TD n° 5

I3 : Structures de données

Exercice 1 [Les étudiants de l'IUT]

On veut stocker des informations sur les étudiants de l'IUT. Chaque étudiant est caractérisé par :

- Son nom (chaîne de caractères)
- Son prénom (chaîne de caractères)
- Son année d'inscription (un entier : 1 ou 2)
- Sa date de naissance (3 entiers : jour, mois, année)
- Son numéro INE (1 entier)

1. Définissez une structure en C etudiant et un type etudiant_t qui permettent de contenir les données d'un étudiant.
2. Quelle est la différence entre la structure etudiant et le type etudiant_t ? Peut-on les utiliser tous les deux ? Quelle est la différence d'utilisation entre la structure etudiant et le type etudiant_t ?
3. Peut-on remplir directement les champs de la structure etudiant ?
4. Définissez de deux façon différentes un étudiant de 1ere année qui s'appelle Alfonse Danslemur, né le 1/2/1993, de numéro INE 987654321.
5. Écrivez une fonction en C qui prend en paramètres les caractéristiques d'un étudiant (nom, prénom, année, date de naissance, INE) et retourne un pointeur vers un élément de type etudiant_t correspondant. Vous pourrez utiliser la fonction strcpy() qui copie une chaîne de caractères dans une autre, et la fonction strlen() qui retourne la longueur d'une chaîne de caractères (sans le \0 terminal).
6. Écrivez une procédure C qui affiche sur une ligne les informations sur un étudiant passé en paramètre.
7. Écrivez une procédure en C qui permet de copier le contenu d'un etudiant_t dans un autre etudiant_t, les deux étant passés en paramètres de la fonction par leur adresse. On suppose que l'espace mémoire correpsondant à l'étudiant d'arrivée est déjà alloué.
8. Lorsque l'on veut supprimer un étudiant, faut-il faire quelque chose ? Écrivez une fonction en C qui supprime un étudiant et retourne un pointeur vers l'espace libéré (et donc inutilisable).

Exercice 2 [Utilisation d'une bibliothèque]

On veut créer une bibliothèque de manipulation des données des étudiants.

1. Comment découpe-t-on le code de la bibliothèque ?
2. Que contient le fichier d'en-tête, appelé etudiant.h ?
3. À quoi sert-il ?

4. Comment l'utilise-t-on ?
5. Comment précise-t-on au compilateur où il doit aller chercher ce fichier ?
6. Que met-on dans le code de la bibliothèque ?
7. Comment appelle-t-on les fonctions de la bibliothèque ?
8. Comment compile-t-on un programme qui fait appel à des fonctions d'une bibliothèque ?
9. Comment compile-t-on une bibliothèque dynamique ?
10. Écrivez un petit programme C qui crée un étudiant, l'affiche, le copie dans un autre étudiant, affiche le second étudiant et détruit les deux étudiants.

Exercice 3 [Compilation avec make]

La commande `make` permet d'automatiser la compilation de programmes. Nous allons dans cet exercice écrire un fichier d'entrée pour `make` (communément appelé Makefile, bien que l'on puisse le nommer autrement) permettant de compiler notre bibliothèque et le programme de test.

La bibliothèque s'appellera `libmanipEtudiant.so` et l'exécutable du programme qui l'appelle s'appellera `etudiant`.

Les fonctions de la biobliothèque sont implémentées dans le fichier manipEtudiant.c, le fichier d'en-têtes s'appelle etudiant.h, et le code du programme de test s'appelle test.c.

1. La cible principale est le programme `etudiant`. De quoi dépend-il ?
2. De quoi dépend la bibliothèque libmanipEtudiant.so ?
3. De quoi dépend le code objet contenant l'implémentation des fonctions de la bibliothèque ?
4. De quoi dépend le code objet correspondant au code du programme de test ?
5. Que doit-on écrire dans le Makefile pour que make appelle le compilateur afin de générer le code objet du programme de test ?
6. Que doit-on écrire dans le Makefile pour que make appelle le compilateur afin de générer le code objet de la bibliothèque ?
7. Que doit-on écrire dans le Makefile pour que make appelle le compilateur afin de générer la bibliothèque ?
8. Que doit-on écrire dans le Makefile pour que make appelle le compilateur afin de compiler l'exécutable ?
9. Que doit-on écrire au début du Makefile pour indiquer à make le point d'entrée de la compilation ?
10. On souhaite ajouter une cible `clean`, qui permet de nettoyer le répertoire en effaçant les résultats de la compilation et les fichiers intermédiaires. Comment est-elle appelée ?
11. Que doit-on écrire dans le Makefile pour mettre en place cette cible ?
12. Mettez en ordre le Makefile complet. Dans quel ordre sont exécutées les cibles ?