Développer à partir des tests

Michel Billaud (michel.billaud@u-bordeaux.fr, michel.billaud@laposte.net)

8 juin 2020

Table des matières

1 Un exercice

Imaginons qu'on vous donne comme exercice de programmation en C

> écrire une fonction qui calcule le nombre de fois où un caractère apparaît dans une chaîne.

2 Tests, corrections, et régressions.

La réalité du développement, c'est que vous allez écrire du code qui ne va probablement pas marcher du premier coup.

Vous allez compiler le code, essayer quelques exemples. Il y en aura un qui ne marche pas. Vous allez chercher à corriger votre code. Puis réessayer. Il y aura d'autres erreurs. Rapidement, vous allez vite en avoir assez, et vous contenter d'un ou deux essais pour décider que ça marche.

Mais avec toutes ces modifications, il se peut qu'un test qui passait au début ne marche plus (c'est ce qu'on appelle une **régression**). Et que vous ne vous en rendiez pas compte parce que vous n'essayez même plus.

3 Automatiser les tests

Autiomatiser, c'est une solution intéressante. L'idée :

1. vous écrivez les tests sous forme de code **dès le début**,
2. ensuite, vous les faites exécuter **systématiquement** chaque fois que vous modifiez votre programme.

Exemple, la chaine de caractères `"abc"` est une bonne base pour vérifier que ça fonctionne avec un caractère en premiere position (`'a'`), en dernière position (`'c'`), ailleurs (`'b'`) ou pas du tout (`'x'`). Ca vous fait 4 tests intéressants, qu'il faudrait faire tourner à chaque fois.

4 Le lancement des tests

Dans le `main` vous pourriez commencer par

```
int main() {
    tester("abc", 'a', 1);
    tester("abc", 'b', 1);
    tester("abc", 'c', 1);
    tester("abc", 'x', 0);
    ...
```

où la fonction `tester` est chargée de vérifier que dans une chaîne, un certain caractère apparaît un certain nombre de fois.

5 La fonction `tester()`

est facile à écrire. Elle affiche la nature du test, et un message bien visible si il y a une erreur :

```
void tester(char chaine[], char lettre, int resultat_attendu)
{
   printf("- dans %s il y a %d fois la lettre %c\n",
          chaine, resultat_attendu, lettre);
   int resultat = nombre_d_occurrences(chaine, lettre);
   if (resultat != resultat_attendu) {
      printf("PERDU on a trouvé %d\n, resultat);
   }
}
```

6 Et voila.

Les tests sont en place : il ne vous reste plus qu'à compléter la fonction

```
int nombre_d_occurrences(char chaine[], char lettre)
{
  ...
}
```

7 Précautions

- prévoir un nombre suffisamment de tests pour être sûr d'avoir couvert les situations "limite", par exemple les bouts de tableau (voir plus haut).
- si un test échoue, n'oubliez pas qu'on peut *aussi* se tromper en rédigeant les tests (un grand classique !).