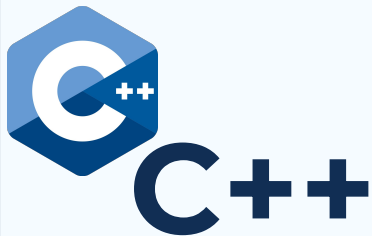


Introducción a la programación



C++ es un lenguaje de programación de alto nivel que combina la programación procedural y orientada a objetos, diseñado para brindar flexibilidad y control, ampliamente utilizado en el desarrollo de software de sistemas, aplicaciones y videojuegos.





CONCEPTOS INICIALES

- **Sentencias:** Instrucción
- **Bloques de Código:** Grupo de Instrucciones agrupadas por caracteres específicos "{" "}"
- **Variable:** Espacio reservado en memoria para almacenar un dato
- **Ámbito:** Contexto que pertenece a un nombre dentro de un programa. (Determina en qué partes del programa una entidad puede ser usada.)
- **Variable local:** Variable accedida únicamente en su bloque de código.
- **Variable global:** Variable accedida por todos los miembros de la entidad.



Declaración de Variables

Una declaración de variable es una instrucción que crea una variable con un **identificador** y un **tipo de dato**. Las variables almacenan información y se utilizan para realizar operaciones.

```
C++ main.cpp > main()
1  int main() {
2      int edad = 22; // Declaración de una variable llamada 'edad' de tipo entero
3  }
4
```



Tipo de Datos

Nombre	Descripción	Tamaño*	Rango de valores*
char	Carácter o entero pequeño	1byte	con signo: -128 to 127 sin signo: 0 a 255
short int (short)	Entero corto	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int	Entero	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)	Entero largo	8bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: verdadero o falso	1byte	true o false
float	Número de punto flotante	4bytes	3.4e +/- 38 (7 digitos)
double	De punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)
long double	Long de punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)



Mostrar Mensajes en Consola

- **std::cout:** La forma más básica es usar la salida estándar `std::cout`, que está incluida en la biblioteca estándar de C++.

```
1  #include <iostream>
2
3  int main() {
4      std::cout << "Hola, mundo!" << std::endl;
5      return 0;
6  }
```

- **printf:** Esta función es parte de la biblioteca de C, pero aún se puede usar en C++.

```
main.cpp > ...
1  #include <stdio.h>
2
3  int main() {
4      printf("Hola, mundo!\n");
5      return 0;
6  }
```



Leer datos desde la consola

- **std::cout:** Se puede utilizar el objeto std::cin para leer datos desde la entrada estándar (normalmente el teclado).

```
main.cpp > ...
1  #include <iostream>
2
3  int main() {
4      int numero;
5      std::cout << "Por favor, ingresa un número: ";
6      std::cin >> numero;
7      std::cout << "Ingresaste: " << numero << std::endl;
8      return 0;
9  }
```

- **scanf:** Otra forma es usando la función de la biblioteca de C scanf.

```
main.cpp > ...
1  #include <stdio.h>
2
3  int main() {
4      int numero;
5      printf("Por favor, ingresa un número: ");
6      scanf("%d", &numero);
7      printf("Ingresaste: %d\n", numero);
8      return 0;
9  }
```



Sintaxis:

```
if ( <expresión booleana> ) {  
    <instrucciones>  
}  
else if ( <expresión booleana> ) {  
    <instrucciones>  
} else {  
    <instrucciones>  
}
```

If - Else if - Else

if: Si el valor de la expresión booleana en paréntesis es verdadero, se ejecutan las instrucciones dentro de las llaves y no sigue evaluando los otros casos. Si no, se ignoran.

else if: Luego de evaluar la expresión en “if”, si el valor de la expresión booleana es falso, se evalúa la expresión en paréntesis. Si esta expresión tiene un valor verdadero, se ejecutan las instrucciones dentro de las llaves. Si no, se ignoran. Pueden existir cero o más estructuras “else if”.

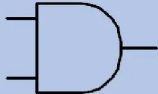
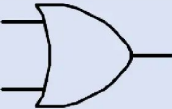
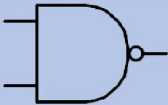
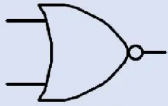
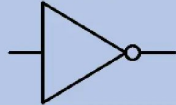
else: Luego de evaluar todas las expresiones en “if” y “else if”, si no se obtuvo expresión verdadera, se ejecutan siempre las instrucciones en “else”. Si no, se ignoran. Pueden existir cero o una estructura “else”, y siempre se encuentra al final.

OPERADORES RELACIONALES

X=5; Y=3

Operación	Operador	Ejemplo	Comparación	Resultado/Binario
Mayor que	>	X > Y	¿Es X mayor que Y?	Verdadero (1)
Menor que	<	X < Y	¿Es X menor que Y?	Falso (0)
Mayor o igual que	>=	X >= Y	¿Es X mayor o igual que Y?	Verdadero (1)
Menor o igual que	<=	X <= Y	¿Es X menor o igual que Y?	Falso (0)
Igual a	==	X == Y	¿Es X igual a Y?	Falso (0)
Diferente a	!=	X != Y	¿Es X diferente a Y?	Verdadero (1)

OPERADORES LOGICOS

Operador	Nomenclatura en Arduino	Significado	Símbolo
AND	&&	Devuelve true cuando la primera <u>Y</u> la segunda condición se cumplen	
OR		Devuelve true cuando la primera <u>O</u> la segunda condición se cumple	
NAND	(i=) && (i=)	Devuelve true cuando <u>NI</u> la primera <u>NI</u> la segunda condición se cumplen	
NOR	(i=) (i=)	Devuelve true cuando <u>NO</u> se cumple alguna de las condiciones	
NOT	i=	Devuelve true cuando <u>NO</u> se cumple una condición	



Sintaxis:

```
switch (<expresión>) {  
    case valor1:  
        <instrucciones>  
        break;  
    case valor2:  
        <instrucciones>  
        break;  
    default:  
        <instrucciones>  
}
```

Switch

switch: es la palabra clave que inicia la estructura de control.

<expresión>: es la expresión integral (como un entero o un carácter) cuyo valor se comparará con los casos.

case valor1, valor2: son etiquetas que representan valores posibles de la expresión. Dentro de cada case, se puede colocar el código que se ejecutará si la expresión es igual al valor correspondiente.

Break; : se utiliza para salir del switch después de que se ha ejecutado el código en un caso. Sin break, el flujo de ejecución continuará en los casos siguientes hasta encontrar un break o salir del switch.

default: es una etiqueta opcional que se utiliza para manejar el caso en el que ninguno de los casos coincida con la expresión. Puedes colocar código aquí para manejar esa situación.



Bucle while

Sintaxis:

```
while (<Expresión booleana> ) {  
    <Instrucciones>  
}
```

La expresión booleana se evalúa, y si su valor es verdadero, se ejecutan las instrucciones entre llaves. Luego se re-evalúa esta expresión. Si la expresión no es verdadera, se ignoran estas instrucciones. Permite instrucciones “break”.



Sintaxis:

```
switch (<expresión>) {  
    case valor1:  
        <instrucciones>  
        break;  
    case valor2:  
        <instrucciones>  
        break;  
    default:  
        <instrucciones>  
}
```

Switch

switch: es la palabra clave que inicia la estructura de control.

<expresión>: es la expresión integral (como un entero o un carácter) cuyo valor se comparará con los casos.

case valor1, valor2: son etiquetas que representan valores posibles de la expresión. Dentro de cada case, se puede colocar el código que se ejecutará si la expresión es igual al valor correspondiente.

Break; : se utiliza para salir del switch después de que se ha ejecutado el código en un caso. Sin break, el flujo de ejecución continuará en los casos siguientes hasta encontrar un break o salir del switch.

default: es una etiqueta opcional que se utiliza para manejar el caso en el que ninguno de los casos coincida con la expresión. Puedes colocar código aquí para manejar esa situación.



Bucle For

Sintaxis:

```
for ( Inicialización; condición ; actualización ) {  
    <Instrucciones>  
}
```

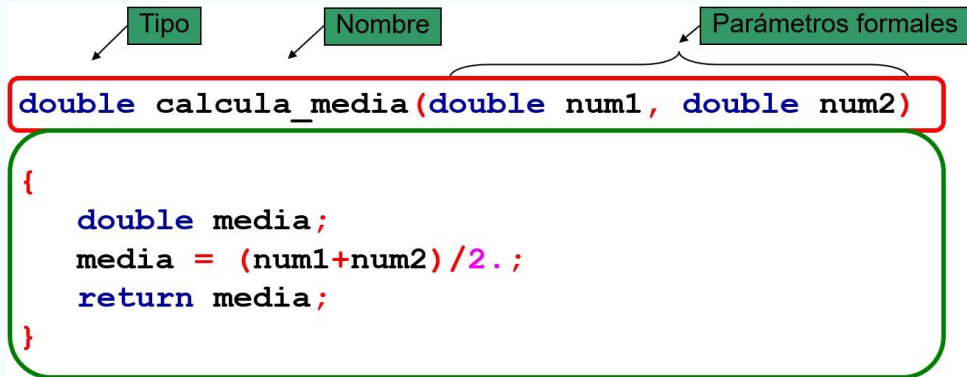
Inicialización: Esta es la parte donde generalmente se inicializan las variables que se utilizarán en el bucle. Se ejecuta una vez al principio del bucle.

condición: Aquí se verifica una condición. Si la condición es verdadera, el bucle continuará ejecutándose. Si es falsa, el bucle se detendrá. Se verifica antes de cada iteración.

actualización: En esta parte, generalmente se actualizan las variables que se utilizan en el bucle. Se ejecuta al final de cada iteración.



Funciones



- Encabezado
- Cuerpo

Las funciones se utilizan para dividir un programa en partes más pequeñas y manejables, lo que facilita la organización del código, la reutilización de código y la mejora de la legibilidad. Cada función en C++ tiene una declaración que especifica su nombre, tipo de valor de retorno (si lo tiene) y los parámetros que acepta, y una definición que contiene las instrucciones que se ejecutan cuando se llama a la función. Las funciones se llaman en otras partes del programa para realizar operaciones específicas.



Procedimientos

Ámbito de la declaración
Nombre del procedimiento
Instrucciones

```

private void limpiar ()
{
    txtNumero1.Clear();
}
    
```

Un procedimiento es un bloque de código con un nombre específico que realiza una tarea específica cuando se llama, pero no tiene un valor de retorno. Los procedimientos se utilizan para llevar a cabo acciones o tareas, cómo imprimir información en la pantalla, realizar cálculos intermedios o modificar datos, sin necesidad de devolver un valor como resultado.



Python

Python es un lenguaje de programación de alto nivel que se utiliza en una amplia variedad de aplicaciones, como desarrollo de software, análisis de datos, inteligencia artificial y desarrollo web. Es conocido por su sintaxis legible, su facilidad de aprendizaje y su versatilidad.



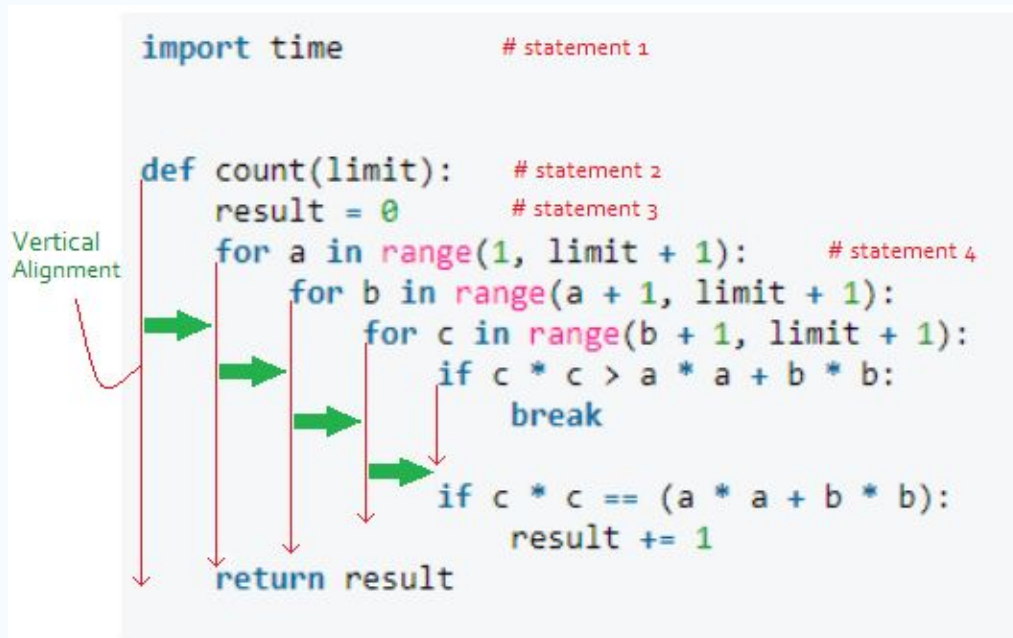


Declaración de Variables

```
7  
8     nombre="Cristian"  
9     edad=22  
10    mi_lista = []  
11
```

En Python, no necesitas declarar el tipo de variable, se asigna automáticamente. Los nombres de variables deben seguir reglas específicas. Pueden ser locales o globales. Python permite cambiar el tipo de variable durante su vida útil. Se puede usar operadores de asignación para hacer operaciones y asignar valores a variables en un solo paso.

Bloques de código (Indentación)



```

import time           # statement 1

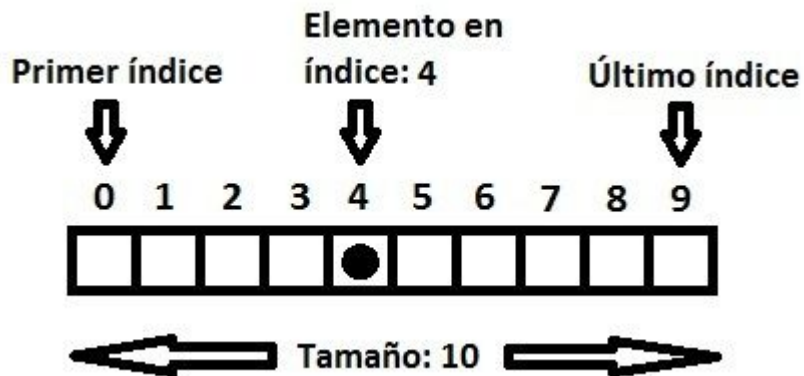
def count(limit):     # statement 2
    result = 0        # statement 3
    for a in range(1, limit + 1): # statement 4
        for b in range(a + 1, limit + 1):
            for c in range(b + 1, limit + 1):
                if c * c > a * a + b * b:
                    break
            if c * c == (a * a + b * b):
                result += 1
    return result
  
```

Vertical Alignment

La indentación en Python se refiere a la forma en que se estructuran los bloques de código. En lugar de utilizar llaves, en Python utiliza la cantidad de espacios o tabulaciones al comienzo de una línea para determinar la pertenencia de una línea a un bloque de código. La indentación es una característica fundamental de Python y contribuye a reducir errores y a mantener un estilo de codificación limpio.



Listas en Python



Una lista en Python es una estructura de datos que permite almacenar una colección ordenada de elementos de diferentes tipos. Se definen utilizando corchetes `[]` y se separan los elementos por comas. Las listas son ordenadas, lo que significa que mantienen el orden de los elementos, son mutables (se pueden modificar), y son heterogéneas (pueden contener elementos de diferentes tipos). Se puede acceder a los elementos de una lista utilizando índices.



Gracias