Initiation à la programmation en C

Correction du TP n°2

Antoine Miné

22 février 2007

Site du cours: `http://www.di.ens.fr/~mine/enseignement/prog2006/`

Exercice 1. Affichage.

```
void ligne(char c,int n)
{
  while (n>0) {
    printf("%c",c);
    n--;
  }
}

void boite(int largeur, int hauteur, int epaisseur)
{
  int i=0;
  while (i<epaisseur) {
    ligne('#',largeur + 2*epaisseur);
    printf("\n");
    i++;
  }
  i=0;
  while (i<hauteur) {
    ligne('#',epaisseur);
    ligne(' ',largeur);
    ligne('#',epaisseur);
    printf("\n");
    i++;
  }
  i=0;
  while (i<epaisseur) {
    ligne('#',largeur + 2*epaisseur);
    printf("\n");
    i++;
  }
}
```

Exercice 2. Exponentielle.

version naïve

```
double puissance(double x,int n)
{
  double r = 1;
  while (n>0) {
```

```
    r *= x;
    n--;
  }
}
```

version rapide

```
double puissance(double x,int n)
{
  double y;
  if (n==0) return 1.;
  y = puissance(x,n/2);
  y = y*y;
  if (n%2) y *= x;
  return y;
}
```

Exercice 3. Exponentielle instrumentée.

```
int compteur;

double mul(double x,double y)
{
  compteur++;
  return x*y;
}

double puissance_interne(double x,int n)
{
  double y;
  if (n==0) return 1.;
  y = puissance_interne(x,n/2);
  y = mul(y,y);
  if (n%2) y = mul(y,x);
  return y;
}

double puissance(double x,int n)
{
  double y;
  compteur = 0;
  y = puissance_interne(x,n);
  printf("puissance par %i: %i appels récursifs\n",n,compteur);
}
```

Exercice 4. Fibonacci récursif.

```
#include <assert.h>
int fibo(int n)
{
  assert(n>=0);
  if (n<=1) return 1;
  return fibo(n-1)+fibo(n-2);
}
```

On utilise l'instruction `assert`, disponible après avoir inclus `assert.h` et qui permet d'interrompre le programme proprement quand une condition n'est pas vérifiée.

Cette version est beaucoup plus lente que celle du TP précédent. Ceci s'explique par le nombre élevé de calculs redondants. Par exemple, `fibo(n-2)` sera calculé deux fois : une fois par appel direct à `fibo(n-2)` dans `fibo(n)`, et une fois depuis `fibo(n-1)` appelé par `fibo(n)`. `fibo(n-3)` sera calculé trois fois, etc. On peut déterminer que le nombre d'opérations effectués par `fibo(n)` est proportionnel à... la valeur de `fibo(n)`.

Exercice 5. Booléens.

1. `!expr`, c'est à dire `expr==0`
2. voici une solution : `(expr1?!expr2 :!!expr2)`
   en voici une autre : `((!!expr1)+(!!expr2) % 2)`
3. `a<=b<=c` se lit comme `(a<=b)<=c` ; selon l'ordre de `a` et `b`, `c` sera comparé à `0` ou `1`. Le message important de l'exercice est que c'est très différent de l'interprétation naïve "`b` est entre `a` et `c`", qui s'écrirait `(a<=b) && (b<=c)` (`b` peut alors être évalué deux fois).

Exercice 6. Tortue Logo.
Voici le programme complet :

```
#include <stdio.h>

double x = 297,y = 419; /* milieu de la page */
int dir = 0;            /* vers le nord */

void tourne_droite()
{
  dir = (dir+1) % 4;
}

void tourne_gauche()
{
  dir = (dir+3) % 4; /* -1 modulo 4 */
}

void avance(double pas)
{
  printf("%f %f moveto ",x,y);
  if (dir==0) y += pas;      /* nord  */
  else if (dir==1) x += pas; /* est   */
  else if (dir==2) y -= pas; /* sud   */
  else if (dir==3) x -= pas; /* ouest */
  printf("%f %f lineto stroke\n",x,y);
}

void courbe(int n)
{
  if (n==0) avance(3);
  else {
    tourne_droite();
    courbe(n-1);
    tourne_gauche();
    courbe(n-1);
  }
}

int main()
{
  printf("%%!postscript\n");
  courbe(12);
```

```
    printf("showpage\n");
    return 0;
}
```

et la courbe dessinée :