Fiche n° 6 de TP

Instructions itératives (1)

Objectifs : manipulation des instructions itératives **`for`** et **`while`** ; itération d'une instruction sur une suite de valeurs.

Prérequis : syntaxe des instructions itératives **`for`** et **`while`**.

Travail minimum : exercices 1 à 6.

Exercice 1

Tapez sans y apporter la moindre modification le programme suivant dans le fichier `boucle.c` :

boucle.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  for (int k = 0; k < 10; ++k ) {
    printf("%d\n", k);
  }
  return EXIT_SUCCESS;
}
```

boucle.c

Que fait ce programme ?

Exercice 2

Tapez sans y apporter la moindre modification le programme suivant dans le fichier `puissance.c` :

puissance.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int x;
  if (scanf("%d", &x) != 1) {
    printf("Erreur de saisie\n");
    return EXIT_FAILURE;
  }

  int v = 1;
  for (int k = 1; k <= x; ++k) {
    v = 2 * v;
    // affiche la progression du calcul lorsque x est grand
    if (k % 1000000 == 0) {
      printf("%d\r", k);
    }
  }
  printf("\nLe resultat du calcul est %d\n", v);

  return EXIT_SUCCESS;
}
```

puissance.c

Exécutez-le avec les valeurs 0, 1, 3, 10 et 30 ? Que fait ce programme ?
Que se passe-t-il si on entre un entier négatif ?
Pourquoi la valeur 31 donne un entier négatif ? Pourquoi toute valeur supérieure ou égale à 32 donne 0 ?

Que se passe-t-il si on entre les valeurs 2147483646 et 2147483647 ? Expliquez la différence de comportement entre ces deux entrées et modifiez le programme pour qu'il s'arrête dans tous les cas.

Exercice 3

Écrivez un programme qui demande un entier n positif en entrée et qui calcule à l'aide d'une boucle et affiche la somme des entiers pairs compris entre 2 et n.

Indication : Pour simplifier la programmation, on supposera (sans le tester dans le programme) que l'utilisateur fourni toujours une valeur beaucoup plus petite que `INT_MAX` afin que le résultat puisse être stocké dans une variable de type `int`.

Exercice 4

1) Écrivez une fonction `somme_impair` qui prend en paramètre un entier n positif puis qui calcule à l'aide d'une boucle la somme des n premiers entiers impairs strictement positifs.

2) Écrivez ensuite un programme qui demande à l'utilisateur un entier n positif en entrée puis qui affiche la somme des n premiers entiers impairs strictement positifs. On fera bien attention de vérifier et de corriger les saisies si besoin.

Indication : Pour simplifier la programmation, on supposera (sans le tester dans le programme) que l'utilisateur fourni toujours une valeur beaucoup plus petite que `INT_MAX` afin que le résultat puisse être stocké dans une variable de type `int`.

Exercice 5

1) Écrivez une fonction `int factorielle(int n)` qui calcule $n!$.

2) Écrivez ensuite un programme qui prend en entrée un entier positif n, puis qui calcule et affiche $n!$.

3) Qu'affiche le programme quand la valeur de n est 12, puis quand la valeur de n est 13 ? Qu'en pensez-vous ?

Exercice 6

Le but de cet exercice est de programmer un minuteur.

1) Écrivez une fonction qui prend en paramètres les minutes et les secondes et qui enlève une seconde.

2) Écrivez ensuite un programme `chrono` qui demande à l'utilisateur de saisir son temps, puis qui affiche le décompte à chaque seconde. Pour celà, on fera appel à `sleep(1)` qui attend 1 seconde avant de poursuivre l'exécution du programme. La fonction `sleep` est déclarée dans le fichier d'en-tête `unistd.h`.
Remarque : Sous windows, la fonction `sleep` n'est pas reconnue. Vous devez remplacer `sleep(1)` par `Sleep(1000)` et `#include <unistd.h>` par `#include <windows.h>`

```
./chrono
Combien de temps au chrono (mm:ss) ?
1:1
01:01
01:00
00:59
00:58
 ...
00:07
00:06
00:05
00:04
00:03
00:02
00:01
```