Capítulo

2

Tipos de Datos

Conceptos Previos

Un lenguaje de programación se implementa construyendo un *traductor*, el cual traduce los programas escritos en dicho lenguaje de programación, a programas escritos en lenguaje máquina o algún otro lenguaje más cercano al lenguaje máquina.

Se distinguen los siguientes tipos de computadoras:

- **Computadora:** Conjunto integrado de algoritmos y estructuras de datos capaz de almacenar y ejecutar programas.
- **Computadora real o de hardware:** Formada por dispositivos físicos.
- **Computadora simulada por software:** Formada por software que se ejecuta en otra computadora.
- **Computadora virtual:** Formada por partes de hardware y de software. Es la que ejecuta los programas traducidos por el traductor.

Tipos de Datos

Todo programa se puede considerar como la especificación de un conjunto de operaciones que se van a aplicar sobre ciertos datos en un orden determinado. Todo lenguaje, con diferencias, incluye:

- **Datos:** Tipos de datos permisibles.
- **Operaciones:** Tipos de operaciones disponibles.
- **Mecanismos de control:** Mecanismo de control del orden en que las operaciones se aplican a los datos.

Se revisarán los tipos de datos más representativos de los lenguajes de programación:

- **Tipos elementales de datos:** Basados en características disponibles a partir del hardware del computador.
- **Tipos de datos estructurados:** Características simuladas por software.

Objetos de Datos

Mientras que los datos almacenados en la memoria del computador tienen una estructura simple, tratados como bytes, los datos almacenados en una computadora virtual tienen una estructura compleja, tratados como pilas, arreglos, etc.

Objeto de datos: Agrupación de uno o más datos en tiempo de ejecución en una computadora virtual.

Un programa en ejecución utiliza muchos objetos de datos. Estos objetos de datos y sus interrelaciones cambian dinámicamente durante la ejecución.

Según quién define los objetos de datos tenemos:

- **Definidos por el programador:** Los crea y manipula explícitamente, a través de declaraciones y enunciados.

- **Definidos por el sistema:** Los crea la computadora virtual (comúnmente, de manera automática para el programador, según se requieran) para "mantenimiento" durante la ejecución del programa, y a los cuales el programador no tiene acceso directo, como por ejemplo: pilas de almacenamiento en tiempo de ejecución, registros de activación de subprogramas, memorias intermedias de archivos y listas de espacio libre.

Los objetos de datos funcionan como contenedores de valores de datos (un número, un carácter, un apuntador a otro objeto de datos, etc.).

Es fácil confundir objeto de datos y **valores de datos**. La distinción se aprecia mejor por su implementación: El primero representa un almacenamiento en la memoria del computador; mientras que el segundo; un patrón de bits.

Los objetos de datos tienen un **tiempo de vida** durante el cual pueden usarse para guardar y recuperar valores de datos.

Los objetos de datos se caracterizan por un conjunto de **atributos**, los que determinan el **número y tipos de valores** que pueden contener, así como la organización lógica de estos valores. Los atributos no varían durante el tiempo de vida de un objeto de datos.

Los objetos de datos participan en **enlaces** durante su tiempo de vida, algunos de los cuales pueden cambiar durante este tiempo. Estos enlaces son:

- **Localidad:** Es la posición que ocupa en la memoria.

- **Valor:** Por lo general resulta de una operación de asignación.

- **Nombre:** Se establece mediante declaraciones y se modifica mediante llamadas y devoluciones de programas.

- **Componente:** Enlace de un objeto de datos a otros de los que forma parte. Se suele representar a través de un apuntador.

Un objeto de datos es **elemental** si su valor de datos se manipula como una unidad, y es una **estructura de datos** si es un agregado de otros objetos de datos.

Variables y Constantes

Una **variable** es un objeto de datos definido y nombrado explícitamente en un programa. Si el objeto de datos es **elemental**, la variable es **simple**. Por lo común, el(los) valor(es) de una variable es(son) modificable(s), mediante operaciones de asignación.

Una **constante** es un objeto de datos, definido y nombrado explícitamente en un programa, y enlazado permanentemente a un valor de datos durante su tiempo de vida.

Una **constante literal** es una constante cuyo nombre es la representación por escrito de su valor.

Una **constante definida por el programador** es una constante cuyo nombre es elegido por el programador.

Dada la característica de una constante, el traductor puede utilizar esta información para realizar optimizaciones.

Tipo de Datos

Un **tipo de dato** es una clase de objeto de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Todo lenguaje tiene un conjunto de **tipos primitivos de datos** que están integrados al lenguaje. Un lenguaje puede proveer recursos para definir nuevos tipos de datos.

Los elementos básicos de la especificación de un tipo de datos son:

- Los **atributos** que distinguen objetos de datos de ese tipo.
- Los **valores** que los objetos de datos de ese tipo pueden tener.
- Las **operaciones** que se pueden realizar con los objetos de datos de ese tipo.

Ejemplo: Para un arreglo de enteros tendríamos:

- Atributos: Número de dimensiones, rango de los índices de cada dimensión, tipo de dato de los componentes, es decir, entero.
- Valores: El conjunto de los números enteros soportados por el tipo de datos entero.
- Operaciones: Sub-indización para seleccionar componentes del arreglo, creación de un arreglo, modificación de las dimensiones, etc.

Los elementos básicos de la implementación de un tipo de datos son:

- La **representación de almacenamiento** usada para representar los objetos de datos de ese tipo.
- Los **algoritmos, procedimientos u operaciones** que manipulan la representación de almacenamiento elegida.

Por último, un tipo de datos tiene una **representación sintáctica**, en la que, tanto la **especificación** como la **implementación** son en gran medida independientes.

Ejemplo: Para un objeto de datos de un tipo en particular:

✍ Los atributos se representan con declaraciones o definiciones de tipo.

✍ Los valores se representan con constantes.

✍ Las operaciones se representan con símbolos especiales, procedimientos integrados o funciones.

Esta información suele ser utilizada por los traductores para determinar el tiempo de creación de enlaces, la representación de almacenamiento a utilizar, revisar errores de tipos, etc.

Especificación de Tipo de Dato Elemental

Los lenguajes de programación suelen tener un conjunto de tipos de datos primitivos elementales, como son: entero, real, carácter, booleano, de enumeración y apuntador.

Se definen dos tipos de operaciones sobre los tipos de datos:

✍ Las **operaciones primitivas** son las que se especifican como parte de la definición del lenguaje.

✍ Las **operaciones definidas por el programador**.

Los objetos de datos requeridos en una operación se denominan **argumentos**.

La **aridad** de una operación corresponde al número de argumentos que utiliza.

El **dominio** de una operación define el conjunto de posibles argumentos de entrada que pueden utilizarse.

El **intervalo** de una operación define el conjunto de posibles resultados que puede producir.

La **acción** de una operación define los resultados que se producen para un conjunto dado de valores.

Ejemplo: La operación binaria de suma entera se puede representar como:

+ : entero x entero ? entero

Ejemplo: La operación raíz cuadrada se puede representar como:

SQRT : real ? real

Una especificación precisa de la acción de una operación requiere más información que únicamente los tipos de datos de los argumentos. Esta especificación se ve dificultada por:

✍ Operaciones que no están definidas para ciertas entradas.

✍ Argumentos implícitos.

✍ Efectos colaterales, esto es, resultados implícitos.

✍ Auto modificación, esto es, sensibilidad historial.

La tabla 2.1 muestra algunos de los tipos de datos primitivos elementales más comunes de los lenguajes de programación utilizados a lo largo del curso.

Especificación de Tipo de Dato Estructurado

Los lenguajes de programación suelen permitir al programador definir nuevos tipos de datos en base a otros tipos ya existentes. Estos tipos se denominan estructurados o complejos. Algunos tipos de datos

estructurados son provistos junto con el lenguaje. Todos los tipos de datos que forman parte de la especificación de un lenguaje se denominan predefinidos. Al resto se les denomina “definidos por el usuario” (o también “definidos por el programador”, pues el usuario del lenguaje es el programador).

Un subtipo de datos se define a partir de un tipo de dato o súper-tipo, donde su conjunto de valores posibles es un subconjunto del súper-tipo.

Tabla 2.1. Tipos de datos primitivos elementales

Tipo de datos	C++	Java	C#
Booleanos	Bool	boolean	Bool
Enteros	char, unsigned char short, unsigned short int, unsigned int long, unsigned long	byte short int long	sbyte, byte char (UNICODE) short, ushort int, uint long, ulong
Reales	Float double long double	float double	float double decimal
Caracteres	char, wchar_t	char (UNICODE)	char (UNICODE)
String	string	string	String
Enumerados	Enum		Enum

Los tipos de datos referencias son aquellos cuyos objetos de datos apuntan a otros objetos de datos. Cuando lo que guarda una referencia es una dirección de memoria, se le llama puntero.