Initiation à la programmation en C

TP n°3

Antoine Miné

1er mars 2007

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

Exercice 1. Dates.
Programmez une fonction qui prend en argument un numéro de jour (entier entre 1 et 31), de mois (entier entre 1 et 12) et renvoie le numéro de jour dans l'année (entre 1 et 366). On utilisera pour cela un tableau contenant le nombre de jours de chaque mois.

Facultatif : modifiez votre fonction pour qu'elle gère les années bissextiles (tous les 4 ans, sauf les multiples de 100 qui ne sont pas multiples de 400).

Exercice 2. Crible d'Ératosthène.
La méthode d'Ératosthène détermine la liste des nombres premiers inférieurs à n de la manière suivante : pour chaque i de 1 à n, on biffe tous les multiples stricts de i inférieurs à n. À la fin, les nombres non biffés sont exactement les nombres premiers. Une optimisation importante consiste à sauter les i qui ont déjà été biffés. En effet, tous les multiples d'un tel i ont été biffés au moment où i a été biffé !
Programmez la méthode d'Ératosthène.

Voici quelques indications pour vous aider à évaluer la vitesse et l'occupation mémoire de votre programme :
- la commande `time`, en préfixe du programme `a.out`, indiquera son temps d'exécution,
- coupez les sorties sur écran pour ne comptabiliser que le temps de calcul, pas celui de l'affichage,
- la commande `top`, à lancer en parallèle dans un autre terminal, indiquera l'occupation mémoire,
- analysez l'influence des options de `gcc` d'optimisation en vitesse (`-O1` à `-O3`) et en taille (`-Os`),
- analysez l'influence des types choisis sur la vitesse d'exécution et l'occupation mémoire,
- comparez avec la méthode "naïve" programmée lors du premier TP.

Exercice 3. Tri de tableaux.
On cherche à trier dans l'ordre croissant les éléments d'un tableau d'entiers. Il existe de très nombreuses méthodes de tri. Voici celle dite du "tri à bulles" : tant que le tableau n'est pas trié, on le parcourt du premier au dernier indice en échangeant les éléments consécutifs qui sont dans le mauvais ordre. Si, lors d'une passe, aucune paire d'éléments n'a été échangée, alors le tableau est trié.

Testez votre tri sur quelques tableaux d'un millier d'éléments :
- un tableau déjà ordonné,
- un tableau dans l'ordre décroissant,
- un tableau initialisé aléatoirement ; on pourra utiliser pour cela la fonction standard `lrand48` (disponible après un `#include <stdlib.h>`) qui renvoie un nouveau nombre au hasard à chaque appel.

Il y deux manières de tester votre tri :
- écrire une fonction d'affichage du tableau et laisser l'utilisateur vérifier de visu,

– écrire une fonction qui vérifie automatiquement que le tableau est bien trié en comparant les couples d'éléments consécutifs.

Note culturelle : dans la pratique, c'est la méthode dite du "tri rapide" qui est employée. Elle est plus complexe à programmer mais, comme son nom l'indique, plus rapide.

Exercice 4. Traduction de boucles.
Traduisez les boucles suivantes en boucles `while` :

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

int main()
{
  int x;
  for (x=10;x<100;x++)
    printf("%i\n",x);
  return 0;
}
```

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

void f(int x, int y)
{
  do {
    if (x%2) { x++; continue; }
    printf("%i\n",x);
    x += 2;
  }
  while (x<y);
}

int main()
{
  f(10,20);
  f(20,10);
  f(1,5);
  return 0;
}
```

Exercice 5. Recherche dans un tableau.
On se donne un tableau T de n entiers et on cherche à savoir si un entier x donné s'y trouve. Une méthode simple consiste à parcourir T à partir de l'indice 0 jusqu'à trouver x (succès) ou atteindre la fin du tableau (échec). Si T est trié en ordre croissant on peut faire légèrement mieux : on peut s'arrêter dès qu'on trouve un élément strictement plus grand que x (échec). Programmez ces deux méthodes.

On propose maintenant une méthode beaucoup plus efficace dans le cas où T est trié : la *recherche dichotomique*. Elle commence par comparer x à l'élément d'indice $n/2$. Si $x = T[n/2]$, on a gagné. Si $x < T[n/2]$, alors on continue notre recherche dans la plage d'indices de 0 à $n/2 - 1$. Enfin, si $x > T[n/2]$, on continue avec la plage d'indices de $n/2 + 1$ à $n - 1$. À chaque étape, on divise ainsi par deux le nombre d'indices à considérer ! La dichotomie s'arrête quand x est trouvé ou quand la plage d'indices est vide (élément absent).