Entraînement à la programmation C

MP2I Lycée Pierre de Fermat

Dans les exercices, autant que possible, divisez votre programme en **fonctions**, sans tout mettre dans le main. Dans les premiers exercices, les fonctions à créer sont précisées, mais dans la suite vous devez organiser vous même votre programme.

Bases du C

Notions à maîtriser :

— Principe de base du C : fonction main, compilation, exécution

— Syntaxe de base du C (instructions, opérations arithmétiques)

— Boucles for, boucles while, fonctions

Interaction

Notions à maîtriser :

— scanf, printf pour lire et écrire dans le terminal

— Arguments de la fonction main

— Utilisation d'aléatoire avec srand et rand

Exercices

— Créer un programme qui lit un entier dans le terminal et affiche le carré de cet entier

— Créer un programme qui lit dans le terminal, et à chaque fois que l'utilisateur rentre un flottant x, affiche l'inverse de x

— Créer un programme qui lit deux entiers a, b dans le terminal et affiche a^b. On pourra utiliser une fonction `int puissance(int a, int b)` qui implémente l'exponentiaton rapide.

— Créer un programme prend un entier n en argument et affiche n entiers aléatoires entre -10 et 10. On pourra créer une fonction `int entier_alea()` générant et renvoyant un entier aléatoire.

Pointeurs / tableaux (avant les vacances de la toussaint)

Notions à maîtriser :

— Types pointeurs, adresse d'une variable

— Tableaux créés avec `float mon_tab[500]`

— Chaînes de caractères

— Tableaux 2D, notation $T[i][j]$

Exercices

— Créer une fonction qui cherche le maximum d'un tableau de n entiers

— Créer une fonction qui remplit un tableau avec n entiers aléatoires entre a et b

— Créer une fonction qui modifie une chaîne de caractères pour mettre toutes les lettres en majuscule

— Créer un programme qui lit dans le terminal et affiche les mots en majuscule, jusqu'à lire "chut"

— Créer une fonction qui remplit un tableau 2D de caractères, où chaque case est une lettre majuscule aléatoire.

— Recoder les fonctions de la librairie `string.h` : strcat, strcmp, strcpy, etc...

Pointeurs / tableaux (après la toussaint)

Notions à maîtriser :

— Tableaux alloués dynamiquement (avec malloc)

— Correspondance malloc - free.

— Chaînes de caractères

— Tableaux 2D, notation $T[i][j]$

Exercices

— Créer une fonction qui **crée et renvoie** un tableau de n entiers aléatoires entre a et b

— Créer une fonction qui modifie une chaîne de caractères pour mettre toutes les lettres en majuscule

— Créer un programme qui lit dans le terminal et affiche les mots en majuscule, jusqu'à lire "chut"

— Créer une fonction qui crée un tableau 2D de caractères, où chaque case est une lettre majuscule aléatoire

Structs

Notions à maîtriser :

— Définition d'un type struct

— Création d'un objet instance d'un type struct, accès aux attributs avec `a.b`

— Renommage d'un type avec typedef

— Pointeurs de structure, syntaxe `a->b`

Exercices

— Créer un type struct pour stocker les informations d'un livre : titre, nombre de mots, auteur

— Créer un type struct pour stocker les informations d'une série de livres : nombre de tomes et liste des tomes.

— Créer une fonction qui compte le nombre total de mots écrits par un auteur.

Exercice : carte de jeu

On veut écrire un morceau du code d'un petit jeu. Dans ce jeu, le personnage se déplace sur une carte 2d composées de cases carrées. Chaque case a :

— Un type de terrain (forêt, plaine, colline, montagne, eau)

— Un nombre de pièces d'or éparpillées

Le personnage a une réserve d'énergie, qui diminue lorsqu'il bouge d'une case à une case adjacente. La quantité d'énergie utilisée pour se déplacer dépend du type des cases d'arrivée et de départ :

— Si la case d'arrivée ou la case de départ est une montagne, le coût du déplacement est 3.

— Sinon, si la case d'arrivée ou la case de départ est une forêt, une colline, ou de l'eau, le coût du déplacement est 2.

— Sinon, on se déplace de plaine en plaine, et le côut d'un déplacement est 0.

Le personnage ramasse les pièces d'or d'une case avant de la quitter, et on doit donc se rappeler du nombre total de pièces ramassées.

— Créer des structures pour stocker les informations d'une case, du personnage, et de la carte du jeu, et des fonctions d'affichage. On pourra afficher une case en affichant l'initiale de son type, et afficher le joueur avec un O

— Créer une fonction qui génère une carte aléatoire.

— Créer une fonction qui effectue le déplacement du personnage dans une direction donnée

— Rajouter une boucle de jeu : lecture du déplacement de l'utilisateur, affichage de la carte, de l'énergie restante et du nombre de pièces ramassées, jusqu'à avoir épuisé toute l'énergie

Fichiers

Notions à maîtriser :

— Ouvrir un fichier en mode écriture/lecture/ajout, fermer un fichier

— fscanf, fprintf pour lire et écrire dans un fichier

Exercices

— Écrire un programme qui prend en argument un nom de fichier, et affiche le contenu de ce fichier

— Écrire un programme qui prend en argument deux noms de fichiers, et copie le contenu du premier dans le deuxième

— Écrire un programme qui prend en argument un nom de fichier, dont on suppose que ce fichier contient une suite d'entiers, et affiche les entiers dans l'ordre croissant

Exercice : données des utilisateurs d'un site web

On veut simuler la gestion d'un site web, en particulier des informations des utilisateurs. Pour cela, on va stocker les informations des utilisateurs dans un grand fichier

— Définir une structure pour stocker les informations d'un utilisateur :

- Pseudo
- Adresse mail
- Date d'inscription

— Écrire une fonction qui génère et renvoie un utilisateur aléatoire. On pourra fixer un tableau avec quelques serveurs mails (gmail.com, hotmail.fr, aol.fr, laposte.net, etc...) pour la génération

— Écrire une fonction qui écrit les données d'un utilisateur dans un fichier, tout sur une ligne, en écrivant à la fin du fichier.

— Écrire une fonction qui lit un fichier contenant les données de plusieurs utilisateurs, et qui remplit et renvoie un tableau avec ces données.

— Écrire un programme qui demande en boucle des instructions à l'utilisateur, et qui permet de :

- Générer un utilisateur aléatoire
- Afficher la liste des utilisateurs par date d'inscription
- Compter le nombre d'utilisateurs ayant une adresse sur un serveur mail donné.