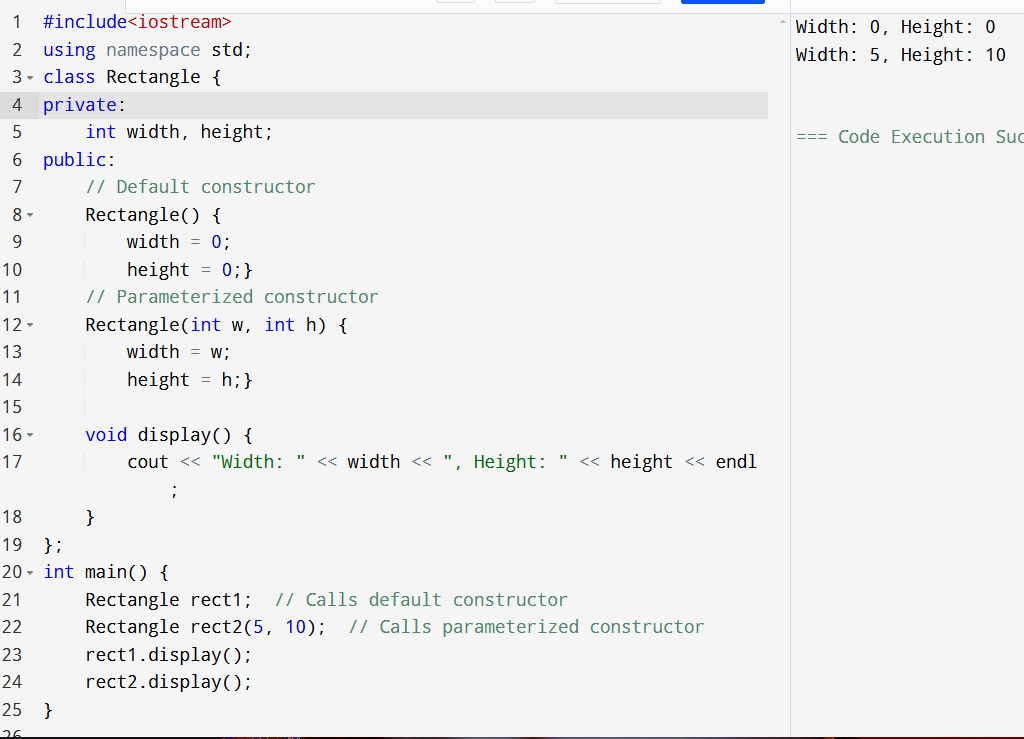
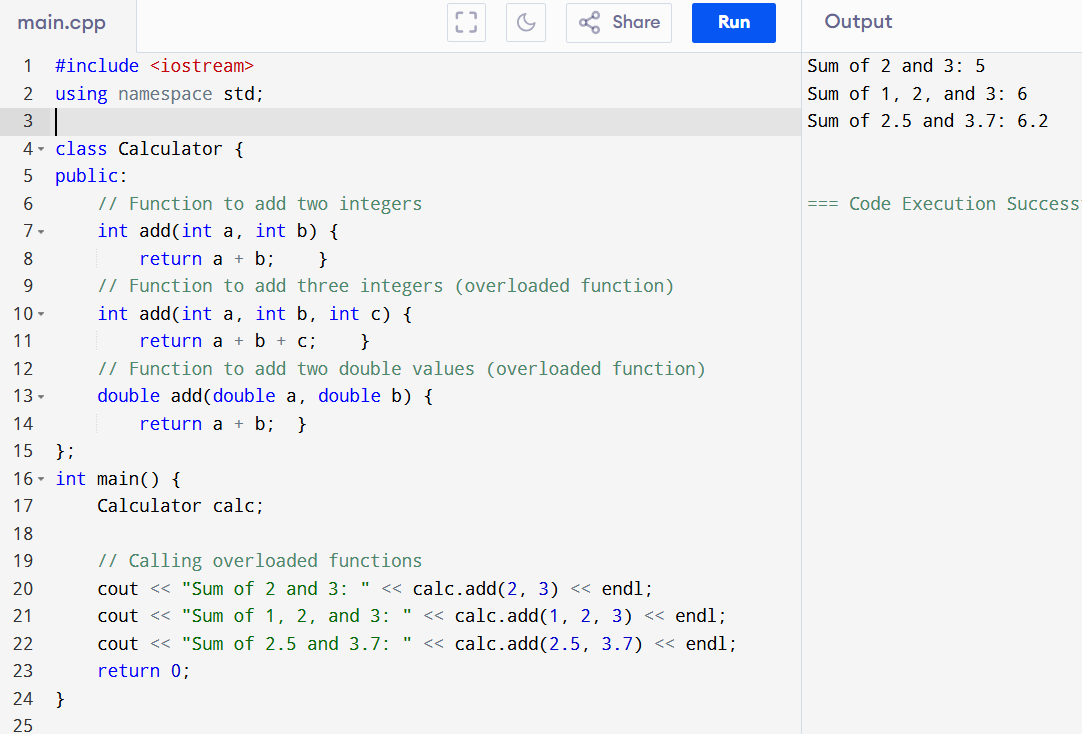
**Constructor with program in C++**In C++, a constructor is a special member function of a class that is automatically called when an object of that class is created. It is used to initialize the object. A constructor has the same name as the class and does not have a return type.

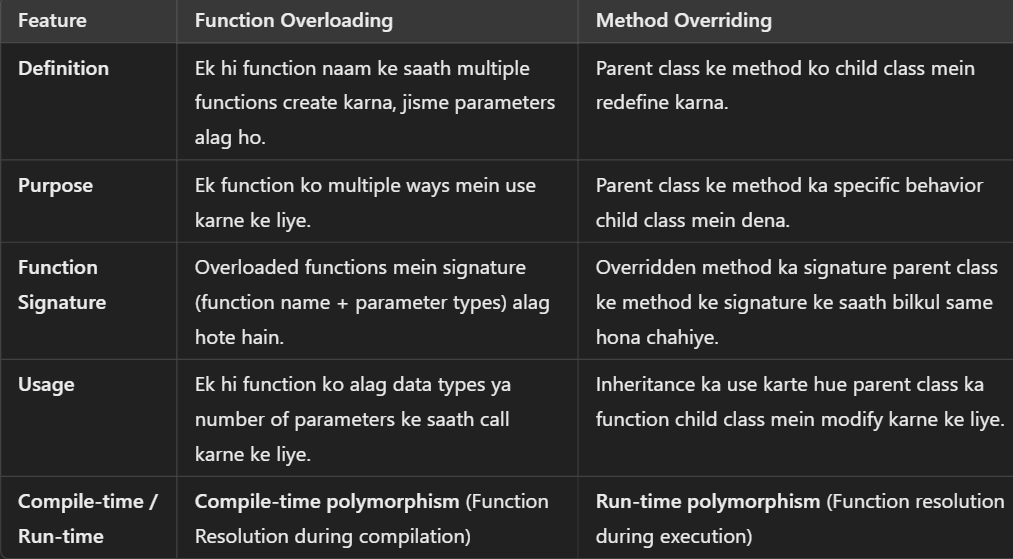
 **Constructor** ki madad se jab object banaya jaata hai, uske saare members ko initial values mil jaati hain bina kisi extra step ke.

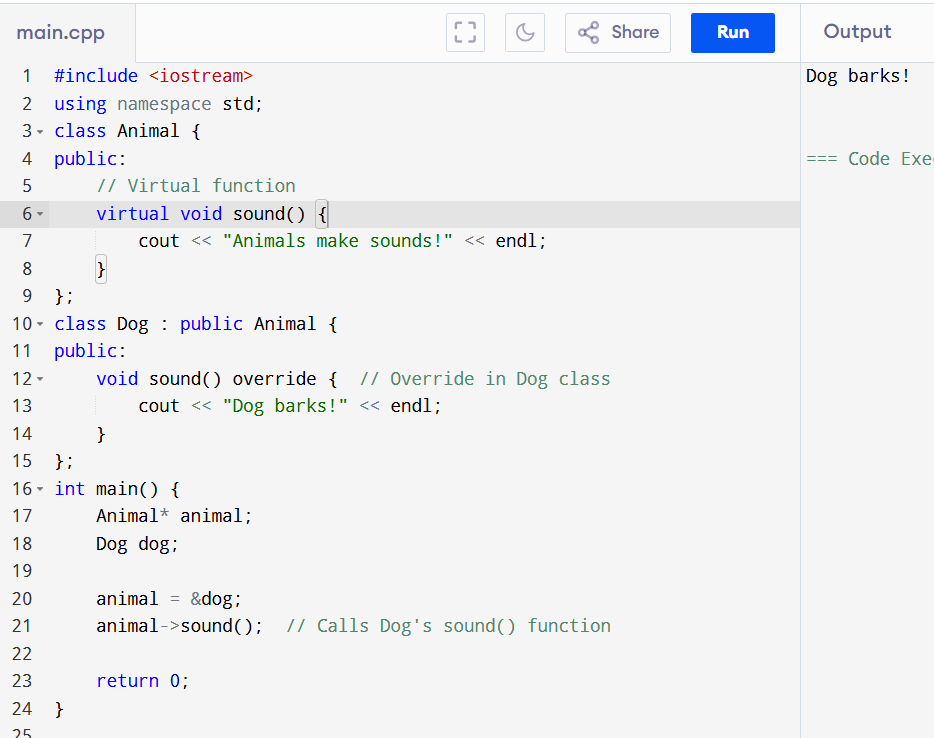
 Agar aap setter functions ka use karte hain, toh aapko manually har object ka initialization karna padta hai, jo bad mein galtiyan karne ka chance badha deta hai.



**Overriding - Overloaded – Overloading**

🡪  
**Overloading (Function Overloading)** (overloaded function)  
  






**Virtual Function:**

**Virtual Function** ek function hota hai jo **parent class** mein declare kiya jaata hai aur jab yeh function **child class** mein override hota hai, tab runtime pe decide hota hai ki kis class ka function call hoga (yani, child class ya parent class ka).

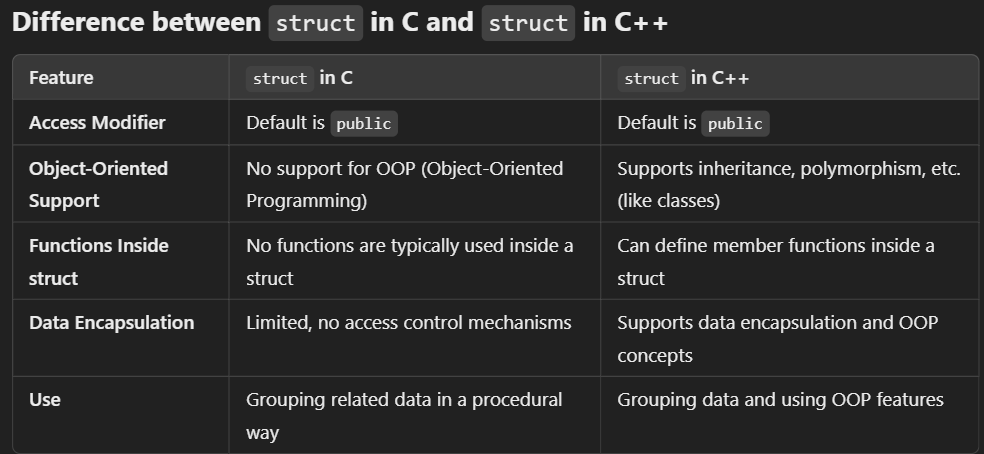
**Virtual function ka use** C++ mein run-time polymorphism achieve karne ke liye hota hai. Iska matlab yeh hai ki aap **base class pointer** ya **reference** ka use karte hue **derived class** ke function ko call kar sakte hain.

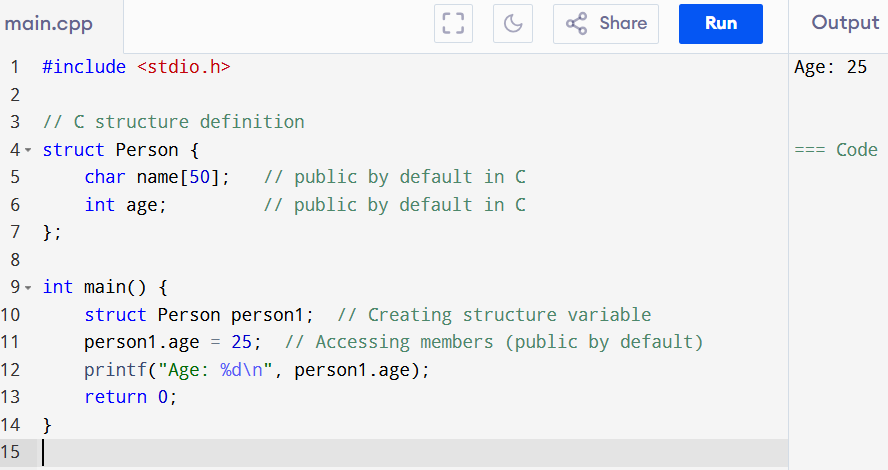
**Structure and Struct**

**Structure** aur **struct** dono ka concept ek hi hai, lekin C++ mein in dono ka use kaafi similar hota hai. **struct** C++ mein ek keyword hai jo structure define karne ke liye use hota hai, aur **structure** ek data type hota hai. Dono ka basic idea ek hi hai, lekin unka usage aur context thoda different ho sakta hai.

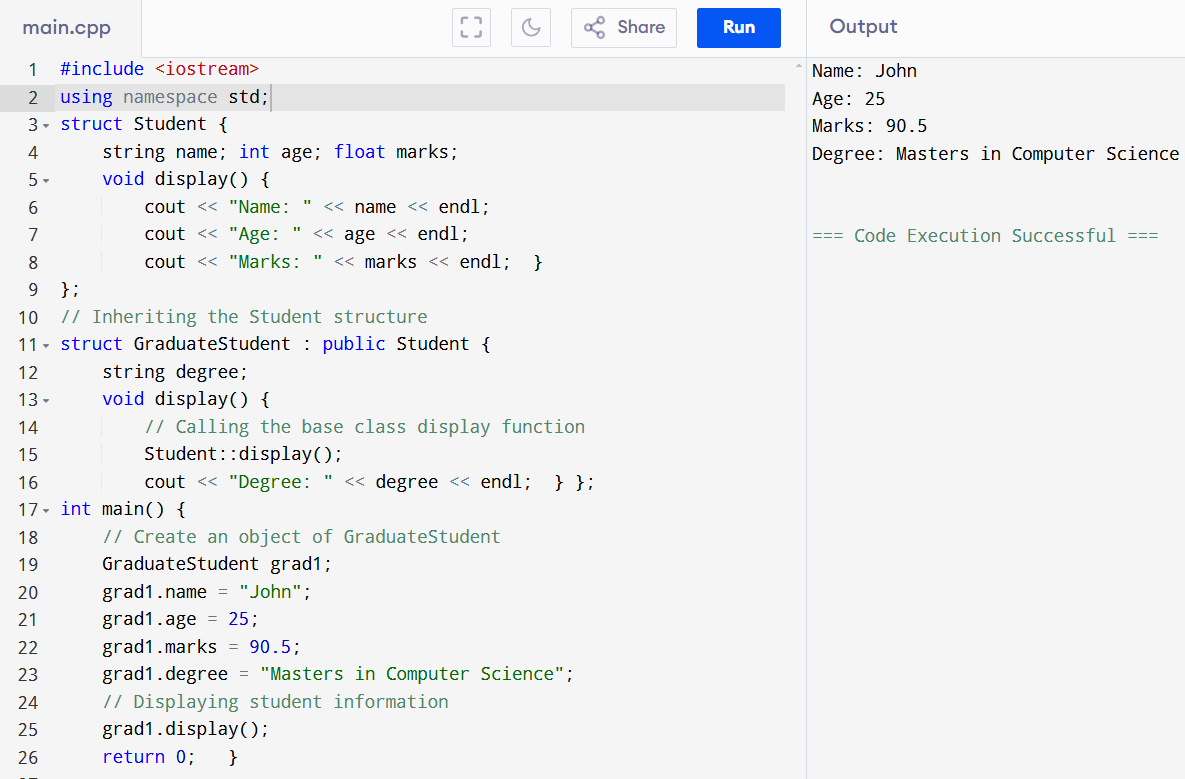
**1. Default Access Specifiers:**

* **C** mein structure ke members ka default access **public** hota hai.
* **C++** mein bhi structure ke members ka default access **public** hota hai, lekin **class** ka default access **private** hota hai.



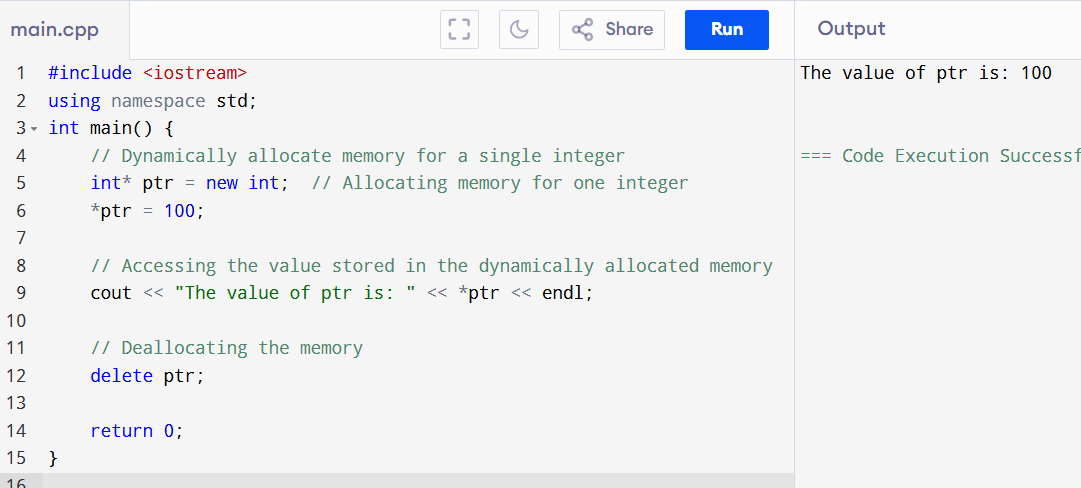


With oops method 🡪



**DYNAMIC MEMORY ALLOCATION :**

Dynamic memory allocation ka use tab kiya jata hai jab aapko runtime pe memory allocate karni hoti hai, jo size compile time pe decide nahi hota. C++ mein dynamic memory allocation ke liye **new** aur **delete** keywords ka use hota hai.



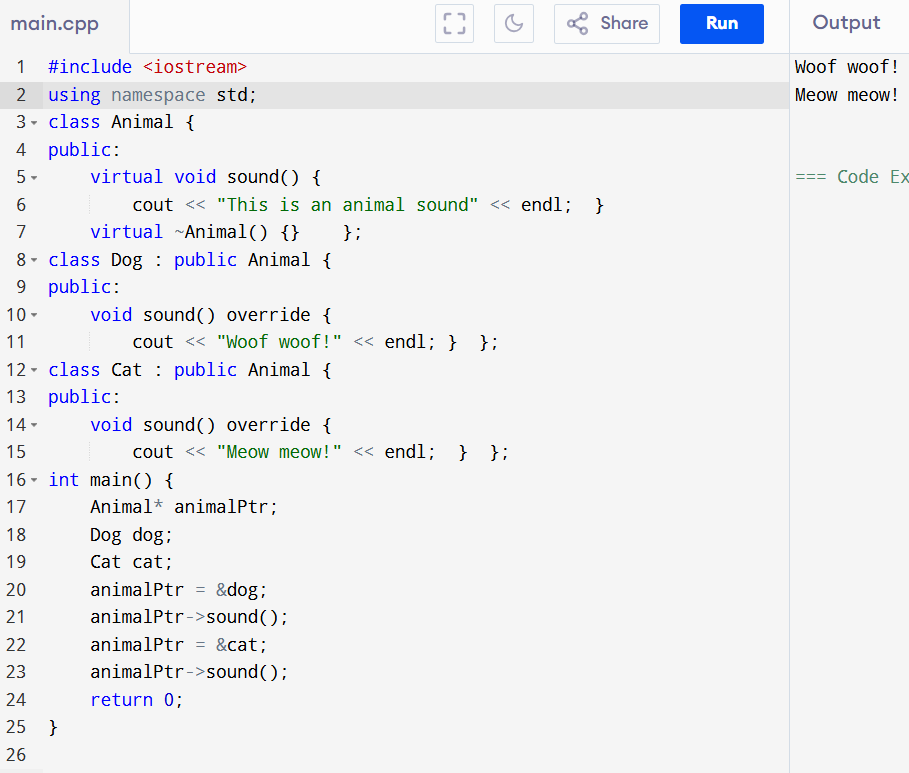
**new int**: Yeh memory allocate karta hai jo ek integer store kar sake.

**delete Operator:**

* **delete** operator dynamically allocated memory ko free karne ke liye use hota hai.
* Jab aap new ka use karte hain, tab memory ko **manually** deallocate karna zaroori hota hai.
* **delete** ka use single variable ke liye hota hai aur **delete[]** ka use array ke liye hota hai.

**Use of Virtual Keyword & Program**

C++ mein **virtual** keyword ka use inheritance aur polymorphism ke concept ko implement karne ke liye hota hai. Jab hum **virtual function** define karte hain, toh C++ ko yeh bata sakte hain ki run-time polymorphism ko support karna hai. **Virtual functions** ka use hum **base class** mein karte hain, jisse **derived class** mein function ko **override** kiya ja sakta hai.



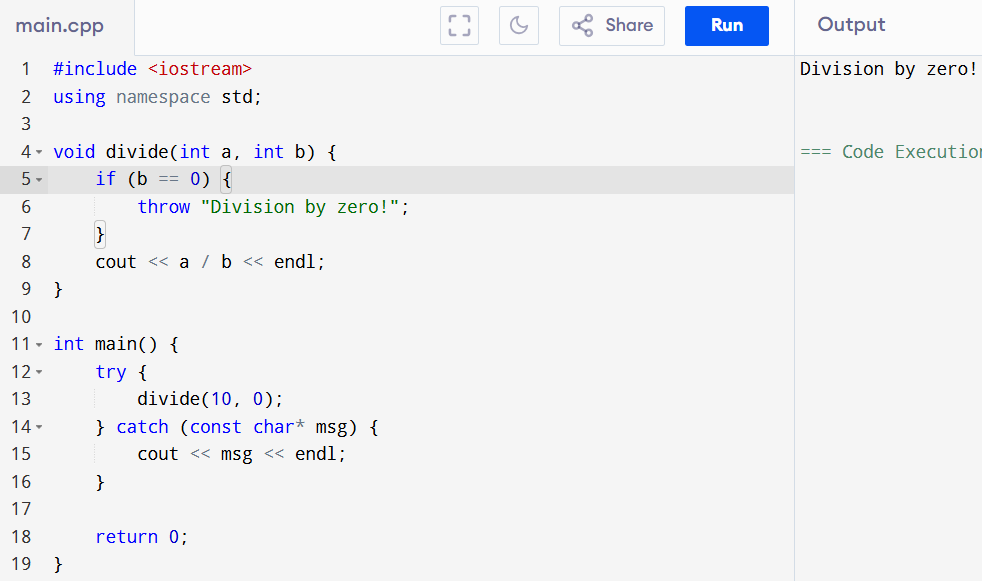
**Exception handling 🡪**

C++ mein exception handling ek mechanism hai jo program ko runtime errors (exceptions) se bachane ke liye use hota hai. Jab koi error occur hota hai, toh program crash nahi hota, balki uss error ko handle karke program ko aage continue karne ka moka milta hai.

C++ mein exception handling ko try, throw, aur catch ke through implement kiya jata hai.

**C++ mein Exception Handling ke 3 main parts hote hain:**

1. **try block**: Yahan pe wo code likhte hain jahan error aa sakta hai.
2. **throw statement**: Agar error aata hai, toh yeh error ko throw (raise) karta hai.
3. **catch block**: Yahan pe hum error ko handle karte hain.

****

 I removed the unnecessary variables x and y and directly called divide with the values 10 and 0.

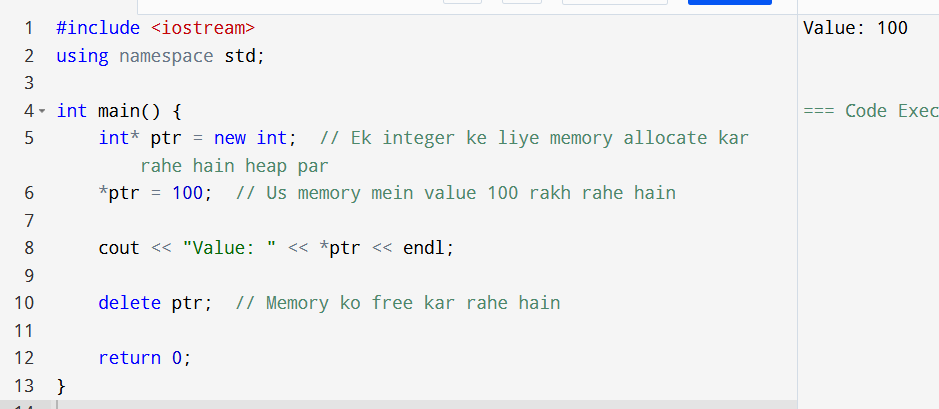
 The divide function directly handles the exception check and prints the result if no exception occurs.

* try **block**: Is block mein woh code hota hai jo exception throw kar sakta hai. Aapke case mein, yeh divide function ko call karta hai 10 aur 0 ke arguments ke sath. Kyunki 0 se divide karna allowed nahi hai, yeh exception throw karega.
* throw **statement**: divide function ke andar, agar b 0 hai, toh throw statement execute hota hai, jo ek exception (is case mein, string "Division by zero!") ko catch block ke paas bhej deta hai.
* **catch block**: Yeh block exception ko catch karta hai jo try block mein throw hoti hai. Yeh ek parameter (is case mein, const char\* msg) leta hai, jo exception hai. Yeh block phir us exception ko handle karta hai, jaise error message print karna.

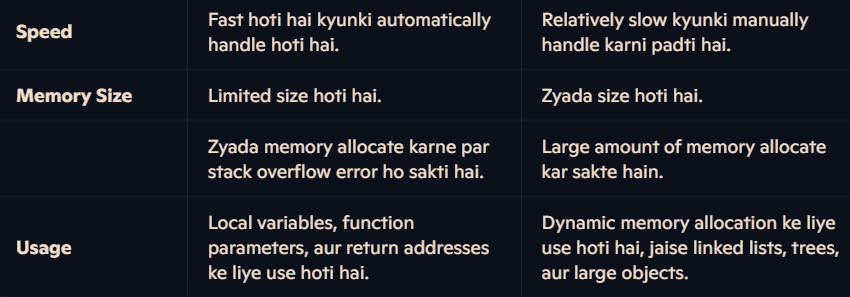
Agar kuch galat hota hai, catch block ensure karta hai ki program crash na ho aur gracefully error ko handle kar sake

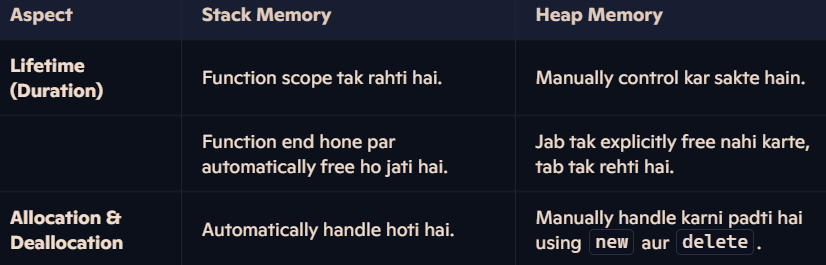
**Heap Memory kya hai?**

Heap memory runtime par dynamically allocate hoti hai. Matlab, jab aapko program ke dauraan memory chahiye hoti hai tab aap usse le sakte ho aur jab kaam ho jaye toh free kar sakte ho.

****

**Difference Heap & Main Memory**

****

****

C++ mein **access modifiers** ka use class members (variables aur functions) ke access ko control karne ke liye kiya jata hai. C++ mein **3 main access modifiers** hote hain:

 **public**

 **private**

 **protected**

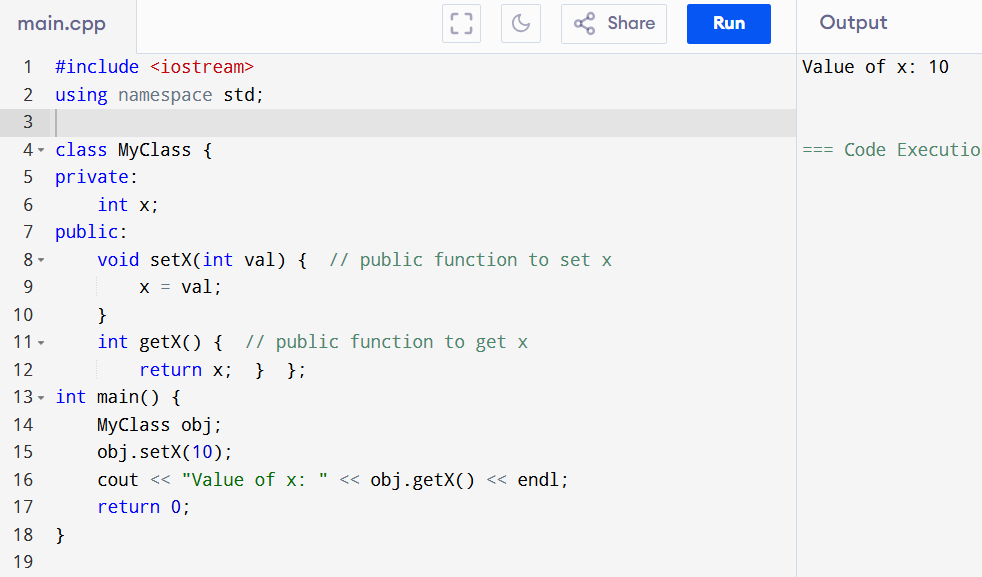
**1. public Access Modifier**

* **Public members** ko class ke **bahar** se **access** kiya ja sakta hai.
* Yeh generally wo functions ya variables hote hain jo users ko interact karne ke liye available hote hain.

****

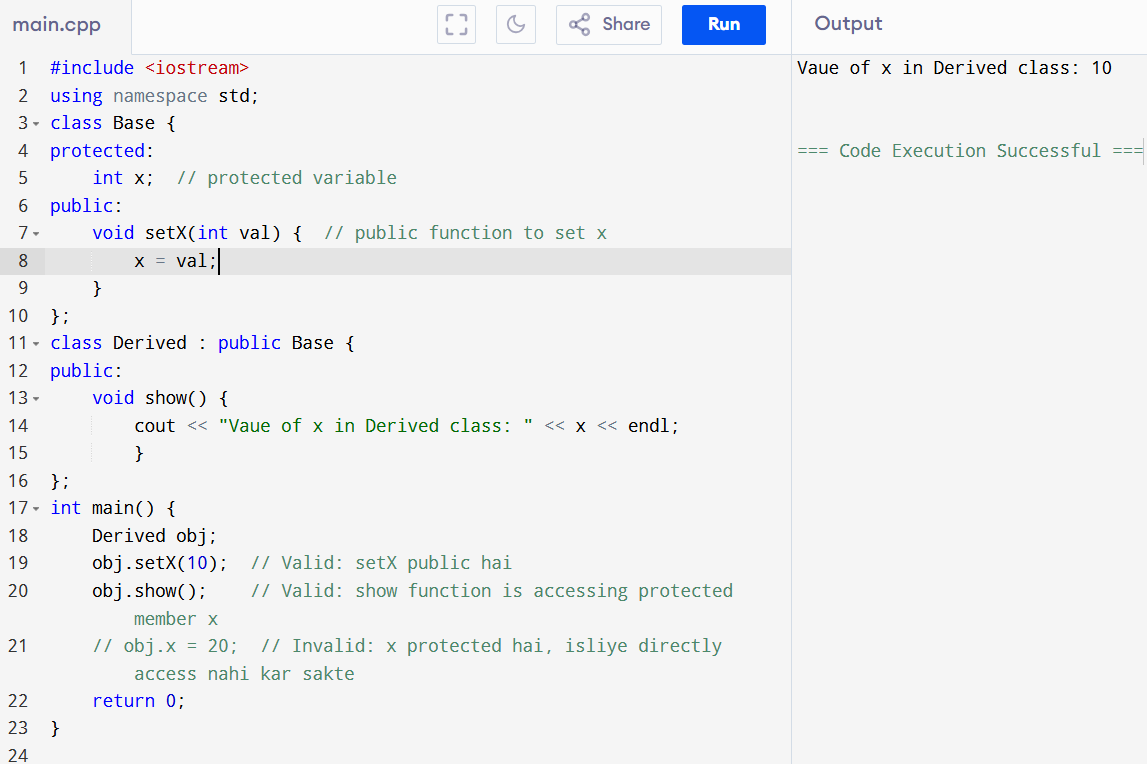
**2. private Access Modifier**

* **Private members** sirf class ke **andar** hi access kiye ja sakte hain.
* Class ke **bahar** se private members ko directly access nahi kiya ja sakta.
* Yeh usually sensitive data ko hide karne aur encapsulation achieve karne ke liye use hota hai.

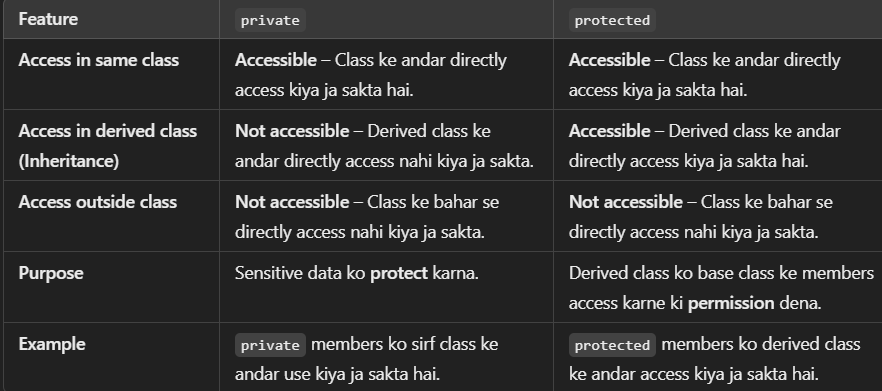
****

**3. protected Access Modifier**

* **Protected members** ko class ke **andar** aur class ke **derived (child) classes** ke **andar** access kiya ja sakta hai.
* **Class ke bahar** se protected members ko directly access nahi kiya ja sakta.
* Yeh **inheritance** ke time pe kaam aata hai, jab aap derived class ke andar parent class ke protected members ko access karna chahte hain.

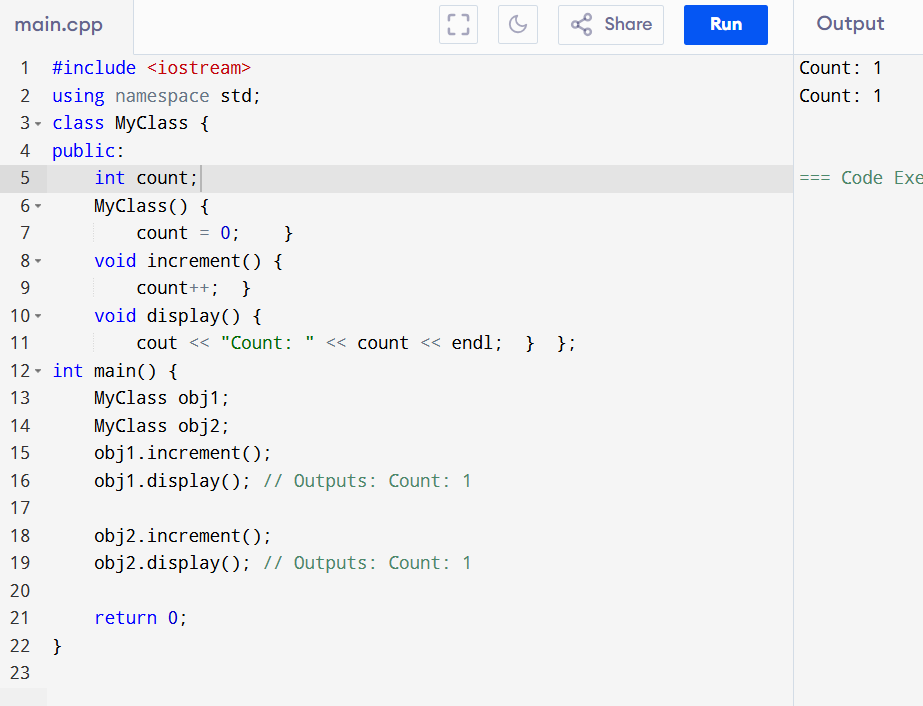


Difference 🡪

****

**STATIC KEYWORD and Without Static 🡪**

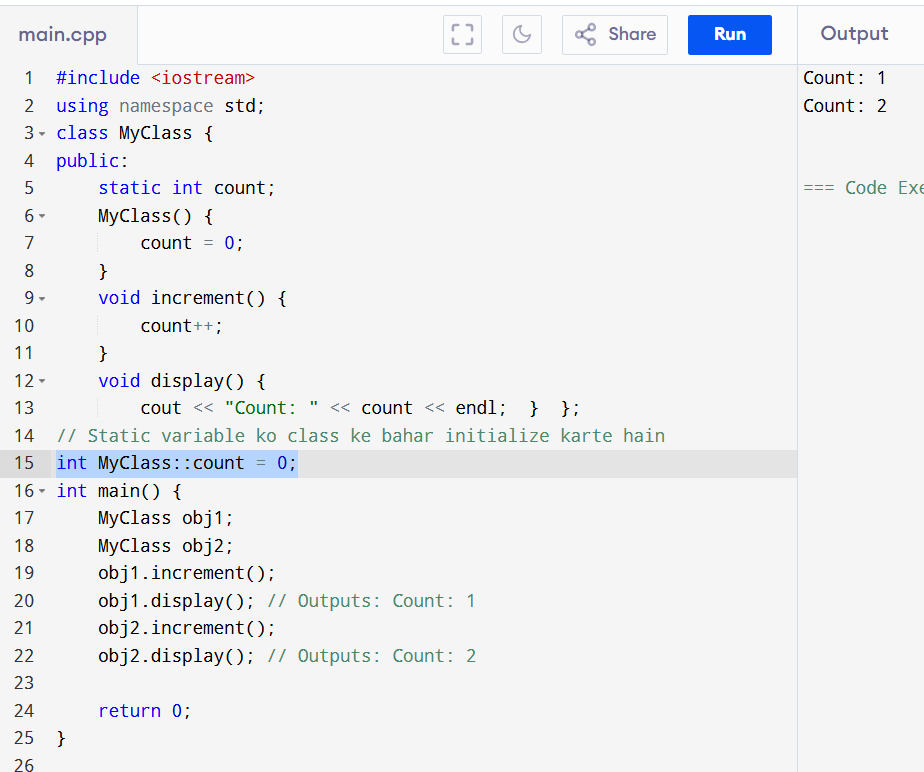
**Without :**

****

 count non-static hai, isliye har object apni alag copy rakhta hai.

 obj1 aur obj2 ke count variables alag-alag hain, isliye dono objects ke increment aur display calls alag-alag results dete hain.

**With Static**

****

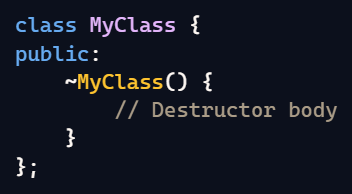
 count static hai, isliye yeh class ke sabhi objects ke beech share hota hai.

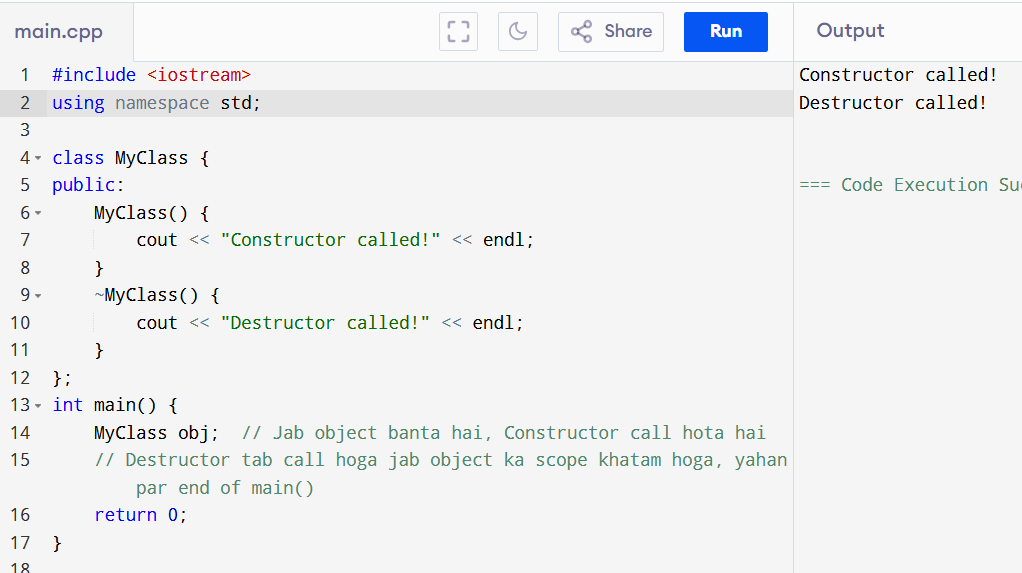
 obj1 aur obj2 ka count variable common hai, isliye increment aur display calls dono objects ke liye same result dete hain.

**Destructor kya hota hai?**

Destructor ek special function hota hai jo tab call hota hai jab object destroy hota hai, matlab jab object ka lifecycle khatam hota hai ya object delete hota hai. Destructor ka kaam resources ko release karna hota hai jo object ne use kiye hain, jaise dynamically allocated memory.

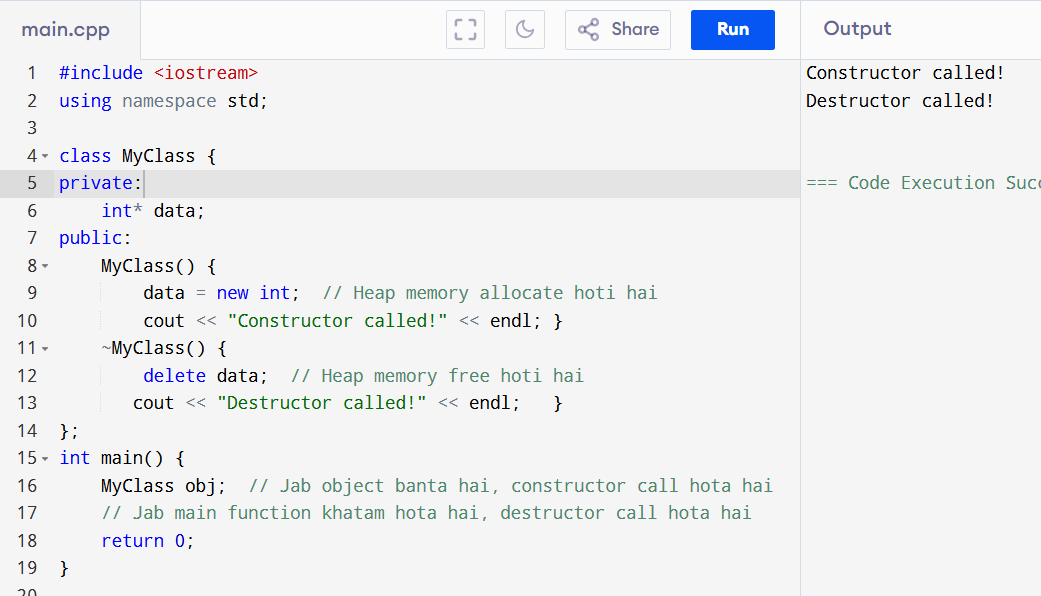
**Destructor ka naam class ke naam ke aage tilde (~) laga ke banaya jaata hai aur iske koi parameters nahi hote.**





**Dynamic Memory Example:**

Jab object dynamically memory allocate karta hai (**heap memory),** to destructor memory ko free karne ke liye use hota hai.

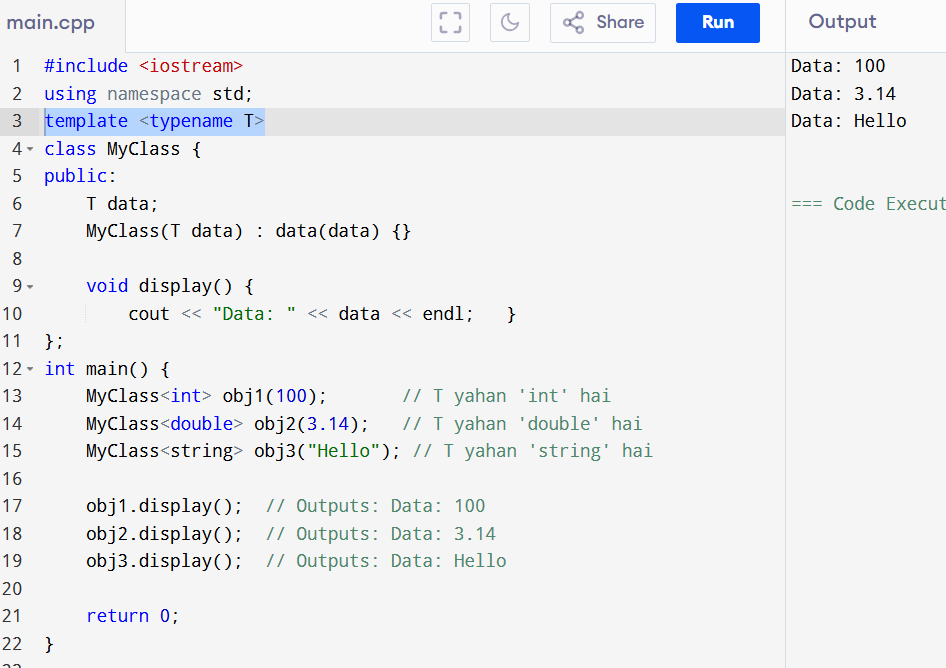


**Template Argument kya hota hai?**

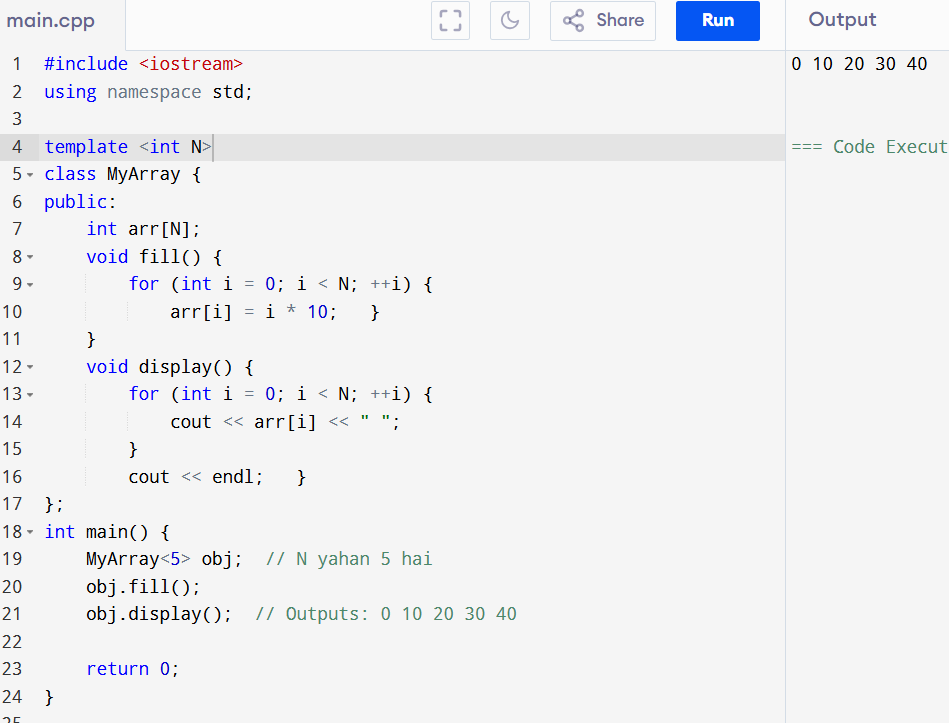
Templates C++ mein ek tareeka hain flexible aur reusable code likhne ka. Template arguments basically woh specific types ya values hain jo aap templates ko dete ho taaki woh unke sath kaam kar sakein.

**Types of Template Arguments:**

1. **Type Argument:** Yeh data type specify karta hai, jaise int, double, etc.
2. **Non-Type Argument:** Yeh ek constant value specify karta hai, jaise int, char, etc.

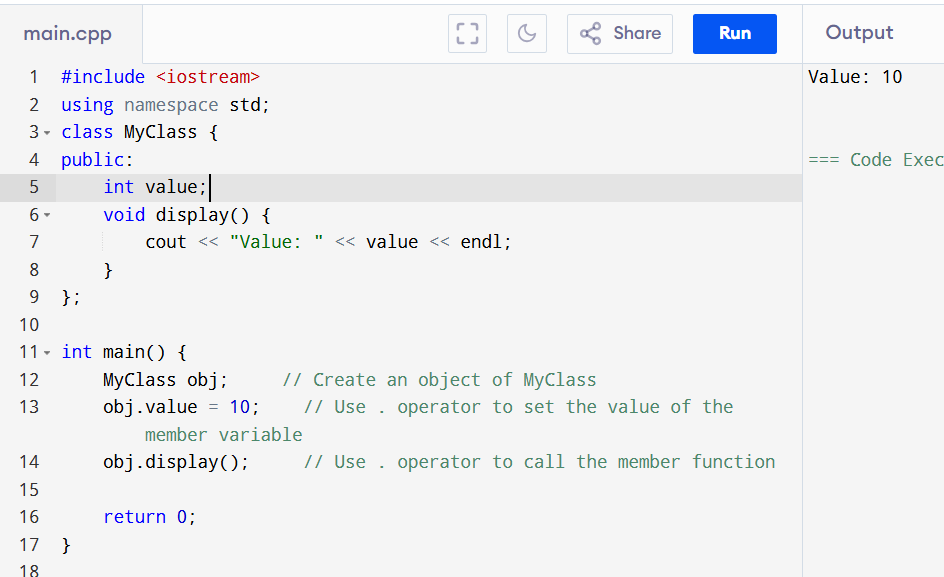


NON- type



. (DOT) operator 🡪

The . (dot) operator in C++ is used to access members (variables and functions) of an object. It's a way to call methods or access properties of a particular instance of a class. Here’s a simple explanation and example:

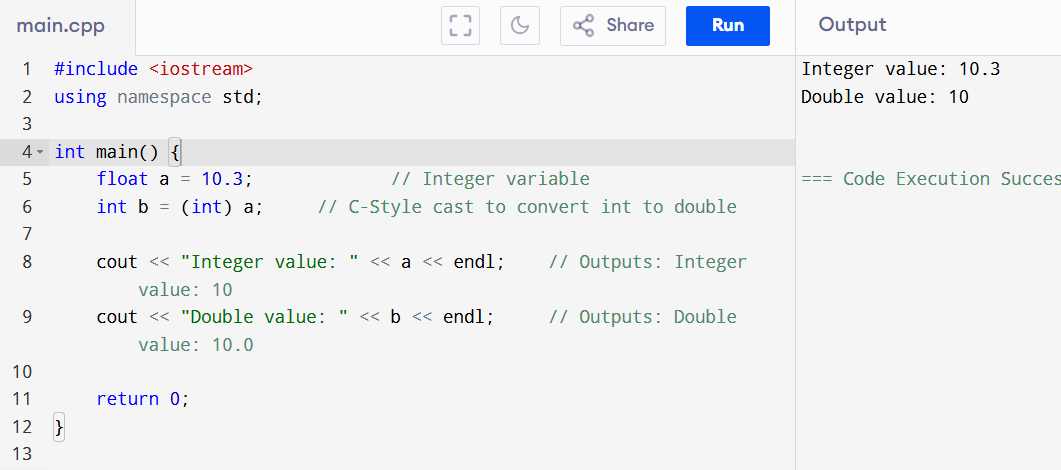


**Casting in C++:**

Casting ka matlab hai ek type ke variable ko doosre type ke variable mein badalna (convert karna)

**C-Style Cast:**

* Sabse simple aur purani tareeka.
* Syntax: (type) variable

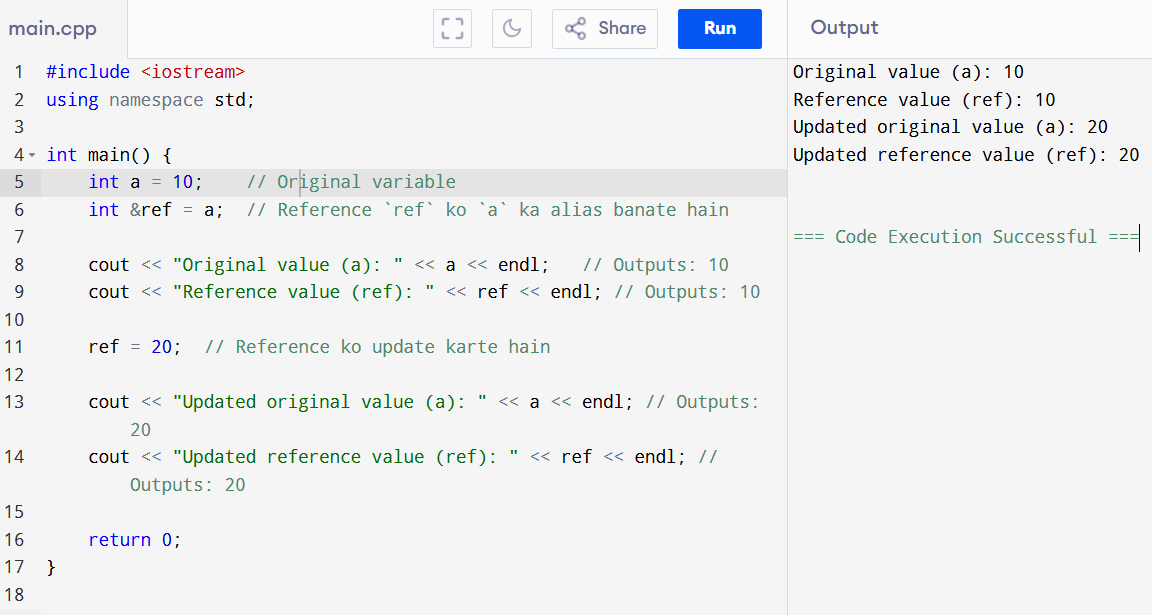


**C++ mein references** ka use variables ko ek alias (nickname) dene ke liye hota hai. References pointers jaise hi hote hain, lekin unhe initialize karna aur use karna zyada simple hota hai. Aayein, simple shabdon mein samjhte hain references ko aur ek example dekhte hain:

 **Declaration:** Reference ko declare karte samay usse initialize karna zaroori hota hai.

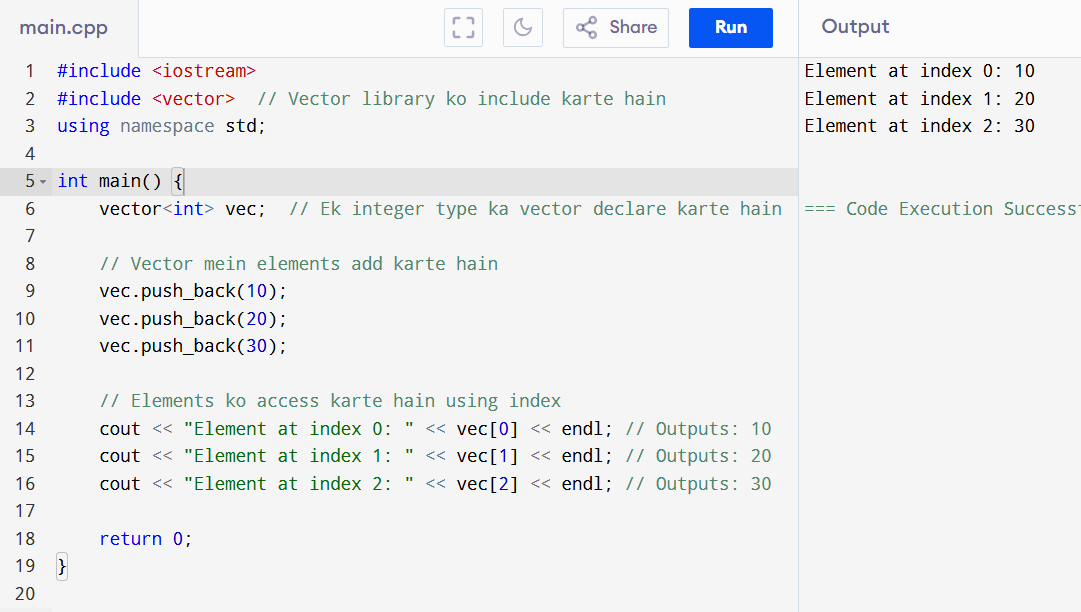
 **Syntax:** type &ref = original\_variable;

 **Alias:** Reference kisi original variable ka alias banta hai. Matlab, koi bhi operation jo reference par kiya jata hai, woh original variable par apply hota hai.

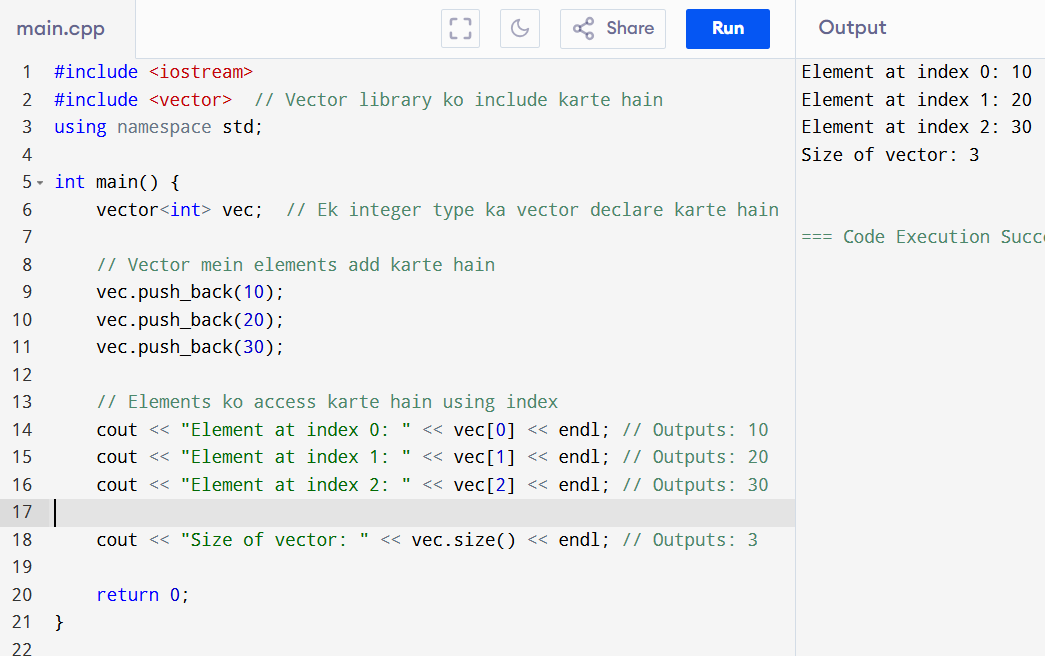


**Vectors kya hain?**

* Vectors C++ mein ek type ka dynamic array hota hai.
* Vectors ka size badhaya ya ghataaya ja sakta hai jab zaroorat ho.



Find size 🡪



Type Def :

Agar aap directly data types use karte hain, toh functionality par koi farak nahi padta. Lekin typedef ka fayda ye hai ki aapke code ko zyada readable aur maintainable banata hai, khas kar jab complex data types ya long declarations hote hain.

Without typedef :



With :



**Advantages:**

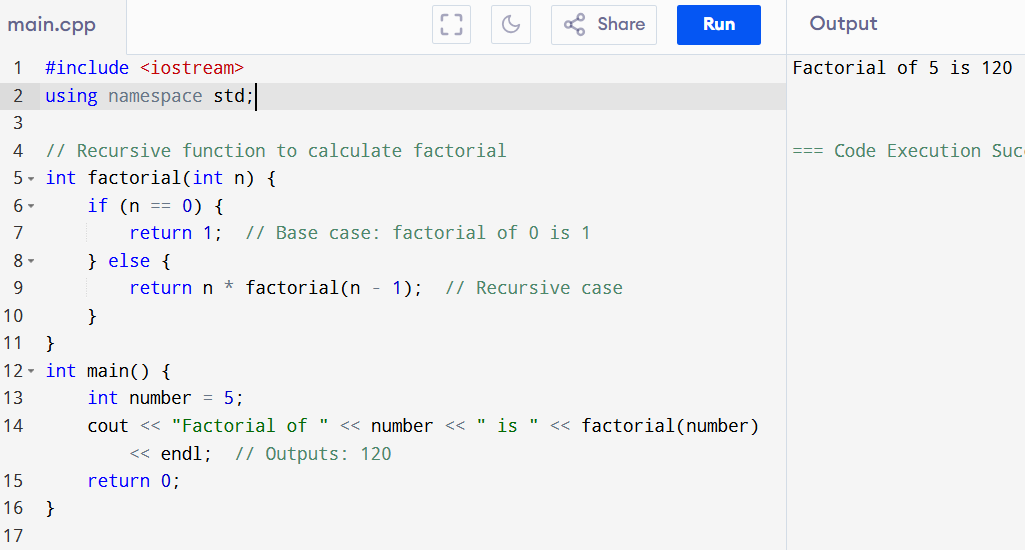
* **Readability:** Code zyada readable aur understandable ho jata hai.
* **Maintenance:** Code ko maintain karna easy ho jata hai.
* **Clarity:** Complex types ko simplify karke clarity badhata hai

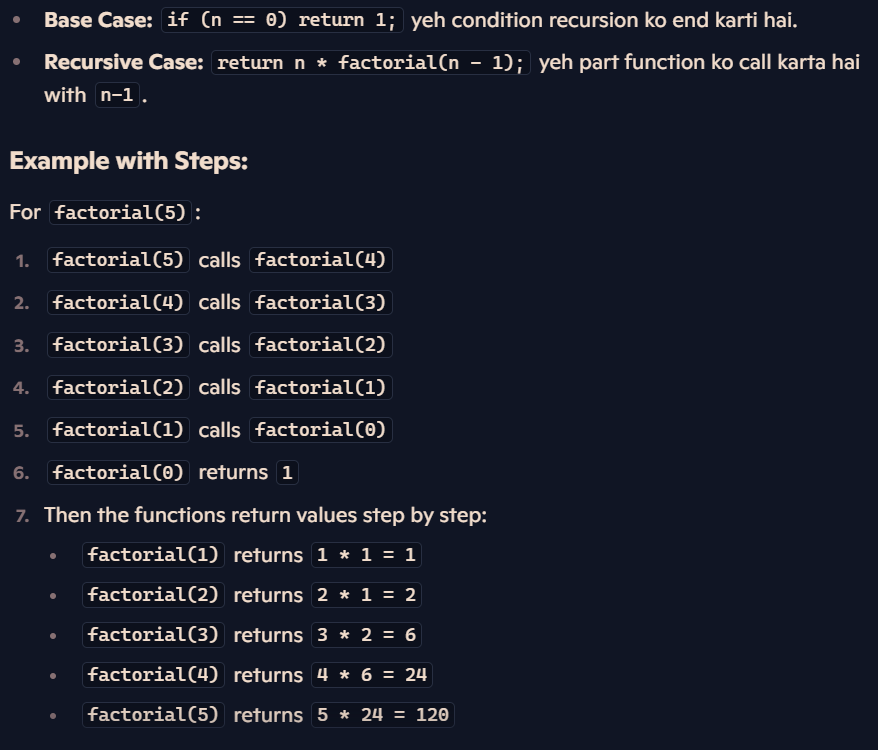
**::**

**Recursive function** ek aisi function hoti hai jo apne aap ko call karti hai. Yeh useful hoti hai jab aapko kisi problem ko smaller subproblems mein tod ke solve karna hota hai. Recursive functions ko use karte waqt base case aur recursive case ka dhyaan rakhna padta hai.

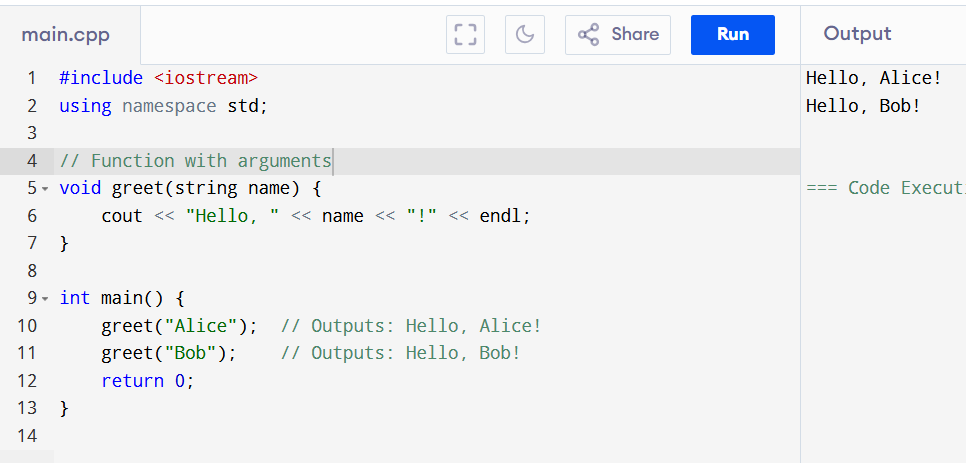
### Key Components of Recursive Function:

1. **Base Case:** Yeh condition hoti hai jo recursion ko end karti hai. Without base case, recursive function infinite loop mein fas sakti hai.
2. **Recursive Case:** Yeh part function ko call karta hai apne aap ko with different arguments.





**ARGUMENT AND WITHOUT ARGUMENT IN C++**



 greet(string name) ek function hai jo ek argument name leta hai.

 greet("Alice") aur greet("Bob") function calls hain jo name ko "Alice" aur "Bob" set karte hain.

Without Arguments 🡪

