Modes de passage des paramètres

Modes de passage des paramètres

Il existe deux modes de passage des paramètres:

1. le mode par copie de valeur
2. le mode par copie d’adresse.

Passage par copie de valeur

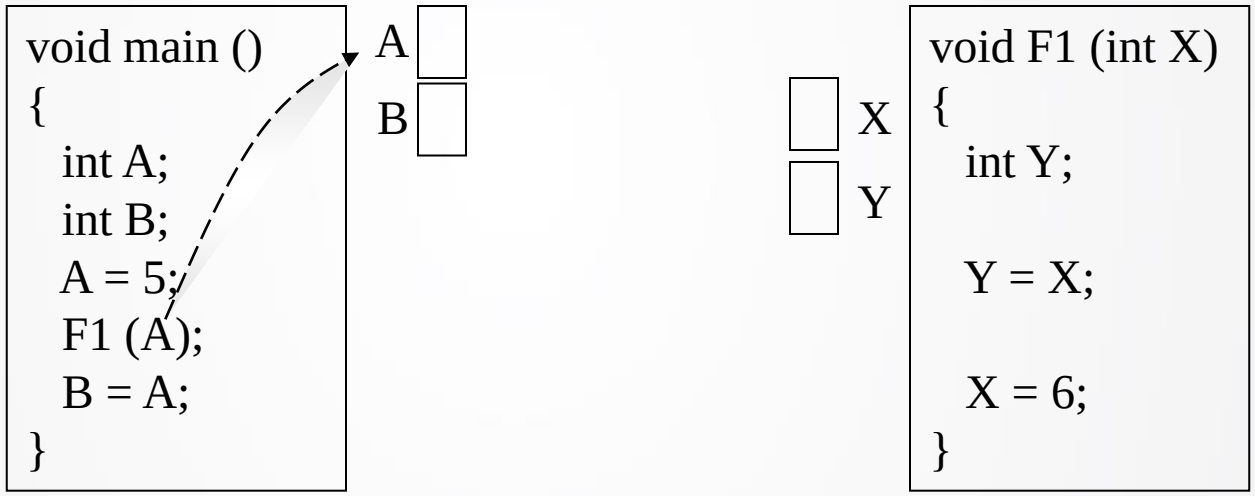
- Au moment de l'appel, la valeur du **paramètre effectif** est *copiée* dans la variable locale désignée par les **paramètres formels** correspondants.
- La fonction travaille sur **des copies** des paramètres **et ne peut pas** les modifier.

Principe :

- la fonction appelante fait une copie de la valeur passée en paramètre,
- passe cette copie à la fonction appelée à l'intérieur d'une variable crée dans l'espace mémoire.
- Cette variable est accessible de manière interne par la fonction à partir de l'argument formel correspondant.

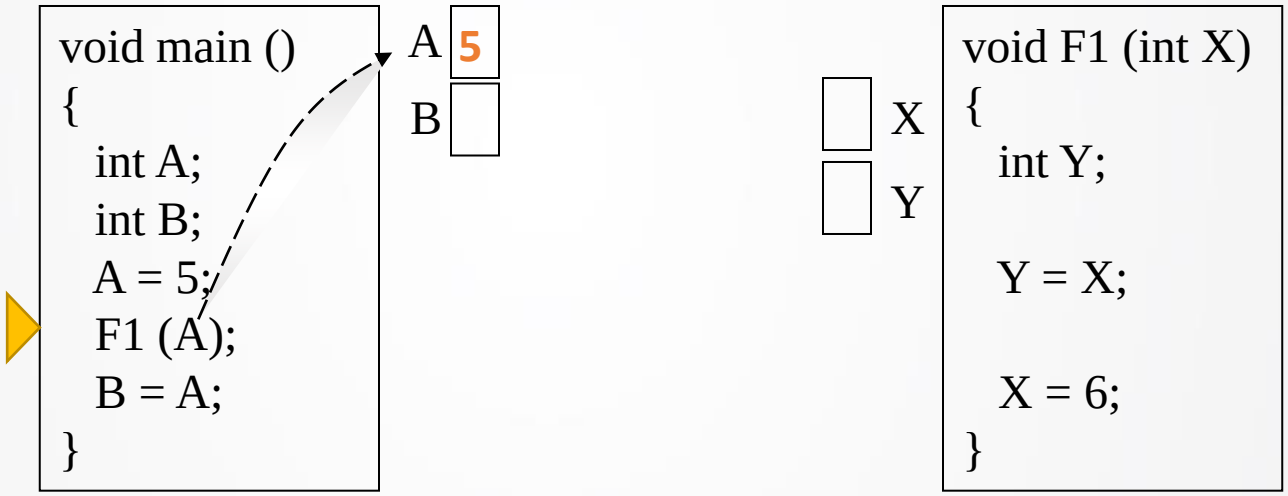
Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



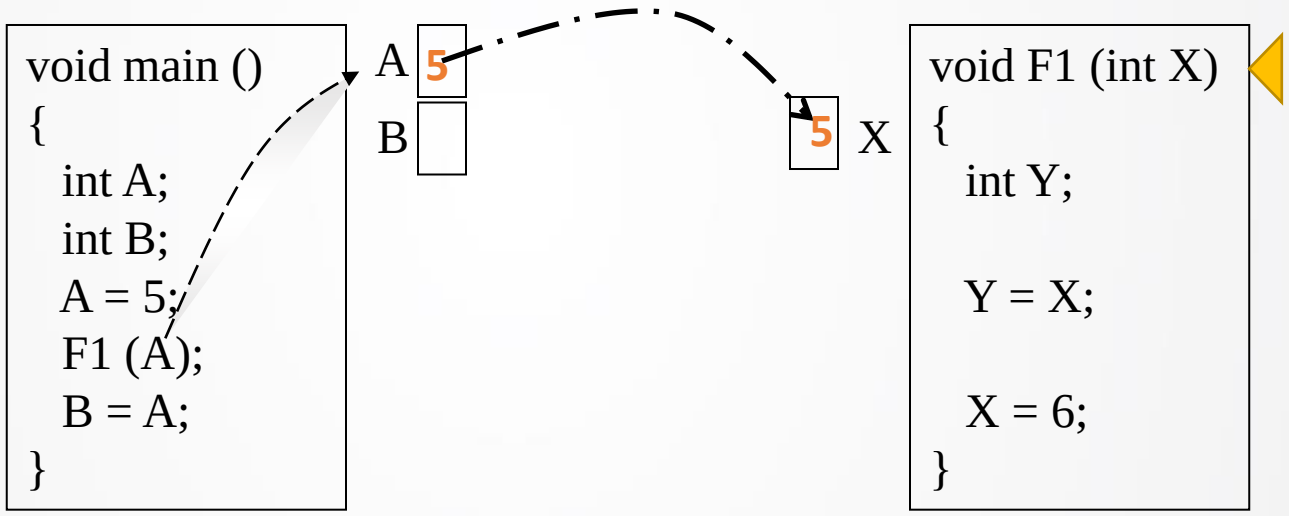
Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



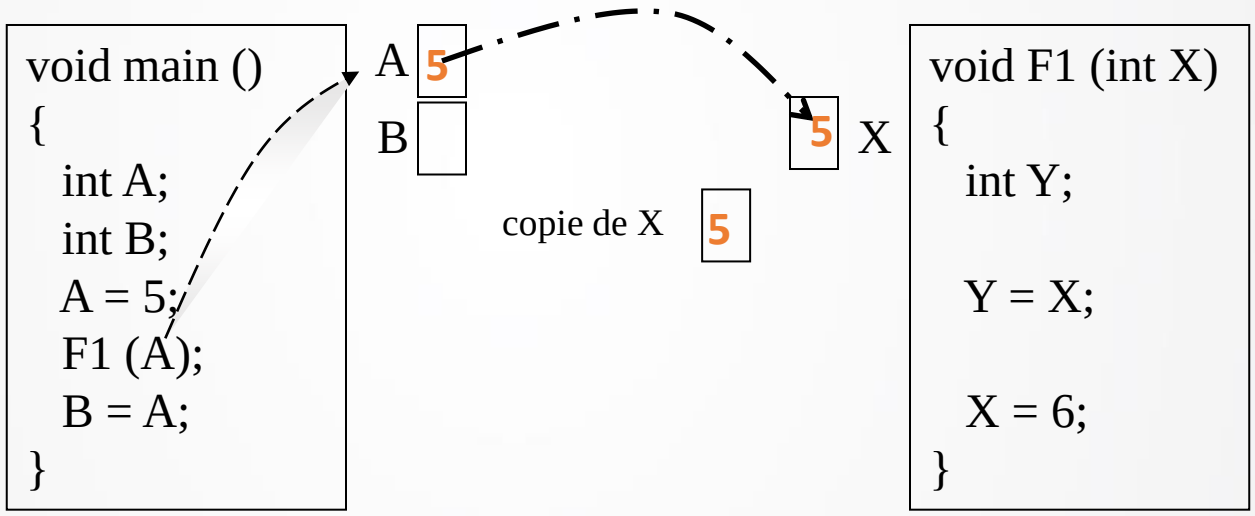
Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



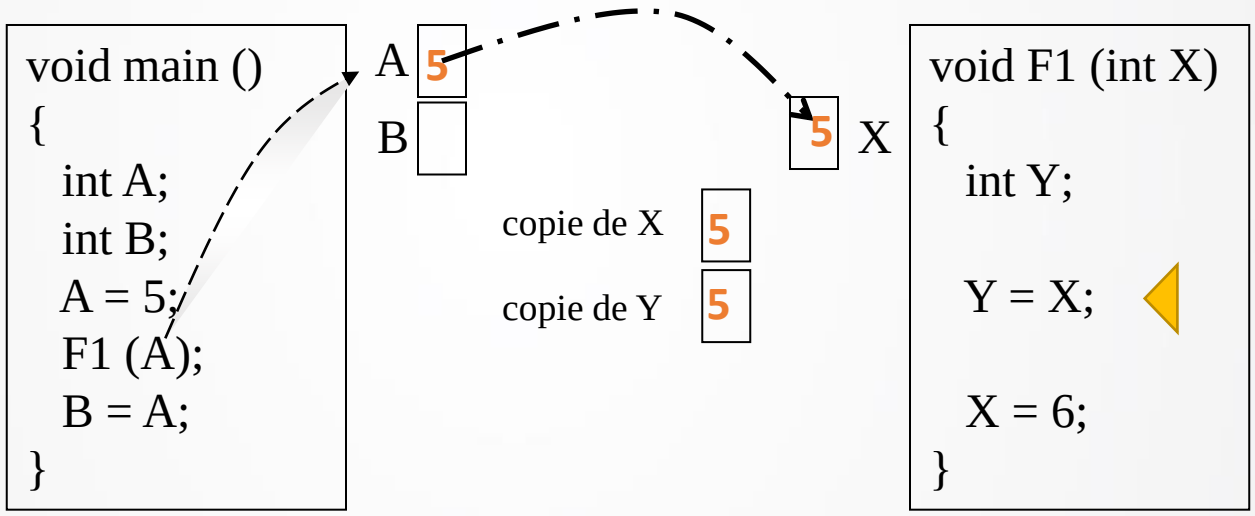
Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



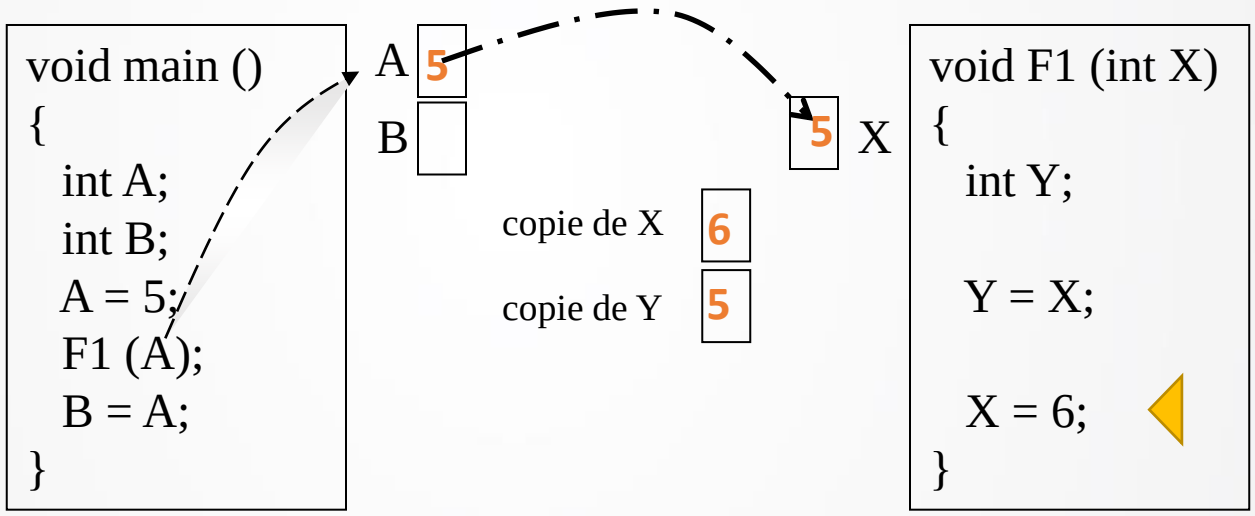
Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



Passage par copie de valeur

Copie des valeurs des paramètres effectifs dans les paramètres formels.



Passage par copie de valeur

```
int max(int v1,int v2);
int main()
{
   int a,b,m;
   printf("introduire a et b\n");
   scanf("%d%d",&a,&b);
   m=max(a,b);
   printf("\ n le maximum est %d",m);
}
int max(int v1,int v2)
{
   if(v1>v2)
            return v1;
    else
            return v2;
}
```

Le valeur de a est copié dans v1
La valeur de b est copié dans v2

L'appel de la fonction prend comme valeur la valeur renvoyée par l'instruction return

Passage par copie de valeur

Que se passe-t-il ?

```
#include <stdio.h>
void mystere(int y){
    y = 3;
}

int main(){
    int y = 7;
    printf("%d \n", y);
    mystere(y);
    printf("%d \n",y) ;
    return 0;
}
```

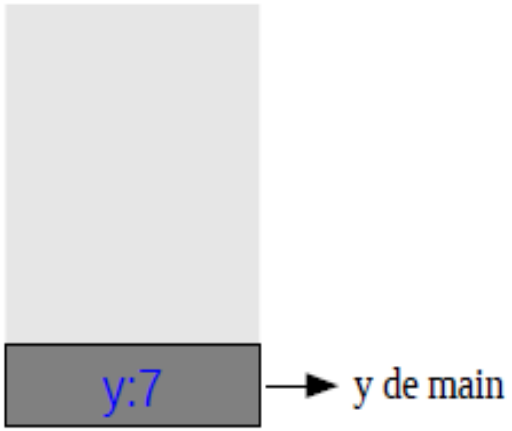
Quelles sont les valeurs affichées ?

Passage par copie de valeur

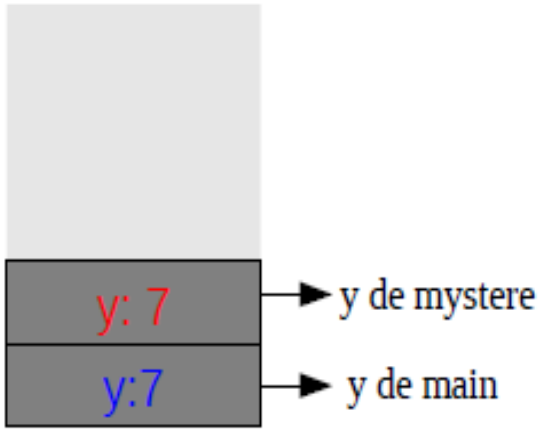
- **Que se passe-t-il ?**

```
#include <stdio.h>
void mystere(int y){
      y = 3;
}
int main(){
      int y = 7;
      printf("%d \n", y);
      mystere(y);
      printf("%d \n",y) ;
      return 0;
}
```

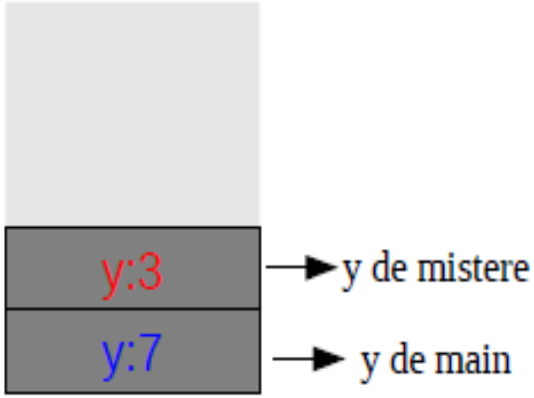
1 - Avant mystere:



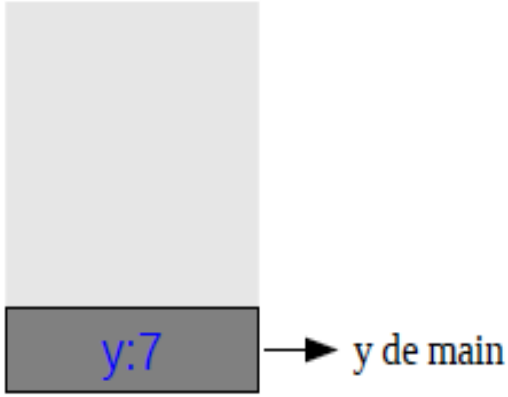
2 – Entrée dans mystere:



3 – avant de sortir de mystere:



4 – Après mystere :



Passage par copie de valeur

- Toutes les modifications effectuées sur le paramètre formel n'affectent que cette valeur locale et **ne sont pas visibles dans la fonction appelante**.
- La fonction ne travaille que sur **une copie** qui va être supprimée à la fin de la fonction.

Passage par adresse

-En C, return ne permet de retourner qu'une seule valeur.

- Parfois une fonction doit retourner plusieurs résultats en sortie.

-Il n'y a qu'une solution pour retourner plusieurs résultats: passer en paramètre l'adresse des variables où seront stockés les résultats.

-Le passage par adresse est aussi utile pour modifier le contenu des variables déclarées dans d'autres fonctions .

Passage par adresse

- Exemple: Une fonction qui retourne le minimum et le maximum de deux entiers a et

```
void min_max(int a, int b, int * pmin, int* pmax)
{
if(a<b) {
            *pmin=a;
            *pmax=b;
       }
  else {
            *pmin=b;
            *pmax=a;
       }
}

int main()
{
    int a,b, min,max;
    printf("Entrer deux entiers");
    scanf("%d  %d", &a,&b);
    min_max(a,b,&min,&max);
    printf("le min est %d le max est %d",min,max);
```

Passage par adresse: Principe

•on appelle passage par adresse quand le paramètre effectif qui est transmis à la fonction lors de l'appel est l'adresse de la variable.

•La fonction ne travaille plus sur une copie de l'objet mais sur l'objet lui même, puisque elle en connait l'adresse.

➔ Le paramètre effectif est alors l'adresse de la variable.

•**Principe :**

•la fonction appelée range *l'adresse transmise* dans un paramètre approprié (de type adresse) qui est une variable locale à la fonction appelée.

•la fonction appelée a maintenant accès, via ce paramètre à la variable de la fonction appelante.

Passage par adresse

- Toute modification sur un paramètre transmis par adresse entraine la modification directe de la variable correspondante.
- Pour le passage par adresse on utilise les pointeurs.