TECNOLÓGICO DE ESTUDIOS SUPERIORES DE JOCOTITLÁN

Ing. Sistemas Computacionales

- Programa para documentar

Código

Docente:

Fernando Mercado Salinas

Grupo: Ic-301

Jocotitlán, México.

30/01/21

Investigación de Operaciones

Estudiante:

- Fernando Orea Martínez

TECNOLÓGICO NACIONAL DE MÉXICO

Introducción

Estos tipos de Programas tienen el propósito de poder ordenar y aclarar un Código independiente o externo, es poder agregar información a nuestro Código, con la finalidad de que otros usuarios puedan leerlo y saberlo interpretar de buena manera.

No solo para otros usuarios sino para nosotros mismos porque a la larga ya no sabremos como interpretarlo o a lo mejor en futuras ocasiones hacer un cambio en el Código por cualquier situación, asi se nos facilitaría más para nosotros que somos creadores del Código.

Un ejemplo donde se puede generar estos tipos de trabajos es en Doxygen que es para C++, java, Objective-C, Python, IDL y VHDL.

Ya que es muy flexible funciona en varias plataformas como es en Windows y Mac.

Desarrollo

Para la mayor parte de los programadores la codificación de algoritmos es la parte más interesante de la construcción de programas; los programadores trabajamos creando mundos virtuales, en donde todas las variables coexisten bajo nuestro perfecto control: esta es la parte más creativa del trabajo del programador. Para construir programas los programadores usamos muchos trucos, desde el uso de estructuras de datos y algoritmos avanzados.

Cuando el estudiante termina sus estudios universitarios y pase a su carrera profesional, su entrenamiento Doxygen le permitirá usar con comodidad cualquier otra herramienta de documentación, inclusive Javadoc, cuyo formato es compatible con Doxygen.

Configuración de Doxygen

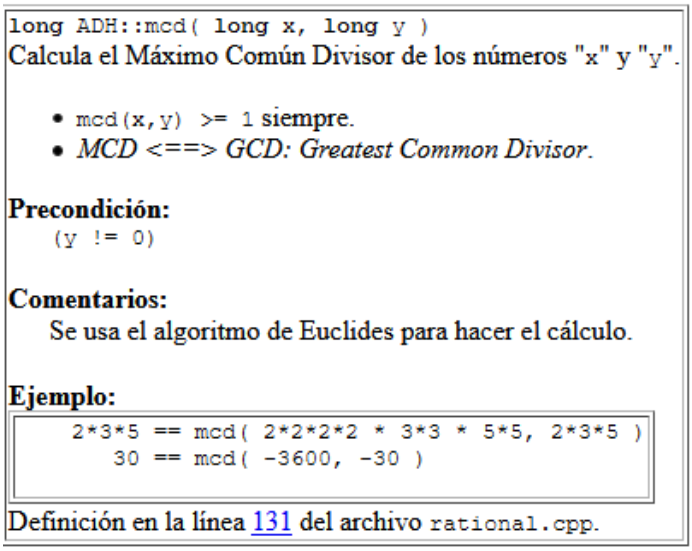
Se muestran varias funciones donde iremos aprendiendo para tener una mejor habilidad como es utilizarlo y asi presentar mejor nuestros códigos

```
/** Calcula el Máximo Común Divisor de los números \c "x" y \c "y".
    - <code> mcd(x,y) >= 1 </code> siempre.
    - MCD <==> GCD: <em> Greatest Common Divisor </em>.

    \pre
    <code> (y != 0) </code>

    \remark
    Se usa el algoritmo de Euclides para hacer el cálculo.

    \par Ejemplo:
    \code
    2*3*5 == mcd( 2*2*2*2 * 3*3 * 5*5, 2*3*5 )
       30 == mcd( -3600, -30 )
    \endcode
*/
long mcd(long x, long y) {
    long g = (x < 0 ? -x : x); // trabaja con valores positivos
    long r = (y < 0 ? -y : y); // "r" es el resto
    long temp;

    do {
        temp = r;
        r    = g % r;
        g    = temp;
    } while (0 != r);

    return g;
}  // mcd()
```

El estilo Doxygen de documentación es bastante simple e intuitivo. La documentación Doxygen comienza con el marcador de principio de comentario "/**" y termina en "*/". Esto permite mezclar la documentación con el código fuente lo que ayuda mucho a que la documentación quede actualizada cuando se le da mantenimiento a un módulo.

El primer renglón de la documentación permite definir para qué sirve el artefacto de programación. Este renglón finaliza con un punto "." que no hay que olvidar y es el que sale la la documentación general del módulo.

Los comandos especiales Doxygen "\pre", "\remarks" y "\par" permiten agregar secciones de precondición, comentarios y ejemplos a la documentación.


long ADH::mcd(long x, long y)

Calcula el Máximo Común Divisor de los números "x" y "y".

- mcd(x,y) >= 1 siempre.
- *MCD <==> GCD: Greatest Common Divisor.*

Precondición:

(y != 0)

Comentarios:

Se usa el algoritmo de Euclides para hacer el cálculo.

Ejemplo:

```
2*3*5 == mcd( 2*2*2*2 * 3*3 * 5*5, 2*3*5 )
   30 == mcd( -3600, -30 )
```

Definición en la línea 131 del archivo rational.cpp.

Hay mucho más comando donde se conforma una documentación en Doxygen, pero eso lo aprenderemos con el paso del tiempo al estar utilizando la plataforma y asi obtener nuevos desafíos y seguir obteniendo habilidades.

Conclusión

Para construir programas es necesario seguir un conjunto de buenas prácticas que facilitan la labor y hacen más fácil el mantenimiento de los módulos. El buen programador usa:

- Abstracción: para diseñar los programas por capas y por módulos.
- Especificación: para definir el rol de cada módulo en el sistema y para facilitar su reutilización.
- Encapsulamiento: para que quede junto lo que debe estar junto.
- Ocultamiento de datos: para evitar que el programador cliente tenga que lidiar con los detalles de implementación.

Que todos estos métodos los iremos aprendiendo poco a poco para asi poder hacer un buen trabajo y obtener habilidades generales a lo que es la programación.