Ingeniería en Administración 4.0

Segundo Tetramestre

Lenguajes de Programación I

Maestro: Fabian Felipe Cruz

Juan Paulo González Barboza

Examen Parcial

1. Introducción al Lenguaje y a su Entorno de Desarrollo

Qué es un lenguaje de programación y cuál es su importancia en el desarrollo de software?

R. Un lenguaje de programación es un conjunto de reglas y símbolos que permiten a los programadores crear programas para computadoras. Son herramientas fundamentales para el desarrollo de software, ya que facilitan la comunicación entre los humanos y las máquinas.

Los lenguajes de programación son importantes porque:

- **Permiten definir el comportamiento de los dispositivos de una computadora.**
- **Permiten especificar qué datos debe procesar un software.**
- **Permiten definir cómo deben almacenarse y transmitirse los datos.**
- **Permiten definir las acciones que debe tomar un software en diferentes circunstancias.**
- **Permiten resolver problemas y automatizar tareas.**

Menciona y compara tres entornos de desarrollo integrados (IDEs) utilizados en la actualidad.

R. Algunos ejemplos de entornos de desarrollo integrados (IDE) son:

- 

  Eclipse

Un IDE gratuito y de código abierto que es popular para el desarrollo de Java

- **NetBeans**

  Un IDE gratuito y de código abierto que es popular para el desarrollo de Java y que también admite otros lenguajes

- **Visual Studio**

  Un IDE popular que admite muchos lenguajes y se puede usar para desarrollar aplicaciones locales y web

2. Estructura de un Programa

Explica la estructura general de un programa en el lenguaje que estás estudiando

R. Existen dos partes o bloques que componen un programa: Bloque de declaraciones: en este se detallan todos los datos que utiliza el programa (constantes, variables, archivos, etc). Bloque de instrucciones: conjunto de acciones u operaciones que se han de llevar a cabo para conseguir los resultados esperados.

¿Por qué es importante seguir una estructura clara en la programación?

R. Las estructuras de datos son útiles porque nos permiten tener una batería de herramientas para solucionar ciertos tipos de problemas. Además, nos permiten hacer un software más eficiente optimizando recursos, algo muy útil para IoT y para los entornos que trabajan con Big Data.

3. Identificadores (Variables y Constantes)

Explica la diferencia entre una variable y una constante.

R. En programación, la diferencia entre una variable y una constante es que el valor de una variable puede cambiar, mientras que el valor de una constante no.

Explicación

- **Una variable es un contenedor de datos que puede cambiar su valor durante la ejecución de un programa.**
- **Una constante es un contenedor de datos con un valor predefinido que no puede cambiar durante la ejecución de un programa.**

Las variables y las constantes son útiles para almacenar datos que se usarán en algoritmos.

Las variables se almacenan en la memoria RAM de la computadora. A cada variable se le asigna un nombre descriptivo o un identificador.

Las constantes se almacenan en un área reservada en la memoria principal del ordenador.

Proporciona ejemplos de cómo se declaran y utilizan en un lenguaje de programación.

R. Para declarar y utilizar variables en un lenguaje de programación, se debe especificar el tipo de dato y el nombre de la variable. El nombre debe ser descriptivo y único dentro del programa.

A continuación, se presentan algunos ejemplos de declaración de variables en diferentes lenguajes de programación:

- **Java**

  La sintaxis para declarar una variable es `tipo nombreDeVariable;` Por ejemplo, `int number;` declara una variable llamada `number` de tipo entero.

- **VBA**

  Se pueden declarar variables como Booleano, Byte, Entero, Long, Moneda, Single, Double, Fecha, Cadena, Objeto o Variant.

- **C++**
  En versiones anteriores, se usaba la palabra clave `typedef` para declarar un nuevo nombre que es un alias para otro nombre. En C++ moderno, se prefiere la palabra clave `using`.

4. Tipos de Datos

¿Qué son los tipos de datos y por qué son importantes en la programación?

R. Los tipos de datos son una forma de clasificar y organizar la información en un programa de computadora. Son importantes porque permiten a los programadores trabajar con datos de manera eficiente y precisa.

Los tipos de datos determinan:

- **Qué valores puede almacenar una variable**
- **Qué operaciones se pueden realizar sobre los valores**
- **Cómo se representan los valores internamente en el computador**

Los tipos de datos son fundamentales para el desarrollo de software y la programación. Su uso adecuado ayuda a prevenir errores y a que el código sea más fácil de leer y mantener.

Clasifica los tipos de datos más comunes y da un ejemplo de cada uno.

R. Los tipos de datos más comunes se pueden clasificar en simples y compuestos, y en numéricos, textuales, multimedia, entre otros.

Tipos de datos simples

- **Enteros: Números sin decimales, como 50, 70 o 300**
- **Flotantes: Números con decimales, como 5.3, 4.7 o 8.24**
- **Caracteres: Representan un solo carácter**
- **Booleanos: Representan valores de verdad, como True o False**

Tipos de datos compuestos

- **Arreglos: Contienen varios valores al mismo tiempo**
- **Cadenas de caracteres: Representan secuencias de caracteres, como texto**
- **Registros: Contienen varios componentes, que pueden ser datos simples o estructurados**
- **Conjuntos: Contienen varios componentes, que pueden ser datos simples o estructurados**

Otros tipos de datos

- **Multimedia, como JPEG, MPEG o WAV**
- **Estructurados, como XML o base de datos MySQL**
- **Código de software, como Java o C**
- **Específicos de un software, como Mesh o 3D CAD**

5. Operadores Aritméticos, Lógicos y Relacionales

Explica la diferencia entre operadores aritméticos, lógicos y relacionales.

R. Los operadores aritméticos realizan operaciones matemáticas, los operadores lógicos combinan pruebas relacionales, y los operadores relacionales comparan valores.

Operadores aritméticos

- **Realizan operaciones matemáticas**
- **En una expresión con operadores aritméticos y relacionales, se realizan primero las operaciones aritméticas**

Operadores lógicos

- **Combinan pruebas relacionales**
- **Devuelven un valor lógico, verdadero (true) o falso (false)**
- **También se llaman operadores Booleanos**
- **Algunos ejemplos de operadores lógicos son AND, OR y NOT**

Operadores relacionales

- **Comparan valores numéricos, de serie de caracteres o lógicos**
- **Determinan si una relación es verdadera o falsa**
- **El resultado de la comparación se puede utilizar para tomar decisiones en el flujo del programa**
- **Algunos ejemplos de operadores relacionales son < (menor que), > (mayor que), <= (menor o igual que) y >= (mayor o igual que)**

Proporciona ejemplos prácticos de su uso

ARITMÉTICOS

Operador	Nombre	Ejemplo	Significado
^	Exponenciación	x^y	x elevado a y
*	Multiplicacion	x*y	x multiplicado por y
/	División	x/y	x dividido entre y
%	Modulo o resto	x%y	resto de la división x/y
+	Suma	x+y	x más y
–	Resta	x-y	x menos y

RELACIONALES

Operador	Nombre	Ejemplo	Significado
<	Menor que	x<y	x menor que y
>	Mayor que	x>y	x mayor que y
==	Igual a	x==y	x igual a y
!=	Distinto a	x!=y	x distinto de y

<=	Menor o igual que	x<=y	x menor o igual a y
>=	Mayor o igual que	x>=y	x mayor o igual a y

LÓGICOS

Operador	Nombre	Ejemplo	Significado
&&	AND	(x<y)&&(x<z)	‘x menor que y’ Y ‘x menor que z’
||	OR	(x<y)||(x<z)	‘x menor que y’ O ‘x menor que z’
!	NOT	!(x<y)	x NO es menor que y

6. Programación Estructurada

¿En qué consiste la programación estructurada y cuáles son sus principios fundamentales?

R. La programación estructurada es una técnica de programación que organiza el código en estructuras lógicas, con el objetivo de mejorar la calidad y el tiempo de desarrollo de un programa.

Sus principios fundamentales son:

- **Utilizar estructuras básicas como secuencia, selección e iteración**
- **Evitar la transferencia incondicional (GOTO)**
- **Utilizar subrutinas o funciones**

La programación estructurada se basa en el teorema del programa estructurado, que establece que cualquier programa puede ser escrito utilizando solo las tres estructuras de control básicas.

Las ventajas de la programación estructurada son:

- **Los programas son más sencillos de entender**
- **La fase de prueba y depuración es más sencilla**
- **El mantenimiento de los programas es más económico**
- **Los programas son más rápidos de crear**

Explica cómo se relaciona con el concepto de funciones o procedimientos en la programación.

R. La programación estructurada se basa en el uso de funciones o procedimientos para organizar el código en estructuras lógicas.

Explicación

- **La programación estructurada es un paradigma de programación que se centra en el uso de funciones o procedimientos para organizar el código.**
- **Las funciones son secciones de un programa que calculan un valor de manera independiente al resto del programa.**
- **La programación estructurada se basa en tres estructuras de control: secuencia, selección e iteración.**
- **La programación estructurada se utiliza para mejorar la claridad, calidad y tiempo de desarrollo de un programa.**
- **La programación estructurada se utiliza en lenguajes como Java, C, Python y C++.**

La programación estructurada se relaciona con el concepto de funciones o procedimientos en la programación de la siguiente manera:

- **La programación estructurada se basa en el uso de funciones o procedimientos para organizar el código.**
- **Las funciones son secciones de un programa que calculan un valor de manera independiente al resto del programa.**
- **Los procedimientos, también conocidos como rutinas, subrutinas o funciones, consisten simplemente en una serie de pasos computacionales que se deben llevar a cabo.**

7. Programación Orientada a Objetos

¿Qué es la programación orientada a objetos (POO) y cómo se diferencia de la programación estructurada?

R. La programación orientada a objetos (POO) y la programación estructurada son paradigmas de programación que se diferencian en el enfoque y en los elementos que utilizan.

La POO se centra en los datos y los objetos, mientras que la programación estructurada se centra en los procedimientos y las estructuras de datos.

Explica los conceptos de clase y objeto con un ejemplo.

R. En programación, una clase es un modelo o plantilla que define las características de un conjunto de objetos, mientras que un objeto es una instancia de esa clase.

Por ejemplo, una clase puede ser la plantilla de una silla, y un objeto sería una silla concreta.

Explicación

- **Una clase es un modelo que define las características y el comportamiento de un conjunto de objetos.**
- **Un objeto es una entidad que contiene un estado y un comportamiento.**
- **La clase es la plantilla, mientras que el objeto es la manifestación concreta de esa plantilla.**

Para crear un objeto, se utiliza la definición de una clase. Por ejemplo, si se define una clase llamada "clase silla", se pueden crear múltiples objetos (sillas) a partir de esa definición.

Algunos ejemplos de clases y objetos son:

- **Persona**

  La clase Persona define los atributos de una persona, como su nombre, edad y estatura. Los objetos de la clase Persona son personas específicas, como tú, tu mamá, el presidente de tu país, etc.

- **Automóvil**

  La clase Automóvil define los atributos de un automóvil, como su peso y color. Los objetos de la clase Automóvil son automóviles específicos, como un coche deportivo o un coche todo terreno.

- **Libro**

  La clase Libro define los atributos de un libro, como su título, autor, ISBN y si está prestado. Los objetos de la clase Libro son libros específicos, como el libro 1984.

- **Bicicleta**
  La clase Bicicleta define los atributos de una bicicleta, como su color, velocidad y marcha. Los objetos de la clase Bicicleta son bicicletas específicas, como tu bicicleta.