Exercice 1 : Environnement de travail

Avant de commancer, il faut se connecter.

Ensuite, lancez un terminal (en utilisant le menu textuel en bas de l'écran, cherchez une application nommée `Terminal`, `Console` ou `xterm`.

Créez un répertoire `TDc/` dans votre répertoire personnel (avec la commande `mkdir TDc`). Positionnez-vous dans ce sous-répertoire (`cd TDc`.

Lancez `emacs`, créez un buffer nommé `prog1.c`. Vérifiez que le mode `C` est utilisé par `emacs` dans ce buffer (barre d'état), sinon activez-le en tapant `M-x c-mode`. Tapez en suite ce petit programme, et sauvegardez-le (clé `C-x C-s`) :

```
char chn[10]={'B','o','n','j','o','u','r''!','\0','a'}; void main(void){printf(chn);}
```

Si votre programme n'apparaît que d'une couleur, utilisez la commande `M-x font-lock-mode`.

N.B. : vous pouvez utiliser un autre éditeur de texte que `emacs`.

Exercice 2 : Compilation

Pour compiler ce programme sous `emacs`, tapez `M-x compile` et surveillez votre minibuffer. Remplacez la commande de compilation proposée par défaut (`make`) par :

```
gcc prog1.c -Wall -o prog1
```

et validez. Surveillez ce qui apparaît dans le buffer `*comilation*`. Corrigez votre programme et répétez les opérations sauvegarde/compilation jusqu'à ce que le message `Compilation finished at ...` apparaisse dans le buffer `*comilation*`.

Dans un terminal, vérifiez (commande `ls -l`) qu'un fichier exécutable nommé `prog1` a bien été créé. Lancez le avec la commande `./prog1`.

Essayez de compiler vous-même votre programme dans le terminal, sans utiliser `emacs`.

Indentez votre programme de manière à le rendre plus lisible.

Remplacez le mot `void` devant `main` par le mot `int`. Quelles sont les conséquences sur la compilation ?

Ajoutez la ligne

```
return 111;
```

juste avent la dernière accolade fermante. Quelles sont les conséquences sur la compilation ? et sur l'exécution ?

Ajoutez la ligne

```
#include<stdio.h>
```

en tête de votre programme. Quelles sont les conséquences sur la compilation ?

Exercice 3 : Variables et opérations

Indications :

- si n est de type entier, alors `scanf("%d",&n)` permet de lire un entier saisi au clavier et de stocker sa valeur dans n.
- si f est de type flottant, alors `scanf("%f",&f)` permet de lire un entier saisi au clavier et de stocker sa valeur dans n.

Nous reviendrons plus tard sur la fonction scanf, sa syntaxe et son utilisation.

Ecrivez, compilez et exécutez les programmes solutions aux exercices suivants :

1. Saisir une variable entière, afficher sa valeur.
2. Saisir deux variables et les permuter avant de les afficher.
3. Saisir une variable de type `float`, afficher sa valeur.
4. Saisir 3 valeurs, afficher leur moyenne arithmétique.
5. Demander à l'utilisateur de saisir deux valeurs `a` et `b`, afficher ensuite la différence entre la moyenne arithmétique $\frac{(a+b)}{2}$ et la moyenne géométrique $\sqrt{ab}$. Pour indication, `sqrt(f)` est la racine carrée du flottant `f`, cette fonction est disponible en important `#include<math.h>`.
6. Nous souhaitons ranger des cartons pesant chacun k kilos dans un camion pouvant transporter M kilos de marchandises. Ecrire un programme `C` demandant à l'utilisateur de saisir M et k, que vous représenterez avec des nombres flottants, et affichant le nombre (entier) de cartons qu'il est possible de placer dans le camion. N'oubliez pas que lorsque l'on affecte une valeur flottante à une variable entière, les décimales sont tronquées.

Exercice 4 : Structures de contrôle

1. En vous inspirant des exemples du cours, écrivez le programme permettant d'afficher les 100 premiers nombres :
   (a) Tous sur la même ligne
   (b) Un par ligne
   (c) 10 par ligne
2. Ecrire un programme pour afficher les nombres premiers inférieurs à 1000 en éliminant les multiples des nombres premiers déjà trouvés.

Exercice 5.

Ecrire trois programmes (max_for.c, max_while.c, max_dowhile.c) qui calculent l'élément maximum du tableau d'entiers suivant :

```
int tbl[10] = { 5,6,7,1,12,3,5,-2,10,3};
```

Chaque programme utilisera respectivement une boucle for, unr boucle while et une boucle de while.

Exercice 6.

La fonction getchar() renvoie un caractère tapé au clavier. En utilisant cette fonction, écrire un programme qui indique si le caractère tapé est : une lettre minuscule, une lettre majuscule, ou un chiffre. S'il s'agit d'un autre caractère, le programme indique qu'il s'agit d'un autre caractère. Ce programme doit également s'arrêter dès que l'utilisateur tape le caractère %.

Exercice 7.

Ecrire, compiler et exécuter les programmes qui sont les solutions des exercices suivants :

1. Convertir un entier en chaîne de caractères
2. Supprimer les commentaires d'un programme en langage C.