

Clase #04 de 27

Pruebas Unitarias con Assert & Introducción a Funciones

Abr 30, Miércoles

Agenda para esta clase

- Pruebas Unitarias con Assert
- Trabajo #1: Valores y Operaciones de Tipos de Datos — Ejemplos
- Funciones I: Introducción

Pruebas Unitarias con Assert

```
#include<cassert>
```

Assert – Introducción (I)

- Ejemplo
 - `assert(2 == 1+1);`
- ¿Qué necesitamos para usarla?
 - `#include <cassert>`
 - Define `assert` como una macro con parámetros, similar a una función
- ¿Para qué sirve?
 - Para realizar comprobaciones internas, verificar y confirmar nuestro trabajo
 - Para aplicar TDD (Test Driven Development) y realizar pruebas automáticas
 - Durante el desarrollo, para validar axiomas (i.e., situaciones que siempre van a ser verdad) y que si no lo son significa que el desarrollo no está completo o correcto
 - Para documentar situaciones lógicamente imposibles y que si ocurren, hay algo fundamentalmente mal en el programa.
- ¿Cuál es su sintaxis?
 - `assert(expresión);`
 - Expresión es generalmente una afirmación, es decir una expresión booleana que vale true
 - La afirmación toma forma de ecuaciones e inecuaciones
 - `1 > 0`
 - `1 < 2`
 - `2+1 = 3`
 - `5 = 2+3`
 - `5 ≠ 1+2`
 - `1 ≠ 3+1`
 - También de expresiones booleanas como
 - `edad > 18 and nacionalidad = argentina`
 - `not EsPrimo(n)`
- ¿Qué hace?
 - En tiempo de ejecución, `assert` evalúa la expresión, si es verdadera no hace nada. Pero si es falso, aborta el programa con un mensaje de error transcribiendo la expresión que dio falsa y su lugar del programa.

Assert – Introducción (II)

- ¿Para qué no se usa?
 - Validar datos recibidos de usuario, archivos u otros sistemas. Por ejemplo:
 - ¿El formato de fecha recibido es el esperado?
 - Para evaluar condiciones que es natural que puedan ser verdaderas o falsas. Por ejemplo:
 - ¿Es un cliente con descuento?
 - Para manejo de errores o excepciones. Al abortar el programa, no permiten recuperación, su mensaje de error es críptico, y pueden estar deshabilitadas en producción
 - ¿Se cortó la conexión?
- ¿Qué hacer cuando todas las pruebas dan bien y el desarrollo finaliza?
 - Las comprobaciones de `assert` demandan tiempos de ejecución
 - Si las pruebas finalizaron correctamente, es ineficiente que nuestro programa valide axiomas en cada ejecución en un ambiente productivo.
 - Necesitamos una forma de deshabilitar la ejecución de `asserts`.
 - Comentar o borrar los `asserts` puede introducir nuevos errores.
 - Para eso está el flag `NDEBUG` (*no debug*, no depurar), que si está presente antes de incluir `cassert` omite los `asserts`.
 - Ese flag se puede establecer en el propio código, pero es más intrusivo y flexible establecerlo o no al iniciar el proceso de creación del ejecutable con la opción `-D`.

30.4.2 Assertions

The standard provides:

Assertions (§iso.7)	
static_assert(e,s)	Evaluate e at compile time; give s as a compiler error message if !e
assert(e)	If the macro NDEBUG is not defined, evaluate e at run time and if !e , write a message to cerr and abort() ; if NDEBUG is defined, do nothing

For example:

```
template<typename T>
void draw_all(vector<T*>& v)
{
    static_assert(is_base_of<Shape,T>(), "non-Shape type for draw_all()");

    for (auto p : v) {
        assert(p!=nullptr);
        // ...
    }
}
```

The **assert()** is a macro found in **<cassert>**. The error message produced by **assert()** is implementation-defined but should contain the source file name (**__FILE__**), and the source line number (**__LINE__**) containing the **assert()**.

Asserts are (as they should be) used more frequently in production code than in small illustrative textbook examples.

<https://en.cppreference.com/w/cpp/error/assert>

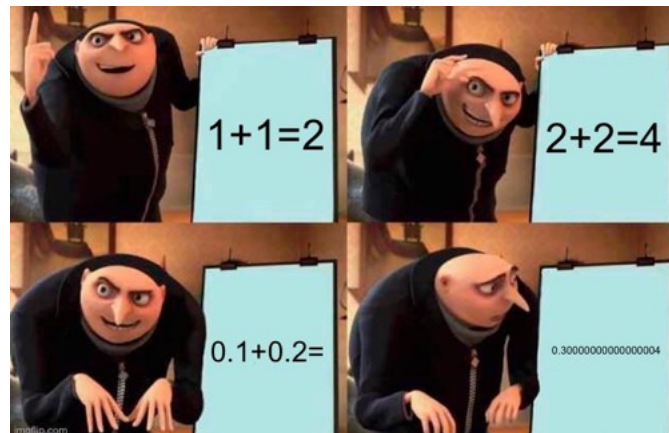
https://en.wikipedia.org/wiki/Assertion_%28software_development%29

Trabajo #1

Valores y Operaciones de Tipos de Datos — Ejemplos

Pruebas Ejemplo

Tipo	Ejemplos
bool	<code>assert(true and false == false);</code>
unsigned	<code>assert(2u * 3u == 6u);</code>
int	<code>assert(7 / 2 == 3);</code>
char	<code>assert('A' + 1 == 'B');</code>
string	<code>assert("ABC".length() == 3);</code>
double	<code>assert(0.1 + 0.2 == 0.30000000000000004);</code>



Valores y Operaciones de Tipos de Datos — Ejemplos

6.1. Objetivos

- Demostrar la aplicación de tipos de datos mediante un programa ejemplo con pruebas.

6.2. Temas

- Valores.
- Operaciones.
- Tipos de datos.
- Representación literal de valores en programas C++.
- `assert`.

6.3. Problema

Diseñar un programa C++ que ejemplifique con pruebas la aplicación de los tipos de datos vistos en clases.

6.4. Restricciones

- Utilice la notación C++ para representar
- No utilice *variables*.
- No extraer valores de `cin`, usar valores literales (constantes).

- No enviar valores a `cout`.

6.5. Tareas

- Este es un *trabajo no estructurado*, que consiste en escribir un programa que ejemplifique el uso de los tipos de datos básicos de C++ vistos en clase: `bool`, `char`, `unsigned`, `int`, `double`, y `string`.



Crédito Extra

¿Son los enumerados en la sección anterior realmente todos los tipos que usamos en clase? Justifique.

Ejemplo 6.1. Crédito Extra

Para cada tipo de dato, agregue una notación literal alternativa, si la tiene.

Ejemplo 6.2. Crédito Extra

Intente probar que la suma de diez veces un décimo (0.1) es uno (1.0).
¿Qué está ocurriendo?

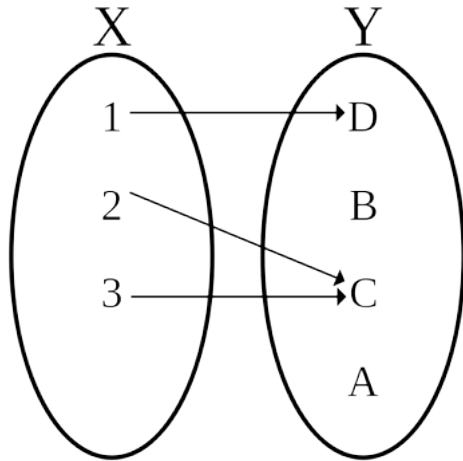
6.6. Productos

```
DD-EjemploTipos
|-- readme.md
`-- EjemploTipos.cpp
```

Funciones I

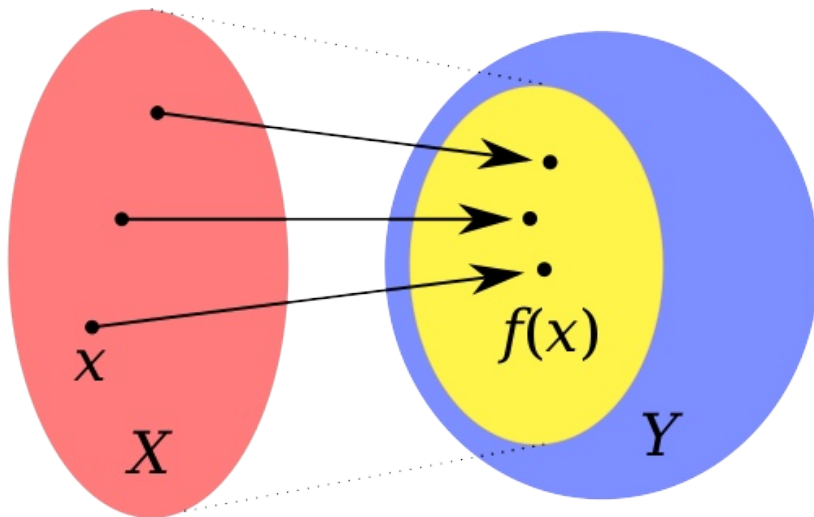
Introducción

Funciones en Matemática y en Lenguajes de Programación: Ejemplo Introductorio



$$f: \mathbb{R} \rightarrow \mathbb{R} / f(x) = 2x + 1$$

```
double f(double);  
double f(double x){ return 2*x + 1; }
```



$$f: X \rightarrow Y$$

- Conjunto de Salida y Dominio
- Conjunto de Llegada y Codominio

Programa Completo

```
#include<cassert>
#include<iostream>
```

```
double f(double);
```

```
int main(){
    std::cout << f(3);
    assert( 2*3+1 == 7 );
    assert( 7 == f(3) );
}
```

```
double f(double x){return 2*x+1;}
```

$$f: \mathbb{R} \rightarrow \mathbb{R} / f(x) = 2x + 1$$

- `assert` para pruebas, no vamos a usar `cin` o `cout` para probar
- Casos de prueba, partición de conjunto de valores
- Terminología
 - Declaración o prototipo de la función
 - Invocación
 - Argumentos
 - Definición o Implementación de la función
 - Parámetros
- Analogía con Matemática

Términos de la clase #04

Definir cada término con la bibliografía

- Pruebas Unitarias con Assert
 - assert
 - cassert
 - Pruebas automáticas
 - Comprobaciones internas
 - TDD (Test Driven Development)
 - NDEBUG
- Funciones I
 - Definición del concepto matemático
 - Notación matemática
 - Conjunto de Salida y Dominio
 - Conjunto de Llegada y Codominio
 - Imagen
 - Prototipo o Declaración de la función
 - Definición o Implementación de la función
 - Casos de pruebas
 - Invocación
 - Argumentos
 - Parámetros

Tareas para la próxima clase

1. Empezar trabajo “Ejemplos de Valores y Operaciones de Tipos de Datos”
2. Leer Funciones:
<https://josemariasola.wordpress.com/aed/papers/#Functions>
3. Investigar sobre tipos punto-flotante y como se comparan

¿Consultas?

Fin de la clase