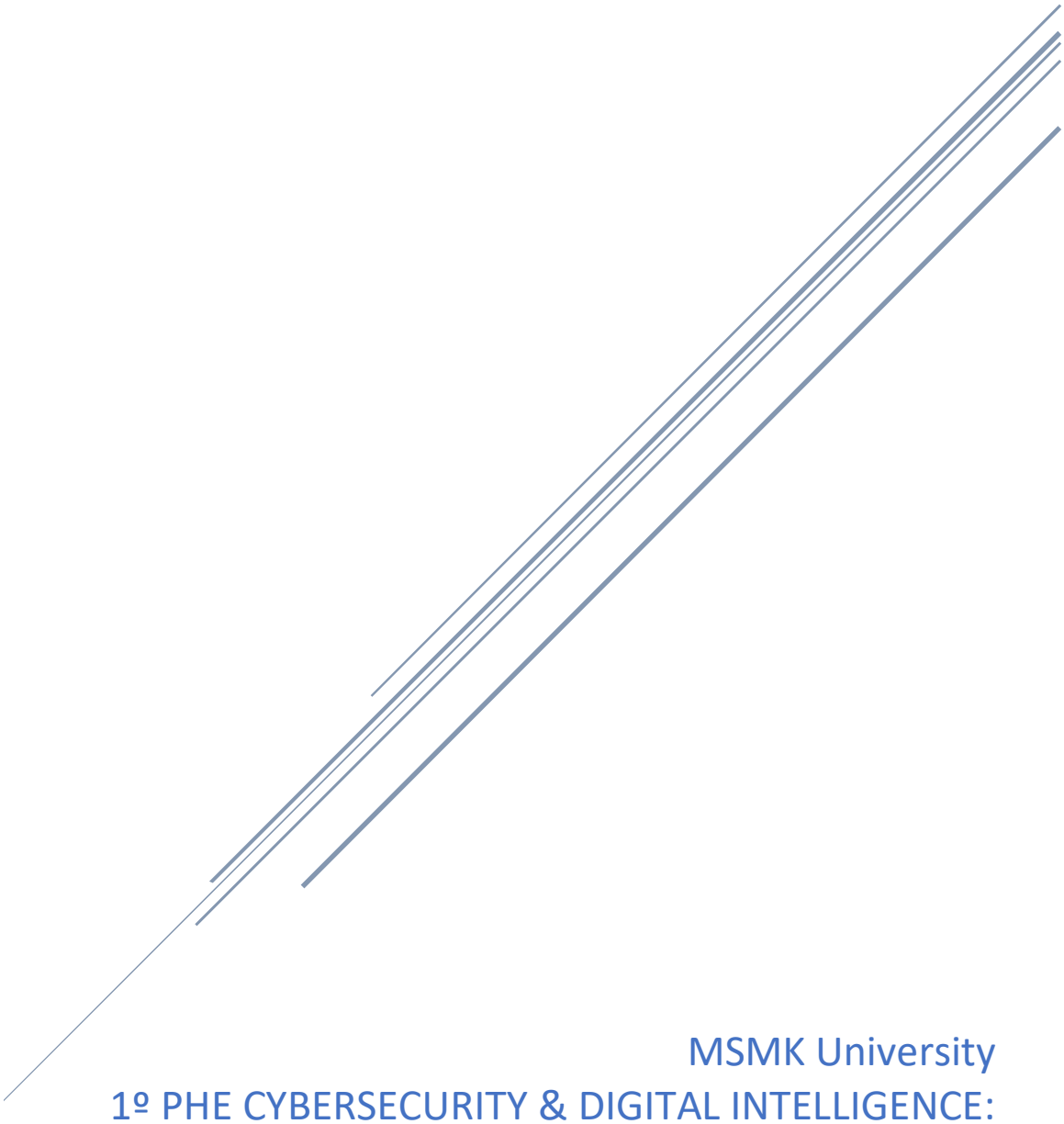
MSMK

INFORME DE LENGUAJES, PARADIGMAS Y ESTÁNDARES DE PROGRAMACIÓN

Jacobo Artem Guinot Almenar



MSMK University

1º PHE CYBERSECURITY & DIGITAL INTELLIGENCE: PROGRAMMING AND CODING

ÍNDICE

INTRODUCCIÓN

Los **lenguajes de programación** nos permiten indicar a los computadores, de forma detallada, qué tienen que hacer y cómo han de hacerlo para, entre otras cosas:

- Navegar por internet.
- Utilizar aplicaciones móviles, como redes sociales, utilidades, juegos…
- Desarrollo de software enfocado para empresas.
- Desarrollo de sistemas en red.
- Automatización de tareas, procesos y manejo de grandes volúmenes de datos.

Se denominan **paradigmas de programación** a las formas de clasificar los lenguajes de programación en función de sus características, aunque se trata de un concepto al que va ligada alguna controversia, debido a que muchos lenguajes de programación no pueden clasificarse estrictamente en un paradigma, sino que incluyen características de varios paradigmas.

Varias de las desventajas a la hora de programar sin hacer uso de los paradigmas de programación serían:

- Código desorganizado y dificultad en la depuración.
- Menor escalabilidad y mayor propensión a errores.
- Dificultad en el mantenimiento temporal y menor eficiencia.

Los **estándares de programación** nos permiten disponer de guías en las etapas de diseño de los sistemas, software o programas que se vayan a desarrollar. Esto implica que cada estándar de programación es dependiente al lenguaje de programación que estes utilizando.

Ventajas de la implementación de estándares:

- Detección temprana de fallos o errores de programación.
- Reducción de la complejidad del código desarrollado.
- Lectura legible del código.
- Mejora significativa en la calidad del código desarrollado.
- Código reusable.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Los **tipos de lenguajes de programación** están clasificados en tres niveles, lenguajes de bajo nivel, lenguajes de medio nivel y lenguajes de alto nivel. Cada lenguaje de programación según su tipo tiene características únicas que los diferencian del resto.

LENGUAJES DE BAJO NIVEL

Los lenguajes de bajo nivel están compuestos por dos lenguajes:

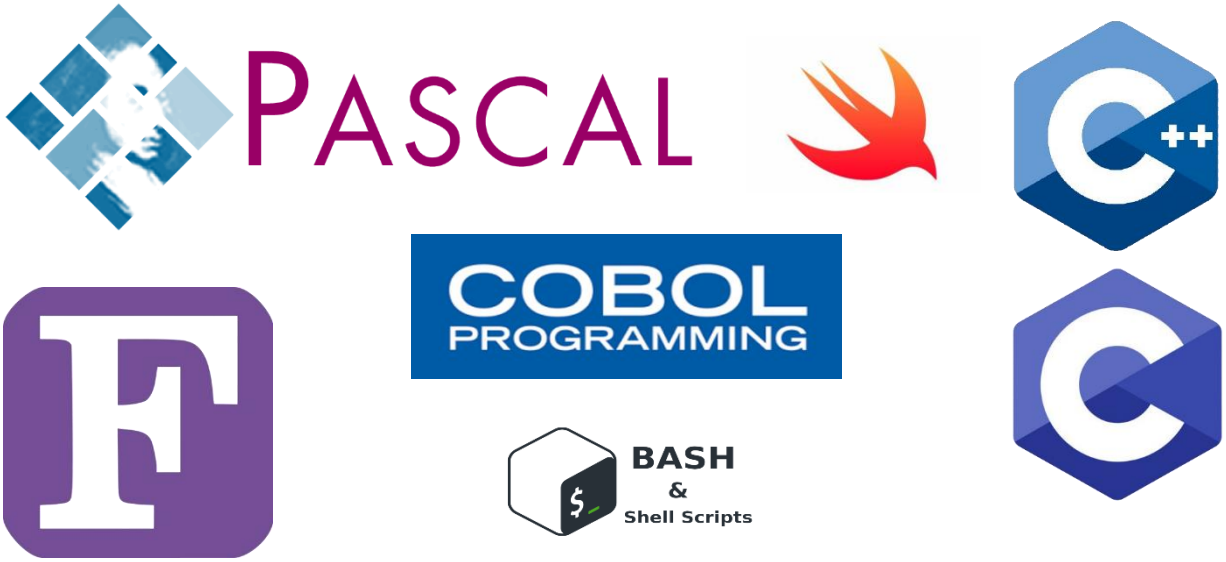
- **Lenguaje máquina**: se basa en el código binario y su complejidad le precede, además es el único que comprenden los ordenadores, y no es útil para la creación de programas o aplicaciones web.
- **Lenguaje ensamblador**: tiene las mismas funciones que el lenguaje máquina, la única diferencia es que esta diseñado para ser escritos por humanos de manera sencilla.

LENGUAJES DE MEDIO NIVEL

Los lenguajes de medio nivel suelen tender a clasificarse como lenguajes de bajo nivel porque siguen dependiendo de ensambladores para que los ordenadores comprendan la sintaxis declarada.

Permiten llevar a cabo tareas mucho más complejas, como el uso de funciones y añadiendo que también permiten los algoritmos de búsqueda y ordenamiento.

Los lenguajes de programación más comunes de medio nivel son **C**, **C++**, **Pascal**, **Cobol**, **Swift, Basic, Bash** y **FORTRAN**.



LENGUAJES DE ALTO NIVEL

Los lenguajes de alto nivel son los más adaptados al lenguaje humano, por lo que en términos generales son los más sencillos de aprender, estos se adaptan al código máquina a través de compiladores.

La principal diferencia entre los lenguajes de alto nivel con los lenguajes de bajo nivel es el nivel de abstracción que ofrece los lenguajes de alto nivel en comparación con los lenguajes de bajo nivel.

Los lenguajes de programación más comunes de alto nivel son **Java**, **Python**, **JavaScript**, **Ruby, PHP, SQL** y **Perl**.

UTILIDADES MÁS COMUNES

A continuación, cada uno de los ejemplos representativos de cada tipo de lenguaje de programación destacará según su utilidad más común usada por el humano para su posterior desarrollo.

- Las utilidades más comunes ofrecidas por los **lenguajes de bajo nivel** son la traducción de la sintaxis empleada en el desarrollo de un código para la previa comprensión por parte del ordenador y su posterior correcta ejecución.
- En cambio, los **lenguajes de medio** y **alto nivel** están dirigidos al desarrollo de aplicaciones de usuario, aplicaciones web, desarrollo de sistemas y enfocado al desarrollo comercial y videojuegos.
- No obstante, el desarrollo de videojuegos puede llegar a un nivel muy bajo, como al nivel de ensamblador, para que se pueda extraer las mayores prestaciones del hardware (Memoria, CPU, GPU…)

Pongamos varios ejemplos de cada lenguaje de programación según su objetivo y sus necesidades sean sencillas:

C & C++

Los lenguajes de programación **C/C++** se utilizarían desde programación de sistemas, hasta nivel de aplicación de consola.

COBOL

Totalmente comercial, "**Common Business Oriented Language**" se utiliza para las entidades financieras, para la gestión de grandes volúmenes de datos de los clientes en relación al uso anterior.

Python

Se utiliza para desarrollar software, páginas webs, automatizar tareas, análisis de datos, es un lenguaje de propósito general, al igual que el Basic.

JavaScript

Está orientado para implementar funciones más complejas en la parte del cliente en un entorno web, por ejemplo: actualización del contenido, funcionalidades interactivas, animación grafica…

PHP

El objetivo de este lenguaje de programación es generar código HTML que se le envía a la parte cliente, sobre la marcha se genera según las peticiones del cliente. Por ejemplo; si el cliente quiere solicitar datos de un producto, el cliente introduce un valor y tu como servidor recibes ese código, el servidor consulta en la base de datos (SQL, MySQL, MariaDB…) y generas en ese momento en HTML la página con los datos que ha solicitado el cliente.

SQL

Como tal "**Structured Query Language**" no es un lenguaje de programación, está enfocado al acceso y consulta de bases de datos relaciones (RDBMS).

PARADIGMAS DE PROGRAMACIÓN

Un paradigma de programación es una manera o estilo de programación de software. Se trata de un conjunto de métodos sistemáticos aplicables en todos los niveles de diseño de programas para resolver problemas computacionales.

PARADIGMA IMPERATIVO

Los programas con el uso de un paradigma imperativo consisten en una sucesión de instrucciones o conjunto de sentencias, por ejemplo: el programador da ordenes concretas para el cumplimiento del código, describe en el paso a paso absolutamente todo lo que hará con su programa.

Algunos de los lenguajes de programación que siguen este paradigma son: **Pascal, COBOL, FORTRAN C, C++…**

Aunque otros enfoques subordinados al paradigma de programación imperativo son:

- Programación estructurada.
- Programación procedimental.
- Programación modular.

PARADIGMA DECLARATIVO

Este en concreto no necesita definir algoritmos puesto que describe el problema en lugar de encontrar una solución al mismo, utiliza el principio del razonamiento lógico para responder a las dudas o cuestiones consultadas.

Este paradigma se divide en dos:

- Paradigma Lógica: "**Prolog**".
- Paradigma Funcional: **Lisp, Scala, Java, Kotlin**.

PARADIGMA ORIENTADO A OBJETOS

Este modelo de paradigma permite separar los diferentes componentes de un programa, simplificando así su creación, depuración y posteriores mejoras. También este modelo disminuye los errores y es propenso a la reutilización del código.

Algunos de los lenguajes de programación orientados a objetos son **Java, Python, C++** o **C#**.

Varios de los diferentes conceptos a los que se sirve la programación orientada a objetos son la abstracción de datos, la encapsulación, los eventos, la herencia, el polimorfismo y modularidad.

ESTÁNDARES DE PROGRAMACIÓN

Los estándares de programación se basan en convenciones para escribir código fuente en ciertos lenguajes de programación, así como los comandos básicos que se utilizan para poder programar.

¿Por qué es tan importante seguir los estándares de programación?

Generar un proyecto con un estilo que garantiza que todos los programadores que contribuyan en el proyecto tengan una forma única de diseñar su código, que como resultado podamos obtener una base de código coherente, seguido de una lectura sencilla y un mantenimiento de código sano.

ESTÁNDARES DE PROGRAMACIÓN POPULARES

- **Priorizar la legibilidad:** es de vital importancia desarrollar un código legible que cualquier programador pueda entender.
- **Arquitectura:** disponer de una estructura determinada a la hora de desarrollar un proyecto es fundamental para conocer la utilidad del código, como funcionará y con qué servicios será compatible.
- **Inserción de comentarios:** insertar comentarios en el código es de gran relevancia para entender en primera instancia que utilidad tiene el código.
- **Comprobaciones y testeos:** realizar comprobaciones realizando un análisis de código estático y dinámico puede evitar en un futuro que puedan surgir problemas en el funcionamiento del mismo.
- **Simplificación:** se recomienda a los programadores simplificar al máximo el desarrollo de su código, asi evitando que puedan encontrarse bugs y un ahorro en el tiempo a la hora de la búsqueda y resolución de errores (**Diseño TopDown**).
- **Control de versiones:** conocer la versión del servicio, librería, framework o módulo que estes haciendo uso, facilita la gestión y prevención de vulnerabilidades y compatibilidad del código.
- **No reproducir fragmentos idénticos de código:** independientemente del lenguaje de programación que estes utilizando, en los lenguajes de programación cuyo uso de paradigma es orientado a objetos lo más recomendable es tratar de encapsular parte del código que se haya podido repetir, en una función de una clase y aprovecharla cuando sea necesario.

El lenguaje de programación Python de alto nivel dispone de un estilo propio en el estándar de programación establecido como PEP 8 o para el lenguaje de programación Java esta Google Java Style Guide.

CARACTERÍSTICAS

En los estándares de programación deben encontrarse diferentes nomenclaturas para que los nombres de las funciones y variables sean auto explicativos, por ejemplo; con únicamente leer el nombre sepas que realiza.

- Nomenclaturas de ficheros.
- Nomenclaturas de funciones y módulos.
- Nomenclaturas de variables.
- Lista de qué hacer y que no hacer.
- Hacer que el código sea simple.

Restricción en el ámbito de las variables, es decir, dentro de lo posible no poner variables globales, una función no debe conocer el valor asignado de otras variables que no son propias o de otras funciones.

La identación y la cohesión debe perseverar en el código desarrollado para una buena práctica de desarrollo.

BENEFICIOS Y CONSECUENCIAS

Los beneficios que se persiguen con el uso de los estándares de programación es el fácil mantenimiento a través de la legibilidad y sobre todo que sea sencillo la corrección de errores, el cambio o modificación de funciones o ampliación de funciones.

Como consecuencias al no seguir de forma apropiada los estándares de programación sugeridos por la comunidad o en el caso de cada lenguaje de programación disponga de la documentación de sus propias guías de estandarización pueden acarrear problemas en las diferentes etapas de desarrollo de una aplicación de usuario, aplicación web, software empresarial, sistemas…

En primer lugar, la frustración del programador a la hora de entender el código, comprobar el ordenamiento, detección de errores o resolución de estos puede ser muy elevada, ya que si no es de su propiedad el código desarrollado no conoce la arquitectura del código.

En segundo lugar, no se logrará una completa integridad del código, no realizar comprobaciones y testeos incrementa la probabilidad de fallas en el código, a esto se le suma, no conocer previamente las versiones de las librerías, módulos o frameworks se están utilizando, pudiendo presentar vulnerabilidades

En ultimo lugar, no respetar las diversas nomenclaturas para que sean totalmente auto explicativas y que el programador pueda comprenderlo de forma sencilla y clara puede derivar a la frustración anteriormente nombrado. También que no este simplificado el código puede ser producto de una mala práctica de estandarización y que tenga que estar lo máximo reducido posible.

CONCLUSIÓN

Como finalización del informe, no podíamos olvidarnos de a que conclusión se ha llegado después de haberse redactado y argumentado adecuadamente.

Básicamente me voy a basar en la relación que tiene conocer y entender que tipos de lenguajes de programación existen, que paradigmas de programación hay y los estándares de programación que han de seguirse con la ciberseguridad.

En primera instancia, como ingeniero de ciberseguridad debo tener en cuenta que la mayoría de los sistemas, aplicaciones u otros servicios digitales están programados por lenguajes de programación, esto implica que conocer que tipo de lenguajes de programación existen me ayuda a poder clasificarlos y que comportamiento tienen según su tipo.

Entender como funciona el código de una aplicación de usuario, aplicación web o software me permite realizar comprobaciones de seguridad con mayor profundidad y menor desconocimiento sobre la legibilidad de lo que estoy leyendo. Por ejemplo:

Si yo dispongo de una aplicación de usuario desarrollada en lenguaje de programación C++ y debo implementar la ingeniería inversa para la obtención de código malicioso inyectado, que mejor conocer como esta compuesto el código según el paradigma de programación asignado al lenguaje de programación previamente nombrado. En el caso de que un método tenga un valor que comprometa la integridad del sistema yo sepa que esa función esta encapsulada y pueda sanitizarla posteriormente. Asimismo, realizar auditorías de código estático o dinámico me puede facilitar la automatización de búsqueda de errores.

Si un ingeniero de ciberseguridad no conoce los estándares de programación puede llegar a presentar problemas a la hora de trabajar con sistemas, aplicaciones, software…

REFERENCIAS

1. Tipos de lenguaje de programación - Assembler Institute
2. High Level Language that is Examples - TecnoBits ▷ →
3. ¿Qué son los paradigmas de programación? (profile.es)
4. Programación Orientada a Objetos en Java - AEPI (asociacionaepi.es)
5. La Importancia de los Estándares de Código - The Dojo MX Blog
6. 7 Buenas prácticas en programación: Los indispensables (epitech-it.es)
7. 8 Buenas prácticas en programación | KeepCoding Bootcamps
8. Lenguajes de programación: tipos, características y diferencias (chakray.com)
9. ▶ Las claves de las Buenas Prácticas en programación - GUÍA 2023 (craft-code.com)