**C++学习笔记——namespace与std命名空间**

注：本文会交替使用“namespace”和其中文翻译“命名空间”，二者含义相同。

**1. namespace用法**

*namespace identifier {*

*// 名称*

*}*

定义一个namespace（命名空间），其中的变量/函数/类名称，只在该命名空间内生效。

如果需要在命名空间外访问其内部的名称，需要使用如下格式，其中’::’被称为作用域解析运算符：

*namespace名称::namespace内部标识符名称*

不同命名空间内可以存在同名的标识符。

可以使用using，来将某一命名空间，或其中的部分标识符，添加到全局命名空间中。之后访问被添加好的标识符时，直接使用其名称即可，无需再指定命名空间。

示例：

图形用户界面, 文本

描述已自动生成

**2. namespace的作用**

namespace是为了解决名称冲突而提出的，其本身是可扩展的。

使用namespace是多人协作开发中大型项目的推荐流程。通常的开发流程为：

(1) 创建一个头文件，在其中声明一个namespace，其中包含基础的类/变量/函数声明。

(2) 需要扩展时，只需要添加一个头文件，在其中再声明一个与(1)中同名的namespace，并将所需代码添加到该namespace中。

(3) 使用#include时，只包含必要代码所在的头文件，而不是所有包含该namespace的头文件。

**3. std命名空间（namespace std）**

在C语言中，没有命名空间的概念。早期的C++在整合C代码时，直接使用C风格的头文件，即以”.h”结尾的头文件<xxx.h>，如<stdio.h>。

由于C头文件所有名称都是全局的，因此出现了大量命名冲突。

为了解决这个问题，同时保持对C的兼容，一方面，C++设计者保留了所有来自C语言头文件<xxx.h>，使得之前的代码能够正常运行；另一方面，他们按照2中所述的namespace开发流程，重构了所有<xxx.h>头文件，将这些头文件中涉及的所有名称，都放到一个统一的命名空间下，并将重构后的头文件，去除”.h”后缀，添加”c”前缀，更名为<cxxx>。

例如，<stdio.h>重构后的名称为<cstdio>；<stdlib.h>被重构后的名称为<cstdlib>。

这个统一的namespace，就是namespace std。

简言之，<xxx.h>头文件中的名称，都不在std命名空间中；而<cxxx>头文件中的名称，都在std命名空间中。我们可以通过一段简单的printf示例代码来验证这一点：

首先，在源文件中包含<cstdio>。此时使用std::printf，能够正确编译和运行：



而后，改为包含<stdio.h>，使用std::printf，则编译失败，原因是找不到printf：



最后，std本身是STL（Standard Library，标准模板库）的子集。