Initiation à la programmation en C

Correction du TP n°4

Antoine Miné

8 mars 2007

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

Exercice 1. Échange.

```
void echange(int* a, int* b)
{
  int temp = *a; /* on a besoin d'une variable temporaire */
  *a = *b;
  *b = temp;
}
```

`echange(&x,&x) ;` n'aura aucun effet sur `x` et est parfaitement valide (si `x` est une variable de type `int`, bien sûr).

Exercice 2. Tri par insertion.

```
void tri(int* tab, /* pointeur sur le premier élément du tableau */
         int  nb   /* taille du tableau */ )
{
  int *min; /* pointeur vers le plus petit élément */
  int i,j;  /* compteurs de boucle */

  for ( i=0; i<nb; i++ ) {

    /* recherche d'un pointeur sur le petit élément de tab[i]...tab[nb-1] */

    /* on commence par tab[i] */
    min = &tab[i];

    /* puis on parcours tab en mettant min à jour si on trouve un élément
       plus petit */
    for ( j=i+1; j<nb; j++ )
      if ( tab[j] < *min ) min = &tab[j];

    /* on l'échange avec tab[i] */
    echange( &tab[i], min );
  }
}
```

Exercice 3. Lecture au clavier.
Fonction de lecture, d'affichage et programme principal :

```c
#include <stdio.h>

/* fonction de lecture dans table */
int lit_clavier(int* tab, int max)
{
  int nb = 0;

  /* boucle tant qu'on a lu strictement moins de 100 éléments */
  while ( nb < max ) {

    /* lit un élément */
    scanf("%i", &tab[nb]);

    if ( tab[nb] == -1 )
      /* on a bien lu nb éléments, d'indice 0 à nb-1 */
      return nb;

    /* un élément valide de plus! */
    nb++;
  }

  return nb;
}

/* fonction d'affichage de tab[0]... tab[nb-1] */
void affiche(int* tab, int nb)
{
  int i;
  for (i=0;i<nb;i++)
    printf("%i\n",tab[i]);
}

int main()
{
  int table[100];
  int nb_lus = lit_clavier(table,100);
  tri(table,nb_lus);
  affiche(table,nb_lus);
  return 0;
}
```

Exercice 4. Triangle de Pascal.

```c
#include <stdio.h>

#define N 10

void calcule()
{
  int lin, col;

  for ( lin=0; lin<N; lin++ ) {

    /* premier élément de la colonne */
    pascal[lin][0] = 1;

    /* milieu de la colonne */
    for ( col=1; col<lin-1; col++ )
      pascal[lin][col] = pascal[lin-1][col-1] + pascal[lin-1][col];
```

```
    /* dernier élément de la colonne */
    pascal[lin][col] = 1;

  }
}

void affiche()
{
  int lin, col;
  for ( lin=0 ; lin<N; lin++ ) {
    for ( col=0; col<lin; col++ )
      /* le format %4i indique qu'on affiche:
         - un entier (comme %i)
         - 4 caractères au moins, en complétant au besoin avec des blancs
      */
      printf("%2i ",pascal[lin][col]);
    printf("\n");
  }
}
```

Exercice 5. Le retour du triangle de Pascal.
La solution proposée ne marche pas car à une itération `j` de la boucle, on écrase la valeur de `ligne[j]` qui sera nécessaire lors de l'itération `j+1`.
Plusieurs solutions sont envisageables :
– garder l'ancienne valeur de `ligne[j]` dans une variable,
– parcourir la boucle de `i-1` à `1` au lieu de `1` à `i-1`.

Exercice 6. Tableaux et pointeurs.
`f1`, `f2` et `f3` sont strictement identiques.
`g1` renvoie la taille de 5 `int` en octets, c'est à dire 20. `g2` n'est pas correct : on ne peut pas utiliser un tableau de taille inconnue dans `sizeof`. `g3` renvoie la taille d'un pointeur, c'est à dire 4 sur une machine 32-bit, 8 sur une machine 64-bit.
`h1` affichera le contenu des 5 premières cases de `b`, c'est à dire, des valeurs aléatoires (`b` n'est pas initialisé). `h2` provoquera une erreur à la compilation car on ne peut pas déclarer un tableau de taille inconnue. `h3` plantera le programme car il fait lire à `f1` le contenu de la mémoire à l'adresse `b` aléatoire (`b` n'est pas initialisé). Il s'agit là de trois erreurs de type très différent !

Triangle de Pascal binaire