wohu1104 关注
创新行者 · 暂无认证

533 3986 1789 547+ 粉丝 周排名 总排名 访问 等级
9352 244 379 144 1626 积分 勋章 获赞 评论 收藏
私信 关注

搜博主文章

热门文章

Linux 中 ls 显示隐藏文件时，但实际未占用第一问题分析 36366
Ubuntu 16.04 安装后修改屏幕分辨率 (xrandr: Failed to get size of gamma for output default) 23596
Python 标准库之 hashlib 16843
docker load 出错 open /var/lib/docker/tmp/docker-import-837327978/blobs/sum: no such file or directory 14913
Ubuntu 安装 NVIDIA 显卡驱动详细步骤（详解CMD: The Nouveau kernel driver is currently in use by your system） 12628

最新评论

gstang 用一个包括动态区域周围保证 ... 进击的小白菜: 我要试试你最后了，也还是没有效果🤔
Go 学习笔记（65）— Go 中函数参数是... FishTalkStudio: 文章写得很详细，例子注释也很清晰，点个赞👍
Ubuntu 16.04 安装后修改屏幕分辨率(xr... -fu atlo.: 有效 太感谢了！
Depends, python2.2-minimal (> 2.7.15_r... wohu1104: 切换到更新版者重新尝试试
Depends, python2.2-minimal (> 2.7.15_r... Starry Sky001: sudo apt-get install apt0ru de 这一步就失败了

您觉得以前为博客“博客评测”吗？
强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

规范的 Commit Message
README 规范和语法文档规范
常用开源协议介绍以及开源软件规范列表

2022年 23篇 2021年 136篇
2020年 241篇 2019年 120篇
2018年 15篇 2014年 1篇

命名空间

原创 wohu1104 于 2020-06-16 21:20:44 发布 · 295 收藏 6
分类专栏：C++
C++ 专栏收录该内容 3 订阅 40 篇文章 订阅专栏

版权

1. 命名空间概念

在 C++ 应用程序中，您可能会写一个名为 `func()` 的函数，在另一个可用的库中也存在一个相同的函数 `func()` ，这样，编译器就无法判断您所使用的是哪一个 `func()` 函数。

因此，引入了命名空间，它为这个问题提供了一个解决方案的问题，它们作为附加信息来区分不同库中相同名称的函数、类、变量等。使用了命名空间即定义了上下文，本质上，命名空间就是定义了一个范围。

命名空间是结尾关键字的名称，有助于降低命名冲突的风险。

2. 定义命名空间

命名空间的定义使用关键字 `namespace` ，后跟命名空间的名称，如下所示：

```
namespace namespace_name {
  // 代码声明
}
```

3. 调用命名空间

为了调用带有命名空间的函数或变量，需要在前面加上命名空间的名称，如下所示：

```
name::code;  // code 可以是变量或函数
```

4. using 指令

可以使用 `using namespace` 指令，这样在使用命名空间时就可以不用在前面加上命名空间的名称。这个指令会告诉编译器，后续的代码将使用指定的命名空间中的名称。

5. 使用示例

5.1 自定义命名空间

```
#include <iostream>
using namespace std;

// 第一个命名空间
namespace first_sapce
{
    void func()
    {
        cout << "first namespace" << endl;
    }
}

// 第二个命名空间
namespace second_space

    void func()
    {
        cout << "second namespace" << endl;
    }
}
int main()
    // 调用第一个命名空间中的函数
    first_sapce::func();
    return 0;
}
```

为了避免频繁输入命名空间名称，也可以使用 `using namespace` 指令，如下所示：

```
using namespace first_sapce;
int main()
{  // 调用第一个命名空间中的函数
    func();
    return 0;
}
```

或者仅使用用到的函数，调用第一个命名空间中的函数：

```
using first_sapce::func;

int main()
{
    func();
    return 0;
}
```

5.2 标准命名空间

1. 未使用 `using namespace` 指令

```
#include <iostream>

int main()
{
    std::cout << "hello,world" << std::endl;
    return 0;
}
```

2. 使用 `using namespace` 指令包含全部元素

通过告诉编译器您要使用名称空间 `std` ，在第 7 行使用 `cout` 和 `endl` 时，就无需显式地指定名称空间了。

```
#include <iostream>

using namespace std;

int main()

    cout << "hello,world" << endl;
    return 0;
}
```

3. 使用 `using namespace` 指令仅包含使用的元素

`using` 指令也可以用来指定命名空间中的特定项目。例如，如果您只打算使用 `std` 命名空间中的 `cout` 部分，您可以使用如下的语句：

```
using std::cout;   // 这种语法结构为 using 声明
```

完整示例函数

```
#include <iostream>

using std::cout;
using std::endl;

int main()

    cout << "hello,world" << endl;
    return 0;
}
```

两者的差别在于：

- 前者让您能够在不显式指定名称空间限定符 `std::` 的情况下使用名称空间 `std` 中的所有元素（ `cout` 、 `cin` 等）；
- 后者让您能够在不显式指定名称空间限定符 `std::` 的情况下仅使用 `std::cout` 和 `std::endl` ；

`using` 指令引入的名称通常位于它的范围规则。名称从使用 `using` 指令开始是可见的，直到该范围结束。此时，在范围以外定义的同名实体是隐藏的。

一个一般性的原则是：除非是 `std` 这种在某个应用领域中大家已经非常熟悉的名字空间，否则最好不要使用 `using` 指令，过度使用 `using` 指令带来的问题是，你已经记不清每个名字来自哪里，结果就是你又陷入名字冲突之中。

显式使用全限定名或者使用 `using` 声明就不存在这个问题。因此，将一个 `using` 指令放在头文件中是一个非常坏的习惯。因为用户就无法避免上述问题。

6. 不连续的命名空间

命名空间可以定义在几个不同的部分中，因此命名空间是由几个单独定义的部分组成的。一个命名空间的各个组成部分可以分散在多个文件中。

所以，如果命名空间中的某个组成部分需要请求定义在另一个文件中的名称，则仍然需要声明该名称。下面的命名空间定义可以是定义一个新的命名空间，也可以是为已有的命名空间增加新的元素：

```
namespace namespace_name {
  // 代码声明
}
```

7. 嵌套命名空间

命名空间可以嵌套。您可以在一个命名空间中定义另一个命名空间，如下所示：

```
namespace namespace_name1 {
  // 代码声明
  namespace namespace_name2 {
    // 代码声明
  }
}
```

可以通过使用 `::` 运算符来访问嵌套的命名空间中的成员：

```
// 访问 namespace_name2 中的成员
using namespace namespace_name1::namespace_name2;

// 访问 namespace_name1 中的成员
using namespace namespace_name1;
```

在上面的语句中，如果使用的是 `namespace_name1` ，那么在该范围内 `namespace_name2` 中的元素是不可以直接使用的。除非使用命名空间说明 `second_space::second()` ，如下所示：

```
#include <iostream>
using namespace std;

// 第一个命名空间
namespace first_space

    void func()
    {
        cout << "first_space" << endl;
    }
// 第二个命名空间
namespace second_space
    {
        void func()
        {
            cout << "second_space" << endl;
        }
    }
using namespace first_space::second_space;

int main ()

    // 调用第二个命名空间中的函数
    func();
    return 0;
}
```

输出结果：

```
second_space
```

```
#include <iostream>
using namespace std;

// 第一个命名空间
namespace first_space

    void func()
    {
        cout << "first_space" << endl;
    }
// 第二个命名空间
namespace second_space
    {
        void second()
        {
            cout << "second_space" << endl;
        }
    }

using namespace first_space;

int main ()

    // 调用第一个命名空间中的函数
    func();
    // second(); 'second' was not declared in this scope
    second_space::second();
    return 0;
}
```

输出结果：

```
first_space
second_space
```

（C++）命名空间与using声明和using编译——概念 小唐TIMMY的博客 · 21
先来复习一下名称空间的属性：1、声明区域：可以在其中进行声明的区域。eg： 在函数外声明变量，其声明区域为其声明的所...
c++命名空间——namespace weixin_34456465的博客 · 202
C++ 命名空间 C++ 允许指定一个名称，例如，您可能会写一个名为 func() 的函数，在另一个可用的库中也存在一个相同的函数 func()。这样，编译器就无法...
参与评论 您还未登录，请先 登录 后发表或查看评论
C++系列: 命名空间 weixin_38956024的博客 · 833
目录 1.什么是命名空间？ 2.实现 2.1 外部引用嵌套命名空间内的符号 2.2 命名空间内容不用加定义时符号的引用 3.最后 1.什么是命名空间？ 就是...
sql中用JOIN USING 简化JOIN ON weixin_33752338的博客 · 69
Mysql 中联接SQL语句中，ON子句的语法格式为：table1.column_name = table2.column_name。当编式这样对联接表的列采用了相同的名称式时，...
Mysql ON子句和USING子句 weixin_34606468的博客 · 99
Mysql ON子句和USING子句：Mysql 中联接SQL语句中，ON子句的语法格式为：table1.column_name = table2.column_name。当编式这样对联接表...
C++ 命名空间 namespace的 前置 Fourier_Legendre的博客 · 5557
#include &<iostdlib.h> #include &<iostream> namespace detail // 定义一个namespace { const int n=99;...
C#方法与参数 常见命名空间汇总 using的使用 main方法参数 weixin_30677993的博客 · 52
本文主要讲 C# 常见命名空间using static 指令 & 参数 & 调用参数方法 前置命名空间&