Together for Tomorrow!
Enabling People

Documentación técnica:

Estructuras de control y de datos

#TecnologíaConPropósito

Índice

1. Tipos de datos e información estructurada

Cualquier programa puede considerarse como un conjunto de operaciones que se realizan en un orden determinado sobre una serie de datos. Cada lenguaje de programación define los tipos de datos permitidos y las operaciones disponibles sobre ellos. La idea de dato, por tanto, está muy ligada a las operaciones concretas que se pueden realizar sobre él. Un ejemplo sería un dato numérico que se puede representar como entero o real, internamente se guardan de forma diferente y las operaciones que se pueden ejecutar sobre cada tipo también son distintas.

Un dato es, por tanto, un conjunto de símbolos que representa un valor alfanumérico. Por ejemplo, un número de teléfono, la altura de una persona o el número del carnet de identidad. El término **información** se refiere a un conjunto de datos, ya procesados, organizados y que están relacionados. Por ejemplo, son datos individuales el nombre de una persona y su fecha de nacimiento, su salario y el nivel de estudios. Sin embargo, se pueden agrupar en una nómina y, organizados por fecha de nacimiento y salario, ofrecen la distribución de sueldos en función de la edad del trabajador. Organizando los mismos datos por nivel de estudios y salario, se obtendrá la distribución de salarios en función de la formación de los empleados.

La representación digital de los datos es necesaria para poder procesarlos y organizarlos. En las computadoras, la información se divide en trozos y cada trozo se representa numéricamente de forma individual, y lo que se maneja al final es ese conjunto de números. En los ordenadores toda la información está almacenada digitalmente: los números, los textos, los audios, los vídeos. Cada carácter de texto (cada letra o símbolo) se representa por un código numérico. Un conjunto de códigos muy conocido y utilizado es el código ASCII (American Standard Code for Information Interchange), pero tiene varios inconvenientes que, aunque codifica muy bien el idioma inglés, sin embargo, el conjunto de caracteres es muy limitado y le faltan muchos caracteres acentuados utilizados en otros idiomas. Por ello, más recientemente se utilizan otros códigos más completos y extensos como **Unicode** (UTF-8 y UTF-16) que pueden representar cualquier carácter incluso de los idiomas chino, japonés y coreano, así como cualquier mensaje codificado con ASCII.

Para representar números o colores, resulta conveniente por su simplicidad utilizar un sistema básico de 2 dígitos: 0 y 1, que se llaman **bits**. Estos valores se

representan de forma muy fácil con un circuito electrónico básico y por ello son la base de la representación de los datos en los sistemas informáticos como se verá más adelante.

Los **tipos de datos primitivos** son los mismos en todos los lenguajes de programación:

1. Numéricos: pueden ser enteros o reales. Los enteros se pueden representar a su vez de cuatro formas diferentes (byte, short, integer y long) según la cantidad de espacio en memoria que se quiera utilizar para guardarlos. El tipo escogido dependerá del valor máximo que necesitemos para ese dato. Los reales se representarán con coma flotante (para indicar la parte decimal) con 7 dígitos significativos (float) o 15 dígitos significativos (double).
2. No numéricos: serán de tipo carácter o lógicos (llamados booleanos). El tipo carácter se almacena en un símbolo alfanumérico y se identifica como char. Cada valor de tipo char almacena un carácter simple Unicode, y pueden ser letras minúsculas, letras mayúsculas, signos de puntuación, números, símbolos especiales y algún carácter de control (nueva línea o intro, por ejemplo). El tipo lógico se conocen también como booleanos por su término en inglés. Tienen dos posibles valores: verdadero (true) o falso (false).

Las variables, son nombres de huecos de memoria que se utilizan en programación para guardar datos y deben ser declaradas en el programa antes de que se utilicen. Habitualmente, al mismo tiempo que se declaran, se suelen inicializar a un valor determinado (por ejemplo, 0) para partir de un primer valor conocido y que sirva de referencia según el programa se va ejecutando. Un ejemplo de declaración e inicialización de tres variables numéricas distintas en Java sería:

```
int total=0, cuenta=20;

float precioItem = 25,4;
```

1. Tipos de datos e información estructurada

Las constantes son, en programación, entidades similares a las variables, con la diferencia de que su valor no cambia nunca y si se intentan modificar en el programa se produce un error.

2. Estructuras de control

Los programas resuelven problemas utilizando una serie de instrucciones organizadas en forma de algoritmos, de forma que le indican al computador qué hacer en cada momento.

Las **instrucciones de asignación** sirven para dar valores a las variables, colocando en su interior un valor determinado. Es importante notar que el signo = no representa una igualdad en programación, sino el efecto de colocar un valor en la variable que aparece a su izquierda. Así, la instrucción:

acumulado = 3;

Guarda un valor de 3 en la variable "acumulado". La sentencia:

acumulado = acumulado + 5;

produciría que la variable "acumulado" guardara un valor de 8.

Las **instrucciones de entrada/salida** recogen datos desde el teclado o desde un fichero de la memoria y los muestran en la pantalla del ordenador.

Las **instrucciones de control** pueden variar el flujo normal de instrucciones, por medio de bucles, iteraciones o condiciones:

- Bucles: son útiles cuando se quiere repetir la misma instrucción o bloque de instrucciones varias veces en el mismo programa. Pueden repetirse varias veces mientras se cumpla una determinada condición.

  Con el bucle tipo **while**, se repite un bloque de sentencias mientras se cumple una determinada condición, es decir, mientras sea verdadera. En el momento en que la condición se vuelve falsa, ya no se repite el bloque de sentencias y el programa avanza hasta la siguiente instrucción.

  while (condicion){

  //Bloque de sentencias

  }

  Con el bucle tipo **do … while**, se asegura que el bloque de sentencias se ejecuta al menos una vez, puesto que la condición de mantenimiento del bucle se evalúa al final de la primera ejecución. Su sintaxis sería:

```
do {
//Bloque de sentencias
} while (condicion);
```

- Iteraciones: el bloque de sentencias se ejecuta un número fijo de veces. Es un tipo de bucle que está controlado por un contador, en cada vuelta se incrementa el contador de uno en uno, y el bucle se para cuando el contador llega a su valor límite. Este tipo de iteración se utiliza cuando se conoce con precisión el número de veces que debe ejecutarse el bloque de sentencias.

```
for (inicialización; condición; incremento) {
// Bloque de sentencias
}
```

- Condiciones: se llaman también sentencias de selección o decisiones ya que evalúan una condición y luego, según el resultado, deciden qué bloque de sentencias – de entre varias – ejecutan. La palabra reservada if (si condicional, en inglés) se utiliza de forma conjunta con else, de la siguiente forma:

```
if (condicion) {
// Primer bloque de sentencias
} else {
// Segundo bloque de sentencias
}
```

Si la variable condición es verdadera se ejecuta el primer bloque de sentencias, si es falsa, el segundo.

SAMSUNG