

# ESTRUCTURA DE LENGUAJE DE PROGRAMACIÓN C++

## Unidad 2

**Operaciones básicas y  
su jerarquía aplicada en  
el lenguaje C++**

## Tabla de Contenido

|   |    |
|---|----|
| 1. Introducción .....   | 3  |
| 2. Estructura de contenido .....  | 4  |
| 3. Operaciones básicas y su jerarquía aplicada en el lenguaje C++ ..... | 5  |
| 3.1. Operadores y operaciones matemáticas .....                         | 5  |
| 3.1.1. Operaciones de asignación .....                                  | 5  |
| 3.1.2. Operadores aritméticos .....                                     | 6  |
| 3.1.3. Operadores relacionales .....                                    | 6  |
| 3.1.4. Operadores lógicos .....   | 7  |
| 3.2. Expresiones y funciones matemáticas .....                          | 10 |
| 4. Material de apoyo .....  | 16 |
| 5. Glosario .....   | 17 |
| 6. Referencias bibliográficas .....                                     | 18 |
| 7. Control del documento .....  | 18 |
| Créditos .....  | 19 |
| Creative Commons .....  | 19 |



## 1. Introducción

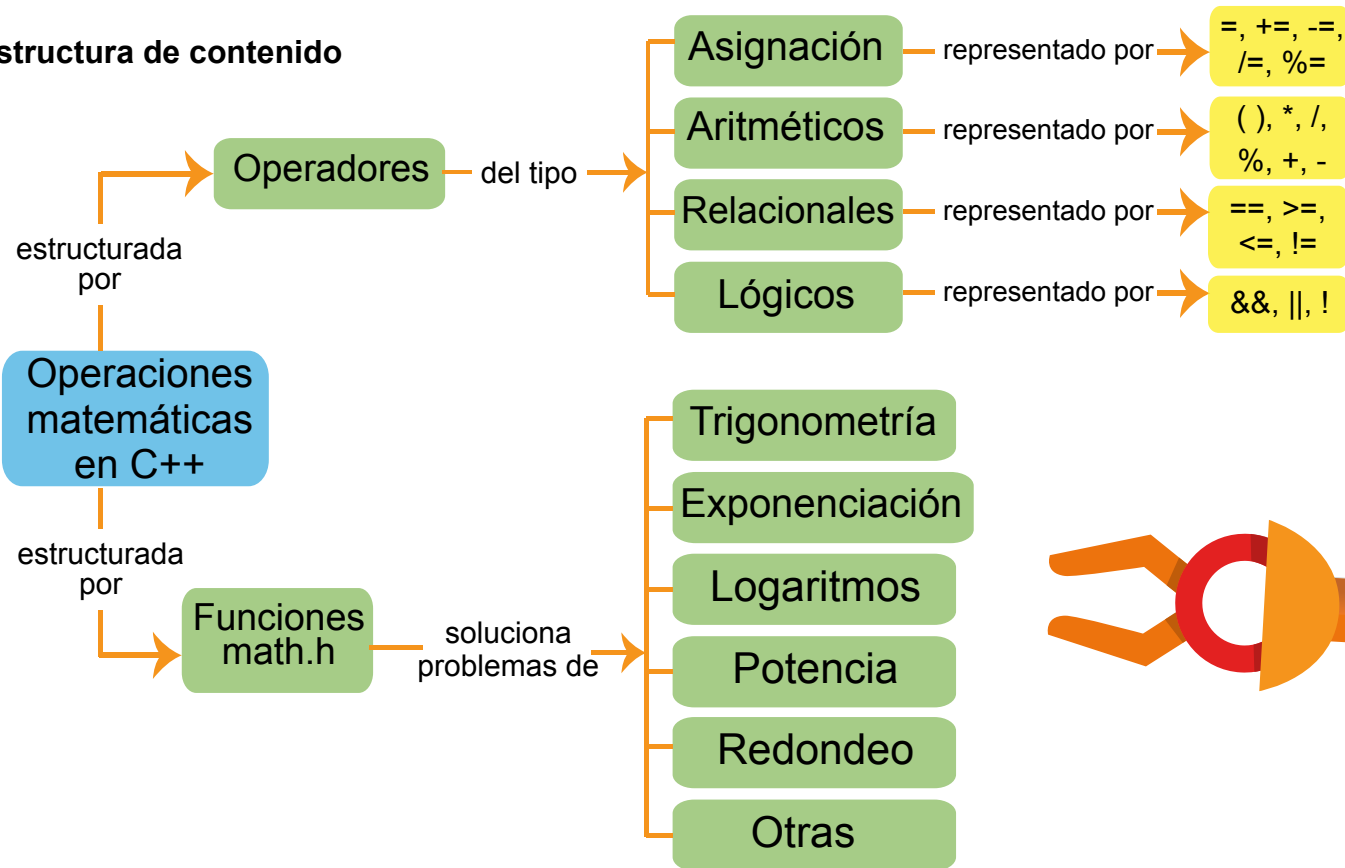
La solución de problemas aritméticos, lógicos y relacionales está dada por la correcta definición de sus expresiones, teniendo en cuenta la organización de sus elementos (operadores y operandos). En el caso de la programación de sistemas informáticos, los operandos son establecidos como variables, constantes o el valor retornado por alguna función predefinida por el lenguaje o establecida por el usuario.

En este resultado de aprendizaje, se profundizará en los operadores aritméticos, relacionales, lógicos y de asignación; esenciales para el nivel de aprendizaje abordado. La identificación de los tipos de datos hace parte de la correcta definición de una expresión. El resultado esperado dependerá también de la declaración del tipo de variable que ocupará el lugar de un operando.

C++ brinda al programador la alternativa de definir expresiones dependiendo de la solución que se requiere. Además, el lenguaje también cuenta con funciones predefinidas de diferente propósito, funciones que requieren de algunos parámetros y que permiten hallar cálculos de forma precisa.



## 2. Estructura de contenido



### 3. Operaciones básicas y su jerarquía aplicada en el lenguaje C++

#### 3.1. Operadores y operaciones matemáticas

En el estudio de la primera unidad de aprendizaje, se realizó un acercamiento a la temática del presente documento al realizar la suma de dos números, en otro programa que se desarrolló no fue necesaria la aplicación de operadores aritméticos o la definición de expresiones, pero en general, la tarea de realizar cálculos está presente en la mayoría de las aplicaciones creadas en C++.

A continuación, se explicarán los distintos tipos de operadores, dando inicio por los de asignación y continuando con los operadores aritméticos, relacionales y lógicos:

##### 3.1.1. Operadores de asignación

Las asignaciones en C++ están dadas por la expresión:

Variable = expresión;

Donde a la variable se le asigna el valor contenido en la expresión, cuyo valor es convertido implícitamente al tipo de dato definido para la variable. Dentro de los operadores de asignación se encuentran los siguientes:

| Símbolo | Uso o sentencia abreviada | Descripción  | Sentencia no abreviada |
|---------|---------------------------|--|------------------------|
| =       | a = b;                    | Se asigna el valor de b a la variable a.                                 | a = b;                 |
| +=      | a += b;                   | Suma b y a y lo asigna a la variable a.                                  | a = a + b;             |
| - =     | a -= b;                   | Resta b de a y asigna el resultado a la variable a.                      | a = a - b;             |
| *=      | a *= b;                   | Multiplica b por a y se asigna el resultado a la variable a.             | a = a * b;             |
| /=      | a /= b;                   | Divide a entre b y se asigna el resultado (el cociente) a la variable a. | a = a / b;             |
| %=      | a %= b;                   | Asigna a la variable a el residuo de la división de a entre b.           | a = a / b;             |





### 3.1.2. Operadores aritméticos

Los operadores aritméticos, permiten definir expresiones para realizar cálculos de suma (+), resta (-), multiplicación (\*), división (/) y módulo o residuo de una división (%). En C++ las reglas de jerarquía son las mismas que en las operaciones matemáticas, pero el paréntesis se puede utilizar para cambiar ese orden. Si se encuentran operadores de la misma jerarquía la evaluación será de izquierda a derecha. La siguiente tabla muestra la jerarquía de los operadores:

| Jerarquía | Operador | Operación                        |
|-----------|----------|----------------------------------|
| 1         | ()       | Paréntesis                       |
| 2         | *, /, %  | Multiplicación, división, modulo |
| 3         | a -= b;  | Suma y resta                     |

Ejemplos:

- Hallar el resultado de la siguiente expresión:  
 $19 - 3 * 4 + 39$ .  
Resultado =  $19 - 3 * 4 + 39$  // se resuelve primero la multiplicación  
Resultado =  $19 - 12 + 39$  // se resuelve la resta y después se suman  
Resultado =  $7 + 39$   
Resultado = 46

- Hallar el resultado de la siguiente expresión:  
 $6 * (8 - 1 * 9 + 22) - 7$ .  
Resultado =  $6 * (8 - 1 * 9 + 22) - 7$  // se resuelva primero lo que hay en el paréntesis.  
Resultado =  $6 * (8 - 9 + 22) - 7$   
Resultado =  $6 * (-1 + 22) - 7$   
Resultado =  $6 * 21 - 7$   
Resultado =  $126 - 7$   
Resultado = 119
- Hallar el resultado de la siguiente expresión:  
 $3 * 4 - 8 \% 6 * 6 + 7$ .  
Resultado =  $3 * 4 - 8 \% 6 * 6 + 7$  // como tenemos operadores de \* y % pero no hay un paréntesis el de mayor jerarquía es el que se encuentra a la izquierda.  
Resultado =  $12 - 8 \% 6 * 6 + 7$   
Resultado =  $12 - 2 * 6 + 7$   
Resultado =  $12 - 12 + 7$   
Resultado =  $0 + 7$   
Resultado = 7

### 3.1.3. Operadores relacionales

Una expresión que contenga un operador relacional dará como resultado una variable del tipo booleano (bool en C++), lo cual indica que solamente puede obtener dos valores: falso (false) o verdadero (true); C++ representa el valor de false como el valor entero cero (0) y cualquier valor distinto de cero representa



el valor true. Así mismo, una expresión relacional representa la comparación de dos operandos compatibles en C++.

A continuación, se presenta la tabla de los operadores relacionales:

| Operador | Significado     | Ejemplo            |
|----------|-----------------|--------------------|
| ==       | Igual a         | 'J' == 'E' (false) |
| !=       | Diferente de    | 4 != 8 (true)      |
| >        | Mayor a         | 9 > 12 (false)     |
| <        | Menor a         | 9 < 12 (true)      |
| >=       | Mayor o igual a | 1 >= 2 (false)     |
| <=       | Menor o igual a | 2 <= 2 (true)      |



### 3.1.4. Operadores lógicos

Al igual que los operadores relacionales, una expresión que contenga operadores lógicos dará como resultado false o true. En la siguiente tabla se describen algunos ejemplos:

| Operador | Descripción  | Ejemplo                     |
|----------|--|-----------------------------|
| ! (not)  | Niega el resultado de la expresión. Si es true el resultado final será false y viceversa.        | !(9 > 12) = true            |
| && (and) | El resultado es true cuando las dos expresiones son true. Cualquier otra combinación dará false. | (9 < 12) && (2 <= 2) = true |
| (or)     | El resultado es true cuando alguna de las dos expresiones es true, o ambas.                      | (1 >= 2)    (4!=8) = true   |



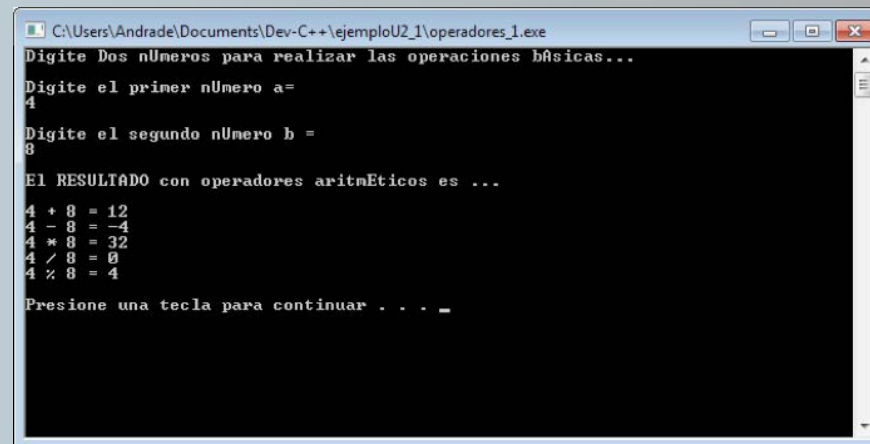
A continuación, se mostrará un ejemplo que permitirá poner en práctica los operadores aritméticos y de asignación. Los operadores de relación y lógicos se pondrán en práctica en la unidad temática 3 cuando se exponga el tema de las estructuras de condición:

Por el momento, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de operadores\_1 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int a, b, suma, resta, mult, div, mod, resultado;
8
9     cout << "Digite Dos números para realizar las operaciones básicas..." ;
10    cout << endl << endl << "Digite el primer número a = " << endl;
11    cin >> a;
12    cout << endl << "Digite el segundo número b = " << endl;
13    cin >> b;
14
15    suma = a + b;
16    resta = a - b;
17    mult = a * b;
18    div = a / b;
19    mod = a % b;
20
21    cout << endl << "El RESULTADO con operadores aritméticos es ...";
22    cout << endl << endl;
23
24    cout << a << " + " << b << " = " << suma << endl;
25    cout << a << " - " << b << " = " << resta << endl;
26    cout << a << " * " << b << " = " << mult << endl;
27    cout << a << " / " << b << " = " << div << endl;
28    cout << a << " % " << b << " = " << mod << endl << endl;
29
30    system("pause");
31    return EXIT_SUCCESS;
32 }
33 }
```

Ilustración 1. Código Proyecto operadores\_1.  
Fuente: Sena

Como se puede observar, entre las líneas 15 y 19 se encuentra código que hasta el momento es nuevo para el proceso de formación y representa las operaciones básicas de la aritmética. La salida en pantalla de la aplicación anterior es la siguiente:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU2_1\operadores_1.exe
Digite Dos números para realizar las operaciones básicas...
Digite el primer número a=
4
Digite el segundo número b =
8
El RESULTADO con operadores aritméticos es ...
4 + 8 = 12
4 - 8 = -4
4 * 8 = 32
4 / 8 = 0.5
4 % 8 = 4
Presione una tecla para continuar . . . _
```

Ilustración 2. Interfaz de usuario Proyecto operadores\_1.  
Fuente: Sena





Elabore un nuevo proyecto en Dev C++ y guárdelo con el nombre de operadores\_2, en él se hará una demostración de los operadores de asignación. Digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int a, b;
8
9     cout << "Digite Dos nUmeros para realizar las operaciones bAsicas..." ;
10    cout << endl << endl << "Digite el primer nUmero a= " << endl;
11    cin >> a;
12    cout << endl << "Digite el segundo nUmero b = " << endl;
13    cin >> b;
14
15    cout << endl << "El RESULTADO con operadores de AsignaciOn es ...";
16    cout << endl << endl;
17    cout << "En este momento 'a' vale " << a << " y el valor de 'b' es " << b;
18    cout << endl;
19    cout << a << " += " << b << " = ";
20    a += b;
21    cout << a << endl << endl;
22
23    cout << "En este momento 'a' vale " << a << " y el valor de 'b' es " << b;
24    cout << endl;
25    cout << a << " -= " << b << " = " ;
26    a -= b;
27    cout << a << endl << endl;
28
29    cout << "En este momento 'a' vale " << a << " y el valor de 'b' es " << b;
30    cout << endl;
31    cout << a << " *= " << b << " = " ;
32    a *= b;
33    cout << a << endl << endl;
34
35    cout << "En este momento 'a' vale " << a << " y el valor de 'b' es " << b;
36    cout << endl;
37    cout << a << " /= " << b << " = " ;
38    a /= b;
39    cout << a << endl << endl;
40
41    cout << "En este momento 'a' vale " << a << " y el valor de 'b' es " << b;
42    cout << endl;
43    cout << a << " %= " << b << " = " ;
44    a %= b;
45    cout << a << endl << endl;
46
47    system("pause");
48    return EXIT_SUCCESS;
49
50 }
```

Ilustración 3.  
Código Proyecto operadores\_2.  
Fuente: Sena

Al realizar las operaciones en las líneas 20, 26, 32, 38 y 44 el valor de la variable 'a' se va modificando, mientras que el valor de 'b' siempre es el mismo:

```
20      a += b;  
26      a -= b;  
32      a *= b;  
38      a /= b;  
44      a %= b;
```

La salida en pantalla de la aplicación anterior es la siguiente:

```
C:\Users\Andrade\Documents\Dev-C++\ejemploU2_2\operadores_2.exe  
Digite Dos números para realizar las operaciones básicas...  
Digite el primer número a=  
8  
Digite el segundo número b =  
5  
El RESULTADO con operadores de asignación es ...  
En este momento 'a' vale 8 y el valor de 'b' es 5  
8 += 5 = 13  
En este momento 'a' vale 13 y el valor de 'b' es 5  
13 -= 5 = 8  
En este momento 'a' vale 8 y el valor de 'b' es 5  
8 *= 5 = 40  
En este momento 'a' vale 40 y el valor de 'b' es 5  
40 /= 5 = 8  
En este momento 'a' vale 8 y el valor de 'b' es 5  
8 %= 5 = 3  
Presione una tecla para continuar . . .
```

Ilustración 4. Interfaz de usuario Proyecto operadores\_2.  
Fuente: Sena

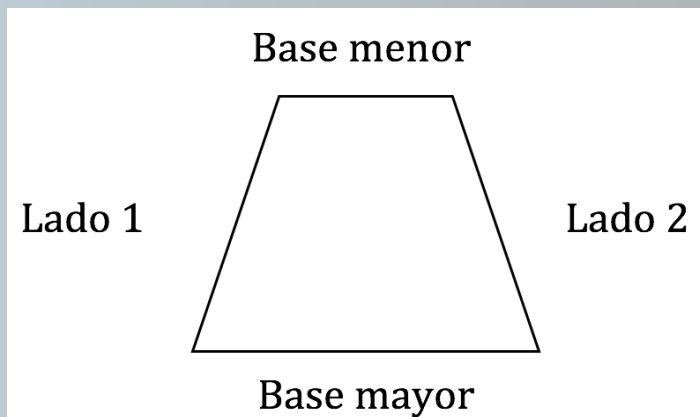
### 3.2. Expresiones y funciones matemáticas

En los ejemplos anteriores se evidenciaron algunas expresiones matemáticas:

```
15      suma = a + b;  
16      resta = a - b;  
17      mult = a * b;  
18      div = a / b;  
19      mod = a % b;
```

Como se puede observar, están compuestas por operadores y operandos con el fin de hallar un resultado. Están definidas con dos operandos, pero las expresiones pueden contener más variables y operadores.

Ejemplo: para hallar el perímetro de un trapecio es necesario sumar sus cuatro lados.



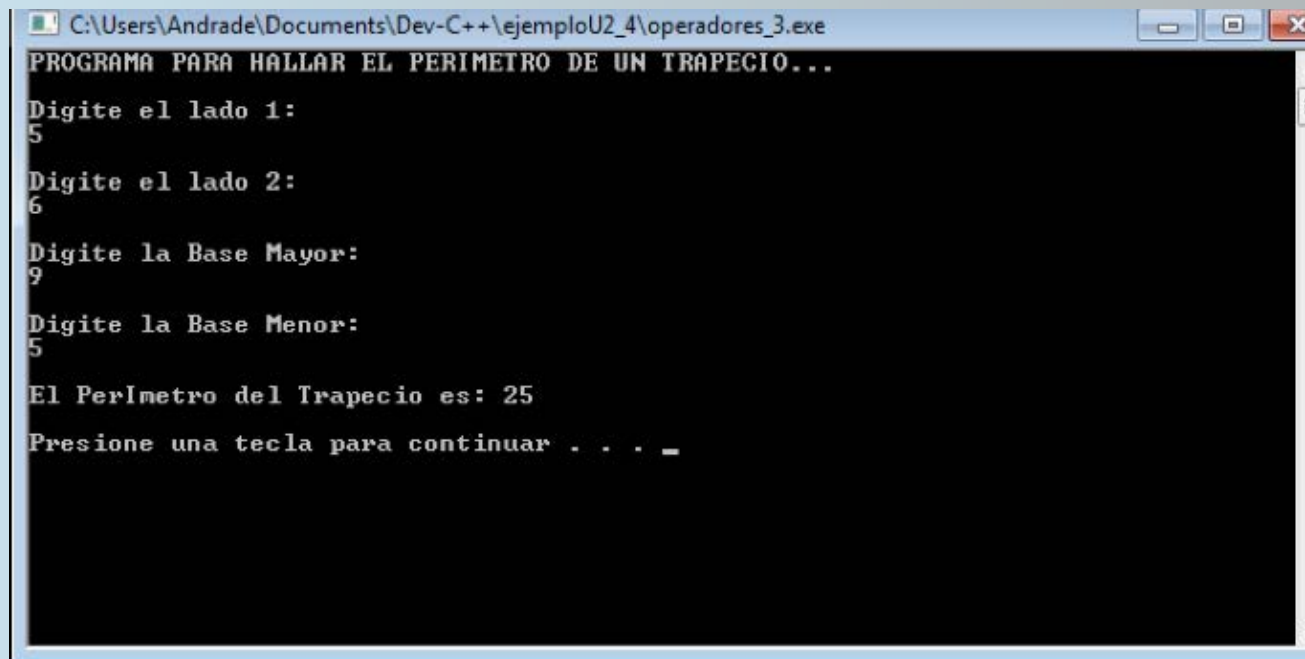
Por consiguiente, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de operadores\_3 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7
8     float lado_1, lado_2, base_Mayor, base_Menor, perimetro;
9
10    cout << "PROGRAMA PARA HALLAR EL PERIMETRO DE UN TRAPECIO..." << endl << endl;
11    cout << "Digite el lado 1: " << endl;
12    cin >> lado_1;
13    cout << endl;
14
15    cout << "Digite el lado 2: " << endl;
16    cin >> lado_2;
17    cout << endl;
18
19    cout << "Digite la Base Mayor: " << endl;
20    cin >> base_Mayor;
21    cout << endl;
22
23    cout << "Digite la Base Menor: " << endl;
24    cin >> base_Menor;
25    cout << endl;
26
27    perimetro = lado_1 + lado_2 + base_Mayor + base_Menor;
28
29    cout << "El PerImetro del Trapecio es: " << perimetro << endl << endl;
30
31    system("pause");
32    return EXIT_SUCCESS;
33
34 }
```

Ilustración 6. Código Proyecto operadores\_3.

Fuente: Sena

La salida del ejercicio debe ser la siguiente:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU2_4\operadores_3.exe
PROGRAMA PARA HALLAR EL PERIMETRO DE UN TRAPECIO...
Digite el lado 1:
5
Digite el lado 2:
6
Digite la Base Mayor:
9
Digite la Base Menor:
5
El PerImetro del Trapecio es: 25
Presione una tecla para continuar . . . _
```

Como se observa en la línea 27, se define la expresión que da solución al problema, compuesta por cuatro (4) operandos y tres (3) operadores:

```
27  perimetro = lado_1 + lado_2 + base_Mayor + base_Menor;
```

Por otro lado, C++ cuenta con funciones propias para realizar cálculos matemáticos disponibles en la Librería <math.h>, la cual tiene definidas funciones para solucionar problemas trigonométricos, exponenciales, logarítmicos, de potencia, de redondeo, mínimo, máximo, entre otros.





En consecuencia, se expone un ejemplo con la demostración de algunas funciones. Así que, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de funciones\_math y digite el siguiente código:

```
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4
5 int main()
6 {
7     cout << "DEMOSTRACION DE FUNCIONES MATEMATICAS EN C++" << endl << endl;
8     cout << "FunciOn fmax para hallar el MAYOR de dos nUmeros: fmax(18.9, 18) = ";
9     cout << fmax(18.9, 18) << endl << endl;
10    cout << "FunciOn fmin para hallar el MENOR de dos nUmeros: fmin(18.9, 18) = ";
11    cout << fmin(18.9, 18) << endl << endl;
12    cout << "FunciOn ceil para redondear a una cifra superior: ceil(18.7) = ";
13    cout << ceil(18.7) << endl << endl;
14    cout << "FunciOn floor para redondear a una cifra inferior: floor(18.7) = ";
15    cout << floor(18.7463) << endl << endl;
16    cout << "FunciOn pow para elevar un nUmero a una potencia: pow(9, 2) = ";
17    cout << pow(9, 2) << endl << endl;
18    cout << "FunciOn sqrt para Hallar la raiz cuadrada de un nUmero: sqrt(81) = ";
19    cout << sqrt(81) << endl << endl;
20    cout << "FunciOn hypot para Hallar la HIPOTENUSA de un triAngulo: hypot(81) = ";
21    cout << hypot(3, 4) << endl << endl;
22
23    system("pause");
24    return EXIT_SUCCESS;
25 }
```

A continuación, se brindará una explicación de la aplicación de las funciones y de los parámetros requeridos por cada una:

```
2 #include <math.h>
```

Primero, en la línea dos (2) se incluye en la cabecera la librería math.h para invocar las funciones.

```
9 cout << fmax(18.9, 18) << endl << endl;
```

En la línea 9 se define la función fmax (variable x, variable y) permite hallar el valor máximo entre dos variables. Éstas deben ir en el paréntesis y separadas por una coma.

```
11 cout << fmin(18.9, 18) << endl << endl;
```

La función fmin (variable x, variable y) permite hallar el valor mínimo entre dos variables.

```
13 cout << ceil(18.7) << endl << endl;
```

En la línea 13 se define la función ceil (variable x) que redondea un valor a una cifra superior. Esta función requiere como parámetro una sola variable.

```
15 cout << floor(18.7463) << endl << endl;
```

En la línea 15 se define la función floor(variable x) que redondea un valor a una cifra superior

```
17 cout << pow(9, 2) << endl << endl;
```

La función pow(variable x, variable y) eleva el primer parámetro a la potencia establecida en el segundo parámetro.

```
19 cout << sqrt(81) << endl << endl;
```

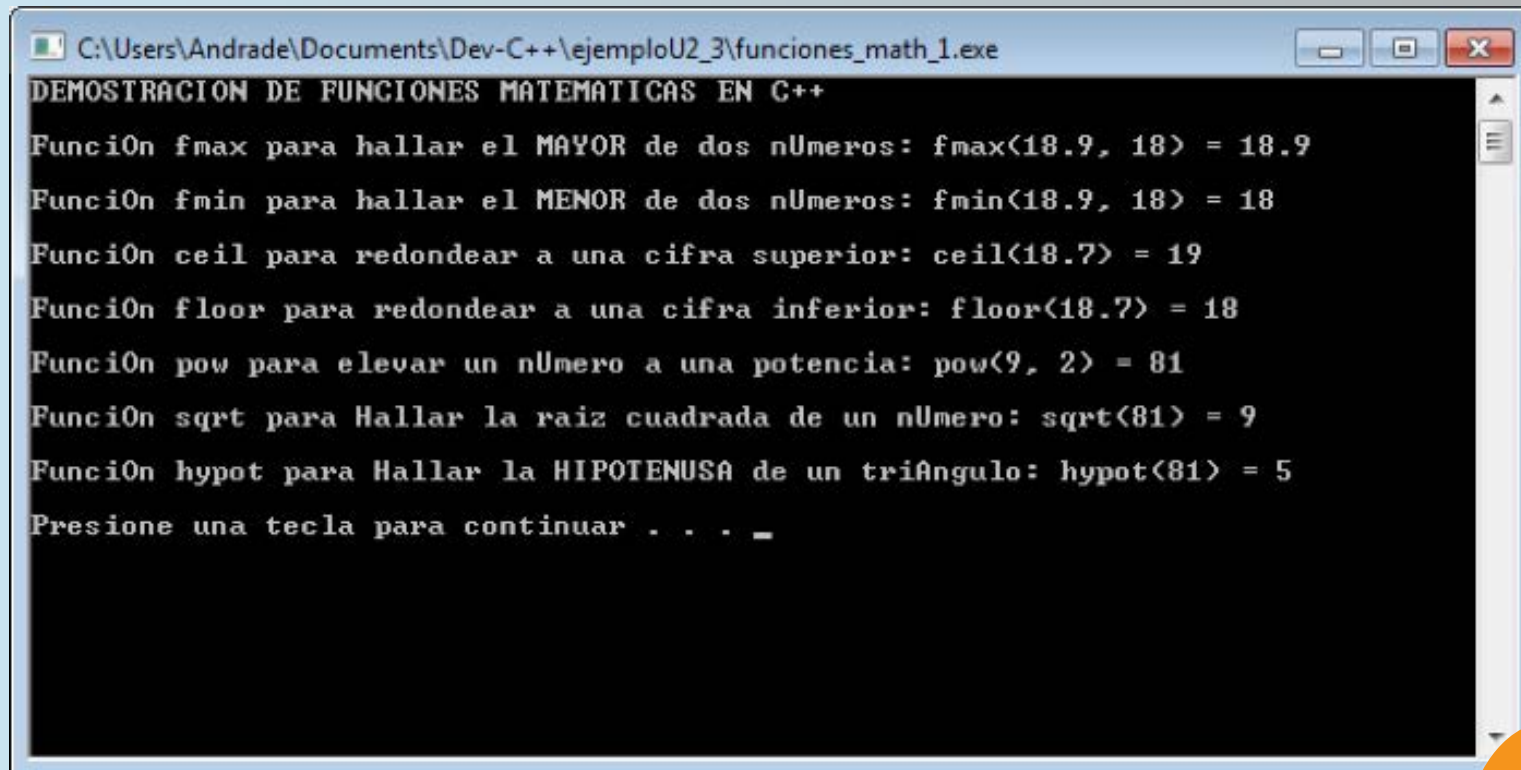
En la línea 19 se define la función sqrt(variable x) que permite hallar la raíz cuadrada de un número.

```
21 cout << hypot(3, 4) << endl << endl;
```

En la línea 21 se define la función hypot (variable x, variable y) que devuelve el valor de la hipotenusa de un triángulo rectángulo. En este caso como parámetros se requieren sus catetos.



Al ejecutar la aplicación el resultado a mostrar es el siguiente:



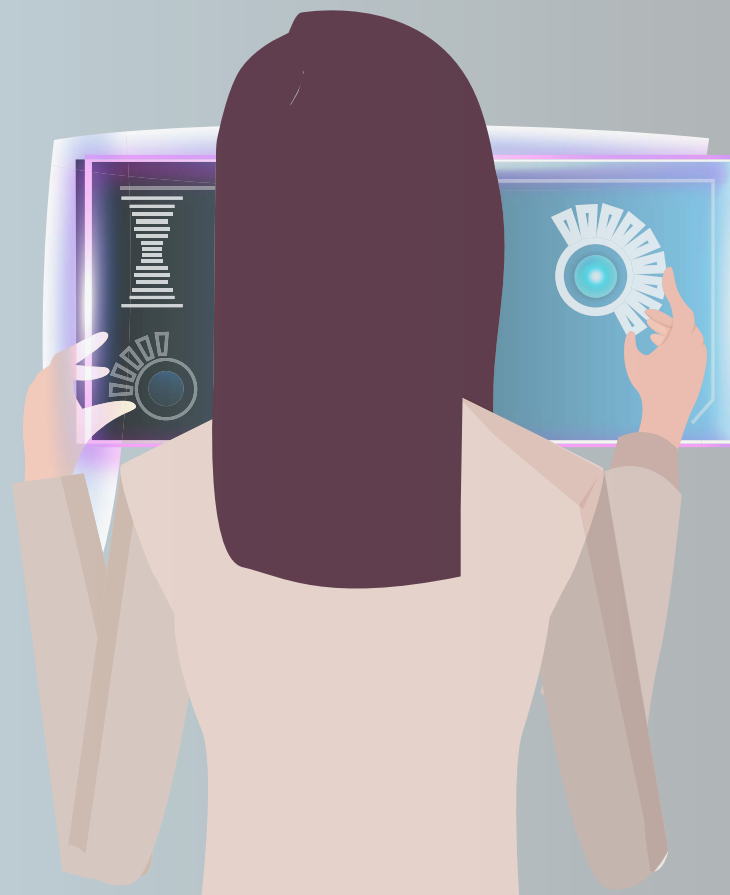
```
C:\Users\Andrade\Documents\Dev-C++\ejemploU2_3\funciones_math_1.exe
DEMOSTRACION DE FUNCIONES MATEMATICAS EN C++
Funcion fmax para hallar el MAYOR de dos nUmeros: fmax<18.9, 18> = 18.9
Funcion fmin para hallar el MENOR de dos nUmeros: fmin<18.9, 18> = 18
Funcion ceil para redondear a una cifra superior: ceil<18.7> = 19
Funcion floor para redondear a una cifra inferior: floor<18.7> = 18
Funcion pow para elevar un nUmero a una potencia: pow<9, 2> = 81
Funcion sqrt para Hallar la raiz cuadrada de un nUmero: sqrt<81> = 9
Funcion hypot para Hallar la HIPOTENUSA de un triAngulo: hypot<81> = 5
Presione una tecla para continuar . . . _
```

Ilustración 9. Interfaz de usuario Proyecto funciones\_math.  
Fuente: Sena



#### 4. Material de apoyo

- Descripción de la librería <math.h>
- Programación en C++.
- Enciclopedia del lenguaje C++







ABCD

## 5. Glosario

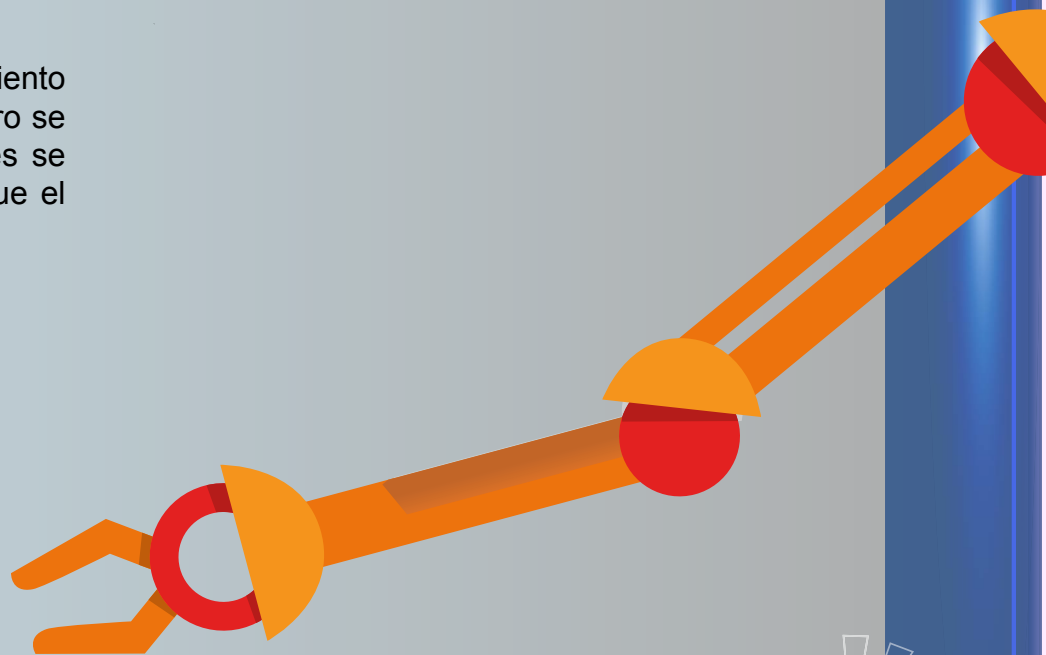
**Estatuto:** es un sinónimo de instrucción.

**Estatuto break:** instrucción que termina inmediatamente un ciclo o una instrucción switch.

**Estatuto for:** es un método para ejecutar un bloque de sentencias un número fijo de repeticiones.

**Estatuto do/while:** tiene un comportamiento similar a while, sólo que en este caso; primero se ejecuta el bloque de instrucciones y después se evalúa la condición. Con esto se asegura que el bloque se ejecutará al menos una vez.

**Estatuto while:** se usa para implementar una estructura de repetición (bucle while) en la que la repetición se controla mediante una expresión booleana y continúa ejecutándose mientras esta expresión permanece cierta, finalizando cuando se hace falsa.





## 6. Referencias bibliográficas

Ceballos, S. F. J. (2009). Enciclopedia del lenguaje C++ (2a. ed.). Madrid, ES: RA-MA Editorial.

Joyanes, L. Sánchez, L. (2006). Programación en C++: un enfoque práctico. España: McGraw-Hill.

Joyanes, L. Zahonero, I. (2007). Estructura de Datos en C++. España: McGraw-Hill.

## 10. Control del documento

|              | Nombre                     | Cargo            | Versión | Fecha         |
|--------------|----------------------------|------------------|---------|---------------|
| <b>Autor</b> | Jorge Eliecer Andrade Cruz | Experto temático | 1       | Junio de 2017 |





### **Créditos**

**Equipo de Adecuación Gráfica**  
**Centro de Comercio y servicios**  
**SENA Regional Tolima**  
Línea de Producción

### **Director Regional**

Félix Ramón Triana Gaitán

### **Subdirector de Centro**

Álvaro Fredy Bermúdez Salazar

### **Coordinadora de formación profesional**

Gloria Ines Urueña Montes

### **Experto temático**

Jorge Eliecer Andrade Cruz

### **Senior equipo de adecuación**

Claudia Rocio Varón Buitrago

### **Asesor pedagógico**

Ricardo Palacio

### **Guionistas**

Genny Carolina Mora Rojas  
Jesús Bernardo Novoa Ortiz

### **Diseño y diagramación**

Diana Katherine Osorio Useche  
Pedro Nel Cabrera Vanegas  
Ismael Enrique Cocomá Aldana

### **Programadores**

Davison Gaitán Escobar  
Héctor Horacio Morales García  
Iván Darío Rivera Guzmán



### **Creatives commons**

**Atribución, no comercial, compartir igual.**

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo comercial.

