TD : tests

Laura Montagnier et Ianis Kelfoun

Février 2024

Q1/ L'objectif de la fonction **assert_float_equals** est de comparer deux flottants. Or, la manière dont les flottants sont représentés et manipulés en informatique peut induire des erreurs et des arrondis.

Tenter de comparer directement des valeurs flottantes peut ainsi mener à des erreurs, puisqu'en pratique, deux valeurs a priori égales peuvent différer très légèrement.

Préciser à la fonction un ε permet de tolérer une faible différence lors de la comparaison entre deux nombres flottants, et d'ainsi se prémunir contre des erreurs liées à la représentation.

```
static void test1 (void **state)
{
    (void) state;
    float prix;
    prix= computePrice(20.0,0);
    assert_float_equal (1.5,prix,0.001);
}
```

Q2/ Lorsque l'on utilise la fonction **assert_true** pour vérifier une égalité, on compare directement les deux valeurs flottantes sans prendre en compte une marge d'erreur sur les valeurs comme expliqué en question 1.

Ainsi, il est plus prudent d'utiliser le type d'assertion **assert_float_equals** pour éviter des erreurs liées non pas aux valeurs théoriques, mais à la manière dont elles sont implémentées et prendre en compte la possibilité qu'elles diffèrent faiblement.

Il vaut donc mieux utiliser **assert_float_equals** que **assert_true**.

Q4/ Si on effectuait un seul test avec plusieurs assertions, le fait qu'au moins une seule des assertions soit vraie ou fausse modifierait la validité du test.

Mais, dans ce cas, rien n'indiquerait quelle assertion est à l'origine de l'échec ou de la réussite, ce qui peut ne pas être pratique.

On privilégie donc l'usage d'un test pour chaque cas. Ainsi, on peut exactement déterminer ce qu'il se passe et savoir exactement quelles assertions sont vraies, et quelles assertions sont fausses, plus facilement.

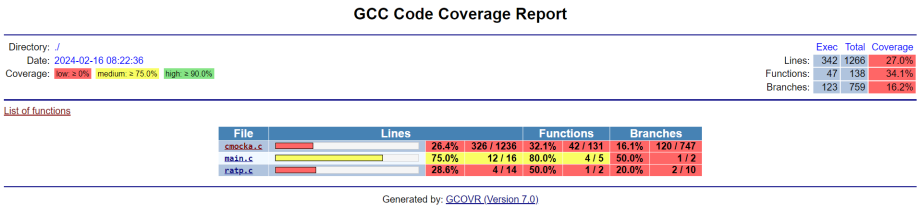
```c
static void Test_adulte(void **state){
    assert_float_equal(1.2, computePrice(13, 0), 0);
}

static void Test_adulteTouriste(void **state){
    assert_float_equal(2.4, computePrice(13, 1), 0);
}

static void Test_enfant(void **state){
    assert_float_equal(0.6, computePrice(11, 0), 0);
}

static void Test_enfantTouriste(void **state){
    assert_float_equal(1.2, computePrice(11, 1), 0);
}
```

Q5/ On a le résultat suivant :



GCC Code Coverage Report

Directory: ./
Date: 2024-02-16 08:22:36
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	342	1266	27.0%
Functions:	47	138	34.1%
Branches:	123	759	16.2%

List of functions

File	Lines		Functions		Branches	
cmocka.c	26.4%	326 / 1236	32.1%	42 / 131	16.1%	120 / 747
main.c	75.0%	12 / 16	80.0%	4 / 5	50.0%	1 / 2
ratp.c	28.6%	4 / 14	50.0%	1 / 2	20.0%	2 / 10

Generated by: GCOVR (Version 7.0)

Q6/ On doit ajouter :

```c
void recupAge(int * age)
{
    *age = (int) mock();
}
```

```c
static void testMock(void **state) {
    will_return(recupAge, 12);
    assert_float_equal(1.2, newComputePrice(72), 0);
}
```