

C++ mutable keyword

Be self controlled and growls
around like a roaring lion looking for someone to devour.
1 Peter 5:8 (Bible)

[Hire with us!](#)

The mutable storage class specifier in C++ (or use of mutable keyword in C++)

auto, register, static and extern are the storage class specifiers in C. typedef is also considered as a storage class specifier in C. C++ also supports all these storage class specifiers. In addition to this C++, adds one important storage class specifier whose name is mutable.

What is the need of mutable?

Sometimes there is requirement to modify one or more data members of class / struct through const function even though you don't want the function to update other members of class / struct. This task can be easily performed by using mutable keyword. Consider this example where use of mutable can be useful. Suppose you go to hotel and you give the order to waiter to bring some food dish. After giving order, you suddenly decide to change the order of food. Assume that hotel provides facility to change the ordered food and again take the order of new food within 10 minutes after giving the 1st order. After 10 minutes order can't be cancelled and old order can't be replaced by new order. See the following code for details.

```
#include <iostream>
#include <string.h>
using std::cout;
using std::endl;

class Customer
{
    char name[25];
    mutable char placedorder[50];
    int tableno;
    mutable int bill;
public:
    Customer(char* s, char* m, int a, int p)
    {
        strcpy(name, s);
        strcpy(placedorder, m);
        tableno = a;
        bill = p;
    }
    void changePlacedOrder(char* p) const
    {
        strcpy(placedorder, p);
    }
}
```



```

    }
    void changeBill(int s) const
    {
        bill = s;
    }
    void display() const
    {
        cout << "Customer name is: " << name << endl;
        cout << "Food ordered by customer is: " << placedorder << endl;
        cout << "table no is: " << tableno << endl;
        cout << "Total payable amount: " << bill << endl;
    }
};

int main()
{
    const Customer c1("Pravasi Meet", "Ice Cream", 3, 100);
    c1.display();
    c1.changePlacedOrder("GulabJammuns");
    c1.changeBill(150);
    c1.display();
    return 0;
}

```

Output:

```

Customer name is: Pravasi Meet
Food ordered by customer is: Ice Cream
table no is: 3
Total payable amount: 100
Customer name is: Pravasi Meet
Food ordered by customer is: GulabJammuns
table no is: 3
Total payable amount: 150

```

Closely observe the output of above program. The values of *placedorder* and *bill* data members are changed from const function because they are declared as mutable.

The keyword mutable is mainly used to allow a particular data member of const object to be modified. **When we declare a function as const, the this pointer passed to function becomes const.** Adding mutable to a variable allows a const pointer to change members.

mutable is particularly useful if most of the members should be constant but a few need to be updateable. Data members declared as mutable can be modified even though they are the part of object declared as const. You cannot use the mutable specifier with names declared as static or const, or reference.

As an **exercise** predict the output of following two programs.

```

// PROGRAM 1
#include <iostream>
using std::cout;

class Test {

```



```
public:
    int x;
    mutable int y;
    Test() { x = 4; y = 10; }
};
int main()
{
    const Test t1;
    t1.y = 20;
    cout << t1.y;
    return 0;
}
```

```
// PROGRAM 2
#include <iostream>
using std::cout;

class Test {
public:
    int x;
    mutable int y;
    Test() { x = 4; y = 10; }
};
int main()
{
    const Test t1;
    t1.x = 8;
    cout << t1.x;
    return 0;
}
```

Source:

The mutable storage class specifier (C++ only)

This article is contributed **Meet Pravasi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

[Static Keyword in C++](#)

[Use of explicit keyword in C++](#)

[override Keyword in C++](#)

[restrict keyword in C](#)

[_Generic keyword in C](#)

[C++ | Static Keyword | Question 6](#)

[C++ | const keyword | Question 1](#)

[C++ | const keyword | Question 2](#)

[C++ | const keyword | Question 3](#)

[C++ | Static Keyword | Question 5](#)



[C++ | const keyword | Question 5](#)

[C++ | const keyword | Question 5](#)

[Understanding "extern" keyword in C](#)

[Understanding "register" keyword in C](#)

[C++ | Static Keyword | Question 4](#)

Article Tags : [C](#) [C++](#) [cpp-storage-classes](#)

Practice Tags : [C](#) [CPP](#)



5

2.3

☐ To-do ☐ Done

Based on **6** vote(s)

[Feedback/ Suggest Improvement](#)

[Add Notes](#)

[Improve Article](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)



A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

[About Us](#)
[Careers](#)
[Privacy Policy](#)
[Contact Us](#)

PRACTICE

[Courses](#)
[Company-wise](#)
[Topic-wise](#)
[How to begin?](#)

LEARN

[Algorithms](#)
[Data Structures](#)
[Languages](#)
[CS Subjects](#)
[Video Tutorials](#)

CONTRIBUTE

[Write an Article](#)
[Write Interview Experience](#)
[Internships](#)
[Videos](#)

@geeksforgeeks, Some rights reserved

