Informática

Unidad 2B: Introducción a los Lenguajes de Programación

Ingeniería en Mecatrónica

Facultad de Ingeniería
Universidad Nacional de Cuyo

Dr. Ing. Martín G. Marchetta
mmarchetta@fing.uncu.edu.ar

UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERIA
en acción continua...

2B – Introducción a los lenguajes de programación

- Conceptos
  - Un algoritmo es un conjunto de instrucciones específicas para llevar a cabo un procedimiento o resolver un problema, usualmente con el requerimiento de que el procedimiento termine en algún punto
    [E.W. Weisstein, Algorithm, MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/Algorithm.html]
  - En informática, un algoritmo usualmente se ***implementa*** en un lenguaje de programación, y se convierte en un ***programa***
  - El algoritmo se apoya en una ***estructura de datos*** para resolver el problema en cuestión
  - La estructura de datos especifica la forma en la que se almacenan y mantienen en la memoria los datos que el programa recibe, crea, modifica, o elimina durante su operación, incluyendo también los ***tipos de dato*** involucrados

2B – Introducción a los lenguajes de programación

- Estructura de datos:
  - Ejemplos
    - Arreglo de 1D (vector)
    - Arreglo 2D (matriz)
    - Arreglo nD (arreglo multidimensional)
    - Listas enlazadas
      - Enlace simple
        - Pilas
        - Colas
      - Enlace doble
  - La estructura de datos puede contener varios elementos de cada tipo (ej: varias listas enlazadas, arreglos, variables primitivas, etc)
  - El tipo de dato de cada elemento de cada sub-estructura determina el rango de valores válidos que se pueden almacenar

2B – Introducción a los lenguajes de programación

- Algoritmos y Estructuras de datos
  - El algoritmo y la estructura de datos asociada deben ser consistentes
  - El algoritmo pone restricciones sobre la estructura de datos:
    - La estructura de datos debe poder contener todos los datos que el algoritmo necesita, los resultados intermedios y el resultado final
    - Los datos se deben poder acceder de la forma en que el algoritmo lo requiere (de manera indizada, recorriendo una lista secuencialmente, etc)
  - El algoritmo puede requerir que los datos estén ordenados de cierta forma en la entrada, y puede necesitar mantenerlos ordenados igualmente durante su operación

2B – Introducción a los lenguajes de programación

- Niveles de abstracción
  - Existen lenguajes de programación con distintos “niveles de abstracción”:
    - lenguajes de “bajo nivel” (cercanos al lenguaje de máquina)
    - lenguajes de “alto nivel” (cercanos al problema a resolver)
  - Algunos ejemplos, del mayor a menor nivel:
    - MatLab
    - Java, C#
    - C++
    - C
    - Assembly
  - La elección del lenguaje depende de la complejidad del problema a resolver, requisitos de acceso al hardware, rendimiento necesario, etc.

2B – Introducción a los lenguajes de programación

- Paradigmas de programación
  - Un paradigma de programación define un ***estilo*** o ***patrón*** de programación: especifica un estilo para la sintaxis y la semántica del lenguaje, es decir, los elementos que existen en el dominio del lenguaje y pueden combinarse para crear programas
  - Existen 3 paradigmas básicos, aunque pueden existir variantes o casos híbridos
    - Imperativo (que incluye el “estructurado” y el “orientado a objetos”). Ej: C, C++, Java, C#
    - Lógico (Ej: Prolog)
    - Funcional (Ej: LISP)

2B – Introducción a los lenguajes de programación

- Interpretación vs. Compilación
  - Los programas escritos en cualquier lenguaje simbólico deben ser ***traducidos*** al lenguaje de máquina para poder ser ejecutados
  - Esto puede hacerse de 2 formas: compilación o interpretación
    - La compilación se realiza antes de ejecutar el programa y consiste en la traducción completa del programa al lenguaje de máquina (generalmente al lenguaje del nivel del sistema operativo, no el de máquina real)
    - La interpretación traduce instrucción por instrucción, a medida que se va requiriendo durante la ejecución

2B – Introducción a los lenguajes de programación

- Interpretación vs. Compilación
  - Compilación:
    - La raliza un programa llamado ***compilador***
    - Si el lenguaje simbólico tiene una relación 1 a 1 con las instrucciones generadas en la traducción, el traductor se llama ***ensamblador*** en vez de compilador
    - Una vez compilado, el programa puede correrse en otras máquinas con la misma plataforma de base (SO, arquitectura de la computadora, etc): el compilador no se necesita más
  - Interpretación:
    - La realiza un ***intérprete***
    - Para ejecutar el programa, el intérprete debe estar presente (por lo tanto para llevarlo a otra computadora, se necesita el intérprete o la máquina virtual correspondiente en ella)

2B – Introducción a los lenguajes de programación

- Lenguajes fuerte y débilmente tipados
  - Esta distinción no tiene una definición precisa
  - La distinción se basa en la forma en la que los lenguajes manejan los tipos de datos
  - Algunos conceptos relacionados
    - Type safety (seguridad de tipos): se relaciona con la fortaleza con la que el lenguaje de pogramación impide o desalienta el uso de tipos de datos incorrectos (ej: asignar un decimal en punto flotante a una variable entera)
    - Static/Dynamic type checking: se relaciona con el momento en el que los tipos de datos se verifican
      - Static: en tiempo de compilación
      - Dynamic: en tiempo de ejecución
  - Ejemplos
    - Lenguajes considerados “fuertemente tipados”: C, C++, Java, C#, Kotlin
    - Lenguajes considerados “débilmente tipados”: Javascript, Python, Ruby, Groovy

2A – Algoritmos y Lenguajes de programación

- Entornos Integrados de Desarrollo (IDE)
  - Los IDE son sistemas que incluyen funcionalidades de apoyo al desarrollo de software
  - Algunas funcionalidades comunes
    - Resaltado de sintaxis
    - Simplificación del proceso de compilación
    - Navegación del código
    - Resaltado de errores
    - Depuración
      - Ejecución paso por paso
      - Inspección de variables y otras estructuras (ej: pila de llamadas)
  - Ejemplos: Eclipse, Netbeans, PyCharm, Spyder

- Proceso de compilación
  - Para convertir un programa simbólico (***fuente***) en un programa ***ejecutable*** se debe:
    - **Compilar** el/los archivos fuente → ***código objeto***
    - Enlazar el/los archivos objeto → ***programa ejecutable***