Table of Contents

Use of explicit keyword in C++

Difficulty Level : Medium ● Last Updated : 02 May, 2022

Explicit Keyword in C++ is used to mark constructors to not implicitly convert types in C++. It is optional for constructors that take exactly one argument and work on constructors(with a single argument) since those are the only constructors that can be used in typecasting.

Let's understand explicit keyword through an example.

Predict the output of the following C++ Program

CPP

```cpp
// C++ program to illustrate default
// constructor without 'explicit'
// keyword
#include <iostream>
using namespace std;

class Complex {
private:
    double real;
    double imag;

public:

    // Default constructor
    Complex(double r = 0.0,
            double i = 0.0) : real(r),
                              imag(i)
    {
    }

    // A method to compare two
    // Complex numbers
    bool operator == (Complex rhs)
    {
        return (real == rhs.real &&
                imag == rhs.imag);
    }
};

// Driver Code
int main()
{
    // a Complex object
    Complex com1(3.0, 0.0);

    if (com1 == 3.0)
        cout << "Same";
    else
        cout << "Not Same";
    return 0;
}
```

Output

```
Same
```

As discussed in this article, in C++, if a class has a constructor which can be called with a single argument, then this constructor becomes a conversion constructor because such a constructor allows conversion of the single argument to the class being constructed.
We can avoid such implicit conversions as these may lead to unexpected results. We can make the constructor explicit with the help of an **explicit keyword**. For example, if we try the following program that uses explicit keywords with a constructor, we get a compilation error.

CPP

```cpp
// C++ program to illustrate
// default constructor with
// 'explicit' keyword
#include <iostream>
using namespace std;

class Complex {
private:
    double real;
    double imag;

public:
    // Default constructor
    explicit Complex(double r = 0.0,
                     double i = 0.0) :
                     real(r), imag(i)
    {
    }

    // A method to compare two
    // Complex numbers
    bool operator == (Complex rhs)
    {
        return (real == rhs.real &&
                imag == rhs.imag);
    }
};

// Driver Code
int main()
{
    // a Complex object
    Complex com1(3.0, 0.0);

    if (com1 == 3.0)
        cout << "Same";
    else
        cout << "Not Same";
    return 0;
}
```

Output

```
Compiler Error : no match for 'operator==' in 'com1 == 3.0e+0'
```

We can still typecast the double values to Complex, but now we have to explicitly typecast it. For example, the following program works fine.

CPP

```cpp
// C++ program to illustrate
// default constructor with
// 'explicit' keyword
#include <iostream>
using namespace std;

class Complex {
private:
    double real;
    double imag;

public:

    // Default constructor
    explicit Complex(double r = 0.0,
                     double i = 0.0):
                     real(r) , imag(i)
    {
    }

    // A method to compare two
    // Complex numbers
    bool operator == (Complex rhs)
    {
        return (real == rhs.real &&
                imag == rhs.imag);
    }
};

// Driver Code
int main()
{
    // a Complex object
    Complex com1(3.0, 0.0);

    if (com1 == (Complex)3.0)
        cout << "Same";
    else
        cout << "Not Same";
    return 0;
}
```

Output

```
Same
```

Note: *The explicit specifier can be used with a constant expression. However, if that constant expression evaluates to true, then only the function is explicit.*

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

RECOMMENDED ARTICLES

Page : 1 2 3

01 Understanding "extern" keyword in C
19, Jul 09

02 Understanding "register" keyword in C
20, Feb 10

03 Comparison of static keyword in C++ and Java
29, Nov 12

04 Function overloading and const keyword
02, Oct 12

05 C++ mutable keyword
06, Jul 14

06 C++ | Static Keyword | Question 1
17, Aug 13

07 C++ | Static Keyword | Question 2
17, Aug 13

08 C++ | friend keyword | Question 1
22, Aug 13

Article Contributed By :

GeeksforGeeks

Improved By : anshikajain26, jayeshtpatel

Article Tags : C Language, C++

Practice Tags : CPP

Improve Article Report Issue

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

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

Load Comments