Programmation en C et structures de données

guillaume.revy@univ-perp.fr

CC1, Vendredi 12/11/2021 de 9h30 à 12h30

Ce programme est à réaliser seul, sous environnement GNU/Linux. Il doit être rendu sur Moodle, sous forme d'un unique fichier `<nom_prenom.c>`. Seuls les documents disponibles sur Moodle sont autorisés. Tout autre document (cours, exercices, ...), ou autre matériel (téléphone portable, ...) est interdit.

Conseils :
- Il est à noter que la qualité primera sur la quantité : un programme court, clair, bien conçu et qui compile sera mieux noté qu'un gros programme mal conçu, incompréhensible, voire qui ne compile pas. Utilisez des noms de variables explicites, indentez votre programme, et n'hésitez pas à le commenter.
- Lisez tout le sujet avant de commencer.
- Veuillez respecter les noms de fonctions et structures proposées dans le sujet.

L'objectif est d'écrire un programme qui permet de manipuler les données relatives aux résultats du baccalauréat, par classe d'âge. Plus particulièrement, nous allons manipuler un ensemble de données. Cet ensemble de données est extrait des données publiques accessibles à l'adresse suivante :

`https://www.data.gouv.fr/fr/datasets/reussite-au-baccalaureat-selon-lage/`.

Exercice 1. Création et manipulation d'une donnée

Nous allons tout d'abord définir ce qu'est une donnée. Une donnée est caractérisée par :
- une année,
- une classe d'âge, représentée par un âge minimum et âge maximum,
- le nombre d'admis,
- et le pourcentage de réussite.

► **1.** Définir le structure `donnee` qui représente une donnée, telle que défini ci-dessus.

► **2.** Écrire une fonction `creer_donnee` qui, étant donnés une année, une classe d'âge, un nombre d'admis et un pourcentage de réussite, crée une donnée, et retourne un pointeur vers cette donnée.

► **3.** Écrire une fonction `afficher_donnee` qui affiche les champs d'une donnée passée en paramètre.

► **4.** Écrire enfin une fonction `liberer_donnee` qui libére la mémoire d'une donnée passée en paramètre par adresse, en garantissant que le pointeur soit remis à `NULL`.

Exercice 2. Création et manipulation d'un jeu de données

Nous allons maintenant définir ce qu'est une jeu de données. C'est un ensemble de données, que nous allons réprésenter par un tableau, donc on ne connait pas nécessairement la taille à l'avance. Plus particulièrement, il sera caractérisé par :
- un tableau de données, plus particulièrement, une tableau de pointeurs de structure,
- la taille maximum du tableau,
- et le nombre d'éléments présents dans le tableau.

► **1.** Définir le structure `jeu_donnee` qui représente un jeu de données, tel que défini ci-dessus.

► **2.** Écrire une fonction `allouer` qui alloue l'espace mémoire nécessaire à la manipulation d'un jeu de données de taille n passée en paramètre, et retourne un pointeur vers ce jeu de données.

► **3.** Écrire une fonction `ajouter_donnee` qui ajoute une donnée passée en paramètre à un jeu de données existant mais non nécessairement déjà alloué.

► **4.** Écrire une fonction `afficher` qui affiche les données d'un jeu de données passé en paramètre.

► **5.** Écrire enfin une fonction `liberer` qui libére toute la mémoire utilisée par un jeu de données passé en paramètre par adresse, en garantissant que le pointeur soit remis à `NULL`.

Comme nous ne savons pas la taille du jeu de données utilisé, nous devons pouvoir modifier sa taille.

► **6.** Écrire une fonction `reallouer` qui modifie la taille du jeu de données en paramètre, en allouant un nouveau tableau de la nouvelle taille.

Exercice 3. Allons plus loin, maintenant

Nous allons pouvoir maintenant écrire des fonctions pour manipuler un jeu de données. À ce stade du TP, pour utiliser un jeu de données représentatif, et que tout le monde utilise le même, vous utiliserez les données du fichier `cc1_data.txt` disponible sur Moodle. Pour construire le jeu de données correspondant, vous pourrez utiliser la fonction `lire` suivante.

```
struct jeu_donnee*
lire(const char* fname)
{
  struct jeu_donnee *J = NULL;
  int annee, a_min, a_max, nb_bac;
  float p_bac;
  FILE *F = fopen(fname, "r");
  if(F != NULL) {
    while(fscanf(F, "%d %d %d %d %f", &annee, &a_min, &a_max,
                 &nb_bac, &p_bac) != EOF)
      J = ajouter_donnee(J, annee, a_min, a_max, nb_bac, p_bac);
    fclose(F);
  }
  return J;
}

int
main(void)
{
  struct jeu_donnee * J = lire("cc1_data.txt");
  // ...
  return 0;
}
```

► **1.** Reprendre la fonction ci-dessus, et l'intégrer à votre programme pour construire le jeu de données contenu dans le fichier `cc1_data.txt`.

► **2.** Écrire une fonction `copie` qui retourne une copie du jeu de données passé en paramètre.

► **3.** En utilisant le tri à bulle, écrire une fonction `tri_bulle` qui trie un jeu de données en paramètre par année croissante, et pour une même année, par pourcentage de réussite décroissant. Tester cette fonction sur les deux jeux de données précédemment créés.

Pour rappel, le principe du tri à bulle est de comparer deux à deux les élements e_1 et e_2 consécutifs d'un tableau, et d'effectuer une permutation de e_1 et e_2 si ceux-ci ne sont pas ordonnés. Ensuite, on continue le tri jusqu'à ce qu'il n'y ait plus de permutation. Si on considère, par exemple, le tableau d'entiers $t = \{3, 0, 2, 1\}$ et le tri croissant, les étapes d'un tri à bulle sont les suivantes :

1. on compare 3 et 0 : ils ne sont pas ordonnés, on les échange $\rightarrow t = \{0, 3, 2, 1\}$,
2. on compare 3 et 2 : ils ne sont pas ordonnés, on les échange $\rightarrow t = \{0, 2, 3, 1\}$,
3. on compare 3 et 1 : ils ne sont pas ordonnés, on les échange $\rightarrow t = \{0, 2, 1, 3\}$.

La plus grande valeur (ici, le 3) a donc été placée à la bonne place (à droite) dans le tableau. Il faut ensuite faire la même chose pour les 3 autres valeurs.

► **4.** Écrire une fonction `extraire` qui, étant donné un jeu de données, construit un nouveau jeu de données (sans modifier le jeu de données initial) qui contient, pour chaque année, la donnée relative à la classe d'âge pour laquelle le nombre d'admis est le plus élevé. Tester cette fonction sur les données ci-dessus.

► **5.** En utilisant le tri fusion, écrire une fonction `tri_fusion` qui trie un jeu de données en paramètre par âge minimum croissant, et pour un même âge minimum, par nombre d'admis décroissant. Tester aussi cette fonction sur les jeux de données précédents.

Le tri fusion est un algorithme de tri qui utilise le paradigme de diviser pour règner. L'algorithme se décrit naturellement de façon récursive.

1. Si le tableau n'a qu'un élément, il est déjà trié.
2. Sinon, le tableau a n éléments
   (a) séparer le tableau en deux parties de taille $\approx n/2$,
   (b) trier récursivement les deux parties avec l'algorithme du tri fusion,
   (c) et fusionner les deux tableaux triés en un seul tableau trié.

Par exemple, si l'on considère toujours le même tableau d'entiers $t = \{3, 0, 2, 1\}$ et le tri croissant, le tri des deux sous-tableaux de tailles $n/2$ renvoie les deux tableaux t_1 et t_2 ci-dessous :

$$t_1 = \{3, 0\} \to \{0, 3\} \quad \text{et} \quad t_2 = \{2, 1\} \to \{1, 2\}.$$

Ensuite, fusionner consiste à utiliser un tableau temporaire t', et y insérer les éléments par ordre croissant, en conservant à chaque fois l'indice du plus petit élément de chaque tableau. Si on note i (resp. j) l'indice du plus petit élément de t_1 (resp. t_2), on a au départ $i = j = 0$. La fusion suit alors les étpes suivantes :

1. $t_1[i] < t_2[j]$, donc on insère $t_1[i]$ dans t' et on incrémente $i \to t' = \{0\}$ et $i = 1$,
2. $t_1[i] > t_2[j]$, donc on insère $t_2[j]$ dans t' et on incrémente $j \to t' = \{0, 1\}$ et $j = 1$,
3. $t_1[i] > t_2[j]$, donc on insère $t_2[j]$ dans t' et on incrémente $j \to t' = \{0, 1, 2\}$ et $j = 2$,
4. il n'y a plus d'élément dans t_2, donc on insère les éléments restants de t_1 dans $t' \to t' = \{0, 1, 2, 3\}$.

► **6.** Écrire une fonction `rechercher` qui, étant donnée une année a, recherche l'indice dans le tableau de la première occurrence de la donnée pour l'année a, et renvoie cet indice ou -1 si cette donnée n'existe pas. Tester cette fonction sur le jeu de données original.

► **7.** Écrire une fonction `supprimer` dont le type de retour est **`void`** et qui, étant donné un jeu de données passé en paramètre, supprime l'élément d'indice i, si cet élément existe.

► **8.** Enfin, écrire une fonction `concatener` qui concatène deux jeux de données passés en paramètre, et retourne le jeu de données ainsi créé.