

# volatile

## C/C++中volatile关键字详解

```
1 // volatile.cpp : 定义控制台应用程序的入口点。
2 //
3 #include "stdafx.h"
4
5 int foo(int argc) {
6
7     return argc + 10;
8 }
9
10 //volatile 易变的
11 int main(int argc, char* argv[])
12 {
13     volatile int n = 0;
14
15     int* pN = const_cast<int*>(&n);
16     //int* pN = (int*)&n;
17     printf("pN %p\r\n", pN);
18     printf("argc %d\r\n", argc);
19     n = foo(argc);
20     printf("n %d\r\n", n);
21     int k = 0;
22     printf("n %d\r\n", n);
23
24     return 0;
25 }
26 /*
27 output:
28 pN 0019FF3C
29 argc 1
30 n 11
31 n 11
32 */
```

# C++作用域

## 全局域或命名空间域

## 局部作用域、块作用域

## 类域 class

## 数据隐藏

在不同的作用域可以定义多个相同名字的变量

在访问的时候，从内向外（块作用域 -> ... -> 全局作用域）

```
1  #include "stdafx.h"
2
3  int n = 100;
4  int main(int argc, char* argv[])
5  {
6      int n = 101;
7
8      printf("n is %d\r\n", n);
9
10     return 0;
11 }
12 /*
13 output: n is 101
14 */
```

## namespace 名字空间、命名空间

同一个名字空间可以在不同的文件（.cpp, .h）中拆开

## 写，实际是在同一个空间中

.cpp

```
1  #include "stdafx.h"
2  #include "test.h"
3
4  namespace CR32 {
5      int n = 32;
6  }
7
8  int main()
9  {
10     printf("n is %d\r\n", CR32::n);
11     printf("m is %d\r\n", CR32::m);
12
13     return 0;
14 }
15 /*
16 output:
17 n is 32
18 m is 2
19 */
```

test.h

```
1  #include "stdafx.h"
2
3  namespace CR32 {
4      int m = 2;
5  }
```

## 名字空间可以嵌套

```
1  #include "stdafx.h"
```

```

2
3 namespace CR32 {
4     int n = 32;
5
6     namespace test {
7         int n = 3;
8     }
9 }
10
11 int main()
12 {
13     printf("n is %d\r\n", CR32::n);
14     printf("-->n is %d\r\n", CR32::test::n);
15
16     return 0;
17 }
18 /*
19 output:
20 n is 32
21 -->n is 3
22 */

```

## 引用名字空间的方法

使用名字空间 `::` 变量名访问 `CR32::n` (建议用法)

声明名字空间 `using namespace CR32;`

```

1 #include "stdafx.h"
2 #include "test.h"
3
4 using namespace CR32;
5
6 int main()
7 {
8     int n = 0;
9
10    {

```

```

11     int n = 1;
12     printf("n is %d\r\n", n);
13 }
14
15 return 0;
16 }
17 /*
18 output:
19 n is 1
20
21 有几点要注意的问题，
22 1 test.h中名字空间的引入
23 2 using namespace CR32;的定义
24 3 main 函数中声明定义了两同名变量n，尽管使用了名字空间，但是给人还是
   一种混乱的感觉。
25 */

```

## test.h

```

1 #include "stdafx.h"
2
3 namespace CR32 {
4     int m = 2;
5 }

```

## 声明只使用名字空间的部分变量或函数 `using CR32::n`

```

1 #include "stdafx.h"
2 #include "test.h"
3
4 int main()
5 {
6     using CR32::m;
7     printf("m is %d\r\n", m);
8
9     return 0;
10 }

```

```
11  /*
12  output:
13  m is 2
14  */
```

```
1  //test.h
2  #include "stdafx.h"
3
4  namespace CR32 {
5      int m = 2;
6  }
```

**::n** 表示使用全局域中的 **n**

```
1  #include "stdafx.h"
2
3  int n = 6;
4
5  int main()
6  {
7      printf("::n is %d\r\n", ::n);
8
9      return 0;
10 }
11 /*
12 output:
13 ::n is 6
14 */
```

**名字空间取别名** **namespace c=CR32;**

```
1  #include "stdafx.h"
2  #include "test.h"
3
```

```

4 namespace c=CR32;
5
6 int main()
7 {
8     printf("CR32 alias c , m is %d\r\n", c::m);
9
10    return 0;
11 }
12 /*
13 output:
14 CR32 alias c , m is 2
15 */

```

```

1 // test.h
2 #include "stdafx.h"
3
4 namespace CR32 {
5     int m = 2;
6 }

```

**using** 语句可以出现在任何可以声明的地方（块作用域，全局域均可以），相当于对应的代码在指定的声明位置展开

```

1 #include "stdafx.h"
2
3 namespace CR32 {
4     int m = 2;
5 }
6
7 int main()
8 {
9     using CR32::m;
10
11    printf("m is %d\r\n", m);
12
13    return 0;

```

```
14 }
15 /*
16 output:
17 m is 2
18 */
```

## namespace 中的变量名或函数会在名称粉碎中添加名字空间域名

```
PS D:\CR32\第1阶段\第2门课 C++ 戚俊老师主讲\CR32\homework\3\classtest\scope\Debug> undname -f ?foo@CR32@@YAXXZ
Microsoft(R) Windows NT(R) Operating System
UNDNAME Version 5.00.1768.1 Copyright (C) Microsoft Corp. 1981-1998
>> ?foo@CR32@@YAXXZ == void __cdecl CR32::foo(void)
```