Logotipo

El contenido generado por IA puede ser incorrecto.

**UNIVERSIDAD TECNOLÓGICA DE HONDURAS**

**CAMPUS SANTA BÁRBARA**

**ASIGNATURA:** ANÁLISIS Y DISEÑO DE ALGORITMOS

**INVESTIGACIÓN DE C++**

**III PARCIAL**

**ALUMNO:** JOSUÉ ALEJANDRO MONTÚFAR ZÚNIGA

(202510050077)

**PRESENTADO A:**

ING/ABOG. JAIR JOSUÉ HERNÁNDEZ HERNÁNDEZ

**PERÍODO 2025-1**

**FECHA DE ENTREGA:**

MARTES 18 DE MARZO DEL 2025

**INVESTIGACIÓN DE C++**

**¿Qué son los comentarios en C++?**

Los comentarios son una herramienta proporcionada por un lenguaje de programación que nos ayudará a apoyar la documentación de los programas en C++ y a que sean fácilmente comprendidos por otros. Los comentarios, son líneas de código, que no son tenidas en cuenta por el compilador en el momento de ejecutar nuestra aplicación, por lo tanto, no están sujetas a restricciones de sintaxis ni nada similar, el uso principal de las líneas de comentario es dar un orden y hacer más entendible nuestro código, especialmente en el caso de que deba en algún momento ser revisado o leído por alguien diferente a nosotros.

Existen dos tipos de comentarios en el lenguaje:

**Comentarios de una sola línea:** Pueden ser colocados en cualquier parte y comienzan por una doble pleca "//", al colocar la doble pleca al comienzo de cualquier línea de código, todo lo que halla de ahí en adelante en esa misma línea será tomado como comentario. Ejemplo:

// Suma de dos números

**Comentarios multi línea:** Van cerrados entre "/\*" y "\*/". Estos comentarios son similares a los anteriores, pero deben tener un comienzo y un final, a diferencia del anterior, al poner los símbolos "/\*" todo el código que haya tanto en la misma línea, como en las líneas debajo de este, se convertirán en comentarios, de manera que debemos cerrarlo en el punto donde dicho comentario termina. Ejemplo:

/\*

Suma de dos números

Creado por Josué Alejandro Montúfar

2024-02-17

\*/

**¿Cuál es la estructura de C++?**

**#include <iostream.h>** Declaración de librerías

**using namespace** std**;** Espacio de nombres estándar

**int** main**(){** Función principal y llave de inicio de la función

cout**<<”Hola Mundo”<<**endl**;** Secuencia de instrucciones

**return 0;** Valor de retorno de la función

**}** Llave de cierre de la función

**#include<iostream.h>** La parte del #include se refiere a la biblioteca de funciones que vamos a utilizar. Es decir, para llamar a una biblioteca en particular debemos hacer lo siguiente:

#include<librería\_solicitada> El estándar de C++ incluye varias bibliotecas de funciones, y dependiendo del compilador que se esté usando, puede aumentar el número.

**using namespace std;** Esta línea indica que se utilizará el espacio de nombres estándar (std), que contiene muchas funciones y objetos comunes. Esto evita tener que escribir std:: antes de cada elemento de la biblioteca estándar (como cout, endl, etc.).

**int main(){** Todo programa en C++ comienza con una función main (), y sólo puede haber una. En C++ el main () siempre regresa un entero, es por eso se antepone “int” a la palabra “main”. Los paréntesis que le siguen contienen lo que se le va a mandar a la función. En este caso no hay nada y significa “void” o vacío, es decir que a la función main no se le está mandando nada. La llave que se abre significa que se iniciará un bloque de instrucciones.

**cout<<”Hola Mundo”<<endl;** Esta es una instrucción. La instrucción cout está definida dentro de la biblioteca iostream.h, que previamente declaramos que íbamos a utilizar. Una función, en este caso main() siempre comienza su ejecución con una instrucción (la que se encuentra en la parte superior), y continúa así hasta que se llegue a la última instrucción (de la parte inferior). Para terminar una instrucción siempre se coloca “;”. Pero además de instrucciones se pueden invocar funciones definidas por el usuario.

**return 0;** Esta es otra instrucción, en este caso la instrucción return determina que es lo que se devolverá de la función main(). Habíamos declarado que main devolvería un entero, así que la instrucción return devuelve 0. Lo cual a su vez significa que no han ocurrido errores durante su ejecución.

**}:** La llave de cierre de la función main() indica la finalización del bloque de instrucciones. En algunos programas, se notará el uso de dobles plecas (“//”). Estas plecas se usan para escribir comentarios de una línea dentro del código del programa. Además, podrá encontrar el uso de “/\*” “\*/” estos caracteres encierran un comentario de varias líneas y cualquier cosa que se escriba dentro de ella no influenciará en el desempeño del programa (estos conceptos fueron definidos previamente).

**¿Qué son las librerías en C++? Mencione 4 ejemplos.**

En C++, se conoce como librerías (o bibliotecas) a cierto tipo de archivos que podemos importar o incluir en nuestro programa. Estos archivos contienen las especificaciones de diferentes funcionalidades ya construidas y utilizables que podremos agregar a nuestro programa, como por ejemplo leer del teclado o mostrar algo por pantalla entre muchas otras más.

Las librerías no son únicamente archivos externos creados por otros, también es posible crear nuestras propias librerías y utilizarlas en nuestros programas. Las librerías pueden tener varias extensiones diferentes, las más comunes son: .lib, .bpl, .a, .dll, .h y algunas más ya no tan comunes.

La declaración de librerías se debe hacer al principio de todo nuestro código, antes de la declaración de cualquier función o línea de código, debemos indicarle al compilador que librerías usar, para saber qué términos estarán correctos en la escritura de nuestro código y cuáles no. La sintaxis es la siguiente:

**#include <nombre de la librería>** o alternativamente **#include "nombre de la librería"**

Todas las librerías estándar de C++ contienen una declaración del espacio de nombre std, es decir que todas las librerías que hacen parte del estándar de C++ colocan entidades dentro de este espacio de nombre.

Por esta razón cuando declaramos el uso del espacio de nombre std por medio de "using namespace std;", podemos evitar estar escribiendo std::cout o std::cin, etc. en nuestro código.

Algunas de las librerías de uso más común de C++ y que forman parte de las librerías estándar de este lenguaje son las siguientes:

1. **iostream**: Parte del a STL que contiene los algoritmos estándar, es quizá la más usada e importante (aunque no indispensable).
2. **math:** Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas.
3. **string:** Parte de la STL relativa a contenedores tipo string; una generalización de las cadenas alfanuméricas para albergar cadenas de objetos.
4. **time:** Útil para obtener marcas de tiempo durante ejecución. Se usa con frecuencia para conocer el tiempo exacto durante un programa.

Entre otras más.

**¿Cuáles son los tipos de datos que usan las variables en C++?**

**Tipos de datos simples en C++**

Los tipos de datos básicos normalmente son los numéricos y en la siguiente tabla podemos ver estos tipos de datos simples en C++, su nombre, descripción, el tamaño definido por el tipo de dato, que proporciona la cantidad de información que podemos guardar en una variable de ese tipo de dato, y el rango de valores que permite almacenar.

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

**Tipos de datos enumerados en C++**

Los tipos enumerados son un mecanismo usado en C++ que nos permite agrupar constantes simbólicas. Ejemplo:

enum dias { lunes, martes, miercoles, jueves, viernes, sabado, domingo };

**Tipos de datos complejos en C++**

* **Matrices / Array:** Los arreglos (array) son un tipo dato complejo, en el sentido de que nos permiten guardar conjuntos de datos, pero siempre datos del mismo tipo. Ejemplo:

int numeros[5] = { 1, 2, 3, 4, 5 };

* **Estructuras:** Las estructuras nos permiten guardar un conjunto de datos, pero esos datos pueden ser de distinto tipo. Ejemplo:

struct tiempo { int hora, minuto, segundo; };

**Salida y entrada de datos en C++ ¿Qué es cout y cin?**

La entrada y salida por consola (o I/O por consola) se refiere a la interacción entre un programa y el usuario a través de la línea de comandos. En C++, esto se logra utilizando flujos de entrada y salida proporcionados por la biblioteca estándar.

**Salida de datos:** std::cout es el objeto de flujo de salida estándar en C++. Se utiliza para mostrar datos en la consola y se usa comúnmente para enviar información al usuario. Para enviar datos a std::cout, se utiliza el operador de inserción (<<).Ejemplo:

**#include <iostream>**

**int** main**() {**

std**::**cout **<<** **"Hola, Mundo!"** **<<** std**::**endl**;**

**return** 0**;**

**}**

**Entrada de datos:** std::cin es el objeto de flujo de entrada estándar en C++. Permite leer datos desde la entrada estándar y se usa comúnmente para leer valores introducidos por el usuario a través del teclado. Para leer datos de std::cin, se utiliza el operador de extracción (>>). Ejemplo:

**#include <iostream>**

**int** main**() {**

**int** number**;**

std**::**cout **<<** **"Introduce un número entero: ";**

std**::**cin **>>** number**;**

std**::**cout **<<** **"El número introducido es: "** **<<** number **<<** std**::**endl**;**

**return** **0;**

**}**

**¿Cuáles son los operadores lógicos y aritméticos en C++?**

**Operador de asignación:** Establece un valor a una variable. Este operador es el =. Ejemplo: x = 3;

**Operadores aritméticos:** Operan sobre números y devuelven un nuevo número.

|  |  |  |
| --- | --- | --- |
| **Operador C++** | **Significado** | **Ejemplo** |
| + | Suma | 6 + 4 |
| - | Resta | 10 – 6 |
| \* | Multiplicación | 10 \* 3 |
| / | División | 20 / 10 |
| % | Módulo (división con residuo) | 21 % 2 |

**Operadores lógicos:** Se utilizan para hacer comparaciones proposicionales de dos expresiones.

|  |  |  |
| --- | --- | --- |
| **Operador C++** | **Significado** | **Ejemplo** |
| ! | Negación “No” | ! true |
| && | Conjunción “Y” | true && true |
| || | Disyunción “O” | true || false |

**Operadores de comparación:** Examinan la relación de dos elementos.

|  |  |  |
| --- | --- | --- |
| **Operador C++** | **Significado** | **Ejemplo** |
| == | Igualdad | a == b |
| != | Desigualdad | a != b |
| <= | Menor o igual que | a <= b |
| >= | Mayor o igual que | a >= b |
| < | Menor que | a < b |
| > | Mayor que | a > b |
| <=> | Comparación de tres bandas | a <=> b |

**¿Cuáles son las estructuras de selección en C++?**

**Estructura de control selectiva simple (if):** Evaluará una expresión lógica, y en caso de que sea verdadero se realizará las instrucciones dentro de su bloque de instrucciones. Sintaxis:

**if** **(** expresión lógica **){**

// Verdadero

**}**

**Estructura de control selectiva doble (if else):** Evaluará una expresión lógica y en caso de que sea verdadera se realizará las instrucciones dentro del primer bloque, y en caso de que sea falso se realizarán las instrucciones dentro del segundo bloque. Sintaxis:

**if (** expresión lógica **){**

// Verdadero

**} else {**

// Falso

**}**

**Estructura de control selectiva múltiple 1 (if else if else):** Es posible anidar las estructuras simples y dobles, es decir, se puede poner un if junto con un else. Sintaxis:

**if (** expresión lógica 1 **){**

// Verdadero caso 1

**} else if (** expresión lógica 2 **){**

// Verdadero caso 2

**} else {**

// Falso

**}**

**Estructura de control selectiva múltiple 2 (switch case):** Esta estructura de control trabaja con alguna variable, y según su valor realizará “x” caso. Sintaxis:

**switch (** variable **) {**

**case 1:** // Acciones 1

**break;**

**case 2:** // Acciones 2

**break;**

**case 3:** // Acciones 3

**break;**

**…**

**…**

**case N:** // Acciones N

**break;**

**default:** // Acciones por defecto

**}**

* **variable:** La variable a evaluar.
* **case:** Caso, indica el valor que de ser igual que variable, se ejecutarán las acciones en ese caso. Las acciones empiezan en los dos puntos (:) y termina en la instrucción break.
* **break:** Romper, sirve para finalizar un bucle o la estructura de control envolvente (en este caso el switch), es decir, una vez que se entre a uno de los case solo podrá finalizar cuando encuentre al primer break. En caso de no poner la instrucción break; al finalizar el grupo de instrucciones de un case, se realizarán todos los grupos de instrucciones desde el case que entró hasta que encuentre un break o finalice todo el switch.
* **default:** En caso de que ninguno de los case anteriores se haya cumplido, se ejecutarán las acciones de default. No es necesario que el switch contenga el default.

**BIBLIOGRAFÍA**

* <https://www.programarya.com/Cursos/C++/Comentarios>
* <https://krys907.wordpress.com/estructura-del-c/>
* <https://www.programarya.com/Cursos/C++/Bibliotecas-o-Librerias>
* <https://openwebinars.net/blog/principales-tipos-de-datos-en-c/>
* <https://www.luisllamas.es/cpp-entrada-y-salida-consola/>
* <https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/c-operators/>
* https://fisiprogramacion.wordpress.com/2014/08/03/c-7-estructuras-de-control-selectivas-en-c/