Programmation en C

Corrigé TD3

1 octobre 2010

```
 1 /* Nombre de jours
 2 ---------------------------------- */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6
 7 int bissextile(int annee)
 8 {
 9   if ((annee %  4) != 0) return 0;
10   if ((annee % 400) == 0) return 1;
11   if ((annee % 100) == 0) return 0;
12   return 1;
13 }
14
15 int numero_jour(int jour,int mois,int annee)
16 {
17   int i,r=jour;
18   int duree_mois[]={31,28,31,30,31,30,31,31,30,31,30,31};
19   if(bissextile(annee)==1) duree_mois[1]=29;
20   for (i=0;i<mois-1;i++) r = r + duree_mois[i];
21   return r;
22 }
23
24 int nombre_jour(int jour,int mois,int annee)
25 {
26   int i,n=0;
27   int r = numero_jour(jour,mois,annee);
```

```
28   for (i=2000;i<annee;i++)
29     if (bissextile(i)==1) n += 366; else n += 365;
30   n = n+r;
31   return n-1;
32 }
33
34 int main (int argc, char *argv[])
35 {
36   int jour,mois,annee;
37   jour = atoi(argv[1]);
38   mois = atoi(argv[2]);
39   annee = atoi(argv[3]);
40   printf("%d jours écoulés \n",nombre_jour(jour,mois,annee));
41
42   return EXIT_SUCCESS;
43 }
```

```
 1 /* Factorielle Recursive
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6
 7 long long factorielle(int n) {
 8   if (n<2) return 1;
 9   return n*factorielle(n-1);
10 }
11
12 int main (int argc, char *argv[]) {
13   int a = atoi(argv[1]);
14   printf("Factorielle(%d) = %Ld \n",a,factorielle(a));
15   return EXIT_SUCCESS;
16 }
```

```
 1 /* Programme de calcul de PGCD de deux entiers
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
```

```
 5 #include <stdio.h>
 6
 7 int pgcd(int x, int y){
 8   if (y==0) return x;
 9   return pgcd(y,x%y);
10 }
11
12 int main (int argc, char *argv[])
13 {
14   int a=atoi(argv[1]);
15   int b=atoi(argv[2]);
16   int c= pgcd(a,b);
17   printf("pgcd (%d,%d) = %d\n",a,b,c);
18   return EXIT_SUCCESS;
19 }
```

```
 1 /* Exponentielle binaire
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6
 7 int expo (int x, int e) {
 8   if (e==0) return 1;
 9   if (e%2 == 0) return expo(x*x,e/2);
10   return expo(x,e-1)*x;
11 }
12
13
14 int main (int argc, char *argv[])
15 {
16   int a = atoi(argv[1]);
17   int b = atoi(argv[2]);
18   printf("%d ^ %d = %d \n",a,b,expo(a,b));
19   return EXIT_SUCCESS;
20 }
```

```
1 /* Suite de Fibonacci - Recursivite
```

```
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6
 7 long long cpt=0;
 8
 9 long long fibo(int n){
10
11   cpt++;
12   if(n<2) return 1;
13   return fibo(n-1)+fibo(n-2);
14 }
15
16 int main (int argc, char *argv[])
17 {
18   int n = atoi(argv[1]);
19   long long f=0;
20
21   f = fibo(n);
22
23   printf("Fibo(%d) = %lld\nEtapes = %lld\n",n,f,cpt);
24   return EXIT_SUCCESS;
25 }
```

```
 1 /* Suite de Fibonacci avec tableau
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6 #define N 10
 7
 8 int main (int argc, char *argv[])
 9 {
10   int tab[N], i;
11
12   tab[0] = 1; tab[1] = 1;
13   for (i=2;i<N;i++) tab[i] = tab[i-1] + tab[i-2];
```

```
14
15   for (i=0;i<N;i++) printf("%d ",tab[i]);
16   printf("\n");
17   return EXIT_SUCCESS;
18 }
```

```
 1 /* Suite de Fibonacci
 2  ------------------------------------------------ */
 3
 4 #include <stdlib.h>
 5 #include <stdio.h>
 6
 7 int main (int argc, char *argv[])
 8 {
 9   int a=1,b=1,tmp;
10   int i, n = atoi(argv[1]);
11
12   for (i=1;i<n;i++) {
13     tmp = a+b;
14     a = b;
15     b = tmp;
16   }
17
18   printf("Fibo(%d) = %d\n",n,b);
19   return EXIT_SUCCESS;
20 }
```

```
 1 /* Collatz
 2 ------------         */
 3
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 int collatz(int n) {
 8
 9   if (n==1) return 1;
10   if (n%2==0) return n/2;
11   return 3*n+1;
```

```
12 }
13
14 int main(int argc, char *argv[]){
15
16   int n = atoi(argv[1]);
17   int cpt = 0;
18
19   while(n != 1) {
20     cpt = cpt+1;
21     printf("%d ",n);
22     n=collatz(n);
23   }
24   printf("\n Le nombre d'etape est %d\n",cpt);
25
26   return 0;
27 }
```

```
 1 /* Programme du calcul des 8 reines et nombre de solutions
 2 ----------------------------------- */
 3
 4 #include <stdio.h>
 5 #include <stdlib.h>
 6
 7 int posDames[8] = {0}; /* je mets une dame par ligne, il faut donc
 8 uniquement se souvenir de la position sur la ligne dans un int */
 9 int solution = 0; /* nombre de solution */
10
11 int abs(int n){
12
13   if (n<0) return -n;
14   else return n;
15 }
16
17 void recursive(int nDames){ /* nDames = ligne en cours et dames restantes */
18
19   int i,j,s;
20
21   if (nDames == 8) {/* si j'arrive a 8, en partant de 0, affiche le resultat */
```

```
22      for(i=0; i<8;i++){
23        for (j=0; j<8; j++)
24      if (j==posDames[i]) printf("1 ");
25      else printf("0 ");
26        printf("\n");
27      }
28      printf("\n\n");
29      solution++;
30    }
31
32    for (i=0; i<8; i++) { /* je remplis les cases en partant du haut */
33      s=0; /* detecte si la position est permise */
34      for (j=0; (j<nDames)&&(s==0); j++)
35        if (posDames[j]==i || (abs(posDames[j]-i) == abs(j-nDames)))
36      s=1;
37      if(s!=1) { /* si s==0, la place est posible et on continue */
38        posDames[nDames]=i;
39        recursive (nDames+1);
40      }
41    }
42  }
43
44  int main(int argc, char *argv[]){
45    recursive(0);
46    printf("\nNombre de solutions: %d\n",solution);
47    return 0;
48  }
```