

TP 3: chaînes de caractères

Lionel Lacassagne

Introduction

Rappel: Le nom des fonctions de test est le nom de la fonction implantant un algorithme préfixée par `test_`.

1 Questions

1.1 Chiffres romains et chiffres arabes

http://fr.wikipedia.org/wiki/Numération_romaine

unités	dizaines	centaines	milliers
1 = I	10 = X	100 = C	1000 = M
2 = II	20 = XX	200 = CC	2000 = MM
3 = III	30 = XXX	300 = CCC	3000 = MMM
4 = IV	40 = XL	400 = CD	4000 = MMMM
5 = V	50 = L	500 = D	
6 = VI	60 = LX	600 = DC	
7 = VII	70 = LXX	700 = DCC	
8 = VIII	80 = LXXX	800 = DCCC	
9 = IX	90 = XC	900 = CM	

Table 1: conversion des nombres arabes en nombres romains

1. Coder la fonction qui affiche la représentation en chiffre romain, de tout nombre compris entre 1 et 4999 en utilisant en interne à la fonction, des tables de conversion. Nom de la fonction: `arabe_romain_table(int n)`. Indice algorithmique: décomposer n en paquets: $n = 1000 \times m + 100 \times c + 10 \times d + u$. Utiliser m , c , d et u pour adresser les différentes tables. L'affichage devra être le suivant:
 - `printf("%d = ");`, puis
 - l'ensemble chiffres romains équivalents à chaque paquet (m , c , d et u), puis
 - un retour chariot du type `printf("\n");` ou `puts("");`.
2. Faire de même uniquement par des calculs (et des boucles pour les symboles qui se répètent). Nom de la fonction: `arabe_romain_calc(int n)`.

2 Manipulation de chaîne

Rappel: en C, les chaînes de caractères sont dites *nul-terminated*: après dernier caractère, il y a un caractère supplémentaire: `'\0'`. Ainsi la chaîne `"123"` est de longueur 3 mais occupe 4 octets en mémoire (avec l'hypothèse de codage des `char` sur 8 bits).

1. Coder la fonction `char* str_end(char *str)` qui renvoie un pointeur contenant l'adresse de fin de chaîne (c'est à dire l'adresse du caractère `'\0'` de fin).
2. Coder la fonction `int str_len(char *str)` qui renvoie la longueur d'une chaîne de caractère. Par exemple `int str_len("123")` vaut 3.
3. Coder la fonction `char* str_dup(char *str)` qui alloue dynamiquement (via `malloc`) une zone mémoire de la taille de la chaîne `str` (utilisation de `str_len()`) et qui renvoie l'adresse de début de cette nouvelle zone mémoire.
4. Coder la fonction `char* str_cat(char *src1, char *src2)` qui alloue une nouvelle zone mémoire pour y concaténer `src1` et `src2` et qui renvoie l'adresse de début de cette zone. Attention, cette fonction doit pouvoir traiter les cas où l'une ou les deux sources sont des pointeurs nuls.
5. Coder la fonction `int int_len(int x)` qui renvoie le nombre de chiffres composant un entier (sans utiliser de fonction mathématique). Par exemple `int_len(234)` renvoie 3.
6. Coder la fonction `int i2str(int x)` qui alloue une nouvelle zone mémoire permettant d'y écrire le nombre `x` sous forme d'un ensemble de caractères de type `char` et qui renvoie l'adresse de début de cette zone. Par exemple `i2str(234)` renvoie "234".
7. Coder la fonction `int ic2str(int x, char c)` qui se comporte comme la fonction `i2str()` et qui ajoute en plus le caractère `c` dans la chaîne. Par exemple `ic2str(234, 'a')` renvoie "234a" et `ic2str(234, '5')` renvoie "2345".
8. Coder la fonction `char* str_count(char *str, char *c, int *count)` qui écrit le nombre d'occurrences du premier caractère de la chaîne `str` dans la variable entière `count` (passée par adresse), ainsi que ce caractère dans la variable caractère `c` (aussi passée par adresse). L'adresse renvoyée est l'adresse du premier caractère différent de `c`. Attention, cette fonction doit être capable de gérer sans erreur, les chaînes vides ("") et les chaînes nulles (dont le pointeur vaut "NULL"). Exemple: `ptr = str_count("1123", &c, &count)` donne `ptr = "23", c = '1'` et `count = 1`.

3 Palindrome

Un palindrome est un mot que l'on peut lire de gauche à droite et de droite à gauche, comme `radar` ou `123321`. Par extension, une chaîne vide ou ne contenant que des espaces est un palindrome.

1. Coder une fonction qui renvoie 1 si la chaîne de caractère est un palidrome et 0 sinon. Nom de la fonction: `is_palindrome1(char *str)`. Hypothèses de fonctionnement: le pointeur est forcément alloué, il n'y a qu'un mot dans la chaîne, tous les caractères sont soit des chiffres, soit des lettres minuscules de l'alphabet latin, et il n'y a pas d'espace.
2. Faire de même dans le cas où les espaces sont permis. Nom de la fonction: `is_palindrome2(char *str)`. Exemples (sans les majuscules ni les accentuées): `"Eliot, rusé, traça sa carte sur toile"` ou `" 12 34 32 1 "`. Attention, il peut aussi y avoir des espaces en début et en fin de chaîne.

4 Commentaire itéré

Un commentaire itéré consiste à dénombrer le nombre d'occurrences des différents éléments (*token*) d'une chaîne de caractères et à créer une nouvelle chaîne contenant cette description. Par exemple la chaîne "2" contient **un 2** la chaîne résultante sera donc "12". De même comme la chaîne "12" contient **un 1** et **un 2**, la chaîne résultante sera "1112".

1. Coder la fonction `char* commentaire(char *str)` qui réalise une itération de l'algorithme de commentaire itéré présenté précédemment. Pour cela, s'aider d'un certain nombre de fonction de manipulation de chaîne de caractères des exercices précédents. Contrainte: tout faire avec de la mémoire dynamique et ne pas utiliser de fonction d'une bibliothèque existante (`string.h`, `mem.h`, ...). En particulier, ne pas oublier de désallouer au fur et à mesure les zones mémoires qui ne sont plus nécessaires, ou dont le pointeur sera utilisé pour pointer une autre zone mémoire.
2. Coder la fonction `char* commentaire_itere(char *str, int iter)` qui appelle `iter` fois la fonction précédente. Là aussi faire attention à bien libérer la mémoire. Contrainte supplémentaire: lors du premier appel de cette fonction, le pointeur passé en argument est un pointeur sur une zone de texte allouée statiquement (`str = "123"`). Cette zone mémoire n'est pas désallouable, car statique, il ne faut donc pas tenter de la désallouer par un `free` (elle sera désallouée automatiquement à la fin de la fonction englobante).
3. Faire un compte rendu (en pdf uniquement) avec des schémas décrivant le fonctionnement de ces deux fonctions.