Le passage des arguments (1)

Que peut-on garantir sur la valeur d'une variable passée en argument lors d'un appel de fonction ?

Par exemple :

```
void f(int x) {
  x = x + 1;
  cout << "x=" << x;
}
int main() {
  int val(1);
  f(val);
  cout << " val=" << val << endl;
  return 0;
}
```

Que vaut `val` ? Qu'affiche le programme ?

Le passage des arguments (2)

On distingue 2 types de passages d'arguments :

passage par valeur
: la variable locale associée à un argument passé par valeur correspond à une **copie** de l'argument (*i.e.* un objet distinct mais de même valeur littérale).

  ☞ Les modifications effectuées à l'intérieur de la fonction *ne* sont donc *pas répercutées à l'extérieur de la fonction*.

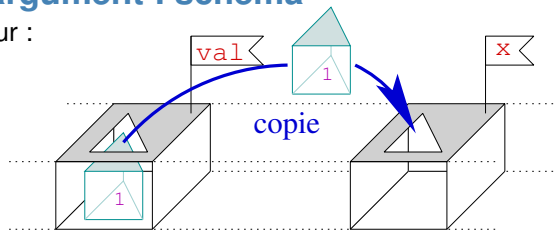
passage par référence
: la variable locale associée à un argument passé par référence correspond à une **référence** sur l'objet associé à l'argument lors de l'appel.

  ☞ Une modification effectuée à l'intérieur de la fonction *se répercute* alors *à l'extérieur* de la fonction.

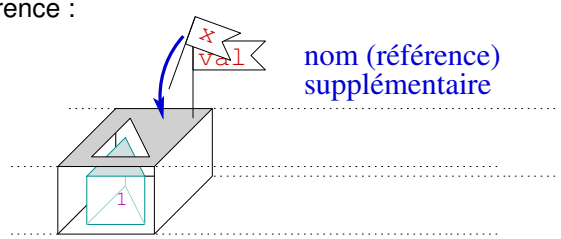
  Le passage par référence doit être explicitement indiqué en utilisant le symbole `&` après le type ; par exemple : `double& x`.

Passages d'argument : schéma

Passage par valeur :



Passage par référence :



Exemple de passage par valeur

```
void f(int x) {
  x = x + 1;
  cout << "x=" << x;
}
int main() {
  int val(1);
  f(val);
  cout << " val=" << val << endl;
  return 0;
}
```

L'exécution de ce programme produit l'affichage :
`x=2 val=1`

Ce qui montre que les modifications effectuées à l'intérieur de la fonction `f()` **ne** se répercutent **pas** sur la variable extérieure `val` associée au paramètre `x` et passée par valeur.

Exemple de passage par référence

```
void f(int& x) {
  x = x + 1;
  cout << "x=" << x;
}
int main() {
  int val(1);
  f(val);
  cout << " val=" << val << endl;
  return 0;
}
```

L'exécution de ce programme produit l'affichage :
`x=2 val=2`

Ce qui montre que les modifications effectuées à l'intérieur de la fonction `f()` **se répercutent** sur la variable extérieure `val` associée au paramètre `x` et passée par référence.

Utilisation du passage par référence

Quand utiliser un passage par référence ?

☞ lorsque l'on souhaite modifier une variable

Par exemple :

- pour saisir une valeur :

```
void saisie_entier(int& a_lire);
...
int i(0);
...
saisie_entier(i);
```

Alternative : retourner la valeur :

```
int saisie_entier();
...
i = saisie_entier();
```

- pour « retourner » plusieurs valeurs :

```
void cartesiennes_vers_polaires(double x, double y,
                                double& angle, double& rayon);
```

Alternative : utiliser les structures (futur cours)

- pour « échanger » des variables :

```
void swap(int& i, int& j);
```