PROGRAMACIÓN ORIENTADA A OBJETOS I

Ing. Juan Angel Calvillo Pérez

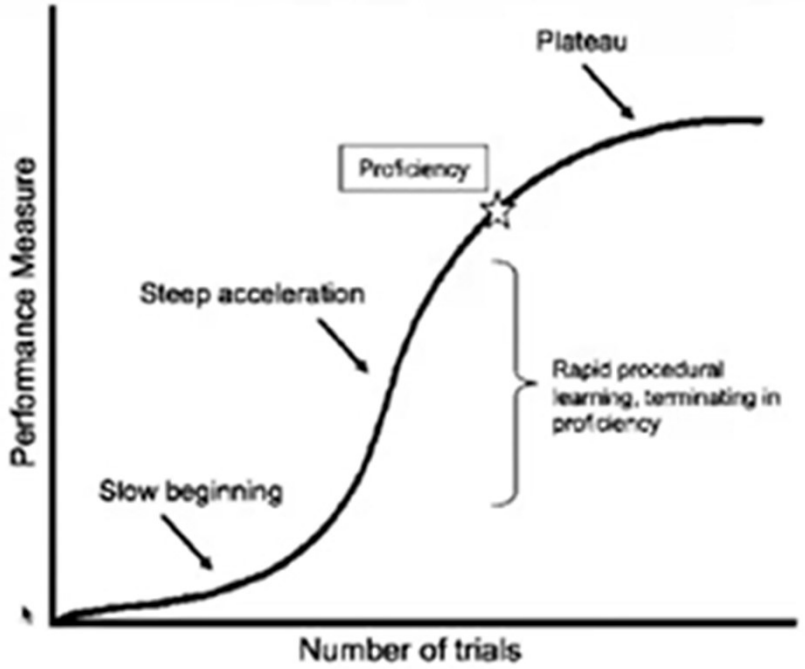
Temario

1 Introducción	1.1. Paradigma orientado a objetos 1.2. Lenguajes orientados a objetos (parcial y absolutamente) 1.3. Beneficios de usar la Programación orientada a objetos 1.4. Concepto de clases (atributos, métodos y mensajes) 1.5. Anatomía de un objeto (información y comportamientos)
2 Interfaces de Desarrollo y el lenguaje de modelado unificado	2.1. Compiladores e IDEs de C# 2.2. Instalación Visual Studio y consola 2.3. Diagramación UML, sintaxis y clases 2.4. Diagramas de paquetes UML 2.5. Diagramas de secuencia UML
3 Conceptos básicos de programación en C#	3.1. Sintaxis y Namespaces 3.2. Tipos primitivos, variables y operadores 3.3. Funciones 3.4. Sentencias de Control (condicionales y ciclos) 3.5. Arreglos 3.6. Funciones de entrada y salida
4 Programación orientada a objetos en C#	4.1. Clases, atributos, métodos y constructores 4.2. Encapsulamiento y control de acceso 4.3. Generalización y Especialización: Herencia 4.4. Sobreescritura de Métodos 4.5. El contexto Static
5 Creación de tipos	5.1 El tipo Object 5.2 Las estructuras Struct y Enum 5.3 Tipos anidados 5.4 Tipos genéricos 5.5 Introducción a Interfaces y clases abstractas

PREVIO

¿Qué es programación?

Programar es una habilidad que como cualquier otra, comienza siendo difícil, se avanza lentamente, y conforme se practica comienza a ser más fácil y avanzas más rápid



¿Qué ya sabes?

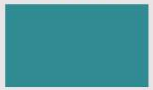
Diagrama de Flujo

- Software Raptor o similar

Terminador/ Iniciador

Esta figura se utiliza para mostrar el inicio del proceso y también donde finaliza.

Proceso



Con esta figura se representan las actividades, tareas y operaciones que se realizan durante el proceso.

Flecha

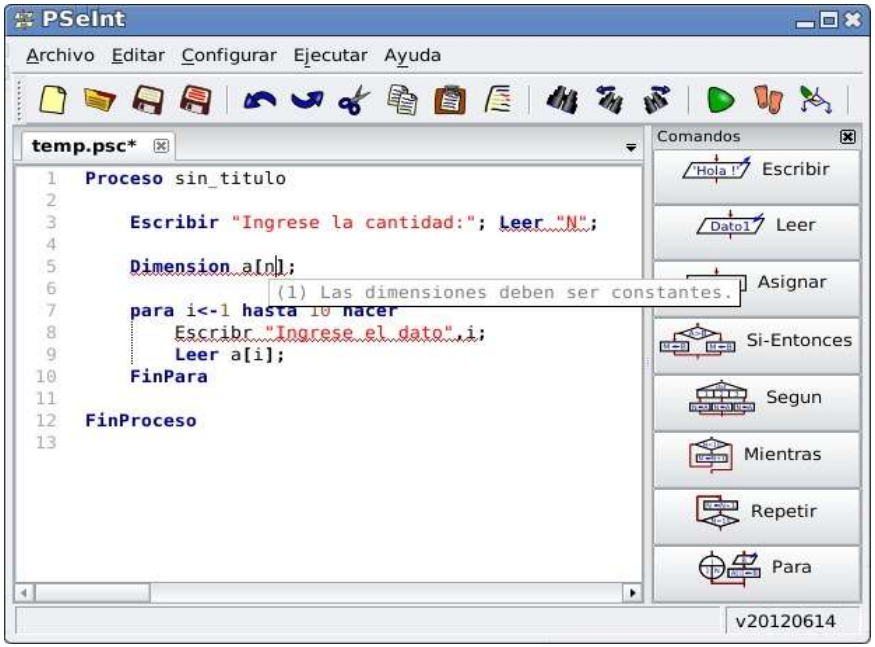
Esta figura indica la dirección del flujo y se utiliza con frecuencia para unir dos etapas sucesivas de un mismo proceso.

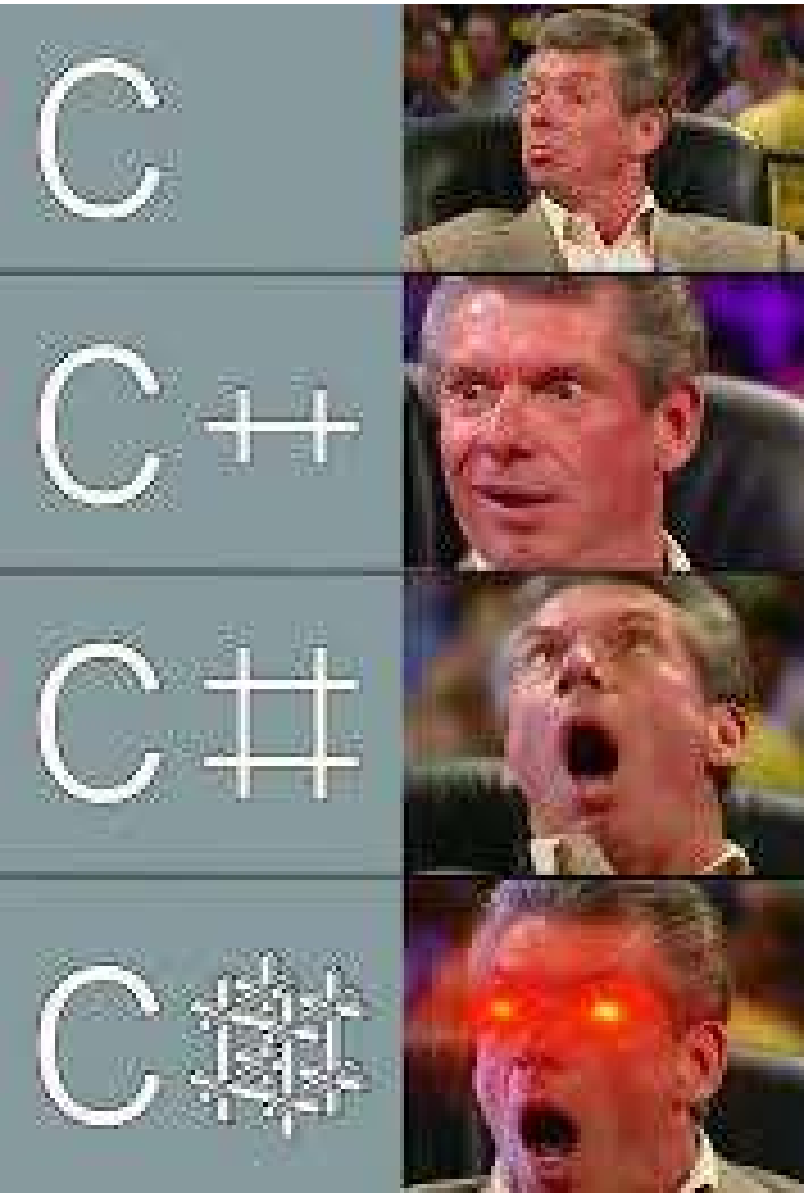
Decisión

En este punto del proceso se debe tomar una determinación y según las flechas que salen de esta figura se determinan las acciones a seguir.

Pseudocódigo

- Software Pseint o similar





¿Que mas?

- Variables
- Operadores
- Expresiones
- Bucles
- Condicionales
- Estructura de datos

Posición	Lenguaje	Última publicación estable	Diseñado por	Influenciado por
#1	Python	Python 3.10-4	Guido van Rossum	ABC, Ada, ALGOL 68, APL, C, C++, CLU, Dylan, Haskell, Icon, Java, Lips, Modula-3, Perl, Standard ML
#2	Java	Java SE 17	James Gosling	Ada 83, C#, C++, CLU, Eiffel, Lisp, Mesa, Modula-3, Oberon, Object Pascal, Objective-C, Smalltalk, Simula67, UCSD Pascal
#3	JavaScript (JS)	ECMAScript 2021	Brendan Eich	AWK, HyperTalk, Java, Scheme
#4	C# (C Sharp)	C# 10.0	Anders Hejlsberg	C++, Cω, Eiffel, F#, Haskell, Icon, J#, J++, Java, ML, Modula-3, Object Pascal, VB
#5	PHP	PHP 8.1.4	Rasmus Lerdorf	C, C++, Hack, HTML, Java, JavaScript, Perl, Tcl
#6	C	C17	Dennis Ritchie	ALGOL 68, Assembly, B (BCPL, CPL), FORTRAN, PL/I
#6	C++	C++20	Bjarne Stroustrup	Ada, ALGOL 68, C, CLU, ML, Mesa, Modula-2, Simula, Smalltalk
#7	R	R 4.1.3	Ross Ihaka y Robert Gentleman	Common Lisp, S, Scheme, XLispStat
#8	TypeScript	TypeScript 4.6.2	Microsoft	C#, Java, JavaScript
#9	Swift	Swift 5.6	Chris Lattner, Doug Gregor, John McCall, Ted Kremenek, Joe Groff y Apple Inc.	C#, CLU, D, Haskell, Objective-C, Python, Ruby, Rust
#10	Objective-C	Objective-C 2.0	Tom Loe y Brad Cox	C, Smalltalk

TEMARIO

1 Introducción

1.1. Paradigma orientado a objetos
1.2. Lenguajes orientados a objetos (parcial y absolutamente)
1.3. Beneficios de usar la Programación orientada a objetos
1.4. Concepto de clases (atributos, métodos y mensajes)
1.5. Anatomía de un objeto (información y comportamientos)

2 Interfaces de Desarrollo y el lenguaje de modelado unificado

2.1. Compiladores e IDEs de C#
2.2. Instalación Visual Studio y consola
2.3. Diagramación UML, sintaxis y clases
2.4. Diagramas de paquetes UML
2.5. Diagramas de secuencia UML

3 Conceptos básicos de programación en C#

3.1. Sintaxis y Namespaces
3.2. Tipos primitivos, variables y operadores
3.3. Funciones
3.4. Sentencias de Control (condicionales y ciclos)
3.5. Arreglos
3.6. Funciones de entrada y salida

4 Programación orientada a objetos en C#

4.1. Clases, atributos, métodos y constructores
4.2. Encapsulamiento y control de acceso
4.3. Generalización y Especialización: Herencia
4.4. Sobreescritura de Métodos
4.5. El contexto Static

5 Creación de tipos

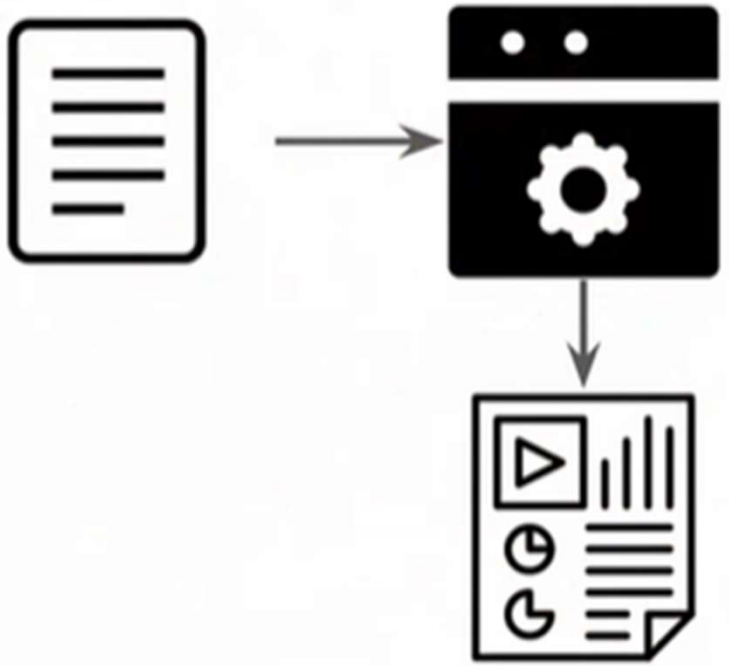
5.1 El tipo Object
5.2 Las estructuras Struct y Enum
5.3 Tipos anidados
5.4 Tipos genéricos
5.5 Introducción a Interfaces y clases abstractas

No soy bueno programando.

Todos pueden programar.

Programar

Es la tarea de procesar información con una computadora, y obtener un resultado



La programación involucra tareas como análisis, generación de algoritmos, medición de rendimiento, eficacia, consumo de recursos, y la implementación de algoritmos en un lenguaje de programación.

Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a chosen programming language, commonly referred to as coding).

Ejercicio hacer pseudocódigo y diagrama de flujo

1. Desarrolle un algoritmo que permita leer dos valores distintos,
   determinar cual de los dos valores es el mayor y escribirlo.

- Desarrolle un algoritmo que realice la sumatoria de los números enteros comprendidos entre el 1 y al 5, es decir, 1 + 2 + 3….
- Desarrolle un algoritmo que permita convertir calificaciones numéricas, según la siguiente tabla:
- A = 10, B =9, C = 8, D = 7, E=6 y F < 5 Se asume que la nota está comprendida entre 1 y 10.