Paramètres et pointeurs

Relation entre paramètres d'une fonction et pointeurs : comment utiliser des valeurs en entrée et en sortie des fonctions.

Remédier à un manque du langage :

A part les tableaux, les paramètres des fonctions sont des variables locales : modifications faites dans la fonction non gardées !

Le paramètre d'une fonction est la copie de l'argument : l'argument n'est pas modifié !

Une fonction ne renvoie qu'une valeur.

Et si la fonction devait 'renvoyer' plusieurs valeurs ? Calcul de plusieurs valeurs que l'on aimerait récupérer !

Paramètres et pointeurs

Une fonction ne renvoie qu'une valeur : règle très stricte, on ne peut pas la changer.

Comment faire ?

Proposition 1 : utiliser un tableau ! On peut y stocker plusieurs valeurs, et les modifications sont gardées !

Mais ces valeurs doivent avoir le même type !

SI la fonction calcule, par exemple, une valeur de type **float** et une de type **char**, on ne peut pas utiliser de tableau (ou alors devient vite compliqué !).

Proposition 2 : utiliser un mécanisme permettant de garder les modifications.

Paramètres et pointeurs

Ce mécanisme est appelé **passage par adresse** ou **passage par pointeur** (c'est la même chose, car un pointeur est une adresse).

Plutôt que de fournir à la fonction une valeur, on va fournir son **adresse** ou un **pointeur sur cette valeur**.

Rappel : une constante n'a pas d'adresse, ce mécanisme n'est utilisable qu'avec des **variables**.

D'ailleurs, peut-on changer la valeur d'une constante numérique ??? Cela n'a pas de sens !

Le mécanisme de transmission argument/paramètre ne change pas, mais c'est ce qui est transmis qui change !

Paramètres et pointeurs

Rappel du mécanisme de recopie argument/paramètre :

prototype : `void entierSuivant(int);`

définition :

```
void entierSuivant(int par)
{
	par = par+1;
}
```

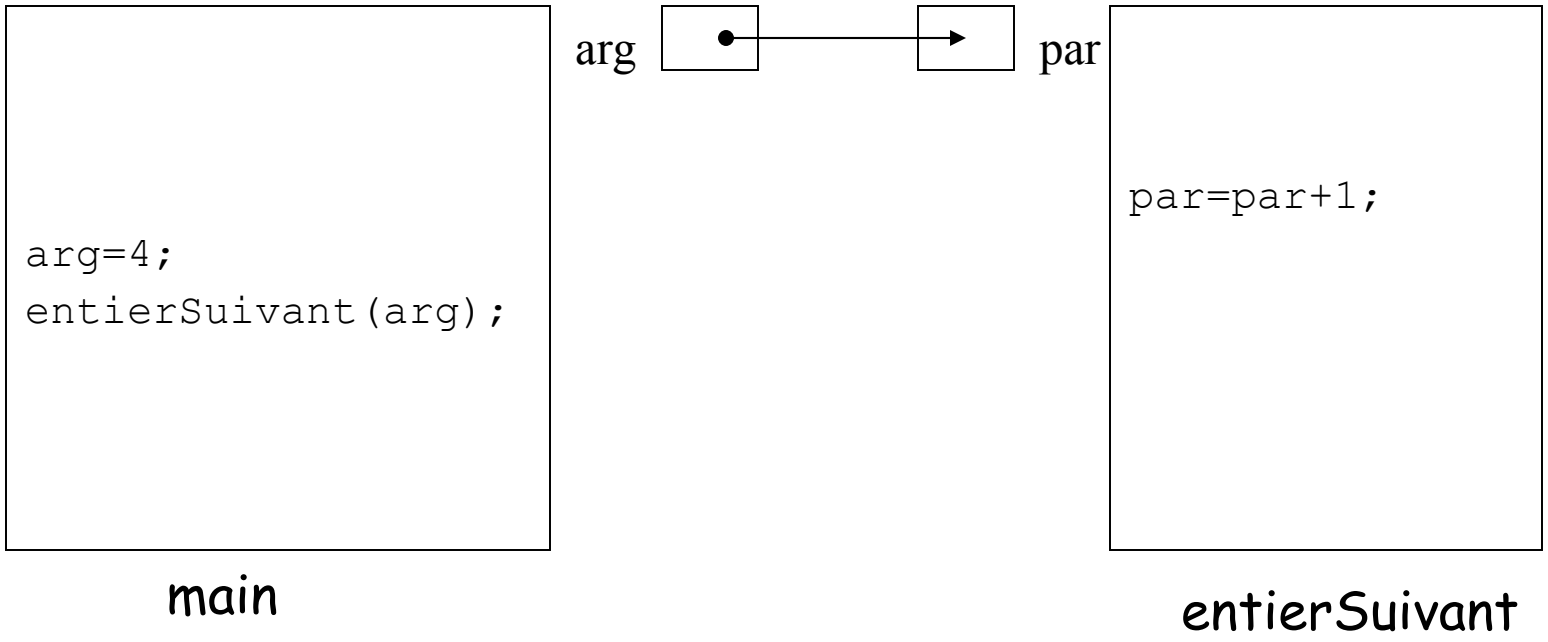
exemple d'utilisation :

```
void main()
{
	int arg;

	arg=4;
	entierSuivant(arg); /* que vaut arg maintenant ? */
}
```

Paramètres et pointeurs

Illustration avec schéma blocs :



Recopie d'adresses

La fonction ne remplit pas son rôle : la valeur de arg n'est pas modifiée (et tant mieux !).

Que se passe-t-il si, au lieu de transmettre une variable entière, on transmet son adresse ?

Au niveau du programme :

le paramètre de la fonction n'est plus une valeur entière, mais un pointeur sur une valeur entière, d'où le prototype de la fonction.

void entierSuivant(int *);

le paramètre est un pointeur : la fonction change, on va modifier le contenu du paramètre, et pas le paramètre :

Recopie d'adresses

```
void entierSuivant(int *par)
{
	*par = (*par)+1;
}
```

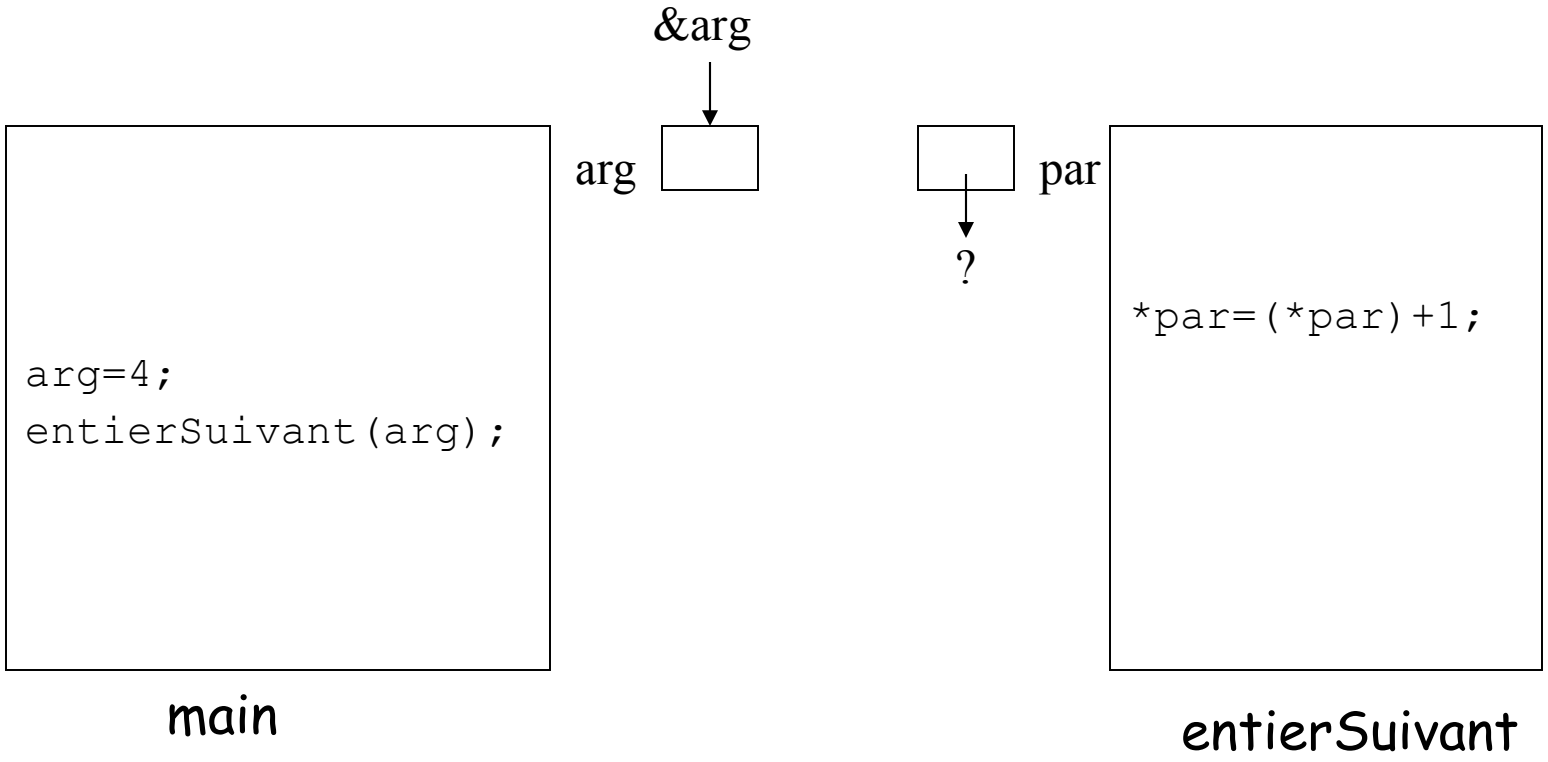
l'appel change : on doit transmettre une adresse, l'adresse de arg :

```
void main()
{
	int arg;

	arg=4;
	entierSuivant(&arg); /* que vaut arg maintenant ? */
}
```

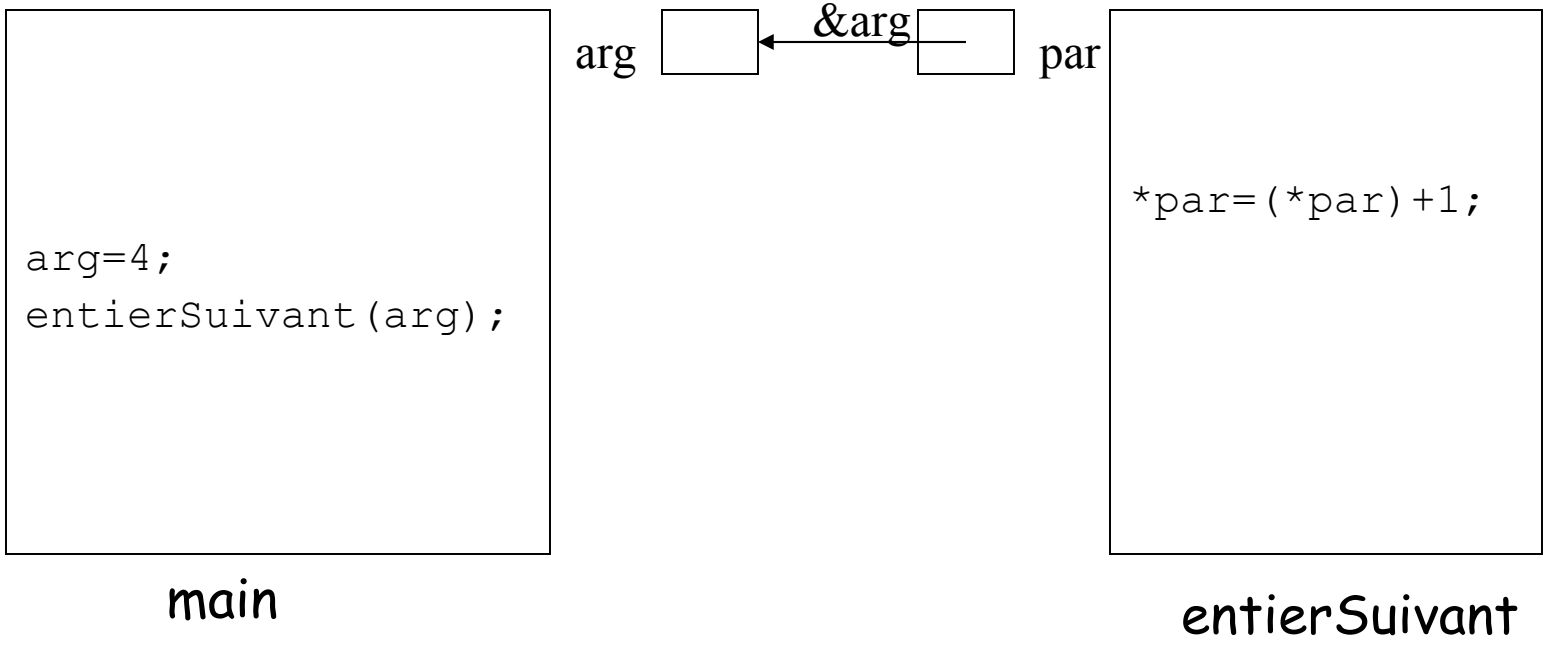
Recopie d'adresses

Reprise du schéma bloc (1/2)



Recopie d'adresses

Reprise du schéma bloc (2/2)



Recopie d'adresses

La recopie a bien lieu : par, pointeur sur entier, reçoit la valeur de &arg, pointeur sur entier, qui pointe sur arg.
on voit sur le schéma bloc que la cellule mémoire qui stocke la valeur de arg est partagée entre la fonction et le programme appelant, parce que l'on a transmis l'adresse.
Lorsque l'on modifie *par, c'est la valeur de arg qui est modifiée !
La variable locale à la fonction est par, le pointeur, et non *par, la valeur pointée.
Ainsi, on respecte les règles pour les passages de paramètre :

- il y a recopie de l'argument dans le paramètre
- le paramètre reste une variable locale de la fonction !

Application

A partir de cet exemple : écrire une fonction qui calcule le minimum et le maximum de 4 valeurs float. On veut récupérer ces valeurs minimum et maximum dans le programme appelant cette fonction.

Retour sur les tableaux

Cette explication permet de voir sous un nouveau jour le comportement des tableaux en tant que paramètres de fonctions.

Remarque déjà faite : lorsque l'on transmet un tableau à une fonction, les changements effectués sur les éléments du tableau sont conservés dans le programme appelant !

Cela n'est plus étonnant : un tableau est en fait un pointeur (constant) sur le premier élément qu'il stocke.

Ainsi, le phénomène de passage par adresse a lieu avec un tableau.

Lorsque l'on utilise un tableau en paramètre de fonction, c'est un **passage par adresse** qui a lieu !