

- C++ es un lenguaje de propósito general
- C++ es usado para la creación de programas y software de computadoras

C++ \Rightarrow Derivan \Rightarrow C

Ejemplo C++:

Hola mundo

- #include <iostream>
using namespace std;

int main()

{
cout << "Hola mundo".
return 0;
}

Carpete "C++ Des De Cero"
archivo "holamundo.cpp"

• C++ ofrece varios encabezados, como iostream.

• #include <iostream>

→ Encabezado usado para definir el Standard Stream Objeto Standard que tiene entrada y salida de datos.

• using namespace std;

→ Dice al compilador que use el Standard Stream.

• int main() { } una función.

↳ El punto de comienzo de todo Programa C++

→ Es Donde el Código Comienza a Ejecutarse

• cout << "Texto".

↳ Objeto de flujo utilizado para acceder a la Pantalla

→ Flujo de Salida de Datos

• return 0; → valor de resultado normal

• return 1; → valor de resultado Anormal o error

→ sirve para terminar la función

Herramientas

1. Entorno De Desarrollo Integrado (IDE)

- SublimeText3
- Code Block
- Dev-C++

2. Compilador

- Gnu C/C++

"Dev C++" "GCC"

BIN

- endl → manipulador de líneas

↳ Sirve para que haya una separación entre ~~dato~~ y texto.

Cout << "texto" << endl;

- \n → manipulador

↳ igual que el endl

Cout << "Texto\n";

- //

↳ come varios de una linea

// La computadora no lo lee

- /* ... */

↳ comentarios de mas de una linea

/* La compu
no lo lee */

Ojo con el terminal o consola

Variables en C++

- Variables: Es espacios de almacenamiento en memoria para guardar datos
- Int: (integer) representa un valor de tipo entero
 - Int IV → Si se pone una coma se pueden almacenar muchas variables.
 - ↳ Identificador de Variable

Int N.V, IV;

- Cin → Permite declarar una variable ya puesta previamente.
→ Cin >> IV;

Las variables se pueden declarar en la misma linea o mediante una segunda orden

Aritmetica Basica

Operacion	Simbolo	Resultado
Suma	+	X + Y
Resta	-	X - Y
Multiplicar	*	X * Y
Dividir	/	X / Y
Modulo	%	X % Y

Condicionales y Loops

- IF → Posibilidad de realidad

IF (condicion) {
} EJECUTAR

Operadores Relacionales

Operador Simbolo resultado

Suma integrante	++	++ V V++
Resta "	--	-- V V--
mayor igual que	>=	V >= V
menor igual que	<=	V <= V
igual que	=	V = V
Diferente que	!=	V != V

Ese en

C++

- Else → se usa cuando el if es falso

if (condicion) {

Ejecutar

X

Se Pueden Poner
Cuantas Sean
necessarios

} Else {

Ejecutar

✓

- Else if → se usa para multiples condiciones

if (condicion) {

Ejecutar

{

Else if (condicion) {

Ejecutar

{

Else {

Ejecutar

}

For-While Loop en C++

- while → Loop que se realize hasta que se satisfaga la condición
 - while (Condición) {
 Ejecutar
}
- for → Loop que se realize desde la condición inicial hasta lograr la condición final
 - ↳ se realizará x número de veces
 - for (init. → valor de inicio; Condición; incremento) {
 Ejecutar
}
- Do...while → Loop que realiza una acción hasta un punto
 - do {
 Ejecutar
} while (condición);

Switch en C++

- Switch → Prueba valores en distintas posibilidades (casos) para determinar si es igual a alguno

```
→ switch (Expresión) {  
    Case Value 1:  
        Ejecutar;  
    Valores Break;  
    Case Value 2:  
        Ejecutar;  
    Break;  
}
```

- Default: → En caso de que ninguno sea verdadero.

```
switch (Expresión) {  
    Case Value  
        Ejecutar;  
    Break;  
    Default:  
        Ejecutar;  
}
```

	(900) slide 10	
	++	

- Break → Finaliza la ejecución de un caso

→ switch (EXpression) {

Case Value:
Break }
Ejecutar

Operadores Logicos

Operadores	NOMBRE OPERADORES	FORMA
&&	And	x&&x
	or	x y
!	not	!x

- AND → El operador "y"

IZQUIERDO	DERECHO	resultado
True	false	false
false	true	false
TRUE	True	True
false	false	false

- OR → El operador "o"

IZQUIERDO	DERECHO	resultado
True	True	True
false	false	false
True	false	True
false	True	True

- NOT → El operador "no"

Sin sentido si tienen que ser
verdaderos si <operator>

Operador | Resultado

True

False

False

True

ANSI

Tipos de Datos, Arrays, Punteros en C++

- floating data → Son numeros flotantes (con coma, Periodicos, Etc).

→ float ~~IV~~

- Strings and Characters Data → Están compuestos Por frases o Letras.

"Frases" ↓
Letras

→ char ~~IV~~

→ wchar ~~IV~~

- Boolean Data → Son los valores de verdadero o falso.

→ True, false, zero, non-zero

→ False;

Int, float,
Double

- Int → Enteros

→ Int ~~IV~~

- Signed → Asignados → Valores Positivos y Negativos

- Unsigned → No asignados → Valores Positivos

- float → Flotantes → 4 Bytes

→ float ~~IV~~

- Double → 8 Bytes

→ Double ~~IV~~

- Long Double → 16 Bytes

→ Double ~~IV~~

Todos
son
flotantes

Son siempre asignados

String, Char, Bool

- Strings → Frases u oraciones.

→ String "Text".

hay que incluir la librería <String> para poder usar estas variables

- Char \rightarrow letters

→ Char 'Letra'

- Booleanos → Los Booleanos son variables de verdad o falsedad

$\rightarrow \text{Bool} \mid \text{IV} = \boxed{\text{True}; \text{false};}$

Registers Part number
Variables.

- Deben iniciar con una leva o un gabinete ()

- Pueden contener Letras o Números

Arrays en C++

- Se usa para declarar multiples variables en una sola variable

→ int Inv [...].

1

\rightarrow cantidad de valores almacenados

- Para Declarar la variable Array debemos de Tener los Valores entre llaves

$$\Rightarrow \text{Int } IV[\dots] = \{1, 2, 3, \dots\}.$$

Tipo de valor

- un array cuenta los numeros empezando del 0

$$\rightarrow \{1, 2, 3\}$$

[0][1] [2]

- Prezoceder el valor de un Arroz:

→ I ✓ [1]

18

Número de
Posición

Arrays multidimensionales en C++

- Contiene uno o mas arrays

Int IA[n][c]; Pueden ser mas
r = row
c = column
Pueden ser distintos

- Se debe ver de forma de tabla

	Columna 0	Columna 1	Columna 2	
Row 0	[0][0]	[0][1]	[0][2]	entre espacios son undato.
Row 1	[1][0]	[1][1]	[1][2]	
Row 2	[2][0]	[2][1]	[2][2]	

Punteros

- Ubicación de variable en memoria

→ & IV | Dara los numeros
↓ | que determinan
ubicación

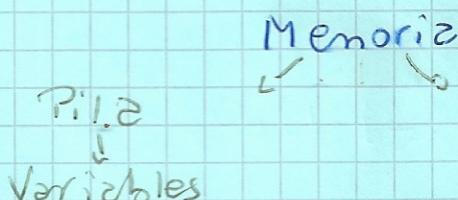
signo que Permite obtener la
ubicación de la variable.

- Un Puntero es una variable con la Dirección de otra
 - sirve para realizar tareas de forma mas simple
 - Todos los Punteros comparten el mismo tipo de datos: numero Hexa Decimal que expresa una Dirección de memoria
 - La única diferencia es el tipo de variable.
- Establecen Punteros usando el asterisco

→ Int → *IP
→ Double → *d?
→ float → *f?
→ Char → *C?

Deben ser definidas con la Palabra
Clave corresp Poniendo antes de
esto.

Memoria Dinámica y Estática en C++



Memoria

El Montón

↓
Programas y
código que se
ejecuta una
vez se inicia
el Programa

USANDO EL OPERADOR
NEW Podremos almacenar
variables en el montón
de código en la memoria
en tiempo de ejecución

→ New (Int)

- Delete → elimina la memoria dinámica cuando ya no se necesita de ella.
→ delete *IV;
- Punteros nulos → Tras eliminar la memoria dinámica el Puntero queda Y se le puede asignar otra memoria.

The sizeof() Operator

Categoría	Tipo	Espacio
Booleano	Bool	1 Byte
Carácter	Char	1 Byte
Integers	Int	2 Bytes
	Short	2 Bytes
	Long	4 Bytes
	Long Long	4 Bytes
Flores	Float	4 Bytes
	Double	8 Bytes
	Long Double	8 Bytes / 16 Bytes.

- El operador sizeof nos permite determinar el espacio en memoria que ocupa un tipo de dato.

→ sizeof(Tipo)

sizeof(13) = 2
sizeof(5) = 1
sizeof(19.9) = 4
sizeof(Shorint) = 2
sizeof(double) = 8

Funciones en C++

• Grupo de declaraciones que realizan actividades específicas.

- Ventajas:

- Reutilizar código.
- Probar funciones individuales.
- Ayuda a no alterar la estructura de un programa cuando se hace un cambio.

• Main → Las funciones principales tienen los tipos ~~return~~ void.

→ al retornar void podemos ver que el resultado se forma perfecto y sin errores.

• Void → Palabra clave para funciones que no retornan un valor.

Definir función

→ return_type function_name (parameter list)

{

Body

}

• return_type → tipo de dato que se requiere → int\char\...

• function_name → nombre de la función.

• parameter list → lista de parámetros que se aplican a una función.
→ Opcional

• Body → declaraciones de una función.

• Para llamar a la función usamos ~~main function~~ main function.

→ EN()

Power la función main
al final

Función Prototipo

→ #include <iostream>
#include <string>
using namespace std;

// Declaración

void PrintSomething(); → Prototipo

int main(){
PrintSomething();
return 0;
}

// Función

void PrintSomething(){
cout >> "Hola mundo";
}

Parámetros De Funciones

- Variables que Permiten Controlar las acciones de la función.

→ void f.a (int x) {

cout << x;

Funcionar
el Parámetro

- El valor se Puede determinar una vez la función es llamada

Múltiples Funciones
en C++

- Se ~~se~~ logra el Poner una variable Tras otras separadas Por coma.

Función Rand()

- Para usarla el Encabezado <cstdlib>

→ int main () {

cout << rand();

- Srand() → genera numeros completamente aleatorios a partir de un valor semilla.

Numeros Random

- Time() → función que da como resultado el valor actual de la hora.

Encabezado <ctime>

NAM VALORES

ARGUMENTOS

- Usados cuando no se especifica un valor para la función.

- Pueden ser usados para llamar ~~otras~~ función en distintas situaciones.

función Sobrecarga.

- Overloading → Permite crear diferentes funciones con el mismo nombre. (es efectivo ~~testear~~ con el Data Type Int.)

Recursion

- Función que se llama a sí misma. Usada para determinar un resultado factorial.

Arrays & función

- Se Pueden usar Arrays como argumentos.

referencias con Punteros

- Por Valor → Pre Definido → Este método copia el valor del Parámetro y lo modifica sin afectar los argumentos.
- Por referencia → Al Cambiar algun Parámetro el Argumento es afectado.

Clases y Objetos en C++

- POD → Programación Orientada a Objetos
- Objetos → formas con su propia definición y características
- Un Objeto Puede tener otros Objetos Pero siguen siendo Objetos Diferentes
- Los Objetos se ven en atributos que los definen
- NO Siempre Son Objetos físicos.

IDENTIDAD
Comportamiento
Atributos

Clases en C++

- Clases → Son el medio por el cual se crean Objetos
 - Describe al Objeto (no es el Objeto)
 - La misma Clase Para múltiples Objetos
- métodos → Otro nombre Para clases y Comportamientos
 - Basicamente es una función que se basa en pertenecer a una clase.

→ Class **IC** {
 Definición:
}

→ CLASS ICS

```
Tipo ← Public:  
metodo [ void IM() {  
    }  
};  
    Ejecutar
```

Set → Define
 SERIAL(...);
Get mDato → Obtener
 de los
 Get IA() { Devuelve }

- Para llamarla

```
→ int main() {  
    IC Objeto;  
    Objeto.IM();  
}
```

Verificar carpetas
C++ desde cero
Verificar archivo
Clases.cpp

ABSTRACCION

- La forma por la cual una que no sea específicamente un objeto con todos sus atributos igual podemos visualizar su forma.

CLASIFICACIONES

En encapsulamiento en C++

- El encapsulamiento consiste en restringir el acceso de la información al resto del código. NO corresponde a la clase o a los objetos creados a partir de esta.
- Nos sirve para evitar problemas e incompatibilidades. Por un cambio en algún atributo, flexibilizando el código y asegurando

Niveles de Protección:

- Public → accesible
- Protected → accesible hasta cierto punto
- Private → No es accesible

CONSTRUCTORES

- Crear el objeto una vez la clase esté definida y con sus atributos.

- Parámetros Predefinidos

Archivos en C++

- Archivos para clases o separados
 - Scope resolution operator

using namespace std;

(h) archivos separados

Destructores

Destructores

- SON LLAMADAS PARA ELIMINAR LOS OBJETOS.
 - SON LLAMADOS:
→ ~ NC() {
 CODIGO
}

Operador de selección.

→ → OJO! Para acceder a un objeto miembro con un Puntero

$\rightarrow \text{NC}^{\text{obj}}; \text{NC}^{\text{Pcr}} = \& \text{obj};$
 $\text{Pcr} \rightarrow \text{NM}();$

Cobsterie

- Valor fijo en una expresión
 - Const. \rightarrow Palabra clave que la distinguen

→ Conse_p, NV = 0;
Date type | resultado.

Los Objetos Constantes NO Puedes llamar
Funciones NO Constantes

- ### • Classes em Diferentes arquivos:

- Fuerce 1 en cabeza (ojo)
 - El archivo (.h) almacena declaraciones prototípicas

#ifndef NC_H
#define NC_H

class NC

5

DT) // bin 5000

3

#endif // NC_H

- La Clase se implementa en el archivo fuente (CPP)

→ `#include "NArchivo"` → Debe estar en la misma carpeta (.h)

Operador de resolución de ámbito

→ Es usado para definir el constructor

→ `myClass :: myClass ()`

Inicializadores miembros

→ Son usadas para ~~asignar~~ determinar valores a la lista miembro

Composition

- Los objetos complejos suelen construirse con objetos más simples.

- En C++ esológico se aplica al usar variables miembro como parte de otras clases.

Siempre Deben Ser DATOS NO CONSTANTES.

Keyword Friend

• Funciones Amigas → Las funciones Amigas son aquellas que al ser declaradas como funciones no miembro igualmente acceden a datos miembro Privados.

→ Se logra al declarar una función externa (amiga) dentro de la clase y precediéndola con la palabra clave friend

→ `class IC {`

Public:

`IC ()`

Código:

`};`

Private:

`int IV;`

`friend void IF (IC &obs);`

Continúa siguiente página

Operando el operador <<

```
void if (IC &obj) {  
    obj.IV = NO;  
    cout << obj.IV;  
}  
Llamado identificador variable.
```

main es friend.

```
int main() {  
    IVC obj;  
    FF(obj);  
}
```

Archivo Friend.

Keyword this

→ Los objetos tienen acceso a su propia dirección a través del Puntero This

→ En una función miembro this puede ser usado para referirse al objeto que la invoca

→ Class IC {

```
public:  
    IC (int IV): var(IV)  
};
```

Private:

```
    int var;
```

```
};
```

(→) → se usa para identificar la variable miembro con el Puntero this

→ cout << var <endl;
→ cout << this->var <endl; } Opciones para imprimir
→ cout << (*this).var <endl; } una variable.

→ se usa para sobre carga de operadores.

Sobre carga De Operadores en C++

→ Los operadores que pueden ser sobreescritos son:

+ - * / % ^ & | ~ ! <= > =

< > <= > == != << >> = = 88 ||

+ = - = /= %= ^= |= <= >= [] ()

-> ->* new new[] delete delete[]

[borrada]

→ Los operadores sobreescritos son funciones definidas con la
palabra clave operator seguida por el operador a sobreescalar

→ Class IC {

public:

int IV2;

IC () {}

IC (int IV) : IV2(IV) {}

IC operator + (IC & obj) {

operator

IC IV;

IV. IV2 = this -> IV2 + obj. IV2;

return IV;

}

}

int main() {

IC obj1(), obj2();

IC IV = obj1 + obj2.

operator sobreescrito

cout << IV. IV2;

}

Herencia en C++

→ La Herencia es un concepto que Permite definir una Clase Basandose en otras Clases.

→ Clase Base → Clase que hereda sus Parámetros.

→ Clase Derivada → Clase que recibe los Parámetros.

→ sintaxis:

Class ICD : Public ICB
↳ Identificador Clase Base
↳ Identificador Clase Derivada

{
Public:

ICD() {};

}.

→ Los Derivadas Heredan todo EXCEPTO:

- Constructores
- Destructores

- Operadores Sobrecargados
- Funciones Amigas.

Miembros Protegidos

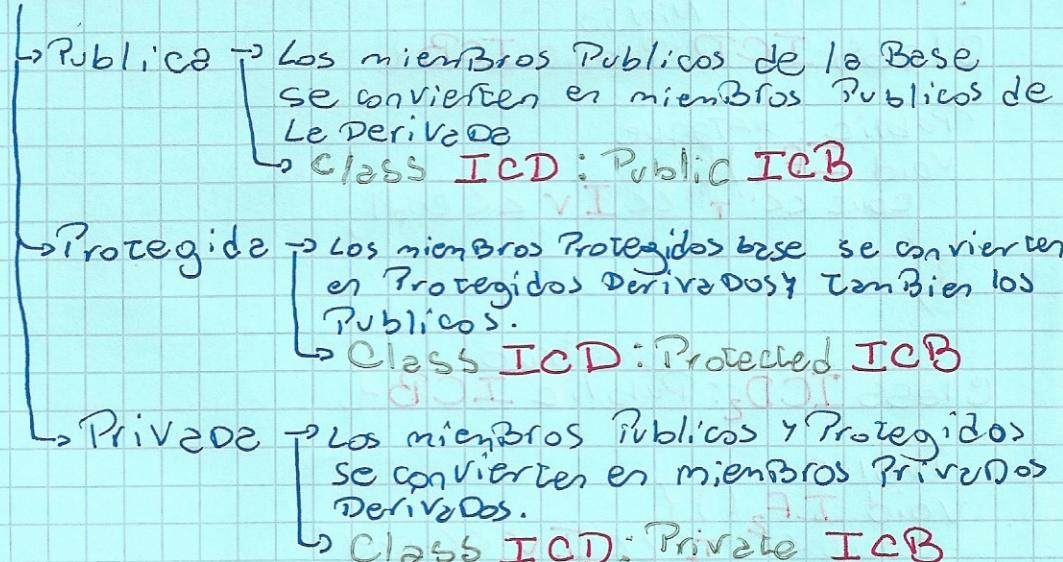
→ Especificadores de acceso

↳ Son los niveles de acceso de una clase → Public } ya
↳ Protected } explicado
↳ Private }

→ Protected → Son similares a los Private

↳ Pueden ser heredados

→ Tipos de herencia → se usan estos especificadores de acceso



Constructores y Destruidores De la Derivada

- Aunque no se hereden, igual pueden ser llamados para objetos derivados
- Los constructores y los destruidores son llamados antes de los derivados

→ B Ctor → Son llamados antes.

D Ctor

D Dtor, el más importante

B Dtor

QOT

COT

Polimorfismo

→ Significa → Tener muchas formas

→ Solo ocurre con una herencia entre clases

→ En C++ Significa que una misma función hará cosas diferentes dependiendo de el objeto que la invoca.

→ Clases ICB → Enemigo → Protector → Defensor → Ofensivo → Estrategia →

Protected: → Ataque

int IV;

Public:

void setIV(int IV);

IV = IV;

}

Código Completo
en Enemigos.CPP.

Class ICD: Public ICB {

Public: PATRÓN

void IF();

cout << "T" << IV << endl;

}

}

QOT

Class ICD₂: Public ICB {

Public:

void IF₂();

cout << "T₂" << IV << endl;

}

Virtual function

Se Pueden aplicar en la Derivada o en Base

- > Permitir la Funcion en las Derivadas De forma corresp Poniendo Debemos convertirlas a Virtual.

comienza
a activar
como se rompió?

virtual void Icf () {}

NECESARIO
PARA Polimorfismo



Funcion

CLASES ABSTRACTAS

- > Funciones miembro virtuales -> Funciones virtuales Puras



↳ No tienen Definicion

↳ Se Pueden Definir en las Derivadas

virtual void If () {}

↳ Así se define.

DEBEN SER DEFINIDAS
EN LA CLASE DERIVADA

- > ABSTRACTAS -> Son Clases Base que se usan Para DEFINIR Funciones virtuales Puras.

TEMPPLATES, EXCEPCIONES Y ARCHIVOS

- > ~~funcion~~ Funcion ~~template~~ -> Permite Crear Funciones capaces de manejar Cualquier Valor.

Plantilla

- > Parámetros de tipo ~~temp~~ -> se usa para definir estas funciones

- > Template -> Palabra clave. ↳ Identificador general

↳ template <class Ig>, < Ig, Ig >
//Funcion

Ig IF(Ig, Iv, Ig Iv) {

return Iv || Iv;

AL LLAMAR LA
FUNCION EN
EL MAIN DEFIDIIMOS
EL DATA TYPE

CLASES TEMPORALES

- > Se usa la misma sintaxis Pero en lugar de crear una Funcion creamos una Clase.

- > La ESPECIALIZACION de Plantilla Permite DECLARAR DIFERENTES maneras DEPENDIENDO DE SU DATATYPE.

- Para realizar esto debemos de comenzar la definición con el palabra clave con Parámetros nulos que se especificarán luego durante el Procedimiento.

EXCEPCIONES

- Es el resultado de un error en la ejecución del código.
- Throw → Palabra clave cuando buscamos mostrar un error
 - ↳ se puede definir la Posibilidad.
- Try → Palabra clave de error
 - ↳ Posibilidades a intentar
- Catch → Opciones que se toman en caso de que el Try no funcione

→ Try {

```
int IV = V;
int IN2 = V;
if (IV > IN2) {
```

```
    throw V;
}
```

```
catch (int X) {
```

cout << "Texto" << X;

- Cin → ingreso de datos de usuario.

↳ cin >> IV

Ejemplo en el archivo
VariablesDeDatos.cpp

ASISTAMAJUJA	EL MUNICIPIO DE
COMIBIQUERIA	EL ESTADO DE
ESTATUA	EL PAIS DE

VICENTE GUERRERO

SANTO DOMINGO

YUCATAN

Archivos EN C++

- Para leer y reescribir archivos en C++.
- Librería: ~~fstream~~ fstream

- TIPOS DE DATOS

↳ ofstream → Flujo de salida. crea y escribe archivos.
↳ ifstream → Flujo de entrada. lee archivos.
↳ fstream → Flujo genérico. capacidades de ofstream y ifstream.

- Para abrir un archivo

```
→ int main() {
```

fstream IO;
IO.open("Ruta del archivo");

Función abrir Predefinida
sin función aparte.

IO << ...
} ↓ "Texto"

el finalizar
IO.close();

- is_open() → Función que revisa si está abierto un archivo
↳ Parámetros según Darios que se Pueden Dar

Parámetro	Significado
ios::app	Añadir al final del archivo
ios::ate	Ir al final del archivo
ios::binary	Abre el archivo en Binario
ios::in	Abrir en modo lectura
ios::out	Abrir en modo escritura
ios::trunc	Borra los datos del archivo