ESIEE IN4A11

Programmation

Algorithmique
Chaînes de caractères et tableaux

Jean-Claude GEORGES

Année scolaire 2007-2008

1 Moyenne(*)

Écrivez une fonction qui calcule la moyenne de n nombres stockés dans un tableau de `double`.
Prototype : `double calculerMoyenne (double * t, int n) ;`

2 Les carrés (*)

Écrivez une fonction qui remplit un tableau de n `int` par les carrés des n premiers nombres entiers et retourne l'adresse du tableau.
Prototype : `double * remplirCarres (double * t, int n) ;`

3 Recherche d'éléments sur critère(*)

Écrivez la fonction qui retourne l'indice du premier élément strictement négatif parmi les n premiers éléments d'un tableau de `double` (`-1` si aucun élément n'est négatif).
Prototype : `int chercherIndicePremierNegatif (double * t, int n) ;`

4 Maximum(*)

Écrivez la fonction qui retourne **la valeur** du plus grand des n premiers éléments d'un tableau de `double`.
Prototype : `double chercherValeurPlusGrand (double * t, int n) ;`

5 Position du maximum(*)

Écrivez la fonction qui retourne **l'indice** du plus grand des n premiers éléments d'un tableau de `double` (en cas d'ex-æquo, l'indice du premier d'entre eux).
Prototype : `int chercherIndicePlusGrand (double * t, int n) ;`

6 Copie (**)

Écrivez la fonction qui copie les n premiers éléments d'un tableau *source* de `double` dans le tableau *destination* de `double` et retourne l'adresse du tableau *destination*. On prendra en compte le fait que les deux tableaux peuvent se recouvrir partiellement.
Prototype : `double * copier (double * destination, double * source, int n) ;`

7 Résistance équivalente (**)

Écrivez une fonction qui calcule la résistance équivalente d'un nombre quelconque de résistances en parallèle.
Rappel : $\frac{1}{R} = \sum_{i=1}^{n} \frac{1}{R_i}$
Prototype : `double calculerResistancesParalleles (double * r, int n) ;`

8 Conversion chaîne de caractères en entier (**)

Écrivez une fonction qui prend en argument une chaîne de caractères représentant un entier en décimal et retourne l'entier équivalent. (`"123"` → `123`). On supposera que la chaîne est correcte et représente bien un entier.
Prototype : `int convertirChaineVersEntier (char * s) ;`

9 Miroir(*)

Écrivez une fonction qui prend en argument une chaîne de caractères, la renverse sur elle-même (`"toto` → `"otot"`) et retourne l'adresse de cette chaîne.

Prototype : `char * refleter (char * s) ;`

10 Mise en majuscules (**)

Écrivez une fonction qui prend en argument une chaîne de caractères, la transforme en majuscules (`"toto"` → `"TOTO"`) et retourne son adresse. On laissera inchangés les caractères *non lettre*. On prendra en compte les caractères accentués ou cédillés en les transformant en leurs majuscules sans accent ou sans cédille. Réfléchissez au problème que posent les ligatures (æ, œ) et proposez une solution.

Prototype : `char * mettreEnMajuscules (char * s) ;`

11 Recherche de motif (1) (**)

Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne `1` si la première chaîne commence par la seconde et `0` sinon. Écrivez un programme pour tester cette fonction.

Prototype : `int testerDebutePar (char * chaine1, char * chaine2) ;`

12 Recherche de motif (2)(**)

Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne la position de la première occurrence de la *chaîne2* dans la *chaîne1* si elle y est présente et `-1` sinon.

Prototype : `int testerPresence (char * chaine1, char * chaine2) ;`

13 Fréquence (**)

Écrivez une fonction qui compte le nombre d'occurrences d'un caractère c dans une chaîne s. La fonction pourra être récursive. Écrivez un programme pour tester cette fonction.

Prototype : `int compterFrequence (char c, char * s)`

14 Chercher/remplacer (**)

Écrivez une fonction qui recherche dans une chaîne chaque caractère c pour le remplacer par un caractère r et retourne l'adresse de la chaîne.

Prototype : `char * chercherRemplacer (char c, char r, char * s) ;`

15 Classement alphabétique (***)

Écrivez une fonction qui prend en argument deux chaînes de caractères et retourne `-1` si la première (mise en majuscule) est avant la seconde (mise en majuscule) par ordre alphabétique, `0` si elles sont identiques (mises en majuscules) et `1` si la première est après la seconde. Prenez en compte les lettres accentuées et les ligatures (bœuf). On pourra utiliser `strcmp` (déclaré dans `string.h`). Par exemple,

`comparer("chat", "chien")` → `-1`
`comparer("zèbre","éléphant")` → `1`
`comparer("corne","COR")` → `1`
`comparer("relevé","relève")` → `0`

Prototype : `int comparer (char * s1, char * s2) ;`

16 Des chiffres en lettres (**)

Écrivez une fonction qui prend en argument un nombre de un chiffre et remplit une chaîne avec l'écriture en lettre de ce chiffre(par exemple, `8` → `"huit"`). On supposera que la mémoire allouée pour la chaîne est suffisante.

Prototype : `char * convertirChiffreVersChaine (int n, char * s) ;`

17 Des nombres en lettres (****)

Généralisez le programme précédent pour faire convertir des nombres `int` en toutes lettres (par exemple, `895` → `"huit cent quatre-vingt-quinze"`). Renseignez-vous sur les règles d'écriture des nombres en français.

Prototype : `char *convertirNombreEnChaine (int n, char *texte) ;`

18 Compteur de mots (***)

Écrivez une fonction qui compte le nombre de mots dans une chaîne de caractères (la documentation devra définir exactement ce que l'on entend par mot). Écrivez un programme pour tester cette fonction.

Prototype : `int compterMots (char * s) ;`

19 Les huit reines (***)

Écrivez une fonction calculant et affichant toutes les solutions au problème des huit reines. Huit reines doivent être placées sur un échiquier 8×8 sans que deux d'entre elles soient en prise, c'est à dire sur une même ligne, même colonne ou même diagonale.

Prototype : `int calculerHuitReines (void) ;`

20 Les tris (***)

Le but de cet exercice est d'étudier et de comparer les différents tris vus en cours. Écrivez les utilitaires suivants :

- **Prototype** : `int verifierTri(double t[], int n)`
  Vérifie qu'un tableau de `double` est trié
- **Prototype** : `void remplirAlea(double t[], int n)`
  Remplit un tableau de `double` par des valeurs pseudo-aléatoires
- **Prototype** : `void echanger(double *px, double *py)`
  Échange les valeurs de deux `double` dont les adresses sont fournies.

Écrivez les fonctions de tris suivantes :

- **Prototype** : `void trierBulle(double t[], int n)`
- **Prototype** : `void trierInsertion(double t[], int n)`
- **Prototype** : `void trierQuick(double t[], int n)`

Testez et validez ces fonctions de tris.