1. **函数定义(function definition)**

函数定义是完整的函数单元，包括函数的使用方式和函数功能的具体实现。C++中，函数定义由两部分组成，函数头（function header）和函数体（function body）。

1. 函数头

函数头的语法形式：**函数返回值类型 函数名（形参表）**

1. 函数返回值类型：各种合法的数据类型，可以是系统预定义的类型，也可以是用户定义的类类型、枚举类型等。这是为函数执行完毕后，返回到调用点时要带回的函数运行结果约定数据类型，取决于函数功能的输出要求。比如整数加法的结果是整型，浮点数数组查找最大值的结果是浮点型，但浮点数数组查找最大值所在下标的结果是整型

* 如果函数返回值类型是**void**，代表函数功能不需要返回结果；
* 如果函数返回值类型是引用或者指针，不应该返回局部变量的引用或者地址。
* 没有函数返回值类型的特殊情况：类的构造函数。（课堂知识范围内）

1. 函数名：用合法的标识符给函数命名。函数名除了用于函数调用，还表示函数代码所在的内存地址。
2. 形参表：调用函数时，为完成函数功能所提供的输入数据定义。

* 形参的个数可以为0个，1个或者多个，多个参数之间用**，**分隔。
* 无论有没有形参，形参表的（）都不可省略。
* 每个形参定义都必须包括参数的数据类型和参数名。比如(int x , int y)，如果写成(int x , y)则不合法。

1. 函数体

函数体是完成函数功能所需要的语句集合，必须包含在**{ }**之间。

* 当函数返回值类型为**void**时，函数体内应用**return;** 语句结束函数运行，返回函数调用点。
* 当函数返回值类型为其他数据类型时，函数体内应用**return *expression*;** 语句结束函数运行，带着***expression***的计算结果返回函数调用点。

1. **函数调用**

函数调用是使用函数，激活函数代码的方法。没有被调用的函数代码不会被执行。main函数在运行程序时被操作系统调用。

函数调用的语法形式：**函数名（实参表）**

编译器解析函数调用的语法是，检查两个正确：

1. 函数调用正确。编译器能找到符合函数调用的函数签名，即查找程序中是否包含和函数名一致且实参表和形参表在参数个数、类型和顺序上都匹配的函数定义或函数声明。

* **函数签名（function signature）**由函数名和形参表组成，程序中可以有函数名一样的多个函数，但每个函数签名必须是独一无二的。
* 函数名相同，但函数签名不同（即形参表不一样）的一组函数构成**函数重载**。

1. 函数返回结果使用正确。函数运行结束后，函数的返回值结果将取代函数调用参与下一步的代码运行。编译器会根据函数的返回值类型判断后续代码对函数结果的使用是否正确。
2. **函数声明（function declaration）**

函数声明用于在编译器没有处理到函数定义之前告知编译器函数的存在，以确保编译器能正常解析函数调用点的处的代码语法。C++中，函数声明用函数原型（function prototype）来完成，函数原型的语法形式：

**函数返回值类型 函数名（形参表\*）**

1. 函数原型和函数头的异同

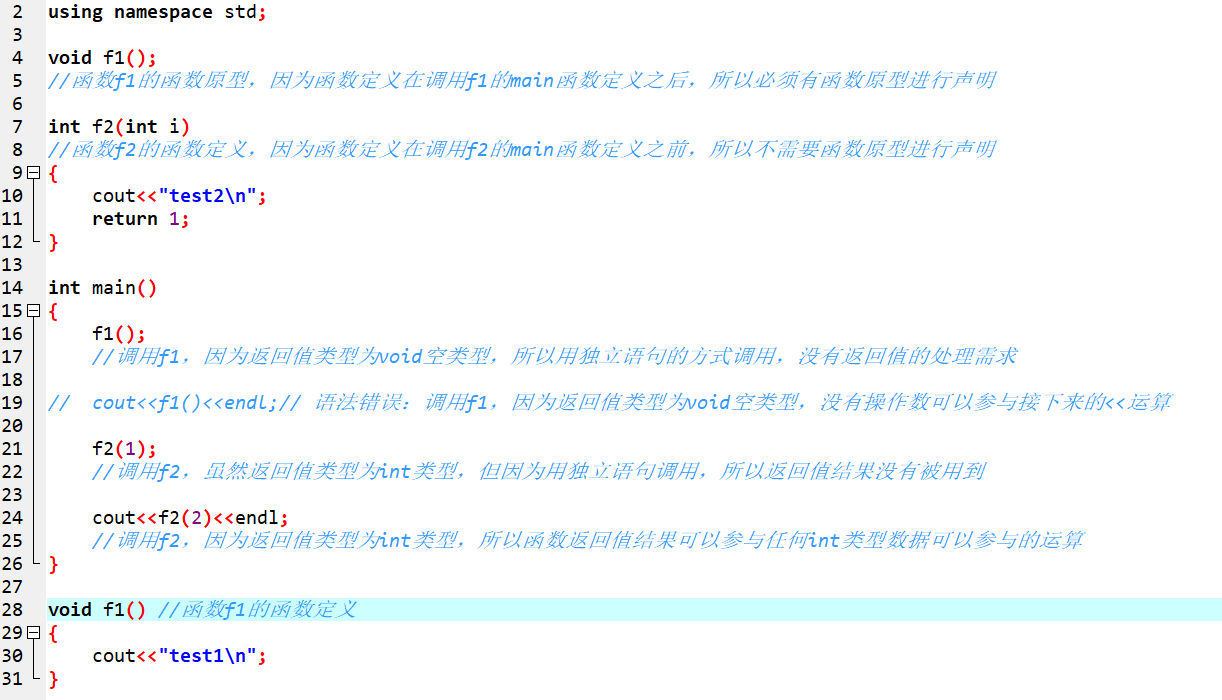
结合函数声明的作用和编译器的语法解析检查，可以看到，函数返回值类型、函数名，以及参数表中的参数个数和参数类型是编译器在分析函数调用前必须清楚的内容，因此这几个组成部分是函数头和函数原型中都必须包括的，函数原型和函数头很相似，可以复制函数头直接作为函数原型。不同之处在于，函数原型仅需要声明的作用，而函数头还兼具将调用点的输入实参保存并带入到函数体运行的实际作用，因此函数头的形参表中参数名是必须的，而函数原型的形参表\*中，参数名可有可无，就算有也不需要和函数头保持一致，对函数体运行没有任何作用。

1. 函数原型的应用场景

函数原型主要用于两种情况：

* 编写源代码时，函数的定义写在了函数调用之后，那么必须用函数原型在函数调用前进行声明，以保证编译通过。
* 将代码接口与实现分离时，函数原型（此时即为函数的接口）写于头文件，而将函数定义写在cpp文件中。编译后，cpp文件所包含的源代码不需要被公开，只需要交付可执行代码。

例1：



例2：

