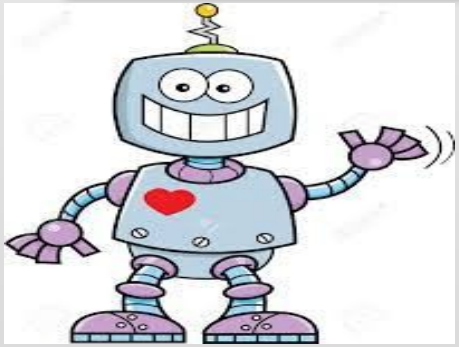
UNER Facultad de Ciencias de la Administración

Licenciatura en Sistemas

Algoritmos y Programación

Temas de la Unidad I



Clase 00
Parte 2

Universidad Nacional de Entre Ríos

Unidad I

Temario

- Lenguajes de Programación.
- Ciclo de vida.
- Sintaxis y Semántica.
- Concepto de Paradigmas de Programación.
- El paradigma imperativo.

Universidad Nacional de Entre Ríos

Lenguajes de Programación

¿ Qué es un Lenguaje de Programación?

Universidad Nacional de Entre Ríos

Lenguajes de Programación

¿ Qué es ?

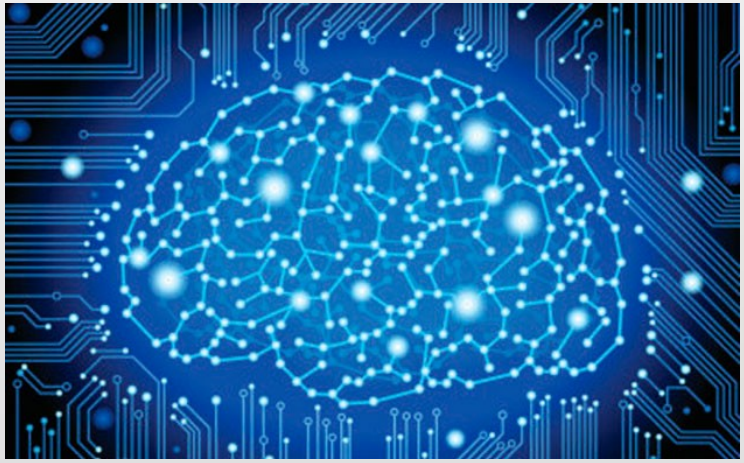
Un Lenguaje de Programación es un conjunto de reglas, notaciones, *símbolos y/o caracteres* que, mediante la aplicación de una *gramática* permite al usuario (programador) expresar su pensamiento y resolver un problema en una computadora.

- **Símbolos y Caracteres**: Alfabeto
- **Gramática** : a) Sintaxis
  b) Semántica
  c) Pragmática

Universidad Nacional de Entre Ríos

Lenguajes de Programación

- Es un **idioma**: Cuenta con un diccionario (los comandos) y una gramática(reglas). El programador traduce las especificaciones de operaciones (que están en su idioma natural y/o en gráficos que especifican su secuencia) en este idioma, elaborando un producto llamado “programa fuente”.



Es una notación formal que permite especificar un modelo de la resolución de un problema

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Clasificación de los Lenguajes

Naturales.

- Propiedad polisémica (distintos significados) (sierra, falda, virus)
- Desarrollados por enriquecimiento progresivo (No de una teoría.)
- Dificultad o imposibilidad de una formalización completa.

Formales.

- Se desarrollan a partir de una teoría establecida.
- Tienen un componente semántico mínimo.
- La sintaxis produce oraciones no ambiguas, en lo que respecta al significado de sus palabras.

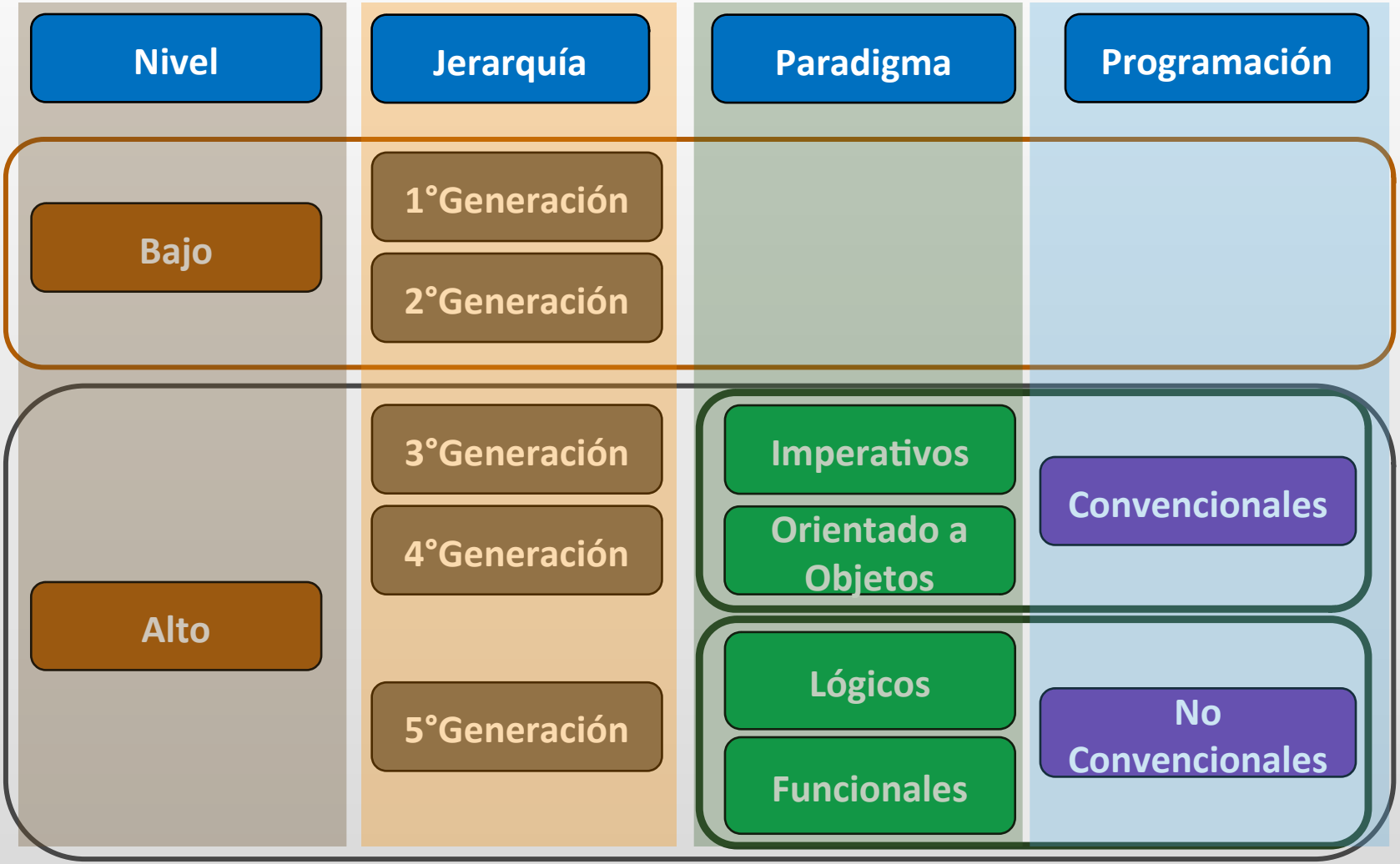
Lenguaje Humano

Lenguaje de Programación

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Clasificación



Universidad Nacional de Entre Ríos

Lenguajes de Programación

Bajo Nivel: Primera Generación

- Dependientes de la máquina.
- Instrucción a Instrucción
- Poco legibles.

Ejemplo:

Restar un valor de una variable

011111 **011111111111** **000000111111**

↓ ↓ ↓

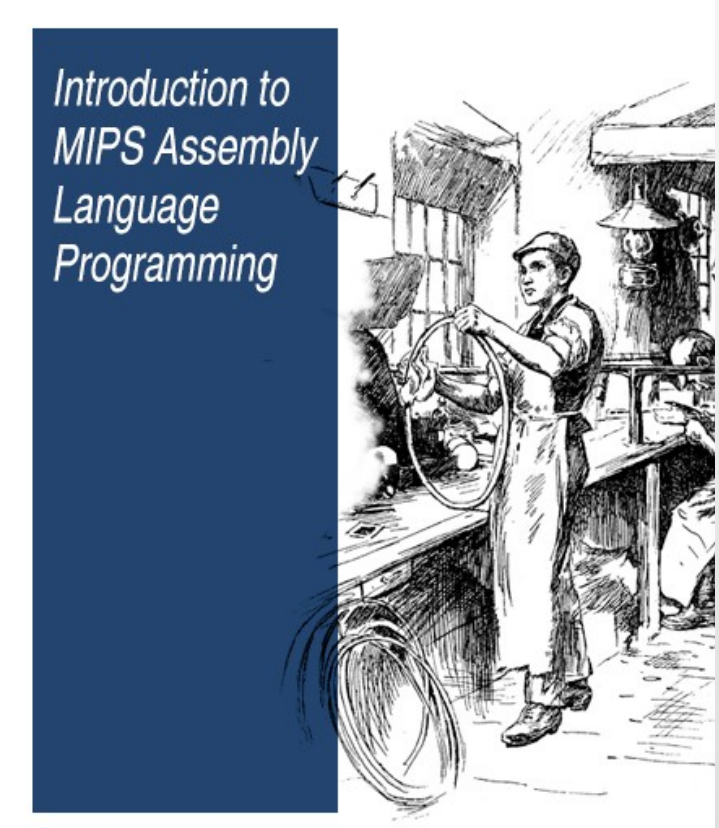
Operación Resta | **Sustraendo** | **Minuendo**

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Bajo Nivel: Segunda Generación

- Concepto de abstracción
- Nombres simbólicos
- Poco legibles.



INSTRUCCIONES EN ASSEMBLER

```
ORG  100h
ADD  AX, 1
MOV  ZX, AX
JMP TERMINO
SIGO:
SUB  ZX, CX
JS  ZXMENOR
```

Universidad Nacional de Entre Ríos

Lenguajes de Programación

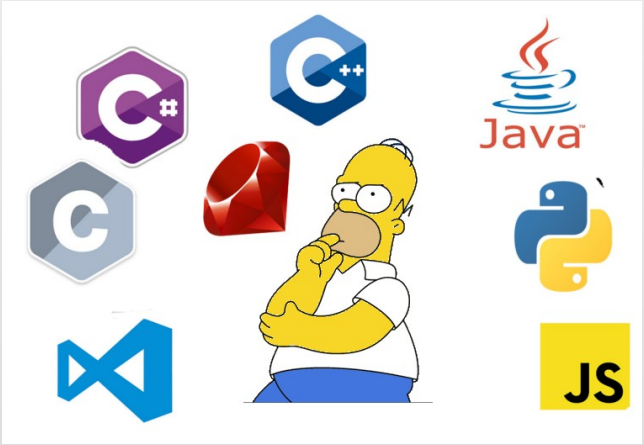
Alto Nivel:

Características

- Macro instrucciones (Ejemplo: Leer <Archivo>)
- Nombres simbólicos
- Lenguajes multipropósito y específicos
- Lenguajes multiplataforma

Ventajas:

- Mas fácil de aprender y escribir
- Mejor documentación
- Mas fácil de mantener



Lenguajes de Programación

Lenguajes convencionales

La mayoría de los lenguajes mas populares han sido diseñados en torno de la arquitectura de la computadora, llamada *arquitectura de von Neumann*.

- Datos y programas almacenados en memoria.
- Memoria y CPU separados
- Entubamiento “piping”
- Concepto de variable.
- Asignación de datos.
- Esquema de control de iteración.

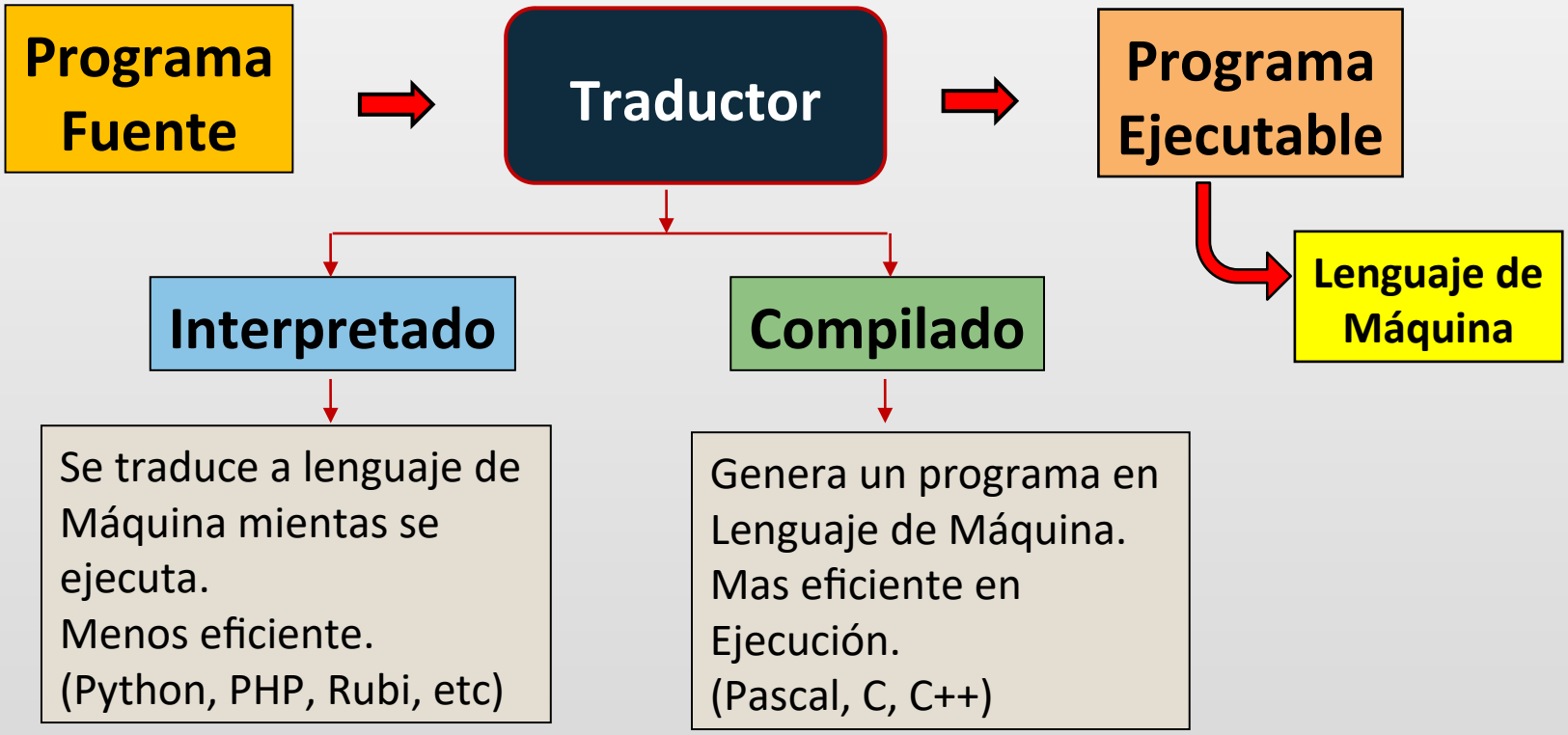
John Von Neumann

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Lenguajes de Alto Nivel (Java, C++, Python, etc.).

Resultado de la codificación: **Programa Fuente**



Universidad Nacional de Entre Ríos

Ciclo de vida del Software

Universidad Nacional de Entre Ríos

Ciclo de vida del software

Ciclo de vida del Software

Especificación y Análisis de Requerimientos

Diseño de Software

Codificación e Implementación

Verificación y Evaluaciòn

Mantenimiento

Universidad Nacional de Entre Ríos

Ciclo de vida del Software

- **Especificación y Análisis de Requerimientos.**
  - Análisis de factibilidad.
  - Identificar y documentar los requerimientos exactos del Sistema
  - *¿Cuál?* Es el problema. *¿Qué?* Debe realizar el sistema.
  - Alcance del Sistema
  - Resultado de la fase: *documento de especificación de requerimientos*.
- **Diseño de Software.**
  - *¿Cómo?* Resolver el problema.
  - Definición de la arquitectura del sistema
  - El método elegido puede afectar la elección del lenguaje utilizado.
  - Resultado de la fase: *documento de especificación de diseño*.
- **Codificación e Implementación.**
  - Elección de una forma de codificación ajustada al diseño.
  - Resultado de la fase: *sistema implementado y documentado*.

Universidad Nacional de Entre Ríos

Ciclo de vida del Software

- **Verificación y Evaluación.**
  - Evaluación de la calidad de la implementación del sistema.
    - ¿Estamos creando correctamente el producto?
    - ¿Estamos creando el producto correcto?
  - Testeo de módulos y testeo de integración entre módulos.
- **Mantenimiento.**
  - Previsión de cambios.
    - Funcionamiento incorrecto.
    - Adición de nuevas capacidades o mejora de las existentes.
    - Adaptación a nuevos ambientes operacionales.

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Sintaxis y Semántica

```
expresión ::= átomo | lista
atomo     ::= número | símbolo
número    ::= [+-]?['0'-'9']+
símbolo   ::= ['A'-'Z''a'-'z'].*
lista::= '(' expresión* ')'
```

Para describir un lenguaje de programación hay que considerar tres aspectos:

- Sintaxis ⇨ Forma del Lenguaje
- Semántica ⇨ Significado del Lenguaje
- Pragmática ⇨ Uso Práctico

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Ejemplo:

a := a + 1

Sintaxis

Instrucción de asignación está constituida por una variable seguida por el operador := seguida por una Expresión.
Expresión: una variable seguida de un operador seguida de una constante.

Semántica

La ejecución de una instrucción de asignación consiste en evaluar la parte derecha de la asignación y modificar con este valor el contenido de la locación de memoria asociada a la variable de la parte izquierda.

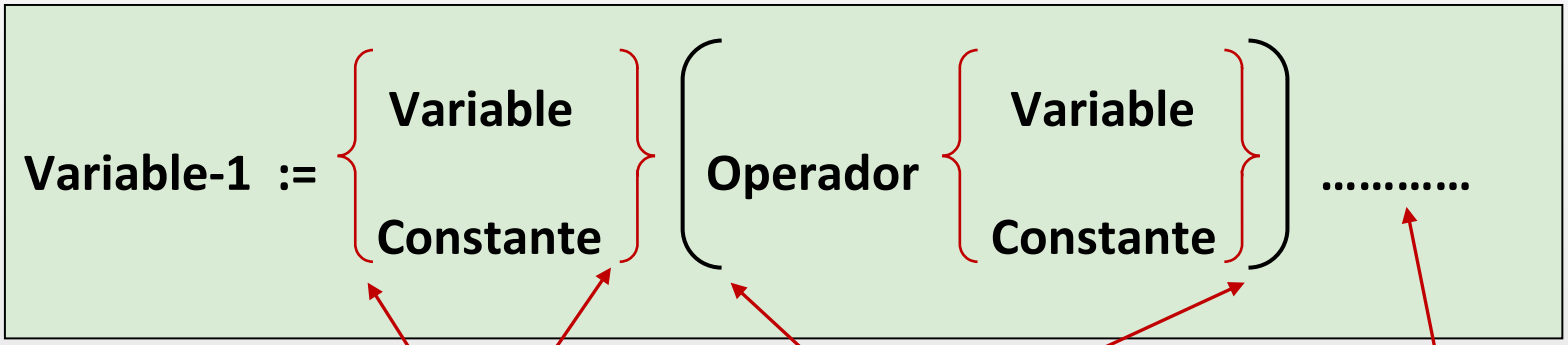
Pragmática

Usamos una instrucción de asignación cuando queremos retener el resultado de una expresión para poder usarlo luego.

Universidad Nacional de Entre Ríos

Lenguajes de Programación

Como sería la sintaxis de una asignación : (un Formato)



Llaves:
Se debe optar por Una Opción

Corchetes:
Es optativo, puede o nó codificarse

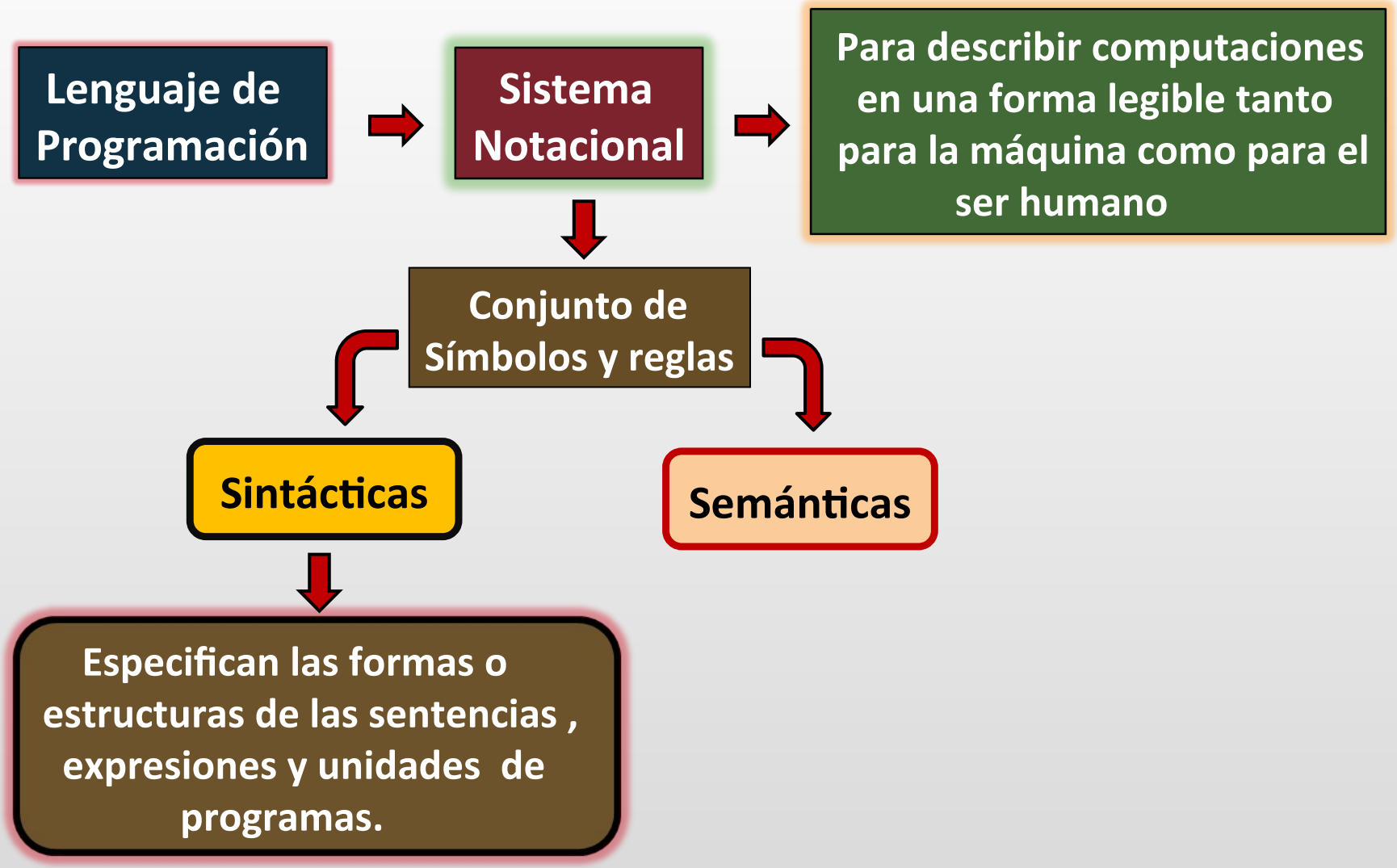
Puntos suspensivos:
Puede repetirse el último (corchete)

Ejemplos:

```
a := 0
a := a + 1 - z
```

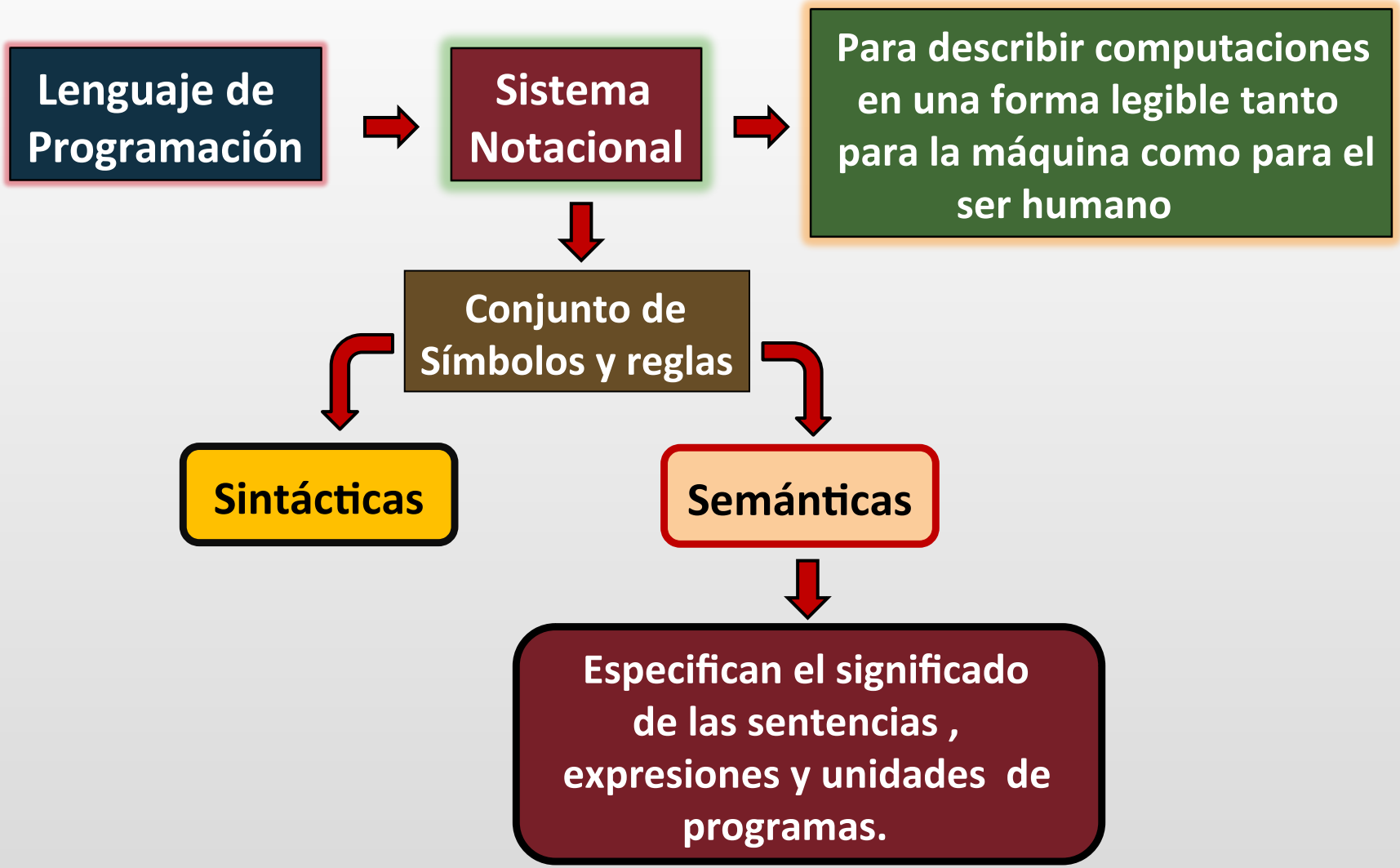
Universidad Nacional de Entre Ríos

Lenguajes de Programación



Universidad Nacional de Entre Ríos

Lenguajes de Programación



Universidad Nacional de Entre Ríos

Paradigmas de Programación

Paradigma:

Según la RAE, la palabra deriva de las palabras griegas que significan "**mostrar**" o "**manifestar**"; y la define como el "ejemplo a seguir o entidad ejemplar de un conjunto. Arquetipo, punto de referencia para imitar".

En términos informáticos: "*modelo computacional de comportamiento o diseño para el desarrollo de programas para la solución de problemas reales".*

Universidad Nacional de Entre Ríos

Paradigmas de Programación

Conceptos:

- Un paradigma de programación es un modelo básico de diseño e implementación de programas.

- Es un modelo que posibilita generar programas conforme con reglas y pautas específicas (tales como estructura modular, fuerte cohesión, alta reusabilidad, etc.).

- Colección de modelos conceptuales integrados que modelan el proceso de desarrollo de software y determinan la estructura de un programa.

Universidad Nacional de Entre Ríos

Paradigmas de Programación

CLASIFICACIÓN

Convencionales

Cercanos a la arquitectura de las computadoras.

No Convencionales

Alejados de la arquitectura de las computadoras.

Convencionales (Procedurales)

- Imperativo
- Orientado a Objetos

No Convencionales (No Procedurales)

- Declarativo o Lógico
- Aplicativo o Funcional

Universidad Nacional de Entre Ríos

Paradigmas de Programación

Paradigmas de Programación

Procedurales

Se describen sentencias que modifican el estado de un programa

En este paradigma se expresa como debe solucionarse un problema especificando una secuencia de acciones a realizar a través de uno o más procedimientos denominados subrutinas o funciones.

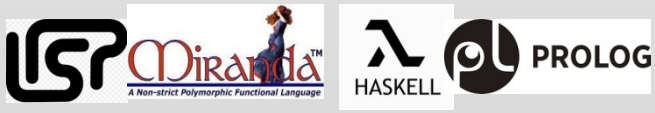
PHP, Java, C, C++, C#, Python, Pascal

No Procedurales

Se describe la lógica de computación necesaria para resolver un problema sin describir un flujo de control de ningún tipo.

No es necesario definir algoritmos puesto que se detalla cual seria la solución del problema pero no cómo llegar a esa solución.

Lisp, Miranda, Haskell, Prolog



Universidad Nacional de Entre Ríos

Paradigmas de Programación

¿ Otros Paradigmas ?

Concurrente.

- Técnicas de programación para expresar el paralelismo entre tareas y para resolver los problemas de comunicación y sincronización entre procesos.
- (Ej. Java, PascalFC, Ada, Occam)

Orientado a Eventos.

- El orden de la ejecución de los programas es determinado por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen. Es posible interactuar con el usuario en cualquier momento de la ejecución.
- (Ej. Javascript, C#, J#, AS3.)

Orientado a Aspectos.

- Hace énfasis en una adecuada modularización de las aplicaciones y posibilitar una mejor separación de las funcionalidades del sistema, eliminado la interdependencia de los módulos.
  - **AspectC++** (compilador que permite desarrollar aspectos en C++), **AspectJ** (extensión Java del proyecto Eclipse para ayudar en el desarrollo orientado a aspectos), **FLOW3** (framework de PHP incluye un módulo para Programación orientada a Aspectos), **AOP SpringFramework 2.5** (Framework de Java que permite programar en el paradigma de Aspectos).

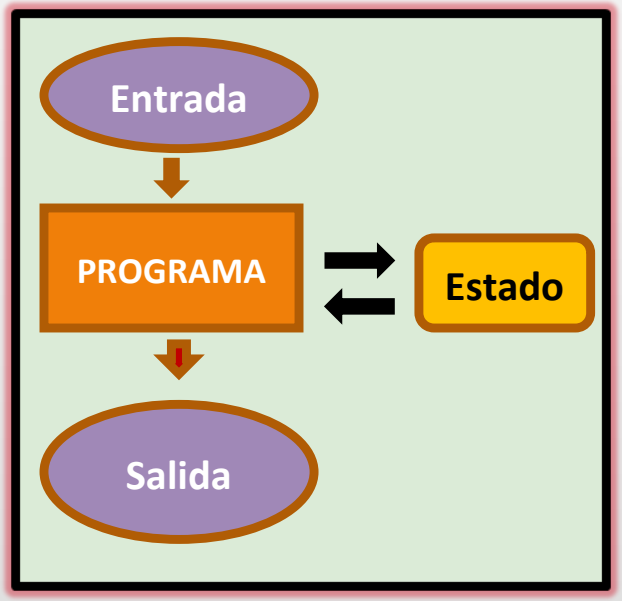
Universidad Nacional de Entre Ríos

Paradigma Imperativo

Principios y Conceptos

El paradigma imperativo debe su nombre al papel dominante que desempeñan las sentencias u órdenes que actualizan variables contenidas en memoria (*Del Latín imperare que significa '"ordenar"*).

En un lenguaje imperativo, un programa es una secuencia de instrucciones que indican el flujo de la ejecución.



Se ajusta a la arquitectura de Von Neumann:

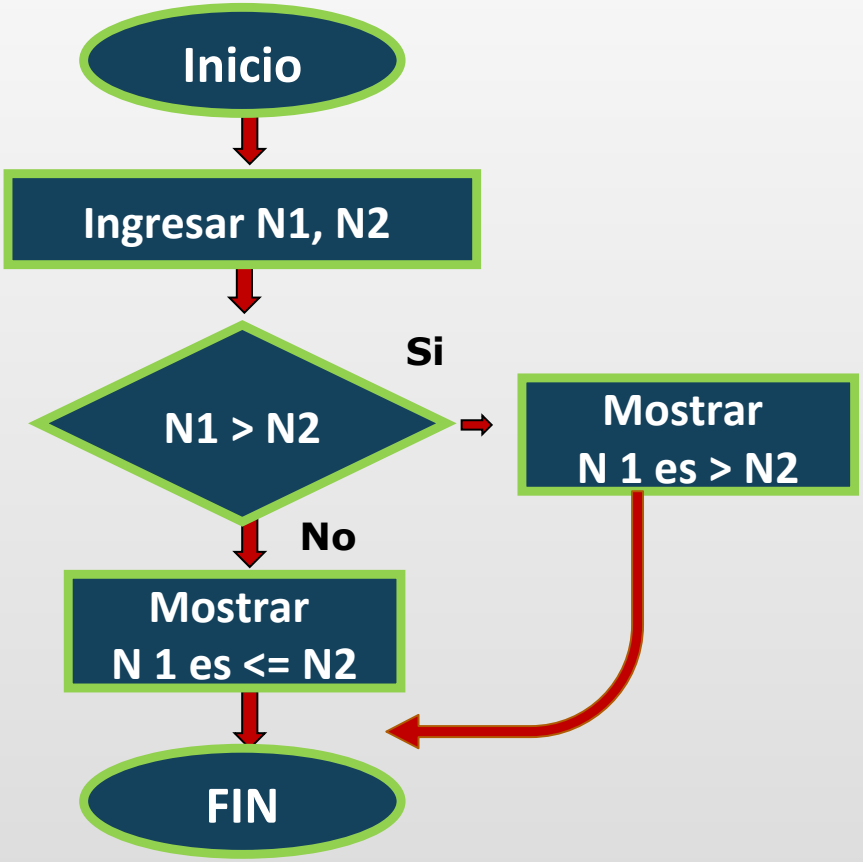
Existe una representación codificada de una computación (programa) en memoria que se va ejecutando secuencialmente y que toma datos de la memoria, efectúa cálculos y actualiza la memoria".

Universidad Nacional de Entre Ríos

Paradigma Imperativo

Principios y Conceptos

Se le suele denominar paradigma algorítmico, dado que un algoritmo se define como "conjunto finito de reglas diseñadas para crear una secuencia de operaciones para resolver un tipo específico de problemas".



Universidad Nacional de Entre Ríos

Paradigma Imperativo

Características Principales:

Concepto de celda de memoria (“variable”) para almacenar valores.

Operaciones de asignación para cambiar los valores de las variables.

Tipos de Datos.

Abstracciones que representan un conjunto de valores junto con las operaciones y relaciones que son aplicables a ellos.

Estructuras de Control.

1. Seleccionar una opción entre algunos flujos de control alternativos
2. Repetir la ejecución de una serie de sentencias.

Universidad Nacional de Entre Ríos

Paradigma Imperativo

Paradigma Imperativo

No Estructurado

- Libertad para definir datos y manipularlos como y donde pareciera mejor.
- Mucha libertad para escribir instrucciones de salto (los famosos **goto's** , incluso del interior de un bloque de instrucciones al interior de otro.
- **Código Spaghetti**
- **Crisis de los ´70**

Estructurados

- Sistema de Tipos con tipos básicos y constructores para especificar tipos compuestos a partir de los Básicos.
- Conjunto de instrucciones básicas y mecanismos para agruparlas y formar instrucciones (o bloques de instrucciones) compuestas que pueden ejecutarse solo de una manera: comenzando en su único punto de inicio y terminando en su único punto de cierre
- No permitir que a través de instrucciones **goto** la ejecución del programa salte del interior de una instrucción al interior de otra.

Universidad Nacional de Entre Ríos

Paradigma Imperativo

No Estructurado

001	Inicio
002	Ingresar Código
003	Si Codigo = 0 goto 010
004	Ingresar Saldo, Importe
005	Si Codigo = 2 goto 008
006	Saldo = Saldo + Importe
007	goto 002
008	Saldo = Saldo – Importe
009	Goto 002
010	Fin

Estructurados

```
Inicio
Ingresar Código
Mientras Código <> 0
    Ingresar Saldo, Importe
    Si Código = 1
        Saldo :=  Saldo + Importe
    Sino
        Saldo := Saldo – Importe
    FinSi
    Ingresar Código
FinMientras
Fin
```

Universidad Nacional de Entre Ríos

Paradigmas de Programación

Paradigma Imperativo

↓

Estructurado

↓

OBJETIVOS

↓

Crear Programas que:

↓

- **Sean entendibles y fáciles de mantener**
- **Estén libre de errores**

Universidad Nacional de Entre Ríos

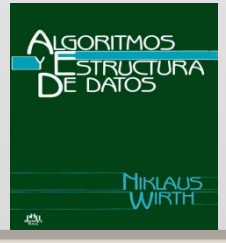
Paradigmas Imperativo

Resumiendo:

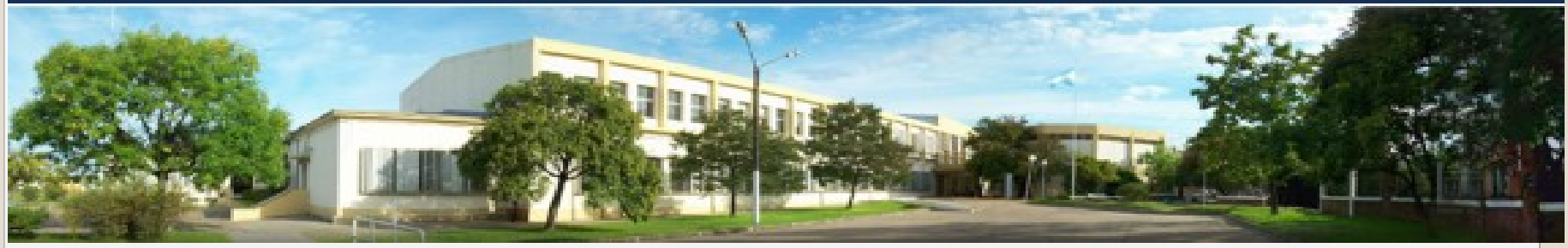
La secuencia y estructuras de control, el concepto de variable, tipos de datos y la operación de asignación son sus componentes fundamentales.

El algoritmo es la estructura informática utilizada para definir las acciones de un programa y las estructuras de datos (variable, array, registro, archivo, etc.) son los elementos pasivos sobre los que actúa el algoritmo.

Algoritmo + Estructura de Datos = Programa



UNER Facultad de Ciencias de la Administración



Algoritmos y Programación

Unidad I

FIN DE LA CLASE