TP1 : Compilation, expressions

- Connectez-vous sous Linux
- A l'aide du bouton droit de la souris, créez un dossier `langageC` sur le Bureau et ouvrez le
- A l'aide du bouton droit, créez un fichier texte
  Créer un document -> Fichier vierge
- Donnez un nom (`exo1.c`), double cliquez dessus : vous êtes sous l'éditeur `pluma` qui vous permettra de taper votre premier programme C (voir exercice 1).
- A l'aide du menu supérieur gauche, ouvrez ensuite un `terminal` qui va vous permettre d'exécuter des commandes unix (voir les commandes élémentaires dans le lien *initiation à UNIX*)
  Applications -> Outils système -> Terminal
- Vous êtes à la « racine » de votre compte et la commande *ls* permet de visualiser tous les fichiers et dossiers qui s'y trouvent, notamment le répertoire `Bureau` qui contient `langageC`
- Descendez dans le répertoire `langageC` à l'aide de la commande *cd* (change directory)
  cd Bureau/langageC
  Remarque : la touche de tabulation permet de compléter automatiquement le début d'un nom de fichier à partir des premières lettres. `lang<tab>` va compléter automatiquement avec `langageC`
- Compilez (*gcc exo1.c –o exo1*) et exécutez (*./exo1*) votre programme pour le tester.

Ce TP doit faire l'objet d'un compte rendu synthétique (une phrase ou deux par question) à renvoyer à votre enseignant.

1. Première compilation, initialisation, incrémentation

Tapez et testez le programme suivant :

```
#include <stdio.h>
void main(void)
{ int cpt ; // compteur
  printf ("je vais compter jusqu'à trois:\n") ;
  cpt =1 ;  // initialisation
  printf("%d\n", cpt) ;
  cpt = cpt+1 ;
  printf("%d\n", cpt) ;
  cpt = cpt+1 ;
  printf("%d\n", cpt) ;
}
```

a) Modifiez votre programme pour qu'il compte jusqu'à 5.

b) Puis de 0 jusqu'à 4 .

c) Puis de 0 jusqu'à 8 par pas de 2

2. Le type char

On rappelle que le type `char` permet de représenter sur 1 octet aussi bien des petits entiers (entre -128 et +127) que des caractères. La distinction se fait uniquement sur l'interprétation du code binaire stocké dans cet octet.

a) Petits entiers

Reprenez le programme précédent de la question a) en déclarant le compteur comme un **`char`** et en l'initialisant à 125. Qu'observez vous ? Comment expliquez vous ce résultat « étrange » ?

b) Caractères et code ASCII

Testez le programme ci-après pour les 3 caractères suivants : 'A' , 'a' , '0' (*zéro*). Qu'en déduisez-vous ?

```
#include <stdio.h>
void main(void)
{ char car, carSuivant; // caracteres a etudier
  printf("entrez un caractère\n") ;
  scanf("%c", &car) ; // lecture d'un caractère au clavier
  // affichage de car et de son code ASCII
  printf("code('%c')", car) ;
  printf("=%d\n", car) ;
  // affichage du caractère suivant et de son code ASCII
  carSuivant = car+1 ;
  printf("code('%c')", carSuivant) ;
  printf("=%d\n", carSuivant) ;
}
```

c) Caractères, encore.

Ecrivez le programme qui lit une minuscule et écrit la majuscule associée.
On remarquera pour ça que minuscule-'a' donne le rang de cette minuscule dans l'alphabet.
De même, majuscule-'A' donne le rang de cette majuscule dans l'alphabet.

3. Expressions

Soient C et F la même température exprimée respectivement en Celsius et en Fahrenheit. La conversion de Fahrenheit en Celsius est donnée par la formule C = 5/9*(F-32).
Par exemple : 100 degrés Fahrenheit est équivalent à 37.777 degrés Celsius.

Ecrivez le programme qui lit une température en degré Fahrenheit, convertit cette température en degré Celsius et l'affiche.

Algorithme :
- définir deux variables **réelles** C et F
- lire F au clavier (utilisez le format **`%f`** pour les réels)
- calculer C à partir de F
- afficher C (utilisez **`%f`** pour les réels)

Vous afficherez peut-être 0° Celsius pour 100° Fahrenheit alors que la formule vous paraît juste !?
Elle l'est dans une écriture mathématique, mais l'est-elle en langage C ?