volatile

C/C++中volatile关键字详解

```
// volatile.cpp : 定义控制台应用程序的入口点。
//
#include "stdafx.h"

int foo(int argc) {

    return argc + 10;
}

//volatile 易变的
int main(int argc, char* argv[])
{
    volatile int n = 0;

    int* pN = const_cast<int*>(&n);
    //int* pN = (int*)(&n);
    printf("pN %p\r\n", pN);
    printf("argc %d\r\n", argc);
    n = foo(argc);
    printf("n %d\r\n", n);
    int k = 0;
    printf("n %d\r\n", n);

    return 0;
}
/*
output:
pN 0019FF3C
argc 1
n 11
n 11
*/
```

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）

```
#include "stdafx.h"

int n = 100;
int main(int argc, char* argv[])
{
    int n = 101;

    printf("n is %d\r\n", n);

    return 0;
}
/*
output: n is 101
*/
```

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`

```
#include "stdafx.h"
#include "test.h"

namespace CR32 {
  int n = 32;
}

int main()
{
  printf("n is %d\r\n", CR32::n);
  printf("m is %d\r\n", CR32::m);

  return 0;
}
/*
output:
n is 32
m is 2
*/
```

`test.h`

```
#include "stdafx.h"

namespace CR32 {
  int m = 2;
}
```

名字空间可以嵌套

```
#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   {
```

```
    int n = 1;
    printf("n is %d\r\n", n);
  }

  return 0;
}
/*
output:
n is 1

  有几点要注意的问题，
  1 test.h中名字空间的引入
  2 using namespace CR32;的定义
  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。
*/
```

`test.h`

```
#include "stdafx.h"

namespace CR32 {
  int m = 2;
}
```

声明只使用名字空间的部分变量或函数 `using CR32::n`

```
#include "stdafx.h"
#include "test.h"

int main()
{
  using CR32::m;
  printf("m is %d\r\n", m);

  return 0;
}
```

```
/*
output:
m is 2
*/
```

```
//test.h
#include "stdafx.h"

namespace CR32 {
  int m = 2;
}
```

`::n` 表示使用全局域中的 `n`

```
#include "stdafx.h"

int n = 6;

int main()
{
  printf("::n is %d\r\n", ::n);

  return 0;
}
/*
output:
::n is 6
*/
```

名字空间取别名 `namespace c=CR32;`

```
#include "stdafx.h"
#include "test.h"

```

```
namespace c=CR32;

int main()
{
  printf("CR32 alias c , m is %d\r\n", c::m);

  return 0;
}
/*
output:
CR32 alias c , m is 2
*/
```

```
// test.h
#include "stdafx.h"

namespace CR32 {
  int m = 2;
}
```

`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

```
#include "stdafx.h"

namespace CR32 {
  int m = 2;
}

int main()
{
  using CR32::m;

  printf("m is %d\r\n", m);

  return 0;
```

```
}
/*
output:
m is 2
*/
```

`namespace` 中的变量名或函数会在名称粉碎中添加名字空间域名