Programmation Procédurale en Langage C – TD2

Conditionnelles, boucles et tableaux

Exercice 1 : Structures conditionnelles

Écrire un **programme** qui lit trois valeurs entières (A, B et C) au clavier et qui affiche la plus grande des trois valeurs en utilisant :

1. `if - else` et une variable d'aide `MAX`
2. `if - else if - ... - else` sans variable d'aide
3. les opérateurs conditionnels `() ? :` et une variable d'aide `MAX`
4. les opérateurs conditionnels `() ? :` sans variable d'aide (optionnel)

Remarque : 1.1 – 16 lignes / 1.2 – 16 lignes / 1.3 – 13 lignes / 1.4 – 12 lignes

Exercice 2 : Structures répétitives

Écrire une **fonction** permettant de calculer la factorielle d'un entier naturel N (rappel : $N! = 1.2.3 \ldots (N-1)N$) en respectant le fait que $0! = 1$. Pour ce faire, utiliser une variable d'aide entière `int f` initialisée à `1`. Le prototype de la fonction sera le suivant :

```
int factorielle (int N) ;
```

1. Utilisez `while`,
2. Utilisez `for`,
3. Utilisez `do {} while`.

Remarque : 2.1 – 5 lignes / 2.2 – 5 lignes / 2.3 – 7 lignes

Exercice 3 : Répétition et affichage

Écrire une **fonction** d'affichage d'un triangle isocèle formé d'étoiles de N lignes (N est donc un paramètre). Le prototype de la fonction sera le suivant :

```
void triangle (int N) ;
```

Exemple pour $N = 5$ (*i.e.* pour l'appel `triangle(5);`) :

```
    *
   * * *
  * * * * *
 * * * * * * *
* * * * * * * * *
```

On utilisera une structure répétitive `for ()` dans une première version, puis un `while ()`. Pourquoi ne peut-on pas utiliser un `do ... while ()` ? *Remarque : for – 7 lignes / while – 10 lignes / do while – 14 lignes*

Exercice 4 : Parcours de tableaux

Avant de continuer, d'abord terminer les exercices 4 et 5 de la feuille de TD1

Écrire une **fonction** dont le prototype est : `int palindrome (char s[]) ;` qui retourne 1 si la chaîne représentée par le tableau `s` est un palindrome, 0 sinon.

NB : un palindrome est une chaîne qui est égale à elle même quelque-soit le sens de lecture.

Remarque : 7 lignes

Exercice 5 : Polynômes

On peut représenter les polynômes $P(X) = \sum_{i=0}^{n} a_i \times X^i$, de degré n, en utilisant des tableaux de dimension $n+1$, et en considérant que chaque case d'indice i représente le coefficient du monôme $a_i \times X^i$.
Par exemple : Soit P_1 le polynôme de degré 6 défini tel que $P_1(X) = 5 + 2 \times X + 8 \times X^2 + 3 \times X^5 + 1/2 \times X^6$.
Le polynôme P_1 est alors représenté par le tableau a de dimension 7 (indexé de 0 à 6) suivant :

a	5	2	8	0	0	3	0.5

L'évaluation du polynôme P_1 donne $P_1(0) = 5$, $P_1(1) = 18.5$, $P_1(2) = 169$, $P_1(3) = 1176.5$, ...

1. Écrire une **fonction** qui prend en paramètre une valeur de X, un tel polynôme P, et qui retourne la valeur $P(X)$. Pour cela, utiliser une première version *naïve* dans laquelle on utilisera la fonction d'élévation d'un réel à une puissance réelle donnée, dont le prototype est ainsi défini dans `<math.h>` :
   `double pow (double x, double p);`
   Le prototype de la fonction à écrire est :

   ```
   double evaluer_polynome (double X, double P[], int n);
   ```

2. Donner ensuite une deuxième version qui utilise le schéma de Horner. Faire deux versions : l'une récursive, l'autre itérative.
   Rappel du schéma de Horner :

$$\begin{aligned} P(X) &= a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots\cdots + a_nX^n \\ &= a_0 + X(a_1 + a_2X + a_3X^2 + \ldots\cdots + a_nX^{n-1}) \\ &= a_0 + X(a_1 + X(a_2 + X(a_3 + X(\ldots\cdots + X(a_n))))) \end{aligned}$$

   Le prototype des fonctions à écrire sont :

   ```
   double evaluer_polynome_rec (double X, double P[], int n); /* version recursive */
   ```

   ```
   double evaluer_polynome_ite (double X, double P[], int n); /* version iterative */
   ```

Remarque : 5.1 – 6 lignes / 5.2a – 2 lignes / 5.2b – 6 lignes

Exercice 6 : Ordre lexicographique (*optionnel*)

Écrire la fonction `strcmp()` qui permet la comparaison de deux chaînes de caractères, et qui retourne une valeur négative si la première est plus petite que la seconde, positive dans le cas inverse, et nulle en cas d'égalité des deux chaînes. Notez bien que `strcmp ("abc", "abcd") < 0`.

Le prototype de la fonction à écrire est donc :

```
int strcmp (const char *s1, const char *s2);
```

NB : Cette fonction est présente dans `string.h`, mais il est bien demandé dans cet exercice d'écrire son code.

Remarque : 5 lignes