Curso:
Codo a codo inicial

Plan de estudios

Nuestro objetivo

Nuestros cursos están destinados a personas jóvenes y adultas, que busquen desarrollarse profesional y personalmente, ampliando sus oportunidades laborales de acuerdo a los desafíos que plantea el Siglo XXI.
El objetivo es brindar herramientas que faciliten la inserción laboral en el sector Informática (IT), y en particular, fomentar la participación de las mujeres, con el fin de mejorar su empleabilidad.

Codo a Codo inicial

El curso de Codo a Codo inicial es perfecto para aquellos que se están iniciando en el mundo de la programación y desean adquirir una base sólida en este campo.
Durante el curso, aprenderás los conceptos esenciales y las habilidades fundamentales necesarias para convertirte en un programador capacitado.
Al finalizar el curso contarás con los conocimientos y habilidades necesarias para dar tus primeros pasos como programador, sentando una base sólida para tu desarrollo profesional en el campo de la programación.

Modalidad

Clases online en vivo 2 veces por semana

Duración

20 semanas

Carga horaria

198 horas

Versión

2023.07.01

Herramientas y tecnologías que aprenderemos


Visual Studio Code


Python

IntelliJ IDEA

Java


GIT


GitHub

MySQL

phpMyAdmin

Temario del curso

MÓDULO I: Algoritmia esencial

Unidad 1: Introducción a la algoritmia

- Procesamiento de información: Concepto y flujo de procesamiento de la información en un sistema computacional.
- Datos: Definición y tipos de datos utilizados en programación.
- Algoritmos: Concepto de algoritmo y su importancia en la resolución de problemas. Características de los algoritmos eficientes.
- Programas: Definición de programa y su relación con los algoritmos. Diferencias entre algoritmo y programa.
- Componentes de un algoritmo: Entrada, procesamiento y salida de información. Explicación detallada de cada componente.
- Diagramas de flujo: Introducción a los diagramas de flujo como herramienta visual para representar algoritmos. Elementos y símbolos utilizados en los diagramas de flujo.

Unidad 2: Análisis y resolución de problemas

- Problemas: Definición de un problema y su importancia en el contexto de la programación. Identificación de los puntos principales de un problema.
- Análisis de problemas: Técnicas y enfoques para analizar problemas de manera sistemática. Identificación de requisitos, restricciones y objetivos del problema.
- Diseño de algoritmos: Utilización de algoritmos para resolver problemas planteados. Selección de estructuras de control y algoritmos adecuados para la resolución del problema.
- Implementación de algoritmos: Traducción de un algoritmo en un lenguaje de programación específico. Uso de estructuras de datos y variables para almacenar y manipular la información necesaria.
- Pruebas y depuración: Realización de pruebas exhaustivas para verificar la corrección del algoritmo. Identificación y solución de errores o fallos en el algoritmo.
- Optimización: Mejora de la eficiencia y rendimiento del algoritmo a través de técnicas de optimización.

Unidad 3: Sintaxis y uso de operadores

- Variables: Definición de una variable y su importancia en la programación. Asignación de valores a variables y manipulación de su contenido.
- Entrada y salida de valores: Uso de métodos de entrada para recibir valores del usuario. Uso de métodos de salida para mostrar resultados por pantalla.
- Hola Mundo: Introducción al programa "Hola Mundo" como una forma tradicional de comenzar en la programación.
- Condicionales: Uso de estructuras condicionales para tomar decisiones en el programa. Uso del condicional "if" para ejecutar diferentes bloques de código según una condición.

- Operadores lógicos: Introducción a los operadores lógicos (AND, OR, NOT) y su uso en las expresiones condicionales.
- Operadores relacionales: Uso de operadores relacionales (<, >, <=, >=, ==, !=) para comparar valores y evaluar condiciones.

Unidad 4: Estructuras secuenciales y repetitivas

- Estructuras secuenciales: Uso de estructuras secuenciales para ejecutar instrucciones en un orden específico. Secuencia de pasos lógicos para realizar una tarea.
- Ciclos con número determinado de iteraciones: Utilización de bucles for y while para ejecutar un bloque de código un número específico de veces.
- Ciclos con número indeterminado de iteraciones: Uso de bucles while para ejecutar un bloque de código hasta que se cumpla una condición de salida.
- Bucles finitos e infinitos: Comprender la diferencia entre bucles finitos (que tienen una condición de salida) y bucles infinitos (que se ejecutan sin una condición de salida).
- Control de bucles: Utilización de instrucciones de control (break, continue) para interrumpir o saltar iteraciones en un bucle.

MÓDULO II: Lenguaje de Programación

Unidad 5: Introducción al lenguaje de programación

- Historia del lenguaje: Contextualización sobre el origen y evolución del lenguaje de programación. Exploración de su relevancia en la industria y casos de uso comunes.
- Uso principal del lenguaje: Identificación de los principales casos de uso y aplicaciones del lenguaje de programación en el desarrollo de software y otras áreas.
- Descarga e instalación del IDE: Guía paso a paso para descargar e instalar el entorno de desarrollo integrado (IDE) necesario para programar en el lenguaje elegido.
- Tipos de datos: Introducción a los tipos de datos básicos del lenguaje, como enteros, flotantes, cadenas y booleanos. Uso de variables para almacenar y manipular estos datos.
- Condicionales: Ampliación de los conceptos de condicionales aprendidos previamente. Uso de estructuras condicionales más complejas, como if-else if-else, para realizar decisiones más elaboradas en el programa.

Unidad 6: Estructuras secuenciales y repetitivas dentro del lenguaje

- Estructuras de control: Ampliación de los conceptos de estructuras de control aprendidos anteriormente. Uso de estructuras como if-else, switch-case y estructuras condicionales anidadas para tomar decisiones en el programa.
- Creación de bucles con un número determinado de iteraciones: Utilización de la sintaxis específica del lenguaje para crear bucles con un número fijo de repeticiones. Uso de bucles for y while para ejecutar un bloque de código un número determinado de veces.
- Creación de bucles con un número indeterminado de iteraciones: Utilización de la sintaxis específica del lenguaje para crear bucles con un número variable de repeticiones. Uso de bucles while y do-while para ejecutar un bloque de código hasta que se cumpla una condición de salida.

Unidad 7: Colecciones

- Creación de Arrays: Introducción a los arrays como estructuras de datos que permiten almacenar múltiples elementos del mismo tipo en una secuencia ordenada. Exploración de la sintaxis para crear y declarar arrays en el lenguaje de programación.
- Listas: Uso de listas para almacenar y manipular colecciones de elementos. Aprendizaje de técnicas para agregar, modificar y eliminar elementos de una lista.
- Tuplas: Utilización de tuplas para almacenar datos de forma inmutable, es decir, que no se pueden modificar una vez creadas. Exploración de las ventajas y aplicaciones de las tuplas en programas.
- Diccionarios: Introducción a los diccionarios como estructuras de datos que permiten almacenar pares clave-valor. Uso de diccionarios para asociar información y realizar búsquedas eficientes en base a claves.
- Recorrido de la colección: Aprendizaje de técnicas para recorrer y acceder a los elementos de una colección, ya sea mediante bucles o métodos específicos del lenguaje.
- Incorporación, modificación y borrado de elementos: Práctica de técnicas para agregar, modificar y eliminar elementos dentro de las colecciones, utilizando métodos y operaciones propias del lenguaje.

Unidad 8: Funciones

- Funciones con parámetros de entrada: Aprendizaje de cómo definir y utilizar funciones que acepten parámetros de entrada. Exploración de la sintaxis para pasar valores a una función y utilizar esos valores dentro de la función.
- Devolución de datos hacia el programa principal: Introducción al concepto de devolución de datos o retorno de valores desde una función hacia el programa principal. Uso de la palabra reservada return para especificar qué valor debe ser devuelto.

- Procesamiento de la información dentro de la función: Práctica de cómo procesar la información dentro de una función, realizando operaciones y cálculos con los parámetros de entrada o variables locales.
- Funciones sin parámetros de entrada: Utilización de funciones que no requieren parámetros de entrada. Exploración de cómo definir y llamar a estas funciones.
- Palabra reservada Void: Explicación de la palabra reservada void y su significado en la definición de funciones. Uso de funciones void para realizar tareas sin retornar valores hacia el programa principal.

Unidad 9: Librerías

- Introducción a las librerías: Explicación del concepto de librerías en el lenguaje de programación y su importancia para la reutilización de código.
- Librerías más utilizadas: Presentación de las librerías más comunes y ampliamente utilizadas en el lenguaje de programación dictado en la cursada. Se explorarán ejemplos prácticos de uso de estas librerías y sus principales características.
- Importación de librerías: Aprendizaje de cómo importar y utilizar las funciones y recursos de una librería en un programa. Exploración de la sintaxis adecuada y las diferentes formas de importación.
- Uso de funciones y métodos de librerías: Práctica de la utilización de las funciones y métodos provistos por las librerías importadas. Se realizarán ejercicios y ejemplos para familiarizarse con las funcionalidades ofrecidas por cada librería.
- Exploración de librerías externas: Introducción a la exploración de librerías externas populares y ampliamente utilizadas en el ámbito del lenguaje de programación. Se brindarán recursos y recomendaciones para buscar, evaluar y utilizar librerías externas en los proyectos.

MÓDULO III: Base de Datos

Unidad 10: Introducción a las bases de datos relacionales

- Concepto y origen de las bases de datos: Explicación de qué son las bases de datos y cómo surgieron. Se abordará la importancia de las bases de datos en el almacenamiento y gestión de la información en aplicaciones y sistemas.
- Objetivos y funcionalidad de las bases de datos: Presentación de los principales objetivos y funcionalidades que brindan las bases de datos relacionales, como la organización eficiente de los datos, el acceso rápido a la información y la integridad de los datos.
- Arquitectura y modelos de bases de datos: Descripción de la arquitectura y los modelos utilizados en las bases de datos relacionales. Se explicarán los conceptos de tablas, filas, columnas y las relaciones entre ellas.
- Instalación de la base de datos: Guía paso a paso sobre la instalación de una base de datos relacional. Se mostrará cómo configurar el entorno de trabajo y realizar la instalación adecuada de la base de datos en el sistema.

- Configuración y gestión básica: Introducción a la configuración y gestión básica de la base de datos. Se abordarán temas como la creación de tablas, la inserción de datos y la realización de consultas básicas.

Unidad 11: Consultas SQL

- Creación de base de datos: Paso a paso para la creación de una base de datos utilizando SQL. Se abordarán los conceptos básicos, como la sintaxis y la estructura de una sentencia SQL para crear una base de datos.
- Creación de tablas: Explicación detallada de cómo crear tablas en una base de datos utilizando SQL. Se cubrirán aspectos como la definición de columnas, tipos de datos, restricciones y claves primarias.
- Inserción, modificación y eliminación de datos: Uso de sentencias SQL para insertar, actualizar y eliminar datos en una tabla. Se explorarán las cláusulas INSERT, UPDATE y DELETE, junto con ejemplos prácticos de su aplicación.
- Consultas y subconsultas: Introducción a las consultas en SQL y cómo utilizar la cláusula SELECT para recuperar datos de una o varias tablas. Se explorarán los operadores de comparación, los operadores lógicos y la cláusula WHERE para filtrar los resultados. También se abordarán las subconsultas y su uso en consultas más complejas.
- Cláusulas: Select, Insert, Update, Delete, Where: Explicación detallada de cada una de estas cláusulas y su aplicación en consultas SQL. Se mostrarán ejemplos prácticos de cómo utilizar estas cláusulas para manipular y recuperar datos de una base de datos.

Contenido clase a clase

Módulo Pseudocódigo

0. Presentación del curso

- Sobre Codo a Codo.
- Objetivo del curso y lineamientos de cursada.
- Conceptos básicos sobre Codo a Codo inicial.
- Herramientas a utilizar.

1. Instalación y presentación del software

- Conceptos básicos Pseint
- Variables: ¿qué son?
- Sintaxis básica en Pseint
- Ejemplos en pseudocódigo.

2. Introducción a la algoritmia

- Que es un algoritmo.
- Como resolver problemas a través de algoritmos.
- Que es un programa.
- Información: Entrada - Proceso - Salida
- Ejemplo de un algoritmo
- Diagrama de flujo

3. Sintaxis básica y uso de variables

- Booleanos
- Null.
- Condicional if
- Resolución de un algoritmo básico utilizando variables.

4. Operadores aritméticos y de asignación

- Definición de una variable.
- Asignación de su valor.
- Suma - resta- multiplicación - división
- Entrada y salida de valores por pantalla

5. Operadores relacionales

- Operadores <, <=, >, >=, ==, != en algoritmos.
- Ejemplos de comparaciones utilizando < y <=.
- Ejemplos de comparaciones utilizando > y >=.
- Ejemplos de comparaciones utilizando == y !=.
- Combinación de operadores relacionales en condiciones.
- Ejemplos prácticos de algoritmos con operadores relacionales.

6. Operadores lógicos

- Operador "and": su función y uso en algoritmos.
- Operador "or": su función y uso en algoritmos.
- Operador "not": su función y uso en algoritmos.
- Ejemplos de algoritmos que utilizan el operador "and".
- Ejemplos de algoritmos que utilizan el operador "or".
- Ejemplos de algoritmos que utilizan el operador "not".

7. Estructuras de control: estructuras secuenciales

- Condicionales
- Selección
- Diagrama de flujo de dicha estructura
- Ejemplo de estructura secuencial
- Resolución de algoritmo por medio de su uso

8. Estructuras de control: estructuras repetitivas

- Definición for
- Definición while
- Ejemplo de algoritmo sin estructura repetitiva.
- Ejemplo de algoritmo con estructura repetitiva

9. Síntesis lo aprendido

- Resolución de problemas por medio de la utilización de algoritmos
- Planteamiento de ejercicios para resolver la próxima clase

10. Resolución de problemas

- Resolución de los problemas planteados la clase anterior
- Repaso de conceptos
- Resolución de dudas de estudiantes

Módulo Lenguaje de programación

11. Lenguaje de programación

- ¿Qué es y para qué se usa Python/Java?
- Descarga e instalación del entorno de trabajo
- Conceptos generales.

12. Conceptos básicos del lenguaje

- Hola Mundo.
- Sintaxis básica
- Variables y cómo definirlas

13. Tipos de datos

- Revisión de los tipos de datos básicos del lenguaje de programación utilizado
- Integer
- Float
- Boolean
- Entrada y salida de información

14. Entrada y salida de datos

- Métodos de entrada de datos desde el usuario, como lectura de consola o interacción con una interfaz gráfica.
- Validación y manejo de errores en la entrada de datos.
- Métodos de salida de datos, como impresión en la consola o generación de archivos.
- Formateo y presentación de datos para su salida.
- Transformación de datos de un tipo a otro, como conversión de cadenas a enteros o viceversa.

15. Estructura secuencial en (Python/Java)

- Estructura secuencial de un programa por medio de estructuras secuenciales
- Utilización de los tipos de datos aprendidos en la clase previa

16. Estructura repetitiva en (Python/Java)

- While y For
- Conceptos generales
- Similitudes y diferencias
- Ejercicio básico

17. Comparación Pseint - (Python/Java)

- Resolución de los mismos ejercicios planteados en la primera parte del curso, ahora con el lenguaje de programación
- Análisis de las facilidades que nos da el IDE

18. Colecciones

- Listas
- Tuplas
- Diccionarios.
- Incorporación, eliminación, modificación de elementos

19. Funciones

- Funciones: ¿Qué son?
- Parámetros de entrada y salida
- Programación lineal vs funciones.
- Conceptos básicos de funciones en programación.

20. Funciones de tipo void

- Definición y características de las funciones void.
- Uso de funciones void para realizar tareas sin retornar valores
- Sintaxis y estructura de las funciones void.
- Declaración y definición de funciones void.

21. Funciones con parámetros

- Argumentos
- Parámetros
- Invocación de funciones void en diferentes contextos.
- Argumentos y argumentos predeterminados.
- Creación de funciones void para realizar operaciones matemáticas simples

22. Funciones return

- Organización y modularización de código utilizando funciones con retorno.
- Mejores prácticas para el uso de funciones con retorno.
- Creación de funciones con retorno para cálculos matemáticos.
- Implementación de funciones con retorno para procesamiento de datos.

23. Funciones recursivas

- Concepto de recursividad:
- Definición de la recursividad y su importancia en la programación.
- Comprender el concepto de llamadas recursivas y la pila de ejecución

24. Funciones avanzadas

- Resolución de problemas avanzados por medio del uso de funciones
- Ventajas de la programación estructurada en funciones

25. Manejo de archivos

- Tipos de archivos (texto, binario, JSON, XML, etc.).
- Lectura y escritura de archivos.
- Apertura, cierre y manipulación de archivos.

26. Librerías básicas del lenguaje

- Definición y función de las librerías en el lenguaje de programación.
- Importancia de las librerías para la reutilización de código.
- Exploración de librerías externas populares y ampliamente utilizadas en el lenguaje de programación

Módulo Base de datos

27. Base de datos - Principios básicos

- ¿Qué es una Base de datos?
- Bases de datos relacionales y no relacionales.
- Instalación de la base de datos

28. Base de datos - Consultas

- Inserción de datos -> Insert
- Consulta de datos -> Select
- Actualización y eliminación de datos -> Update and Delete

29. Base de datos - SubConsultas

- Subconsultas en la cláusula WHERE .
- Subconsultas en la cláusula HAVING
- Subconsultas en la cláusula FROM .

30. GIT

- Introducción a GIT y GitHub.
- Comandos básicos.
- Creación de repositorios y ramas.

31. Taller: Muestra de proyectos 1

- Muestra de proyectos

32. Taller: Muestra de proyectos 2 y Consultas

- Muestra de proyectos
- Consultas EFI (Examen Final Integrador)

33. Cierre de cursada

- Recomendaciones para el EFI.
- Retrospectiva: fortalezas y oportunidades de mejora.
- Próximos pasos. Después del curso, ¿qué puedo hacer? ¿Qué otras opciones ofrece Codo a Codo?
- Cierre de cursada.

Clase especial - Grabada - Manejo de Excepciones:

- Introducción a las excepciones y su importancia en el manejo de errores.
- Uso de bloques try-catch para capturar y manejar excepciones.
- Definición y uso de excepciones personalizadas.

Perfil profesional del egresado

Al finalizar el curso de Codo a Codo inicial podrás:

- Dominar los conceptos básicos de programación: Tendrás un sólido entendimiento de los conceptos fundamentales de programación, como algoritmos, variables, condicionales, bucles y estructuras de datos.
- Desarrollar habilidades en pseudocódigo y diagramas de flujo: Serás capaz de expresar algoritmos utilizando pseudocódigo y representar visualmente los flujos de control mediante diagramas de flujo.
- Familiarizarte con lenguajes de programación: Te sentirás cómodo trabajando con lenguajes de programación como Python o Java, y podrás utilizarlos para desarrollar aplicaciones y resolver problemas.
- Manipular datos y realizar operaciones de entrada/salida: Comprenderás cómo manejar diferentes tipos de datos, realizar operaciones aritméticas y utilizar métodos de entrada/salida para interactuar con el usuario.
- Dominar el uso de estructuras de control: Tendrás un buen dominio del uso de estructuras condicionales (if-else) y estructuras de repetición (for, while) para controlar el flujo de ejecución de un programa.
- Comprender los fundamentos de las bases de datos: Obtendrás una comprensión básica de las bases de datos, incluyendo la creación de tablas, inserción y consulta de datos utilizando SQL.
- Adquirir habilidades en el manejo de archivos: Serás capaz de leer y escribir archivos de diferentes tipos, como archivos de texto o archivos JSON, y manipular su contenido.
- Utilizar librerías y frameworks en tu desarrollo: Estarás familiarizado con el uso de librerías y frameworks básicos en el desarrollo de software, aprovechando la reutilización de código y mejorando tu productividad.
- Comprender los conceptos de control de versiones: Tendrás una comprensión básica de los conceptos de control de versiones y serás capaz de utilizar herramientas como Git y GitHub para administrar y colaborar en proyectos de programación.
- Desarrollar habilidades de resolución de problemas y pensamiento lógico: Adquirirás habilidades para abordar problemas de manera lógica y analítica, descomponiendo problemas complejos en tareas más pequeñas y aplicando algoritmos y estructuras de datos adecuados.
- Comunicarte y trabajar en equipo: Podrás comunicarte eficazmente con otros miembros del equipo de desarrollo de software, participar en la revisión de código y colaborar en proyectos grupales.
- Mantenerte en un aprendizaje continuo: Comprenderás la importancia del aprendizaje continuo en el campo de la programación y estarás motivado para mantenerte actualizado sobre las últimas tendencias y tecnologías.
- Con este perfil profesional, estarás preparado para integrarte en equipos de desarrollo de software, empresas de tecnología o emprender proyectos propios en el

ámbito de la programación. Además, estarás listo para continuar tu aprendizaje y desarrollo profesional en este campo en constante evolución

Proyectos a desarrollar durante la cursada

El proyecto se construye a lo largo de la cursada y sirve para validar tu aprendizaje, poniéndolo en práctica con un proyecto propio. Las pautas del proyecto se les informará al inicio de la cursada.
El proyecto será una excelente herramienta para enriquecer tu portfolio y así poder aumentar tus probabilidades de empleabilidad en el sector IT.

PROYECTO INTEGRADOR FINAL

- Diseño y desarrollo de una aplicación completa: Crearás una aplicación desde cero, abordando tanto el diseño como la implementación de todas sus funcionalidades.
- Caso de uso con entrada, procesamiento y salida de datos: Trabajarás en un caso de uso específico, diseñando la lógica para la entrada de datos, su procesamiento y la posterior salida de resultados.
- Temática personalizada según comisión: Podrás elegir una temática relevante a tu interés o problemática específica para desarrollar tu proyecto.
- Validación rigurosa del procesamiento de datos: Realizarás una validación exhaustiva del procesamiento de datos para garantizar la integridad y precisión de los resultados.
- Utilización de librerías o frameworks pertinentes: Aplicarás librerías o frameworks relevantes a la temática del proyecto para mejorar la eficiencia y calidad de la aplicación.
- Implementación de prácticas de seguridad y manejo de errores: Integrarás prácticas de seguridad y manejo adecuado de errores para asegurar la confiabilidad y robustez de la aplicación.
- Presentación y documentación del proyecto: Realizarás una presentación y documentación concisa que describirá las funcionalidades, proceso de desarrollo y decisiones tomadas durante el proyecto.

Requerimientos

Requerimientos para cursar

- Este curso no posee requerimientos previos para realizar su cursada.

Requisitos para obtener el diploma

- Asistir al 65% de las clases en vivo (clases sincrónicas).
- Acceder semanalmente al Aula Virtual.
- Realizar los ejercicios obligatorios semanales.
- Aprobar el Trabajo Práctico Obligatorio.
- Aprobar el EFI (Examen Final Integrador).