IN 101 - TD 03

28 septembre 2011

Corrigé

ENSTA ParisTech

Solution 1 : Date

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int bissextile(int annee) {
5    if ((annee % 400) == 0) {
6      return 1;
7    }
8    if ((annee % 100) == 0) {
9      return 0;
10   }
11   if ((annee % 4) == 0) {
12     return 1;
13   }
14   return 0;
15 }
16
17 int numero_jour(int jour, int mois, int annee) {
18   int i, r;
19   int duree_mois[] = {31,28,31,30,31,30,31,31,30,31,30,31};
20   if (bissextile(annee) == 1) {
21     duree_mois[1]=29;
22   }
23   r = jour;
24   for (i=0; i<mois-1; i++) {
25     r = r + duree_mois[i];
26   }
27   return r;
28 }
29
30 int nombre_jours(int jour, int mois, int annee) {
31   int i, n;
32   n = numero_jour(jour, mois, annee);
33   for (i=2000; i<annee; i++) {
34     if (bissextile(i) == 1) {
35       n += 366;
36     } else {
37       n += 365;
38     }
39   }
40   return n-1;
41 }
42
43 int main (int argc, char* argv[]) {
44   int jour, mois, annee;
```

```
45   jour = atoi(argv[1]);
46   mois = atoi(argv[2]);
47   annee = atoi(argv[3]);
48   printf("%d jours écoulés depuis le 1er janvier 2000.\n",
49                                       nombre_jours(jour,mois,annee));
50   return 0;
51 }
```

Solution 2 : Programmation récursive

2.a] La fonction factorielle :

```
1 long long factorielle(int n) {
2   if (n < 2) {
3     return 1;
4   }
5   return n * factorielle(n-1);
6 }
```

2.b] Le PGCD :

```
1 int pgcd(int x, int y){
2   if (y == 0) {
3     return x;
4   }
5   return pgcd(y, x%y);
6 }
```

2.c] L'exponentiation :

```
1 int expo (int x, int e) {
2   if (e==0) {
3     return 1;
4   }
5   if (e%2 == 0) {
6     return expo(x*x,e/2);
7   }
8   return expo(x,e-1)*x;
9 }
```

Solution 3 : Suite de Fibonacci

3.a] La version récursive :

```
1 long long fiborec(int n){
2   if (n < 2) {
3     return 1;
4   }
5   return fibo(n-1) + fibo(n-2);
6 }
```

3.b] La version utilisant un tableau :

```
#define N 100
long long fibotab(int n) {
  int i;
  long long tab[N];
  tab[0] = 1;
  tab[1] = 1;
  for (i=2; i<=n; i++) {
    tab[i] = tab[i-1] + tab[i-2];
  }
  return tab[n];
}
```

3.c] La version intelligente, sans tableau :

```
long long fibo(int n) {
  int i;
  long long a, b, c;
  a = 1;
  b = 1;
  for (i=2; i<=n; i++) {
    c = a + b;
    a = b;
    b = c;
  }
  return b;
}
```

Solution 4 : Conjecture de Collatz

```
#include <stdio.h>
#include <stdlib.h>
int collatz(int n) {
  if (n == 1) {
    return 1;
  } else if (n%2 == 0) {
    return n/2;
  }
  return 3*n+1;
}
int main(int argc, char* argv[]){
  int n = atoi(argv[1]);
  int cpt = 0;
  while(n != 1) {
    cpt = cpt+1;
    printf("%d ", n);
    n=collatz(n);
  }
  printf("\nLe nombre d'étape est %d.\n", cpt);
  return 0;
}
```

Solution 5 : Problème des 8 reines (à la maison)

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

int posDames[8] = {0}; /* on met une dame par ligne, il faut donc
      uniquement se souvenir de la position sur la ligne dans un int */
int solution = 0;      // nombre de solutions

int abs(int n){
  if (n < 0) {
    return -n;
  }
  return n;
}

void recursive(int nDames){ /* nDames = ligne en cours et dames restantes */
  int i,j,s;
  if (nDames == 8) {     // si on atteint 8 on a une solution
    for(i=0; i<8; i++){  // on l'affiche
      for (j=0; j<8; j++) {
        if (j == posDames[i]) {
          printf("1 ");
        } else {
          printf("0 ");
        }
      }
      printf("\n");
    }
    printf("\n\n");
    solution++;
  }

  for (i=0; i<8; i++) { // on remplit les cases en partant du haut
    s=0;                // detecte si la position est permise
    for (j=0; (j<nDames)&&(s==0); j++) {
      if (posDames[j]==i || (abs(posDames[j]-i) == abs(j-nDames))) {
        s=1;
      }
    }
    if(s != 1) {        // si s==0 la place est possible et on continue
      posDames[nDames]=i;
      recursive (nDames+1);
    }
  }
}

int main(int argc, char* argv[]){
  recursive(0);
  printf("\nNombre de solutions: %d.\n", solution);
  return 0;
}
```