ICDL
The Digital Skills Standard

ICDL Profesional

FUNDAMENTOS DE PROGRAMACIÓN

Programa de estudio 1.0

Documento del programa de estudio

Objetivo

Este documento presenta el programa de estudio para el módulo de Fundamentos de programación. El programa de estudio describe, a través de los aprendizajes, el conocimiento y las competencias necesarias que debería poseer un candidato para el módulo de Fundamentos de programación. El programa de estudio también ofrece una base para el examen teórico y práctico que comprende este módulo.

Descargo de responsabilidades

A pesar de que ICDL Foundation ha tomado todos los recaudos para la elaboración de esta publicación, ICDL Foundation, en calidad de editor, no garantiza la que la información contenida aquí esté completa, ni tampoco ICDL Foundation será responsable de ningún error, omisión, imprecisión, pérdida o daño que surja en virtud de dicha información o de toda instrucción o recomendación presente en esta publicación. ICDL Foundation se reserva el derecho, a su entera discreción, de realizar cambios en cualquier momento y sin previo aviso.

Fundamentos de programación

Este módulo expone conceptos y habilidades esenciales en relación con la habilidad de utilizar pensamiento computacional y programación para crear simples programas de computación.

Objetivos del módulo

Los candidatos exitosos serán capaces de:

- Comprender conceptos claves relacionados con los fundamentos de la programación y las actividades típicas al crear un programa.
- Comprender y utilizar técnicas de pensamiento computacional como descomposición de problemas, reconocimiento de patrones, abstracción y algoritmos para analizar un problema y desarrollar soluciones.
- Escribir, probar y modificar algoritmos de programas con diagramas de flujo y pseudocódigos.
- Comprender los principios y términos claves asociados con la programación y la importancia de un código bien estructurado y documentado.
- Comprender y utilizar sentencias o expresiones de programación como variables, tipos de datos y lógicas en un programa.
- Mejorar la eficacia y funcionalidad al utilizar iteraciones, instrucciones condicionales, procedimientos y funciones, como también eventos y comandos en un programa.
- Probar y depurar un programa y asegurar que cumple con los requisitos antes de implementarse.

CATEGORÍA	ÁREA DE CONOCIMIENTO	REF.	UNIDAD DE TRABAJO
1 Términos de programación	*1.1 Conceptos clave*	1.1.1	Definir el término “programación”.
		1.1.2	Definir el término “pensamiento computacional”.
		1.1.3	Definir el término “programa”.
		1.1.4	Definir el término “código”. Distinguir entre código fuente, código objeto.
		1.1.5	Comprender los términos “descripción de programa” y “especificación de programa”.
		1.1.6	Reconocer las actividades típicas al crear un programa: análisis, diseño, programación, prueba, mejoramiento.
		1.1.7	Comprender la diferencia entre un lenguaje formal y un lenguaje natural.
2 Métodos de pensamiento computacional	*2.1 Análisis de problemas*	2.1.1	Describir los métodos típicos utilizados en pensamiento computacional: descomposición, reconocimiento de patrones, abstracción, algoritmos.
		2.1.2	Utilizar el método de descomposición de problemas para analizar datos y procesos o dividir un problema complejo en partes pequeñas.

CATEGORÍA	ÁREA DE CONOCIMIENTO	REF.	UNIDAD DE TRABAJO
		2.1.3	Identificar patrones en problemas pequeños, divididos.
		2.1.4	Utilizar el método de abstracción para ignorar detalles innecesarios al analizar un problema.
		2.1.5	Comprender cómo se utilizan los algoritmos durante el pensamiento computacional.
	2.2 Algoritmos	2.2.1	Definir el término “secuencia” de las construcciones de programación. Describir el propósito de secuenciar al diseñar algoritmos.
		2.2.2	Reconocer los posibles métodos de representar problemas como: diagramas de flujo, pseudocódigo.
		2.2.3	Reconocer símbolos de diagramas de flujo como: inicio/final, proceso, decisión, entrada/salida, conector, línea de flujo.
		2.2.4	Describir la secuencia de operaciones representadas por un diagrama de flujo, un pseudocódigo.
		2.2.5	Escribir un algoritmo preciso basado en una descripción utilizando una técnica como: diagrama de flujo, pseudocódigo.
		2.2.6	Corregir errores en un algoritmo: pérdida de un elemento del programa, secuencia incorrecta, resultado incorrecto de decisión.
3 Introducción para programar	*3.1 Introducción*	3.1.1	Describir las características de un código bien estructurado y documentado como: indentación comentarios adecuados, nombres descriptivos.
		3.1.2	Utilizar simples operadores aritméticos para realizar cálculos en un programa: +, -, /, *.
		3.1.3	Comprender la prioridad de los operadores y del orden de evaluación en expresiones complejas. Comprender cómo utilizar paréntesis para estructurar expresiones complejas.
		3.1.4	Comprender el término “parámetro”. Describir el propósito de los parámetros en un programa.
		3.1.5	Definir el término “comentario” de las construcciones de programación. Describir el propósito de un comentario en un programa.
		3.1.6	Usar comentarios en programas.
	3.2 Tipos de datos y variables	3.2.1	Definir el término “variable” de las construcciones de programación. Describir el propósito de una variable en un programa.
		3.2.2	Definir e inicializar una variable.

CATEGORÍA	ÁREA DE CONOCIMIENTO	REF.	UNIDAD DE TRABAJO
		3.2.3	Asignarle algún valor a una variable.
		3.2.4	Utilizar variables con nombres adecuados en un programa para calcular, almacenar valores.
		3.2.5	Utilizar tipos de datos en un programa: cadena, carácter, entero, punto flotante, booleano.
		3.2.6	Utilizar tipos de datos agregados en un programa como: matriz, lista, registro.
		3.2.7	Utilizar datos ingresados por un usuario en un programa.
		3.2.8	Utilizar datos de salida a una pantalla en un programa.
4 Programación	*4.1 Lógica*	4.1.1	Definir el término “prueba lógica” de las construcciones de programación. Describir el propósito de una prueba lógica en un programa.
		4.1.2	Reconocer tipos de expresiones lógicas booleanas para generar un valor verdadero o falso como: =, >, <, >=, <=, <>, !=, ==, AND, OR, NOT.
		4.1.3	Utilizar expresiones lógicas booleanas en un programa.
	4.2 Iteración	4.2.1	Definir el término “bucle” de las construcciones de programación. Describir el propósito y las ventajas de hacer bucles en un programa.
		4.2.2	Reconocer tipos de bucle utilizados para iterar como: *for*, *while*, *repeat*.
		4.2.3	Utilizar tipos de iteración (bucles) en un programa como: *for*, *while*, *repeat*.
		4.2.4	Comprender el término “bucle infinito”.
		4.2.5	Comprender el término “recursividad”.
	4.3 Condicionalidad	4.3.1	Definir el término “instrucción condicional” de las construcciones de programación. Describir el propósito de las instrucciones condicionales en un programa.
		4.3.2	Utilizar instrucciones condicionales *IF...THEN...ELSE* en un programa.
	4.4 Procedimientos y funciones	4.4.1	Comprender el término “procedimiento”. Describir el propósito de un procedimiento en un programa.
		4.4.2	Escribir y nombrar un procedimiento en un programa.
		4.4.3	Comprender el término “función”. Describir el propósito de una función en un programa.

CATEGORÍA	ÁREA DE CONOCIMIENTO	REF.	UNIDAD DE TRABAJO
		4.4.4	Escribir y nombrar una función en un programa.
	4.5 Eventos y comandos	4.5.1	Comprender el término “evento”. Describir el propósito de un evento en un programa.
		4.5.2	Utilizar controladores de eventos como: click del mouse, entrada de teclado, clic de un botón, temporizador.
		4.5.3	Utilizar librerías genéricas disponibles: math, random, time.
5 Prueba, depuración e implementación	*5.1 Ejecutar, probar y depurar*	5.1.1	Comprender las ventajas de probar y depurar un programa para resolver errores.
		5.1.2	Comprender los tipos de error en un programa como: sintaxis, lógica.
		5.1.3	Ejecutar un programa.
		5.1.4	Identificar y arreglar un error de sintaxis en un programa como: ortografía incorrecta, puntuación faltante.
		5.1.5	Identificar y arreglar un error de lógica en un programa como: expresión booleana incorrecta, tipo de dato incorrecto.
	5.2 Implementar	5.2.1	Comparar el programa con los requisitos de la descripción inicial.
		5.2.2	Describir el programa completo, comunicando el propósito y el valor.
		5.2.3	Identificar mejoras, adiciones al programa que puedan satisfacer necesidades adicionales, relacionadas.