

C++ 修饰符类型

C++ 允许在 **char**、**int** 和 **double** 数据类型前放置修饰符。修饰符用于改变基本类型的含义，所以它更能满足各种情境的需求。

下面列出了数据类型修饰符：

- signed
- unsigned
- long
- short

修饰符 **signed**、**unsigned**、**long** 和 **short** 可应用于整型，**signed** 和 **unsigned** 可应用于字符型，**long** 可应用于双精度型。

修饰符 **signed** 和 **unsigned** 也可以作为 **long** 或 **short** 修饰符的前缀。例如：**unsigned long int**。

C++ 允许使用速记符号来声明**无符号短整数**或**无符号长整数**。您可以不写 **int**，只写单词 **unsigned**、**short** 或 **unsigned long**，**int** 是隐含的。例如，下面的两个语句都声明了无符号整型变量。

```
unsigned x;  
unsigned int y;
```

为了解 C++ 解释有符号整数和无符号整数修饰符之间的差别，我们来运行一下下面这个短程序：

实例

```
#include <iostream>  
using namespace std;  
/*  
 * 这个程序演示了有符号整数和无符号整数之间的差别  
 */  
int main()  
{  
    short int i; // 有符号短整数  
    short unsigned int j; // 无符号短整数  
    j = 50000;  
    i = j;  
    cout << i << " " << j;  
    return 0;  
}
```

当上面的程序运行时，会输出下列结果：

```
-15536 50000
```

上述结果中，无符号短整数 50,000 的位模式被解释为有符号短整数 -15,536。


C++ 中的类型限定符

类型限定符提供了变量的额外信息。


限定符	含义
const	const 类型的对象在程序执行期间不能被修改改变。
volatile	修饰符 volatile 告诉编译器不需要优化volatile声明的变量，让程序可以直接从内存中读取变量。对于一般的变量编译器会对变量进行优化，将内存中的变量值放在寄存器中以加快读写效率。
restrict	由 restrict 修饰的指针是唯一一种访问它所指向的对象的方式。只有 C99 增加了新的类型限定符 restrict。

← C++ 常量

C++ 存储类 →



5 篇笔记

 写笔记