**Universidad Tecnológica de Honduras**



**Asignatura:**

**Análisis Y Diseño de algoritmos**

**Catedrático:**

**Jair Josué Hernández**

**Tema:**

**C++**

**Tarea #1 Tercer Parcial**

**Alumno/ No. de Cuenta:**

**Luis José Ramos Sarmiento/ 202510050088**

**Santa Bárbara, 18 de marzo de 2025**

**Investigar comentarios y estructura de 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. Por ejemplo, es muy común usar las líneas de comentarios, para dar una breve explicación de como funciona cierta parte de un código, lo cual permite identificar todo con mayor rapidez. Existen dos tipos de comentarios en el lenguaje:

Comentarios de Una Sola Línea: Pueden ser colocados en cualquier parte y comienzan por un doble slash "//", al colocar el doble slash al comienzo de cualquier línea de código, todo lo que halla de ahí en adelante en esa misma línea será tomado como comentario, ten en cuenta que el doble slash solo convierte en comentario al texto que haya después de él y que pertenezca a su misma línea, las líneas de abajo de este, no se verán afectadas.

Comentarios Multi-línea: Van cerrados entre "/\*" y "\*/". Estos comentarios son similares a los anteriores, pero deben tener un comienzo y un fina, 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

**Que son los encabezados de c++ (librerías) y de 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.

Al poder incluir estas librerías con definiciones de diferentes funcionalidades podremos ahorrarnos gran cantidad de cosas, imaginemos por ejemplo que cada vez que necesitemos leer por teclado, debamos entonces crear una función que lo haga (algo realmente complejo), al poder contar con las librerías en C++, podremos hacer uso de una gran variedad de funciones que nos facilitarán la vida y aumentarán la modularidad de nuestros códigos.

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.

Ejemplo #1

En el siguiente ejemplo veremos el uso de la librería stdlib.h que posee una gran variedad de funcionalidades, para este ejemplo usaremos la función rand que nos permite generar un número aleatorio.

#include <stdlib.h>

#include <iostream>

#include

using namespace std;

int main ()

{

// Se genera una semilla diferente cada vez (basada en el tiempo exacto de ejecución)

// Se debe llamar siempre, para generar números realmente diferentes cada vez

// La funciçon time viene de la libtrería time.h

srand (time(NULL));

cout << ("Se va a generar un numero aleatorio ....\n");

cout << ("El numero generado es : ");

cout << rand(); //Se genera el número con rand y se muestra en pantalla

return 0;

}

Ejemplo #2

En el siguiente ejemplo veremos el uso de la librería string.h que nos permite básicamente crear y manipular muy fácilmente cadenas de caracteres

#include <string.h>

#include <iostream>

using namespace std;

int main ()

{

cout << ("Hola! Por favor ingrese su nombre ....\n");

string cadena = "Hola "; //Se le da un valor inicial al string

string nombre; //Esta cadena contendrá el nombre

cin >> nombre; //Se lee el nombre

cadena = cadena + nombre; //Se juntan el saludo con el nombre usando "+"

cout << (cadena); //Se muestra el resultado final.

return 0;

}

Ejemplo #3

En este ejemplo usaremos la biblioteca vector, iterator y list, para crear una lista doblemente enlazada, agregar algunos elementos en ella (incluido un vector) y recorrerla.

#include <iostream>

#include <list>

#include <vector>

using namespace std;

int main ()

{

list<int> miLista; //Creamos

list<int>::iterator it;

for (int i=10; i<=50; i = i+10)

{

miLista.push\_back(i);

}

//Se agrega un 10 20 30 40 50 a la lista

it = miLista.begin(); //Se estabelce el iterador al comienzo de la lista

it++; //Movemos el iterador una posición (ahora estamos en 20)

//Se inserta un 11 en la posición del 10 11 20 30 40 50

miLista.insert (it,11);

miLista.insert (it,3,18);

//En la actual posición del 20 insertamos un 18 tres veces

//10 11 18 18 18 20 30 40 50

//Creamos un vector que poseerá dos elementos con valor de 19

vector<int> myvector (2,19);

//Insertamos ese vector en la lista, enviando los iteradores del propio vector

miLista.insert (it,myvector.begin(),myvector.end());

//10 11 18 18 18 19 19 20 30 40 50

cout << "Mi lista contiene lo siguiente:";

for (it=miLista.begin(); it!=miLista.end(); it++)

{

cout << ' ' << \*it; //Se muestra el valor al cual está aùntando el iterador

}

return 0;

}

Ejemplo #4

Este programa utilizará varias librerías para ver la funcionalidad multiple en C++

#include <iostream> // Para entrada y salida estándar (cin, cout)

#include <vector> // Para usar arrays dinámicos (std::vector)

#include <cmath> // Para funciones matemáticas (sqrt, pow, etc.)

#include <string> // Para manipulación de cadenas de texto (std::string)

int main() {

// Ejemplo con <iostream>

std::cout << "Hola, mundo!" << std::endl;

// Ejemplo con <vector>

std::vector<int> numeros = {1, 2, 3, 4, 5};

std::cout << "Elementos del vector: ";

for (int num : numeros) {

std::cout << num << " ";

}

std::cout << std::endl;

// Ejemplo con <cmath>

double numero = 16.0;

double raiz = std::sqrt(numero);

std::cout << "La raíz cuadrada de " << numero << " es " << raiz << std::endl;

// Ejemplo con <string>

std::string nombre = "C++";

std::cout << "Estamos aprendiendo " << nombre << std::endl;

return 0;

}

**Tipos de datos que se usan en c++**

Tipos básicos:

* Enteros: Almacenan números sin decimales.
* char: Un solo carácter (letra, número, símbolo). Ejemplo: char letra = 'A';.
* short: Enteros pequeños. Ejemplo: short edad = 10;.
* int: Enteros de tamaño medio. Ejemplo: int numero = 12345;.
* long: Enteros grandes. Ejemplo: long poblacion = 1000000;.

Punto flotante: Almacenan números con decimales.

* float: Precisión simple. Ejemplo: float precio = 3.14159;.
* double: Precisión doble. Ejemplo: double area = 6.283185307179586;.
* long double: Precisión extendida. Ejemplo: long double volumen = 1.234567890123456789;.
* bool: Almacena valores booleanos (verdadero o falso). Ejemplo: bool isTrue = true;.
* void: Indica la ausencia de valor. Ejemplo: void funcion();.

**Tipos avanzados:**

Punteros: Almacenan la dirección de memoria de otra variable. Ejemplo: int\* numeroPtr = &numero;.

Referencias: Alias de otra variable. Ejemplo: int& referenciaNumero = numero;.

Arrays: Colecciones de datos del mismo tipo. Ejemplo: int numeros[5] = {1, 2, 3, 4, 5};.

Estructuras: Agrupan datos de diferentes tipos. Ejemplo: struct Persona {string nombre; int edad;};.

Clases: Plantillas para crear objetos con propiedades y métodos. Ejemplo: class Persona {public: string nombre; int edad; void saludar();};.

**Entrada y salida de datos en c++ CIN y COUT**

**Cin**

**#include "iostream"**

**#include "string"**

**using namespace std;**

**int main()**

**{**

**cout << "Hola! Este es un ejemplo en C++" << "\n" << "Por favor ingrese su nombre:" << "\n";**

**//La instrucción \n es un salto de línea Mostrando los textos separados**

**string nombre;//En esta variable estará almacenado el nombre ingresado.**

**cin >> nombre; //Se lee el nombre**

**cout << "Bienvenido al sistema " << nombre << ". Gracias por usar nuestra aplicación" << "\n";**

**return 0;**

**}**

**Cout**

**#include "iostream"**

**#include "string"**

**using namespace std;**

**int main()**

**{**

**//El valor de esta variable se mostrará en pantalla**

**string salida1 = "Ejemplo de salida";**

**//Este valor también se mostrará en pantalla.**

**int numero = 2;**

**//Estos valores se concatenarán en una única salida**

**string salida2 = "Desde ProgramarYa.";**

**//Se concatenan y muestran los valores por pantalla con cout<<**

**cout << salida1 << " " << numero << ". " << salida2 << "\n";**

**return 0;**

**}**

**Operadores lógicos y aritméticos en c++**

**1. Operadores Aritméticos**

**Los operadores aritméticos se utilizan para realizar operaciones matemáticas básicas.**

| **Operador** | **Descripción** | **Ejemplo** |
| --- | --- | --- |
| **+** | **Suma** | **a + b** |
| **-** | **Resta** | **a - b** |
| **\*** | **Multiplicación** | **a \* b** |
| **/** | **División** | **a / b** |
| **%** | **Módulo (resto)** | **a % b** |
| **++** | **Incremento (suma 1)** | **a++ o ++a** |
| **--** | **Decremento (resta 1)** | **a-- o --a** |

**Los operadores lógicos se utilizan para combinar o negar expresiones booleanas (verdadero o falso). Son útiles en estructuras de control como if, while, etc.**

| **Operador** | **Descripción** | **Ejemplo** |
| --- | --- | --- |
| **&&** | **AND (y)** | **a && b** |
| **`** | **OR (o)** | **`** |
| **!** | **NOT (negación)** | **!a** |

**Investigar las estructuras de selección**

if: Ejecuta un bloque de código si la condición es verdadera.

else: Ejecuta un bloque de código si la condición del if es falsa.

else if: Permite evaluar múltiples condiciones.

switch: Permite seleccionar uno de muchos bloques de código para ejecutar.