Guía del curso

Introducción

Descripción del curso

Este curso es una excelente base para aprender a programar y emerger en el ámbito del desarrollo de software.

Esta centrado en la enseñanza del diseño de algoritmos eficientes y sus respectivos programas como conceptos fundamentales de la programación.

Se emplean prácticas como ejemplos para aprender a diseñar algoritmos y desarrollar sus correspondientes programas en un lenguaje de programación.

Se plantean ejercicios para prácticar y resolver problemas prácticos y un proyecto final donde se propone un problema típico de la programación de modo que en su solución aplique todo lo adquirido en el curso.

Objetivos y expectativas del curso

Adquier las habilidades esenciales de programación y herramientas necesarias para resolver problemas y escribir código de manera efectiva.

Formato del curso

Se manejará material didáctico en diapositivas y un manual de prácticas. Las diapositvas irán indicando los temas a ir aprendiendo.

La base de la programación es la lógica, con ella podremos elaborar algoritmos y de estos se parte para desarrollar los programas.

El curso como lo indica la descripción y el objtivo es aprender a programar y para aprender no hay otra cosa que “haciendo”, esto es, programando. Vaya la analogía, para aprender a nadar no basta con ver como nadan ni leyendo manuales, se aprende “nadando”.

El curso esta planteado para impartirse mediante técnicas expositivas, de dialogo/discusión y demostrativas.

El manual será utilizado como guía para seguir el avance del aprendizaje, se inicia desde lo básico y se irá complicando cada vez más. Para ello se tienen ejempos y prácticas para que usted pueda comprobar su funcionamiento. Los ejercicios se plantean para que usted

aumente su habilidad en los temas tratados. Se pretende que se desarrollen durante a sesión presencial y si por cuestiones de tiempo se terminen como actividades exclase.

Importancia de la programación en el mundo actual

La importancia de la programación en el mundo actual es un tema fundamental para motivar y tener una perspectiva clara de por qué la programación es relevante en la sociedad actual. Aquí hay algunas ideas:

- **Introducción a la programación:** Qué es la programación y cómo se relaciona con la creación de software y aplicaciones.
- **Ubicuidad de la tecnología:** Cómo **C**la tecnología está en todas partes en nuestra vida cotidiana. Desde teléfonos inteligentes hasta electrodomésticos, automóviles y sistemas de entretenimiento, la programación es esencial en todos estos dispositivos.
- **Automatización y eficiencia:** Cómo la programación se utiliza para automatizar tareas, lo que ahorra tiempo y recursos. Ejemplos concretos, como la automatización de procesos en empresas y la producción de bienes.
- **Innovación y creatividad:** Cómo la programación es una herramienta clave para la innovación. Empresas y emprendedores utilizan la programación para desarrollar nuevas ideas y soluciones para diversos problemas.
- **Oportunidades laborales:** La creciente demanda de profesionales de la programación en el mercado laboral. Existen salarios competitivos y la posibilidad de trabajar en una variedad de industrias.
- **Educación y aprendizaje:** Cómo aprender a programar no solo es beneficioso para las futuras oportunidades laborales, sino que también fomenta el pensamiento lógico, la resolución de problemas y la creatividad.
- **Accesibilidad:** No es exclusiva para expertos en informática. Cualquier persona, sin importar su edad o formación previa, puede aprender a programar.
- **Ejemplos de aplicaciones reales:** Ejemplos concretos de cómo la programación ha tenido un impacto en campos como la medicina, la ciencia, la industria del entretenimiento y la investigación.
- **Tendencias futuras:** Tendencias emergentes en el campo de la programación, como la inteligencia artificial, la robótica, la ciberseguridad y la programación web.

- **Desafíos y ética:** Junto con los beneficios, también hay desafíos y cuestiones éticas en la programación, como la privacidad de los datos, la seguridad cibernética y la toma de decisiones éticas en la inteligencia artificial.

Módulo 0. Puesta a punto de la computadora

Para llevar a cabo el curso de manera adecuada se requerirá que su equipo de cómputo este instalado los siguientes programas/aplicaciones.

- Visual Studio Code
- Python
- HPCompiler

Para ello remitasé a las respectivas guías de instalación.

A fin de saber si nuestra computadora esta punto es crear un directorio (carpeta) exprofeso para el curso, digamos *Curso Introduccion a la Programación con Python*. Poner este directorio en nuestro Visual Studio Code y crear un archivo, digamos *Saludo.py* y en él escribir el siguiente texto *print ("Hola Robert")* , se guarda y luego en la consola de comando (Terminal) del sistema operativo ejecutar el siguiente comando *python3 Saludo.py*. Debe aparecer la leyenda Hola Robert en la consola. Python3 es por la version, pero puede omitirse y quedar simplemente *python Saludo.py*.

Práctica 1.1. Poner a punto su computadora.

Módulo 1 Lógica de Programación

1.1 Definición de la Lógica de Programación

Introducción

Problema: Es una situación que requiere solución por parte de agentes externos.
Algoritmo: Es una serie de pasos que resuelven un problema.
Programa: Es una serie de pasos que resuelven un problema utilizando un lenguaje de programación.

- Tipos de problemas:
  - Sociales - Alcoholismo
  - Económicos – Pobreza
  - Culturales – Descriminación
  - Salud – Obesidad
  - Computacionales - Ordenamiento

- Tipos de algoritmos:
  - Textuales
  - Pseudocódigo
  - Diagramas de flujo de datos (DFD)

- Tipos de lenguajes de programación
  - Python
  - PHP
  - Java

1.1.1 Definición de la lógica de programación y su relevancia

La lógica de programación se refiere a la capacidad de pensar de manera lógica y diseñar algoritmos para resolver problemas de manera efectiva utilizando un lenguaje de programación. Implica la creación de secuencias de instrucciones lógicas y coherentes que una computadora puede seguir para realizar una tarea específica. La lógica de programación es esencial para transformar un problema en un conjunto de pasos que una computadora puede entender y ejecutar.

Ejemplo 1.1. Algoritmo de Suma

Un ejemplo simple de lógica de programación es un algoritmo para sumar dos números. Un algoritmo es un conjunto de pasos lógicos y ordenados para realizar una tarea. El siguiente pseudocódigo describe el proceso de suma:

Inicio
Pedir al usuario que ingrese el primer número (A)
Pedir al usuario que ingrese el segundo número (B)
Suma = A + B
Mostrar "La suma es:", Suma
Fin

Este algoritmo utiliza instrucciones para (en forma textual) "solicitar datos al usuario, realiza una operación matemática (suma) y muestra el resultado". La lógica detrás del algoritmo es fundamental para que la computadora realice la operación de manera efectiva.

Ejercicio 1.1. Tomando en cuenta el ejemplo 1.1, ¿tiene sentido si cambíamos el orden de solicitud de datos?, digamos primero solicitar el segundo dato y realizar otras operaciones aritméticas como la resta, multiplicación, división y módulo. Argumente con ejemplos de datos.

1.1.2 Pensamiento algorítmico como enfoque para resolver problemas

El pensamiento algorítmico es una habilidad fundamental en programación y resolución de problemas. Implica la capacidad de descomponer un problema complejo en pasos más pequeños y lógicos, y luego diseñar un algoritmo para resolverlo. Este enfoque es esencial para la programación efectiva y se aplica a una amplia gama de campos, desde la ciencia de la computación hasta la resolución de problemas en la vida cotidiana.

Importancia del Pensamiento Algorítmico

- **Simplificación de Problemas:** El pensamiento algorítmico permite abordar problemas complejos al dividirlos en tareas más pequeñas y manejables. Esto facilita la resolución de problemas al abordar cada componente por separado.

- **Claridad y Estructura:** Al diseñar algoritmos, es necesario establecer una secuencia lógica de pasos. Esto fomenta la claridad y la estructura en la resolución de problemas, lo que facilita la comprensión y la colaboración.

- **Eficiencia:** Al desarrollar algoritmos, se busca optimizar la eficiencia en la solución de problemas. El pensamiento algorítmico implica considerar cómo realizar tareas de manera más rápida y con menos recursos.

Ejemplo 1.2. Imagina que deseas crear un algoritmo para hacer una taza de té. Aunque parece simple, el proceso puede descomponerse en una serie de pasos lógicos (pensamiento algorítmico):

Inicio
Hervir agua.
Colocar una bolsita de té en una taza.
Verter el agua caliente en la taza.
Dejar reposar durante unos minutos.
Retirar la bolsita de té.
Fin

Cada uno de estos pasos es una parte del algoritmo para hacer una taza de té. El pensamiento algorítmico implica definir claramente estos pasos y su secuencia para lograr el resultado deseado.

Ejercicio 1.2. Plantee otras situaciones parecidas a la de hacer una taza de te, donde se resuelva de manera lógica. Escrita el algoritmo en pseudocódigo su solución de cada una de ellas.

Ejercicio 1.3. De acuerdo a cada situación de los ejercicios anteriores, qué puede cambiar de orden sin afectar la solución y cúales no puede cambiar?

Ejercicio 1.4. ¿Tiene sentido pensar lógicamente (tener pensamiento lógico o pensamiento algorítmico) para resolver un problema? Argumente.

Aplicación en programación

El pensamiento algorítmico es esencial en la programación, donde se utiliza para diseñar algoritmos que resuelvan problemas informáticos. Los programadores descomponen tareas en pasos más pequeños, utilizan estructuras de control y operaciones lógicas para crear programas eficientes.

El pensamiento algorítmico facilita la simplificación de problemas, la optimización de soluciones y la creación de algoritmos efectivos. Fomenta la claridad, la estructura y la eficiencia en la toma de decisiones y resolución de problemas en el mundo de la programación.

Práctica 1.2. Tomando en cuenta el ejemplo 1.2, desarrollar un programa en Python que muestre los pasos a seguir para preparar una taza de té.

Práctica 1.3. Tomando en cuenta el ejemplo 1.1, desarrollar un programa en Python que sume los dos operandos.

Ejercicio 1.5. Desarrolle un programa Python que muestre las operaciones suma, resta, multiplicación y división indicadas del ejercicio 1.1.

Ejercicio 1.6. Desarrolle un programa Python para cada una de las situaciones que mostró en el ejercicio 1.2.

1.2 Álgebra de Boole y Operaciones Lógicas

1.2.1 Conceptos básicos del álgebra de Boole

El álgebra de Boole, también conocida como álgebra booleana, es un sistema matemático que se utiliza en la lógica de programación para manipular valores lógicos. Fue desarrollada por el matemático George Boole y se basa en la teoría de conjuntos y la lógica formal. Los conceptos básicos del álgebra de Boole son fundamentales para la toma de decisiones en programación y para el diseño de algoritmos.

En el álgebra de Boole, trabajamos con dos valores lógicos fundamentales:

- Verdadero (representado como 1 o "True").
- Falso (representado como 0 o "False").

Estos valores lógicos son la base de todas las operaciones en álgebra de Boole.

1.2.2 Operaciones lógicas (AND, OR, NOT) y su aplicación en programación

El álgebra de Boole incluye varias operaciones lógicas básicas que se aplican a valores lógicos. Las operaciones más comunes son:

AND (Y lógico): La operación AND devuelve verdadero (1) si ambos operandos son verdaderos, de lo contrario, devuelve falso (0). Su símbolo es "∧".

OR (O lógico): La operación OR devuelve verdadero (1) si al menos uno de los operandos es verdadero, de lo contrario, devuelve falso (0). Su símbolo es "∨".

NOT (NO lógico): La operación NOT invierte el valor lógico de su operando. Si el operando es verdadero, NOT lo convierte en falso, y viceversa. Su símbolo es "¬".

Tablas de Verdad

Las tablas de verdad son una representación de las operaciones lógicas en álgebra de Boole. Muestran todos los posibles valores de entrada y los resultados correspondientes de una operación. Por ejemplo, aquí está la tabla de verdad para la operación AND:

A	B	A ∧ B
0	0	0
0	1	0
1	0	0

1 | 1 | 1

Esta tabla muestra todos los posibles valores de A y B y el resultado de la operación AND.

En programación, el álgebra de Boole se utiliza para tomar decisiones y controlar el flujo de un programa. Por ejemplo, en una estructura condicional "if", se utiliza la operación AND para evaluar múltiples condiciones antes de ejecutar una acción. El álgebra de Boole es esencial para la creación de lógica en algoritmos y programas. Comprender estos conceptos es fundamental para cualquier programador.

Ejercicio 1.7. Realizar las tablas de verdad para la operación OR y otra para la NOT.

Ejercicio 1.8. Existe otra operación booleana conocida como XOR (OR exclusivo) el cual devuelve falso si ambos operandos son verdaderos o ambos son falso. Realizar la tabla de verdad para ésta operación.

1.2.3 Uso de Tablas de Verdad para Evaluar Expresiones Lógicas

Las tablas de verdad son una herramienta esencial en la lógica de programación y el álgebra de Boole. Permiten analizar y evaluar expresiones lógicas, determinando los valores lógicos resultantes de todas las posibles combinaciones de sus componentes. Las tablas de verdad son fundamentales para comprender y verificar la lógica detrás de las expresiones y son ampliamente utilizadas en programación y diseño de circuitos.

Estructura de una Tabla de Verdad

Una tabla de verdad está compuesta por tres partes principales:

Columnas para Variables Lógicas: Cada variable lógica en la expresión se representa en una columna separada. Las variables pueden tener dos valores: verdadero (1) o falso (0).
Columnas para Operaciones Lógicas: Si la expresión incluye operaciones lógicas como AND, OR o NOT, se incluyen columnas adicionales para representar los resultados de esas operaciones.
Columna para Resultado Final: La última columna muestra el resultado de la expresión completa en función de los valores de entrada.

Ejemplo 1.3. Se desea determinar la tabla de verdad para evaluar la expresión lógica: (A AND B) OR (NOT C). Se muestran todas las combinaciones posibles de valores para A, B y C:

```
A | B | C | (A AND B) OR (NOT C)
-------------------------------------------
0 | 0 | 0 |      1
0 | 0 | 1 |      0
0 | 1 | 0 |      1
0 | 1 | 1 |      0
1 | 0 | 0 |      1
1 | 0 | 1 |      0
1 | 1 | 0 |      1
1 | 1 | 1 |      1
```

este es el resultado final primero se hace and A y B luego un NOT C

En esta tabla, evaluamos la expresión para todas las combinaciones posibles de valores de A, B y C. La columna final muestra el resultado de la expresión completa en función de estos valores.

Aplicación en Programación

En programación, las tablas de verdad son útiles para verificar y depurar expresiones lógicas en estructuras condicionales, bucles y otros contextos. Ayudan a comprender cómo se comportará un programa en función de diferentes entradas lógicas y a identificar posibles errores lógicos.

El uso de tablas de verdad es una habilidad fundamental en programación y diseño de circuitos. Permite evaluar expresiones lógicas de manera sistemática y garantizar que un programa se comporte de la manera esperada en diferentes situaciones. Comprender y utilizar tablas de verdad es esencial para cualquier programador que trabaje con lógica y toma de decisiones en programación.

Ejercicio 1.9. Para las siguientes expresiones, crea su tabla de verdad y evalúa su valor lógico para todas las combinaciones posibles de valores:

1. (A AND B) OR (NOT A AND C)
2. A AND (B OR NOT A) AND C
3. (A OR B) AND (C AND NOT D)
4. (A OR B) AND (NOT C OR NOT D)

Práctica 1.4. Desarrollar el algoritmo y programa en Python para mostrar el funcionamiento de un caso del ejemplo 1.3. Posteriormente, modifique la expresión para evaluar la primer evaluación del ejercicio 1.9.

Los operadores lógicos tienen prioridades, el operador NOT es de mayor prioridad, le sigue el operador AND y el operador NOT es el de menor prioridad, por lo que daría el mismo resultado las expresiones: (A AND B) OR (NOT C) y A AND B OR NOT C.

Ejercicio 1.10. Plantea mediante una tabla de verdad todas las posibilidades que le pudiera pasar a un antro si tiene ciertas reglas de acceso, las cuales son: Si son mayores de edad (presetan INE por si hubiese duda) y son pareja (H y M) o bien si son mujeres solas. En caso de ser menores, a menos que estén en una lista especial (permiso otorgado por familiar o una institución).

Práctica 1.5. Desarrollar del algoritmo y programa en Python para mostrar el funcionamiento del ejercicio 1.10 para un caso.