TD de Langage C

Programmation Structurée

L3 - EEA

Université de Marne La Vallée

2009-2010

Gautard Valérie, Christophe FLOUZAT

18 février 2010

Résumé

Ces TD seront effectués sous un environnement Linux. La compilation se fera à l'aide du compilateur GCC. Les codes sources seront édités à l'aide d'un éditeur de texte simple au choix (VI, EMACS, NANO, gedit, kate, ...). Chaque section correspond à une séance de TD. Chaque sous section correspond à un exercice. Les exercices marqués d'une * sont facultatifs, il est entendu par là qu'ils pourront être omis par manque de temps. Aucune autre raison ne sera acceptée. A vos claviers ...

Pour compiler un programme C on utilise la commande suivante :

```
gcc -o fichierexe source.c
```

1 Saisie, affichage et opérations arithmétiques de base

1.1 Hello world

Faire un programme affichant sur le terminal le message suivant :

```
**********
* bonjour *
**********
```

1.2 Afficher des nombres

Ecrire un programme qui affiche sur le terminal :

- Un entier (forme standard)
- Un entier sous forme hexadécimal
- Un flottant
- Un flottant en notation scientifique (ex : 2500 => 2.5e3)

1.3 Saisie de nombres

Ecrire un programme qui lit successivement 2 entiers et renvoie leur somme. Modifier le programme pour 2 flottants, puis pour un entier et un flottant (deux solutions)

1.4 Arithmétique de base

Ecrire un programme de conversion degrés Celsius/Fahrenheit.

Pour rappel :

$$F = 9/5 \times C + 32$$

1.5 Saisie multiple*

Ecrire un programme de résolution d'équation du premier degré qui prend en entrée une expression du type :

a.x+b=c (ex : 2x+3=11)

où a, b et c sont des entiers (dans un premier temps)

... Dans la suite, pour chaque question il sera demandé de créer une ou plusieurs fonctions par exercice, la fonction main ne contiendra alors que des appels à ces fonctions et les entrées/sorties nécessaires pour tester ces fonctions.

2 Structures conditionnelles

2.1 If then else...

Créer pour chaque test possible entre 2 entiers (=,>=,<=,>,<,!=) une fonction qui affiche le résultat.

ex : a=1 b=2 affiche "A n'est pas égal a B"

2.2 Test de parité

Ecrire une fonction qui renvoie sous forme booléenne (0 pour pair, 1 pour impair) la parité d'un entier passé en paramètre.

2.3 Logique booléenne

Pour chaque cas, écrire une fonction déterminant

- si 3 entiers sont tous égaux
- si au moins deux des trois entiers sont égaux.

2.4 Test alternatifs*

Reprendre la question 2.1 en donnant cette fois une version alternative des tests d'égalité et de différence.

2.5 Switch case...

Créer une fonction qui affiche les entiers de 0 a 10 en toutes lettres.
ex : 0 affiche "zero", 1 affiche "un", ...
Si l'entier entré n'est pas compris entre 0 et 10, afficher un message d'erreur.

* Reprendre la question en retournant un code erreur en cas d'erreur

3 Les boucles

3.1 Compteur

Réaliser une fonction qui affiche à l'écran les entiers de 0 à 100 compris en utilisant trois types de boucle (while,do while,for).

3.2 Verification de saisie

Ecrire un programme qui demande à un utilisateur d'entrer un nombre entre 0 et 10 inclus. Si le nombre entré n'est pas compris entre 0 et 10, demander a l'utilisateur une nouvelle saisie jusqu'à ce que le nombre entré soit valide.

3.3 Plus ou moins

Réaliser un jeu de plus ou moins...
Le jeu de plus ou moins consiste à deviner un nombre tiré aléatoirement. Le joueur propose un nombre et le jeu lui répond si celui-ci est au dessus ou en dessous du nombre à trouver. Si le joueur découvre le nombre exact en moins de dix coups il a gagné, sinon il a perdu.

Pour réaliser ce programme, c'est à vous de découper judicieusement les traitements en différentes fonctions.
Pour générer un nombre aléatoire, utiliser la bibliothèque "alea.h" (alea.h et alea.c sont à récupérer sur le compte de l'enseignant). Cette bibliothèque fournit deux fonctions :

_ void initRand(int max) permet d'initialiser la séquence pseudo-aléatoire.
_ int nRand() renvoie un entier aléatoire entre 0 et max.

3.4 Prototype et bibliotheques *

Reprendre la question précédente en prenant soin de déclarer les prototypes de chaque fonction créée.
Inspirer vous des sources alea.h et alea.c pour créer une bibliothèque plusoumoins.h qui contiendra toutes vos fonctions.

4 Récursivité

4.1 Factorielle

Ecrire une fonction qui renvoie la factorielle d'un entier positif. Pour cela utiliser la propriété récursive de la factorielle :

Pour rappel : $n ! = n \times (n-1) !$

$0 ! = 1$

*Générez un message d'erreur si le nombre entré est négatif.

5 Les tableaux

5.1 Création de tableau

Créer un tableau contenant 10 entiers générés aléatoirement.

5.2 Fonction d'affichage

Écrire une fonction qui affiche un tableau d'entiers.

5.3 Fonction de tri

Écrire une fonction qui trie les valeurs d'un tableau d'entier.

*proposer 3 algorithmes différents pour cette fonction.

6 Les Pointeurs

6.1 Les bases...

Écrire un programme simple...(juste le main())

...déclarer un pointeur sur un entier P
...déclarer deux entiers A et B, leur donner une valeur
...faire pointer P sur l'adresse de A
...afficher la valeur de A en passant par P
...afficher l'adresse de B

6.2 Pointeur et tableau

Réécrivez la fonction de tri de tableau d'entier en utilisant la notation par pointeur.

7 Allocation dynamique

7.1 Tableau dynamique

Déclarer un tableau d'entier d'une case. Créer une fonction permettant d'ajouter une valeur à un tableau qui prend en paramètre la valeur à ajouter et un pointeur sur le tableau et qui renvoie un pointeur sur le tableau résultant.

8 Structures et énumérations

8.1 Les énumérations

Déclarer une énumération permettant de nommer les jours de la semaine en toutes lettres. Créer un type "jdls" pour cette énumération. Tester le type que vous venez de créer.

8.2 Les structures

Créer une structure "heure" qui contient trois champs de type entier sur 8bits, "h" pour les heures, "m" pour les minutes et "s" pour les secondes. Créer le type associé.
Ecrire une fonction "heure scanHeure()" qui renvoie un pointeur sur une structure de type "heure" et permet de saisir au clavier une heure complète.
Ecrivez une fonction "void printHeure(heure h)" qui permet d'afficher l'heure "h" sous la forme h :m :s.
Ecrivez une fonction "*heure addHeure(heure *h1,*h2)" qui allouera dynamiquement une nouvelle structure de type "heure" pour y ranger le résultat de l'addition des heures h1 et h2 (dont les pointeurs sont passés en paramètres) et renverra un pointeur sur cette nouvelle structure.

8.3 Un peu plus de structures*

Créer une structure "evenement" avec un nom, un jour de la semaine et une heure (utiliser les types créés précédemment). Déclarer le type associé à la structure "evenement".
Ecrire enfin les fonctions "void printEvent(evenement e)" permettant l'affichage d'un événement, et "*evenement scanEvent()" permettant de saisir un événement (l'événement en question sera créé dynamiquement).
Créer un tableau dynamique de pointeur vers des éléments de type "evenement" (vide a l'initialisation) ainsi que les fonctions "*evenement addEvent(evenement e)","*evenement remEvent(char nom[])" pour ajouter et supprimer un événement, et enfin "void printEvents(evenement **evTab)" pour afficher tout les événements de notre tableau.
Tester...