Notion de portée locale (1)

Les notions de *portée* ***locale*** v.s. ***globale*** définies pour les blocs sont également valides dans le cadre des fonctions[5].

Comme dans le cas des blocs, la *portée* permet de résoudre les problèmes de conflits de référençage entre entités ***externes*** (i.e. définies à l'extérieur du corps de la fonction) et entités ***internes*** (i.e. définies dans le corps-même de la fonction).

Exemple:

```
int i(2);
void f(int a);
void main {
   f(i);
}
void f(int a) {
   int i(10);
   cout << a*i << endl;
}
```

Quelle variable est référencée par le `i` de l'instruction `cout << a*i << endl` ?

5. Cela est bien naturel, puisque le corps d'une fonction n'est autre qu'un bloc.

Notion de portée locale (2)

Les règles utilisées en C++ sont:

- En cas de conflit de référence entre une entité interne et une entité externe, **l'entité interne est systématiquement choisie**; on dit que les **références internes sont de portée locale**, i.e. limitées aux entités internes de la fonction. On dit également que les entités internes ***masquent*** les entités externes.
- Les **arguments formels** d'une fonction constituent des **entités internes** à sa définition.
- Les références (non ambiguës) à des entités externes sont possibles[6]. Il est néanmoins préférables de les éviter, en **explicitant systématiquement** toute référence à une entité externe **par le biais d'un argument formel**.

6. Une fonction modifiant des variables externes sera dite «*à effet de bord*», ce qui est fortement déconseillé.

Notion de portée locale (3)

Schématiquement, ces règles peuvent se représenter par: