Facultad de Ciencias de la Administración

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

PROGRAMACIÓN I

Unidad I – Introducción a la programación de computadoras

Lenguajes y Paradigmas de programación

2022 - Primer Cuatrimestre

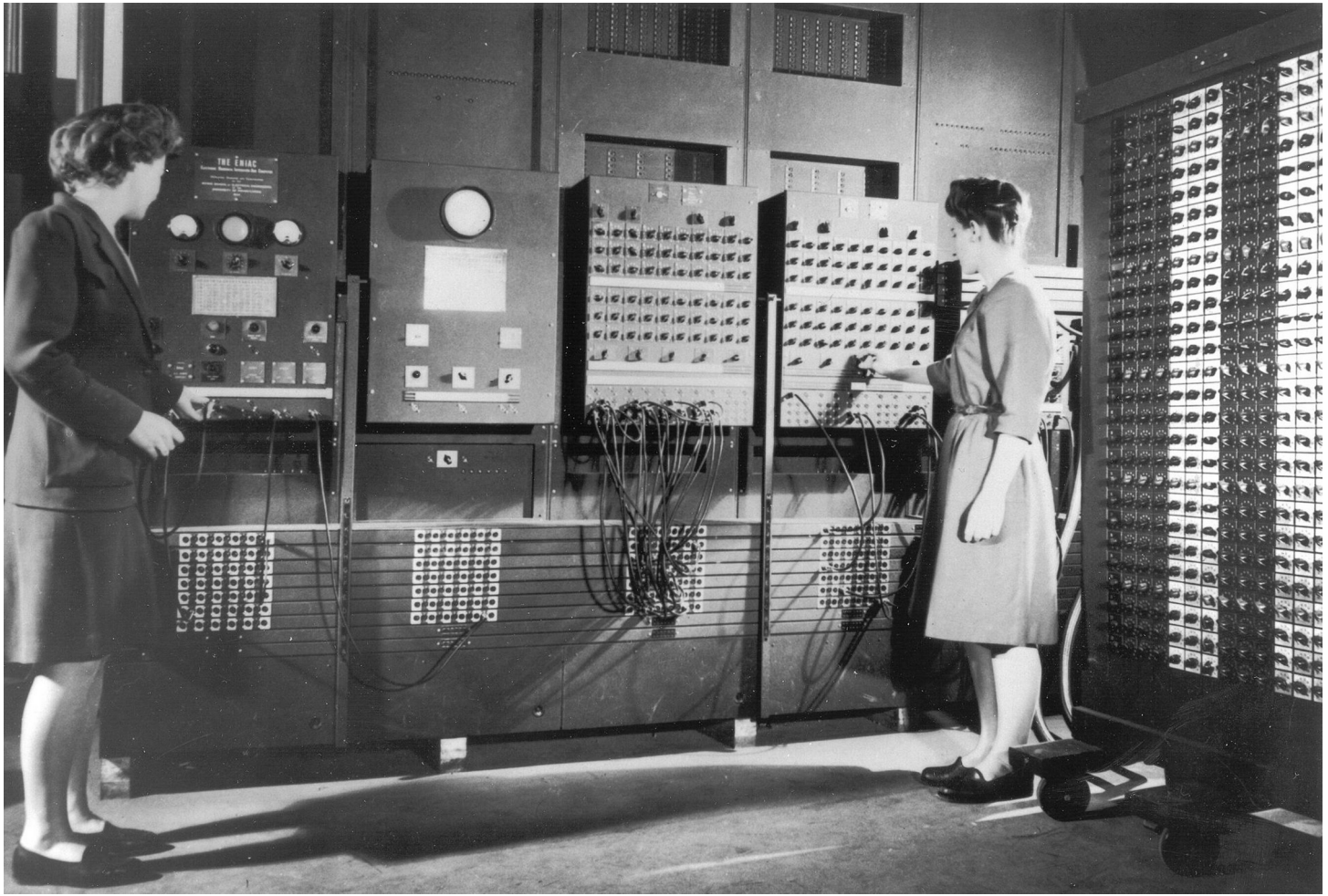
Tecnicatura Universitaria en Desarrollo Web

Facultad de Ciencias de la Administración

Universidad Nacional de Entre Ríos

- **Objetivos**
  - Comprender los conceptos de algoritmo, programa y lenguaje de programación.
  - Conocer e identificar cada una de las etapas del ciclo de vida de desarrollo de software.
- **Temas a desarrollar:**
  - **Algoritmos y programas**
    - Programa. Concepto. Algoritmos. Concepto, características.
    - Computadoras. Características y Componentes.
  - **Lenguajes de programación**
    - Definición y Niveles.
  - **Paradigmas de programación**
    - Definición y Clasificaciones.
  - **Ciclo de vida de desarrollo de software**
    - Etapas.

Cuando pensamos en alguien programando... ¿pensamos en esto?



O ¿pensamos en esto?

Lenguajes de programación

- *Un **lenguaje de programación** es un conjunto de símbolos (alfabeto) y un conjunto de reglas para combinar dichos símbolos que se usan para **expresar** programas puedan ser entendidos por computadoras.*

- Constan de un ***léxico***, una ***sintaxis*** y una ***semántica***.
  - **Léxico**: Conjunto de símbolos permitidos o vocabulario.
  - **Sintaxis**: Reglas que indican cómo realizar las construcciones del lenguaje.
  - **Semántica**: Reglas que permiten determinar el significado de cualquier construcción del lenguaje.
- En la actualidad existen cientos de lenguajes de programación que permiten escribir programas en distintos ámbitos. Cada uno de ellos fue creado para cumplir con un propósito determinado.
- Podemos llegar a preguntarnos... ¿Cuál es el mejor lenguaje de programación?
  - Lo cierto es que cada lenguaje tiene sus fortalezas y debilidades.

Lenguajes de programación (2) – Clasificación por nivel

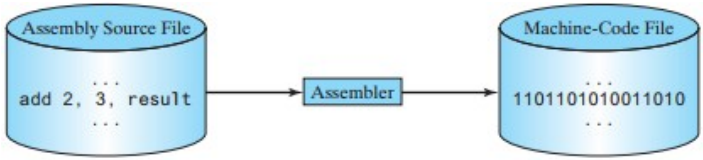
- Podemos clasificar los lenguajes de programación por su nivel:
  - Lenguajes de bajo nivel: código máquina.
  - Lenguajes de nivel medio: Assembler (asm) o lenguaje ensamblador.
  - Lenguajes de alto nivel: C, C++, Java, Python, JavaScript, etc.

Lenguajes de programación (3) – Clasificación por nivel

- El ***lenguaje de bajo nivel***, es directamente el **lenguaje propio de la máquina**.
- El mismo está **íntimamente** relacionado con el hardware, y normalmente está formado por largas secuencias de ceros y unos que indican a la computadora las instrucciones básicas a ejecutar que conforman un programa.
- El código máquina resulta difícil de manejar para los seres humanos. Por este motivo, con el avance y difusión de las computadoras, se empezaron a utilizar abreviaturas para representar las operaciones básicas de las computadoras, que es lo que se conoce como **lenguaje ensamblador**.
- Por ejemplo; el programador que escribe la instrucción de `ADD 0184` para una máquina IBM antigua, tendría que haber escrito:
  - **`00010000000000000000000010111000`**

Lenguajes de programación (3) – Clasificación por nivel

- Características del **lenguaje máquina** y **ensamblador:**
  - Dependientes del procesador.
  - Requiere el conocimiento del hardware del procesador.



  - Utiliza instrucciones muy elementales, que realizan operaciones básicas.
  - Los programas ocupan poca memoria logrando alta velocidad de ejecución.
- Es de destacar que los **lenguajes de bajo** y **medio nivel** requieren un amplio conocimiento del hardware y además se necesitan muchas instrucciones para llevar a cabo tareas sencillas.
- Con el objeto de acelerar el proceso de desarrollo de programas, surgieron los **lenguajes de alto nivel**, que permiten realizar tareas más complejas con instrucciones simples.
- Es así, que para traducir estos programas de alto nivel a código máquina surgieron los **compiladores** y los **intérpretes**.

Lenguajes de programación (4) – Clasificación por nivel

- Características de los **Lenguajes de alto nivel**:
  - Independientes del procesador.
  - Permiten desconocer detalles del hardware.
  - Escritura cercana al lenguaje natural.
  - Poseen librerías con funciones de entrada/salida, matemáticas, etc

Lenguajes de programación (5) – Clasificación por nivel

```c
#include <stdio.h>
int main()
{
    int num1, num2;
    float avg;
    num1 = 2;
    num2 = 8;
    avg= (float)(num1+num2)/2;
    printf("Average of %d and %d is: %.2f",num1,num2,avg);
    return 0;
}
```

C

Promedio de dos números

```python
num1, num2 = 2, 8
avg = (num1 + num2)/2
print("Average of {0} and {1} is: {2}".format(num1, num2, avg))
```

Python

```
.LC0:
        .string "Average of %d and %d is: %.2f"
main:
        stp     x29, x30, [sp, -32]!
        add     x29, sp, 0
        mov     w0, 2
        str     w0, [x29, 28]
        mov     w0, 8
        str     w0, [x29, 24]
        ldr     w1, [x29, 28]
        ldr     w0, [x29, 24]
        add     w0, w1, w0
        scvtf   s1, w0
        fmov    s0, 2.0e+0
        fdiv    s0, s1, s0
        str     s0, [x29, 20]
        ldr     s0, [x29, 20]
        fcvt    d0, s0
        adrp    x0, .LC0
        add     x0, x0, :lo12:.LC0
        ldr     w2, [x29, 24]
        ldr     w1, [x29, 28]
        bl      printf
        mov     w0, 0
        ldp     x29, x30, [sp], 32
        ret
```

ARM64

Paradigmas de programación

- *Un **paradigma** es una teoría o conjunto de teorías cuyo núcleo es aceptado por todos los integrantes que usan ese paradigma, y que sirve como modelo para resolver problemas utilizándolo*.

- En el mundo del desarrollo de software un paradigma de programación indica el método y la forma en la que se deben estructurar y organizar las tareas que debe realizar el programa que se está desarrollando.
- Algunos de los paradigmas más populares son:
  - **Paradigma imperativo**: se caracteriza por enfocarse en cómo se pretenden realizar los cálculos por medio de instrucciones secuenciales para formar algoritmos que resuelvan una tarea específica.
    - El paradigma imperativo es el más utilizado en los lenguajes de programación, de forma única o en conjunción con otros paradigmas.
    - Ejemplos cotidianos de este paradigma se pueden encontrar en las recetas de cocina o en cualquier guía paso a paso.

Paradigmas de programación (2)

- **Paradigma procedural:** derivado del paradigma imperativo, añadiendo funcionalidades extra como la creación de procedimientos o funciones que pueden ser llamadas en las secuencias de código como si fueran instrucciones simples.
  - De esta forma los programas están organizados y modularizados en funciones específicas que no tienen por qué residir en el mismo archivo. Esto favorece la modularización y organización del código.
- **Paradigma orientado a objetos:** este paradigma es uno de los más populares junto con el imperativo y se basa en encapsular las entidades principales del programa en objetos que pueden contener tanto datos como comportamiento (métodos para interactuar con otros datos u otros objetos).
  - Este paradigma es especialmente útil, puesto que favorece la modularidad, el encapsulamiento, la reusabilidad y la escalabilidad de cualquier programa.
  - Muchas implementaciones de este paradigma cuentan con la herencia de objetos. Implementan este paradigma los lenguajes de programación Java, C++, Rust, Python o JavaScript.

Paradigmas de programación (3)

- **Paradigma declarativo:** Está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla.
  - Los lenguajes de programación declarativos describen el resultado final deseado, en lugar de mostrar todos los pasos de trabajo. Para alcanzar el objetivo se determina automáticamente la vía de solución.
  - Como la programación declarativa no determina el "cómo", sino que funciona a un nivel de abstracción muy alto, este paradigma deja margen para la optimización. Si se ha desarrollado un procedimiento de ejecución mejor, el algoritmo integrado lo encuentra y lo aplica.
- Los distintos lenguajes declarativos se pueden subdividir, a su vez, en dos paradigmas, el de la **programación lógica** y el de la **programación funcional**.
  - **Programación lógica**
    - Basada en las lógicas formales. Todo programa escrito en un lenguaje de programación lógico es un conjunto de sentencias lógicas, expresando hechos y reglas sobre un dominio.
    - Los lenguajes de programación lógica utilizan una estrategia de búsqueda para producir los resultados en función de los hechos y reglas. Por ejemplo: **backtracking**,

Paradigmas de programación (3)

- **Paradigma funcional:** se basa en funciones matemáticas que se centran en los cambios de estado del programa por medio de la mutación de variables.
  - Tiene su origen en el cálculo lambda y presenta características muy potentes, como la recursividad, el uso de funciones de orden superior o el uso de funciones puras que definen que la misma función, con los mismos argumentos, siempre debe devolver el mismo resultado (evitando cualquier efecto colateral). Ejemplos: Haskell, Miranda y otros híbridos como Lisp, Scala y OCalm.

Bibliografía

- Pablo A. García, Marcelo A. Haberman, Federico N. Guerrero: ***"Programación E1201: curso de grado"***. 1Era Edición. Ed. Editorial de la UNLP. 2021.
- Óscar Ramírez Jiménez: ***"Python a fondo"*** 1era Edición. Ed. Marcombo S.L.. 2021.
- Allen Downey. ***"Think Python"***. 2Da Edición. Green Tea Press. 2015.
- Eirc Matthes: ***"Python Crash Course"***. 1era Edición. Ed. No Starch Press. 2016.
- Zed A. Shaw: ***"Learn Python 3 the Hard Way"***. 1era Edición. Ed. Addison-Wesley. 2017.
- Armando E. De Giusti.: ***"Algoritmos, datos y programas con aplicaciones en Pascal, Delphi y Visual Da Vinci"***. Ed. Buenos Aires Prentice Hall; Pearson Educación. 2001.