**Alumno:** Ortuño, Bruno

**MARCO TEÓRICO**

1. Que entiende por desarrollo de software profesional y que diferencia existe con el aficionado

2. Describir los tipos de productos de Software

3. Que entiende por calidad de software?

4. Que son los atributos Emergentes?

5. Que es un CASE, que etapas cubre del Desarrollo de Software.

6. Que es un Ide, por que son tan populares.

7. Que diferencia tienen con los editores de Texto?

8. Que “problema” pueden acarrear la utilización de un IDE?

9. Describir que es C, en que lo supero C++ y algunas diferencias.

10. Cuando Utilizar C y cuando C++

11. Describir el Proceso de Compilación de un Programa en C++

—------------------------------------------------------------------------------------------------------------------

1. El desarrollo de software profesional es la creación de software destinado a ser utilizado por otros aparte de su desarrollador, con un enfoque en propósitos comerciales específicos o para su inclusión en otros dispositivos. Implica la creación de sistemas completos que van más allá de un solo programa. La diferencia principal con el desarrollo de software aficionado radica en que el software profesional se realiza en equipos, se mantiene y cambia a lo largo de su implementación, y requiere una mayor atención a la documentación, especificación y diseño del sistema, al ser utilizado por otras personas aparte del desarrollador.
2. Existen dos tipos de productos de software:

- **Productos genéricos (Enlatados):** Consisten en sistemas independientes que se producen por una organización de desarrollo y se venden en el mercado abierto a cualquier cliente que desee comprarlos. Ejemplos

de este tipo de productos incluyen software para PC, así como las llamadas “aplicaciones verticales” diseñadas para cierto propósito específico, tales como sistemas de información de stock, sistemas de contabilidad o sistemas para mantener registros de pacientes.

- **Productos personalizados (Custom o a la medida):** Son sistemas que están

destinados para un cliente en particular. Un contratista de software

desarrolla el programa especialmente para dicho cliente. Ejemplos de

este tipo de software incluyen los sistemas de control para dispositivos

electrónicos, sistemas escritos para apoyar cierto proceso

empresarial y los sistemas de control de tráfico aéreo.

1. La calidad del software es más que lo que el software puede hacer. Implica considerar cómo se comporta el software mientras se ejecuta, así como la estructura y organización de los programas del sistema y la documentación asociada. Esto se refleja en los atributos no funcionales del software, como el tiempo de respuesta ante la interacción del usuario y la legibilidad del código del programa.
2. El conjunto específico de atributos que se espera de un sistema de software depende de su aplicación, y así emergen 4 tipos de atributos: **el mantenimiento**, escribiendo código de forma que pueda evolucionar para satisfacer necesidades cambiantes del cliente; **la confiabilidad y seguridad**, el software no debe causar daño fisico ni economico en caso de falla, y los usuarios malintencionados no deben tener la posibilidad de acceder al sistema o dañarlo; **la eficiencia**, el software no tiene que desperdiciar los recursos del sistema, incluyéndose en esta categoria la capacidad de respuesta, tiempo de procesamiento, manejo de memoria, etc.; **la aceptabilidad**, el software debe ser aceptable al tipo de usuarios para quienes se diseña, necesitando ser comprensible, utilizable y compatible con otros sistemas que ellos usan.
3. Las herramientas de desarrollo del software ( o CASE, por las siglas de Computer-Aided Software Engineering) son programas usados para apoyar las

actividades del proceso de la ingeniería de software. En consecuencia, estas herramientas incluyen editores de diseño,diccionarios de datos, compiladores, depuradores, herramientas de construcción de sistema, etcétera.

Las herramientas de software ofrecen apoyo de proceso al automatizar algunas actividades del proceso y brindar información sobre el software que se desarrolla.

Cubre el Upper CASE, constituido por la planificación, análisis y diseño del software, así como el Lower CASE, que consta de la implementación, pruebas y mantenimiento.

1. Un IDE (integrated development environment) es una suite de componentes o módulos que permiten no solo escribir código fuente, sino también generar con él un programa ejecutable, aunque también es posible encontrar entornos de

desarrollo integrados, que no contienen un compilador incorporado en sí mismos, sin embargo, por lo general pueden configurarse para trabajar en conjunto con uno de ellos, para así poder producir un archivo ejecutable (a no ser que el lenguaje

que se esté empleando no requiera compilación)

Todo esto los vuelve una herramienta extremadamente popular a la hora de desarrollar software por parte de programadores.

1. Tienen la diferencia de traer consigo herramientas que facilitan el desarrollo de software, como es facilidades en la escritura de código (identación y formato asistido, coloreo de palabras clave, autocompletado de código de común uso, etc) así como en su testeo (herramientas de compilación y debug incluidas, consola, etc) Es decir, no sólo son editores de texto, sino un conjunto de herramientas pensadas para el programador.
2. Quizás el mayor peligro al usar cualquier IDE es no comprender la cadena de herramientas, es decir, el conjunto de herramientas que se utilizan en secuencia para compilar y ejecutar el código fuente.
3. C es un lenguaje de programación procesal que funciona en el nivel más bajo de abstracción y, por lo tanto, es un lenguaje de programación de sistemas. Está compilado, es liviano y ofrece administración manual de memoria. La fuerza del lenguaje de programación C reside en el rendimiento y tiene la capacidad de usarse para codificar una amplia variedad de plataformas. Por lo tanto, es el lenguaje de programación se puede usar para codificar casi cualquier cosa.

C++ surgió a partir de C,ofreciendo un bajo nivel de abstracción y requiriendo una gestión manual de la memoria, pero superandolo al permitir la implementación de un paradigma orientado a objetos.

Las principales diferencias entre ambos son raíz de la orientación de C++ por el paradigma de objetos, permitiendo tipos de datos definidos por el programador y el uso de encapsulamiento, herencia y polimorfismo.

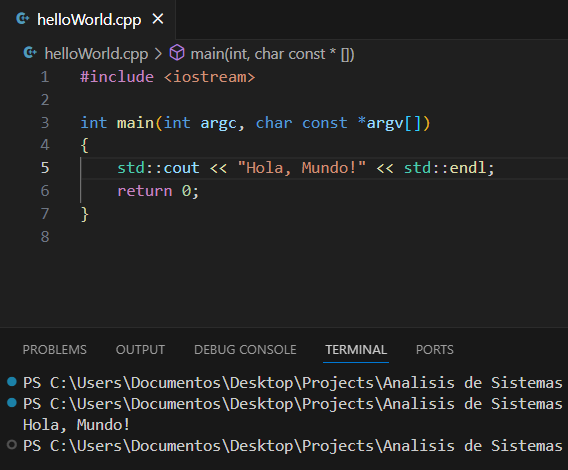
1. Debe elegirse C sobre C ++ cuando se realice la codificación de sistemas verdaderamente pequeños (resulta en un poco menos sobrecarga en comparación

con el uso de C ++), la aplicación deba ser increíblemente estable (código impermeable y resultados de control al eliminar las abstracciones de C ++), o no se tiene un compilador de C ++ en la plataforma elegida para el desarrollo de aplicaciones.

Debe elegirse C ++ sobre C cuando se desea desarrollar una aplicación que funcione directamente con el hardware de la computadora, desarrollar una aplicación que se ocupe del desarrollo de aplicaciones, o el proyecto apunta a un procesamiento de nivel extremadamente bajo.

1. La compilación es el proceso que es llevado a cabo por el compilador, donde se traduce el código fuente a código máquina en la forma de un archivo objeto, el cual aún no es un ejecutable como tal, sino que aún requiere del proceso de vinculación, realizado por el enlazador, para recién poder ser ejecutado como una aplicación por parte del usuario. Esto se debe a que el archivo objeto compilado sólo posee la traducción del código fuente a código máquina, pero no tiene consigo las posibles dependencias incluidas en sus instrucciones (como pueden ser funciones, objetos, métodos, etc.), y si estas no son incluidas dentro del código, la aplicación no podría ejecutarse.

**MARCO PRÁCTICO:**

****