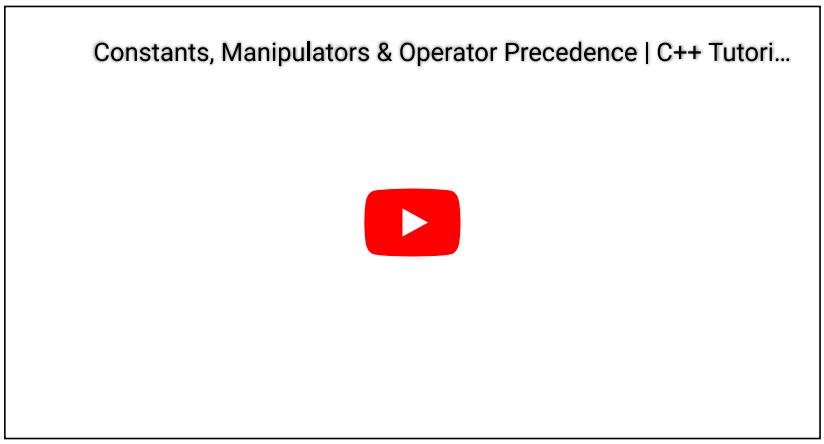

Constants, Manipulators & Operator Precedence | C++ Tutorials for Beginners #8

In this series of our C++ tutorials, we will visualize the constants, manipulator, and operator precedence in C++ language in this lecture. In our last lesson, we discussed the reference variable and typecasting in C++. If you haven't read the previous tutorial, click (**ADD THE LINK OF the PREVIOUS LECTURE**).

In this C++ tutorial, the topics which we are going to cover today are given below:

- **Constants in C++**
- **Manipulator in C++**
- **Operator Precedence in C++**

Constants in C++

Constants are unchangeable; when a constant variable is initialized in a program, its value cannot be changed afterwards. An example program for constants is shown in figure 1.

```
9     // Constants in C++
10    const float a = 3.11;
11    cout<<"The value of a was: "<<a<<endl;
12    a = 45.6;
13    cout<<"The value of a is: "<<a<<endl;
14    return 0;
15 }
```

Figure 1: Constants in C++

As shown in figure 2, a constant float variable "a" is initialized with a value "3.11" but when we tried to update the value of "a" with a value of "45.6" the compiler throw us an error that the constant variable is being reassigned a value. An error message can be seen in figure 2.

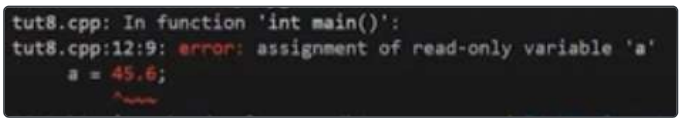

Figure 2: Constant Program Error

Manipulator

In C++ programming, language manipulators are used in the formatting of output. The two most commonly used manipulators are: "**endl"** and "**setw"**.

- "**endl"** is used for the next line.
- "**setw"** is used to specify the width of the output.

An example program to show the working of a manipulator is shown in figure 3.

```
int a =3, b=78, c=1233;
cout<<"The value of a without setw is: "<<a<<endl;
cout<<"The value of b without setw is: "<<b<<endl;
cout<<"The value of c without setw is: "<<c<<endl;

cout<<"The value of a is: "<<setw(4)<<a<<endl;
cout<<"The value of b is: "<<setw(4)<<b<<endl;
cout<<"The value of c is: "<<setw(4)<<c<<endl;
return 0;
```

Figure 3: Manipulators in C++

As shown in figure 3, we have initialized three integer variables "**a, b, c"**. First, we printed all the three variables and used "**endl"** to print each variable in a new line. After that, we again printed the three variables and used "**setw(4)**," which will set there width to "**4"**. The output for the following program is shown in figure 4.

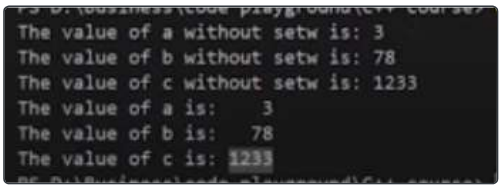

Figure 4: Manipulators Program Output

Operator Precedence & Operator Associativity

Operator precedence helps us to solve an expression. For example, in an expression "**int c = a*b+c"** the multiplication operator's precedence is higher than the precedence of addition operator, so the multiplication between "**a & b"** first and then addition will be performed.

Operator associativity helps us to solve an expression; when two or more operators have the same precedence, the operator associativity helps us to decide that we should solve the expression from "**left-to-right"** or from "**right-to-left"**.

Operator precedence and operator associativity can be seen from here. An example program for operator precedence and operator associativity is shown in figure 5.

```
// Operator Precedence
int a =3, b=4;
// int c = (a*5)+b;
int c = ((((a*5)+b)-45)+87);
cout<<c;
return 0;
```

Figure 5: Operator Precedence & Associativity Example program

As shown in figure 5, we initialized two integer variables and then wrote an expression "**int c = a*5+b;"** on which we have already discussed. Then we have written another expression "**int c = ((((a*5)+b)-45)+87);".** The precedence of multiply is higher than addition so the multiplication will be done first, but the precedence of addition and subtraction is same, so here we will check the associativity which is "**left-to-right"** so the addition is performed first and then subtraction is performed.

Code as described/written in the video

```
#include<iostream>
#include<iomanip>

using namespace std;

int main(){
    // int a = 34;
    // cout<<"The value of a was: "<<a;
    // a = 45;
    // cout<<"The value of a is: "<<a;
    // Constants in C++
    // const int a = 3;
    // cout<<"The value of a was: "<<a<<endl;
    // a = 45; // You will get an error because a is a constant
    // cout<<"The value of a is: "<<a<<endl;

    // Manipulators in C++
    // int a =3, b=78, c=1233;
    // cout<<"The value of a without setw is: "<<a<<endl;
    // cout<<"The value of b without setw is: "<<b<<endl;
    // cout<<"The value of c without setw is: "<<c<<endl;
```

← Previous

Next →