

```
<!--Estudio Shonos-->
```

GUIA PARA NO  
MORIR CON C++ P1{

```
<Por="Marjorie Reyes"/>
```

}



# Contenidos

- 01 Estructura básica
- 02 Comentarios
- 03 Salidas
- 04 Variables
- 05 Tipos de Datos
- 06 Entradas
- 07 Operadores
- 08 Condicionales
- 09 Ciclos

La  
estructura  
por defecto  
de un  
programa en  
C++ {



ejemplo.cpp

```
#include <nombre_header>
using namespace nombre_namespace
```

```
int main(){
}
```

#Quote #Programming #Selfcare

}

## Puntos clave de la estructura {

#include

header

Un header es un archivo de C++ que contiene funciones y comandos.

using

namespace

Un namespace se utiliza para organizar el código en grupos lógicos y darles nombres.

int main

()

Se utiliza para indicar el punto de partida del programa.

{ }

Dentro de las llaves se coloca el código que será el cuerpo de la función.

}

La  
estructura  
por defecto  
de un  
programa en  
C++ {



ejemplo.cpp

```
#include <iostream>

using namespace std;
int main(){
    cout << "Hola mundo";
}
```

#Quote #Programming #Selfcare

}

# Comentarios

{

Se utilizan para explicar notas, sentencias o el funcionamiento del código.

Comentarios  
unilíneas //

Comentarios  
multilíneas /\* \*/



ejemplo.cpp

```
#include <iostream>

using namespace std;
int main(){
    cout << "Hola mundo";
}
```

#Quote #Programming #Selfcare

}

# Salidas en C++ {

las salidas se utilizan para verificar que la computadora está siguiendo el flujo esperado y arreglar problemas con el código.



ejemplo.cpp

```
cout << "mensaje a mostrar";  
cout << 22;
```

/\* Tomar en cuenta:

- Encerrar entre comillas el texto
- El punto y coma al final
- El signo es el de menor que \*/

#Quote #Programming #Selfcare

}

## Salidas en C++ {

Los saltos de línea en un cout se pueden realizar por medio de un `<< endl` o por medio de un `"\n"`.



ejemplo.cpp

```
cout << "This is the first line" << endl;  
cout << "This is the second line\n";  
cout << "This is the third line";
```

#Quote #Programming #Selfcare

}



## Case-sensitive {

C++ así como otros lenguajes de programación como java, python y C distinguen entre mayúsculas y minúsculas.



ejemplo.cpp

```
cout << "mensaje a mostrar";  
Cout << 22;
```

```
/* Cout no funcionará */
```

```
#Quote #Programming #Selfcare
```

}

# Variables {

Las variables se utilizan para almacenar valores que te serán útiles en el programa.

A una variable se le asigna un nombre para poder utilizarla luego.

Pueden cambiar de valor a lo largo del programa.



ejemplo.cpp

```
/* Declaración de variables */  
TipoDeDato NombreVariable;  
  
/* Asignación de Valores */  
NombreVariable = Valor;  
  
/* Declaración y asignación */  
TipoDeDato NombreVariables = Valor;
```

#Quote #Programming #Selfcare

}

# Variables {

- Declarar varias variables del mismo tipo en una sola línea.
- Los nombres de las variables deben iniciar con una letra o un guión bajo.
- Utilizar convenciones como Pascal o Camel case.



ejemplo.cpp

```
int x=8, y=2, z=10, a=78;
```

```
/* Válidos */
```

```
int _MyVar2;
```

```
int Var_3;
```

```
/* No válidos */
```

```
int 3var;
```

```
int var&iable;
```

```
/* Pascal case */
```

```
string NombreCompleto;
```

```
/* Camel case */
```

```
string nombreCompleto;
```

```
#Quote #Programming #Selfcare
```

}

# Tipos de Datos {

- Cadenas de caracteres
- Caracteres
- Número enteros
- Números decimales
- Booleanos
- Auto



ejemplo.cpp

```
int valor = 10;
double valor = 10.5;
string valor = "hola";
float valor = 10.5f;
char valor = 'A';
bool valor = true;
auto valor = 4585;

/* double es más preciso que
float, float debe llevar una f al
final del valor */
```

#Quote #Programming #Selfcare

}

## Entradas {

Se utilizan para que el usuario pueda ingresar un valor.

Para ello se utiliza el comando:  
`cin >>`



ejemplo.cpp

```
// Una sola entrada por línea  
int a, b;  
cin >> a;  
cin >> b;
```

```
// Varias entradas por línea  
int a, b;  
cin >> a >> b;
```

#Quote #Programming #Selfcare

}

# Operadores aritméticos {

- Suma
- Resta
- Multiplicación
- División
- Residuo



ejemplo.cpp

```
int points = 28;  
int level = 3;  
int result = points * level;  
int result = points / level;  
int result = points + level;  
int result = points - level;  
int result = points % level;
```

#Quote #Programming #Selfcare

}

# Operadores relacionales {

- Mayor
- Menor
- Mayor igual
- Menor igual
- Diferente
- Igual

Estos dan como  
resultado un  
booleano (verdadero  
o falso)



ejemplo.cpp

```
int points = 28;  
int level = 3;  
int result = points > level;  
int result = points < level;  
int result = points >= level;  
int result = points <= level;  
int result = points != level;  
int result = points == level;
```

#Quote #Programming #Selfcare

}

# Operadores lógicos {

- And
- Or
- Not

Estos dan como resultado un booleano (verdadero o falso)



ejemplo.cpp

```
int temp = 5;
```

```
bool value = temp >= 36 && temp <= 38
```

```
bool value = temp >= 36 || temp <= 38
```

```
bool value = !(temp >= 36 && temp <= 38)
```

```
#Quote #Programming #Selfcare
```

}



# Operadores increment y decrement {

Se utilizan para  
aumentar o  
disminuir un valor  
un número a la vez.



ejemplo.cpp

```
int num = 5;
```

```
num ++;
```

```
num --;
```

```
//Equivalen a
```

```
num = num + 1;
```

```
num = num - 1;
```

```
#Quote #Programming #Selfcare
```

```
}
```

Es posible  
acortar  
algunos  
operadores {



ejemplo.cpp

```
int num = 5;  
num += 3;  
num -= 3;
```

#Quote #Programming #Selfcare

}

# Condicionales

{

Se utilizan para tomar decisiones a lo largo del código. La sentencia se ejecuta solamente si cumple la condición.

- If
- Else
- Else if
- Switch



ejemplo.cpp

```
if(condition) {  
    //some code  
}  
else if(condition) {  
    //some other code  
}  
else {  
    //some other code  
}
```

#Quote #Programming #Selfcare

}

# Condicionales

## {

Se utilizan para tomar decisiones a lo largo del código. La sentencia se ejecuta solamente si cumple la condición.

- If
- Else
- Else if
- Switch



ejemplo.cpp

```
switch(eleccion) {  
  case valor:  
    //código  
    break;  
  case valor:  
    //código  
    break;  
  default:  
    //código  
}
```

```
// Break sale de la condición  
// Default si ninguna condición se  
cumple
```

#Quote #Programming #Selfcare

}

# Ciclos {

Se utilizan para repetir un bloque de código múltiples veces o para iterar.

- While
- Do While
- For

También pueden utilizar la sentencia break y continue.



ejemplo.cpp

```
while(condición){  
    //bloque de código  
}
```

```
do{  
    //bloque de código  
}while(condición);
```

#Quote #Programming #Selfcare

}

# Ciclos {

Se utilizan para repetir un bloque de código múltiples veces o para iterar.

- While
- Do While
- For

También pueden utilizar la sentencia break y continue.



ejemplo.cpp

```
for(int i=1;i<=5;i++) {  
    //bloque de código  
}
```

/\* Partes:

1. Inicializar (corre una vez)
2. Condición (se evalúa siempre)
3. Actualización (se ejecuta siempre)

#Quote #Programming #Selfcare

}

```
<!--Estudio Shonos-->
```

# Gracias {

En la siguiente clase veremos:

- Arreglos
- Punteros
- Memoria dinámica
- Funciones

}