Plusieurs variables

Précédemment, nous n'avons utilisé qu'une seule variable, mais les programmes en utilisent généralement plusieurs. Le programme suivant illustre cela : il utilise deux variables nommées *largeur* et *longueur* afin de calculer l'aire en mm² d'une feuille A4 (21 cm × 29,7 cm), et il enregistre le résultat dans une variable nommée *surface*. Le contenu de cette dernière variable est ensuite affiché.

```
int largeur = 210;
int longueur = 297;
int surface = longueur * largeur;

cout << surface << endl;
```

↳

```
62370
```

À la fin du programme, nous avons donc les boîtes suivantes :

largeur	longueur	surface
210	297	62 370

Variable inexistante

Si on fait appel à une variable qui n'existe pas (ou pas encore), on obtient une erreur. Par exemple, le programme suivant définit une variable *longueur*, et tente ensuite d'afficher le contenu d'une variable nommée *largeur* qui n'a jamais été définie.

```
int longueur = 297;
cout << largeur << endl;
```

↳

```
error: 'largeur' was not declared in this scope
```

Il faut faire particulièrement attention au fait que les minuscules et majuscules ne sont pas considérées comme équivalentes. Ainsi, la variable nommée *longueurFil* n'a strictement rien à voir avec la variable nommée *longueurfil*.

```
int longueurFil = 10;
cout << longueurfil << endl;
```

↳

```
error: 'longueurfil' was not declared in this scope
```

Type redondé

Si l'on recopie le type de la variable quand on lui affecte une nouvelle valeur, on obtient une erreur :

```
int force = 4;
int force = 5;
```

↳

```
line 4:8: error: redeclaration of 'int force'
line 3:8: note: 'int force' previously declared here
```

nous disant qu'on a tenté de créer une variable avec un nom qui était déjà pris. Il ne faut indiquer le type qu'une seule fois :

```
int force = 4;
force = 5;
```

Déclaration d'une variable dans un bloc

Si l'on crée une variable dans un bloc, elle n'existe plus quand on quitte le bloc. Par exemple :

```
{
   int force = 6;
}
cout << force << endl;
```

↳
```
error: 'force' was not declared in this scope
```

Après l'accolade, la variable *force* définie dans le bloc n'existe plus. On obtient donc une erreur si on essaie d'y avoir recours.

Remarquez en passant qu'il est possible d'écrire un bloc, sans qu'il n'y ait rien de spécial comme une boucle. Le bloc sert alors à limiter la *portée* de ce qui est défini à l'intérieur. Nous vous reparlerons dans cette notion dans un prochain chapitre.

Si on rencontre une erreur de la forme `error: 'xxxxx' was not declared in this scope`, on pensera donc à bien vérifier que quand on utilise la variable, elle est définie, au bon endroit et avec strictement le même nom.