UNIVERSITÉ PARIS DESCARTES

DUT Informatique - 1ère année
SDA - Structures de Données et Algorithmes

TRAVAUX DIRIGES – Semaine n°1

Thèmes

- Analyse, déclaration, documentation et définition de fonctions

Spécifications

Pour toutes les fonctions étudiées : analysez, prototypez (en choisissant le type des paramètres le plus adapté aux données), gérez le domaine de définition de la fonction, documentez le prototype, donnez quelques tests unitaires (assertions) puis codez la fonction.

Exercice 1. Miroir

Vous devez réaliser une fonction rendant le miroir d'un nombre, c'est-à-dire la valeur obtenue en lisant le nombre de droite à gauche. Par exemple, le miroir de 1345 est 5431. Pour ce faire, vous pouvez employer les fonctions *sprintf* et *sscanf* dont le mode d'emploi est donné en annexe.

Exercice 2. Palindrome

Vous devez réaliser une fonction vérifiant qu'un nombre est un palindrome (i.e. sa valeur est la même que le nombre soit lu de gauche à droite ou inversement). Cette fonction doit renvoyer en C++ une valeur de type bool = {false, true} (type de base du langage C++).

Exercice 3. Saisie

Vous devez réaliser une fonction de saisie d'un nombre positif. La fonction doit prendre en entrée un message d'invitation à la saisie d'un entier naturel et renvoyer la valeur d'un entier naturel (positif ou nul).

Exercice 4. Conjecture

Une conjecture annonce qu'à partir de tout nombre si on l'additionne avec son miroir puis si l'on reproduit itérativement cette même opération avec le résultat de la somme, on obtient un palindrome après un nombre fini d'étapes (éventuellement nul). Voici l'application de cette conjecture au nombre 168.

168 + 861 = 1029 + 9201 = 10230 + 03201 = 13431 est un palindrome

Ecrivez un programme en C++ réalisant cette suite d'additions jusqu'à l'obtention d'un palindrome. Vous respecterez le format d'affichage de l'exemple ci-dessus.

ANNEXE. sscanf

```
#include <stdio.h>
    int sscanf( const char *buffer, const char *format [, argument ] ... );
```

Usage :

La fonction *sscanf* fonctionne exactement de la même façon que *scanf* mis à part que les données ne sont pas lues à partir de l'entrée standard mais à partir de la chaîne de caractères buffer passée en paramètre. sscanf (comme *scanf*) renvoie le nombre de données ayant pu être converties.

Exemple :

```
int main() {
    char s1[3] =  "15" , s2[5] =  "abcd" ;
    int res, a;

    res = sscanf(s1, "%d", &a) ;          // a = 15 et res = 1
    printf("%d, %d\n", a, res) ;          // affiche "15, 1"
    res = sscanf(s2, "%d", &a) ;          // la conversion échoue,
                                          // a n'est pas affecté, res est égal à 0
    printf("%d, %d\n", a, res) ;          // affiche "15, 0"
}
```

ANNEXE. sprintf

Synopsis :

```
#include <stdio.h>
    int sprintf( char *buffer, const char *format [, argument ] ... );
```

Usage :

La fonction *sprintf* fonctionne exactement de la même façon que *printf* mis à part que les données ne sont pas envoyées sur la sortie standard mais dans la chaîne de caractères buffer passée en paramètre. *sprintf* (comme *printf*) renvoie le nombre de données ayant pu être converties.

Exemple :

```
int main() {
    char s[256];
    int res, a=10;

    res = sprintf(s, "%d * %f = %f", a, 2.5, a*2.5) ;    // s = "10 * 2.5 = 25.0"
    printf("%s, %d\n", s, res);                          // affiche "10 * 2.5 = 25.0, 3"
}
```