PORTADA

Nombre Alumno / DNI	**Hugo Carvajal / 5048895T**
Título del Programa	**1ºPHE CYBERSECURITY & DIGITAL INTELLIGENCE**
Nº Unidad y Título	**UNIT 1-PROGRAMMING & CODING**
Año académico	**2023-2024**
Profesor de la unidad	**Gabriela García**
Título del Assignment	**AB FINAL**
Día de emisión	**13 de octubre de 2023**
Día de entrega	**Fecha en la que entrega el alumno esté AB Final. 30/01/2024**
Nombre IV y fecha	
Declaración del estudiante	**Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio.** **Entiendo que hacer una declaración falsa es una forma de mala práctica.** **Fecha: 30/01/2024** **Firma del alumno: Hugo Carvajal**

Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.

MSMK University

ÍNDICE

INTRODUCCIÓN

Los lenguajes de programación son importantes porque permiten a las personas interactuar con dispositivos informáticos, automatizar tareas, desarrollar software, automatizar tareas y respaldar los avances tecnológicos.

Los paradigmas de programación son importantes porque proporcionan un marco conceptual para el desarrollo de software e influyen en la estructura del código y la eficiencia en la resolución de problemas. Elegir el paradigma correcto puede tener un impacto significativo en la calidad y el éxito de un proyecto de software.

Los estándares de código son importantes porque proporcionan un conjunto definido de reglas para un lenguaje de programación o estilo de programación. Este estilo permite que cada ingeniero involucrado en un proyecto diseñe el código a su manera, lo que da como resultado una base de código consistente que es más fácil de leer y mantener.

TIPOS DE LENGUAJE de PROGRAMACIÓN:

- Lenguajes de bajo nivel
- Lenguajes de medio nivel
- Lenguajes de alto nivel

Descripción general de los lenguajes de alto nivel, medio nivel y bajo nivel.

Lenguajes de alto nivel: Son los más adaptados al lenguaje humano. Tienen que ser traducidos a lenguaje maquina para poder ser ejecutados por un ordenador. Los lenguajes de alto nivel se adaptan al código máquina a través de traductores y compiladores.

Lenguajes de medio nivel: Su objetivo es reunir las ventajas tanto de alto nivel como de bajo nivel. Por un lado, permite una programación más cercana al hardware que los lenguajes de nivel alto, lo que se traduce en una mayor eficacia y rendimiento. Por otro lado, también es más fácil de leer y entender que los lenguajes de nivel bajo.

Lenguajes de bajo nivel: Los lenguajes de bajo nivel son aquellos que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan.

Ejemplos representativos de cada tipo y sus usos más comunes.

Ejemplo de alto nivel: Phyton puede ser usado para prácticamente cualquier tarea que requiera un lenguaje de programación. Diseño web, creación de aplicaciones, bases de datos, automatización de tareas…

Ejemplo de medio nivel: C utilizado esencialmente para el desarrollo de software y la creación de aplicaciones de escritorio.

Ejemplo de bajo nivel: Lenguaje ensamblador es el lenguaje de programación utilizado para escribir programas informáticos de bajo nivel, y constituye la representación más directa del Código máquina específico para cada arquitectura de computadoras legible por un programador.

PARADIGMAS DE PROGRAMACIÓN

Descripción detallada de los principales paradigmas: Imperativo, Declarativo, Orientado a Objetos, Funcional, Lógico, entre otros.

IMPERATIVO: El paradigma imperativo apareció en los 50 con los primeros lenguajes de programación. También es llamado procedimental o algorítmico. La programación en el paradigma imperativo consiste en determinar que datos son requeridos para el cálculo, asociar a estas direcciones de memoria y efectuar, paso a paso, una secuencia de transformaciones en los datos almacenados.

ALGUNAS VENTAJAS:

- Su relativa simplicidad y facilidad de implementación de los compiladores
- La capacidad de reutilizar el mismo código en diferentes lugares en el programa sin copiarlo
- La capacidad de ser muy modular o estructurado

ALGUNAS DESVENTAJAS:

- Los datos son expuestos a la totalidad del programa
- Dificultad para relacionarse con los objetos del mundo real
- Difícil crear nuevos tipos de datos reduce la extensibilidad

DECLARATIVO: El paradigma declarativo está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla.

ALGUNAS VENTAJAS:

- Ventaja de ser razonados matemáticamente, lo que permite el uso de mecanismos matemáticos para optimizar el rendimiento de los programas.
- Son fiables y expresivos

ALGUNAS DESVENTAJAS:

- Basado en una forma de pensar no habitual en las personas

ORIENTADO A OBJETOS: El Paradigma orientado a objetos se empezó a desarrollar entre los años 1960 y 1970. Este consiste en que los programas se estructuran alrededor de "objetos", que son instancias de clases. Estas clases actúan como moldes que definen las propiedades y comportamientos comunes de los objetos. Un objeto puede contener datos (atributos) y funciones (métodos) que operan en esos datos.

ALGUNAS VENTAJAS:

- Alta calidad del código.
- Bajo costo en fases de desarrollo.
- Modificabilidad: La facilidad de añadir o suprimir nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.
- Encapsulamiento: Nos permite proteger la integridad de los datos.
- Amplia documentación

ALGUNAS DESVENTAJAS:

- Curva de aprendizaje: La necesidad de utilizar bibliotecas de clases obliga a su aprendizaje y entrenamiento.
- La ejecución de programas orientados a objetos es más lenta.
- Tiempo en fase de diseño.

Funcional: Este paradigma se basa en un conjunto de funciones que pueden ser evaluadas para obtener un resultado. El paradigma funcional está basado en conceptos que vienen de la matemática.

ALGUNAS VENTAJAS:

- No contienen sentencias de asignación
- Como las funciones puras no cambian ningún estado y dependen completamente de la entrada

ALGUNAS DESVENTAJAS:

- Es difícil de mantener, ya que durante la codificación evolucionan muchos objetos.
- La reutilización es muy complicada y necesita una constante refactorización.

Lenguajes representativos de cada paradigma y sus características distintivas.

PARADIGMA IMPERATIVO

Lenguajes representativos: Basic, C, D, Fortran, Pascal, Perl, PHP y Lua

Características:

- Describe cómo debe realizarse el cálculo, no el porqué.
- Las variables son celdas de memoria que contienen datos (o referencias), pueden ser modificadas, y representan el estado del programa.
- La sentencia principal es la asignación.
- Basado en el modelo de cómputo de máquinas de Turing y sobre todo en las máquinas RAM (registro + acceso aleatorio a memoria)

PARADIGMA DECLARATIVO

Lenguajes representativos: SQL y Prolog

Características:

- Describe las relaciones y restricciones entre variables sin especificar la secuencia de pasos.
- Menos énfasis en el control del flujo y más en la lógica.

PARADIGMA ORIENTADO A OBJETOS

Lenguajes representativos C++, Phyton , PHP, Java, VisualBasic 6.0 entre otros.

Características:

- Encapsulamiento: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.
- Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre
- Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos.
- Modularidad: la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

PARADIGMA FUNCIONAL

Lenguajes representativos: Lisp y Haskell

Características:

- No existe operación de asignación.
- Las “variables” almacenan definiciones o referencias a expresiones.
- Basado en el concepto (matemático) de función.
- La operación fundamental es la aplicación de una función a una serie de argumentos. La evaluación se guía por el concepto de sustitución.
- Un programa consiste en una serie de definiciones (de funciones, de tipos de datos.)
- Las estructuras de control básicas (y generalmente únicas) son la composición y la recursión.

PARADIGMA LÓGICO

Lenguajes representativos: ALF, CLP, Elf, Flang, Godel, KLIC, Logtalk, Prolog.

Características:

- Los programas para los lenguajes de programación lógicos son un conjunto de hechos y reglas.
- La sintaxis de los lenguajes de programación lógicos es notablemente diferente de los lenguajes de programación imperativos.
- Unificación de términos.
- Visión lógica de la computación.
- El programa se transforma en un conjunto de declaraciones formales de especificaciones que deben ser correctas por definición.
- No tiene un algoritmo que indique los pasos que detallen la manera de llegar a un resultado.

ESTÁNDARES DE PROGRAMACIÓN

Introducción a la importancia de seguir estándares.

Seguir estándares de programación en el desarrollo de software mejora la calidad del código y también facilita la colaboración, la interoperabilidad y la adaptabilidad a medida que los proyectos evolucionan. La adopción de estándares es esencial para construir sistemas informáticos robustos y sostenibles en el largo plazo.

Descripción de algunos estándares de programación populares y sus características principales.

OWASP Top 10: es una organización que se compromete a mejorar la seguridad de las aplicaciones web. Como tal, su proyecto OWASP Top 10 proporciona una lista de las vulnerabilidades de seguridad de aplicaciones web más comunes y de mayor impacto. La última versión de OWASP Top 10 está directamente correlacionada con ID de CWE específicos e incluye metadatos de riesgo.

Características:

- Lista de las diez amenazas de seguridad más críticas en aplicaciones web.
- Proporciona pautas y buenas prácticas para mitigar riesgos de seguridad.
- Ayuda a los desarrolladores a construir aplicaciones más seguras.

MISRA C/C++: Desarrollado por Motor Industry Software Reliability Association, describe un subconjunto del lenguaje C o C++ y las pautas para su uso para mejorar la seguridad de la aplicación. Aunque originalmente estaba destinado a aplicaciones automotrices, se usa en todas las industrias, particularmente para aplicaciones críticas para la seguridad.

Características:

- Uso seguro de la biblioteca de C++
- Incluye directrices para la programación orientada a objetos, considerando la encapsulación, herencia y polimorfismo**.**

Beneficios de adherirse a estándares y consecuencias de no hacerlo.

Beneficios

Detección temprana de fallas: Al buscar cumplir con los estándares que establezcas es más sencillo detectar posibles errores desde la revisión de código, evitando que esos problemas lleguen a producción.

Reducción de la complejidad: El cumplir con las reglas acerca del estilo de código ayuda a construir código más limpio, permitiendo detectar fácilmente oportunidades para simplificar funciones.

Código de fácil lectura: El respetar los estándares en un proyecto les permite a nuevos miembros del equipo acoplarse más fácilmente al ritmo de trabajo y a entender mejor el código de los proyectos existentes.

Código reusable: Contar con segmentos de código que pueden ser consumidos por más de un servicio, gracias al uso de buenas prácticas, hace menos frecuente la repetición de código.

Consecuencias de no adherirse a estándares

Mayor tendencia a errores: El no utilizar estándares puede aumentar la probabilidad de cometer errores en el código.

Código menos legible: Puede resultar un gran problema ya que puede haber una falta de coherencia en el código tanto para el desarrollador original como para los que mas adelante trabajen en él.

Dificultad en el mantenimiento del código.

CONCLUSIÓN

Reflexión sobre la importancia de entender los diferentes lenguajes, paradigmas y estándares y cómo estos influyen en el desarrollo de software.

En resumen, entender los diferentes lenguajes, paradigmas y estándares mejoran nuestras habilidades técnicas y también nos ayuda a abordar problemas de manera más efectiva y contribuye a un desarrollo de software de mayor calidad.

REFERENCIAS

Introducción:

Avaible at: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

https://desarrolloweb.com/articulos/paradigmas-programacion

https://blog.thedojo.mx/2021/10/05/estandares-de-calidad-en-el-software.html

Tipos de Lenguajes de Programación

Avaible at: https://assemblerinstitute.com/blog/tipos-lenguaje-programacion/

Descripción general de los lenguajes de alto nivel, medio nivel y bajo nivel.

Avaible at: https://assemblerinstitute.com/blog/tipos-lenguaje-programacion/

https://www.ismos.es/que-es-un-lenguaje-de-medio-nivel-ejemplos/

https://es.wikipedia.org/wiki/Lenguaje_de_bajo_nivel

Ejemplos representativos de cada tipo y sus usos más comunes.

Avaible at: https://assemblerinstitute.com/blog/tipos-lenguaje-programacion/

https://www.ecured.cu/Lenguaje_ensamblador

Paradigmas de Programación

Descripción detallada de los principales paradigmas: Imperativo, Declarativo, Orientado a Objetos, Funcional, Lógico, entre otros.

Avaible at: https://es.slideshare.net/JFREDYOLAYARAMOS/paradigma-imperativo-39302522

https://www.ecured.cu/Programaci%C3%B3n_declarativa

https://ferestrepoca.github.io/paradigmas-de-programacion/poo/poo_teoria/index.html

https://wiki.uqbar.org/wiki/articles/paradigma-funcional.html

https://www.lifeder.com/programacion-funcional/

Lenguajes representativos de cada paradigma y sus características distintivas.

Avaible at: https://kevinldp.wordpress.com/

Estándares de Programación:

Introducción a la importancia de seguir estándares.

Avaible at: https://blog.thedojo.mx/2021/10/05/estandares-de-calidad-en-el-software.html

Descripción de algunos estándares de programación populares y sus características principales.

Avaible at: https://es.parasoft.com/blog/an-ounce-of-prevention-software-safety-security-through-coding-standards/

https://www.mathworks.com/discovery/misra-c.html

Beneficios de adherirse a estándares y consecuencias de no hacerlo.

Avaible at: https://blog.thedojo.mx/2021/10/05/estandares-de-calidad-en-el-software.html