

Passages et retours

par adresse

par valeur

Pointeur (C/C++)

```
void start()
{
  int a = 4;
  Aff(a); → 4
  Pointeur(&a);
  Aff(a); → 5
}
```

void Pointeur(int * a)

```
{
  Aff(*b); → 4
  *b = *b + 1;
  Aff(*b); → 5
}
```

Référence (C++)

```
void start()
{
  int a = 4;
  Aff(a); → 4
  Reference(a);
  Aff(a); → 5
}
```

void Reference(int & b)

```
{
  Aff(b); → 4
  b = b + 1;
  Aff(b); → 5
}
```

Retour Valeur (C/C++)

```
void start()
{
  int a = 4;
  Aff(a); → 4
  a = Retour(a);
  Aff(a); → 5
}
```

int Valeur(int b)

```
{
  Aff(b); → 4
  b = b + 1;
  Aff(b); → 5
  return b;
}
```

Valeur (C/C++)

```
void start()
{
  int a = 4;
  Aff(a); → 4
  Valeur(a);
  Aff(a); → 4
}
```

void Valeur(int b)

```
{
  Aff(b); → 4
  b = b + 1;
  Aff(b); → 5
}
```

Variable d'origine modifiée

Notes : utiliser une référence ou un pointeur est équivalent. L'utilisation d'une référence requiert moins de symboles. A l'appel, un passage par référence ne se distingue pas d'un passage par valeur !!! Le retour par valeur ne doit être utilisé que pour des types courts : int, float.

	Définition	Déclaration
Variable Avec Init	Type Id; Type Id = Id2;	extern Type Id;
Fonction	Type Id(Type Id1, Type Id2) { ... }	Type Id(Type Id1, Type Id2);
Classe	Class Id { public : ... };	Class Id;

Note : la déclaration d'une classe ne permet pas d'utiliser ses fonctions et ses paramètres

Passage d'objets en paramètres d'une fonction :

Il faut **éviter** de passer ou de retourner des objets par valeur car cela nécessite la création d'objets intermédiaires. **Utilisez** uniquement des pointeurs ou des références. Si vous voulez garantir l'absence de modification des objets à l'intérieur de la fonction appelée, utilisez le qualificatif **const** : void fnt(const P3 &P); ou void fnt(const P3 *P);


Lilian BUZER - ESIEE

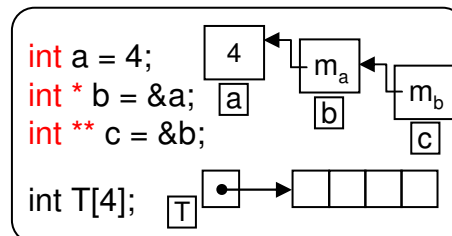
Dept. Informatique

Fiche rappel langage C/C++

Tableaux

Statique	Dynamique
Type ID[cte];	Type ID[var];
Type ID[cte][cte];	Type ID[var][var];
int ID[] = { 4, 5 };	Type * ID = new Type[var]; delete [] ID;

int T[40][50] n'est pas équivalent à un int ** car il y a allocation d'un tableau 1D de 40*50 cases. 
T[i][j] correspond à T[i+j*50], le type est int T[][50]



Symboles & * . ->

Instructions :

&(var) → adresse de la variable
*(adr) → valeur présente à cette adr
obj.a → accès à la variable de l'objet
obj.fnt() → appel de la fnt de l'objet
pt->a → accès à la var de l'objet pointé
pt->fnt() → accès à la fnt de l'objet pointé

Déclarations :

int * p; pointeur sur un entier
int & a = b; référence sur un entier
T a; a est un objet de type T