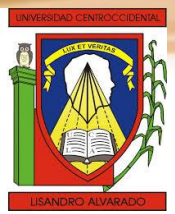


Unidad II: El Paradigma Orientado a Objeto.
Tema 4: Lenguaje Formal

Coordinación de Introducción a la Computación

Contenido

1. **Paradigmas de Programación.**
   - ✓ Programación Imperativa
   - ✓ Programación Lógica
   - ✓ Programación Funcional
   - ✓ Programación Declarativa
   - ✓ Programación Estructurada
   - ✓ Programación Dirigida por Eventos
   - ✓ Programación Orientada a Aspectos
   - ✓ Programación Orientada a Objetos
2. **Paradigma Orientado a Objeto (POO)**
   - ➢ **Definición y Ventajas.**
   - ➢ **Características.**
   - ➢ **Elementos básicos de la POO:**
     - ✓ Clase.
     - ✓ Objeto.
     - ✓ Atributo.
     - ✓ Método.
   - ➢ **Propiedades:**
     - ✓ Abstracción
     - ✓ Encapsulamiento/Ocultamiento.
     - ✓ Modularidad.
3. **Clases, Objetos y sus interrelaciones dentro del mundo real.**
4. **Lenguaje Formal**
   - ✓ Palabras claves de uso frecuente en lenguaje formal.
   - ✓ Elementos propios del lenguaje formal usado
   - ✓ Constructor
   - ✓ **Funciones**
     - ✓ Elementos de una función
     - ✓ Funciones que no retornan valor. Estructura básica
     - ✓ Funciones que retornan valor. Estructura básica.
   - ✓ **Parámetros**
     - ✓ Definición
     - ✓ Estructura básica
     - ✓ Parámetros por referencia
     - ✓ Parámetros por valor
   - ✓ **Métodos**
     - ✓ Sintaxis
   - ✓ **Diferencia entre función y método**
   - ✓ **Forma de escribir:**
     - ✓ Identificadores
     - ✓ Comentario

4. Lenguaje Formal.

El lenguaje formal a usar en esta asignatura para este lapso académico es C++

Palabras claves de uso frecuente en C++

Pseudolenguaje	Lenguaje C++
inicio	{
fin	}
clase	class
privado	private
público	public
si	if
sino	else
y	and
retornar	return
escribir	cout<<
leer	cin>>
principal()	main()

4. Lenguaje Formal.

Elementos propios de C++

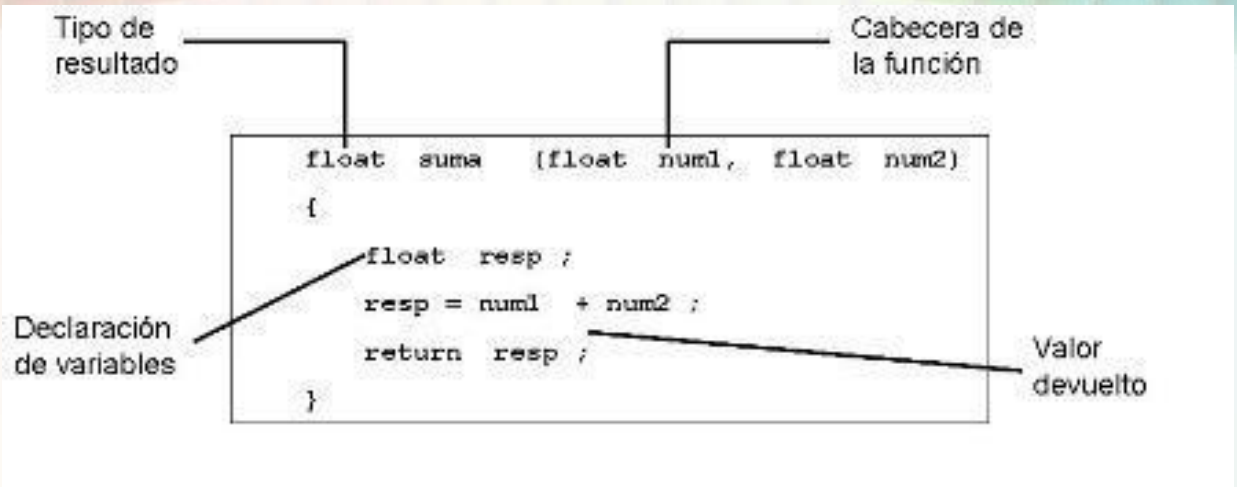
- Cada sentencia en C++ debe terminar con punto y coma (;)
- Sólo en la definición de la estructura de la clase luego de } debe ir obligatoriamente punto y coma (;)
- La relación de pertenencia entre la clase y la función se establece mediante el uso de dos puntos seguidos (::)
- Donde se utilice parámetros de entrada ↓ no se indica ningún valor, en tanto que los parámetros de salida ↑se indica con &

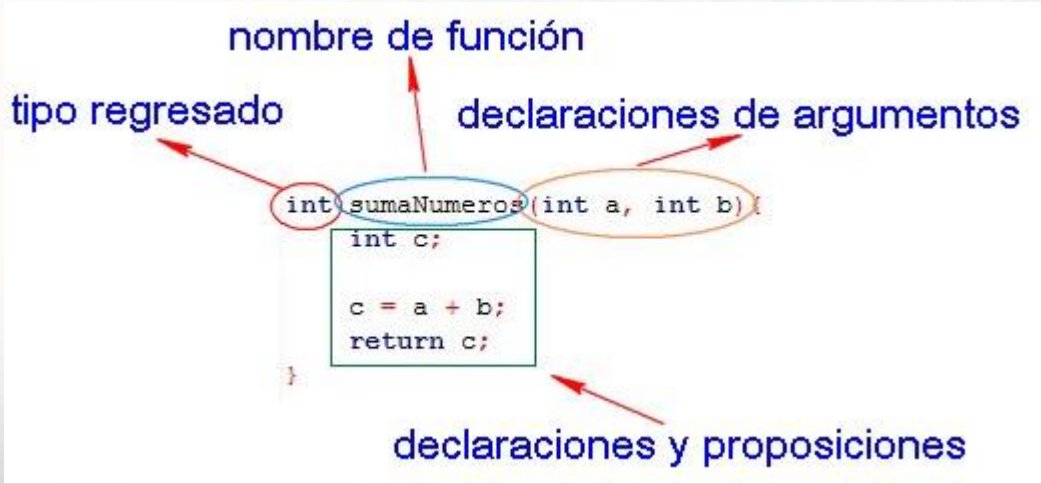
Constructor

Es un método que se ejecuta automáticamente cuando se crea un objeto de una clase, tiene el mismo nombre que la propia clase. Cuando se define un constructor no se puede especificar un valor de retorno

Funciones

Elementos de una función:





Funciones

Tipos de Funciones:

- **La que no retorna un valor.** Normalmente se usa para modificar y/o actualizar los atributos del objeto y tiene la siguiente estructura:

<identificador de función> <(Parámetros)>

- **La que retorna un valor**. Normalmente se usa para realizar cálculos utilizando los atributos de la clase y devolver un resultado, o devuelve el valor de un atributo. Tiene la siguiente estructura:

<tipo> <identificador de función> <(Parámetros)>

NOTA:

Si un método no retorna valor se deja en blanco el tipo de retorno.

Si un método no necesita parámetros se escriben sólo los paréntesis

Parámetros

Un parámetro o argumento es una variable que puede ser recibida por una función. Son usados por la función para determinar su comportamiento en tiempo de ejecución

Su estructura básica es:

<tipo> <tipo parámetro> <Identificador>

Donde:

tipo : Indica el tipo de dato que se pasa (entero, real, lógico, alfanum)

tipo parámetro: Por referencia (↑) o por valor (↓)

Un parámetro por referencia (↑) es aquel en el cual la variable mantiene sus cambios o actualizaciones cuando finaliza la función que los recibe.

Ejemplo : Interfaces de Entrada (IE)

Un parámetro por valor (↓) es aquel en el cual la variable no mantiene sus cambios o actualizaciones cuando finaliza la función que los recibe.

Ejemplo: métodos set, Interfaces de Salida (IS)

Métodos
Sintaxis

Método que retorna un valor
(Métodos que realizan operaciones matemáticas o lógicas)

<tipo**>** ClaseIdentificador metodo2 (parámetros)
inicio
  <tipo> varLocal
   //Cuerpo del método (operación matemática o lógica)
   retornar varLocal
fin

Nota: varLocal debe ser del mismo tipo que se establece en la cabecera de la función

Funciones y Métodos

Función

- Es un módulo en el que se divide un programa o sistema.
- Resuelve una tarea específica.

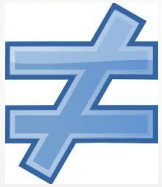
Elementos de una función

- Nombre
- Tipo de dato de retorno
- Lista de parámetros o argumentos (pueden ser cero o más) que la función debe recibir para realizar su tarea.
- El código o instrucciones de procesamiento que no es más que las sentencias que debe ejecutar la función

Método

- Un método es una función que pertenece a una clase
- Todo método es una función, pero no todas las funciones son métodos.
- Los métodos se denominan funciones miembro.

Método  **Función**

Función  **Método**

Forma de escribir:

Identificadores

variables:

float promedio

int numVentas

atributos:

float suedo

int numHijos

string nombre

métodos:

float calcularSueldo

int calcularPromedio

string determinarNombreVehiculo

constantes:

const int MAXIMO = 1

const float PORCENTAJE = 0.12

const float PI = 3.1416

Forma de escribir:

Comentarios

```
/*
Comentario  que abarca
 más de una línea
*/

// Comentario de una línea
```

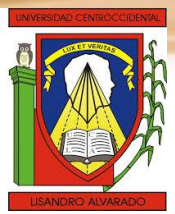
```
/*
    Seccion.h
    Created on: XX/XX/XX
    Author:  XXXXXXX
 */

#ifndef SECCION_H_
#define SECCION_H_
#include "Alumno.h"

class Seccion {

private:
        int num;  // declaración de un atributo

public:
        Seccion();
        void setNum(int n);
        int getNum ();
};
#endif /* SECCION_H_ */
```



Unidad II: El Paradigma Orientado a Objeto.
Tema 4: Lenguaje Formal

Coordinación de Introducción a la Computación