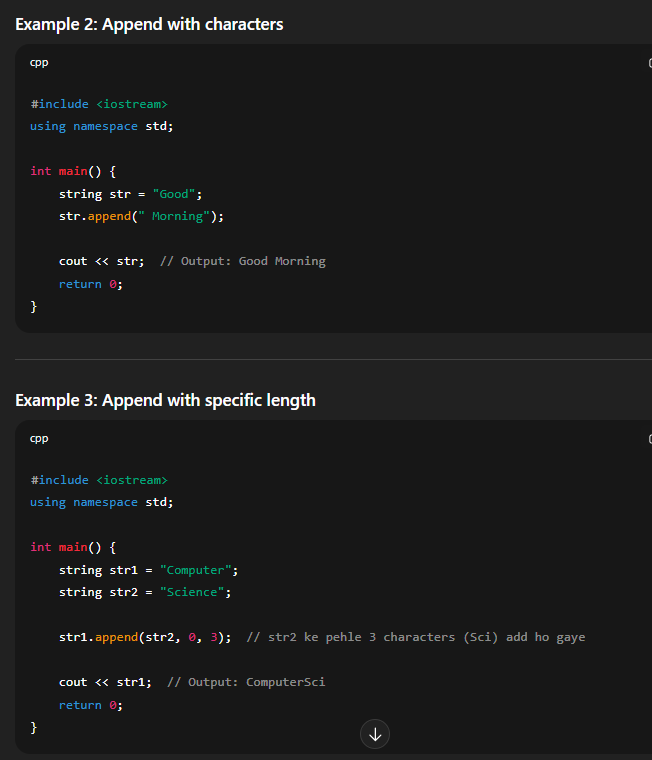
**Append() Function in C++**

👉 Append() ka matlab hota hai ek string k andar dusri string ya characters add karna (jornay ka kaam).

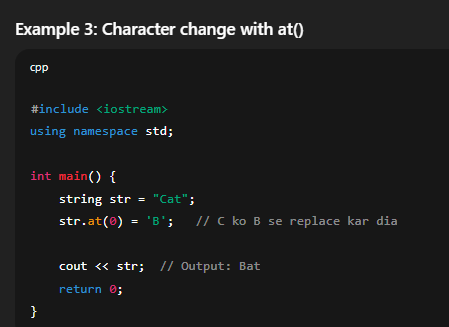




**at() Function in C++**

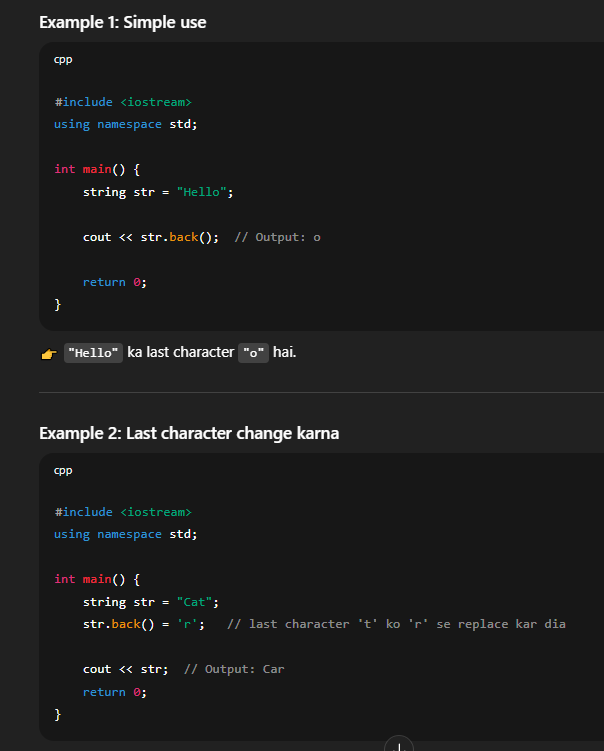
👉 at() ek string ke specific character (letter) ko access karne ke liye use hota hai.

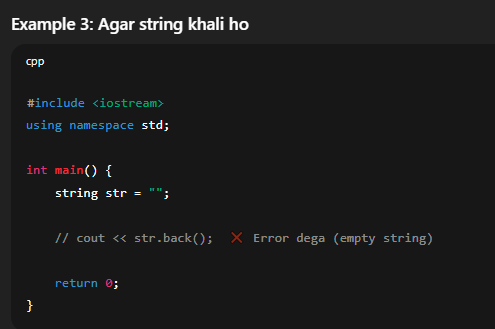




**back() Function in C++**

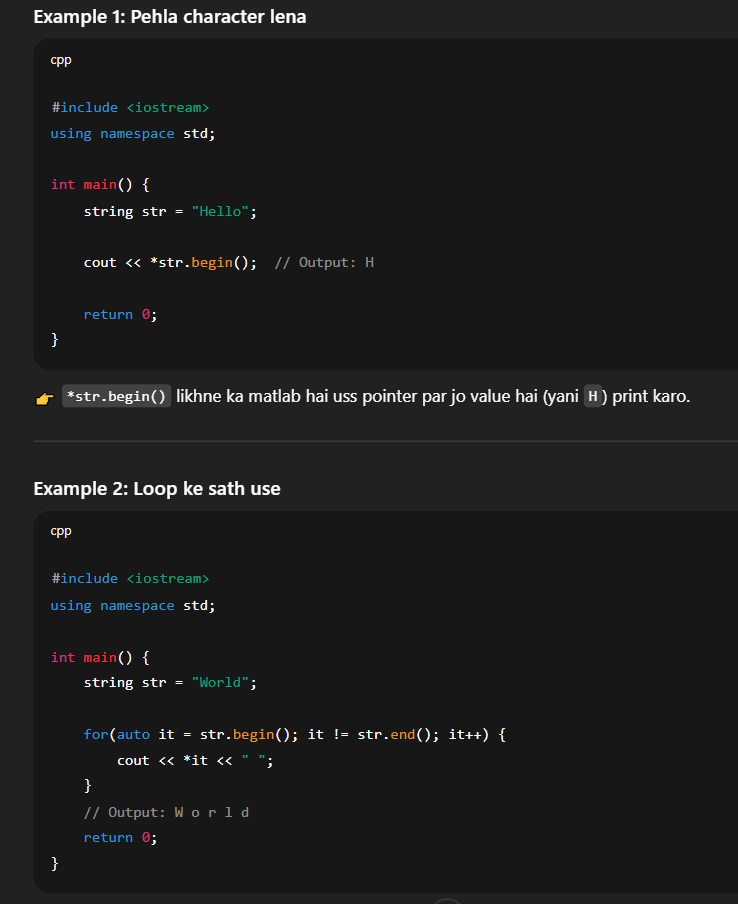
👉 back() ek string ka last character return karta hai.

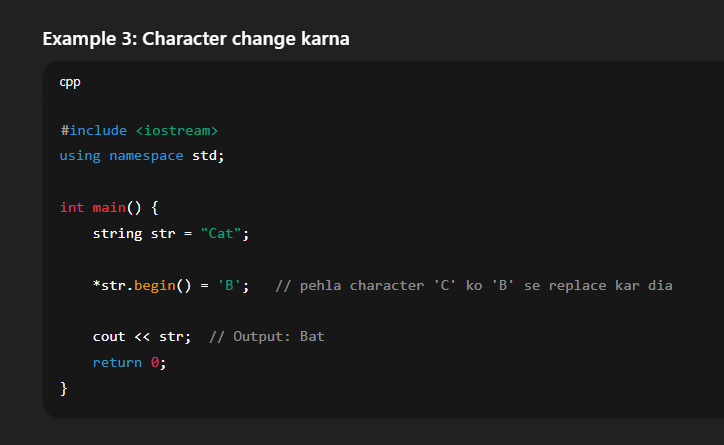


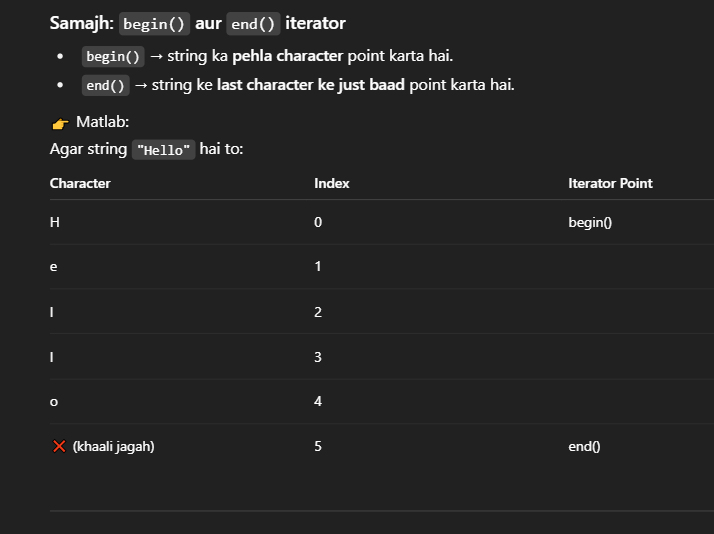


**begin() Function in C++**

👉 **begin() ek pointer (iterator) return karta hai jo string ke pehle character par point karta hai.**  
Matlab string ke **first letter ka address** deta hai.







**c\_str() Function in C++**

👉 *c\_str() ek function hai jo C++ string ko C-style string (character array / const char) mai convert karta hai.*\*

**Basic Idea**

C++ mai do tarah ki strings hoti hain:

1. **C++ string** → string name = "Ali";
   * Ye modern hai, easy hai.
2. **C-style string (char array)** → "Ali" actually hota hai A l i \0
   * Ye purana system hai jo C language mai use hota tha.

Ab problem ye hai ke **C++ string aur C-style string alag alag hote hain**.  
Kuch functions (jaise printf, strlen, file functions) sirf **C-style string** accept karte hain, na ke C++ string.

Is liye c\_str() banaya gaya.  
👉 Ye **C++ string ko C-style string mai convert karta hai**.

**Simple Example Samajh**

#include <iostream>

using namespace std;

int main() {

string name = "Ali";

cout << name; // direct print ho gaya ✅

cout << name.c\_str(); // ye bhi print karega ✅

return 0;

}

Dono ka output same hai → **Ali**  
Farq sirf itna hai:

* name ek **C++ string** hai
* name.c\_str() ek **C-style string** ban jata hai

**Jab zaroorat parti hai**

Example: printf() function (jo C language ka hai).

#include <iostream>

#include <cstdio>

using namespace std;

int main() {

string name = "Ali";

// printf direct string accept nahi karta

// printf("Name: %s", name); ❌ Error

printf("Name: %s", name.c\_str()); // ✅ Correct

return 0;

}

👉 Yahaan .c\_str() use karke humne C++ string ko aisi form mai convert kia jo printf samajhta hai.

**Real Life Example**

Socho tumhara ek **modern mobile charger** hai jo USB-C pe chalta hai.  
Lekin tumhari purani device sirf **micro-USB** support karti hai.  
Tumhe ek **converter** chahiye hoga na?

⚡ Waise hi:

* string = USB-C (modern)
* c\_str() = converter
* char\* = micro-USB (purana system)

✅ **Summary (Easy):**

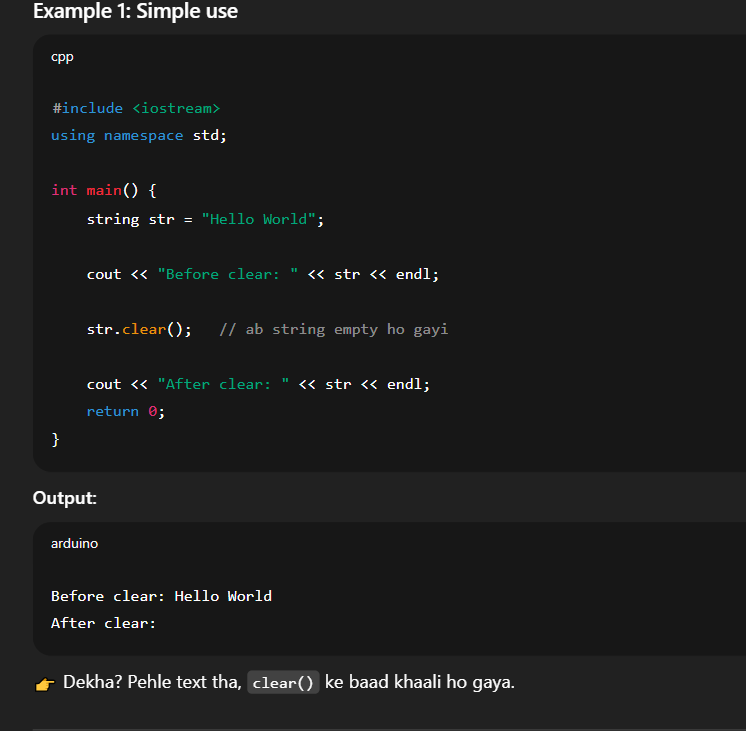
* c\_str() ek converter hai.
* Ye C++ string ko purane C-style string mai badal deta hai.
* Tab use hota hai jab koi purana function string ko samajhta nahi.

**clear() Function in C++ (String ke liye)**

👉 **clear() function ek string ko empty (khaali) kar deta hai.**  
Matlab string ke andar jo bhi text hai sab delete ho jata hai aur string bilkul blank ho jati hai.

**Maqsad (Purpose)**

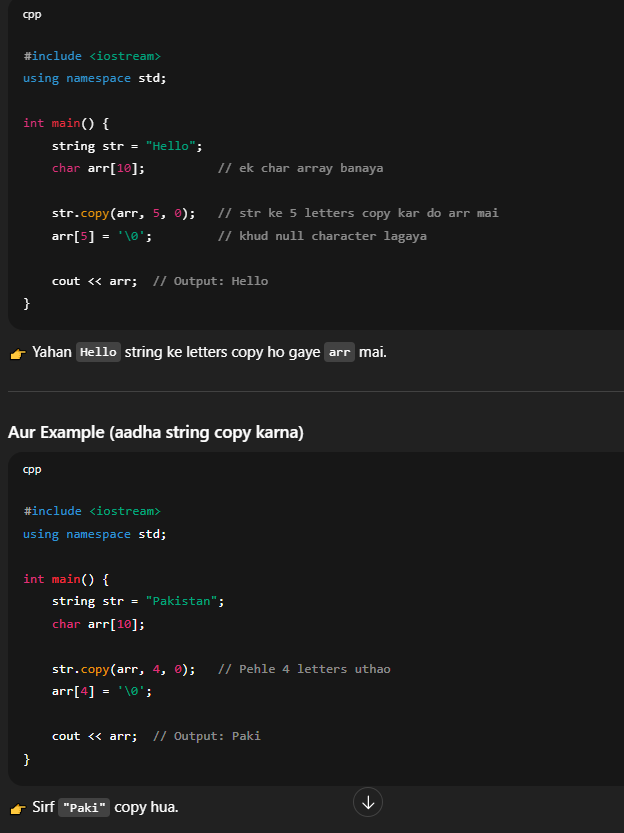
* String ke andar ka data remove karna.
* Nayi value dalne se pehle string ko reset karna.
* Memory free karna (zyada use ke cases mai).





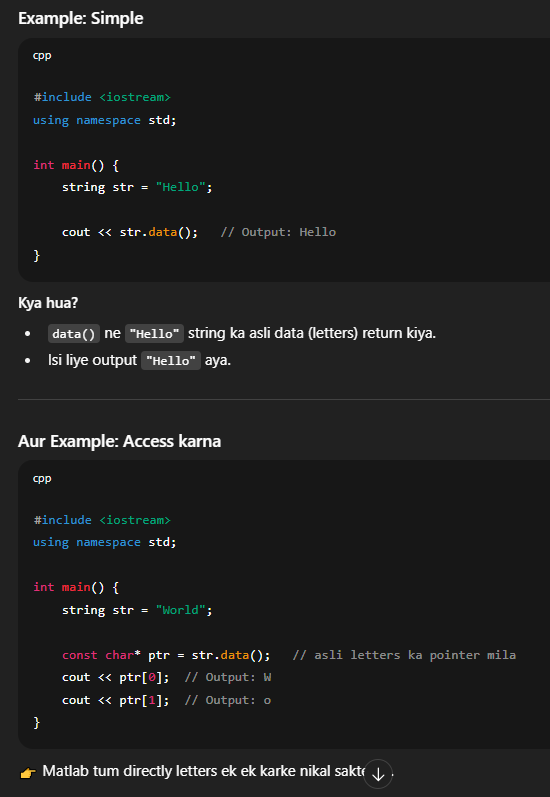
**copy()**

👉 copy() ka kaam hai ek **C++ string ke letters ko dusre character array (char[]) mai daalna**.  
Yani jo likha hai string mai, usko copy karke ek aur jagah bhejna.

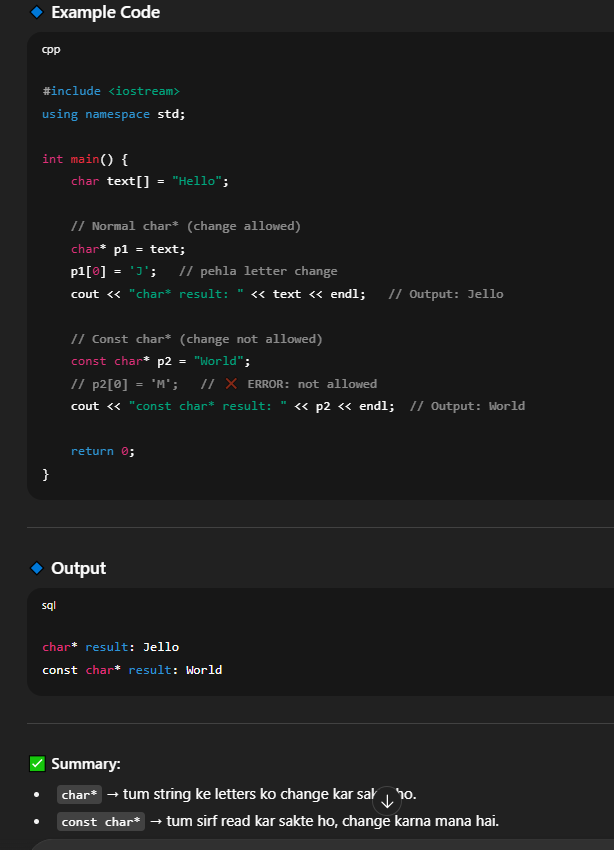


data()

👉 data() ek **shortcut** hai jo tumhe string ke andar ka **asli letters (char array)** de deta hai.  
Yani directly string ke letters ko point kar deta hai.



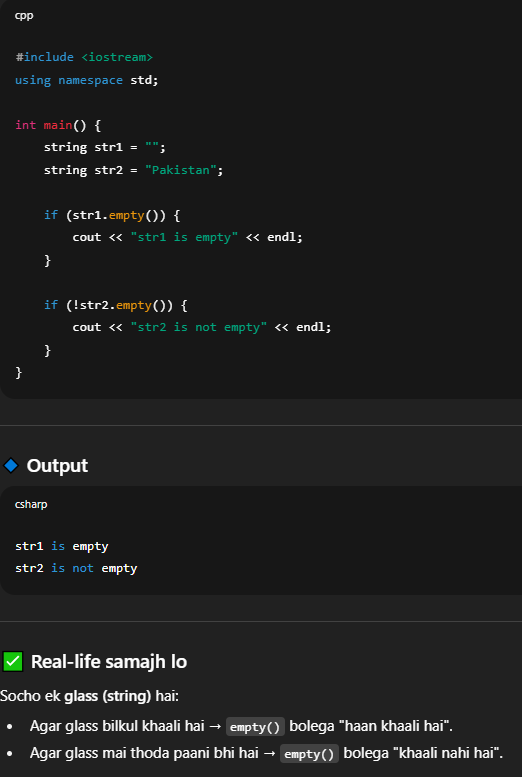
char\* aur const char\* ka fark



**empty() ka maqsad**

👉 Ye check karta hai ke **string khaali (empty)** hai ya nahi.

* Agar string khaali hai → true (1) return karega.
* Agar string mai kuch text hai → false (0) return karega.



**end() ka maqsad**

👉 end() ek **iterator** return karta hai jo **string ke last character ke baad** wali jagah ko point karta hai.

⚡ Important:

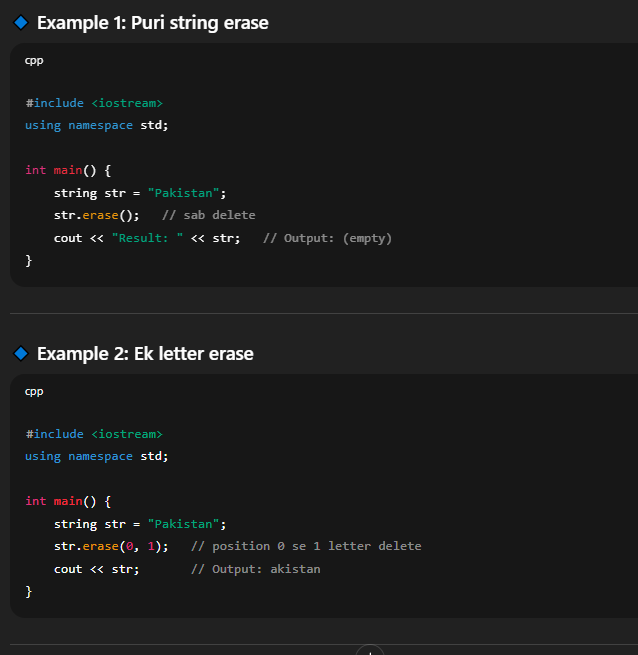
* end() **last letter ko point nahi karta**.
* Ye **last ke aglay (one-past-the-end)** position ko point karta hai.

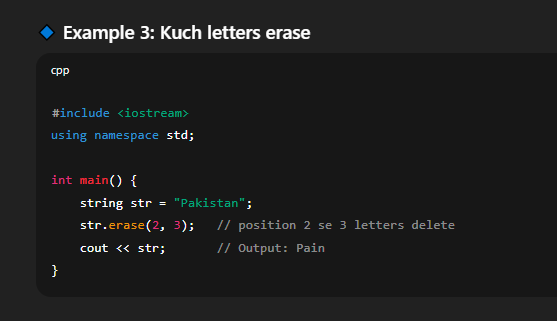


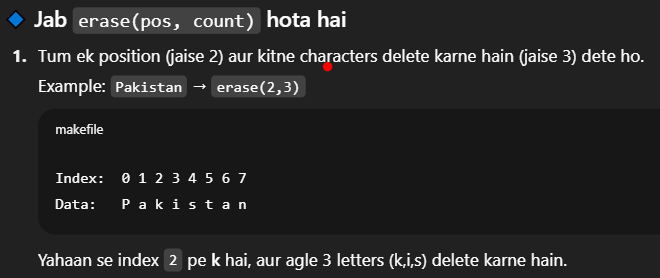
**erase() ka maqsad**

👉 Ye string ke andar se **characters delete (mitata)** hai.  
Tum chaho to:

1. **poori string khaali** kar do.
2. **sirf ek letter** hata do.
3. **kuch letters ka group** hata do.



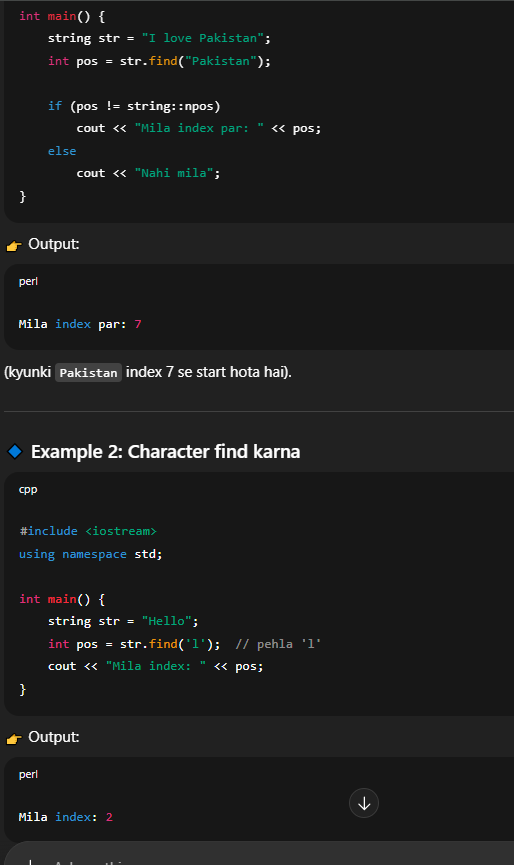


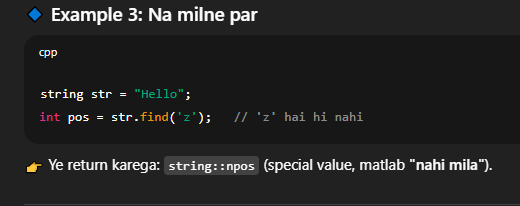


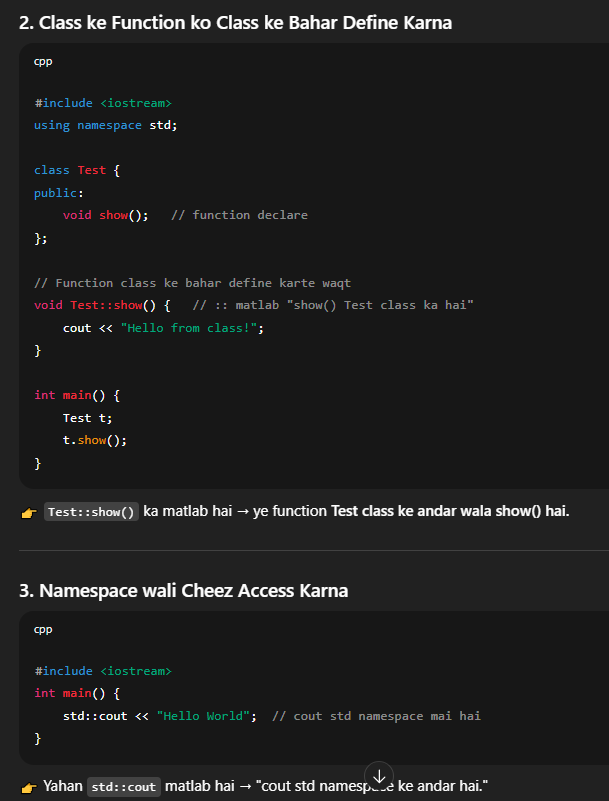
**🔹 find() ka maqsad**

👉 Ye string ke andar **kisi word ya character ko search (dhoondhne)** ke liye use hota hai.

* Agar mil jaye → to uska **index number (position)** return karta hai.
* Agar na mile → to **string::npos** return karta hai (iska matlab hai "nahi mila").

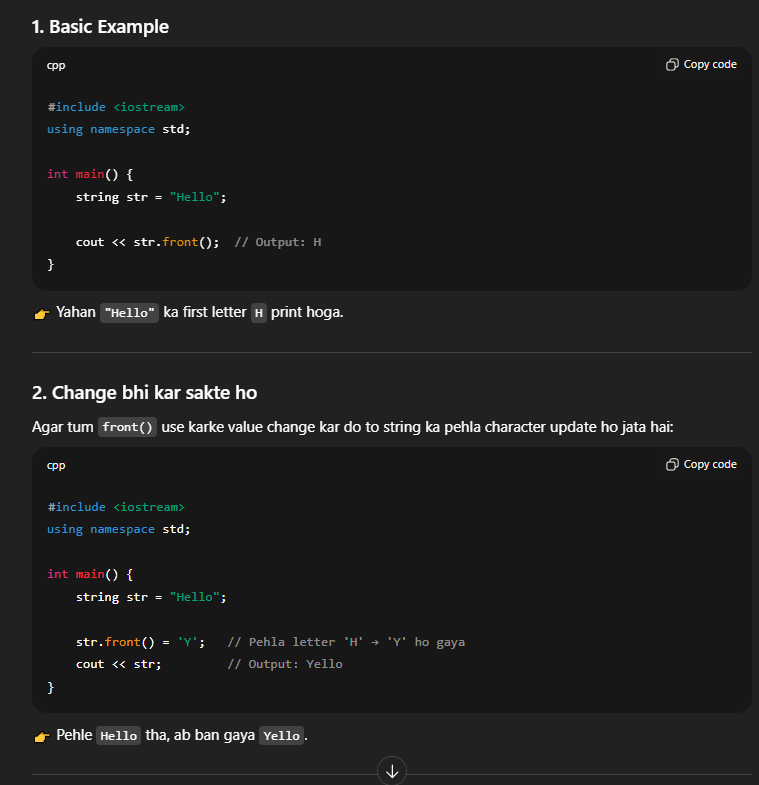






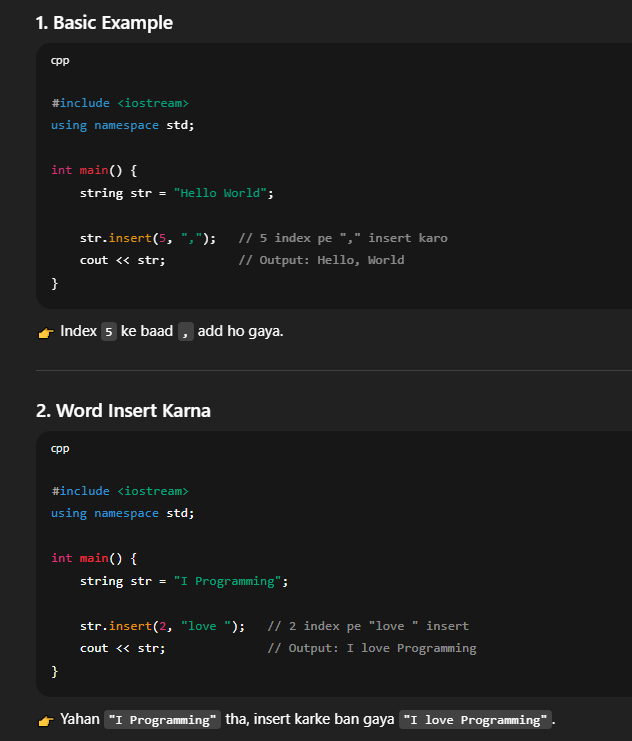
**front() ka Maqsad**

👉 front() ka kaam hota hai **string ka sabse pehla character return karna**.



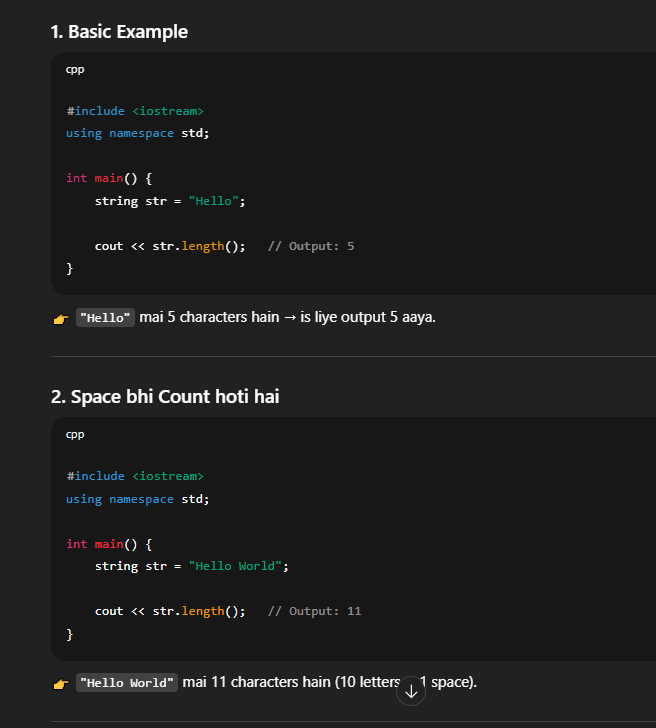
**insert() ka Maqsad**

👉 insert() ka kaam hota hai string ke andar **naya character ya naya word beech mai daalna** (jis position pe tum bologe).



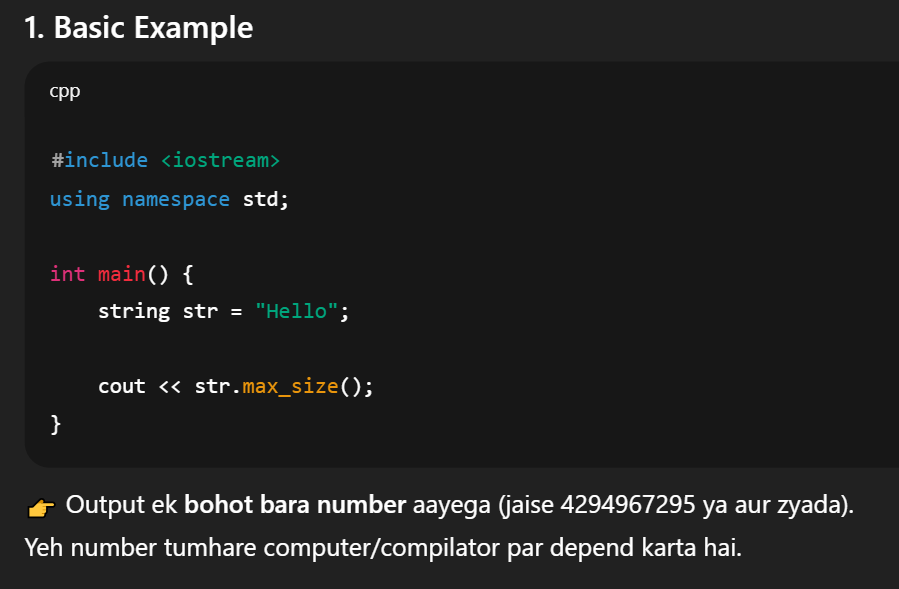
**length() ka Maqsad**

👉 length() ka kaam hai string ke **characters ki total ginti (count)** batana.



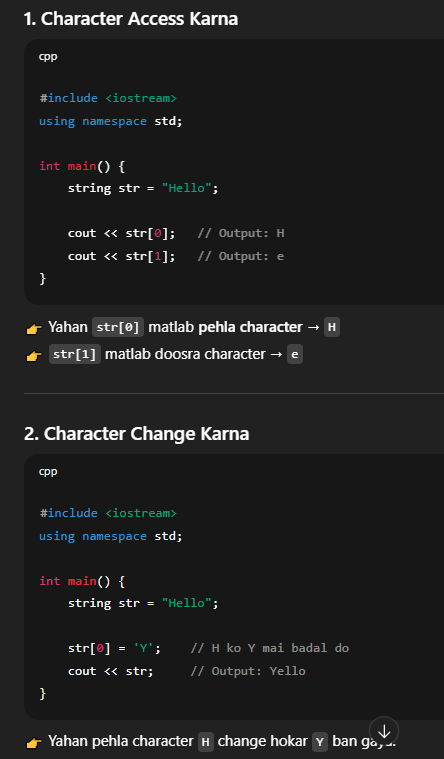
**max\_size() ka Maqsad**

👉 max\_size() ka kaam hota hai yeh batana ke **string theoretically kitne maximum characters store kar sakti hai** (computer ki memory par depend karta hai).



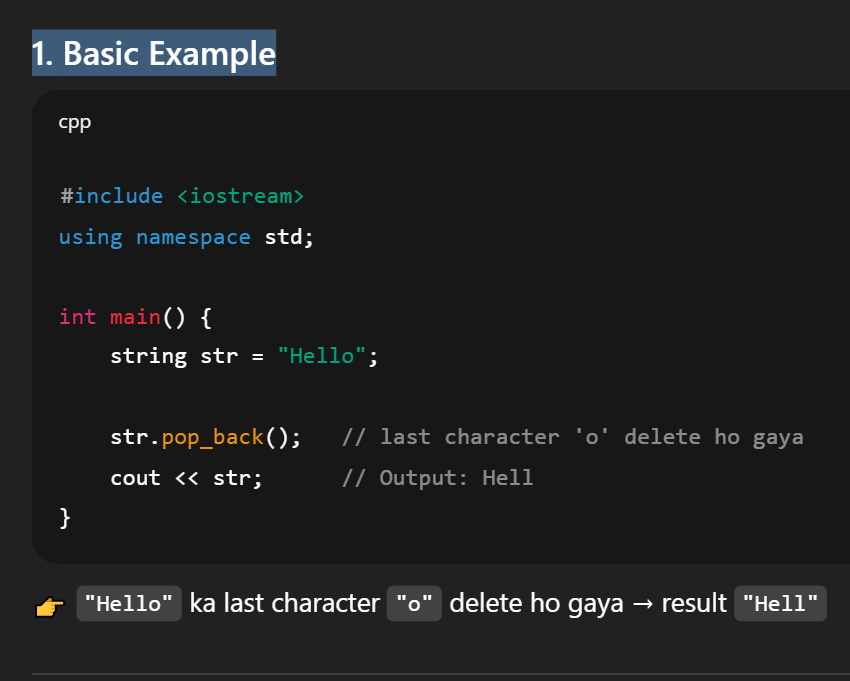
**operator[] ka Maqsad**

👉 operator[] ka kaam hai **string ke andar kisi specific position (index) ka character nikalna ya change karna**.



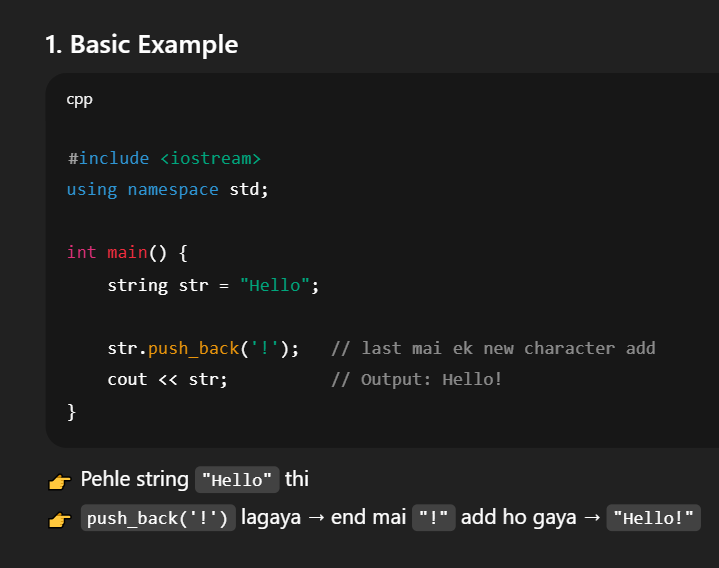
**🔹 pop\_back() ka Maqsad**

👉 pop\_back() ka kaam hai **string ka last (aakhri) character delete karna**.



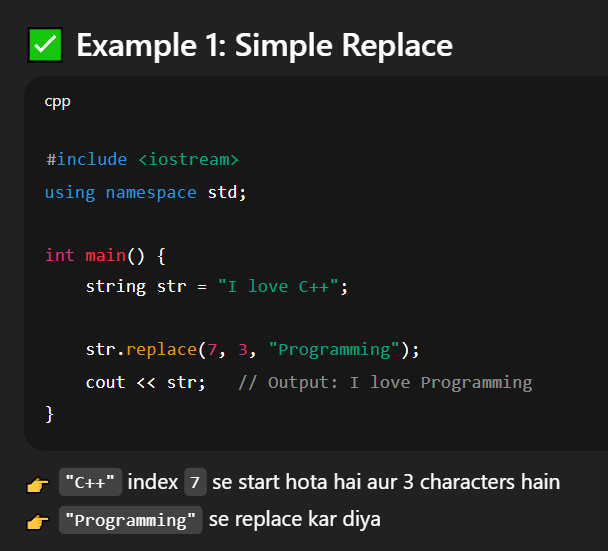
**push\_back() ka Maqsad**

👉 push\_back() ka kaam hota hai **string ke end (last mai) ek naya character add karna**.



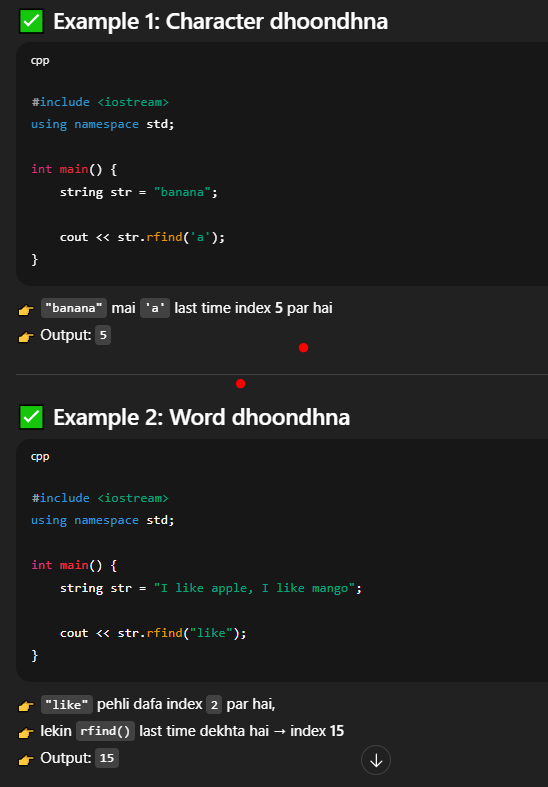
**🔹 replace() ka Maqsad**

👉 replace() ka kaam hai **string ke andar kisi hissa (part) ko nikal kar naya text dalna**.  
Yani ek jagah se kuch characters **replace** (tabdeel) karna.



**🔹 rfind() ka Maqsad**

👉 rfind() ka kaam hai **string ke andar kisi word/character ko dhoondna (search karna) lekin right side (aakhri se) se start karke**.  
Yani yeh **last occurrence** ka index deta hai.

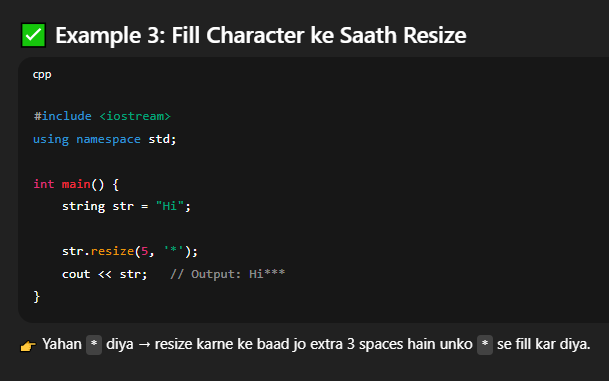


**🔹 resize() ka Maqsad**

👉 resize() ka kaam hai **string ki length (size) ko change karna**.

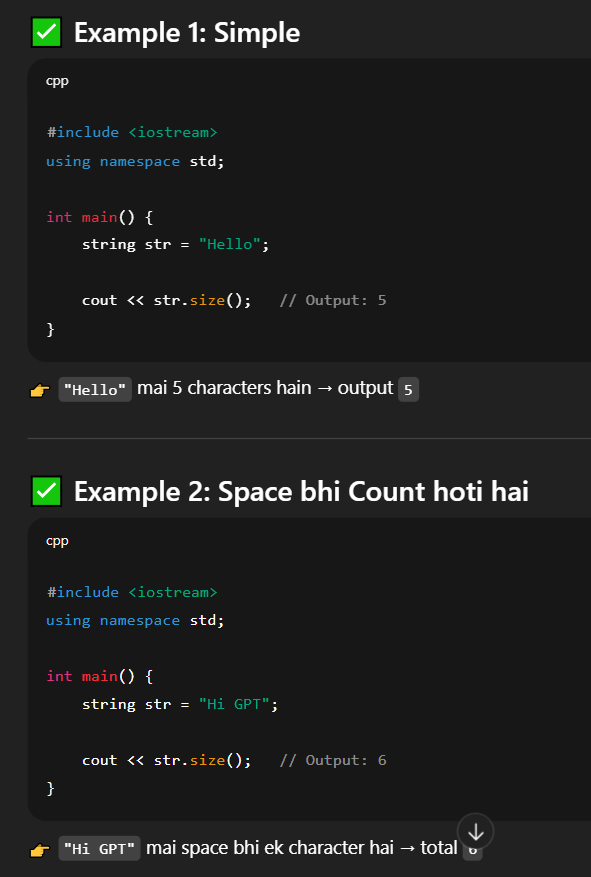
* Agar **naya size chhota** kar diya → string cut ho jati hai.
* Agar **naya size bada** kar diya → string ke aakhir mai extra characters (default '\\0' ya tum jo specify karo) add ho jate hain.

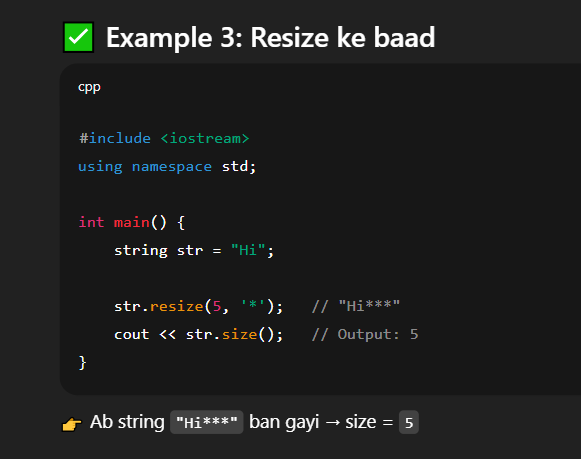




**🔹 size() ka Maqsad**

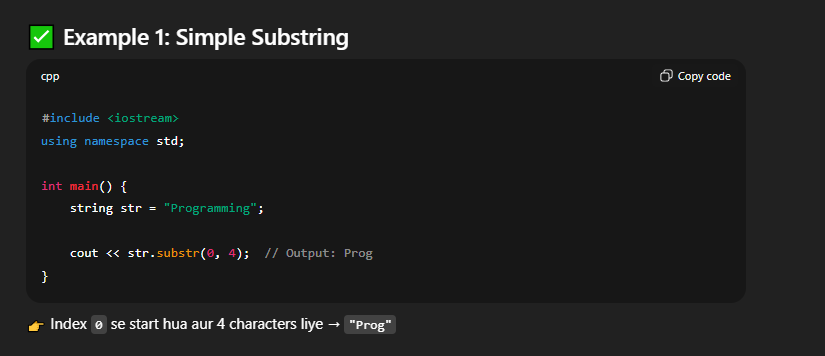
👉 size() string mai kitne **characters** hain, ye count karke return karta hai.  
👉 Matlab string ki **length** batata hai.

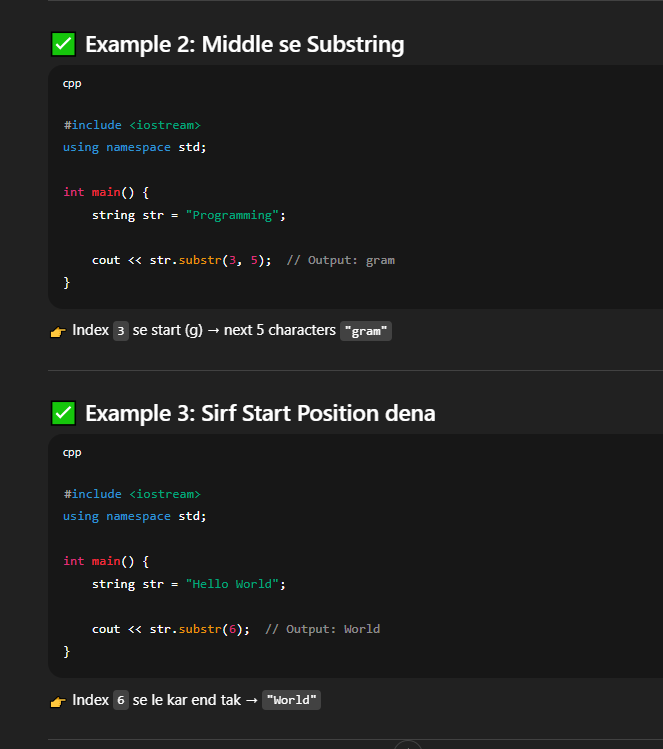




**🔹 substr() ka Maqsad**

👉 substr() ka matlab hota hai **string ka chhota hissa (substring) nikalna**.  
👉 Tum starting position aur kitne characters chahiye wo decide karte ho.





**🔹 swap() ka Maqsad**

👉 swap() do strings ki **values ko exchange (badal)** deta hai.  
👉 Matlab jo pehla string mai likha hai wo dusre mai chala jata hai aur jo dusre mai tha wo pehle mai aa jata hai.

