Increment (++) and Decrement (–) Operator Overloading in C++

Difficulty Level : Medium • Last Updated : 16 Nov, 2022

Operator overloading is a feature in object-oriented programming which allows a programmer to redefine a built-in operator to work with user-defined data types.

Why Operator Overloading?

Let's say we have defined a class Integer for handling operations on integers. We can have functions add(), subtract(), multiply() and divide() for handling the respective operations. However, to make the code more intuitive and enhance readability, it is preferred to use operators that correspond to the given operations(+, -, *, / respectively) i.e. we can replace the following code.
Example:

```
Replace
i5 = divide(add(i1, i2), subtract(i3, i4))

by a simpler code:
i5 = (i1 + i2) / (i3 - i4)
```

Overloading the Increment Operator

The operator symbol for both prefix(++i) and postfix(i++) are the same. Hence, we need two different function definitions to distinguish between version.

Here is the code to demonstrate the same.

Example: Pre-increment overloading

CPP

```
// C++ program to demonstrate
// prefix increment operator overloading

#include <bits/stdc++.h>
using namespace std;

class Integer {
private:
    int i;

public:
    // Parameterised constructor
    Integer(int i = 0)
    {
        this->i = i;
    }

    // Overloading the prefix operator
    Integer& operator++()
    {
        ++i;
        // returned value should be a reference to *this
        return *this;
    }

    // Function to display the value of i
    }
};

// Driver function
int main()
{
    Integer i1(3);

    cout << "Before increment: ";
    i1.display();

    // Using the pre-increment operator
    Integer i2 = ++i1;

    cout << "After pre increment: " << endl;
    cout << "i1: ";
    i1.display();
    cout << "i2: ";
    i2.display();
}
```

Output

```
Before increment: i = 3
After post decrement:
i1: i = 4
i2: i = 4
```

Example: Post-Increment Overloading

CPP

```
// C++ program to demonstrate
// postfix increment operator
// overloading
#include <bits/stdc++.h>
using namespace std;

class Integer {
private:
    int i;

public:
    // Parameterised constructor
    Integer(int i = 0)
    // Overloading the postfix operator
    Integer operator++(int)
    {
        // returned value should be a copy of the object before increment
        Integer temp = *this;
        ++i;
        return temp;
    }

    // Function to display the value of i
    void display()
    {
        cout << "i = " << i << endl;
    }
};

// Driver function
int main()
{
    Integer i1(3);

    cout << "Before increment: ";
    i1.display();

    // Using the post-increment operator
    Integer i2 = i1++;

    cout << "After post increment: " << endl;
    cout << "i1: ";
    i1.display();
    cout << "i2: ";
    i2.display();
}
```

Output

```
Before increment: i = 3
After post increment:
i1: i = 4
i2: i = 3
```

Overloading the Decrement Operator

Similarly, we can also overload the decrement operator as follows:

CPP

```cpp
// C++ program to demonstrate
// prefix decrement operator
// overloading

#include <bits/stdc++.h>
using namespace std;

class Integer {
private:
    int i;

public:
    // Parameterised constructor
    Integer(int i = 0)
    {
        this->i = i;
    }

    // Overloading the prefix operator
    Integer& operator--()
    {
        --i;
        // returned value should be a reference to *this
        return *this;
    }

    // Function to display the value of i
    void display()
    {
        cout << "i = " << i << endl;
    }
};

// Driver function
int main()
{
    Integer i1(3);

    cout << "Before decrement: ";
    i1.display();

    // Using the pre-decrement operator
    Integer i2 = --i1;

    cout << "After pre decrement: " << endl;
    cout << "i1: ";
}
```

Output

```
Before decrement: i = 3
After pre decrement:
i1: i = 2
i2: i = 2
```

Example: Post-Decrement Overloading

CPP

```cpp
// C++ program to demonstrate
// postfix decrement operator
// overloading
#include <bits/stdc++.h>
using namespace std;

class Integer {
private:
    int i;

public:
    // Parameterised constructor
    Integer(int i = 0)
    {
        this->i = i;
    }

    // Overloading the postfix operator
    Integer operator--(int)
    {
        // returned value should be a copy of the object before decrement
        Integer temp = *this;
        --i;
        return temp;
    }

    // Function to display the value of i
    void display()
    {
        cout << "i = " << i << endl;
    }
};
```

```
{
    Integer i1(3);

    cout << "Before decrement: ";
    i1.display();

    // Using the post-decrement operator
    Integer i2 = i1--;

    cout << "After post decrement: " << endl;
    cout << "i1: ";
    i1.display();
    cout << "i2: ";
    i2.display();
}
```

Output

```
Before decrement: i = 3
After post decrement:
i1: i = 2
i2: i = 3
```

Related Articles

1. Operator Overloading '<<' and '>>' operator in a linked list class
2. C++ Increment and Decrement Operators
3. Pre-increment (or pre-decrement) With Reference to L-value in C++
4. Increment (Decrement) operators require L-value Expression
5. Pre-increment and Post-increment in C/C++
6. Overloading New and Delete operator in c++
7. Rules for operator overloading
8. Overloading Subscript or array index operator [] in C++

9. C++ Program to concatenate two strings using Operator Overloading

10. Operator overloading in C++ to print contents of vector, map, pair, ..

Previous

Types of Operator Overloading in C++

Next

Pre-increment and Post-increment in C/C++

Article Contributed By :

Abhishek De
@Abhishek De

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : kamkaz, thotasravya28, sackshamsharmaintern, lakshmisrinivas365

Article Tags : cpp-operator, cpp-operator-overloading, C++

Practice Tags : CPP, cpp-operator

Report Issue

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Interview Corner

Company Preparation

Preparation for SDE

Company Interview Corner

Experienced Interview

Internship Interview

Competitive Programming

Aptitude

Python

Python Tutorial

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC/SSC/BANKING

SSC CGL Syllabus

SBI PO Syllabus

IBPS PO Syllabus

UPSC Ethics Notes

UPSC Economics Notes

UPSC History Notes