Travaux Pratiques d'algorithmique n°2

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Pointeurs et tris

Dans cette séance, nous aborderons la gestion des pointeurs sous divers aspects (rappels de syntaxe, manipulation des structures (opérateur ->)).
Nous verrons ensuite deux tris naïfs.

▸ Exercice 1.

On considère le programme suivant :

```
#include <stdio.h>
int f( int *x, int y )
{
   (*x)++;
   return *x+y;
}
void main(void)
{
  int *ad1, *ad2, *ad3;
  int n = 10, p = 20 ;
  ad1 = &n;
  ad2 = &p;
  *ad1 = 3;
  *ad2 = *ad1 + *ad2 + p;
  printf("n = %d, p =  %d, *ad1 = %d, *ad2 =  %d\n", n, p, *ad1, *ad2 );
  p = f( ad1, *ad2 );
  printf("n = %d, p =  %d, *ad1 = %d, *ad2 =  %d\n", n, p, *ad1, *ad2 );
  *ad3 = *ad2;
  printf("*ad2 = %d, *ad3 =  %d\n", *ad2, *ad3 );
}
```

Ce programme est syntaxiquement correct mais produit un `core` *à l'exécution. Pourquoi ?*

▸ Exercice 2. *(Permutation circulaire)*

Écrire une fonction permettant d'effectuer une permutation circulaire sur quatre éléments réels. Le tableau suivant illustre le comportement de la fonction sur un exemple :

Variables	*a*	*b*	*c*	*d*
Avant	*1.8*	*23.1*	*10.3*	*-4.0*
Après	*23.1*	*10.3*	*-4.0*	*1.8*

▸ **Exercice 3.** ***(Mise à jour de données)*** *Écrire deux fonctions permettant de saisir et d'afficher le contenu d'une variable de type* `Personne` *défini ci-dessous.*

```
struct personne
{ char          Nom[MAX];
  unsigned int  Annee_Naissance;
  float         Credit;
};

typedef struct personne Personne;
```

Dans le cadre de la mise à jour des bases de données en vue du passage à l'euro et à l'an 2000*, écrire deux fonctions modifiant une* `Personne` *:*

- *la fonction* `PassageEuro` *qui mettra à jour la valeur du crédit initialement en francs (1 euro = 6.55957 francs) ;*
- *la fonction* `PassageAn2000` *modifiera le champ* `Annee_Naissance` *au cas il où vaudrait entre* 0 *et* 99 *(pour éviter une confusion de siècle).*

▸ **Exercice 4.** ***(Tri par sélection)***

Écrire une fonction `void TriSelection(double t[], int n);` *qui effectue le tri par sélection d'un tableau de* `double` *de taille* `n`*.*

Le principe de ce tri est de rechercher le minimum de la suite $(t_0, t_1, ..., t_{n-1})$ *et de l'échanger avec* t_0*. Ainsi, après un parcours, l'élément les plus petit se trouve en première position. On recommence alors en échangeant le minimum de* $(t_1, t_2, ..., t_{n-1})$ *avec* t_1*, etc.*

Faites le calcul de la complexité de cet algorithme.

▸ **Exercice 5.** ***(Tri par insertion)***

Écrire une fonction `int Insere(double t[], int n, int taille, double x);` *qui effectue l'insertion* ***triée*** *de l'élément* `x` *dans le tableau* `t` *de taille* `taille` *et contenant* `n` *éléments (on supposera le tableau déjà trié).*

Écrire une fonction `TriParInsertion` *qui effectue le tri par insertion d'un tableau de* `double`*. La méthode consiste à insérer successivement chacun des éléments* t_i *dans le tableau trié* $(t_0, t_1, ..., t_{i-1})$*. Remarque : aucune copie de tableau n'est nécessaire.*

Évaluer la complexité de ce tri.

▸ **Exercice 6.**

Écrire une fonction `RechercheDicho` *qui prend en argument un tableau de* `double`*, sa taille* `n` *et un élément* `x` *qui effectue* ***itérativement*** *la* ***recherche dichotomique*** *de l'élément dans le tableau (on supposera le tableau déjà trié). Cette fonction renvoie un indice de* `x` *dans le tableau ou* `-1` *s'il n'est pas présent. Évaluer sa complexité.*

Écrire une fonction qui implémente la recherche du minimum et du maximum d'un tableau trié. Que pensez vous de la complexité de cette fonction ?