Overloading the << Operator for Your Own Classes

11/04/2016 • 2 minutes to read •

In this article

Output streams use the insertion (`<<`) operator for standard types. You can also overload the `<<` operator for your own classes.

Example

The `write` function example showed the use of a `Date` structure. A date is an ideal candidate for a C++ class in which the data members (month, day, and year) are hidden from view. An output stream is the logical destination for displaying such a structure. This code displays a date using the `cout` object:

C++ Copy

```
Date dt(1, 2, 92);

cout <<dt;
```

To get `cout` to accept a `Date` object after the insertion operator, overload the insertion operator to recognize an `ostream` object on the left and a `Date` on the right. The overloaded `<<` operator function must then be declared as a friend of class `Date` so it can access the private data within a `Date` object.

C++ Copy

```
// overload_date.cpp
// compile with: /EHsc
#include <iostream>
using namespace std;

class Date
{
    int mo, da, yr;
```

```
public:
    Date(int m, int d, int y)
    {
        mo = m; da = d; yr = y;
    }
    friend ostream& operator<<(ostream& os, const Date& dt);
};

ostream& operator<<(ostream& os, const Date& dt)
{
    os << dt.mo << '/' << dt.da << '/' << dt.yr;
    return os;
}

int main()
{
    Date dt(5, 6, 92);
    cout << dt;
}
```

Output | Copy

```
5/6/92
```

Remarks

The overloaded operator returns a reference to the original `ostream` object, which means you can combine insertions:

C++ | Copy

```
cout <<"The date is" <<dt <<flush;
```

See also

[Output Streams]

Is this page helpful?

Yes No