

# Estructuras Básicas de Algoritmos y de Datos I

José Otero

<sup>1</sup>Departamento de informática  
Universidad de Oviedo

1 de octubre de 2008

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

- El valor de los indicadores se codifica en binario.
- Se almacenan en la memoria del computador.
- **objeto de almacenamiento u objeto:** espacio que ocupa se denomina.
- **Constante:** objeto cuyo contenido (valor del indicador) no puede ser cambiado.
- **Variable:** objeto cuyo contenido puede ser cambiado.
- Cada objeto tiene uno o más nombres (identificadores) y pertenece a un tipo de datos.

- 1 Concepto de variable y constante
- 2 Tipos elementales**
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

- Subconjunto de los enteros.
- Su cardinal depende del computador y del lenguaje de programación.
- Las operaciones aritméticas no producen decimales.
- El proceso de cálculo se interrumpe si se produce un resultado que no pertenece a ese subconjunto.
- Los operadores aritméticos aplicables a enteros son  
 $+, -, *, /, \%$
- Son todos binarios (necesitan dos operadores). Además el  $-$  representa el signo negativo, en este caso es unario.

- La división produce resultados de tipo entero (sin decimales).
- Ejemplos:
  - $5/2$  vale 2
  - $3/7$  vale 0
- El operador módulo (%) produce el resto de dividir los enteros a los que se aplica. Se cumple la igualdad:
  - $(m/n) * n + (m \% n) = m$
- Ejemplos:
  - $5 \% 7$  vale 5
  - $11 \% 3$  vale 2

Existen distintos subconjuntos de enteros:

- Distinto cardinal (rango).
- Con o sin signo (abarcando solo los positivos o positivos y negativos).

| Denominación                    | mínimo      | máximo     |
|---------------------------------|-------------|------------|
| <code>short int</code>          | -32768      | 32767      |
| <code>int</code>                | -2147483648 | 2147483647 |
| <code>unsigned short int</code> | 0           | 65535      |
| <code>unsigned int</code>       | 0           | 4294967295 |

En algunos computadores existen los tipos:

- `long int` con mayor rango que `int`.
- `unsigned long int` con mayor rango que `unsigned int`.

Cuanto mayor es el rango, más memoria ocupan los valores y más tardan en realizarse las operaciones matemáticas.



Subconjunto de los reales. Operaciones con decimales. Número finito de dígitos. Redondeo. Los operadores aritméticos definidos sobre los reales son los siguientes:

- + suma
- - resta y cambio de signo
- \* multiplicación
- / división

- El tipo real en C++ admite variaciones con distinto grado de precisión y rango:

| Denominación | mínimo       | máximo      | cifras significativas |
|--------------|--------------|-------------|-----------------------|
| float        | $-10^{38}$   | $10^{38}$   | 7                     |
| double       | $-10^{308}$  | $10^{308}$  | 15                    |
| long double  | $-10^{4932}$ | $10^{4932}$ | 19                    |

- Cuanto mayor es el rango, más memoria ocupan y más tardan en realizarse las operaciones.

El tipo booleano se declara con la palabra reservada `bool`.  
Sólo puede tomar dos valores:

- `true` verdadero.
- `false` falso.

Los operadores que se aplican a valores de tipo booleano son:

- `!` negación (unario, se escribe *antes* del operando).
- `||` o, disyunción (binario).
- `&&` y, conjunción (binario).

| x     | y     | x  y  | x&&y  | !x    |
|-------|-------|-------|-------|-------|
| true  | true  | true  | true  | false |
| true  | false | true  | false |       |
| false | true  | true  | false | true  |
| false | false | false | false |       |

Existen operadores que se *aplican* a tipos distintos del booleano pero *producen* resultados de tipo booleano. El resultado es `true` si la relación que se expresa es verdadera, `false` en caso contrario.

- `<` menor
- `<=` menor o igual
- `>` mayor
- `>=` mayor o igual
- `==` igual
- `!=` distinto

Ejemplos:

- `7.3<=5` es `false`, `4!=7` es `true`

**NOTA:** es mala práctica de programación aplicar `==` a reales.

Es el conjunto de los caracteres que se pueden usar en el computador. Se declara mediante la palabra reservada `char`.

- El más usado es el ASCII.
- Existen caracteres imprimibles (letras, dígitos, signos matemáticos, de puntuación ...)
- Y también no imprimibles o de control (retornos de carro, tabuladores, ...)
- Existe una ordenación que se puede usar con los operadores relacionales.
  - Las letras están ordenadas alfabéticamente.
  - Las mayúsculas preceden a las minúsculas.
  - Los dígitos están ordenados de menor a mayor.

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples**
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

Un `short int` se almacena en dos bytes, un `int` en cuatro. Los enteros `unsigned` se codifican en binario. El resto se codifica utilizando la “codificación en complemento a dos”.

- El mayor `short int` es  $0111111111111111 = 2^{15} - 1$
- El mayor `int` es  $2^{31} - 1$ .
- Números negativos: se invierte bit a bit la codificación en binario de su valor absoluto y se suma 1 al resultado.

Ejemplo:

- $7 = 4 + 2 + 1 = (0000000000000111)_2$   
 $-7 = (1111111111111000)_2 + 1 = (1111111111111001)_2$

- El menor `short int` es  $-2^{15}$  y
- El menor `int`  $-2^{31}$ .
- En los números negativos el bit más a la izquierda es 1 y en los positivos es 0.
- Ventajas de `Ca2`: La suma y la resta se realizan de la misma forma que en el caso de los enteros sin signo:
  - $-7 + 2 = (1111111111111001)_2 + (00000000000010)_2 = (1111111111111011)_2 = -(000000000000101)_2 = -5$



- Por una parte se codifica la **mantisa** del número (toda entera o toda fracción).
- Por otro lado el **exponente**.
- El valor del número es la base (normalmente 2) elevada al exponente  $\times$  mantisa.
- El uso de punto decimal es innecesario.
  - Por ejemplo 6,3125, tomando diez bits para la mantisa (toda entera) y seis para el exponente, representando este en complemento a 2, sería: 0001100101 110111
- El signo del número real se suele codificar aparte, siendo 1 si el número es negativo y 0 si es positivo.
- Un `float` ocupa 32 bits, un `double` 64 y un `long double` 96.

- Un carácter ocupa un byte.
- Un carácter se codifica mediante un número positivo.
- La tabla ASCII codifica 256 caracteres. Fragmento:

| Char  | Dec | Oct  | Hex  | Char | Dec | Oct  | Hex  | Char | Dec | Oct  | Hex  | Char | Dec | Oct  | Hex  |
|-------|-----|------|------|------|-----|------|------|------|-----|------|------|------|-----|------|------|
| (nul) | 0   | 0000 | 0x00 | (sp) | 32  | 0040 | 0x20 | @    | 64  | 0100 | 0x40 | `    | 96  | 0140 | 0x60 |
| (soh) | 1   | 0001 | 0x01 | !    | 33  | 0041 | 0x21 | A    | 65  | 0101 | 0x41 | a    | 97  | 0141 | 0x61 |
| (stx) | 2   | 0002 | 0x02 | "    | 34  | 0042 | 0x22 | B    | 66  | 0102 | 0x42 | b    | 98  | 0142 | 0x62 |
| (etx) | 3   | 0003 | 0x03 | #    | 35  | 0043 | 0x23 | C    | 67  | 0103 | 0x43 | c    | 99  | 0143 | 0x63 |
| (eot) | 4   | 0004 | 0x04 | \$   | 36  | 0044 | 0x24 | D    | 68  | 0104 | 0x44 | d    | 100 | 0144 | 0x64 |
| (enq) | 5   | 0005 | 0x05 | %    | 37  | 0045 | 0x25 | E    | 69  | 0105 | 0x45 | e    | 101 | 0145 | 0x65 |
| (ack) | 6   | 0006 | 0x06 | &    | 38  | 0046 | 0x26 | F    | 70  | 0106 | 0x46 | f    | 102 | 0146 | 0x66 |
| (bel) | 7   | 0007 | 0x07 | '    | 39  | 0047 | 0x27 | G    | 71  | 0107 | 0x47 | g    | 103 | 0147 | 0x67 |

- Un booleano se codifica mediante un bit.
- El valor 1 se corresponde con `true`
- El valor 0 se corresponde con `false`

**NOTA:** Es mala práctica de programación usar 1 o 0 en un programa en lugar de `true` o `false`.

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++**
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

El nombre (identificador) por el que nos referimos a una variable, constante y otros elementos en C++ cumple:

- El primer carácter es una letra o el carácter '\_' (subrayado o guión bajo)
- Los siguientes pueden ser letras, el carácter '\_' y dígitos, en cualquier disposición.
- Las palabras que forman parte de la definición del lenguaje están reservadas y no pueden usarse como identificadores. Por ejemplo, una variable no puede llamarse `int`.
- **NOTA:**
  - La 'ñ' o 'Ñ' no pueden usarse como parte de un identificador.
  - No se pueden usar vocales acentuadas o con diéresis.
  - El espacio en blanco o los signos de puntuación no pueden formar parte de un identificador.

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++**
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

Antes de que un objeto pueda ser usado en C++ mediante un identificador, este tiene que ser declarado:

- Cada tipo se codifica de distinta forma.
- Cada tipo ocupa distinta cantidad de memoria.

La mayor parte de las declaraciones son también definiciones. Es posible declarar primero un objeto y definirlo después.

Una declaración consta de cuatro partes:

- Especificador (opcional para variables, obligatorio para constantes).
- Nombre de un tipo.
- Nombre (identificador).
- Valor inicial precedido de = (opcional para variables, obligatorio para constantes).

**NOTA:** opcional quiere decir que no es necesario para que la sintaxis de la declaración sea correcta, pero puede ser necesario para el correcto funcionamiento del programa.

- Las variables se definen de la forma siguiente:

```
especificador tipo nombre=valor inicial;
```

- La sintaxis mínima, eliminando los elementos opcionales es:

```
tipo nombre;
```

Donde `nombre` es el nombre de la variable y `tipo` el nombre de su tipo. Ejemplo:

- `int base;`

- `float coef;`

- Las constantes se definen como las variables, precedidas del especificador `const` y seguidas obligatoriamente de su valor. `const tipo nombre=valor;`

- Por ejemplo,

```
const float PI=3.1416;
```

- Pueden declararse varias variables del *mismo* tipo separando nombres y valores iniciales por comas:

- Por ejemplo: `float delta=.75,total;`



- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones**
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos
  - Entrada o lectura
  - Salida o escritura

Una expresión se define inductivamente como:

- Una constante.
- Una variable.
- Una función de una expresión.
- Dos expresiones relacionadas con un operador.

NOTA: Por el momento, una función es una entidad que posee un nombre y que equivale al valor obtenido al realizar determinado cálculo con una serie de valores, encerrados entre paréntesis, a continuación del nombre de la función y separados entre si por comas.

Por ejemplo, si  $x$  es el nombre de una variable y  $e$  es el nombre de una constante, las siguientes son expresiones:

- $e$
- $10.97$
- $x$
- $x/10.97$
- $\exp(x/10.97)$
- $e*\exp(x/10.97)$

- **Evaluar** una expresión consiste en reemplazar los nombres de las variables y de las constantes por su valor y realizar el cálculo indicado.
- Un computador efectúa esa tarea con las expresiones presentes en un programa.
- Todas las variables y funciones que intervengan en la expresión tienen que tener valor definido.
- El resultado de evaluar una función pertenece a un tipo, así existen expresiones de tipo entero, real, booleano, carácter,...

- **Conversión (cast) implícita de tipos:** El tipo de la expresión es el del tipo de mayor rango que exista en esa expresión. Por ejemplo si interviene un entero y un real, el entero se convierte a real (se añaden decimales nulos) y la expresión es de tipo real:  $3/2$  vale 1, pero  $3/2.0$  vale 1.5
- **Conversión (cast) explícita de tipos:** Se puede forzar la conversión de un valor de un tipo a otro de estas dos formas:
  - `(tipo) valor` heredada del antiguo C.
  - `tipo (valor)` incorporada a C++.

En el caso del ejemplo anterior `3/(float)2` o `3/float(2)`

**NOTA:** esta conversión también se puede realizar con expresiones.

## Precedencia y asociatividad:

- Se realizan primero las operaciones indicadas por los operadores de mayor prioridad.
- Si tienen la misma prioridad, el orden de ejecución lo establece la asociatividad.

Prioridades mayores en la parte superior de la tabla:

| Operadores         | Asociatividad       |
|--------------------|---------------------|
| ( ) [ ]            | izquierda a derecha |
| - (unario) ! ++ -- | derecha a izquierda |
| * / %              | izquierda a derecha |
| + -                | izquierda a derecha |
| < <= > >=          | izquierda a derecha |
| == !=              | izquierda a derecha |
| &&                 | izquierda a derecha |
|                    | izquierda a derecha |
| = += -= *= /= %=   | derecha a izquierda |

- Operandos de tipo entero, carácter o convertidos a `int` mediante `cast`.
- Operadores involucrados:
  - Multiplicativos `*` `/` `%`
  - Aditivos `+` `-`
- La división de dos enteros produce un resultado entero (sin decimales).
- El operador `%` (resto o módulo) devuelve el resto de la división de dos enteros.
- Existen funciones que evalúan a entero, como por ejemplo `abs ( )`, que devuelve el valor absoluto de un número entero.

- Los operadores empleados en las expresiones aritméticas son todos los vistos y además los paréntesis '( )'.
- Las operaciones indicadas por los operadores se realizan en el orden determinado por su precedencia y asociatividad.
- Los cálculos indicados por operadores de mayor precedencia se realizan antes que los indicados por operadores de menor precedencia.
- El operador  $-$  unario es el de mayor precedencia.
- La precedencia de  $*$  /  $\%$  es mayor que la de  $+$   $-$ .
- Cualquier expresión encerrada entre paréntesis se evalúa y es reemplazada por su valor antes de continuar la evaluación del resto de la expresión.



- Al menos un operando real:  $1 / 2$  evalúa a cero pero  $1 / 2 . 0$  evalúa a  $0 . 5$
- Operandos de tipo real o convertidos a real mediante cast.
- Operadores involucrados:
  - Multiplicativos  $*$   $/$
  - Aditivos  $+$   $-$
- Se mantiene la precedencia y la asociatividad.
- Los  $( )$  tienen el mismo efecto.

## Algunas funciones matemáticas que evalúan a real:

| Notación matemática | Notación C++           | Descripción         |
|---------------------|------------------------|---------------------|
| $\sqrt{y}$          | <code>sqrt(y)</code>   | Raíz cuadrada       |
| $ y $               | <code>abs(y)</code>    | Valor absoluto      |
| $e^y$               | <code>exp(y)</code>    | Exponencial         |
| $x^y$               | <code>pow(x, y)</code> | Potenciación        |
| $\ln(y)$            | <code>log(y)</code>    | Logaritmo Neperiano |
| $\log(y)$           | <code>log10(y)</code>  | Logaritmo decimal   |
| $\sin(y)$           | <code>sin(y)</code>    | Seno                |
| $\cos(y)$           | <code>cos(y)</code>    | Coseno              |
| $\tan(y)$           | <code>tan(y)</code>    | Tangente            |
| $\text{asen}(y)$    | <code>asin(y)</code>   | Arcoseno            |
| $\text{acos}(y)$    | <code>acos(y)</code>   | Arcocoseno          |
| $\text{atan}(y)$    | <code>atan(y)</code>   | Arcotangente        |

**NOTA:** para usar estas funciones es necesario escribir el principio del programa `#include<cmath>`

- Operadores:  $\&\&$  (conjunción),  $\|\|$  (disyunción),  $!$  (negación).
- En otras disciplinas se usan otros símbolos:

| Operación lógica                               | operador C++ |
|--|--------------|
| $\wedge, Y$ , conjunción, producto lógico, AND | $\&\&$       |
| $\vee, O$ , disyunción, suma lógica, OR        | $\ \ $       |
| $\neg, NO$ , negación, NOT                     | $!$          |

- Los operadores  $\|\|$  y  $\&\&$  son binarios.
- El operador  $!$  es unario.

- Los operadores relacionales se aplican a tipos no booleanos pero evalúan a booleano.
- Evalúan a `true` si lo que se expresa en la condición es cierto y `false` en caso contrario.
- Operadores relacionales: `==`, `!=`, `<`, `<=`, `>`, `>=`.
- La precedencia se ha mostrado anteriormente.
- Se pueden usar `( )` para alterar la precedencia.
- Se pueden combinar operadores booleanos, relacionales, aritméticos y llamadas a funciones:

```
x > 2 * a + sqrt(c) | | a != c
```

**NOTA:** es mala práctica de programación utilizar el resultado de una expresión booleana como si su valor fuese 1 o 0, es decir, de tipo entero.

- Probablemente las únicas expresiones de tipo carácter que veamos serán constantes.
- Una constante de tipo carácter se encierra entre comillas simples:
  - Por ejemplo 'x' es la constante de tipo carácter cuyo valor es precisamente el carácter x.
- El mismo carácter si no se encierra entre comillas simples, se entiende que es un identificador.
  - Por ejemplo `int x=11;` declararían una variable de tipo entero y le asignaría el valor 11.

- La asignación consiste en dar valor a una variable.
- Sintaxis: `variable=expresion;`
  - `variable` es el nombre de una variable
  - `expresion` es una expresión del mismo tipo que `variable` o de un tipo que se pueda convertir a él.
  - El signo = que figura entre la expresión y la variable es el operador de asignación.
- La acción se desarrolla en dos fases:
  - 1 Se evalúa el valor de `expresion`
  - 2 Se almacena en las posiciones de memoria asociadas a `variable` el resultado de la evaluación. Si los tipos difieren, previamente se convierte el tipo de la expresión al de la variable

- En una asignación como la siguiente:

`variable=variable+1;`

Si `variable` es `int` y su valor inicial es 2, después de efectuarse la evaluación tendrá el valor 3.

- Notación expandida/comprimida:

| Notación expandida  | Notación comprimida |
|---------------------|---------------------|
| <code>x=x+1;</code> | <code>x++;</code>   |
| <code>x=x-1;</code> | <code>x--;</code>   |
| <code>x=x+a;</code> | <code>x+=a;</code>  |
| <code>x=x-a;</code> | <code>x-=a;</code>  |
| <code>x=x*a;</code> | <code>x*=a;</code>  |
| <code>x=x/a;</code> | <code>x/=a;</code>  |

- 1 Concepto de variable y constante
- 2 Tipos elementales
  - Tipo entero
  - Tipo real
  - Tipo booleano
  - Tipo carácter
- 3 Codificación de los valores de los tipos simples
  - Codificación de un entero
  - Codificación de un real
  - Codificación de un carácter
  - Codificación de un booleano
- 4 Reglas para la construcción de un nombre en C++
- 5 Declaraciones y definiciones en C++
  - Definición de variables y de constantes
- 6 Expresiones
  - Expresiones de tipo entero
  - Expresiones de tipo real
  - Expresiones de tipo booleano
  - Expresiones de tipo carácter
  - Acción de asignación
- 7 Acciones de entrada y salida de datos**
  - Entrada o lectura
  - Salida o escritura



- Acción en que un valor se introduce en el entorno del procesador por medio de un periférico diseñado a tal efecto ( por ejemplo, el teclado).
- Una lectura es una asignación, se toma el valor y se almacena en el espacio de memoria asignado a una variable. Si una variable se llama  $x$ , la lectura se expresa:  
`cin>>x;`
- Si  $x$  es int el usuario/a teclearía el valor de la variable (por ejemplo 1436) y al finalizar pulsaría la tecla return, momento en el que se asignaría 1436 a  $x$ .

- No es necesario que, en un programa en C++, las acciones de lectura correspondientes a cada dato figuren en una línea separada
- Se pueden escribir consecutivamente con el teclado varios valores separados por espacios en blanco, tabuladores o retornos de carro y pulsando la tecla return después de introducir el último valor.
- `cin>>x>>y>>z;` pide por el teclado tres valores y los almacena en las variables `x`, `y`, `z`.
- Los nombres de las variables no se mostrarían por la pantalla.
- Es adecuado que el programa informe al usuario sobre los datos a introducir.

- Comunica al exterior un valor, por ejemplo mediante el monitor.
- La escritura del valor de una variable  $x$  se expresa del siguiente modo:  
`cout<<x;`
- Para escribir constantes de tipo carácter se entrecomillan usando comillas simples:  
`cout<<'A' ;` (mostraría por la pantalla 'A' sin las comillas)
- Para mostrar cadenas de caracteres se escriben entre comillas dobles:  
`cout<<"Hola" ;` (mostraría por la pantalla la cadena "Hola" sin las comillas)

- Es posible escribir varias acciones de salida en una misma línea:

```
cout<<x<<' '<<y<<' '<<z;
```

Los valores aparecen separados por espacios (' ', es una cte. de tipo carácter que representa un espacio)

- Si se desea pasar a la siguiente línea en el monitor se utiliza endl.

```
cout<<"El resultado es:"<<endl<<x;
```

La orden anterior mostraría la cadena "El resultado es:", de nuevo sin comillas, y en la línea siguiente el valor de la variable x.

- Si se aplica a expresiones, se evalúan antes.
- Para usar cin y cout hay que escribir al principio del programa:

```
#include<iostream>  
using namespace std;
```