# TP n°3 - Codage des nombres

**Notions abordées**

- codage des entiers en binaire, en décimal
- codage des réels, problèmes de précision
- types d'entiers proposés en C
- à nouveau : boucle while, fonctions, affichage et saisie, branchement conditionnel

Toutes les fonctions doivent être commentées et testées. Par commentée, on entend que la fonction doit être munie d'une description faisant suite à d'éventuelles hypothèses sur ses arguments, et que le rôle de chaque variable est facilement identifiable, soit grâce à son nom, soit grâce à une remarque ou à un invariant de boucle.

# Codage des entiers en C (complément de cours)

## Exercice 1 Récupérer des informations grâce à `limits.h`

La mémoire de l'ordinateur est composée de bits, qui peuvent enregistrer soit la valeur 0 soit la valeur 1. La gestion de la mémoire en C est telle qu'on travaille toujours avec des paquets de 8 bits consécutifs, c'est ce qu'on appelle des **octets**. Lorsqu'on crée une variable en C, le programme réserve une zone mémoire pour stocker la valeur de cette variable. Cette zone est constituée d'un certain nombre (entier) d'octets consécutifs. La taille de cette zone dépend du type de la variable, et pour certains types elle dépend aussi de la machine utilisée. De cette taille dépend aussi le domaine de valeurs que pourra prendre la variable.

**Question 1** 

Combien de valeurs différentes peuvent êtres encodées dans une zone mémoire de $n$ octets. Autrement dit combien de valeurs sont a priori possibles pour une variable d'un type encodé sur $n$ octets ?

**Question 2** 

La librairie `limits.h` permet d'avoir accès aux tailles utilisées pour chaque type grâce à la fonction **sizeof**. Allez sur cahier de prépa et téléchargez les fichiers nommés `domaine_des_types.c` et `taille_des_types.c`. Dans un premier temps, compilez et lancez `taille_des_types.c` **seulement**.

**Question 3** 

Sachant que les types `int, short int` et `long int` utilisent le codage des entiers relatifs vu en cours, et que les types `unsigned int, unsigned short int` et `unsigned long int` utilisent le codage des entiers naturels en base 2 (aussi vu en cours), déterminez les domaines de valeurs possibles pour

chacun de ces types.

**Question 4** 🖵
Compilez et exécutez le fichier `domaine_des_types.c` pour vérifier votre proposition.

**Question 5** ✍
Utilisez les informations recueillies dans cet exercice pour remplir le tableau suivant (dans le III laissé vide dans le cours), avec les types `bool`, `char`, `int`, `short int`, `long int`, `unsigned int`, `unsigned short int` et `unsigned long int`. *Vous pouvez factoriser pas mal d'infos si vous mettez à la suite un type et le type non signé qui va avec.*

| type | spécificateur de format | taille en octet / taille en bits | nombre de valeurs possibles | domaine |
|---|---|---|---|---|
| | | | | |

# Exercice 2 Que se passe-t-il si...

**Question 1** 🖵 / ✍
Que se passe-t-il **si on donne une valeur négative pour une variable de type non signé ?**
Exécuter les instructions suivantes, au besoin exécuter des variantes pour comprendre puis expliquer ce qu'il se passe.

```c
unsigned char c = -5;
unsigned short s = -5;
printf("c=%u donc c est positif ? : %d \n", c, c>=0);
printf("s=%u donc s est positif ? : %d \n", s, s>=0);
```

**Question 2** 🖵 / ✍
Que se passe-t-il **si on dépasse les bornes ?**
Exécuter les instructions suivantes, au besoin exécuter des variantes pour comprendre puis expliquer ce qu'il se passe.

```c
char cc = 130;
char dd = 136;
printf("cc=%d\n, dd=%d\n",cc,dd);
```

puis

```c
char ee = 258;
char ff = 500;
char gg = -148;
char hh = -340;
printf("ee=%d\n ff=%d\n  gg=%d\n hh=%d\n",ee,ff,gg,hh);
```

**Question 3** bonus
Si vous voulez faire plus de tests de ce genre, comprendre ce qui est affiché, encodé... je vous propose une base de quelques tests dans le fichiers nommé **petits_tests.c** déposé sur cahier de prépa.
Le terminal n'étant pas capable d'afficher tous les caractères spéciaux, il peut être judicieux de rediriger l'affichage du programme dans un fichier texte qu'on pourra ensuite ouvrir avec un éditeur de texte. Après compilation de **petits_tests.c** en un exécutable **petits_tests**, cela se fait par la commande suivante dans le terminal. `./petits_tests > petits_tests.txt`

# Codage des réels en C (complément de cours)

## Exercice 3 Prise en main

Cet exercice doit être fait après une courte présentation au tableau des nombres à virgules flottante.

**Question 1** 🖳 / ✍
Ouvrez un navigateur internet et rendez-vous sur la page
`https://www.h-schmidt.net/FloatConverter/IEEE754.html`.

Essayez de coder divers nombres réels, des positifs, des négatifs, des nombres entre 0 et 1, 0, +l'infini et -l'infini aussi. On peut par exemple essayer de voir comment sont codés $1, \frac{3}{2}, \frac{7}{4}, 3, 2, 1...$

Essayez de résumer quel entier encode le nombre flottant écrit

$$s \;\; e_{k-1}\, e_{k-2} \dots e_0 \;\; m_1\, m_2 \dots m_l$$

*Attention il peut y avoir des cas particuliers, une seule formule ne suffit peut-être pas*

**Question 2** ✍
Est-ce qu'on a vraiment proposé un encodage capable de couvrir tous les réels ?
Quelles sont les limitations ? Quels paramètres de l'encodage jouent sur ces limitations.

## Exercice 4 Égalité ?

**Question 1** 🖳
Faites plusieurs tests d'égalités entre des nombres flottants, ou entre nombre flottants et nombres entiers. Les tests pour lesquels vous attendiez une évaluation à vrai ont-ils tous été évalués à vrai ?

**Question 2** 🖳
Définir une fonction qui teste l'égalité entre deux flottants à $10^{-9}$ près.

# Codage des entiers naturels en base 10

## Exercice 5 Pour commencer...

**Question 1** 🖳
Codez une fonction `affiche_les_chiffres` qui affiche un à un les chiffres en base 10 d'un entier naturel, en commençant par le chiffre des unités.

**Question 2** 🖳
Codez une fonction `somme_des_chiffres` qui calcule la somme des chiffres en base 10 d'un entier naturel.

**Question 3** 🖳
Codez une fonction `miroir` qui prend en entrée un entier naturel et qui retourne l'entier naturel dont l'écriture en base 10 est le miroir de celle du nombre pris en entrée.

## Exercice 6 Critères de divisibilité

**Question 1** ✍
Rappelez à quelle condition sur ses chiffres en base 10 un entier naturel $n$ est-il divisible par 10, par 2, par 5, par 9 ? Prouvez le critère de divisibilité par 9.

**Question 2** 🖳
Codez une fonction `est_multiple_10` qui prend en entrée un entier naturel et qui décide s'il est divisible par 10 en se basant sur les chiffres de ce nombre.

**Question 3** 🖳
Codez une fonction `est_multiple_5` qui prend en entrée un entier naturel et qui décide s'il est divisible par 5, sans utiliser le modulo 5, mais en se basant sur les chiffres de ce nombre.

**Question 4** 🖳
Codez une fonction `est_multiple_9` qui prend en entrée un entier naturel et qui décide s'il est divisible par 9, sans utiliser le modulo 9, mais en se basant sur les chiffres de ce nombre. *Vous pourrez peut-être utiliser une fonction déjà codée dans l'exercice 1.*

## Exercice 7 Numéro de sécurité sociale

Voir exercice 6 de la feuille d'exercices n°1 si vous ne l'avez pas encore fait.

# Codage des entiers naturels en base 2

## Exercice 8 Nombre dicté

**Question 1** 🖳
Codez une fonction `dicte_droite_a_gauche` qui propose à un utilisateur de saisir les chiffres d'un nombre écrit en base 2 de droite à gauche, et qui retourne en sortie l'entier correspondant.

**Question 2** 🖳
Codez une fonction `dicte_gauche_a_droite` qui propose à un utilisateur de saisir les chiffres d'un nombre écrit en base 2 de gauche à droite, et qui retourne en sortie l'entier correspondant.

## Exercice 9 Tour de magie binaire

Voir l'exercice 9 du TP n°2 (qui était déjà l'exercice 7 de la feuille d'exercice 1).
Vous pouvez me demander une démonstration du tour avant de faire l'exercice.