Rapport du TP 5 d'IGI-2001

Rémi Nicole

Chapitre 1

Exercice 1

Listing 1.1 – Programme

```c
#include <stdio.h>

float max3float(float f1, float f2, float f3)
{
    float temp;
    return (f1 > ( temp = (f2 > f3) ? f2 : f3 ) ? f1 : temp);
}

int main () {
    float x1 = 3.5, x2 = 10. , x3 = 25.3 , res = 0.;
    printf("%f\n", res = max3float(x1, x2, x3));
    return 0;
}
```

Listing 1.2 – Résultat

```
25.299999
```

Chapitre 2

Exercice 2

Listing 2.1 – Programme

```c
#include <stdio.h>

void affiche_bin(short x)
{
    int i, m;
    unsigned short sz = sizeof(x) * 8;
    for (i = 0, m = 0x8000; i < sz; ++i, m >>= 1) {
        printf("%i", (x & m) >> ((sz-1)-i));
    }
    printf("\n");
}

int main () {
    unsigned short x = 0b0000111000101100,
                   masque1 = 0b1111111100000000,
                   masque2 = 0b0000000011111111,
                   poidsfort = x & masque1,
                   poidsfaible = x & masque2;

    // %x allows hexadecimal representation
    // But there is no binary representation conversion specifier
    printf("masque 1 : %d => %x\n", masque1, masque1);
    printf("masque 2 : %d => %x\n", masque2, masque2);
    printf("Poids fort de x : %d => %x\n", poidsfort, poidsfort);
    printf("Poids faible de x : %d => %x\n", poidsfaible, poidsfaible);
    affiche_bin(x);

    return 0;
}
```

Listing 2.2 – Résultat

```
masque 1 : 65280 => ff00
masque 2 : 255 => ff
Poids fort de x : 3584 => e00
Poids faible de x : 44 => 2c
0000111000101100
```

2.1 Question 1

Il existe un spécificateur de conversion qui affiche une variable en hexadécimal (soit `x` ou `X`) mais il n'existe pas de spécificateur de conversion pour afficher une variable en binaire.

Chapitre 3

Exercice 3

Listing 3.1 – Programme

```c
#include <stdio.h>

int main () {
    int tab[100], i;
    for (i = 0; i < 100; ++i) {
        tab[i] = i;
        printf("%i ", tab[i]);
    }
    printf("\n");
    return 0;
}
```

Listing 3.2 – Résultat

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
    23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
    42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
    61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
    80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
    99
```

Chapitre 4

Exercice 4

Listing 4.1 – Programme

```c
#include <stdio.h>
#include <stdlib.h>

// Go through the table and remove all non prime multiple of the prime parameter
void removeMultiples(const int prime, int* tab, const int size)
{
    int i;
    // start with prime as index as the table is ordered
    for (i = prime + 1; i < size; i++) {
        if(tab[i] % prime == 0)
            tab[i] = 0;
    }
}

int main (int argc, char const* argv[]) {
    if (argc != 2)
        printf("Usage : %s N\n", argv[0]);
    else {
        int n = atoi(argv[1]), tab[n], i;

        // Generate the table
        for (i = 0; i < n; ++i) {
            tab[i] = i + 1;
        }

        printf("| ");
        for (i = 1; i < n; ++i) {
            if(tab[i] != 0) {
                removeMultiples(tab[i], tab, n);
                printf("%i \t| ", tab[i]);
            }
        }
```

```
33          printf("\n");
34      }
35      return 0;
36  }
```

Listing 4.2 – Résultat

```
| 2      | 3      | 5      | 7      | 11     | 13     | 17     | 19
        | 23     | 29     | 31     | 37     | 41     | 43     | 47
       | 53     | 59     | 61     | 67     | 71     | 73     | 79
       | 83     | 89     | 97     |
```