Doxygen

Created	@March 25, 2025 9:55 AM

Documentación para Desarrollo con Doxygen

1. Introducción a la Documentación en Desarrollo de Software

La documentación es una parte esencial en el desarrollo de software, ya que facilita el mantenimiento, la colaboración y la escalabilidad del código. Existen varias herramientas para generar documentación automática a partir del código fuente:

Herramientas de Documentación Comunes

- **Doxygen**: Soporta múltiples lenguajes como C++, C, Java y Python. Genera documentación en HTML, PDF y otros formatos.
- **Javadoc**: Utilizado principalmente en Java.
- **Sphinx**: Enfocado en Python y documentación basada en reStructuredText.
- **MkDocs**: Basado en Markdown, útil para documentación de proyectos en general.

De todas estas herramientas, **Doxygen** es una de las más utilizadas en C++.

---

2. Introducción a Doxygen

Doxygen permite generar documentación a partir de comentarios en el código fuente. Soporta HTML, LaTeX (PDF), y otros formatos.

Instalación de Doxygen

En Linux (Ubuntu/Debian):

```
sudo apt install doxygen graphviz
```

En Windows:

- Descargar desde https://www.doxygen.nl/ e instalar.

En macOS:

```
brew install doxygen
```

---

3. Uso Básico de Doxygen

Generación de un archivo de configuración

Ejecuta:

```
doxygen -g Doxyfile
```

Esto crea un archivo `Doxyfile` con las configuraciones necesarias.

Generar documentación en HTML y LaTeX

```
doxygen Doxyfile
```

Los archivos generados estarán en las carpetas `html/` y `latex/`.

Para abrir la documentación en HTML:

```
xdg-open html/index.html  # Linux
start html/index.html     # Windows
open html/index.html      # macOS
```

---

4. Comentarios en el Código para Doxygen

Comentarios de Documentación

Doxygen reconoce varios formatos de comentarios:

```
/**
 * @brief Esta es una clase de ejemplo.
 *
 * La clase `Vehicle` representa un vehículo genérico.
 */
class Vehicle {
public:
    /**
     * @brief Constructor de la clase Vehicle.
     * @param brand Marca del vehículo.
         * @param speed Velocidad inicial.
     */
    Vehicle(std::string brand, int speed);

    /**
     * @brief Acelera el vehículo.
     *
     * Incrementa la velocidad en 10 unidades.
     */
    void accelerate();

    /**
     * @brief Frena el vehículo.
     *
     * Reduce la velocidad en 10 unidades.
     */
    void brake();
};
```

Etiquetas Útiles en Doxygen

- `@brief` - Breve descripción.
- `@param` - Parámetro de una función.
- `@return` - Valor de retorno de una función.
- `@see` - Referencia a otra función o clase.
- `@author` - Nombre del autor.

- `@version` - Versión del código.
- `@date` - Fecha de creación.
- `@todo` - Lista de tareas pendientes.

Ejemplo con `@todo`:

```
/**
 * @todo Implementar un método para frenar con ABS.
 */
void brakeWithABS();
```

---

5. Uso Avanzado de Doxygen

Generar Diagramas con Graphviz

Para habilitar diagramas UML en la documentación, edita el `Doxyfile` y cambia:

```
HAVE_DOT = YES
```

Esto generará diagramas de dependencias entre clases y funciones.

Documentación de Archivos y Módulos

```
/**
 * @file vehicle.h
 * @brief Definición de la clase Vehicle.
 */
```

```
/**
 * @defgroup VEHICLES Módulo de Vehículos
 * Agrupa clases relacionadas con vehículos.
 *
 * @{ */
```

Generación de Documentación en PDF

```
cd latex
make
```

El PDF se generará en `latex/refman.pdf`.

---

6. Buenas Prácticas para Documentación

- Documentar todas las clases y métodos públicos.
- Usar `@param` y `@return` correctamente.
- Mantener la documentación actualizada.
- Evitar descripciones redundantes o demasiado genéricas.

---

7. Ejercicios Prácticos

Ejercicio 1: Documentar una Clase `Person`

Crea un archivo `person.h` y documenta la siguiente clase con Doxygen:

```
class Person {
private:
    std::string name;
    int age;

public:
    Person(std::string n, int a);
    void setName(std::string n);
    std::string getName() const;
    void setAge(int a);
    int getAge() const;
};
```

Genera la documentación y verifica que aparezcan los comentarios.

Ejercicio 2: Agregar Diagramas de Clases

1. Modifica `Doxyfile` para habilitar los diagramas (`HAVE_DOT = YES`).

2. Usa `@uml{}` en un comentario para representar la relación entre `Vehicle` y `Car`.

Ejemplo:

```
/**
 * @brief Relación entre Vehicle y Car.
 * @uml{
 * class Vehicle {
 *   string brand;
 *   int speed;
 *   + accelerate();
 *   + brake();
 * }
 *
 * class Car extends Vehicle {
 *   int doors;
 * }
 * }
 */
```

Ejercicio 3: Generar Documentación en PDF

1. Ejecuta `doxygen Doxyfile`.
2. Entra a la carpeta `latex/` y ejecuta `make`.
3. Abre `refman.pdf` y revisa los contenidos.

---

8. Resumen y Recursos

Doxygen es una herramienta poderosa para documentar código C++ de manera efectiva. Su integración con Graphviz permite visualizar relaciones entre clases y módulos.

Recursos adicionales:

- Sitio oficial de Doxygen
- Ejemplos de Doxygen en GitHub
