Examen

Langage Procédural : C

5 avril 2023

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, documentation du langage C autorisée, communications électroniques interdites. Les librairies standard du C sont autorisées. Aide à la génération de code (copilot, chatgpt ou autre) interdite.

Durée 1h30 individuel ; barème approximatif

Vous devez créer dans la fonction `main` le code vous ayant permit de tester les fonctions répondant à chaque question. Vous devez fournir vos sources et script de compilation (ou Makefile), sans les exécutables, que vous posterez sous la forme d'une seule archive sur *madoc.univ-nantes.fr* dans l'emplacement prévu dans le cours *INFO3 - S6 - Langage C*.

Les exercices de cet examen sont indépendants, mais partageront la même fonction `main`. Il n'y aura donc qu'un seul exécutable à produire.

Ne restez pas bloqué sur un exercice ; si vous ne pouvez répondre à la question, considérez la fonction associée fonctionnelle et passez à la question suivante.

> 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.
>
> *D'une manière générale, pour chaque question, pensez bien à tester votre code dans la fonction `main`.*
>
> *La nature et le nombre d'argument de chaque fonction à créer est à votre discrétion et fait partie de la notation.*
>
> *Les questions peuvent être traitées de manière indépendantes et être testées en fournissant un jeu de données écrit directement dans le code.*
>
> *Vous pouvez factoriser du code en créant des fonctions qui ne sont pas demandées explicitement dans les différentes questions.*

1 Statistique sur un fichier (7 points)

Nous désirons créer un programme permettant d'obtenir les statistiques d'utilisation de chaque caractère.

Question 1. (1 point) Structure de donnée des statistiques

Créer une structure mémoire permettant de stocker les informations associées à un comptage du nombre d'occurrence de tous les caractères possibles.

Vous pouvez par exemple considérer que chaque caractère possible peut être codé sur un octet (type `unsigned char`).

Question 2. (2 points) Gestion du fichier

Créer une fonction `StatistiqueFichier` qui prends en entrée le nom d'un fichier. Cette fonction doit ensuite ouvrir le fichier en effectuant toutes les vérifications d'erreur associées nécessaire et ne pas oublier de libérer à la fin les ressources créées.

Question 3. (1 point) Lecture du fichier

Une fois le fichier ouvert, ajoutez à la fonction `StatistiqueFichier` le code nécessaire à la lecture du fichier caractère par caractère.

Question 4. (1.5 points) Statistiques

Ajouter à la fonction `StatistiqueFichier` le code nécessaire au calcul du nombre d'occurrence de chaque caractère (ou octet) du fichier.

Question 5. (1.5 points) Renvoi des données

modifiez le prototype de la fonction `StatistiqueFichier` afin que les statistiques calculées précédemment soient exploitables dans la fonction appelante.

2 Agglomération d'arbres binaires (13 (+2) points)

Nous désirons gérer dans cet exercice une structure de donnée arborescente associée à un algorithme agglomératif. Cette structure est une forêt qui est définie comme un ensemble d'arbres étiquetés complets selon la définition suivante :

— chaque nœud de l'arbre (intermédiaire ou feuille) porte une valeur
— tout nœud interne (c'est-à-dire qui n'est pas une feuille) a deux fils non-vides
— une feuille comporte une lettre en plus de sa valeur

Dans notre problème, la forêt initiale est formée de plusieurs arbres à un nœud portant chacun une lettre différente, et munie d'une valeur agglomérative (par l'opérateur +) et ordonnée (par l'opérateur <). Nous allons définir un algorithme qui va former à partir de cette forêt un unique arbre agglomérant l'ensemble des arbres initiaux.

L'algorithme est de type glouton : il choisit à chaque étape les deux arbres dont les nœuds racine ont la valeur minimale, soit x et y, et les remplace par l'arbre formé de l'agglomération de x et de y et ayant comme valeur la somme des deux valeurs de x et de y. La figure 1 représente les étapes de la construction pour l'alphabet source A, B, C, D, E, F, avec les valeurs $V(A) = 0.10$, $V(B) = 0.10$, $V(C) = 0.25$, $V(D) = 0.15$, $V(E) = 0.35$ et $V(F) = 0.05$.

Question 1. (2 points) Représentation de l'arbre

Créer une structure `SNoeud` représentant un nœud d'un arbre. Cette structure est composée :
— d'une valeur (entière),
— d'un indicateur permettant de dire s'il s'agit d'une feuille ou d'un nœud intermédiaire
— et soit d'une lettre codée sur un octet (type `unsigned char`), soit d'un couple de pointeurs sur une autre structure `SNode` (lien vers sous-arbre gauche et droit).

Si vous ne savez pas comment coder le choix dans la structure, laissez tous les attributs de cette structure.

Question 2. (1.5 points) Création des feuilles

Créer une fonction `CreerFeuille` permettant de créer une nouvelle instance de la structure `SNoeud` associant un caractère à une valeur entière. La création du nœud doit être faite sur le tas.

Question 3. (1.5 points) Fusion des nœuds

Créer une fonction `AssocierNoeud` permettant de créer un nœud intermédiaire de l'arbre comme indiqué dans le préambule. La fonction prends en paramètre deux nœuds (feuille ou intermédiaire) et en génère un nouveau initialisé avec les bonnes valeurs : la valeur entière correspondant à la somme des valeurs des deux nœuds, le fils gauche et le fils droit.

De la manière similaire à la question précédente, le nouveau nœud devra être créé sur le tas.

Question 4. (2 points) Algorithme glouton

Créer une fonction `EtapeGlouton` prenant en paramètre un tableau de pointeurs vers `SNoeud` ainsi que sa taille. Cette fonction recherchera dans le tableau les deux nœuds de plus petite valeur et créera un nouveau nœud intermédiaire.

Ce nœud intermédiaire remplacera un des deux nœuds choisis dans le tableau ; tandis que la case occupée par le pointeur vers deuxième nœud sera remplie avec la valeur `NULL`.

Question 5. (1.5 points) Création de la forêt

Créer une fonction `CreerForet` prenant en paramètre un tableau de pointeurs vers `SNoeud` ainsi que sa taille. Cette fonction doit exécuter l'algorithme glouton jusqu'à qu'il ne reste plus qu'un seul nœud dans le tableau. La fonction retournera ce dernier nœud.

Question 6. (1 point) Programme principal

Créer dans la fonction principale de votre programme le code permettant de construire la forêt donnée en exemple dans le préambule (en convertissant les valeurs en entier en les multipliant par 100).

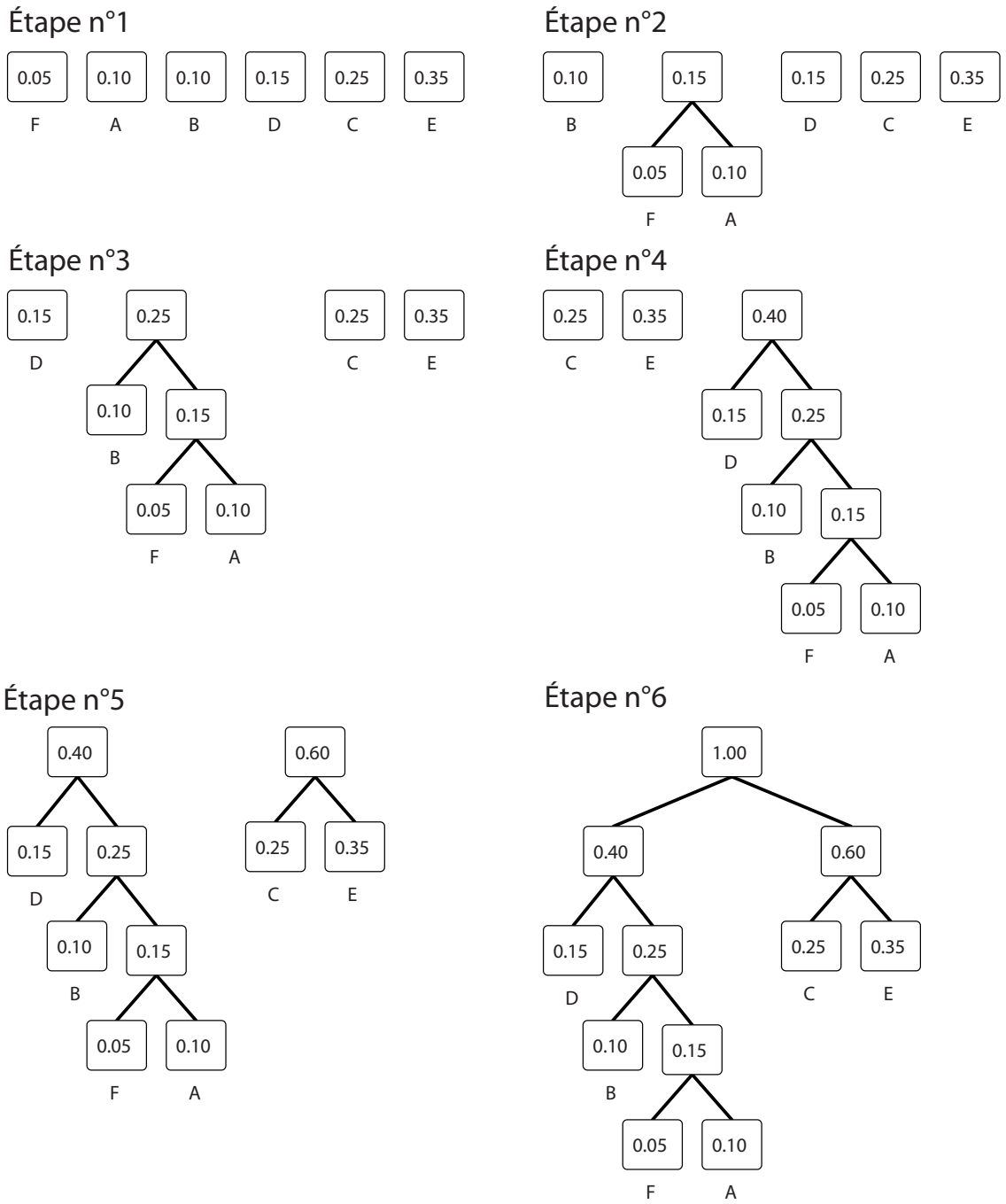

FIGURE 1 – Étapes de la construction de l'algorithme glouton pour l'alphabet source A, B, C, D, E, F, avec les valeurs $V(A) = 0.10$, $V(B) = 0.10$, $V(C) = 0.25$, $V(D) = 0.15$, $V(E) = 0.35$ et $V(F) = 0.05$.

Question 7. (2 points) Sérialisation

Créer une fonction `AfficherNoeud` permettant d'écrire sur la sortie standard la représentation textuelle de l'arbre produit. Chaque nœud intermédiaire sera affiché en indiquant sa valeur, le sous-arbre gauche, puis le sous-arbre droit `(valeur,G,D)`. Chaque nœud feuille sera affiché en indiquant sa valeur et le caractère concerné `valeur,code ASCII du caractère)`. L'affichage devra être similaire à celui donné dans l'exemple suivant pour l'arbre présenté dans le préambule :

```
(100,(40,(15,D),(25,(10,B),(15,(5,F),(10,A)))),(60,(25,C),(35,E)))
```

Question 8. (1.5 points) Libération de la mémoire

Faites en sorte de libérer proprement la mémoire à la fin de votre programme.

Question 9. (+2 points de bonus) Désérialisation

Créer une fonction `GenererForet` qui prends en paramètre une chaîne de caractère de la forme générée dans la question 7. Cette fonction doit construire l'arbre correspondant à la chaîne donnée et renvoyer l'arbre généré.