. Il est décomposé en quatre exercices, les trois premiers peuvent être considérés comme indépendants, le quatrième permet de les regrouper en un seul programme. Dans la première séance vous traiterez les exercices 1, 2 et 3 mais seulement les questions 1 et 2 de chacun. Pour chaque exercice vous avez à écrire : l'algorithme, le programme qui implémente directement cet algorithme, la réécriture de cet implémentation en utilisant une fonction ou une procédure, la réécriture du programme en utilisant l'appel à cette fonction. Vous fournirez un `Makefile` permettant de compiler au choix la version avec et sans fonction mais produisant le même exécutable.

N'oubliez pas qu'à l'issue de chaque TP, vous devrez soumettre sur eCampus un fichier d'archive (`.tar`) contenant l'intégralité du répertoire TPxx et dont le nom correspond à *nomdefamille*_TPxx (par exemple l'étudiant Gilles Rousselle soumettant le TP N°2 déposera le fichier `rousselle_TP2.tar`. Seule la dernière soumission de votre TP sera corrigée !

Rappelons que la commande `tar` peut être utilisée ainsi :

```
cd ..
tar cvf rousselle_TP2.tar TP2/
```

Problème 1

On désire écrire une petite calculatrice élémentaire permettant de calculer le résultat de l'opération a op b, où op correspond à l'un des quatre opérateurs : `+`, `-`, `*` ou `/` et a et b sont des réels. Le programme demande à l'utilisateur de saisir les opérandes a,

b et l'opérateur op, effectue l'opération et affiche son résultat si cette opération est valide et un message d'erreur sinon.

1. Ecrivez l'algorithme correspondant.
2. Ecrivez le programme `calculatrice` en langage C implémentant cet algorithme.
3. Transformez cette implémentation en proposant une fonction `calcul()` implémentant le calcul. Cette fonction doit **retourner** la valeur calculée (et non l'afficher). L'exécutable s'appellera `calculatriceAvecFonction`.

   Rappelons que la programmation structurée correspond, notamment, à une programmation modulaire où chaque module correspond à un sous-ensemble du programme final. En langage C, ces sous-programmes correspondent à des *fonctions* ou à des *procédures*. Le notion de fonction en programmation s'apparente beaucoup à la notion de fonction en mathématiques. Par exemple, la fonction f qui associe une valeur réelle à deux valeurs réelles telle que :

$$\begin{array}{rcl} f : \mathbb{R}^2 & \mapsto & \mathbb{R} \\ x, y & \rightarrow & f(x,y) = \frac{(4x^3+6x^2+1)\sqrt{y+1}}{3-y} \end{array}$$

   aurait pour algorithme :

```
Fonction f(x,y) en réel
Declaration
  parametre x en réel
  parametre y en réel
  variable resultat en réel
Fin declaration
Debut
  resultat ← (4*x*x*x+6*x*x+1)*racine(y+1)/(3-y)
  retourner resultat
Fin
```

   Elle se traduirait en langage C par :

```c
double f(double x, double y)
{
  double resultat;

  resultat = (4*x*x*x+6*x*x+1)*sqrt(y+1)/(3-y);
  return resultat;
}
```

L'utilisation de cette fonction C, son *appel*, peut être utilisée dans n'importe quelle expression et correspond au réel équivalent au calcul. Lors de cet appel, les instructions de la fonction sont exécutées avec chaque paramètre (ici `x` et `y`) qui est initialisé avec la valeur placée entre parenthèse dans l'ordre respectif d'apparition des paramètres et des valeurs. Ainsi :
`printf("f(%f,%f) = %f\n",1.6,2.0,f(1.6,2.0));`
affichera :
`f(1.600,2.000) = 56.714`.

Problème 2

On désire écrire un programme qui calcule la somme des premiers nombres impairs inférieure ou égale à un nombre n. Ainsi, la somme des entiers impairs inférieure ou égale à 15 est égale à 9 car $1+3+5=9$ alors que $1+3+5+7=16$. Le nombre n doit être demandé à l'utilisateur et le programme affichera le résultat.

1. Ecrivez l'algorithme correspondant.
2. Ecrivez en langage C le programme `sommePremiersImpairs` implémentant cet algorithme.
3. Transformez cette implémentation en proposant une fonction `sommepremiersimpairs()` qui retourne la somme, alors qu'on lui a donné le nombre n en paramètre ; proposez le programme `sommePremiersImpairsAvecFonction`.

Problème 3

On veut écrire un programme qui affiche la table de multiplication de 1 à 10 respectant **exactement** la forme suivante :

```
       1   2   3   4   5   6   7   8   9  10
-------------------------------------------
 1  |  1   2   3   4   5   6   7   8   9  10
 2  |  2   4   6   8  10  12  14  16  18  20
 3  |  3   6   9  12  15  18  21  24  27  30
 4  |  4   8  12  16  20  24  28  32  36  40
 5  |  5  10  15  20  25  30  35  40  45  50
 6  |  6  12  18  24  30  36  42  48  54  60
 7  |  7  14  21  28  35  42  49  56  63  70
 8  |  8  16  24  32  40  48  56  64  72  80
 9  |  9  18  27  36  45  54  63  72  81  90
10  | 10  20  30  40  50  60  70  80  90 100
```

1. Ecrivez l'algorithme correspondant. Dans un premier temps ne vous souciez pas de l'apparence respectant exactement l'alignement ; dans un second temps le résultat affiché doit exactement (au caractère près) celui qui vous est proposé.
2. Ecrivez le programme `tableMultiplication` en langage C implémentant cet algorithme.
3. Transformez cette implémentation en proposant une procédure `affiche_table()` implémentant cet affichage. En langage C, une procédure est une fonction qui ne retourne rien, et le motclé *rien* se traduit par `void`. Proposez le programme `tableMultiplicationAvecProcedure`.

Problème 4

On se propose ici d'implémenter un menu permettant de demander à l'utilisateur de choisir d'effectuer une opération simple, de calculer la somme des premiers nombres impairs inférieure ou égale à un entier ou d'afficher la table de multiplication. Se menu se présentera sous la forme :

```
Que desirez-vous faire :
    (1) Effectuer un calcul.
    (2) Calculer la somme des premiers impairs.
    (3) Afficher la table de multiplication.
    (0) Quitter.

Sélectionnez 0, 1, 2 ou 3 puis appuyez sur Entrée.
  -->
```

— Si l'utilisateur sélectionne 1 alors le programme demande la saisie d'une opération :

```
Calcul à effectuer : 12.2 * 4
Le résultat est :  48.80000
```

Le texte souligné reproduit un exemple de saisie par l'utilisateur.

— Si l'utilisateur sélectionne 2 alors le programme demande l'entrée de n et affiche la somme des premiers nombres impairs inférieure ou égale à n.

```
Entrez un entier : 15
La somme des entiers impairs inférieure à 15 est 9.
```

— Si l'utilisateur sélectionne 3 alors le programme affiche la table de multiplication.

— Si l'utilisateur sélectionne 0 alors le programme se termine.

— Après la saisie de toute autre valeur, ou après l'affichage du calcul, de la somme ou de la table, le programme affiche de nouveau le menu.

— Ecrivez la procédure `menu()` qui, en s'appuyant sur les fonctions et procédures déjà écrites, gère cet affichage et cet interaction avec l'utlisateur.
— Proposez le programme `principalTP2` permet de mettre en place le menu.