

Architecture des ordinateurs

Corrigé du TP 1: Représentation des nombres entiers

1. Représentation binaire d'un entier positif

Ecrire un programme en C qui donne la représentation binaire sur 8 bits d'un entier compris entre 0 et 255. Pour cela, vous pourrez utiliser la division entière et les modulus. Soit a et b deux entiers Le calcul a/b est appelé *division entière* de a par b et son résultat est la partie entière inférieure de la valeur réelle $\frac{a}{b}$. En C, a modulo b (le reste de la division entière de a par b) s'écrit $a\%b$.

Correction :

```
#include <stdio.h>

int main (void)
{
    int n;

    /* lit la valeur de n */
    scanf ("%d", &n);

    /* puis écrit les différents bits de n (il y en a 8) */
    printf ("%d", (n/128)%2);
    printf ("%d", (n/64)%2);
    printf ("%d", (n/32)%2);
    printf ("%d", (n/16)%2);
    printf ("%d", (n/8)%2);
    printf ("%d", (n/4)%2);
    printf ("%d", (n/2)%2);
    printf ("%d", (n/1)%2);
    printf ("\n");

    return 0;
}
```

2. Écrire un programme en C qui représente un entier positif (entre 0 et 255) en base 8 (octal).

Correction :

```
#include <stdio.h>

int main (void)
{
    int n;

    /* lit la valeur de n */
    scanf ("%d", &n);

    /* représente n en octal */
    printf ("%d", (n/64)%8);
    printf ("%d", (n/8)%8);
    printf ("%d", (n/1)%8);
    printf ("\n");
}
```

```

    return 0;
}

```

La fonction suivante donne, à partir d'un nombre entre 0 et 36, le code ASCII du caractère le représentant : (10→A, 11→B, ..., 35→Z). Recopiez-la dans votre code source (entre les « #include » et le « main() ») pour pouvoir l'utiliser.

```

int d2c (int n)
{
    if (0 <= n && n < 10)
        return '0' + n;
    else if (n < 36)
        return 'A' + (n - 10);
    else
        return '?';
}

```

3. Écrire un programme en C qui représente un entier positif (entre 0 et 255) en base 16 (hexadécimal).

Correction :

```

#include <stdio.h>
#include "d2c.c"

int main (void)
{
    int n;

    /* lit la valeur de n */
    scanf ("%d", &n);

    /* représente n en hexadécimal */
    printf ("%c", d2c (n/16));
    printf ("%c", d2c (n%16));
    printf ("\n");

    return 0;
}

```

4. Représentation en base quelconque

Utiliser cette fonction pour écrire un programme qui représente un entier positif dans une base quelconque entre 1 et 36.

Correction :

```

#include <stdio.h>
#include "d2c.c"

int main(void)
{
    int n, b, d;

    /* lit la valeur de n */
    printf ("votre nombre: ");
    scanf ("%d", &n);

    /* lit la valeur de la base b */
    printf ("votre base: ");
    scanf ("%d", &b);
}

```

```

    /* cherche la plus grande puissance de b inférieure à n */
    d = 1;
    while (d <= n)
        d = d * b;
    d = d / b;

    /* parcourt les différentes puissances de b dans l'ordre décroissant */
    while (d > 0) {
        printf ("%c", d2c ((n/d)%b));
        d = d / b;
    }
    printf ("\n");

    return 0;
}

```

5. Valeur maximale

Ecrire un programme qui donne la valeur maximale d'un entier. Pour cela, on part d'un entier de valeur 1 et on le multiplie par 2 jusqu'à atteindre un dépassement de capacité, c'est-à-dire jusqu'à revenir à 0. Si pour cela, le nombre a été multiplié n fois, la valeur maximale est $2^n - 1$. Tester ce programme avec un entier signé (type `int` en C) puis avec un entier non signé (type `unsigned int`).

Correction :

```

#include <stdio.h>

main() {
    unsigned int n = 1, i;

    i=0;
    while ( n > 0) {
        n = n*2;
        i++;
    }
    printf("valeur max : 2^%d - 1\n", i);
}

/*
 * valeur max : 2^32 - 1
 *
 * avec int :
 * valeur max : 2^31 - 1
 *
 */

```