Understanding constexpr specifier in C++

constexpr is a feature added in C++ 11. The main idea is performance improvement of programs by doing computations at compile time rather than run time. Note that once a program is compiled and finalized be developer, it is run multiple times by users. The idea is to spend time in compilation and save time at run time (similar to template metaprogramming)

constexpr specifies that the value of an object or a function can be evaluated at compile time and the expression can be used in other constant expressions. For example, in below code product() is evaluated at compile time.

```
// constexpr function for product of two numbers.
// By specifying constexpr, we suggest compiler to
// to evaluate value at compiler time
constexpr int product(int x, int y)
{
    return (x * y);
}

int main()
{
    const int x = product(10, 20);
    cout << x;
    return 0;
}
```

Output :

```
200
```

A function be declared as constexpr

1. In C++ 11, a constexpr function should contain only one return statement. C++ 14 allows more than one statements.
2. constexpr function should refer only constant global variables.
3. constexpr function can call only other constexpr function not simple function.
4. Function should not be of void type and some operator like prefix increment (++v) are not allowed in constexpr function.

constexpr vs inline functions:

Both are for performance improvements, inline functions are request to compiler to expand at compile time and save time of function call overheads. In inline functions, expressions are always evaluated at run time. constexpr is different, here expressions are evaluated at compile time.

Example of performance improvement by constexpr:

Consider the following C++ program

```
// A C++ program to demonstrate use of constexpr
#include<iostream>
using namespace std;

constexpr long int fib(int n)
{
    return (n <= 1)? n : fib(n-1) + fib(n-2);
}

int main ()
{
    // value of res is computed at compile time.
    const long int res = fib(30);
    cout << res;
    return 0;
}
```

When the above program is run on GCC, it takes **0.003 seconds** (We can measure time using time command)

If we remove const from below line, then the value of fib(5) is not evaluated at compile time, because result of constexpr is not used in a const expression.

```
Change,
  const long int res = fib(30);
To,
  long int res = fib(30);
```

After making above change, time taken by program becomes higher **0.017 seconds**.

constexpr with constructors:

constexpr can be used in constructors and objects also. See this for all restrictions on constructors that can use constexpr.

```
//  C++ program to demonstrate uses of constexpr in constructor
#include <bits/stdc++.h>
using namespace std;

//  A class with constexpr constructor and function
class Rectangle
{
    int _h, _w;
public:
    // A constexpr constructor
    constexpr Rectangle (int h, int w) : _h(h), _w(w) {}

    constexpr int getArea ()  {   return _h * _w; }
};

//  driver program to test function
int main()
{
    // Below object is initialized at compile time
    constexpr Rectangle obj(10, 20);
    cout << obj.getArea();
    return 0;
}
```

Output :

```
200
```

constexpr vs const

They serve different purposes. constexpr is mainly for optimization while const is for practically const objects like value of Pi.

Both of them can be applied to member methods. Member methods are made const to make sure that there are no accidental changes by the method. On the other hand, the idea of using constexpr is to compute expressions at compile time so that time can be saved when code is run.

const can only be used with non-static member function whereas constexpr can be used with member and non-member functions, even with constructors but with condition that argument and return type must be of literal types.

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:

C++ final specifier

_Noreturn function specifier in C

Using a variable as format specifier in C

Difference between %d and %i format specifier in C language

Understanding static_assert in C++ 11

Understanding nullptr in C++

Understanding "extern" keyword in C

Understanding "register" keyword in C

OpenCV | Understanding Brightness in an Image

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

OpenCV | Understanding Contrast in an Image

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

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

Improved By : Shun Xian Cai, nidhi_biet

Article Tags : C C++ CPP-Library

Practice Tags : C CPP

5

1.8

To-do Done

Based on **5** vote(s)

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