Diseño y Análisis de Algoritmos

MSc. Enrique Paiva - 2023

Biografía

Técnico Electromecánico

CTN-2010

Ingeniero Mecatrónico

FIUNA-2018

Máster en Electrónica de Potencia

UCSA-2021

Publicaciones en conferencia

9 papers

Publicaciones en revista

1 paper

Cooperación con el LSPyC

2016

Profesor en FIUNA

SCA2 (2018-2021)
IA (2019-actual)

Jefatura de Electrónica - ANDE

DPPM-2020

Estructura del Curso

Tabla de Contenidos

Lenguajes de Programación

Un **lenguaje de programación** es un conjunto de reglas y símbolos que se utilizan para escribir programas de ordenador. Es un lenguaje formal diseñado para comunicar instrucciones a una máquina, con el objetivo de realizar una tarea.

Lenguajes de Bajo Nivel:

- Lenguaje Ensamblador (Assembler)
- Lenguaje de Máquina

Lenguajes de Alto Nivel:

- Fortran
- C/C++
- Java
- Python
- MATLAB

...

C/C++

Lenguaje de programación C/C++

1. Eficiencia de recursos

C/C++ es un lenguaje de programación de nivel intermedio, lo que significa que proporciona un control directo sobre el hardware de la computadora, lo que permite un uso más eficiente de los recursos del sistema, especialmente en entornos con recursos limitados, como los microcontroladores.

2. Velocidad de ejecución

La eficiencia de recursos también se traduce en una mayor velocidad de ejecución de los programas escritos en C/C++, lo que es crucial en el caso de los microcontroladores, donde los programas deben ejecutarse en tiempo real.

3. Acceso directo a la memoria y los registros

C/C++ permite un acceso directo a la memoria y los registros del microcontrolador, lo que proporciona un mayor control y una mayor capacidad de optimización del código.

Lenguaje de programación C/C++

4. Portabilidad

C/C++ es un lenguaje de programación portable, lo que significa que el mismo código fuente puede ser compilado en diferentes plataformas, lo que permite una mayor flexibilidad en la elección de los microcontroladores.

5. Amplia comunidad de desarrolladores

C/C++ es un lenguaje de programación muy popular con una gran comunidad de desarrolladores, lo que significa que hay una gran cantidad de recursos, herramientas y bibliotecas disponibles para los programadores de microcontroladores.

Fases en la programación en C/C++

1. Diseño

En esta fase se define el problema a resolver y se diseña la solución del mismo. Se pueden crear diagramas de flujo, pseudocódigos, diagramas UML, entre otros, para planificar la solución.

2. Codificación

En esta fase se escribe el código fuente del programa. El código fuente se escribe en un editor de texto o en un IDE (Entorno de Desarrollo Integrado) y se guarda en un archivo con extensión .cpp.

3. Compilación

En esta fase se traduce el código fuente en código objeto (archivo binario) mediante un compilador de C++. El compilador verifica que el código esté bien estructurado, que las variables estén declaradas correctamente y que no haya errores de sintaxis.

Fases en la programación en C/C++

3. Enlazado

(Linker) En esta fase se combinan los archivos objeto generados en la fase de compilación con los archivos de biblioteca para crear el archivo ejecutable. El enlazador también resuelve las referencias a funciones y variables externas.

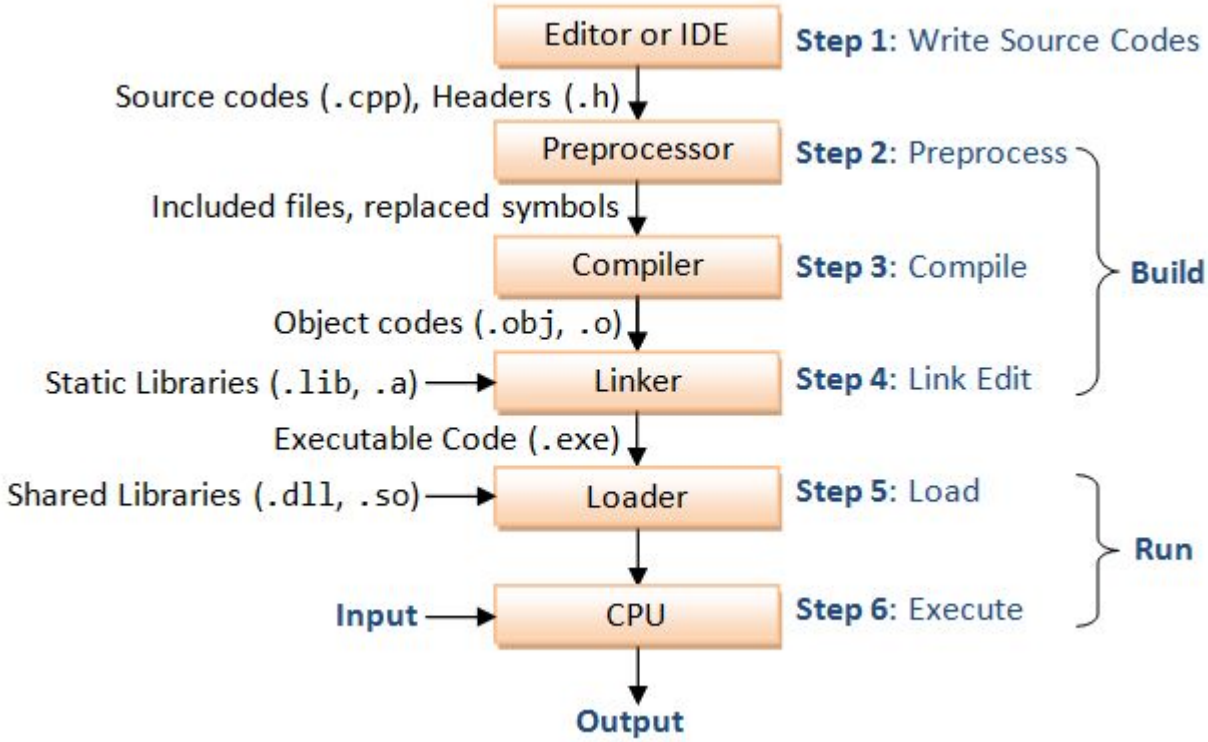
4. Depuración

En esta fase se busca y corrige cualquier error o fallo que el programa pueda presentar. Se utilizan herramientas de depuración como un depurador para identificar y corregir errores en el código fuente.

5. Ejecución

En esta fase se ejecuta el archivo ejecutable generado en la fase de enlazado. El programa realiza las operaciones especificadas en el código fuente y produce los resultados esperados.

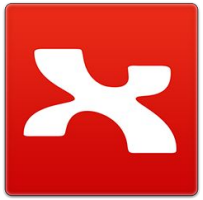
Fases en la programación en C/C++



https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp0_Introduction.html

Herramientas para el Diseño C/C++

StarUML



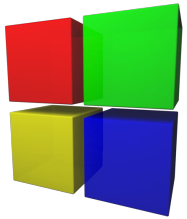
Xmind

Github

Overleaf

IDEs C/C++

Dev-C++



Code::Blocks

VSCode

Eclipse

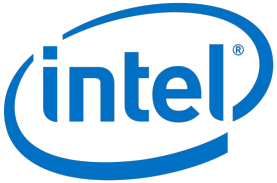
NetBeans

Compiladores en C++

GCC

Clang

Microsoft Visual C++

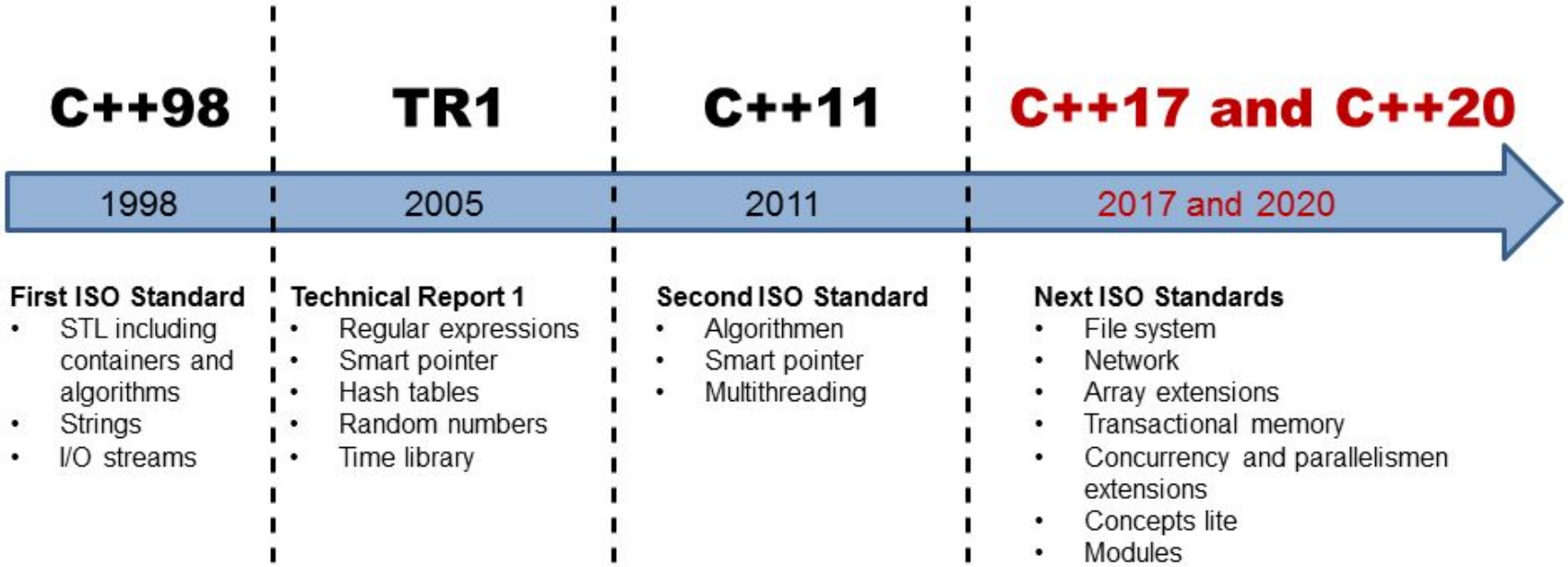


Intel C++ Compiler

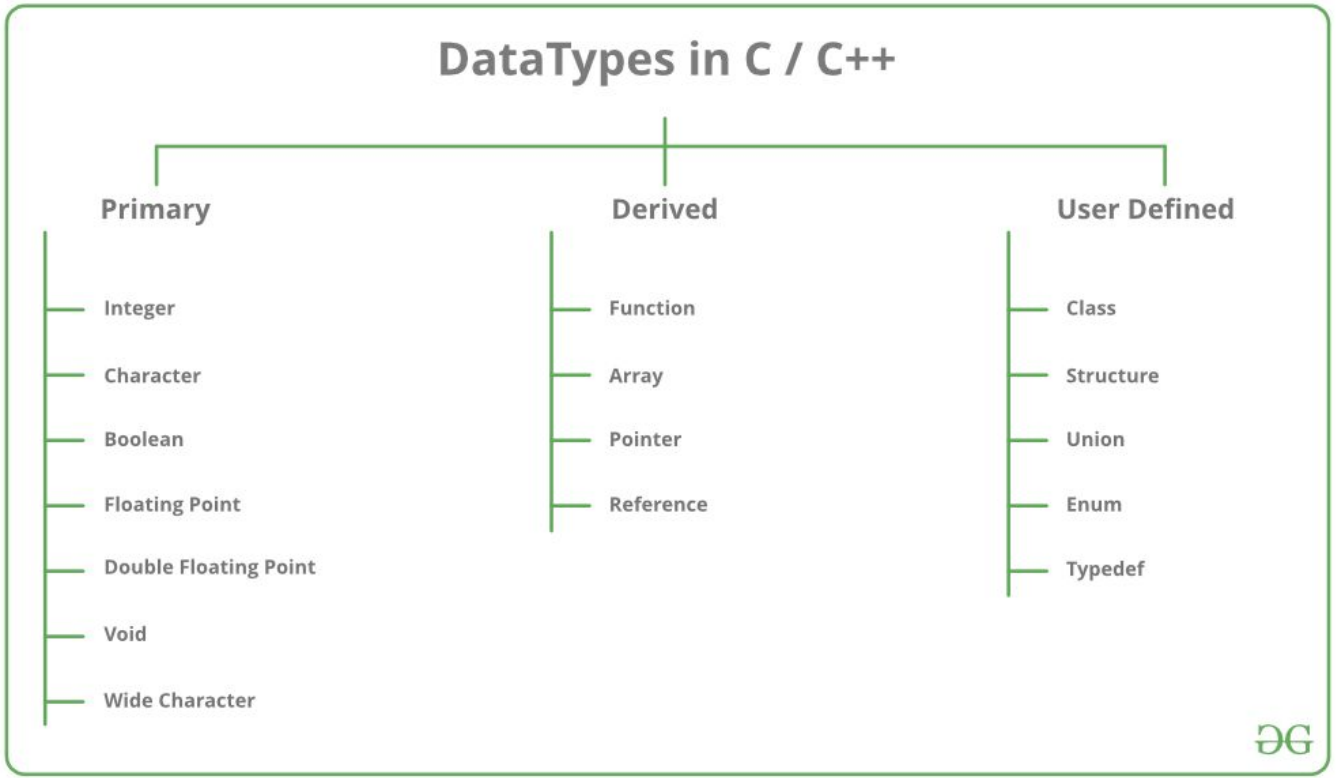
Linkers C++

Propiedades	Librerías Estáticas	Librerías Dinámicas
Tiempo de Enlazamiento	En el último paso del proceso de compilación	Durante el proceso de enlazamiento cuando el archivo ejecutable y librerías son agregadas en memoria.
Funcionamiento	Enlazamiento	Sistema Operativo
Memoria	Aumenta el tamaño del programa porque las funciones de las librerías estáticas sin construidas en el archivo ejecutable	Reducido, porque múltiples programas llaman a la librería, una sola instancia de la librería dinámica es necesaria en la memoria.
Actualizaciones	Si hay un cambio en la librería, el archivo ejecutable debe ser compilado.	Si hay cambios en la librería, no es necesario recompilar.
Tiempo	Los archivos ejecutables cargan mucha memoria en cada tiempo de ejecución, significando mas tiempo de ejecución.	Es rápido debido a que la librería ya está en la memoria.
Problemas de Compatibilidad	No es posible, debido a que todo el código se encuentra en un único archivo.	Depende de que tan compatible sea la librería. Si la librería es eliminada, el programa no funcionará.

Versiones C++


C++98
TR1
C++11
C++17 and C++20
1998
2005
2011
2017 and 2020
First ISO Standard
• STL including containers and algorithms
• Strings
• I/O streams
Technical Report 1
• Regular expressions
• Smart pointer
• Hash tables
• Random numbers
• Time library
Second ISO Standard
• Algorithmen
• Smart pointer
• Multithreading
Next ISO Standards
• File system
• Network
• Array extensions
• Transactional memory
• Concurrency and parallelismen extensions
• Concepts lite
• Modules

Tipos de Datos C++ code



Tipos de Datos C++ code

Name	Type	Range
std::int8_t	1 byte signed	-128 to 127
std::uint8_t	1 byte unsigned	0 to 255
std::int16_t	2 byte signed	-32,768 to 32,767
std::uint16_t	2 byte unsigned	0 to 65,535
std::int32_t	4 byte signed	-2,147,483,648 to 2,147,483,647
std::uint32_t	4 byte unsigned	0 to 4,294,967,295
std::int64_t	8 byte signed	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
std::uint64_t	8 byte unsigned	0 to 18,446,744,073,709,551,615

Tipos de Datos C++

Floating point type	Number of bits	Min value	Max value	Closest value to 0
float	32 bits	-3.4 E+38	+3.4 E+38	± 1.17549 E-38
double	64 bits	-1.79769 E+308	+1.79769 E+308	±2.22507 E-308
long double	80 bits	-1.18 E+4932	+1.18 E+4932	±3.36 E-4932

Tipos de Datos C++ code

```
string s = "I am sorry, Dave.";
           indices 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

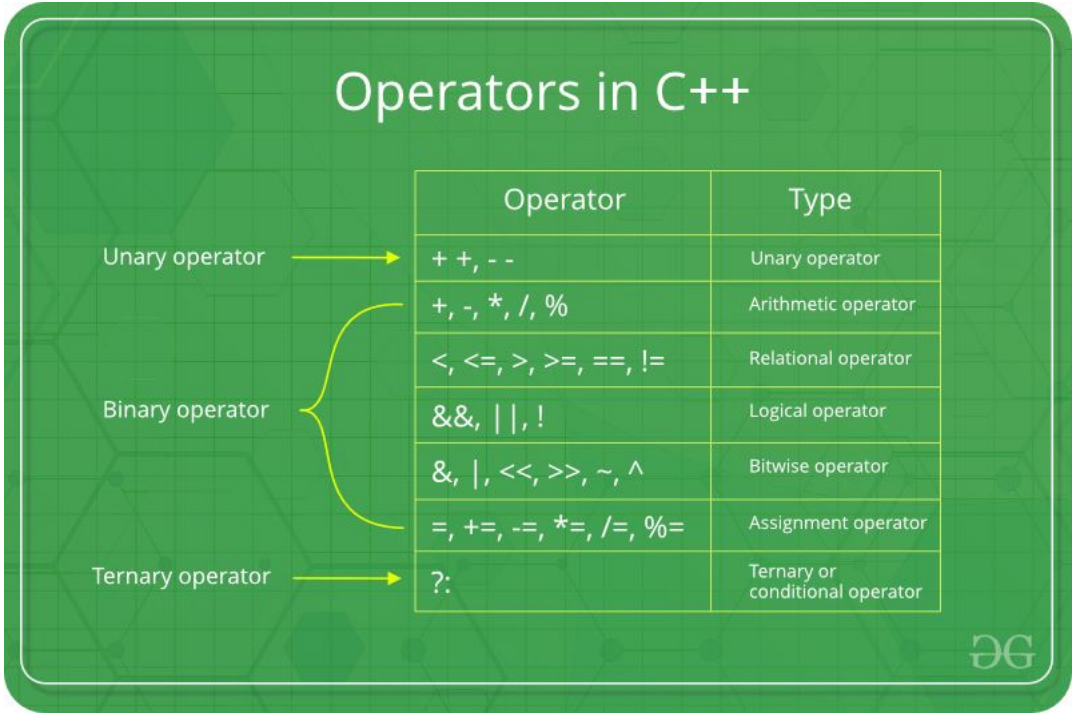
```
s.insert(5, "very ")      ⇒  s = "I am very sorry, Dave."
s.erase(6, 2)             ⇒  s = "I am sry, Dave."
s.replace(12,5,"Frank")   ⇒  s = "I am sorry, Frank"
s.resize(4)               ⇒  s = "I am"
s.resize(20, '?')         ⇒  s = "I am sorry, Dave.???";
```

changes string object

```
s.find("r")     →  7        (first occurrence from begin)
s.rfind("r")    →  8        (first occurrence from end  )
s.find("X")     →  string::npos              (not found)
s.find('a',5)   →  13       (first occurr. starting at 5)
s.substr(5,6)   →  "sorry,"  (returns new string object)
s.contains("sorry") → true                       (C++23)
s.ends_with("ave.") → true                       (C++20)
s.starts_with('I')  → true                       (C++20)
```

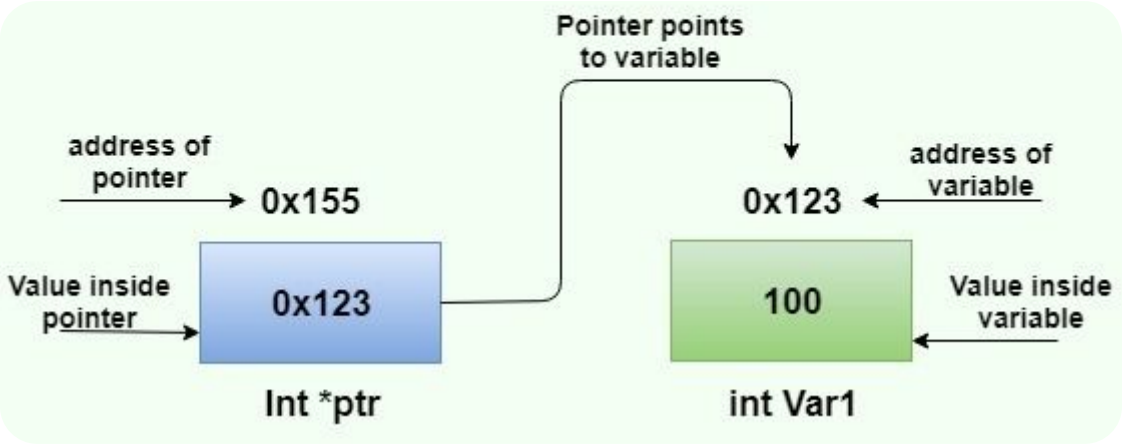
no change to string object

Operadores C++ code



Punteros C++

Un puntero guarda en una variable la dirección de otras variables en memoria. De tal forma que podemos manipular datos guardados mediante su posición en memoria.



Punteros C++ code

Operador Dirección de: **& variable_name;** Pudiendo ser esta variable de cualquier tipo.
Operador Dereferenciar: **data_type * pointer_name;** Pudiendo

Tipos de punteros:

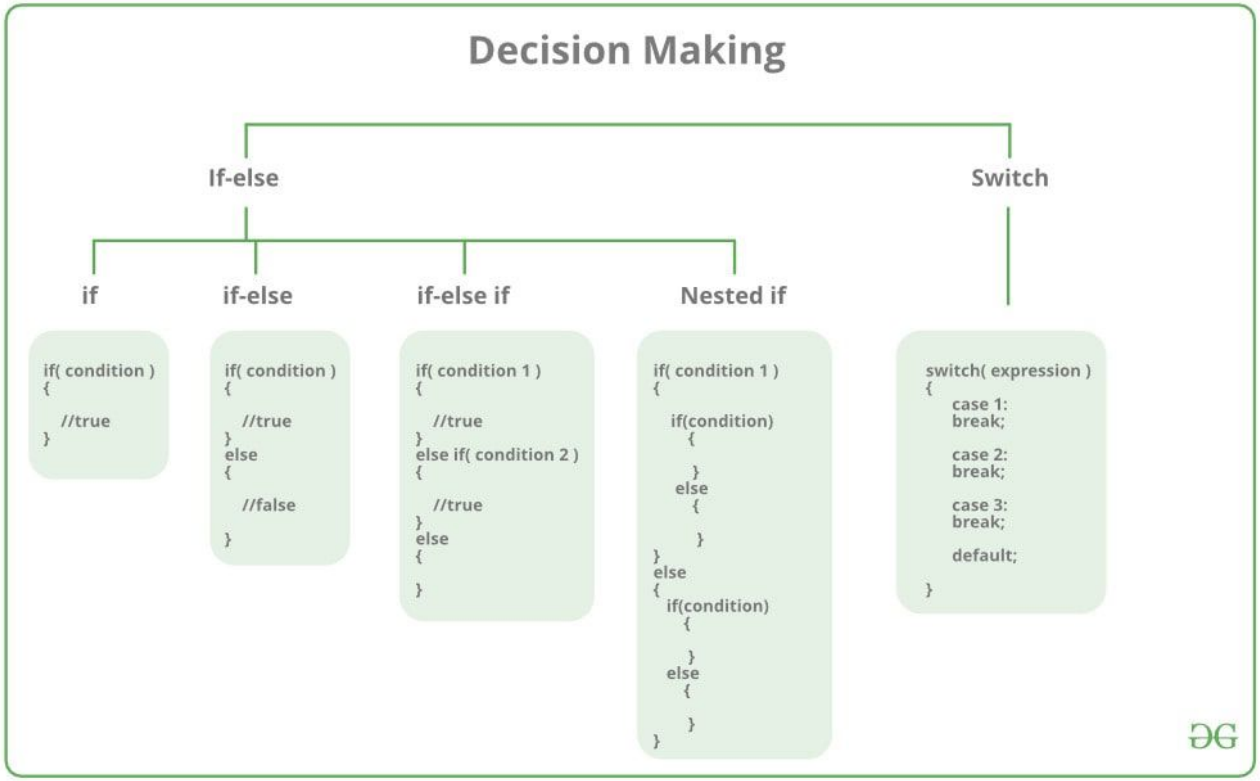
- Punteros a enteros
- Punteros de Arreglos
- Puntero a Estructuras
- Puntero a funciones
- Punteros dobles
- Punteros nulos
- Punteros Void
- Punteros No Inicializados
- Puntero a Constantes

Tamaño de los punteros:

- 8 bytes for a 64-bit System
- 4 bytes for a 32-bit System

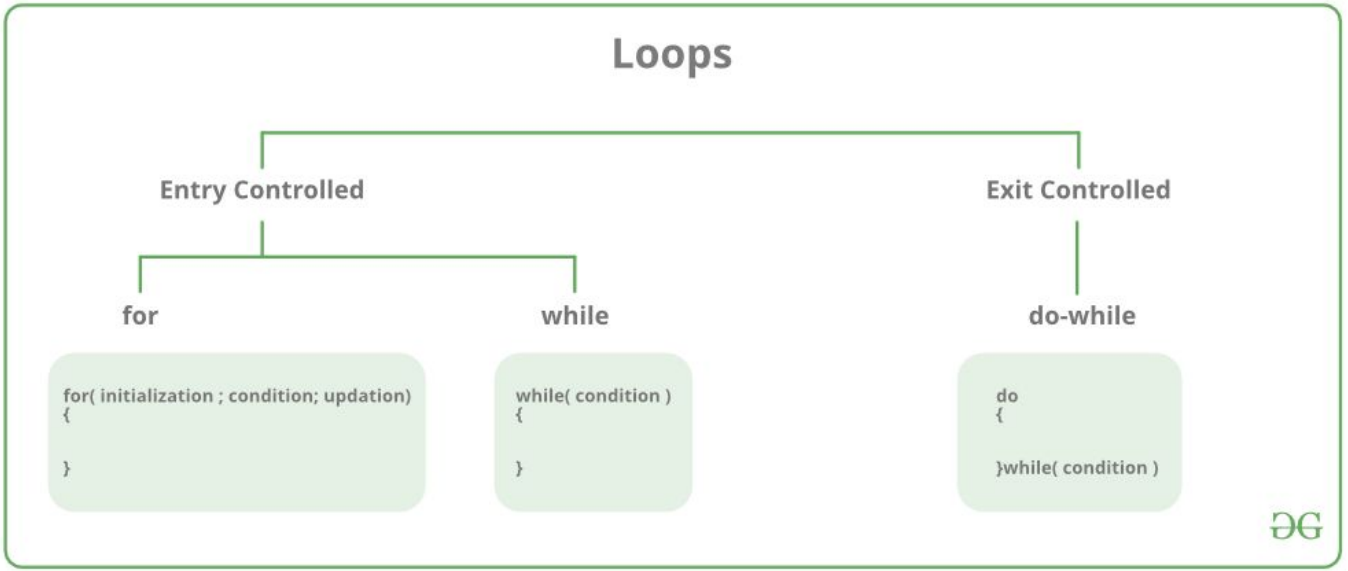
Estructuras de Control de flujo C++

code



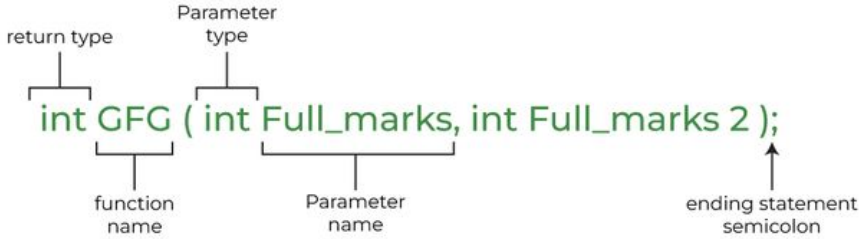
Loops C++

code


Loops
Entry Controlled
Exit Controlled
for
while
do-while
for(initialization ; condition; updation)
{
}
while(condition)
{
}
do
{
}while(condition)

Funciones C++

Una función es un conjunto de sentencias que toman entradas, realizan algún cálculo específico y producen una salida.



- Las funciones nos ayudan a reducir la redundancia de código. Si la funcionalidad se realiza en varios lugares en el software, entonces en lugar de escribir el mismo código, una y otra vez, creamos una función y la llamamos en todas partes. Esto también ayuda en el mantenimiento, ya que tenemos que cambiar en un solo lugar si hacemos futuros cambios en la funcionalidad.
- Las funciones hacen que el código sea modular. Considere un archivo grande con muchas líneas de código. Se hace muy fácil de leer y utilizar el código si el código se divide en funciones.
- Las funciones proporcionan abstracción. Por ejemplo, podemos utilizar funciones de biblioteca sin preocuparnos de su trabajo interno.

Funciones C++ code

Formas más populares de pasar parámetros:

- Paso por valor: En este método de paso de parámetros, los valores de los parámetros reales se copian a los parámetros formales de la función y los dos tipos de parámetros se almacenan en diferentes posiciones de memoria. Por lo tanto, cualquier cambio realizado dentro de las funciones no se refleja en los parámetros reales de la persona que llama.

- Paso por referencia: Tanto los parámetros reales como los formales hacen referencia a las mismas ubicaciones, por lo que cualquier cambio realizado dentro de la función se refleja en los parámetros reales de la persona que realiza la llamada.

Ejercicio 1.

La conjetura de Collatz

Las siguientes secuencias iterativas son definidas por el conjunto de números enteros positivos:

$n \rightarrow n/2$ (n si es par)
$n \rightarrow 3n + 1$ (n si es impar)

Usando la regla de arriba y empezando por el número 13, podemos generar la siguiente secuencia:

$13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

Se puede observar que esta secuencia (empezando por el 13 y finalizando en 1) contiene 10 términos.

¿Qué número inicial, por debajo del mil, produce la cadena más larga?

Ejercicio 2.

Fibonacci

Cada término de una secuencia de Fibonacci es generado por los dos términos previos. Empezando con 1 y 2, los primeros 10 términos serán:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Considerando los términos de la secuencia Fibonacci aquellos valores que no excedan los dos millones, encontrar la suma de los valores impares.

Clases C++

La programación orientada a objetos (OOP) es un paradigma de programación que se basa en el concepto de objetos, que son instancias de una clase, y se enfoca en la organización del código en unidades de software reutilizables y modulares.

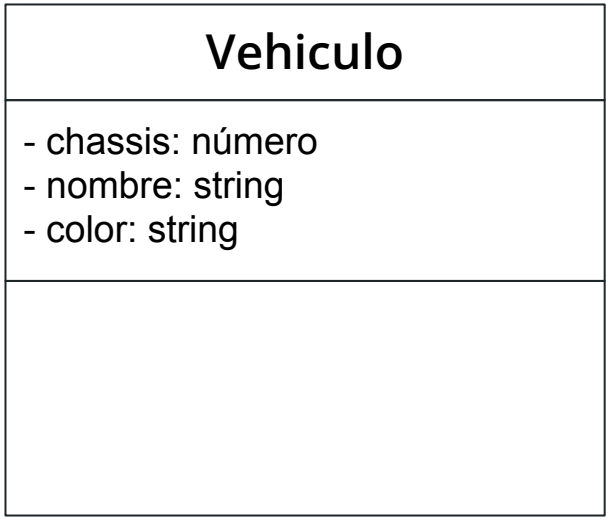
Las principales características de la programación orientada a objetos son:

- Clases y objetos: Una clase es una plantilla que define las propiedades y el comportamiento de un objeto. Los objetos son instancias de una clase, y cada objeto tiene su propia identidad, estado y comportamiento.
- Encapsulamiento: Es el proceso de ocultar la complejidad interna de una clase a través del uso de modificadores de acceso como public, private y protected. Esto permite proteger los datos y métodos de una clase para que no puedan ser accedidos o modificados desde fuera de la clase.
- Herencia: Es la capacidad de una clase de heredar los atributos y métodos de otra clase, lo que permite la reutilización de código y la creación de clases especializadas.
- Polimorfismo: Es la capacidad de una clase de presentar varias formas, es decir, una misma operación puede tener diferentes implementaciones en distintas clases.
- Abstracción: Es el proceso de identificar los atributos y métodos importantes de una clase y enfocarse solo en ellos, ignorando los detalles irrelevantes. Esto permite crear clases más simples y fáciles de entender y utilizar.

Clases C++

Atributos

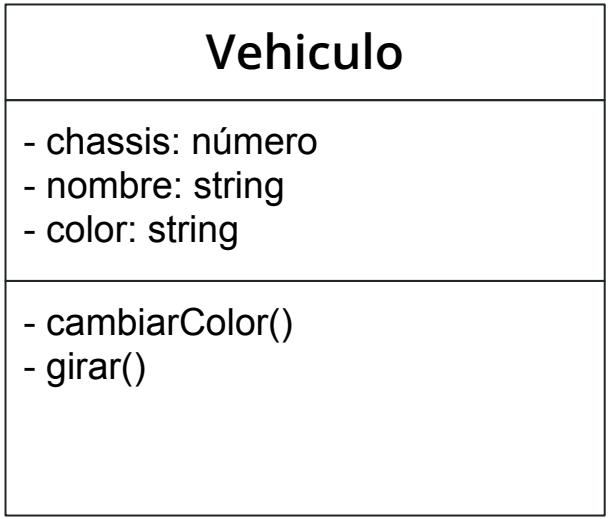
Un dato importante que contiene valores que describen cada instancia de esa clase.



Clases C++

Métodos

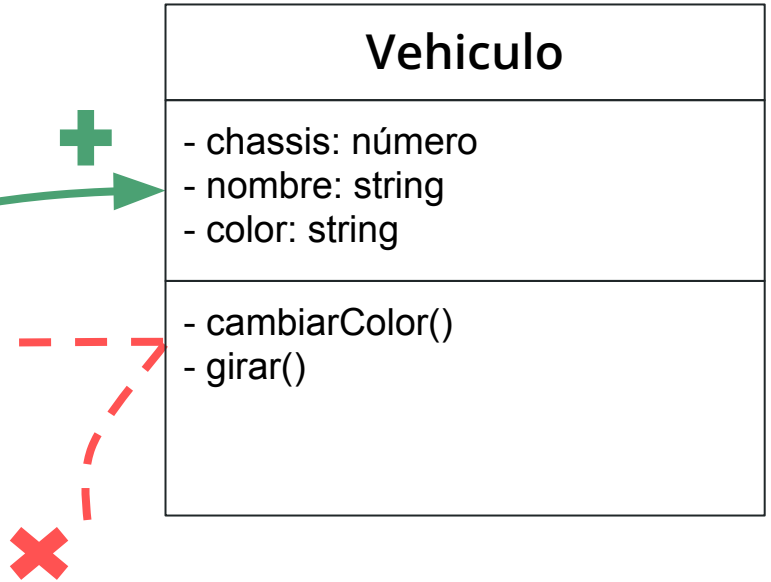
Permiten especificar características o comportamientos de la clase.



Clases C++

Encapsulamiento

- Público (+)
- Privado (-)
- Protegido (#)



Clases C++

Herencia

Clase Derivada

Toyota

-electrico: bool

- acelerar()

Nissan

- acelerar()

Vehiculo

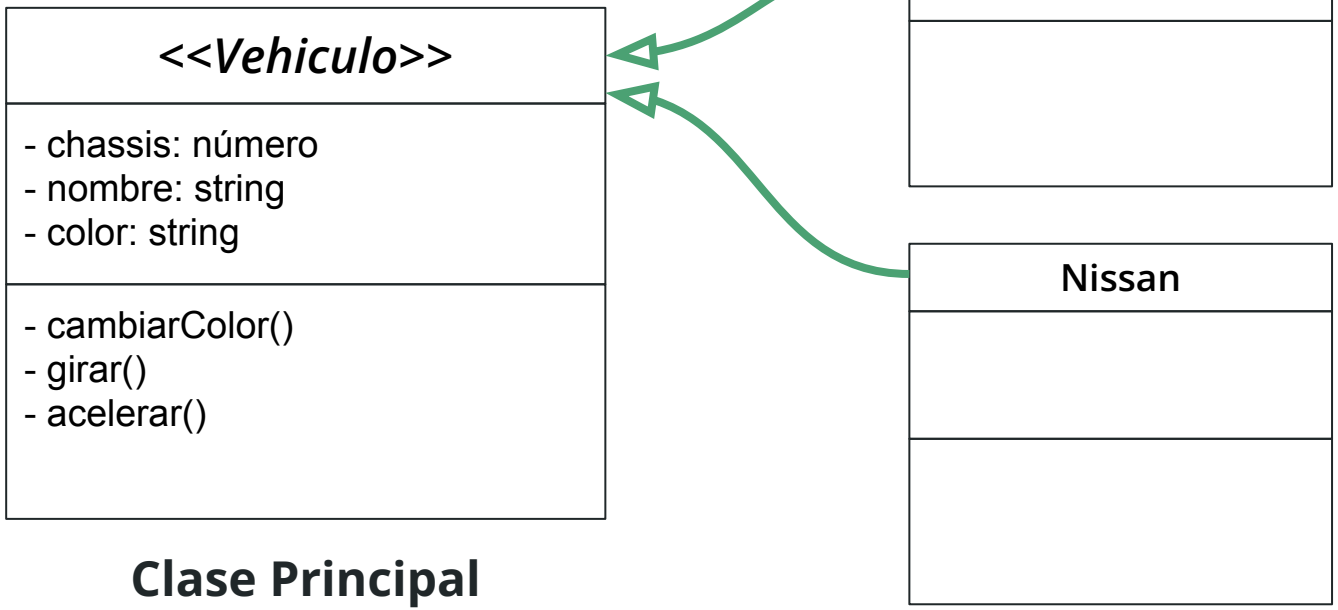
- chassis: número
- nombre: string
- color: string

- cambiarColor()
- girar()

Clase Principal

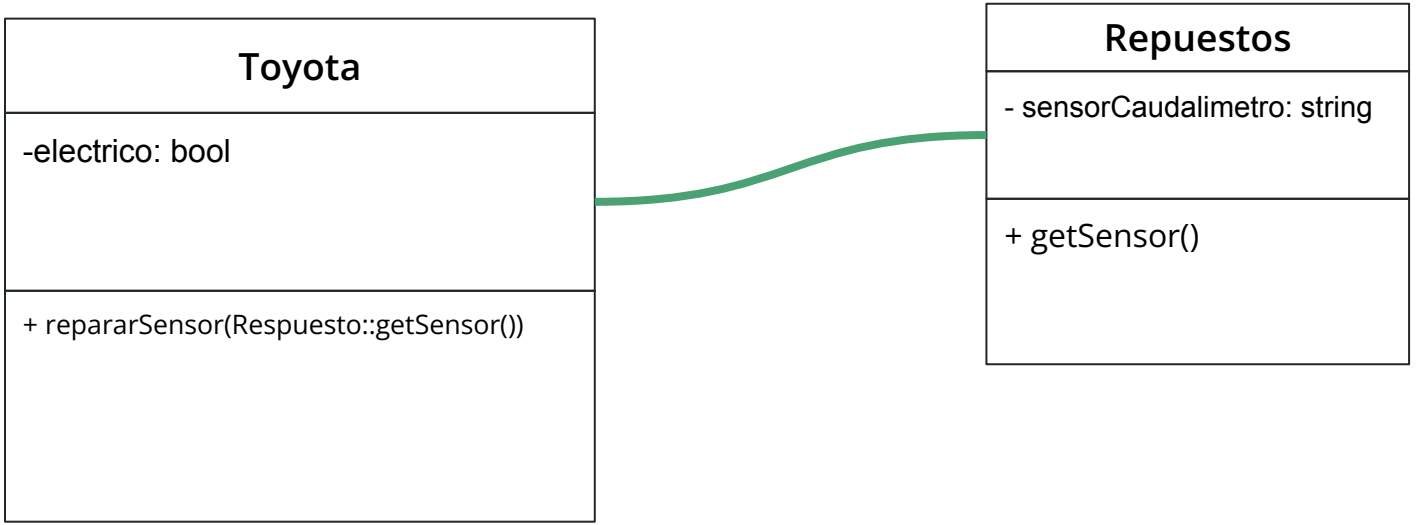
Clases C++

Herencia



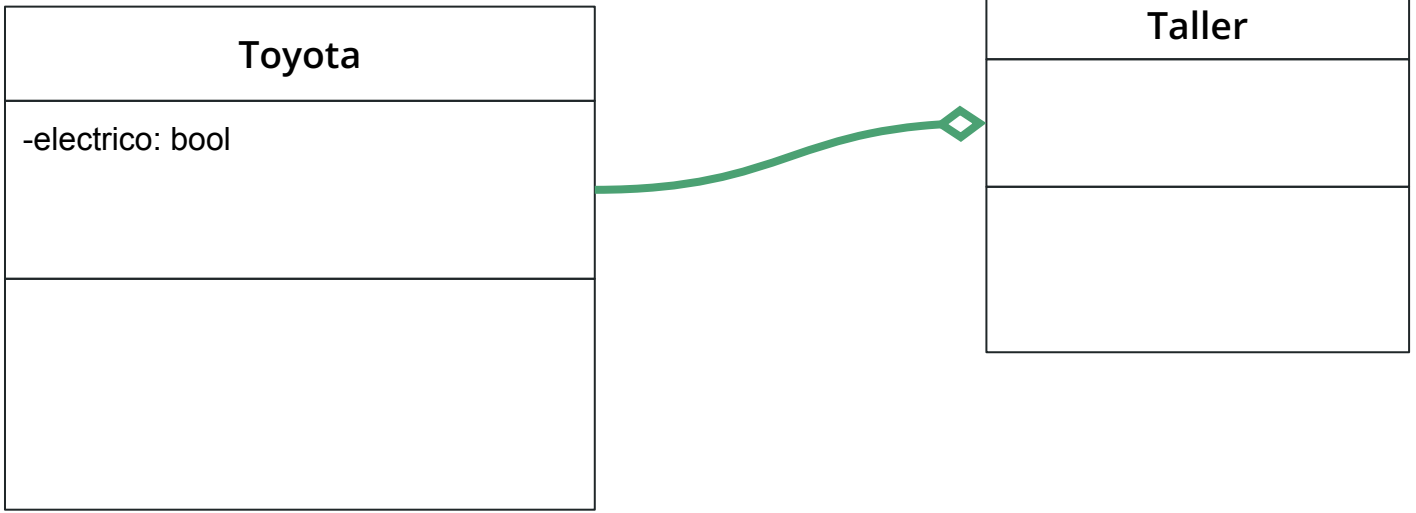
Clases C++

Asociación



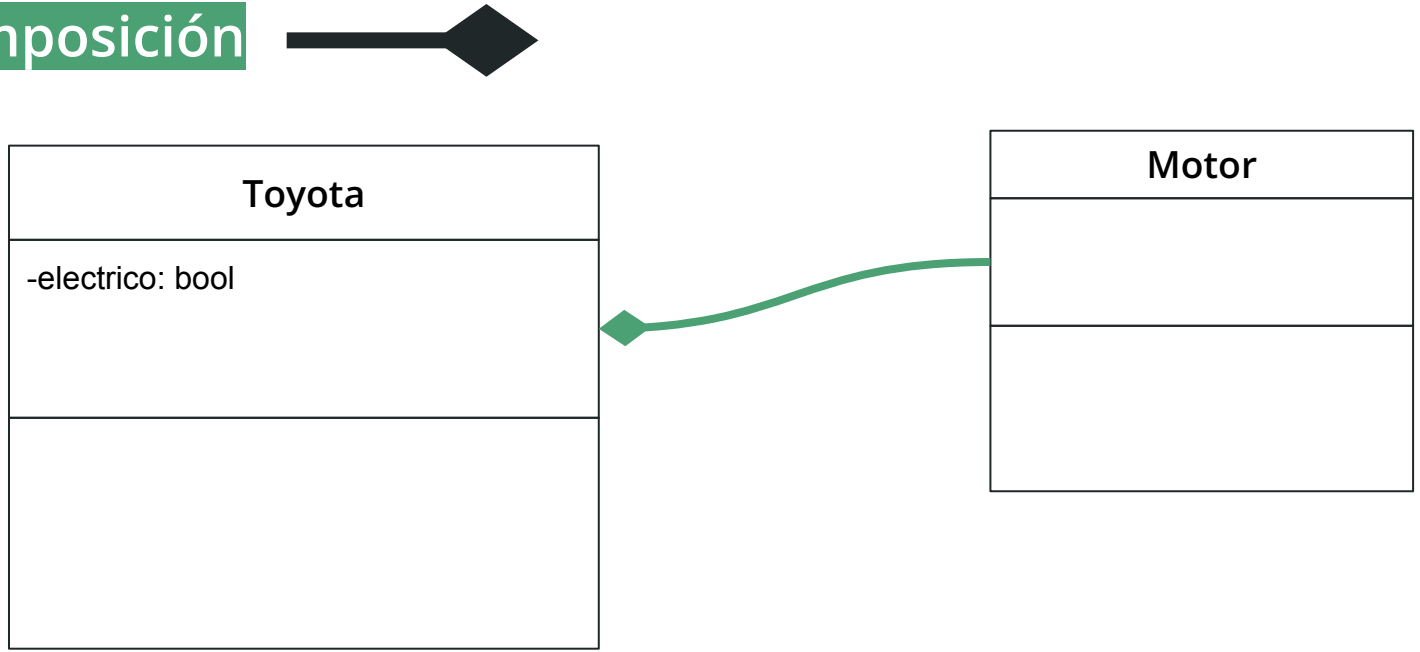
Clases C++

Agregación



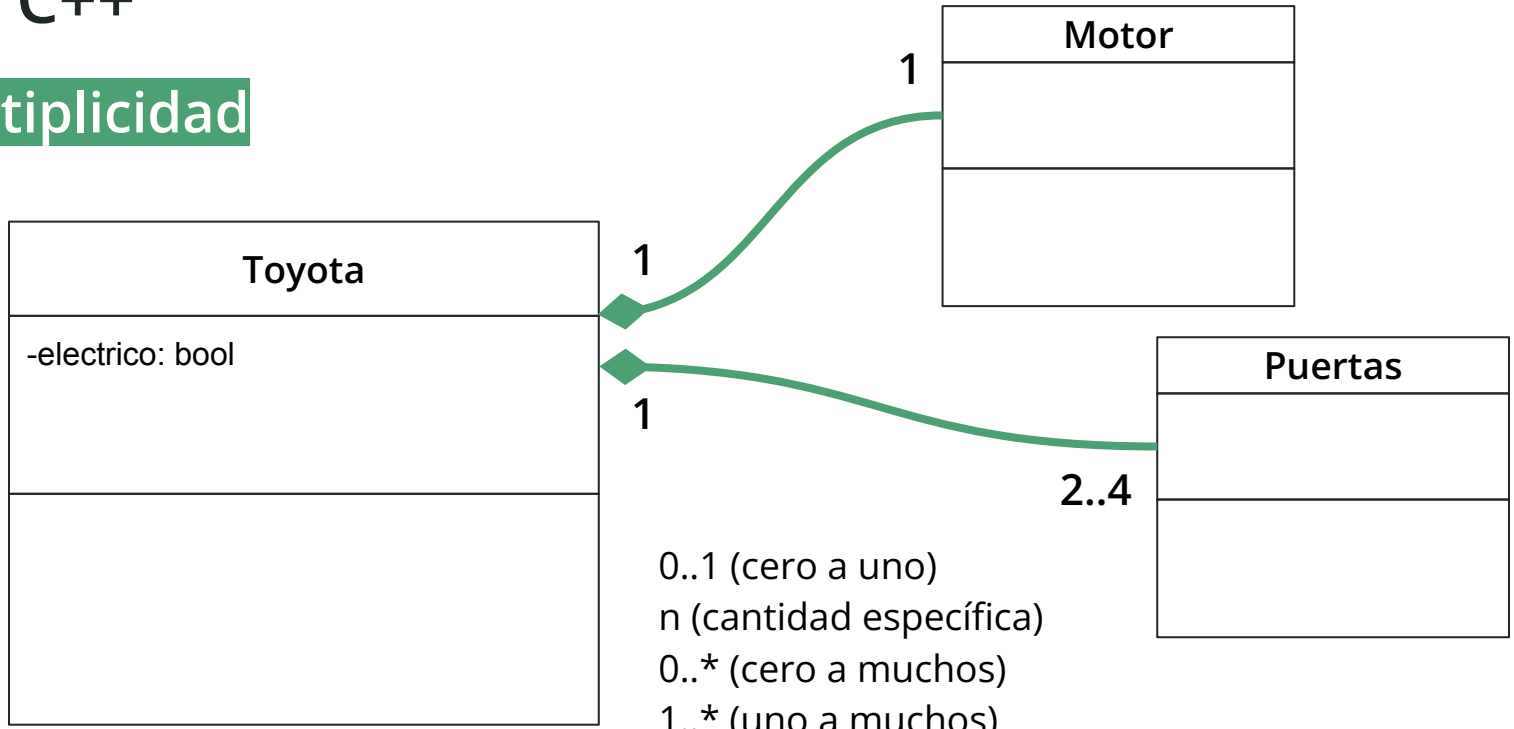
Clases C++

Composición



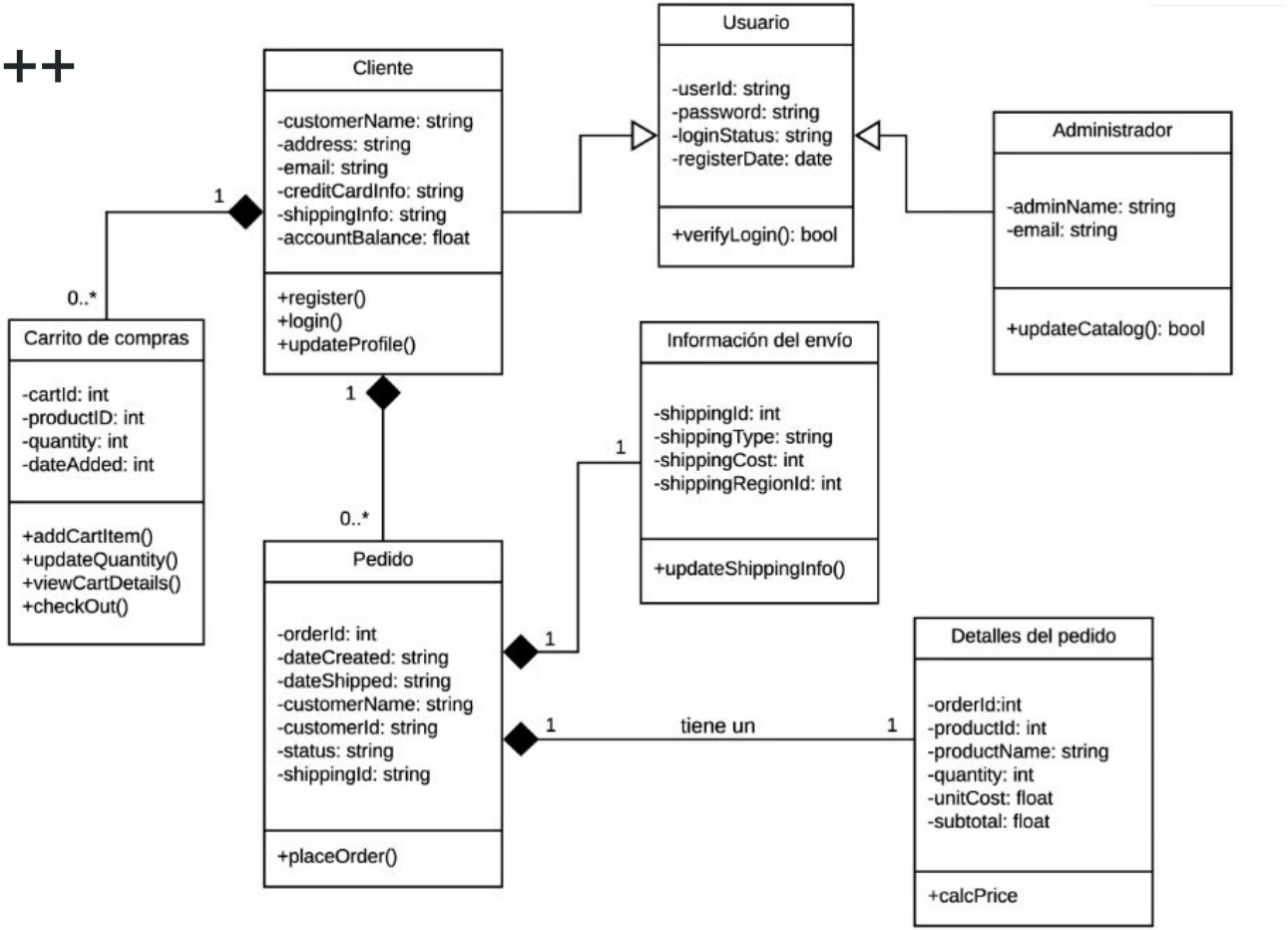
Clases C++

Multiplicidad



0..1 (cero a uno)
n (cantidad específica)
0..* (cero a muchos)
1..* (uno a muchos)
m..n (rango específico)

Clases C++


Usuario
-userId: string
-password: string
-loginStatus: string
-registerDate: date
+verifyLogin(): bool
Cliente
-customerName: string
-address: string
-email: string
-creditCardInfo: string
-shippingInfo: string
-accountBalance: float
+register()
+login()
+updateProfile()
Administrador
-adminName: string
-email: string
+updateCatalog(): bool
1
0..*
Carrito de compras
-cartId: int
-productID: int
-quantity: int
-dateAdded: int
+addCartItem()
+updateQuantity()
+viewCartDetails()
+checkOut()
1
0..*
Información del envío
-shippingId: int
-shippingType: string
-shippingCost: int
-shippingRegionId: int
+updateShippingInfo()
1
Pedido
-orderId: int
-dateCreated: string
-dateShipped: string
-customerName: string
-customerId: string
-status: string
-shippingId: string
+placeOrder()
1
1
tiene un
1
Detalles del pedido
-orderId:int
-productId: int
-productName: string
-quantity: int
-unitCost: float
-subtotal: float
+calcPrice