Tema 1: Datos y algoritmos

La información ha sido siempre uno de los recursos más preciados, en la era actual donde los ordenadores y la informática, en general, juegan un papel muy importante, esta información es más fácilmente accesible.

El mundo empresarial pretende conseguir no sólo la mayor cantidad de datos, sino, sobre todo, una rápida respuesta y una organización adecuada de tal magnitud de información. En la velocidad de obtención influye primordialmente la potencia del computador, el hardware, (aunque también importa el diseño y estructuración de los datos). Mientras que en el modo de obtención, organización y distribución tiene mayor peso el software utilizado. Un programa mejor desarrollado obtendrá datos más ajustados, más fiables, flexibles y claros.

En los siguientes capítulos se estudiará el modo de obtener la información deseada a partir de unos recursos primarios, es como un sistema de producción, se tiene una entrada y mediante un proceso de refinado o transformación se obtiene el producto manufacturado.

1.1. Datos y tipos de datos.-

Puede decirse que el proceso de desarrollo de un programa es el conjunto formado por unos datos de entrada, una caja negra que transforma o elabora esos datos y unos datos finales a la salida.

Hasta ahora, se ha estado tratando la información en términos generales, pero es necesario detallar que se entiende por dato desde el punto de vista del tratamiento informático. Puede definirse un dato como el elemento de información procesada o elaborada en un formato específico con significado.

Desde el punto de vista del ordenador, todos los datos se traducen en ristras de números binarios (bits 0's y 1's) ordenados de una forma determinada para poder expresar su significado y por tanto ser considerados realmente como tales.

Para el programador, estos son fundamentalmente de dos tipos:

> **Numéricos**: Se pueden representar en números enteros y reales. Los enteros corresponden a números completos, es decir no tienen componente decimal y pueden ser tanto positivos como negativos, estando su rango normalmente en -32768 a 32767. Algunos ejemplos pueden ser:
>
> ```
> 1995          -23          5678
> 0             145          -345
> ```
>
> Los números reales llevan siempre asociado un punto decimal, pudiendo tomar cualquier valor dentro de la recta real, tanto positivos como negativos. Así por ejemplo:

```
678.9          -45.0          34.0
345.6          23.4           -6.9
```

También es posible utilizar la notación científica o coma flotante cuando se necesitar representar datos de muy pequeño valor o muy grande por ejemplo:

```
1.2e-23 = 1.2 10 -23          1.69e-19 = 1.6 10 -19
6.67e34 = 6.67 10 34          2.12e10 = 2.12 10 10
```

Alfanuméricos:Se consideran como caracteres de cualquier tipo, incluido dígitos, que vienen codificados por un valor según una tabla de códigos de traducción (ASCII, EBCDIC, ...), por ejemplo:

```
"ABC"          "$%&"          'a'
```

Estos dos tipos de datos serían los elementales, todos los demás se forman con los anteriores. Imagínese que se desea representar la información que recoge la mejor marca de un corredor de los cien metros lisos. Se necesitarían dos tipos de datos:

Un dato de tipo número real que exprese la marca conseguida (p.ej.: 9,82)

Un dato de tipo cadena de caracteres para el nombre del atleta (p.ej.: "Ben Johnson")

El primer dato es una especificación del numérico (en concreto los reales), mientras que el segundo es un conjunto ordenado de datos alfanuméricos, llamado cadena de caracteres. Ambos podrían formar un tipo de datos más complejo que se llamase 'velocista', el cual tendría unos valores determinados para cada individuo.

Con el ejemplo anterior se ha utilizado un nuevo tipo de datos denominado *cadena de caracteres* (también conocido como *string*). Una cadena de caracteres es una sucesión finita de los mismos, éstos pueden ser de diferentes características, pudiendo contener:

- Espacios en blanco.
- Letras alfabéticas: A,B,C,...
- Dígitos: 0,1,2,3,...
- Símbolos: %, $, (,), ...

Para distinguir el tipo de datos cadena de caracteres hay que escribirlo entre comillas, p.ej.: "Carl Lewis".

1.2. Variables y constantes.-

Los datos en un ordenador necesitan ser manejados mediante herramientas que

permitan almacenarlos en memoria, leerlos, operar con ello, etc. Para tal cometido son necesarios los conceptos de identificador, variable, constante, operador y expresiones.

En lenguajes de alto nivel se necesita utilizar nombres para identificar los objetos que se deseen manupilar: variables, constantes, procedimientos, etc. Los identificadores son los nombres que se designan a dicho propósito. Estos permitirán elegir nombres significativos que sugieran lo que representan. Los identificadores se construyen de acuerdo a las reglas de sintaxis del lenguaje específico. Estas básicamente se pueden enunciar de la siguiente forma:

- Deben comenzar con una letra (A a Z).
- El segundo carácter y posteriores puede ser letra o dígito, admitiéndose también el carácter de subrayado, pero el espacio en blanco.
- No hay límites en cuanto su extensión.
- No se pueden usar como identificador palabras claves propias del lenguaje de programación que constituyen la base de sentencias, funciones y ordenes.

Constantes

Durante la ejecución de programas hay ciertos valores que no cambian, tales valores se llaman constantes. Por tanto se pueden definir las constantes como datos cuyo valor no cambia durante la ejecución del programa. Existen tantos tipos de constantes como tipos de datos, tales como constantes enteras, reales, carácter, etc.

En determinados lenguajes, como Pascal o C, existe la posibilidad de que ciertas constantes de uso muy frecuente puedan recibir un nombre (identificador) con el que se les reconocerá a lo largo de todo el programa. Por ejemplo:

```
PI = 3.141592           constante pi
e = 2.71828             base de los logaritmos naturales
IVA = 16                Valor del I.V.A.
Director = “Miguel”     Nombre del director
```

Los nombres de estas constantes serían *PI, e, IVA, Director*. Esta declaración de constantes hace que su valor se almacene en una posición de memoria.

Variables

Las variables, al contrario que las constantes, son objetos que pueden cambiar su valor durante la ejecución del programa. Para nombrarlas se utilizan los identificadores que deben seguir las reglas mencionadas anteriormente, si bien es buena práctica de programación utilizar nombres de variables significativos que sugieran lo que representan ya que eso hará los algoritmos y programas más legibles y fáciles de comprender. Así por ejemplo, si se desea utilizar una variable para guardar el número de personas hospedadas en un hotel se podría poner

como nombre de la variable: *num_personas*. Si se considerase, sin embargo los identificadores *entero* o *numero* proporcionan menos información en cuanto a lo que almacenan, con lo cual si se tuviesen que manejar distintos números para distintas situaciones, se encontraría una dificultad de comprensión e identificación.

Al igual que en caso de las constantes, la declaración de las variables origina una reserva de una posición de memoria del ordenador que es etiquetada con el correspondiente identificador. En el caso de las variables, sin embargo, el contenido de dicha posición de memoria, y por tanto el valor de la variable, se puede cambiar. Debe tenerse presente que normalmente no se almacena ningún valor inicial en la variable, teniendo esta un valor no definido, lo cual implica que antes de usarla se debe asignar un valor inicial.

El tipo de una variable puede ser uno de los tipos de datos descritos anteriormente. Debiéndose declarar el tipo de la variable en el momento de su definición en la cabecera del programa o algoritmo.

1.3. Operadores, funciones y expresiones.-

A los tipos de datos se les pueden aplicar operaciones para obtener otros datos, las operaciones se expresan mediante operadores, éstos son símbolos que significan una determinada operación dependiendo del lenguaje de programación que se utilice (generalmente tienen el mismo significado).

Existen muchos tipos de operadores clasificados básicamente en matemáticos, lógicos, relacionales y de asignación. Se estudiarán con más detalle en los capítulos de programación en C, ahora tan sólo se verán unos pocos para realizar algunos ejemplos.

- Matemáticos:
  - adición (+): `a + b`
  - resta (-): `a - b`
  - producto (*): `a * b`
  - división (/): `a / b`

- Lógicos:
  - AND lógico (Y): `a Y b`
  - OR lógico (O): `a O b`
  - NOT lógico (NO): `NO a`

- Relacionales:
  - igual que (==): `a == b`
  - mayor que (>): `a > b`
  - menor que (<): `a < b`
  - mayor o igual que (>=): `a >= b`
  - menor o igual que (<=): `a <= b`

- Asignación:
  - asignar (=) : `a = b` Asignar el valor de la variable '*b*' a la variable '*a*'

Los operadores pueden ser monádicos, si se aplica a un único operando o valor por ejemplo negación (*NO a)*, signo negativo *(-a*) o diádicos, si se aplican a dos operandos por ejemplo suma (*a* + *b*), resta (*a* - *b*.).

Además de los operadores , existen una serie de funciones predefinidas que consisten en operaciones complejas aplicadas a uno o más argumentos o parámetros de la función. Por ejemplo, *SQRT(x)* es la función que devuelve la raíz cuadrada del número x. Escribir *SQRT*(4) sería lo mismo que escribir la constante 2.

El conjunto formado por la combinación correcta de operandos, funciones y operadores se denomina **expresión**. Dependiendo de la forma de asociar estos elementos se pueden construir dos tipos de expresiones:

Expresiones básicas:

operando	*operador*	*operando*	*a + b*
operador	*operando*		NO a

Expresiones complejas:

expresión	*operador*	*operando (a + b) + d*
operando	*operador*	*expresiónsin (a) / (c +b)*
expresión	*operador*	*expresión(c + d) * (a -b)*

Se puede observar como las expresiones se pueden combinar entre ellas para formar expresiones más complejas. Véanse una serie de ejemplos:

```
7 + 20 <--> 27
(31 + 5)/2 * 3 <--> 36/2*3 <--> 18*3 <--> 54
```

(las expresiones se resuelven de izquierda a derecha a no ser que existan paréntesis que se resuelven antes)

longitud de circunferencia de radio `4 <--> 2*PI*radio`

1.4. Algoritmos.-

El término algoritmo es en parte similar a los términos receta, proceso, método, técnica, etc. Además de ser un conjunto finito de reglas que dan lugar a una

secuencia de operaciones para resolver un problema específico, debe cumplir las siguientes condiciones:

- Tiene que acabar siempre tras un número finito de pasos.

- Cada paso debe definirse de modo preciso; las acciones a realizar han de estar especificadas para cada caso rigurosamente y sin ambigüedad.

- Debe existir un conjunto especificado de objetos, cada uno de los cuales constituye los datos iniciales de un caso particular del problema. A este conjunto de datos se le llama entrada del algoritmo.

- Debe existir un conjunto de objetos, cada uno de los cuales constituye la salida que debe obtener el algoritmo para los diferentes casos particulares. A este conjunto de datos se les llama conjunto de salida.

- Todas las acciones que realice deben ser lo bastante básicas para poder ser efectuadas de modo exacto en un intervalo de tiempo finito por el procesador que ejecute el algoritmo.

En definitiva un algoritmo describe el método mediante el cual se realiza una tarea. Este consiste en una secuencia de instrucciones, realizadas adecuadamente, las cuales dan lugar al resultado deseado. Debe estar expresado de forma que el procesador, el ejecutor del algoritmo, lo entienda para poder ejecutarlo, es decir, el procesador debe ser capaz de interpretar el algoritmo, lo que significa que debe ser capaz de entender lo que significa cada paso, y llevar a cabo la operación correspondiente.

Desde el punto de vista informático un algoritmo es un proceso que ofrece una solución a un problema determinado si ésta existe y además, siempre obtendrá los mismos resultados para valores idénticos a la entrada tantas veces se repita. En este proceso existen pasos intermedios en los cuales pueden surgir otros datos que se llaman intermedios y que sólo interesan dentro del proceso algorítmico.

Un ejemplo de algoritmo sería:

```
ALGORITMO_SUMA
     LEER A
     LEER B
     CALCULAR C = A + B
     ESCRIBIR RESULTADO
FIN_SUMA
```

Como puede observarse se compone de cuatro instrucciones o pasos que escriben la suma de dos números determinados A y B. Este algoritmo necesitaría, como se observa en la ilustración, dos datos de entrada A y B y obtendría una salida C.

A continuación se muestra un el algoritmo que calcula el área de un triángulo en función de su base y su altura.

```
ALGORITMO_AREA
     LEER ALTURA
     LEER BASE
     CALCULAR AREA = (ALTURA*BASE)/2
     ESCRIBIR RESULTADO
FIN_AREA
```

Los datos que se pasan a un algoritmo suelen ser bastante específicos, pero generalmente la información se presenta en su estado natural como la materia prima, en modo rudo. Se necesitará, por tanto, estructurarla para que pueda ser manejada y presentada con el formato deseado independientemente de como se origine.