

Manzo Jimenez Joceline

D M A
13 02 2025

Scribble®

GSAPFRXM

http://discord.gg/GSAPFRXM

Nombre, la materia.

• Metodos

Objeto = humano

atributos

altura float

Peso float

Raza string

genero string

edad int

tipo de sangre string

nacionalidad string

idioma string

estado civil string

clase social string

religión string

cultura string

Fuerza int

Resistencia int

agilidad booleano

punteria int

color de ojos string

color de cabello string

lentes booleano

complexión string.

discapacidad booleano.

• * Numerico

• * texto

• * booleano

double
long
int
float

chat
string

True 1
False 0

string: hereda
de varios.

Metodos: { Constructor
get
set

mover
comer
dormir
reproducir
atacar
defender.

tomar
regenerar

mover
aprender
conducir.

(velocidad)

nadar.

escuchar

ver

morir.

- Porque aprender programación orientada a objetos?

- Vas a programar más rápido. Tener una análisis previo de lo que estás realizando te ayudará a generar código mucho más veloz.
- ¿Qué es encapsulamiento?, ¿Qué es abstracción?, ¿Qué es herencia?, ¿Qué es polimorfismo?

Sinior -

Junior - Pasa a sinior cuando enseña a alguien

```
"MyClass"    object
- MyClass MyObj;
- int myNum;
string myString;

- myObj.myNum = 15;
myObj.myString = "some text";

cout << myObj.MyNum << "\n";
cout << myObj.myString;
return 0;
```

} regresa un entero (return cero por el int (que es un entero))

* Funciones

* Variables globales (se o modifica)

* Variables locales (solo puede ser modificado dentro del código)

```
void = vacio. myFunction() {
    cout << "I just got executed";
}
```

Justo ~~mañana~~ Definir antes la clase.

My Class = MC ^{como se escribe}

Herencia
abstracción
Polimorfismo
encapsulamiento.

Herencia: permite crear nuevas clases a partir de clases existentes, reutilizando atributos y métodos. La herencia facilita la organización jerárquica de clases y promueve la reutilización de código.

Polimorfismo: se refiere a la capacidad de un objeto de tomar múltiples formas. Esto permite que diferentes clases implementen métodos con el mismo nombre pero con comportamientos diferentes.

Abstracción: es un principio que permite reducir la complejidad de los sistemas de software. Se centra en las características esenciales de un objeto, ocultando los detalles de implementación.

Encapsulamiento: es un mecanismo para reunir datos y métodos dentro de una estructura ocultando la implementación del objeto, es decir, impidiendo el acceso a los datos por cualquier medio que no sean los servicios propuestos.

SDLC es la abreviatura de software Development Lifecycle, que en español significa ciclo de vida del desarrollo de software. Es un proceso estructurado que ayuda a planificar, crear, probar e implantar aplicaciones de software.

En SDLC permite a los desarrolladores crear y entregar aplicaciones de software a tiempo, dentro del presupuesto y según las necesidades del cliente.

- Permite desarrollar software de alta calidad, a bajo costo y dentro el menor tiempo posible.
- Fomenta una mejor colaboración en equipo.

Tipos de datos y su longitud en C++ (Para mañana)

Definición de constructor

- Destructor

camel
case

snake
case

Scribe®

Tipo de dato nombre-Variable = "Valor"

int

float

long

double

char

bool

nombre Clase {

// Atributos

// Metodos

}

Palabra reservada: Class no se puede usar como nombre de una variable

if

for

while

do

switch

else

cout

char

int

Definición clase constructor

class de una clase constructor
class Car { (solo en la clase)
public:

string brand; } Atributos

string model; } Publicos.

int year;

Car (string x, string y, int z) {

brand = x;

model = y;

year = z;

}

};

decimal
Entero.
float int main () {
Car car obj1 ("BMW",
X5, 1999);
} return 0;

En C++, los tipos de datos tienen diferentes tamaños, dependiendo del tipo de dato y de la versión del compilador.

Tipos de datos y su tamaño

int: 4 bytes, es la opción predeterminada para los valores enteros

double: 8 bytes, es la opción predeterminada para valores de punto flotante

bool: 1 byte, representa valores que puede ser true o false

char: 1 byte, se utiliza para representar caracteres individuales

wchart_t: 2 bytes, representa valores de caracteres "anchos" que se pueden codificar en formato UNICODE.

long: 8 bytes, Representa un intervalo mucho mayor de valores enteros.

unsigned char: 1 byte, C++ no tiene un tipo byte integrado. Use unsigned char para representar un valor de byte.

Un constructor es un método llamado por el tiempo de ejecución cuando se crea una instancia de la clase o de la estructura. Una clase puede tener varios constructores que toman argumentos diferentes. Los constructores permiten asegurarse de que las instalaciones del tipo son válidas cuando se crean.

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

class Alumno std;
public:
    string nombre;
    string apellido Paterno;
    string apellido Materno;
    float calificación;
```

mostrar Cal () {
 return calificación;
}

Alumno (String a, String b, string c) {

nombre = a, Edad

Apellido Paterno = b; Fecha Nac.

Apellido Materno = C; matricula

```
void calificar(Float x) {  
    calificacion = x;  
    calcular edad();
```

float main() {
 float cul:

Alumno Obj("Jocelin", "Manzo", "Jimenez");

alumno obj1 , calificar (101);
cout << alumno obj1 .

Return \emptyset ; cout << mostrar Cal(C)

col = alumno obj 1. mostrar Cal ()

edad
int

Objeto → instancia de clase

class alumno {
=====;
class materia {
=====;
};

```
int main() {  
    //  
    return x;  
}
```

```
class Alumno {
```

```
public:
```

```
    string nombre;
```

```
    string apellidoP;
```

```
    string apellidoM;
```

```
    int calificación;
```

```
    int edad;
```

```
    string matrícula;
```

```
    string fechaNacimiento;
```

```
Alumno (String a, String b, String c, int d, int e, String
```

```
    f, String g) {
```

```
    nombre = a;
```

```
    apellidoP = b;
```

```
    apellidoM = c;
```

```
    calificación = D;
```

```
    edad = e;
```

```
    matrícula = F;
```

```
    FechaNacimiento = g;
```

```
}
```

```
void calificar (float x) {
```

```
    calificación = X;
```

```
}
```

```
void calcular Edad (int añoNacimiento) {
```

```
    int añoactual = 2025; // año actual
```

```
    int edad = añoactual - añoNacimiento;
```

```
    cout << "Tu edad es " << edad << "años" << endl;
```

```
}
```

```
int main () {  
    Alumno ("Joceline")  
    "Manzo", "Jimenez");  
    alumno calificación (8);  
}
```

```
int main () {  
    int añoNacimiento;  
    cout << "ingresa tu año de nacimiento:";  
    cin >> añoNacimiento;  
    calcular Edad (año Nacimiento)  
    return 0;
```

3

Métodos y campos.

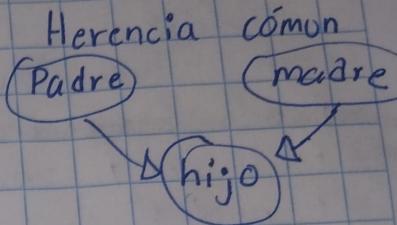
Alumno estudiante ("Juan", "Lopez", "Campos");

estudiante.nombre = "Jose";

Palabra Reservada

- * Público.
- * Privado.
- * protegido.

Herencia multiple



Q: ¿Qué es un destructor?

Se utiliza para liberar recursos y realizar tareas de limpieza antes de que el objeto sea eliminado.

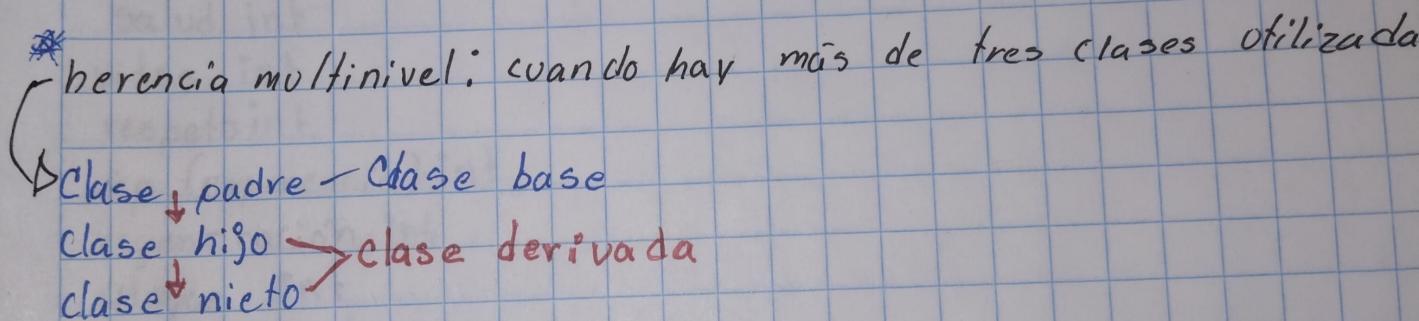
- 1 = Nombre
- 2 = Parámetros
- 3 = Cuerpo
- 4 = Retorno.

Utiliza

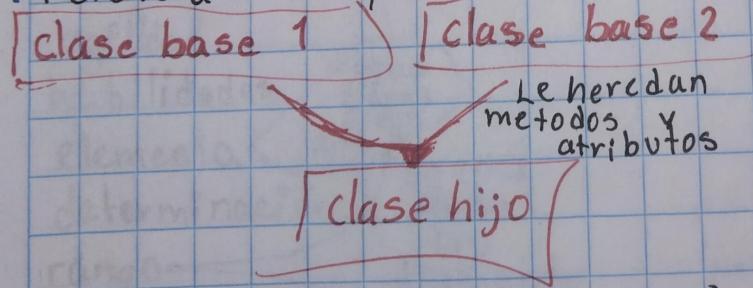
- Liberar memoria
- Crear recursos
- Realizar tareas de limpieza.

ciclo plus. = .cpp

- Elementos o miembros de un método público se puede acceder desde cualquier otra clase.
- Especificadores de acceso protegido.
* Clase derivada es la que hereda atributos o métodos de otra clase base = "padre".
Clase es útil para reutilización de código, o cuando creas una nueva clase.



* Herencia multiple



crear repositorio. (Visual estudio 0)

- Repositorio debe ser privado (como subir la farca)
- * git clone (descargar git bash)

personaPrincipal

Fuerza int

velocidad

close habilidad Mental

Inteligencia

Guerrero

arquero

mago

carisma

suerte

vision

altura float

peso float

resistencia int

salud int

vidas int

respecto int

xp (experiencia).int

regeneracion.int

ataque int

defensa int

magia. int

agilidad.

habilidades

agua
tierra
fuego
aire

elemento.

determinacion.

rango

debilidades.

coraje

Bronce

plata

oro

diamante

Metodos

atrapar()

defender()

comer()

descansar()

dormir()

curarse()

revivir()

farmear()

entrenar()

morir()

```
#include <estudio.h> #include <time.h>
#include <stdlib.h>
```

```
class Luchador {
```

```
public:
```

```
int ataque;
```

```
int vida;
```

```
int defensa;
```

```
Luchador (int x, int y) {
```

```
vida = 100;
```

```
ataque = x;
```

```
defensa = y;
```

```
}
```

```
void setVida (int s) {
```

```
vida = getVida() - s;
```

```
}
```

```
int getVida () {
```

```
return vida;
```

```
}
```

```
};
```

```
void main( ) {
```

```
Luchador santo (4, 2);
```

```
Luchador blueDemon (3, 3);
```

```
int descuento = 0;
```

Santo	A	D	
4	2	100	
3	3	100	

```
rand ( ) % 10 + 1;
```

```
for (int i = 1, i < 10, i++) {
```

```
i = 1, i < 0, i++ ) {
```

```
descuento = (A - D) + RANDOM
```

```
rand ( ) % 10 + 1;
```

```
descuento = (santo.ataque - blueDemon.defensa) + rand ( ) % 10 + 1;
```

```
blueDemon.setVida (descuento);
```

```

descuento = (blueDemon.ataque - santo.defensa) + (rand () % 10 / 10);
santo.setVida (descuento);
if (santo.getVida () <= 0) {
    cout << "gana blue Demon";
    break;
} else if (blueDemon.getVida () <= 0) {
    cout << "gana santo";
    break;
}

```

- OOP = que significa programación orientada a objetos.
- 4 características de la programación orientada a objetos
- Abstracción
- Herencia
- Encapsulamiento
- Polimorfismo
- Qué es una palabra reservada
 - ↳ Sintaxis de una clase
NombreDeLaClase NombreDelObjeto([Parametro1, P2...Pn]);
pokemon pikachu();
 - Palabra reservada para romper un ciclo: for
(Cuantas formas hay para romper un ciclo?
Break, return, continue, excepciones, Flags, goto.)
 - La palabra "continue" para que se utilice para la integración de un bucle o for para saltarlo en la interación actual, bucle y pasa a la siguiente
- Que es el polimorfismo = se refiere a la capacidad de un objeto para formar diferentes formas.
- Que es abstracción, un ejemplo = es cuando clasificas un metodo de por ejemplo: de un animal si es herbívoro o carnívoro
- un while y un do ... while = diferencia

```
int main () {
```

// Producto agregado

```
Producto Sabritas ("sabritas", 15.50, 50);  

Producto coca ("coca", 10.00, 30);  

Producto pan ("pan", 7.50, 20);
```

Venta tienda

```
tienda.agregar Producto (sabritas);  

tienda.agregar Producto (coca);  

tienda.agregar Producto (pan);
```

```
class producto {  

public:  

    String Nombre;  

    float precio;  

    producto (String nom, float pre) {  

        nombre = nom;  

        precio = pre;  

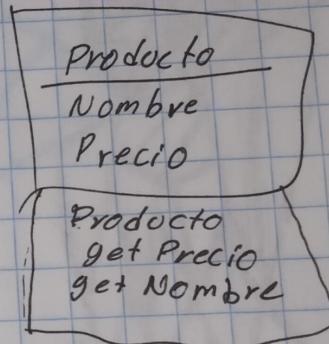
    }  

    float get precio () {  

        return precio;  

    }  

}
```



```
class venta {  

public:  

    float total = 0;  

};  

calcularTotal () {  

    return total  

}
```

```
String get Nombre () {  

    return nombre;  

}  

}
```

```
int main () {  

    float producto p1 ("pizza", 220);  

    float producto p2 ("chicken", 70);  

    float producto p3 ("refresco", 30);  

    venta v999;  

    v999.total = v999.total + p1.precio + p2.precio  

    + p3.precio;  

    cout << calcularTotal ();  

    return v999.total;  

}
```

Hard code

[run]