Travaux Dirigés d'algorithmique n°1

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Adresses, passage par adresse et tableaux

Lors de cette séance, nous allons tout d'abord rappeller la structure d'un programme en langage *C*, puis nous aborderons deux notions essentielles :

- le passage d'une variable par adresse ;
- les tableaux (statiques et dynamiques).

▸ **Exercice 1.** *Que fait le programme suivant :*

```
#include<stdio.h>

int IsPrime(int);

void main(void)
{ int ent,r;

  printf("Entrez un entier: ");
  scanf("%d",&ent);
  r = IsPrime(ent);
  printf("%d\n",r);
}

int IsPrime(int n)
{ int i,borne;

  borne = n/2 + 1;
  i = 2;
  while(i<borne)
  { if ((n%i) == 0) return 0;
    i++;
  }
  return 1;
}
```

▸ **Exercice 2.** *Que fait la fonction suivante :*

```
int Div(float x, float y, float *res)
{
  if ( y == 0 ) return 0;
  *res = x/y;
  return 1;
}
```

▸ **Exercice 3.** *On considère le programme suivant :*

```
#include <stdio.h>

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\n",n,p);
  printf("*ad1 = %d, *ad2 =  %d\n",*ad1,*ad2);
  *ad3 = *ad2;
  printf("*ad2 = %d, *ad3 =  %d\n",*ad2,*ad3);
}
```

Ce programme est syntaxiquement correct mais ne marche pas à l'exécution. Pourquoi ?

▸ **Exercice 4.** *Un nombre est dit parfait s'il est égal à la somme de ses diviseurs propres. Par exemple,* 6 *est parfait* ($6 = 1 + 2 + 3$)*. Ecrire une fonction* `EstParfait` *qui prend en argument un entier positif* `n` *et qui retourne* 1 *si* `n` *est parfait et* 0 *sinon. En déduire une fonction* `Parfait` *qui affiche les nombres parfaits inférieurs à* 1000*.*

▸ **Exercice 5.** *Le but de cet exercice est d'implémenter les opérations de base sur les tableaux.*

- *Ecrire une fonction permettant de saisir un tableau d'entiers. Le prototype de la fonction sera* `int SaisirTab(int t[], int *nb)` *où* `t` *est le tableau contenant les entiers et* `nb` *est l'adresse de la variable contenant le nombre d'éléments du tableau. La fonction retournera* 1 *si la saisie s'est réalisée correctement,* 0 *sinon ;*
- *écrire une fonction* `int * SaisirTab_Bis(int *nb)` *qui prend en argument la taille du tableau en paramètre par adresse, et qui retourne l'adresse du tableau alloué ;*

- *écrire une fonction de prototype* `void AffichageTab(int t[], int nb)` *qui affiche le contenu du tableau* `t` *de taille* `nb` *;*
- *écrire une fonction* `Moyenne` *qui retourne la moyenne des valeurs contenues dans un tableau* `t` *de taille* `nb` *;*
- *écrire une fonction* `Swap` *qui échange les valeurs de deux entiers quelconques ;*
- *en déduire une fonction* `Partitionnement(int t[], int nb)` *qui réarrange le tableau* `t` *de telle façon que toutes les valeurs inférieures ou égales à la moyenne du tableau se trouvent à gauche du tableau, et toutes les valeurs supérieures à droite. (Cette fonction ne devra pas utiliser de tableau intermédiaire.)*