Examen

Langage Procédural : C

23 mars 2021

La compréhension du sujet fait partie de l'examen. Il ne sera donc répondu à aucune question. Le barème prend en compte l'argumentation fournie dans les commentaires associés aux réponses.

Tout document et fichier source personnel autorisés, communications électroniques interdites. Les librairies standard du C sont autorisées.

Durée 1h30 individuel ; barème approximatif

Pour chaque exercice, vous devez créer une fonction `main` indépendante et fournir vos sources et script de compilation (ou Makefile). Vous posterez l'ensemble de ces sources sous la forme d'une seule archive sur *madoc.univ-nantes.fr* dans l'emplacement prévu dans le cours *Info3 - Langage procédural - C*

> Veillez à respecter la séparation de la déclaration du prototype et du corps de vos fonctions. Il est TRÈS fortement recommandé de compiler vos programmes régulièrement au fil de votre avancement dans les questions.

1 Petits exercices indépendants (10 points)

Chaque question ci-dessous est indépendante mais doit former un seul programme. Vous joindrez donc une seule fonction *main* dans laquelle figurera les tests effectués pour chaque exercice.

EXERCICE 1. (2 points) Écrire la fonction `conjugue` qui prends en paramètre une chaine de caractère contenant un verbe régulier du 1er groupe (terminé par 'er') et qui en affiche la conjugaison au présent de l'indicatif. Vous devrez contrôler qu'il s'agit bien d'un verbe du bon groupe avant de le conjuguer.

EXERCICE 2. (2 points) Écrire la fonction `concat` qui prends deux chaînes de caractères en paramètre et qui retourne une troisième chaine contenant la première moitié de la première chaine et la deuxième moitié de la deuxième. Faites attention à la manière de gérer votre mémoire.

EXERCICE 3. (3 points) Écrire la fonction `trim` qui prends une chaîne de caractères en paramètre et qui la modifie en supprimant les caractères vides (espace, tabulation, retour à la ligne) en début et en fin de chaîne.

EXERCICE 4. (3 points) Écrire la fonction `exists` qui prends deux chaînes de caractères en paramètre et qui vérifie l'existence de la deuxième chaine dans la première. La position de la première occurrence de cette dernière dans la première chaine sera renvoyée ou -1 si elle n'y figure pas. Sur cet exercice, il est interdit d'utiliser les fonctions de la librairie standard du C (par exemple toute fonction figurant dans *string.h*).

Vous pourrez tester sur les exemples suivants :

- `exists("Chaine dans le texte", "dans");` ⇒ `7`
- `exists("Chaine xt dans le texte", "xte");` ⇒ `20`
- `exists("beaucoup ppplus dur", "pplus");` ⇒ `10`
- `exists("encore un test", "langage");` ⇒ `-1`

2 Des fichiers et des matrices (10 points)

Dans ce programme vous devrez analyser un fichier. À titre d'exemple, le fichier *links.txt* vous est proposer en téléchargement depuis madoc dans la section de l'examen.

Ce fichier contient une matrice carrée de booléens (représentés par les caractères `'0'` et `'1'`). La première ligne du fichier contient le nombre d'éléments en colonne et en ligne. Les lignes suivantes contiennent les données en ligne, dans lesquelles chaque colonne est séparée par un ou plusieurs caractères blancs (espace ou tabulation).

EXERCICE 1. (1 point) *Préparation :* créer la définition d'une structure de données nommée `SMatrixFile` capable de stocker les informations issus du fichier. Cette structure devra contenir :

- Une chaine de caractère donnant le nom du fichier. Vous ne devrez pas faire de supposition sur la taille de la chaine de caractère.
- La taille de la matrice (i.e. nombre de ligne ou de colonne).
- La matrice elle-même.

EXERCICE 2. (4 point) *Lecture du document :* créer une fonction `MatrixFileInit` qui prends en paramètre un nom de fichier et retourne une instance de type `SMatrixFile` après avoir analysé le fichier. Ce parseur doit pouvoir détecter des erreurs : nombre de lignes ou colonnes ne correspondant pas, erreur de syntaxe...

EXERCICE 3. (2 points) *Nettoyage de la mémoire :* créer une fonction `MatrixFileUninit` qui prends en paramètre une instance de type `SMatrixFile` et libère proprement toute la mémoire allouée dans la fonction `InitMatrixFile`. Soyez judicieux dans le choix du type du paramètre.

EXERCICE 4. (3 points) *Statistiques :* créer une fonction `MatrixFileStats` qui prends en paramètre une instance de type `SMatrixFile` et ressort plusieurs statistiques :

- Nombre de Vrai et de Faux sur la totalité de la matrice.
- Ratio Vrai/Faux sur la totalité de la matrice.
- Nombre de Vrai et de Faux par ligne et par colonne.
- Ratio Vrai/Faux par ligne et par colonne.
- Est-ce que la matrice est symétrique ?
- Est-ce que la matrice est triangulaire ? Si oui, préciser supérieure ou inférieure.

Vous êtes libres de choisir la manière de retourner tous ces paramètres... Mais soyez judicieux !