PROGRAMACIÓN AVANZADA

Pruebas unitarias

PRUEBAS UNITARIAS

Una prueba unitaria es solo un código que llama a otro código para determinar si se comporta como se espera.

¿POR QUÉ NO LAS USAMOS CON FRECUENCIA?

Por lo general, los programadores no estamos dispuestos a invertir tiempo escribiendo pruebas unitarias.

Sobre todo, porque muchas veces una sola prueba puede ser difícil de hacer.

Además, realizar el programa en su totalidad es de por sí engorroso, y no planificamos continuar diseñando pruebas unitarias una vez que ha sido terminado.

¿CÓMO SE VE UNA PRUEBA UNITARIA?

Ejemplo:

```
void this_is_a_unit_test(void)
{
	int next = get_next_fibonacci(5);
	ASSERT_EQUAL(next,8);
}
```

Se llama a la función con una entrada de 5 y se espera obtener un 8.

ASSERT EQUAL, ¿QUÉ ES?

Es una macro que permite comparar dos valores enteros.

¿Siempre debemos definir macros…?

- No, hay plataformas que permiten que las pruebas unitarias se ejecuten por sí solas, y también permiten la definición y ejecución de macros.

¿Cuáles son estas plataformas?

PLATAFORMAS DE PRUEBAS UNITARIAS PARA C

Unity

CppUTest

GoogleTest.

REALIZAR Y EJECUTAR PRUEBAS UNITARIAS

Nuestro programa puede imprimir en pantalla el resultado de una o varias pruebas.

Incluso, puede escribirse un programa con el único propósito de ejecutar las pruebas.

Ejemplo:

```
void this_is_a_unit_test(void) {
      int next = get_next_fibonacci(5);
      if (next == 8)
            printf("Fibonacci OK");
      printf("Error en Fibonacci");}
```

```
int main(){
      …
      this_is_a_unit_test();
      …
}
```

PRUEBAS CORRECTAS E INCORRECTAS

Ejemplo de función:

```
int suma(int a, int b) {
      return a + b + 1; }
```

Ejemplo de prueba incorrecta:

```
int prueba_suma_0_0() {
      return suma(0, 0) == 1; }
```

Ejemplo de prueba correcta:

```
int prueba_suma_0_0() {
      return suma(0, 0) == 0; }
```

¿Por qué?

- ¡Podemos decir que lo que está incorrecto es la función!
- Las pruebas deben ser lo más descriptivas posibles para evitar confusiones.
- Incluso se pueden diseñar e implementar primero que el código.

ORIENTACIÓN

Incluir pruebas unitarias a los códigos de la Tarea 3, aunque estas pruebas unitarias no serán evaluadas de momento.

¡Es importante que nuestro código incluya siempre pruebas unitarias y que esté correctamente estructurado en módulos y funciones!

MUCHAS GRACIAS