Pre-increment (or pre-decrement) With Reference to L-value in C++

Difficulty Level : Medium ● Last Updated : 22 Jun, 2022

Prerequisite: Pre-increment and post-increment in C/C++

In C++, pre-increment (or pre-decrement) can be used as l-value, but post-increment (or post-decrement) can not be used as l-value.
For example, following program prints *a = 20* (++a is used as l-value)

l-value is simply nothing but the memory location, which has an address.

CPP

```
// CPP program to illustrate
// Pre-increment (or pre-decrement)
#include <cstdio>

int main()
{
    int a = 10;

    ++a = 20; // works
    printf("a = %d", a);
    printf("\n");
    --a = 10;
    printf("a = %d", a);
    return 0;
}
```

Output:

```
a = 20
a = 10
```

Time Complexity: O(1)

The above program works whereas the following program fails in compilation with error *"non-lvalue in assignment"* (a++ is used as l-value)

CPP

```
// CPP program to illustrate
// Post-increment (or post-decrement)
#include <cstdio>

int main()
{
    int a = 10;
    a++ = 20; // error
    printf("a = %d", a);
    return 0;
}
```

Error:

```
prog.cpp: In function 'int main()':
prog.cpp:6:5: error: lvalue required as left operand of assignment
 a++ = 20; // error
     ^
```

How ++a is Different From a++ as lvalue?

It is because ++a returns an *lvalue*, which is basically a reference to the variable to which we can further assign — just like an ordinary variable. It could also be assigned to a reference as follows:

```
int &ref = ++a; // valid
int &ref = a++; // invalid
```

Whereas if you recall how a++ works, it doesn't immediately increment the value it holds. For clarity, you can think of it as getting incremented in the next statement. So what basically happens is that, a++ returns an *rvalue*, which is basically just a value like the value of an expression that is not stored. You can think of a++ = 20; as follows after being processed:

```
int a = 10;

// On compilation, a++ is replaced by the value of a which is an rvalue:
10 = 20; // Invalid

// Value of a is incremented
a = a + 1;
```

That should help to understand why a++ = 20; won't work. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

144

Related Articles

1. lvalue and rvalue in C language
2. Output of the program | Dereference, Reference, Dereference, Reference....
3. When do we pass arguments by reference or pointer?
4. Reference to a pointer in C++ with examples and applications
5. Passing Reference to a Pointer in C++
6. Difference between Call by Value and Call by Reference
7. Can C++ reference member be declared without being initialized with declaration?
8. Different ways to use Const with Reference to a Pointer in C++