Avant-propos : la qualité logicielle

Vous avez appris à réaliser des calculs et à afficher le résultat avec `cout`. C'est un bon début, puisque par définition, c'est le rôle d'un ordinateur de réaliser automatiquement des calculs et opérations logiques. Cependant, on voit vite une limite. Imaginez que vous devez réutiliser le résultat d'un premier calcul dans un second calcul, comment faire ?

Actuellement, la seule solution que vous connaissez est de réécrire le premier calcul dans le second calcul. Par exemple, pour additionner 123 et 456, puis multipliez le résultat par 789, vous devez écrire :

```
cout << 123 + 456 << endl;
cout << (123 + 456) * 789 << endl;
```

Sur un calcul aussi simple, cela ne semble pas poser de problème particulier. Le problème pourrait se présenter pour des calculs plus complexes. Mais en fait, même sur des calculs simples, cette approche est limitante. Imaginons par exemple que vous souhaitez utiliser le résultat du premier calcul dans une série d'autres calculs. Par exemple :

```
cout << 123 + 456 << endl;
cout << (123 + 456) * 789 << endl;
cout << (123 + 456) * 428 << endl;
cout << (123 + 456) * 384 << endl;
cout << (123 + 456) * 126 << endl;
cout << (123 + 456) * 842 << endl;
cout << (123 + 456) * 962 << endl;
cout << (123 + 456) * 107 << endl;
cout << (123 + 456) * 640 << endl;
cout << (123 + 456) * 862 << endl;
cout << (123 + 456) * 364 << endl;
```

Cela commence à être pénible de devoir réécrire le premier calcul à chaque ligne. Imaginez maintenant que vous ne souhaitez plus calculer la somme de 123 et 456, mais la somme de 321 et 654. Que se passe-t-il ?

Il faut modifier chaque ligne, cela prend du temps et vous risquez de faire une erreur en recopiant le premier calcul.

On comprend vite que cette méthode, bien que correcte au niveau syntaxe, pose des problèmes. L'écriture d'un code C++ qui compile n'est pas suffisant pour créer un programme correct. Il faut également écrire un code qui respecte un certain nombre de règles de qualité pour les logiciels (*Software quality*).

Il existe plusieurs définitions des règles de qualité logicielle, nous allons voir rapidement celle de la norme ISO/IEC 9126. Un logiciel de qualité doit être :

- **fonctionnel**, c'est-à-dire qu'il doit répondre aux besoins des utilisateurs ;
- **fiable**, c'est-à-dire qu'il doit donner les résultats attendus ;
- **conviviale**, c'est-à-dire qu'il doit être facile d'utilisation ;
- **efficace**, c'est-à-dire qu'il doit être le plus performant possible en utilisant le minimum de ressources possible ;
- **maintenable**, c'est-à-dire qu'on doit pouvoir facilement le corriger ou lui ajouter des fonctionnalités ;
- **portable**, c'est-à-dire qu'on doit pouvoir l'utiliser facilement dans différents environnements.

Ces critères sont importants parce qu'ils permettent de distinguer un "bon" code d'un "mauvais" code. Le critère "ça compile" n'est pas suffisant.

Dans un vrai programme, il ne sera bien sûr pas toujours possible de respecter tous ces critères, il faudra donc faire des choix sur les critères que vous allez respecter et ceux qui seront moins importants dans un contexte de travail donné. Mais le point principal est qu'il faudra toujours que la décision de ne pas respecter des critères de qualité soit une décision consciente et réfléchie, c'est-à-dire que vous aurez parfaitement identifié en quoi votre code ne respecte par ces critères et quel impact cela aura sur la qualité de votre logiciel.

Si on revient au code d'exemple précédent, on comprend alors que ce

code ne respecte pas l'un des critères de qualité logiciel : la maintenabilité. En effet, si on souhaite modifier le premier calcul, il est nécessaire de modifier plusieurs lignes de code (et dans un programme complexe, cela peut se traduire par la modification de plusieurs milliers de ligne de code). Dans un code correctement maintenable, il ne faudrait changer idéalement qu'une seule ligne, celle contenant le premier calcul.

DRY Don’t Repeat Yourself

On utilise souvent l'acronyme DRY (ne pas se répéter) pour résumer cette problématique. Même s'il est assez simple de copie-coller des lignes de code plusieurs fois, ce peut devenir compliqué - et source potentielle d'erreurs - lorsque vous aurez besoin de modifier les lignes de code que vous avez copié. Il sera donc préférable de factoriser votre code au maximum.

Chapitre précédent	Sommaire principal	Chapitre suivant

Cours, C++