Travaux Dirigés d'algorithmique n°1

Cours d'informatique de Deuxième Année

—L2.1—

Quelques exercices d'introduction

Quelques exercices de remise en forme et introduction de la notion de temps nécessaire à l'exécution d'un algorithme.

▸ Exercice 1. (Révisions)

Que font les programmes suivants ?

1. 1^{er} *programme :*

```
 1.  #include <stdio.h>
 2.  #define MAX 2
 3.
 4.  void toto(int *n, int p)
 5.  {
 6.    *n = 3;
 7.     p = 2;
 8.    *n = *n+p;
 9.  }
10.
11.  void truc(int *n, char *c)
12.  {
13.    *n = *n+1;
14.    *c = 'X';
15.  }
16.  int main(void)
17.  {
18.    char t[MAX];
19.    int i,j;
20.
21.    toto(&i, j);
22.    printf("i=%d, j=%d\n", i,j);
23.    t[0] = 'a';
24.    t[1] = 'b';
25.    printf("%c %c \n", t[0], t[1]);
26.    i = 0;
27.    truc(&i, &t[i]);
28.    printf("%d %c %c \n", i, t[0], t[1]);
29.  }
```

2. 2^{eme} *programme :*

```
 1.  #include <stdio.h>
 2.
 3.  int bidule(int y,int *x)
 4.  {
 5.    int p=1,i=1;
 6.
 7.    if (y<=0) return 0;
 8.
 9.    while (y>p)
10.    {
11.     i++;
12.     p=p*i;
13.    }
14.    if (y!=p) return 0;
15.
16.    *x=i;
17.    return i;
18.  }
19.  int main(void)
20.  {
21.    int a,r;
22.
23.    a=-1;
24.    r=bidule(5,&a);
25.    printf("%d \n",r);
26.  }
```

On indiquera l'état de chaque variable au cours de l'exécution.

▸ **Exercice 2. (Multiplication de deux entiers)**

1. *On suppose que la touche* `'*'` *de votre ordinateur est en panne. Comment faire pour calculer le produit de deux entiers n et m ?*
   Donner une estimation du nombre d'opérations effectuées par l'ordinateur pour appliquer votre méthode ?
2. *Multiplication égyptienne*
   Les anciens Egyptiens savaient additionner deux entiers, mutiplier et diviser un entier par 2. Pour multiplier deux entiers, ils utilisaient l'algorithme suivant :

```
fonction Mult(n, m entiers, n ≥ 0) retourne un entier
{
   x ← n ; y ← m ; r ← 0 ;
   tantque x > 0 faire {
      si x est impair alors r ← r + y ;
      x ← quotient(x, 2) ;
      y ← 2y ;
   }
   retour(r) ;
}
```

   (a) *Décomposer le calcul pour $n = 19$ et $m = 25$.*
   (b) *Estimer le nombre d'opérations effectuées pour appliquer cette méthode.*
   (c) *Le traduire en C. Que faut-il modifier pour avoir le produit de deux entiers quelconques ?*

▸ **Exercice 3. *(Rotation dans un tableau)***

Le but de cet exercice est d'effectuer, dans un tableau de n caractères, la rotation à gauche de k positions ($0 < k < n$). Par exemple, pour $n = 8$ et $k = 3$, le tableau | A | B | C | D | E | F | G | H | *devient, après rotation,* | D | E | F | G | H | A | B | C |

Pour celà on propose trois solutions :

1. *Utiliser un tableau auxiliaire.*
2. *Ecrire tout d'abord une fonction qui effectue une rotation à gauche d'une seule position puis l'utiliser pour résoudre le problème.*
3. *Ecrire une fonction qui inverse les éléments de la portion de tableau délimitée par deux indices i et j (par exemple si $i = 2$ et $j = 5$, la fonction transforme le tableau précédent en* | A | B | F | E | D | C | G | H |*), puis utiliser cette fonction pour résoudre le problème.*

▸ **Exercice 4. (Les structures)**

On définit la structure `Point_2D` *de la manière suivante :*

```
struct point_2D{
  int x;
  int y;
};

typedef struct point_2D point;
```

1. *Ecrire une fonction* `float distance(point pt1,point pt2)` *qui renvoit la distance entre deux point.*
2. *Ecrire une fonction* `void translation(point * pt,int dx,int dy)` *qui fait translater le point pt à l'aide des valeurs données par dx et dy*