INFOB231, TP2 Introduction au langage C

Sereysethy Touch

7 octobre 2019

Note :
— Que des exercices marqués * sont obligatoires, les travaux doivent être rendus pour le **14/10/2019** à 18h00.
— Pour chaque TP, créez un répertoire unique qui correspond au numéro de TP, et puis pour chaque exercice, créez un répertoire exo1, exo2, etc.
— Clonez un dépôt git à l'adresse suivante : `https://github.com/UNamurCSFaculty/1920_INFOB231_GXX` où XX correspond à votre numéro de groupe (01, 02, etc.)
— Déposez vos travaux dans votre dépôt git.
— Pour tous ces exercices, il faut créer un fichier `Makefile` qui permet de compiler vos programmes. Si votre programme nécessite une instruction particulière pour l'exécuter, il faut ajouter un autre fichier texte `README`.
— Il est recommandé de créer plusieurs fichiers (en tête .h, source .c) pour bien structurer vos programmes.

Itérations

Exercice 1* - PGCD

Écrivez un programme en C qui saisit deux entiers et puis affiche le PGCD de ces deux entiers. Le calcul de PGCD doit être fait de manière itérative. Le programme devra afficher le résultat à la fin de l'exécution.

Exercice 2* - Suite des nombres impaires

Écrivez un programme en C qui fait la somme d'une suite des nombres entiers impaires $\leq n$. n est saisi lors l'exécution du programme. Le programme devra afficher le résultat à la fin de l'exécution.

Exercice 3* - Suite de Fibonacci

Une suite de Fibonacci n notée F_n est définie comme suivante :

$$F_0 = 0, F_1 = 1,$$

et

$$F_n = F_{n-1} + F_{n-2}$$

Ecrivez un programme en C qui affiche la suite de Fibonacci n. n est saisi lors de l'exécution du programme. Veillez à bien tester la valeur de n, sachant que $n \geq 2$.

Fonction récursive

Exercice 4* - Factoriel n !

Ecrivez un programme en C qui saisit un nombre entier n et calcule le factoriel de n. Le programme devra faire appel à une fonction $factoriel(n)$ qui prend en paramètre un nombre entier n et retourne son factoriel. Le programme devra afficher le résultat à la fin de l'exécution.

Exercice 5* - PGCD encore

Réécrivez l'exercice 1 en utilisant une fonction récursive. Le calcul de PGCD est définie comme une fonction $pgcd(a, b)$ qui prend en paramètre deux entiers a et b et retourne le PGCD de ces deux derniers. Le programme devra afficher le résultat à la fin de l'exécution.

Tableaux

En C, on peut définir un type tableau (*array*) qui permet de stocker plusieurs variables de même types.

Déclaration : *`type idf[taille]`*

où :

— `type` définit le type d'élément de tableau,

— `idf` est le nom de tableau et

— `taille` définit la taille de tableau, l'index de tableau commence à partir de 0.

Pour accéder à un élément d'un tableau il faut connaître son index, et on y accède par *`idf[index]`*.

Exemple :

```
#define MAX 3
int main(void) {
  float notes[MAX];
  int i;
  //init
  for (i = 0; i < MAX; i++) {
    notes[i] = 0.0f;
  }
```

```
  return 0;
}
```

Exercice 6* - Palindrome

Écrivez un programme qui saisit une chaîne de caractères et puis fait appel à une fonction qui détermine si une chaîne de caractères est un palindrome. Un palindrome est un chaîne de caractères qui se lit dans les deux sens.

Exemple d'exécution :

```
$ ./palindrome
Input string: anana
"anana" is a palindrom.
```

Exercice 7* - Comptage de caractères

Écrivez un programme en C qui saisit une chaîne de caractères et puis fait appel à deux fonctions :

- une fonction qui prend en paramètre une chaîne de caractères et détermine sa longueur, sachant qu'une chaîne de caractère se termine par un symbol `'\0'`.
- une fonction qui prend aussi en paramètre une chaîne de caractères et compte la fréquence de chaque caractère de A à Z présent dans la chaîne de caractères passée en paramètre. On ne distingue pas entre des caractères majuscules et minuscules.

Le programme affiche à la fin la longueur de la chaîne de caractères et aussi les fréquences de chaque caractère.

Exercice 8* - Substring

Ecrivez un programme en C qui prend deux chaînes de caractères notés s_1 et s_2 passés en argument (à partir de ligne de commande) et qui affiche à la fin la position de début et de fin de la chaîne s_1 dans la chaîne s_2. Autrement dis, le travail consiste à déterminer si la chaîne s_1 est une sous-chaîne de la chaîne s_2.

Exemple d'exécution :

```
$ ./substring jour bonjour
'bon' is a substring of 'bonjour'
Start index: 3
End index: 6
```