# TD 1

## Exercice 1

Soit le programme C suivant :

```
(1) #include <stdio.h>
(2) int main(void)
(3) {
(4) int v;
(5) float m, Ec;
(6) m = 1e6;
(7) v = 30;
(8) Ec = (m * v * v) / 2;
(9) printf("m = %e v = %d ec = %e\n", m, v, Ec);
(10) return 0;
(11) }
```

1. Dessiner la pile immédiatement après l'exécution de l'instruction 8.
2. Exécuter le programme.

## Exercice 2. Déclarations et identificateurs

Parmi les déclarations suivantes, lesquelles sont correctes et lesquelles sont fausses :

```
float var, var1, VAR2;
real x,y,z;
int void, main;
char RS-232;
char 1carac, 2carac;
int ma_variable;
double var1; var2; var3;
unsigned char C='c';
float réel;
```

## Codages des nombres

### Caractères

Les variables de type `char` sont codées sur 1 octet. Le nombre entier qui les représente correspond au codage ASCII du caractère correspondant. Les codes ASCII principaux sont :

- '0' → 48, '1'→ 49 etc,
- 'A' → 65, 'B' → 66 etc,
- 'a'→ 97, 'b' → 98 etc.

Soit la séquence d'instructions suivante :

```
char c1, c2, c3;
c1 = '0'; c2 = 'A'; c3 = 'a';
```

1. Écrire les octets représentant le stockage des valeurs : c1, c2, c3, c1+2, c2+c3.

## Représentations des entiers

1. Donner la représentation de l'entier 18 dans le type `short int` (2 octets)

   Rappel : pour les nombres écrits en base 2, la représentation de (`-x`) se construit connaissant celle de `x`, en allant de la droite vers la gauche : on laisse les premiers zéros puis le premier un inchangés, on intervertit ensuite tous les autres chiffres. (Remarque, pour le zéro (0000 0000) il n'y a pas de premier un et don on ne fait rien. Cette règle s'appelle également le complément à 2).
2. En déduire la représentation de l'entier `-18` dans le même type
3. On rappelle que le type char permet en fait de représenter des entiers signés sur un seul octet ; quel est l'intervalle des entiers représentables dans ce type ?
4. L'exécution du programme ci-dessous donne les résultats indiqués. Expliquer pourquoi.

```
#include <stdio.h>
int main()
{ char c = 'a';
printf("%d\n",c);
c = c+c;
printf ("%d\n",c);
return 0;
}
97
-62
```

## Représentation des réels en virgule flottante

Le codage des réels en virgule flottante se fait sur 4 octets selon la norme IEEE simple précision comme suit, telle que

$$\text{les 32 bits : } \underbrace{\text{signe}}_{\text{1 bit}}\ \underbrace{\text{exposant}}_{\text{8 bits}}\ \underbrace{\text{mantisse}}_{\text{23 bits}}$$

représentent le réel

$$(-1)^s \times 1, m \times 2^{e-127}$$

avec $s$ le signe, $m$ la mantisse et $e$ l'exposant.

Par exemple. $(3.625)_{10} = 2^1 + 2^0 + \frac{1}{2} + \frac{1}{2^3} = (11.101)_2$. On se ramène ensuite à l'écriture normalisée IEEE $(11.101)_2 = (-1)^0 \times 1.1101 \times 2^1$ ($e = 128$). La représentation de $(3.625)_{10}$ est donc 01000000011010000...000.

On considère la séquence d'instruction suivante :

```
float f, g, h;
f = 2; g = 0.5; h = 0.25;
```

Donner les 4 octets qui représentent les valeurs : `f, g, h, 1.0-g, f+g+h, f+g+h + 10.5`.

# Opérateurs

## 1 Exercice 1

Soit le programme C suivant :

```
#include <stdio.h>
int main(void)
{
int i, j = 2;
float x = 2.5;
i = j + x;
x = x + i;
printf("x = %f \n",x);
return 0;
}
```

1. Qu'affiche ce programme ?
2. pourquoi ?
3. Modifiez le pour qu'il affiche le résultat demandé

### Exercice 2

Soit le programme suivant :

```
{
int i = 5, j=2;
float x, y;
x = i/j;
printf("a = %f \n",x);
}
```

1. Qu'affiche ce programme ?
2. pourquoi ?
3. Modifiez le pour qu'il affiche le résultat demandé

### Exercice 3

On considère la séquence d'instructions suivante :

```
int a = 5, b = 3, c = 2;
int x = 0, y = 0, z = 1;

a = b;                x || (y && z);                  a & b;
a + b - c;            (x == y) || (x != z);           (a | b) & c;
(a * c) / b;          (a < b) != ( !(b >= c) );       (a ^ b) || c;
```

Évaluer chacune des expressions, ou déclarations.