Conceptos generales sobre las variables y la memoria

En matemática, a un objeto o expresión que representa un valor de un determinado tipo le asignamos un nombre que llamamos variable. Las variables tienen pues un valor de un tipo (por ejemplo, si definimos x = 3, el nombre x queda asociado al valor 3 de tipo numérico y, en el contexto de esa definición, son expresiones equivalentes x < 2 y 3 < 2).

En computación, un objeto *tiene algo más*: una representación interna (la secuencia de ceros y unos que representa un valor numérico en representación binaria) y un lugar en la memoria del computador. Un objeto en computación *(data object)* es un contenedor para valores, es decir, un lugar en el computador donde los valores pueden ser depositados y recuperados. Un objeto está caracterizado por un conjunto de atributos, el más importante de los cuales es su tipo. Los atributos determinan la cantidad y el tipo de valores que un objeto puede contener y también determinan la organización de esos valores. Un valor puede ser un carácter, un número, etc. y se representa por una secuencia particular de ceros y unos en la memoria del computador. Pueden existir muchas copias del mismo valor en la memoria, esto significa que la misma secuencia de ceros y unos se repetirá.

Por ejemplo, en la figura se ilustra, a la izquierda, las instrucciones del texto del programa y, a la derecha, el objeto como contenedor para valores *(data object)*, cuya dirección de memoria es 23AC4001. Al momento de la asignación n=17, el valor 17 de la variable n se guarda como secuencia de ceros y unos en el contenedor.

En el texto del programa	En la memoria del computador	
…		
n = 17	23AC4001	000000000010001
...		

Una de las operaciones que altera el valor de un *data object* es la asignación: por ejemplo en el caso de arriba, luego de la instrucción n = n + 1 el valor del contenedor de valores 23AC4001 será 18 (10010 en representación binaria).

En el texto del programador	En la memoria del computador	
n = 17	23AC4001	000000000010001
…		
n = n + 1;	23AC4001	000000000010010
...		

En el curso de la ejecución de un programa, algunos objetos existen desde el inicio, otros son creados dinámicamente durante la ejecución, otros son destruidos, otros persisten hasta el final de la ejecución. O sea que un objeto tiene un *tiempo de vida* durante la ejecución y mientras dure, puede

ser usado para almacenar valores. Un objeto puede ser simple como 23AC4001 arriba o puede ser estructurado, por ejemplo, una secuencia de celdas de memoria para almacenar una lista de valores.

A diferencia de la matemática donde, en determinado contexto, un objeto está asociado a un valor y un tipo, en computación un objeto puede tener *asociaciones* a diferentes atributos: por ejemplo, el objeto arriba, está asociado a un lugar en la memoria (23AC4001), un nombre (n), un valor en el texto del programa (17), su valor binario interno (10001), un tipo (numérico). A su vez, otras asociaciones pueden ser creadas durante la ejecución, por ejemplo, si usamos alguno de los atributos como argumentos (o parámetros) de un sub programa (o programa).

Memoria

Los datos en la memoria se representan en codificación binaria (secuencia de 0 y 1), donde 8 bits constituyen un byte. La memoria se considera como una tabla indizada cuyas celdas son datos de tamaño de un byte (8 bits), que se acceden a través del índice correspondiente. En el ejemplo arriba, si el programa necesita el dato 17, accede al mismo a través del índice 23AC4001 de la tabla memoria. Un dato de un programa puede ser por ejemplo una secuencia de datos, en cuyo caso ocupa varias celdas en la tabla (la memoria). En la memoria no solo se almacenan los datos, sino también *el programa*, que especifica las operaciones a realizar con los datos.

¿Quién realiza las operaciones?

En un computador, además de la memoria, existe una **Unidad Central de Proceso** (**CPU** por sus siglas en inglés) que es la que se encarga de ejecutar las operaciones.

¿Cómo sabemos lo que hace el computador?
Si tenemos un programa como este:

```
# factorial : N → N
def factorial (x):
    fact = 1;
    for i in range(2, x+1);
       fact = fact * i
return(fact)

n = input (“ingrese numero natural: “)
print “el resultado es “, factorial(n)
```

Si lo ejecutamos, ¿cómo sabemos el resultado?

Existe también un subsistema de **Entrada/Salida** (**I/O** por sus siglas en inglés) que realiza la comunicación con el programador, el usuario, u otros dispositivos externos, a través de un BUS. Las sentencias input y print operan en dicho subsistema. El valor ingresado en n con “input” se pasa a x al aplicar la función a n en factorial(n), con ese valor se realizan las operaciones de la función, se devuelve el resultado en return (fact) y se imprime con “print”.

Memoria con el valor 17 en la celda 23AC4001