Travaux Dirigés sur le langage *C* n°.1

Cours d'Informatique de Deuxième Année

—Deug SSM M3—

Ce premier *TD* constitue une introduction au langage *C*. Les objectifs sont les suivants :

- Etude de la syntaxe de base du *C*,
- Mise en évidence de quelques différences par rapport au *Pascal*,
- Panorama des types usuels du langage,
- Opérations arithmétiques, booléennes et sur les bits.

Les exercices proposés sont essentiellement des applications de ce que vous avez vu en cours.

Les exercices commençant par ce sigle sont plus subtils dans le sens où ils ne sont pas seulement des applications directes du cours.

Les exercices commençant par ce double sigle demandent encore plus d'attention.

La syntaxe et les entrées-sorties...

▸ **Exercice 1**. Que fait le programme *C* suivant ?

```
#include <stdio.h>

void main(void)
{
  printf("bonjour...\n");
}
```

▸ **Exercice 2**. Ecrire un programme qui lit un caractère sur l'entrée standard, l'affiche sur la sortie standard et passe à la ligne.

▸ **Exercice 3**. Ecrire un programme qui demande à l'utilisateur de rentrer un caractère; le programme devra alors afficher le caractère ainsi que son code ASCII.

▸ **Exercice 4**. Refaire le même exercice, mais cette fois le programme devra afficher le caractère et son code ASCII seulement si c'est une minuscule[1]. Si c'est une majuscule, on la transformera en minuscule puis on fera le même travail. Enfin, si c'est autre chose, on affichera juste son code ASCII.

▸ **Exercice 5**. Donner une **fonction** qui renvoie le minimum entre deux entiers passés en argument.

▸ **Exercice 6**. En déduire le programme complet de calcul du minimum de deux entiers lus sur l'entrée standard.

▸ **Exercice 7**. Ecrire une fonction qui teste l'égalité de deux entiers.

---

(1) On rappelle que les minuscules sont stockées dans le code ASCII de manière contigüe et dans l'ordre alphabétique.

▸ **Exercice 8**. Que dire de la fonction suivante ?

```
int biz(int a, int b)
{ int c=a;
  if (a=b)
  { b=c;
    printf("0\n0\n");
  }
  else
  { printf("%d\n",b);
    a*=2;
    printf("%d\n",a);
  }
  return a==0;
}
```

Les boucles...

▸ **Exercice 9**. Que calcule la fonction suivante ?

```
int calc(int n)
{ int i=2,tmp,un=1,un_1=0;
  if (n==0) return un_1;
  if (n==1) return un;
  while (i<=n)
  { tmp=un;
    un+=un_1;
    un_1=tmp;
    i+=1;
  }
  return un;
}
```

▸ **Exercice 10**. Exprimer avec un autre type de boucle, les instructions suivantes : on précisera par ailleurs la valeur de la variable `i`[2] à la sortie de chaque boucle.

```
for(i=5;i>0;i=i-1){...}

i=5;for(;i>0;i-=1){...}

i=10;while(i>0 || i%2) i-=1;

i=10;while(i>0 && !(i%2)) i-=1;

i=0;do { i=i+1; ...; } while(i!=5);
```

▸ **Exercice 11**. Ecrire un programme qui compte et affiche le nombre de caractères lus sur l'entrée standard.

Exercices supplémentaires

▸ **Exercice 12**. Ecrire une fonction de multiplication d'un entier par 2 sans utiliser l'opérateur `*` : on se servira d'opérations élémentaires sur les bits sans se soucier des problèmes de dépassement de capacité. En déduire selon le même principe, une fonction de multiplication par 2^n.

▸ **Exercice 13**. Ecrire une fonction prenant en argument un `char d` et un `int i`. La fonction devra vérifier que l'entier `i` est bien compris entre 0 et 7, et dans ce cas, positionnera à 1 le `i`-ième bit du `char d`. La valeur retournée sera de toute façon, celle de `d`.

[2] On supposera que `i` est une variable entière signée.