CAPÍTULO 3

Operadores y expresiones

Introducción

Este capítulo muestra cómo C++ hace uso de los operadores y expresiones para la resolución de operaciones. Los operadores fundamentales que se analizan en el capítulo son: *aritméticos*, *lógicos* y *relacionales*; de *manipulación de bits*; *condicionales* y *especiales*. Además, se analizan las conversiones de tipos de datos y las reglas que sigue el compilador cuando concurran en una misma expresión diferentes tipos de operadores. Estas reglas se conocen como reglas de *prioridad* y *asociatividad*.

3.1. Operadores y expresiones

Los programas C++ constan de datos, sentencias de programas y expresiones. Una *expresión* es, una sucesión de operadores y operandos debidamente relacionados que especifican un cálculo. C++ soporta un conjunto potente de operadores unitarios, binarios y de otros tipos.

3.2. Operador de asignación

El operador de asignación es un operador cuya sintaxis es la siguiente:

```
variable = expresión;
```

donde `variable` es un identificador válido de C++ declarado como `variable`. El operador = asigna el valor de la expresión derecha a la variable situada a su izquierda. Este operador es asociativo por la derecha, eso permite realizar asignaciones múltiples. Así, `A = B = C = D = 12`; equivale a `A = (B =(C = (D = 12)))` que asigna a las variables `A`, `B`, `C` y `D` el valor `12`. Esta propiedad permite inicializar varias variables con una sola sentencia. Además del operador de asignación = C++ proporciona cinco operadores de asignación adicionales dados en la Tabla 3.1.

EJEMPLO 3.1. *Programa que declara variables y hace uso de operadores de asignación.*

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

```
int main(int argc, char *argv[])
{
  int codigo, CoordX, CoordY;
  float fahrenheit, valor;

  codigo = 3467;
  valor = 10;
  valor *= 3;
  CoordX = 525;
  CoordY = 725;
  CoordY -= 10;
  fahrenheit = 123.456;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```


Tabla 3.1. Operadores y equivalencias de asignación de C++

Símbolo	Uso	Descripción	Sentencia abreviada	Sentencia no abreviada
=	a = b	Asigna el valor de *b* a *a*.	m = n	m = n
*=	a *= b	Multiplica *a* por *b* y asigna el resultado a la variable *a*.	m *= n;	m = m * n;
/=	a /= b	Divide *a* entre *b* y asigna el resultado a la variable *a*.	m /= n;	m = m / n;
%=	a %= b	Fija *a* al resto de *a/b*.	m %= n;	m = m % n;
+=	a += b	Suma *b* y *a* y lo asigna a la variable *a*.	m += n;	m = m + n;
-=	a -= b	Resta *b* de *a* y asigna el resultado a la variable *a*.	m -= n;	m = m - n;

3.3. Operadores aritméticos

Los operadores aritméticos de C++ sirven para realizar operaciones aritméticas básicas. Siguen las reglas algebraicas típicas, de jerarquía o prioridad, clásicas de matemáticas. Estos operadores vienen recogidos en la Tabla 3.2.

Tabla 3.2. Operadores aritméticos

Operador	Tipos enteros	Tipos reales	Ejemplo
+	Suma	Suma	x + y
-	Resta	Resta	b - c
*	Producto	Producto	x * y
/	División entera: cociente	División en coma flotante	b / 5
%	División entera: resto		b % 5

Los paréntesis se pueden utilizar para cambiar el orden usual de evaluación de una expresión determinada por su *prioridad* y *asociatividad*. La prioridad de los operadores y su asociatividad se recogen en la Tabla 3.3.

Tabla 3.3. Prioridad y asociatividad

Prioridad (mayor a menor)	Asociatividad
+, - *(unitarios)*	izquierda-derecha (→)
*, /, %	izquierda-derecha (→)
+, -	izquierda-derecha (→)

EJEMPLO 3.2. *Evaluación de expresiones.*

a) ¿Cuál es el resultado de evaluación de la expresión: `50 - 4 * 3 + 2`?

```
50 - 4 * 3 + 2
50 - 12 + 2
38 + 2
32
```

b) ¿Cuál es el resultado de evaluación de la expresión: `5 * ( 10 - 2 * 4 + 2 ) - 2`?

```
5 * ( 10 - 2 * 4 + 2 ) - 2
5 * ( 10 - 8 + 2 ) - 2
5 * ( 2 + 2 ) - 2
5 *  4  - 2
20 - 2
18
```


c) ¿Cuál es el resultado de la expresión: `7 * 5 - 6 % 4 * 4 + 9`?

```
7 * 5 - 6 % 4 * 4 + 9
35 - 6 % 4 * 4 + 9
35 - 2 * 4 + 9
35 - 8 + 9
27 + 9
36
```

d) ¿Cuál es el resultado de la expresión: `15 * 14 - 3 * 7`?

```
15 * 14 - 3 * 7
210 - 3 * 7
210 - 21
189
```

e) ¿Cuál es el resultado de la expresión: `3 + 4 * (8 * (4 - (9 + 3) / 6)))`?


```
3 + 4 * (8 * (4 - (9 + 2) / 6)))
3 + 4 *(8 * (4 - 11 / 6))
3 + 4 *(8 * (4 - 1))
3 + 4 *(8 * 3)
3 + 4 * 24
3 + 96
99
```

EJEMPLO 3.3. *Programa que lee el radio, calcula y visualiza la longitud de la circunferencia de ese radio, y el área del círculo del mismo radio.*

```
#include <cstdlib>
#include <iostream>
#define pi 3.141592
using namespace std;

int main(int argc, char *argv[])
{
    float radio, longitud, area;
```

```
    cin >> radio;                          //lectura del radio
    longitud = 2 * pi * radio;
    area = pi * radio * radio;
    cout << " radio = " << radio <<  endl;
    cout << " longitud = " << longitud <<  endl;
    cout << " area = " << area <<  endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

EJEMPLO 3.4. *Desglosar cierta cantidad de segundos introducida por teclado en su equivalente en semanas, días, horas, minutos y segundos.*

El programa lee el número de segundos y realiza las conversiones, teniendo en cuenta que una semana tiene 7 días, un día tiene 24 horas, una hora 60 minutos, y un minuto 60 segundos.


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

int main(int argc, char *argv[])
{
  int semanas, dias, horas, minutos, segundos, acu;

  cout << "Introduzca segundos ";
  cin >> acu;
  segundos = acu % 60;
  acu = acu / 60;
  minutos = acu % 60;
  acu = acu / 60;
  horas = acu % 24;
  acu = acu / 24;
  dias = acu % 7;
  semanas = acu / 7;
  cout << "segundos en semanas dias horas minutos y segundos " << endl;
  cout << " numero de semanas " <<  semanas << endl;
  cout << " numero de dias " << dias << endl;
  cout << " numero de horas " <<  horas << endl;
  cout << " numero de minutos " <<  minutos << endl;
  cout << " numero de segundos " <<  segundos << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```


En resultado de ejecución es el siguiente:

```
Introduzca segundos 4567456
 segundos en semanas dias horas minutos y segundos
 numero de semanas 7
 numero de dias 3
 numero de horas 20
 numero de minutos 44
 numero de segundos 16
Presione una tecla para continuar . . .
```

3.4. Operadores de incrementación y decrementación

De las características que incorpora C++ una de las más útiles son los operadores de *incremento* ++ y *decremento* --. Estos operadores unitarios suman o restan 1 respectivamente a la variable y tienen la propiedad de que pueden utilizarse como sufijo o prefijo. El resultado de la e xpresión puede ser distinto, dependiendo del contexto. En la Tabla 3.4 se recogen los operadores de incrementación y decrementación.

Tabla 3.4. Operadores de incrementación (++) y decrementación (--)

Incrementación	Decrementación
++n, n++	--n , n--
n += 1	n -= 1
n = n + 1	n = n - 1

Si los operadores ++ y -- están de prefijos, la operación de incremento se efectúa antes que la operación de asignación; si los operadores ++ y -- están de sufijos, la asignación se efectúa en primer lugar y la incrementación o decrementación a continuación.

EJEMPLO 3.5. *Diferencias entre operadores de preincrementación y postincrementación.*

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int m = 10, n;

  n = ++m;              // primero se incrementa m y luego se asigna a n
  cout << " m = " << m << " n = " << n <<endl ;
  n = m++;              // primero se asigna a n y luego se incrementa m
  cout << " m = " << m << " n = " << n <<endl ;
  cout << " m = " << m++ << endl;
  cout << " m = " << ++m << endl;
  n = 5;
  m = ++n * --n;        //++n pone n a 6, luego --n pone n a 5, luego m =25
  cout << " n = " << n << " m = " << m << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Resultado de ejecución:

```
m = 11 n = 11
m = 12 n = 11
m = 12
m = 14
n = 5 m = 25
```

3.5. Operadores relacionales

C++ soporta el tipo `bool` que tiene dos literales `false` y `true`. Una expresión *booleana* es, por consiguiente, una secuencia de operandos y operadores que se combinan para producir uno de los v alores `true` y `false`. C++ utiliza el tipo `int` para representar los valores verdadero *(true)* y falso *(false)*. El valor entero 0 representa a falso y cualquier valor distinto de cero a ver-

dadero. Operadores tales como >= y == que comprueban una relación entre dos operandos se llaman *operadores relacionales* y se utilizan en expresiones de la forma:

```
expresión1 operador_relacional expresión2
```

expresión$_1$ *y expresión*$_2$	expresiones compatibles C++
`operador_relacional`	un operador de la Tabla 3.5

La Tabla 3.5 muestra los operadores relacionales que se pueden aplicar a operandos de cualquier tipo de dato estándar: `char`, `int`, `float`, `double`, etc.

Tabla 3.5. Operadores relacionales de C

Operador	Significado	forma	Ejemplo	
==	*Igual a*	a == b	'A'== C'	falso
!=	*No igual a*	a != b	2 != 4	verdadero
>	*Mayor que*	a > b	5 > 6	falso
<	*Menor que*	a < b	'a'<'c'	verdadero
>=	*Mayor o igual que*	a >= b	'B'>='b'	falso
<=	*Menor o igual que*	a <= b	2 <= 1	falso

3.6. Operadores lógicos

Los *operadores lógicos* se utilizan con expresiones para devolver un valor *verdadero* (cualquier entero distinto de cero) o un valor *falso* (0). Los operadores lógicos de C son: **not** (!), **and** (&&) y **or**(||). El operador unitario lógico `! (not, no)` produce *falso* (cero) si su operando es *verdaderos* (distinto de cero) y vice versa. El operador binario lógico `&& (and, y)` produce *verdadero sólo* si ambos operandos son *verdaderos* (no cero); si cualquiera de los operandos es *falso* produce *falso*. El operador binario lógico `|| (or, o)` produce *verdadero* si cualquiera de los operandos es *verdadero* (distinto de cero) y produce *falso* sólo si ambos operandos son *falsos*.

La precedencia de los operadores es: los operadores matemáticos tienen precedencia sobre los operadores relacionales, y los operadores relacionales tienen precedencia sobre los operadores lógicos.

> El operador ! tiene prioridad más alta que &&, que a su vez tiene mayor prioridad que ||. La asociatividad es de izquierda a derecha.

Evaluación en cortocircuito. En C++ los operandos de la izquierda de && y || se evalúan siempre en primer lugar; si el valor del operando de la izquierda determina de forma inequívoca el valor de la expresión, el operando derecho no se evalúa. Esto significa que si el operando de la izquierda de && es falso o el de || es verdadero, el operando de la derecha no se evalúa. Esta propiedad se denomina *evaluación en cortocircuito*.

Las sentencias de asignación se pueden escribir de modo que se puede dar un valor de tipo `bool` o una variable `bool`.

EJEMPLO 3.6. *Sentencias de asignación a los tipos de variables bool.*

```
int n;
float x ;
char car;
bool r, esletra, espositiva;

r =  (n > -100) && (n < 100);       // true si n están entre -110 y 100
esletra = (( 'A'<= car) && (car <= 'Z'))
          (('a'<= car) && (car <= 'Z'));
          //si car es una letra la variable es letra true, y toma false en otro caso
espositivo = x >= 0;
```

3.7. Operadores de manipulación de bits

Los operadores de manipulación o tratamiento de bits (*bitwise*) ejecutan operaciones lógicas sobre cada uno de los bits de los operandos. Estas operaciones son comparables en eficiencia y en velocidad a sus equivalentes en lenguaje ensamblador. Cada operador de manipulación de bits realiza una operación lógica bit a bit sobre datos internos. Los operadores de manipulación de bits se aplican sólo a variables y constantes `char`, `int` y `long`, y no a datos en coma flotante. La Tabla 3.6 recoge los operadores lógicos bit a bit.

Tabla 3.6. Operadores lógicos bit a bit

Operador	Operación
&	`y (and)` *lógica bit a bit.*
|	`o (or)` *lógica (inclusiva) bit a bit.*
^	`o (xor)` *lógica (exclusiva) bit a bit (or e xclusive, xor).*
~	*Complemento a uno (inversión de todos los bits).*
<<	*Desplazamiento de bits a izquierda.*
>>	*Desplazamiento de bits a derecha.*

3.7.1. OPERADORES DE DESPLAZAMIENTO DE BITS (>>, <<)

Efectúa un desplazamiento a la derecha (`>>`) o a la izquierda (`<<`) de `numero_de_bits` posiciones de los bits del operando, siendo `numero_de_bits` un número entero. Los formatos de los operadores de desplazamiento son:

1. `valor << numero_de_bits;` **2.** `valor >> numero_de_bits;`

EJEMPLO 3.7. *Sentencias de manipulación de bits.*

variable	V. entero	V binario	
`X`	9	00001001	1001
`Y`	10	00001010	1010
`x & y`	8	00001000	1001 & 1010 `and` bit a bit
`x | y`	11	00001011	1001 | 1010 `or` bit a bit
`x ^ y`	3	00000011	1001 ^ 1010 `xor` bit a bit
`x << 2`	36	01000100	1000**00** aparecen dos ceros a la derecha
`y >> 2`	2	00000010	10

El siguiente programa comprueba los valores indicados en la tabla anterior.

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

int main(int argc, char *argv[])
{ int x = 9, y = 10, x_y_y, x_or_y, x_Xor_y;

  x_y_y = x & y;
  x_or_y = x | y;
  x_Xor_y = x ^ y;
  cout <<  "x = " <<  x << " \n";
  cout <<  "y = " <<  y << " \n";
  cout <<  "x_y_y = " <<  x_y_y << " \n";
```

```
    cout <<  "x_or_y = " <<  x_or_y << " \n";
    cout <<  "x_Xor_y = " <<  x_Xor_y << " \n";
    x = x << 2;
    y = y >> 2;
    cout <<  "x << 2 = " <<  x << " \n";
    cout <<  "y >> 2  = " <<  y << " \n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

El resultado de ejecución del programa anterior es:

```
x = 9
y = 10
x_y_y = 8
x_or_y = 11
x_Xor_y = 3
x << 2 = 36
y >> 2  = 2
```


3.7.2. OPERADORES DE ASIGNACIÓN ADICIONALES

Los operadores de asignación abreviados están disponibles también para operadores de manipulación de bits. Estos operadores se muestran en la Tabla 3.7.

Tabla 3.7. Operadores de asignación adicionales

Símbolo	Uso	Descripción
`<<=`	`a <<= b`	Desplaza a a la izquierda `b` bits y asigna el resultado a `a`.
`>>=`	`a >>= b`	Desplaza a a la derecha `b` bits y asigna el resultado a `a`.
`&=`	`a &= b`	Asigna a `a` el valor `a & b`.
`^=`	`a ^= b`	Establece `a` a *a* `^` *b*.
`||=`	`a ||= b`	Establece `a` a *a* `|` *b*.

3.7.3. OPERADORES DE DIRECCIONES

Son operadores que permiten manipular las direcciones de los objetos. Se recogen en la Tabla 3.8.

Tabla 3.8. Operadores de direcciones

Operador	Acción
*	Lee o modifica el valor apuntado por la expresión. Se corresponde con un puntero y el resultado es del tipo apuntado.
&	Devuelve un puntero al objeto utilizado como operando, que debe ser un *lvalue* (variable dotada de una dirección de memoria). El resultado es un puntero de tipo idéntico al del operando.
.	Permite acceder a un miembro de un objeto agre gado (*unión, estructura*).
->	Accede a un miembro de un objeto agre gado (*unión, estructura*) apuntado por el operando de la izquierda.

3.8. Operador condicional ?

El **operador condicional** ?, es un operador ternario que de vuelve un resultado cuyo v alor depende de la condición comprobada. El formato del operador condicional es:

```
expresion_L ? expresion_v : expresion_f;
```

Se evalúa la expresión lógica `expresion_L` y su valor determina cuál es la expresión a ejecutar; si la condición es *verdadera* se ejecuta `expresion_v` y si es falsa se ejecuta `expresion_f`. La precedencia de `?` es menor que la de cualquier otro operando tratado hasta ese momento. Su asociati vidad es a derecha.

EJEMPLO 3.8. *Uso del operador condicional ?.*

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int x = 10, y =12, mayor, menor;
  bool z;

  z =  x >= y ? true: false ;        // z toma el valor de false.
  mayor = x >= y ? x : y;            // calcula y almacena el mayor
  menor = x >= y ? y : x;            // calcula y almacena el mayor
  cout <<  "x = " <<  x << " \n";
  cout <<  "y  = " <<  y << " \n";
  cout <<  "el mayor es = " <<  mayor << " \n";
  cout <<  "el menor es = " <<  menor << " \n";
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Resultados de ejecución:

```
x = 10
y  = 12
el mayor es = 12
el menor es = 10
```


3.9. Operador coma ,

El *operador coma* permite combinar dos o más e xpresiones separadas por comas en una sola línea. Se e valúa primero la expresión de la izquierda y lue go las restantes e xpresiones de izquierda a derecha. La e xpresión más a la derecha determina el resultado global de la expresión. El uso del operador coma es:

```
expresión1, expresión2, expresión3, ..., expresión
```

Cada expresión se evalúa comenzando desde la izquierda y continuando hacia la derecha.

EJEMPLO 3.9. *Uso del operador condicional ,.*

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
    int i, j, r, k;
```

```
    r = j = 10, i = j,  k = (i++, i+1) ;
    cout <<" i= "<<  << " j= " << j << " r= " << r <<" k= " << k <<"\n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

El resultado de ejecución es:

```
i= 11 j= 10 r= 10 k= 12
```

`j` toma el valor de `10`, `r` toma el valor de `j` que es `10`, `i` toma el valor de `10`. Posteriormente `i` toma el valor `11`; se ejecuta la expresión `i+1` que es `12` y se asigna a `k`.

3.10. Operadores especiales `( )`, `[ ]` y `::`

C++ admite algunos operadores especiales que sirven para propósitos diferentes. Cabe destacar: `()`, `[]` y `::`.

El operador `()` es el operador de llamada a funciones. Sirve para encerrar los argumentos de una función, efectuar conversiones explícitas de tipo, indicar en el seno de una declaración que un identificador corresponde a una función y resolver los conflictos de prioridad entre operadores.

El operador `[]` sirve para designar un elemento de un array. También se puede utilizar en unión con el operador `delete`; en este caso, indica el tamaño del array a destruir y su sintaxis es: *`delete [tamaño_array] puntero_array;`*

El operador `::` es específico de C++ y se denomina *operador de ámbito de resolución*, y permite especificar el alcance o ámbito de un objeto. Su sintaxis es:

`class::miembro` o bien *`::miembro`*

3.11. El operador `sizeof`

C++ proporciona el operador `sizeof`, que toma un argumento, bien un tipo de dato o bien el nombre de una variable (escalar, array, registro, etc.), y obtiene como resultado el número de bytes que ocupa. El formato del operador es:

```
sizeof(nombre_variable)
sizeof(tipo_dato)
sizeof expresión
```


EJEMPLO 3.10. *El siguiente programa escribe el tamaño de los tipos de datos en su ordenador.*

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

int main(int argc, char *argv[])
{
    cout << " el tamaño de variables de esta computadora son:\n";
    cout << " entero: " << sizeof(int) << '\n';
    cout << " entero largo: " << sizeof(long int) << '\n';
    cout << " rael: " << sizeof(float) << '\n';
    cout << " doble: " << sizeof(double) << '\n';
    cout << " long doble: " << sizeof(long double) << '\n';
    cout << " long doble: " << sizeof 20 << '\n';
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

3.12. Conversiones de tipos

Las conversiones de tipos pueden ser *implícitas* (ejecutadas automáticamente) o *explícitas* (solicitadas específicamente por el programador). C++ hace muchas con versiones de tipos automáticamente: convierte valores cuando se asigna un v alor de un tipo a una variable de otro tipo; convierte valores cuando se combinan tipos mixtos en e xpresiones; convierte valores cuando se pasan argumentos a funciones.

Conversión implícita. Los tipos fundamentales (básicos) pueden ser mezclados libremente en asignaciones y expresiones. Las conversiones se ejecutan automáticamente: los operandos de tipo más bajo se con vierten a los de tipo más alto de acuerdo con las siguientes reglas: si cualquier operando es de tipo `char`, `short` o enumerado se convierte en tipo `int`; si los operandos tienen diferentes tipos, la siguientes lista determina a qué operación con vertirá. Esta operación se llama promoción inte gral.

```
int,  unsigned int,  long, unsigned long, float, double
```

El tipo que viene primero, en esta lista, se convierte en el que viene se gundo, etc.

Conversiones explícitas. C++ fuerza la conversión explícita de tipos mediante el operador de *molde (cast)*. El operador *molde* tiene el formato *`(tiponombre)valor`*. Convierte *`valor a tiponombre`*.

Por ejemplo dada la declaración: `float x;`

```
x = 3/2              asigna a x el valor de 1,
x = (float)3/2       asigna a x el valor de 1.5
```

EJEMPLO 3.10. *El siguiente programa muestra conversiones implícitas y explícitas de enteros y caracteres.*

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

int main(int argc, char *argv[])
{
    char c = 'Z' +1 ;      // asigna a c el siguiente carácter de 'Z'

    cout <<'A'  <<" " << (int)'A' <<  endl;   //carácter y número ASCII
    cout <<'0'  <<" " << (int)'0' <<  endl;   //carácter y número ASCII
    cout <<'a'  <<" " << (int)'a' <<  endl;   //carácter y número ASCII
    cout << c  << " " << (int) c <<  endl;    //carácter y número ASCII
    cout <<'Z'+1<<" "<<(char)('Z'+1 ) << endl; //número ASCII y carácter
    system("PAUSE");
    return EXIT_SUCCESS;
}
```


La ejecución del programa anterior es:

```
A 65
0 48
a 97
[ 91
91 [
```

3.13. Prioridad y asociatividad

La **prioridad** o **precedencia** de operadores determina el orden en el que se aplican los operadores a un valor. A la hora de evaluar una expresión hay que tener en cuenta las siguientes re glas y la Tabla 3.9.

- Los operadores del grupo 1 tienen mayor prioridad que los del grupo 2, y así sucesivamente.

- Si dos operadores se aplican al mismo operando, el operador con mayor prioridad se aplica primero.
- Todos los operadores del mismo grupo tienen igual prioridad y asociati vidad.
- La asociatividad *izquierda-derecha* significa aplicar el operador más a la izquierda primero, y en la asociati vidad *derecha-izquierda* se aplica primero el operador más a la derecha.
- Los paréntesis tienen la máxima prioridad.

Tabla 3.9. Prioridad y asociatividad de los operadores

Prioridad	Operadores	Asociatividad
1	:: x -> [] ()	I – D
2	++ -- ~ ! - + & * sizeof	D – I
3	.* ->*	I – D
4	* / %	I – D
5	+ -	I – D
6	<< >>	I – D
7	< <= > >=	I – D
8	== !=	I – D
9	&	I – D
10	^	I – D
11	|	I – D
12	&&	I – D
13	||	I – D
14	?: *(expresión condicional)*	D – I
15	= *= /= %= += -= <<= >>= &= ||= ^=	D – I
16	, (*operador coma*)	I

I - D : *Izquierda – Derecha* D - I : *Derecha – Izquierda.*

EJERCICIOS

3.1. *Determinar el valor de las siguientes expresiones aritméticas:*

15 / 12	15 % 12
24 / 12	24 % 12
123 / 100	200 % 100

3.2. *¿Cuál es el valor de cada una de las siguientes expresiones?*

a) 10 * 14 – 3 * 2
b) –4 + 5 * 2
c) 13 – (24 + 2 * 5) / 4 %3
d) (4 – 40 / 5) %3
e) 4 * (3 + 5) – 8 * 4 % 2 – 5
f) –3 * 10 + 4 * (8 + 4 * 7 – 10 * 3) / 6

3.3. *Escribir las siguientes expresiones aritméticas como expresiones de computadora: La potencia puede hacerse con la función pow(), por ejemplo* $(x + y)^2$ *==pow(x+y,2).*

a) $\frac{x}{y} + \quad 1$
b) $\frac{x+y}{x-y}$
c) $x + \frac{y}{z}$
d) $\frac{b}{c+d}$
e) $(a+b)\frac{c}{d}$
f) $[(a+b)^2]^2$
g) $\frac{xy}{1-4x}$
h) $\frac{xy}{mn}$
i) $(x+y)^2 \,.\, (a-b)$

3.4. *Escribir las sentencias de asignación que permitan intercambiar los contenidos (valores) de dos variables x, e y de un cierto tipo de datos.*

3.5. *Escribir un programa que lea dos enteros en las variables x e y, y, a continuación, obtenga los valores de: a) x / y; b); x % y. Ejecute el programa varias veces con diferentes pares de enteros como entrada.*

3.6. *Una temperatura dada en grados Celsius (centígrados) puede ser convertida a una temperatura equivalente Fahrenheit de acuerdo a la siguiente fórmula:* $f = \frac{9}{5} c + 32$. *Escribir un programa que lea la temperatura en grados centígrados y la convierta a grados Fahrenheit.*

PROBLEMAS

3.1. *La relación entre los lados (a,b) de un triángulo y la hipotenusa (h) viene dada por la fórmula:* $a^2 + b^2 = h^2$. *Escribir un programa que lea la longitud de los lados y calcule la hipotenusa .*

3.2. *Escribir un programa que lea un entero y, a continuación, visualice su doble y su triple.*

3.3. *Escriba un programa que lea los coeficientes a, b, c , d, e, f de un sistema lineal de dos ecuaciones con dos incógnitas y muestre la solución.*

$$\begin{cases} ax + by = c \\ cx + dy = f \end{cases}$$

3.4. *La fuerza de atracción entre dos masas,* m_1 *y* m_2 *separadas por una distancia d, está dada por la fórmula:*

$$F = \frac{G * m1 * m2}{d^2}$$

donde G es la constante de gravitación universal, $G = 6.673.\text{x } 10^{-8}$ $\text{cm}^3/\text{g. seg}^2$
Escriba un programa que lea la masa de dos cuerpos y la distancia entr e ellos y, a continuación, obtenga la fuerza gravitacional entre ella. La salida debe ser en dinas; un dina es igual a gr . cm/seg^2

3.5. *La famosa ecuación de Einstein par a conversión de una masa m en ener gía viene dada por la fórmula:* $E = cm^3$, *donde c es la velocidad de la luz y su valor es:* $c = 2.997925 \times 10^{10}$ m/sg. *Escribir un programa que lea una masa en gramos y obtenga la cantidad de energía producida cuando la masa se convierte en energía. Nota: Si la masa se da en gramos, la fórmula produce le energía en ergios.*

3.6. *Escribir un programa para convertir una medida dada en pies a sus equivalentes en:* a) *yardas; b) pulgadas; donde c) centímetros, y d) metros* (1 pie = 12 pulgadas, 1 yarda = 3 pies, 1 pulgada = 2,54 cm, 1 m = 100 cm). *Leer el número de pies e imprimir el número de yardas, pies, pulgadas, centímetros y metros.*

3.7. *Escribir un programa en el que se introduzca como datos de entrada la longitud del perímetro de un terreno, expresada con tres números enteros que representen hectómetros, decámetros y metros respectivamente, y visualice el perímetro en decímetros.*

3.8. *Escribir un programa que solicite al usuario una cantidad de eur os y transforme la cantidad en eur os en billetes y monedas de curso legal (cambio óptimo).*

SOLUCIÓN DE LOS EJERCICIOS

3.1. *Hay que tener en cuenta que / en el caso de números enteros calcula el cociente de la división entera, y % calcula el resto de la división entera. Por tanto los resultados son:*

```
15 / 12 = 1                    15 % 12 = 3
24 / 12 = 2                    24 % 12 = 0
123 / 100 = 1                  123 % 100 = 23
200 / 100 = 2                  200 % 100 = 0
```

3.2. *La solución por pasos de cada uno de ellos por pasos es:*

a)
```
10 * 14 - 3 * 2
140 - 3 * 2
140 - 6
134
```

b)
```
4 + 5 * 2
4 + 10
14
```

c)
```
13 - (24 + 2 * 5) / 4 %3
13 - (24 + 10) / 4 %3
13 - 34 / 4 %3
13 - 8 %3
13 - 2
11
```

d)
```
(4 - 40 / 5) % 3
(4 - 8) %3
-4 % 3
-1
```

e)
```
4 * (3 + 5) - 8 * 4 % 2 - 5
4 * 8 - 8 * 4 % 2 - 5
32 - 8 * 4 % 2 - 5
32 - 32 % 2 - 5
32 - 0 - 5
32 - 5
27
```

f)
```
-3 * 10 + 4 * (8 + 4 * 7 - 10 * 3) / 6
-30 + 4 * (8 + 4 * 7 - 10 * 3) / 6
-30 + 4 * (8 + 28 - 10 * 3) / 6
-30 + 4 * (8 + 28 - 30) / 6
-30 + 4 * (36 - 30) / 6
-30 + 4 * 6 / 6
-30 + 24 / 6
-30 + 4
-26
```


3.3. *La potencia puede hacerse con la función* `pow( )`*, por ejemplo* $(x + y)^2$ `== pow(x+y,2)`

a) `x / y + 1`	*d*) `b / (c + d)`	*g*) `x * y /(1 - 4 * x)`
b) `(x + y) / (x - y)`	*e*) `(a + b) * (c / d)`	*h*) `x * y / (m * n)`
c) `x + y / z`	*f*) `pow(pow(x + y, 2),2)`	*i*) `pow(x + y,2)*(a - b)`

3.4. *Si* `Aux` *es una variable del mismo tipo que* `x` *e* `y` *las sentencias pedidas son:*

```
Aux = x;
x = y;
y = Aux;
```

3.5. *Una codificación del programa solicitado es:*

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

```
int main(int argc, char *argv[])
{
    int x , y ;

    cin >> x >> y ;
    cout << x / y <<" "<< x % y<<  endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Al ejecutar varias veces el programa se observa que / obtiene el cociente entero, y % obtiene el resto de la división entera.

3.6. *Una codificación del programa solicitado es:*


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

int main(int argc, char *argv[])
{
   float c, f;

   cout << "Introduce grados ";
   cin >> c;
   f = c * 9 / 5 + 32;
   cout << " grados fahrenheit = " << f << endl;
   system("PAUSE");
   return EXIT_SUCCESS;
}
```


SOLUCIÓN DE LOS PROBLEMAS

3.1. *El código solicita los lados, y visualiza los lados y la hipotenusa:*

```
#include <cstdlib>
#include <iostream>
#include <math.h>
using namespace std;

int main(int argc, char *argv[])
{
   float a, b, h;

   cout << "Introduce los lados ";
   cin >> a >> b;
   h = sqrt( a * a + b * b);
   cout << " lado 1 = " << a << endl;
   cout << " lado 2 = " << b << endl;
   cout << " hipotenusa = " << h << endl;
```

```
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Un resultado de ejecución es:

```
Introduce los lados 3 4
 lado 1 = 3
 lado 2 = 4
 hipotenusa = 5
```

3.2. *La codificación pedida es:*

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  int x;

  cout << " dame un numero entero ";
  cin >> x;
  x = 2 * x;
  cout << "su doble es " << x  << " su triple es " << 3 * x << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Un resultado de ejecución es:

```
 dame un numero entero 6
su doble es 12 su triple es 36
Presione una tecla para continuar . . .
```

3.3. *Un sistema lineal de dos ecuaciones con dos incógnitas* $\begin{cases} ax + by = c \\ cx + dy = f \end{cases}$ *tiene solución única si y solamente si* `a * e - b * d` *es distinto de cero, y además la solución viene dada por las expresiones siguientes:*

$$x = \frac{ce - bf}{ae - bd} \qquad y = \frac{af - cd}{ae - bd}$$

El programa codificado solicita al usuario los valores a, b, c, d, e, f, y en caso de que exista solución la muestra.

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

int main(int argc, char *argv[])
{
  float a, b, c, d, e, f, denominador, x, y;

  cout << " Introduzca el valor de a de b y de c ";
  cin >> a >> b >> c;
```

```
cout << " Introduzca el valor de d de e y de f " ;
cin >> d >> e >> f;
denominador = a * e - b * d;
if (denominador == 0)
  cout << " no solucion\n";
else
{
  x = (c * e - b * f) / denominador;
  y = (a * f - c * d) / denominador;
  cout << " la solucion del sistema es\n";
  cout << " x = " << x << " y = " << y << endl;
 }
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Un resultado de ejecución del programa anterior es:

```
 Introduzca el valor de a de b y de c 2 3 5
 Introducca el valor de d de e y de f 1 6 7
 la solucion del sistema es
 x = 1 y = 1
Presione una tecla para continuar . . . _
```

3.4. *Se declara la constante de gravitación universal* `G = 6.673e-8`*, así como las variables* `masa1`*,* `masa2`*,* `distancia`*,* `fuerza` *para posteriormente aplicar la fórmula y presentar el resultado:*

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

int main(int argc, char *argv[])
{
  const float G = 6.673e-8;

  float masa1, masa2, distancia, fuerza;
  cout << " Introduzca la masa de los dos cuerpos en gramos:\n ";
  cin >> masa1 >> masa2;
  cout<<" Introduzca la distancia entre ellos en centimetros:\n ";
  cin >> distancia;
  if (( masa1 <= 0 ) || ( masa2 <= 0) || ( distancia <= 0 ))
    cout << " no solucion\n";
  else
  {
     fuerza = G * masa1 * masa2 / (distancia * distancia);
     cout << " la solucion es: \n";
     cout << " Fuerza en dinas = "<< fuerza << endl;
  }
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Una ejecución es la siguiente:

```
 Introducza la masa de los dos cuerpos en gramos:
 4567 5267
 Introduzca la distancia entre ellos en centimetros:
 3
 la solucion es:
 Fuerza en dinas = 0.17835
Presione una tecla para continuar . . .
```

3.5. *Una codificación es la siguiente:*

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

int main(int argc, char *argv[])
{
  float m, energia;

  const float c = 2.997925e+10;
  cout << " introduzca masa\n ";
  cin >> m;
  energia = c * m * m * m;
  cout << " energia en ergios : " << energia;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

3.6. *El programa lee el número de pies y realiza las transformaciones correspondientes de acuerdo con lo indicado.*

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

int main(int argc, char *argv[])
{
  float pies, pulgadas, yardas, metros, centimetros;

  cout << " Introduzca pies: \n ";
  cin >> pies;
  pulgadas = pies * 12;
  yardas = pies / 3;
  centimetros = pulgadas * 2.54;
  metros = centimetros / 100;
  cout << " pies " << pies << endl;
  cout << " pulgadas " << pulgadas << endl;
  cout << " yardas " << yardas << endl;
  cout << " centimetros " << centimetros << endl;
  cout << " metros " << metros << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

3.7. *El programa que se codifica lee los hectómetros, decámetros y metros y realiza las conversiones correspondientes.*

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

int main(int argc, char *argv[])
{
  int hectometros, decametros, metros, decimetros;

  cout << " Introduzca hectometros, decametros y metros ";
  cin >> hectometros >> decametros >> metros;
  decimetros = ((hectometros * 10 + decametros) * 10 + metros) * 10;
  cout << " numero de decimetros es "<< decimetros << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Un resultado de ejecución del programa anterior es:

```
 Introduzca hectometros, decametros y metros 4 35 5
 numero de decimetros es 7550
Presione una tecla para continuar . . .
```

3.8. *Para obtener el menor número de billetes y monedas de euro basta con comenzar por el billete o moneda de más alto valor. Para obtener el número o cantidad que hay que tomar de esa moneda o billete, se calcula el cociente de la cantidad dividido por el valor. El resto de la división entera de la cantidad dividido por el valor es la nueva cantidad con la que hay que volver a hacer lo mismo, pero con el siguiente billete o moneda en valor. Si se considera el sistema monetario del euro, los billetes y monedas son: billetes (quinientos, doscientos, cien, cincuenta, veinte, diez y cinco euros); monedas (dos y un euro; cincuenta, veinte, diez, cinco, dos y un céntimo de euro). El programa lee la cantidad en euros en una variable real* `CantidadOriginal`. *La transforma a céntimos de euros y se realizan los cálculos. Posteriormente se visualizan los resultados.*

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

int main(int argc, char *argv[])
{
  int  quinientos, doscientos, cien, cincuenta, veinte, diez;
  int cinco, dos, uno, cincuentac, veintec, diezc, cincoc, dosc, unc;
  float CantidadOriginal;
  long int cantidad;

  cout << "Introduzca cantidad en euros ";
  cin >> CantidadOriginal;
  CantidadOriginal*=100;  // se pasa de euros con decimales a centimos
  cantidad = (int) CantidadOriginal;   // se trunca a centimos de euro
  quinientos = cantidad / 50000;  cantidad = cantidad % 50000;
  doscientos = cantidad / 20000;  cantidad = cantidad % 20000;
  cien = cantidad / 10000;  cantidad = cantidad % 10000;
  cincuenta = cantidad / 5000;  cantidad = cantidad % 5000;
  veinte = cantidad / 2000;  cantidad = cantidad % 2000;
```

```
diez = cantidad / 1000;  cantidad = cantidad % 1000;
cinco = cantidad / 500;  cantidad = cantidad % 500;
dos = cantidad / 200;  cantidad = cantidad % 200;
uno = cantidad / 100; cantidad =cantidad % 100;
cincuentac = cantidad / 50; cantidad = cantidad % 50;
veintec = cantidad / 20;  cantidad = cantidad % 20;
diezc = cantidad / 10;  cantidad = cantidad % 10;
cincoc = cantidad / 5; cantidad = cantidad % 5;
dosc = cantidad / 2;  cantidad = cantidad % 2;
unc = cantidad;
cout << " cambio en moneda con el menor numero " << endl;
cout << " cantidad original en centimos: "<< CantidadOriginal << endl;
cout << " billetes de quinientos euros: " <<  quinientos << endl;
cout << " billetes de doscientos euros: "<< doscientos << endl;
cout << " billetes de cien euros : " <<  cien << endl;
cout << " billetes de cincuenta euros: " <<  cincuenta << endl;
cout << " billetes de veinte euros: "<< veinte << endl;
cout << " billetes de diez euros : " <<  diez << endl;
cout << " billetes de cinco euros: " <<  cinco << endl;
cout << " monedad de dos euros: "<< dos << endl;
cout << " monedad de un euro: " <<  uno << endl;
cout << " monedas de cincuenta centimos de euros: "<<cincuentac<<endl;
cout << " monedad de veinte centimos de eruro: "<< veintec << endl;
cout << " monedad de diez centimos de euro: " <<  diezc << endl;
cout << " monedas de cinco centimos de euros: " <<  cincoc << endl;
cout << " monedad de dos centimos de eruro: "<< dosc << endl;
  cout << " monedad de un centimo de euro: " <<  unc << endl;
  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Una ejecución del programa anterior es:

```
Introduzca cantidad en euros 9988.88
cambio en monedad con el menor numero
 cantidad original en centimos: 998888
 billetes de quinientos euros: 19
 billetes de doscientos euros: 2
 billetes de cien euros : 0
 billetes de cincuenta euros: 1
 billetes de veinte euros: 1
 billetes de diez euros : 1
 billetes de cinco euros: 1
 monedad de dos euros: 1
 monedad de un euro: 1
 monedas de cincuenta centimos de euros: 1
 monedad de veinte centimos de eruro: 1
 monedad de diez centimos de euro: 1
 monedas de cinco centimos de euros: 1
 monedad de dos centimos de eruro: 1
 monedad de un centimo de euro: 1
Presione una tecla para continuar . . .
```

EJERCICIOS PROPUESTOS

3.1. Escribir un programa que acepte un año escrito en cifras arábigas y visualice el año escrito en números romanos, dentro del rango 1000 a 2100.

Nota: Recuerde que V = 5, X = 10, L = 50, C = 100, D = 500 y M = 1.000.

IV = 4 XL = 40 CM = 900
MCM = 1900 MCML = 1950 MCMLX = 196
MCMXL= 1940 MCMLXXXIX = 1989

3.2. Escribir un programa que lea la hora de un día de notación de 24 horas y la respuesta en notación de 12 horas. Por ejemplo, si la entrada es 13:45, la salida será:

1: 45 PM

El programa pedirá al usuario que introduzca e xactamente cinco caracteres. Por ejemplo, las nueve en punto se introduce como

09:00

3.3. Escribir un programa que determine si un año es bisiesto. Un año es bisiesto si es múltiplo de 4 (por ejemplo 1984). Sin embargo, los años múltiplos de 100 sólo son bisiestos cuando a la v ez son múltiples de 400 (por ejemplo, 1800 no es bisiesto, mientras que 2000 si lo es).

3.4. Construir un programa que indique si un número introducido por teclado es positi vo, igual a cero, o negativo, utilizar para hacer la selección el operador ?.

3.5. Escribir un programa que lea dos enteros y calcule e imprima su producto, cociente y el resto cuando el primero se divide por el segundo.

3.6. Escribir un programa que lea tres números y nos escriba el mayor y el menor.

3.7. Escribir un programa que solicite al usuario la longitud y anchura de una habitación y, a continuación, visualice su superficie y perímetro.

3.8. Escribir un programa que lea cuatro números y calcule la media aritmética.

3.9. Escribir un programa que lea el radio de un círculo y calcule su área, así como la longitud de la circunferencia de ese radio.

3.10. Escribir un programa que lea el radio y la altura de un cono y calcule su volumen y área total.

3.11. Escribir un programa que lea tres enteros de tres dígitos y calcule y visualice su suma y su producto. La salida será justificada a derecha.

3.12. Escribir un programa que lea tres números y si el tercero es positi vo calcule y escriba la suma de los tres números, y si es ne gativo calcule y escriba su producto.

3.13. Se desea calcular el salario neto semanal de los trabajadores de una empresa de acuerdo a las siguientes normas:

Horas Semanales trabajadas < 38 a una tasa dada.
Horas extras (38 o más) a una tasa 50 por 100 superior a la ordinaria.
Impuestos 0 por 100, si el salario bruto es menor o igual a 600 euros.
Impuestos 10 por 100, si el salario bruto es mayor de 600 euros.