ÍNDICE

CÓDIGO

3.3. TIPOS DE DATOS

El tipo de una variable indica qué *valores* puede tener y qué *operaciones* pueden realizarse sobre ella. La motivación para la definición de tipos en los lenguajes de programación surge a diferentes niveles:

- **Nivel de la máquina**. Los valores soportados directamente por la máquina son clasificados en *tipos primitivos*, tales como entero, carácter, real y Booleano. Por ejemplo, dado que la instrucción máquina para realizar una operación entre enteros es diferente a la instrucción para realizar esa misma operación entre reales, los compiladores necesitan tener información acerca del tipo de los datos, a fin de poder generar el código máquina correspondiente a las operaciones.
- **Nivel del lenguaje**. Además de los tipos primitivos, los lenguajes de programación soportan *tipos estructurados*, tales como arrays, estructuras, listas, etc. que son construidos a partir de los tipos primitivos de datos. La construcción del lenguaje que permite definir tipos estructurados se denomina *constructor de tipo*.
- **Nivel del programador**. Los tipos definidos por el programador son agrupaciones de datos y funciones. Un ejemplo son los *tipos enumerados*. Otro las *clases*.

El tipo de dato de la variable condiciona el número de posiciones de memoria que son necesarias para almacenarla. Por ejemplo, el espacio de almacenamiento necesario para guardar una variable de tipo Booleano, cuyo valor puede ser *true* o *false*, en general no será el mismo que el necesario para guardar una variable de tipo real. Asimismo, no se necesita el mismo espacio para almacenar una variable simple, que para almacenar una variable vectorial.

El valor de la variable se almacena en memoria codificado como una o varias palabras binarias, las cuales normalmente se almacenan en un conjunto de posiciones de memoria consecutivas.

En función del tipo de dato de la variable, la palabra o palabras binarias que representan su valor se interpreta de una forma u otra. Por ejemplo, una misma secuencia de bits puede ser interpretada como un entero o como un carácter. Dependiendo del tipo de dato de la variable, la interpretación será una u otra.

3.3.1. Tipos primitivos

Los tipos de datos cuya definición no se basa en otros tipos de datos se denominan *tipos primitivos*. Los tipos primitivos soportados por los primeros lenguajes de programación eran únicamente numéricos. Correspondían con los tipos *número entero* y *número real*. Los lenguajes de programación actuales ofrecen una amplia gama de tipos primitivos de datos, entre los cuales típicamente están los descritos a continuación.

– **Entero**. Los números enteros son soportados directamente por el hardware del ordenador, ya que pueden ser codificados en base binaria como una cadena de bits, en la cual el bit más significativo representa el signo. Cuanto mayor sea la longitud (el número de bits) del tamaño de palabra que se usa para almacenar los números enteros, mayor será el rango de valores que puedan tomar los enteros.

  Algunos lenguajes soportan diferentes tipos de números enteros. La diferencia entre ellos es el número de bits de la palabra empleada para almacenar los datos. Por ejemplo, Java soporta los cuatro tipos de enteros siguientes:

Tipo	Número de bits	Rango
`long`	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
`int`	32	-2.147.483.648 a 2.147.483.647
`short`	16	-32.768 a 32.767
`byte`	8	-128 a 127

– **Real en coma flotante**. Los números reales se representan en formato de *coma flotante*. Es decir, como un coeficiente multiplicado por 10 elevado a un exponente entero. El coeficiente es un número real compuesto por un único dígito entero, seguido de una coma y un número fijo de dígitos decimales. Por ejemplo, en el número en coma flotante 2.3462e-3 el coeficiente es 2.3462 y el exponente es -3.

  Como en el caso de los tipos enteros, los lenguajes suelen soportar varios tipos real, de modo que el programador pueda seleccionar aquel cuyo rango y precisión en el coeficiente (esto es, número de decimales del coeficiente) se ajusta más a las necesidades de la aplicación. Por ejemplo, Java soporta dos tipos reales: simple precisión (**float**) y doble precisión (**double**).

Tipo	Número de bits	Rango
`double`	>64	1.7e-308 a 1.7e+308
`float`	>32	3.4e-38 a 3.4e+38

En aplicaciones de cálculo numérico es recomendable trabajar con reales en doble precisión, ya que permiten representar los números de manera más precisa. Esto es debido a que los números en doble precisión tienen al menos el doble de dígitos fraccionales en el coeficiente que los números en simple precisión. Por otra parte, en algunos casos trabajar en doble precisión es más rápido que en simple precisión, ya que los procesadores modernos están optimizados para trabajar en doble precisión. Normalmente las funciones matemáticas transcendentes (trigonométricas, raíz cuadrada, etc.) devuelven valores del tipo real con doble precisión.

– **Booleano**. Los datos de este tipo pueden tomar dos posibles valores: *true* y *false*. ALGOL 60 fue el primer lenguaje que introdujo este tipo de dato, el cual es soportado por casi todos los lenguajes que se desarrollaron posteriormente. Una excepción es el lenguaje C, en el cual las condiciones lógicas se representan mediante expresiones numéricas: cuando una expresión numérica se interpreta como una condición lógica, vale *true* si la expresión es distinta de cero y *false* si vale cero.

  Aunque el ordenador internamente podría representar un valor Booleano mediante un único bit (por ejemplo, si el bit vale 0 representa *false*, y si vale 1 representa *true*), normalmente el dato Booleano se almacenan ocupando una posición de memoria, ya que es la unidad de memoria más pequeña que puede ser direccionada eficientemente.

– **Caracter**. Este tipo de datos se usa para almacenar caracteres. El conjunto de caracteres estándar conocido como ASCII (*American Standard Code for Information Interchange*) tiene 128 caracteres, y el conjunto extendido de 8 bits, ISO Latin-1, tiene 256 caracteres (dado que $2^8 = 256$). Este último es el soportado por C y C++. Así pues, los datos de tipo carácter son almacenados en palabras de 8 bits en C y C++.

  Otro conjunto estándar de caracteres es *Unicode*, el cual define un conjunto completo e internacional de caracteres que incluye todos los caracteres que pueden encontrarse en todas las lenguas de la humanidad, incluyendo los caracteres de los alfabetos latín, griego, árabe, cirílico, hebreo y muchos más. Java fue el primer lenguaje en soportar este código, para lo cual los datos de tipo carácter en Java son almacenados en palabras de 16 bits.