Understanding nullptr in C++

Consider the following C++ program that shows problem with NULL (need of nullptr)

```
// C++ program to demontrate problem with NULL
#include <bits/stdc++.h>
using namespace std;

// function with integer argument
int fun(int N)   { cout << "fun(int)"; }

// Overloaded function with char pointer argument
int fun(char* s)  { cout << "fun(char *)"; }

int main()
{
    // Ideally, it should have called fun(char *),
    // but it causes compiler error.
    fun(NULL);
}
```

Output:

```
16:13: error: call of overloaded 'fun(NULL)' is ambiguous
     fun(NULL);
```

What is the problem with above program?

NULL is typically defined as (void *)0 and conversion of NULL to integral types is allowed. So the function call fun(NULL) becomes ambiguous.

```
// This program compiles (may produce warning)
#include<stdio.h>
int main()
{
   int x = NULL;
}
```

How does nullptr solve the problem?

In the above program, if we replace NULL with nullptr, we get the output as “fun(char *)”.

nullptr is a keyword that can be used at all places where NULL is expected. Like NULL, nullptr is implicitly convertible and comparable to any pointer type. **Unlike NULL, it is not implicitly convertible or comparable to integral types**.

```
// This program does NOT compile
#include<stdio.h>
int main()
{
   int x = nullptr;
}
```

Output:

```
Compiler Error
```

As a side note, **nullptr is convertible to bool.**

```
// This program compiles
#include<iostream>
using namespace std;

int main()
{
   int *ptr = nullptr;

   // Below line compiles
   if (ptr) { cout << "true"; }
   else { cout << "false"; }
}
```

Output:

```
false
```

There are some unspecified things when we compare two simple pointers but comparison between two values of type nullptr_t is specified as, comparison by <= and >= return true and comparison by < and > returns false and comparing any pointer type with nullptr by == and != returns true or false if it is null or non-null respectively.

```
// C++ program to show comparisons with nullptr
#include <bits/stdc++.h>
using namespace std;

// Driver program to test behavior of nullptr
int main()
{
    // creating two variables of nullptr_t type
    // i.e., with value equal to nullptr
    nullptr_t np1, np2;

    // <= and >= comparison always return true
    if (np1 >= np2)
        cout << "can compare" << endl;
    else
        cout << "can not compare" << endl;

    // Initialize a pointer with value equal to np1
    char *x = np1;  // same as x = nullptr (or x = NULL
                    // will also work)
    if (x == nullptr)
        cout << "x is null" << endl;
    else
        cout << "x is not null" << endl;

    return 0;
}
```

Output :

```
can compare
x is null
```

This article is contributed by Utkarsh Trivedi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Understanding static_assert in C++ 11

Understanding constexpr specifier in C++

Understanding "register" keyword in C

Understanding "extern" keyword in C

OpenCV | Understanding Brightness in an Image

OpenCV | Understanding Contrast in an Image

Understanding "volatile" qualifier in C | Set 2 (Examples)

Understanding "volatile" qualifier in C | Set 1 (Introduction)

Understanding Lvalues, PRvalues and Xvalues in C/C++ with Examples

Understanding ShellExecute function and it's application to open a list of URLs present in a file using C++ code

fill in C++ STL

Conditional or Ternary Operator (?:) in C/C++

forward_list insert_after() function in C++ STL

Count of distinct remainders when N is divided by all the numbers from the range [1, N]

Article Tags : C C++ cpp-pointer

Practice Tags : C CPP

👍
7

2.8

□ To-do □ Done

Based on **22** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

Load Comments

GeeksforGeeks
A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY
About Us
Careers
Privacy Policy
Contact Us

LEARN
Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE
Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE
Write an Article
Write Interview Experience
Internships
Videos