Passage de paramètres

Shockwave

Passage de paramètres donnée

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

int carre (int a){
  return a*a;
}

int main(){
  int val = 2, car;
  car = carre(val + 1);
  printf("carre=%d \n", car);

  return 0;
}
```

On a défini une fonction qui retourne le carré d'un entier et on y fait appel dans la fonction principale. Avant l'appel, **var** vaut 2 dans main et **car** est quelconque. Quand on appelle la fonction carre, **val** vaut toujours 2 dans main, **car** est toujours quelconque mais on copie la valeur de **var** dans carre de sorte que son paramètre effectif soit 3. Après l'appel, **val** vaut 2 par contre **car** vaut 9. Rien de difficile ici. C'est ce qui s'appelle le passage de paramètre par valeur ou par copie.

Passage de paramètres résultat

Nous travaillons désormais avec une procédure qui a des paramètres donnée-résultat ou résultat. **ATTENTION**, ce qui suit ne s'applique pas pour des tableaux en donnée-résultat ou résultat.

Dans le prototype ainsi que dans le corps d'une procédure, l'identificateur d'un paramètre formel donnée-résultat ou résultat doit obligatoirement être précédé d'un *.

Au niveau de l'appel, la référence à l'objet utilisé comme paramètre effectif est son identificateur précédé du caractère **&**. C'est le passage de paramètre par référence ou par adresse. En effet, & devant une variable fournit l'adresse de cette variable. Ainsi, afin de permettre le stockage de notre résultat, on travaille avec des adresses et non des valeurs.

Tout ceci est une histoire de pointeur. On verra cela plus-tard. Pour les tableaux, faire comme le passage de paramètre par valeur, donc pas de * et de &.

```c
// Un exemple: la procedure echanger

#include <stdlib.h>
#include <stdio.h>

void echanger (int *x, int *y){  // parametres formels en donnee-resultat donc *
  int aux = *x;  // je declare une variable entiere aux initialise a *x
  *x=*y;         // je remplace le valeur de *x par celle de *y
  *y = aux;      // maintenant *y contient la valeur *x
}

int main(){
  int x, y;

  printf("Saisir x et y: \n");  // on appelle l'utilisateur a saisir 2 valeurs
  scanf("%d %d", &x, &y);       // j'explique la syntaxe plus loin

  echanger(&x,&y);              // On effectue l'echange

  printf("Apres echange, x vaut  %d, y vaut %d \n", x, y);  // on affiche le resultat

  return 0;
}
```

Saisie : scanf

```c
// Un exemple pour apprendre la syntaxe

#include <stdio.h>   // bibliotheque qui contient la definition de scanf
#include <stdlib.h>

int main(){

  int n = 3, m;
  double x = 1.5, y;

  printf("Saisir m: \n");
  scanf("%d", &m);        // scanf travaille sur des adresses!

  printf("n + m = %d \n", n+ m);

  printf("Saisir y: \n");
  scanf("%lf", &y);     // on utilise lf pour un double

  printf("x - y = %f \n", x - y);

  return 0;
}
```

char	%c
int	%d
long	%ld
float	%f
double	%lf
tableau de caractères	%s

Petite subtilité, un pointeur est une variable qui contient une adresse, donc dans un scanf, si on utilise un pointeur, on n'écrit pas &. A garder en tête pour plus tard !