新起点电脑教程

第 1 章

C++语言基础

本章要点

- 什么是 C++
- 搭建 C++开发环境
- 编写第一个 C++程序

本章主要内容

想必大家都听说过“C 语言”吧，有些读者甚至使用过 C 语言，因为它是大中专院校的基础课程之一。而 C++语言是对 C 语言的重大改进，C++的最大特点是通过“类”成了一门“面向对象”的语言。在本章的内容中，将介绍学习 C++语言必备的基础知识和搭建开发环境的方法。

1.1 什么是 C++

C++是在 C 语言的基础上发展起来的一种面向对象编程语言，支持过程化程序设计、数据抽象、面向对象程序设计、制作图标、泛型等多种程序设计风格。

↑扫码看视频

1.1.1 C++的发展历史

C++和 C 语言确实有很大的渊源。C 语言之所以起名为“C”，是因为它主要参考那个时候的一门叫 B 的语言，它的设计者认为 C 语言是 B 语言的进步，所以就起名为 C 语言；但是 B 语言并不是因为之前还有个 A 语言，而是 B 语言的作者为了纪念他的妻子，他的妻子的第一个字母是 B；当 C 语言发展到顶峰的时刻，出现了一个版本叫 C with class，这就是 C++最早的版本。其特点是在 C 语言中增加了关键字 class 和类，那时有很多版本的 C 都希望在其中增加类的概念。后来 C 标准委员会决定为这个版本的 C 起个新的名字，并征集了很多个名字，最后采纳了其中一个能人的意见，以 C 语言中的运算符“++”来体现它是 C 语言的进步，所以就叫作 C++，并成立了 C++标准委员会。

在成立 C++标准委员会后，美国 AT&T 贝尔实验室的本贾尼·斯特劳斯特卢普博士在 20 世纪 80 年代初期发明并实现了 C++(最初这种语言被称作“C with class”)。一开始 C++是作为 C 语言的增强版出现的，从给 C 语言增加类开始，不断地增加新特性，到后来的虚函数(virtual function)、运算符重载(operator overloading)、多重继承(multiple inheritance)、模板(template)、异常(exception)、RTTI、命名空间(name space)等逐渐被加入标准。1998 年，国际标准化组织(ISO)颁布了 C++程序设计语言的国际标准 ISO/IEC 1488—1998。

C++是具有国际标准的编程语言，通常称作 ANSI/ISO C++。1998 年是 C++标准委员会成立的第一年，以后每 5 年视实际需要更新一次标准。2011 年 8 月 12 日，国际标准化组织和国际电工委员会(IEC)旗下的 C++标准委员会(ISO/IEC JTC1/SC22/WG21)公布了 C++11 标准，并于 2011 年 9 月出版。

2012 年 2 月 28 日的国际标准草案(N3376)是最接近于 C++11 标准的草案(仅编辑上的修正)，此次标准为自 C++98 发布后 13 年来第一次重大修正。C++的最新正式标准 C++11 于 2014 年 8 月 18 日公布。

1.1.2 C++的优点和缺点

C++支持多种编程范式，包括面向对象编程、泛型编程和过程化编程。其编程领域广泛，

常用于系统开发、引擎开发等应用领域，是最强大编程语言之一，支持类、封装、重载等特性。C++语言有很多优点，下面列出了几条比较重要的优点。

- C++语言灵活，运算符的数据结构丰富、具有结构化控制语句、程序执行效率高，而且同时具有高级语言与汇编语言的优点。与其他语言相比，C++可以直接访问物理地址，与汇编语言相比又具有良好的可读性和可移植性。
- C++语言具备了 C 的简洁、高效，接近汇编语言等特点，对 C 的类型系统进行了改革性的扩充，因此 C++比 C 更安全，C++的编译系统能检查出更多的类型错误。另外，由于 C 语言的广泛使用，因而极大地促进了 C++的普及和推广。
- C++语言支持面向对象的特征，虽然与 C 的兼容使得 C++具有双重特点，但它在概念上完全与 C 不同，更具面向对象的特征。
- 出于保证语言的简洁和运行高效等方面的考虑，C++的很多特性都是以库(如 STL)或其他形式提供的，而没有直接添加到语言本身里，这样使得 C++更加容易上手。
- C++引入了面向对象的概念，使得开发人机交互类型的应用程序更为简单、快捷。很多优秀的程序框架包括 Boost、Qt、MFC、OWL、wxWidgets、WTL 等使用的就是 C++。

对于广大初学者来说，C++比较难学。C++语言由于本身复杂，其编译系统受到 C++复杂性的影响，对于很多没有编程基础的读者来说，会感觉 C++比较难以入门。

1.2　搭建 C++开发环境

Microsoft Visual Studio 2017 是微软推出的全新专用开发工具，是一个集成的开发环境工具。Microsoft Visual Studio 2017 是一款功能齐全且可以扩展的免费 IDE，能够创建适用于 Windows、Android 和 iOS 的应用程序，以及 Web 应用程序和云服务。

↑扫码看视频

1.2.1　Visual Studio 2017 的新功能

- 更优秀的性能和工作效率：更加专注于新型、现代化的移动、云和桌面开发功能，与以前的版本相比，现在的 Visual Studio 启动速度更快、响应能力更强、使用的内存更少。
- 可以使用 Azure 开发云应用：通过内置的 Azure 工具套件，可以轻松地创建由 Microsoft Azure 提供支持的云应用。借助于 Visual Studio，可以轻松配置、构建、调试、打包和部署 Azure 上的应用和服务。
- 跨平台开发：可以向任意目标平台无缝提供软件，使用.NET Core 编写在 Windows、Linux 和 macOS 操作系统上运行的应用和库。

- 支持 AI 开发：通过使用 Visual Studio Tools for AI，可以生成、测试和部署 AI 程序，实现与 Azure 机器学习无缝集成的深入学习 AI 解决方案。

1.2.2 安装 Microsoft Visual Studio 2017

在微软公司推出的 Microsoft Visual Studio 2017 安装包中，主要包含如下三个版本。

- 企业版：能够提供点对点的解决方案，充分满足正规企业的要求。这是功能最为强大的版本，价格最贵。
- 专业版：提供专业的开发者工具、服务和订阅。其功能强大，价格适中，适合于专业用户和小开发团体。
- 社区版：提供全功能的 IDE，完全免费，适应于一般开发者和学生。

安装 Microsoft Visual Studio 2017 企业版的具体流程如下。

第 1 步 登录微软 Visual Studio 官网：https://www.visualstudio.com/zh-hans/，如图 1-1 所示。

图 1-1 微软 Visual Studio 官网

第 2 步 单击“下载 Visual Studio”下的 Enterprise 2017 链接开始下载，如图 1-2 所示。下载后得到一个 exe 格式的可安装文件“vs_enterprise__2050403917.1499848758.exe”，如图 1-3 所示。

图 1-2 单击“Enterprise 2017”链接

图 1-3 可安装文件

第 3 步　鼠标右键单击下载文件“ vs_enterprise__2050403917.1499848758.exe”，选择使用管理员模式进行安装。在弹出的界面中单击“ 继续”按钮，表示同意许可条款，如图 1-4 所示。

图 1-4　单击“继续”按钮

第 4 步　在弹出的“ 正在安装”界面中选择所要安装的模块，本书内容需要选择安装如下模块。

- 通用 Windows 平台开发。
- .NET 桌面开发。
- ASP.NET 和 Web 开发。
- 数据存储和处理。
- 使用.NET 的移动开发。

上述各模块的具体说明如图 1-5 所示。在左下角可以设置安装路径。

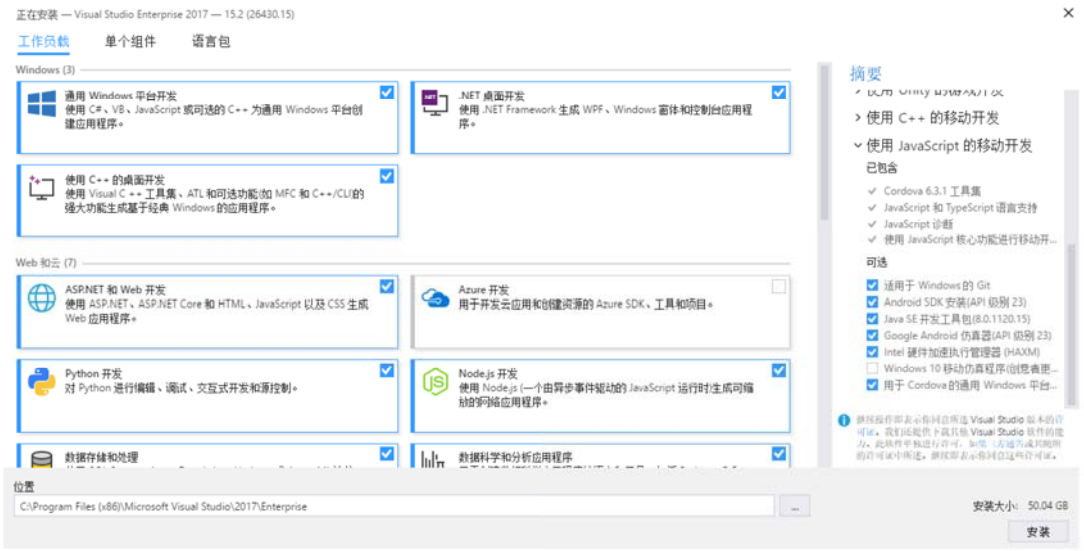

图 1-5　“正在安装”界面

第 5 步　单击“ 安装”按钮后弹出安装进度界面，这个过程比较耗费时间，读者需要耐心等待，如图 1-6 所示。

第 6 步　安装成功后的界面效果如图 1-7 所示。

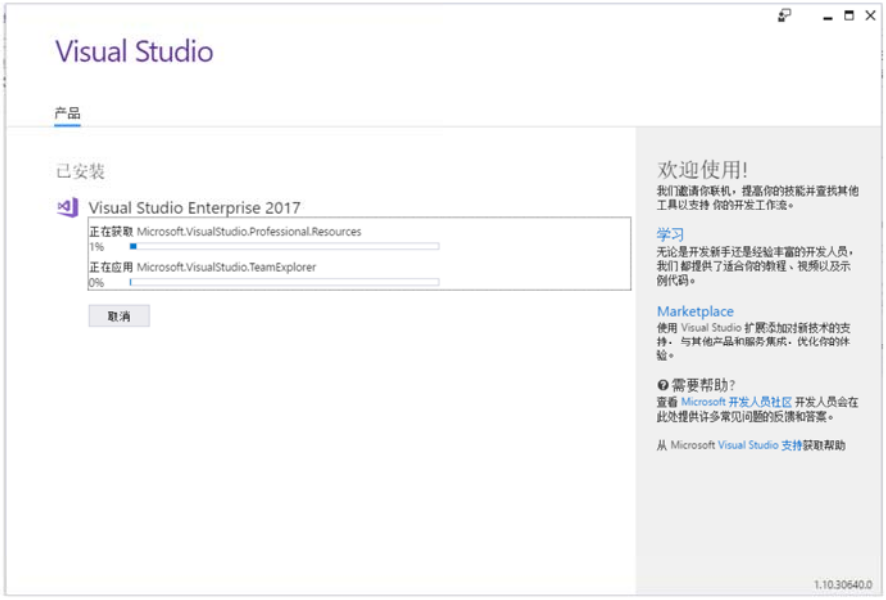

图 1-6　安装进度界面

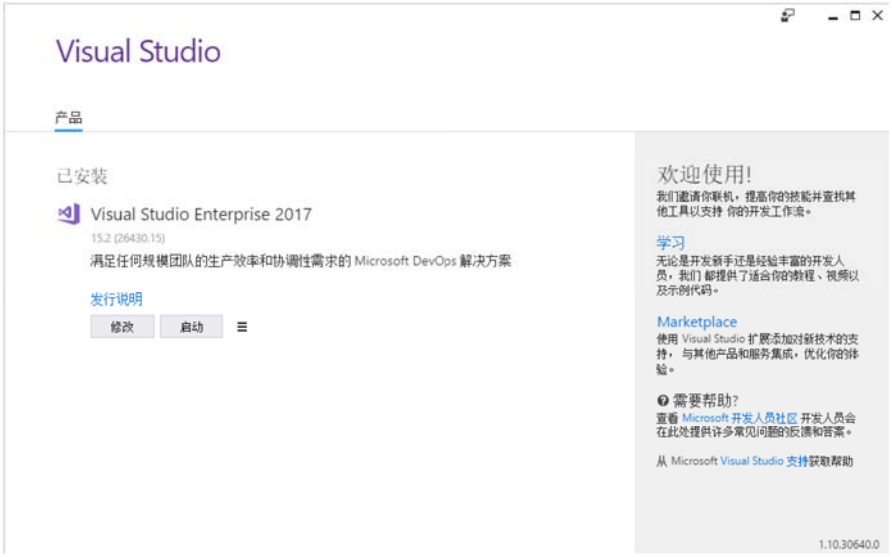

图 1-7　安装成功界面

第 7 步　单击“开始”按钮，在“所有应用”中单击 Visual Studio 2017 图标即可启动刚安装的 Visual Studio 2017，如图 1-8 所示。

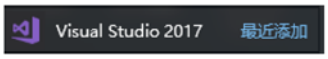

图 1-8　启动菜单

1.3　实践案例与上机指导

通过本章的学习，读者基本可以掌握 C++语言基础和搭建 C++开发环境的知识。其实 C++语言基础和搭建 C++开发环境的知识还有很多，这需要读者通过课外渠道来加深学习。下面通过练习操作，以达到巩固学习、拓展提高的目的。

↑扫码看视频

在接下来的实例中，将使用 Visual Studio 2017 编写一段 C++程序，使读者初步了解 C++程序的结构、语法规则和表达方式，并初步体验开发工具的魅力。

实例 1-1：在屏幕中输出指定的字符串

源文件路径：daima\1\1-1

第 1 步　打开 Visual Studio 2017，选择“文件”→“新建”→“项目”菜单命令，如图 1-9 所示。

图 1-9　选择“项目”命令

第 2 步　在弹出的“新建项目”对话框中，在左侧“模板”中选择 Visual C++选项，在中间选中“Win32 控制台应用程序”选项，在下方的“名称”文本框中设置本项目的名称为“C++1”，如图 1-10 所示。

第 3 步　单击“确定”按钮后出现“欢迎使用 Win32 应用程序向导”界面，如图 1-11 所示。

第 4 步　单击“下一步”按钮后出现“应用程序设置”界面，在“应用程序类型”中选中“控制台应用程序”单选按钮，在下方的“附加选项”中选中“预编译头”复选框，如图 1-12 所示。

第 5 步　单击“完成”按钮后会创建一个名为“C++1”的项目，并自动生成一个名为“C++1.cpp”的程序文件。如图 1-13 所示。

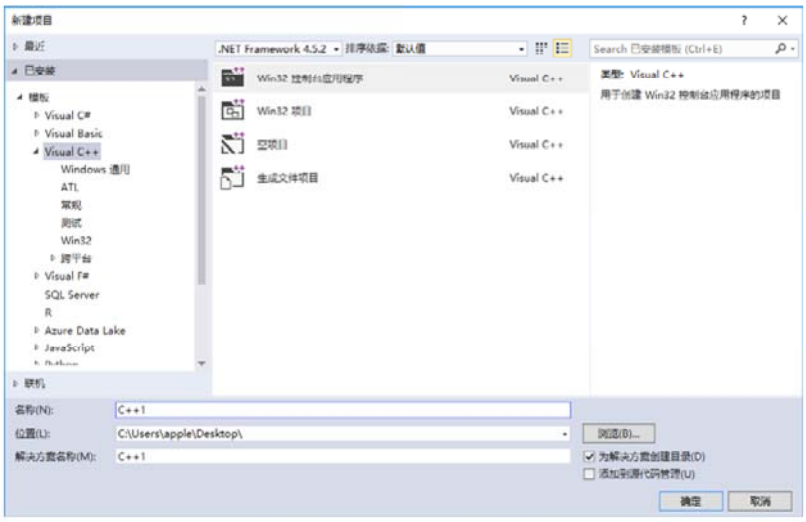

图 1-10 “新建项目”对话框

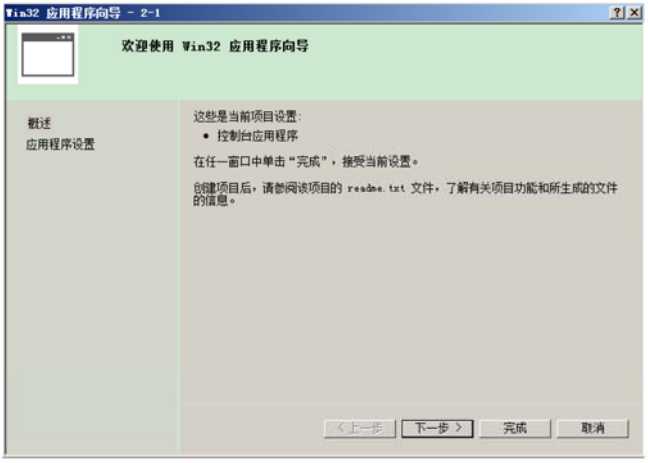

图 1-11 “欢迎使用 Win32 应用程序向导”界面

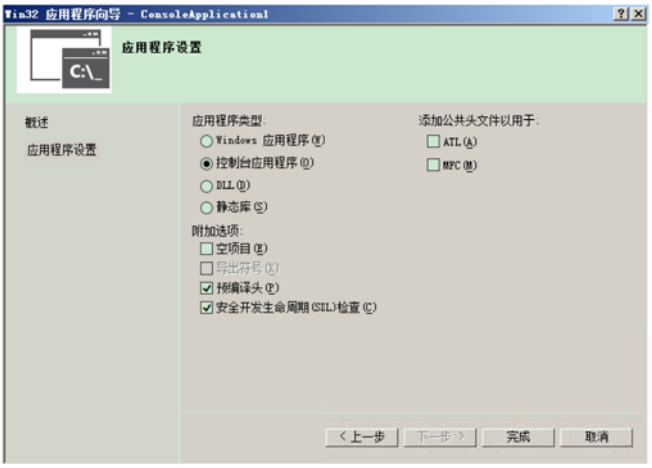

图 1-12 “应用程序设置”界面

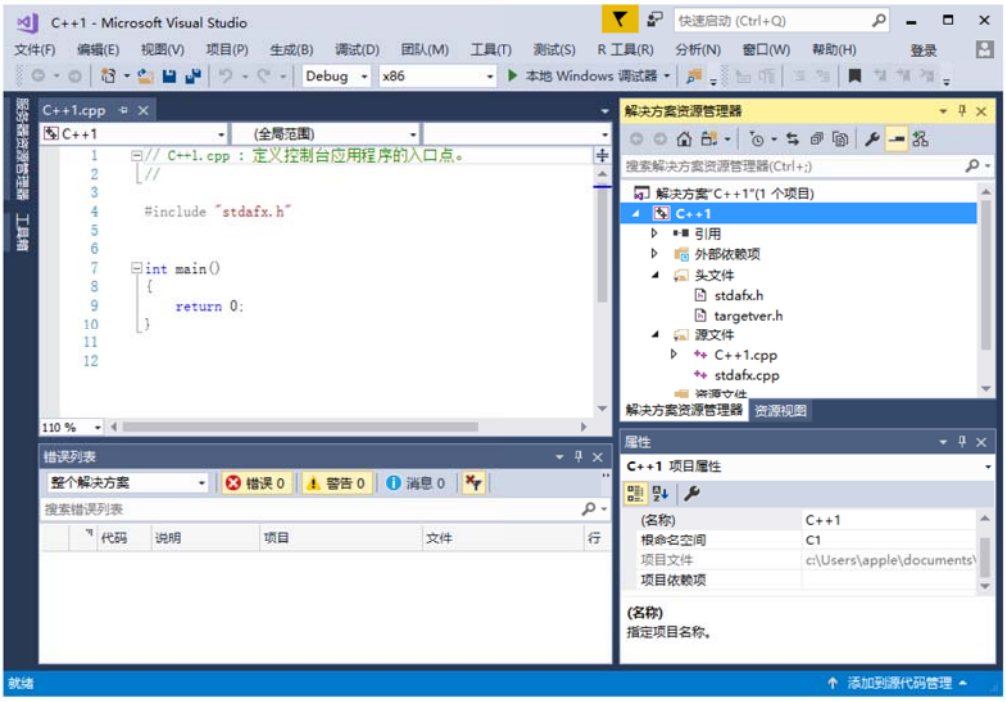

图 1-13　自动生成的文件 c++1.cpp

第 6 步　将如下代码复制到文件 c++1.cpp 中。

```
#include "stdafx.h"
#include "iostream"
using namespace std;
int main(){
    cout << "这是第一个C++程序！" << endl;
    cout << "这是第一个C++程序！" << endl;
}
```

在 Visual Studio 环境中，namespace 是一个表示范围的标识符。命名空间用关键字 namespace 来定义。命名空间是 C++的一种机制，用来把单个标识符下的大量有逻辑联系的程序实体组合到一起。该标识符作为此组群的名字。因为 C++标准程序库中的所有标识符都被定义在一个名为 std 的 namespace 中，所以在 Visual Studio 版的 C++程序中都必须引用上述代码。

执行后的效果如图 1-14 所示。

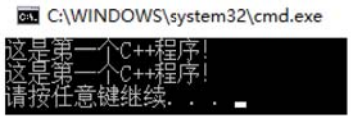

图 1-14　Visual Studio 2017 版的执行效果

1.4 思考与练习

本章详细讲解了 C++语言基础和搭建 C++开发环境的知识，在讲解过程中，通过具体实例介绍了搭建 C++开发环境的方法。通过本章的学习，读者应能熟悉 C++语言基础和搭建 C++开发环境的知识，并掌握它们的使用方法和技巧。

1. 选择题

(1) 在 20 世纪 90 年代，最流行的 C++开发工具是(　　)。

A. Visual C++ 6.0　　B. Visual Studio 2005　　C. DEV C++

(2) 假设变量 a= 23，b= 124，则$a & $b 的结果是(　　)。

A. 20　　B. 127　　C. 147　　D. -121

2. 判断对错

(1) 2012 年 2 月 28 日的国际标准草案(N3376)是最接近 C++11 标准的草案(仅编辑上的修正)，此次标准为自 C++98 发布后 13 年来的第一次重大修正。C++的最新正式标准 C++11 于 2014 年 8 月 18 日公布。(　　)

(2) Microsoft Visual Studio 2017 是微软推出的全新专用开发工具，它是一个集成的开发环境工具，是一款功能齐全且可扩展的免费 IDE。(　　)

3. 上机练习

(1) 输出“大家好才是真的好！”。

(2) 输出节日的祝福。

新起点电脑教程

第 2 章

C++程序的基本结构

本章要点

- 什么是面向对象
- 分析 C++的程序结构
- 必须遵循的编码规范
- 输入和输出

本章主要内容

C++语言不但吸取了传统汇编语言的优点，而且开创了全新的面向对象语言世界。从 C++语言诞生之日起，软件领域彻底进入面向对象时代。C++语言的最重要特质是面向对象，当然，除了面向对象以外，C++还有很多其他方面的优点。在本章的内容中，将详细讲解 C++程序的基本结构。

2.1 什么是面向对象

面向对象(Object Oriented，OO)是一种软件开发方法，是一种对现实世界理解和抽象的方法。C++是一门面向对象的编程语言，在本节的内容中，将简要介绍面向对象的基本知识和C++面向对象编程的流程。

↑扫码看视频

2.1.1 两种对象的产生方式

在编程语言中，对象的产生通常基于如下两种方式。

1. 基于原型

原型的概念已经在认知心理学中被用来解释概念学习的递增特性，原型模型本身就是企图通过提供一个有代表性的对象来产生各种新的对象，并由此继续产生更符合实际应用的对象。而原型-委托也是OOP(面向对象)中的对象抽象，是代码共享机制中的一种。

2. 基于类

一个类提供了一个或多个对象的通用性描述。从形式化的观点看，类与类型有关，因此一个类相当于是从该类中产生的实例的集合。而在一切皆为对象的背景下，在类模型基础上还诞生了一种拥有元类的新对象模型，即类的本身也是一种其他类的对象。

2.1.2 C++面向对象编程的流程

面向对象编程方法是学习C++编程的指导思想。当使用C++进行编程时，应该首先利用对象建模技术来分析目标问题，抽象出相关对象的共性，对它们进行分类，并分析各类之间的关系；然后再用类来描述同一类对象，归纳出类之间的关系。编程大师Coad和Yourdon在对象建模技术、面向对象编程和知识库系统的基础之上设计了一整套面向对象的方法，具体来说分为面向对象分析(OOA)和面向对象设计(OOD)。对象建模技术、面向对象分析和面向对象设计共同构成了系统设计的过程，如图2-1所示。

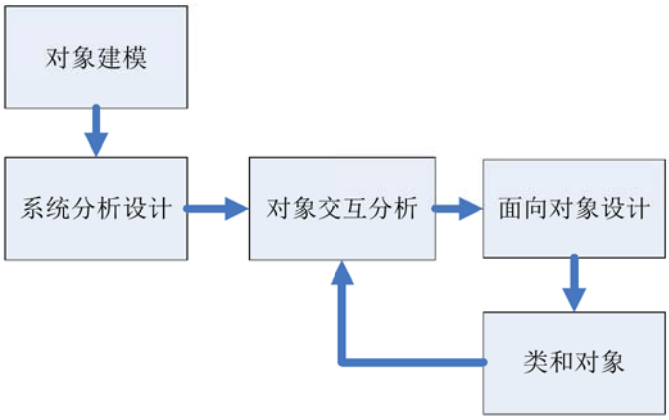

图 2-1　系统设计处理流程

2.2　分析 C++的程序结构

程序结构是程序的组织结构，它包括语句结构、语法规则和表达式，内容中包含了代码组织结构和文件组织结构。在编写 C++程序的过程中，我们必须严格遵循这些规则，才能编写出高效、易懂的程序。

↑扫码看视频

2.2.1　初识 C++程序结构

请读者先看如下一段 C++代码。

```
//这是一个演示程序，它从命令行读入一个整数，然后加 1 再输出
#include <stdafx.h>
#include <iostream.h>
int main(){
    int x;
    cout<<"请输入一个数字：";
    cin>>x;
    x=x+1;
    cout<<"x=x+1="<<x<<endl;
    return 0;
}
```

在上述代码中，将整段程序划分为如下三个部分。

(1)　注释部分，即上述代码中的首行，用双斜杠标注。

```
//这是一个演示程序，它从命令行读入一个整数，然后加 1 再输出
```

注释部分即对当前程序的解释说明部分，通常会说明此文件的作用和版权等信息。

(2) 预处理部分，即上述代码中的第 3 行：

```
#include <iostream.h>
```

预处理即在编译前需要提前处理的工作。例如此段代码表示编译器在预处理时，将文件 iostream.h 中的代码嵌入到该代码指示的地方，此处的#include 是编译命令。在文件 iostream.h 中声明了程序需要的输入输出操作信息。

(3) 主程序部分，即剩余的代码：

```
int main(){
    int x;
    cout<<"请输入一个数字：";
    cin>>x;
    x=x+1;
    cout<<"x=x+1="<<x<<endl;
    return 0;
}
```

此部分是整个程序的核心，用于实现此程序的功能。C++的每个可执行程序都有且只有一个 main 函数，它是程序的入口点。执行 C++程序后，首先会执行这个函数，然后从该函数内调用其他需要的操作。下面依次分析上述代码的主要功能。

- 第 2 行：表示定义一个 int 类型的变量，并命名为 x，后面的分号表示此条代码到此结束。

```
int x;
```

- 第 3 行：表示通过 cout 输出一行文字。此处的 cout 是 C++中预定义的系统对象，当程序要向输出设备输出内容时，需要在程序中设置此对象，输出操作符用“<<”表示，表示将“<<”右边的内容输出到“<<”左边的对象上。例如此行代码表示在标准输出设备上输出字符串文字“请输入一个数字：”。

```
cout<<"请输入一个数字：";
```

- 第 4 行：cin 代表标准输入设备的对象，即 C++中的预定义对象。当程序需要从输入设备接受输入时，就需要在程序中使用该对象。输入的操作符是“>>”，表示将“>>”左边接受的输入放到右边的对象中。当程序执行到该代码时，会停止并等待来自标准输入设备的输入。输入完毕后按 Enter 键，cin 会接受输入并将输入放到对应的对象中，然后跳到下一行代码开始执行。

```
cin>>x;
```

- 第 5 行：“+”表示加法运算，将“+”两边的数字相加；“=”表示赋值之意，将“=”右边的运算结果放到“=”左边的对象中去。

```
x=x+1;
```

- 第 6 行：这是一条在标准输出设备上输出文字的代码，包含了 3 个输出操作符：第 1 个操作输入了文字 x=x+1；第 2 个操作输出对象 x 保存的值；第 3 个操作的右边是 endl，表示回车换行之意。

```
cout<<"x=x+1="<<x<<endl;
```

➢　第 7 行：本行表示跳出当前程序，即返回操作系统，使用数字 0 作为返回值。

```
return 0;
```

智慧锦囊

很多编译器并不特别要求函数 main 必须有返回值，例如 VC++，但是为了使读者养成好的习惯，建议都设有返回值。

2.2.2　C++程序的文件组织

C++程序的文件组织是指在一个 C++项目中的构成文件，如果是一个简单的 C++程序，仅仅需要几行代码就可以完成，这时 C++程序的文件结构是最简单的，只需一个文件即可保存所有的代码。这种简单情况不需要进行详细讲解，例如图 2-2 展示了一个典型的 C++程序的文件结构。

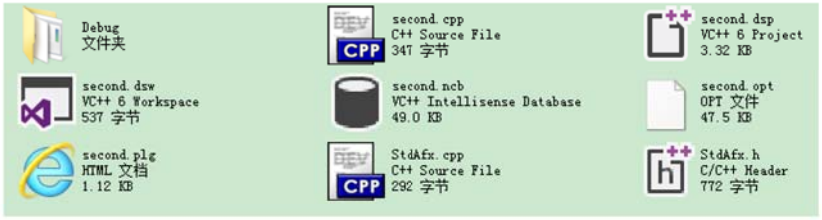

图 2-2　简单 C++项目的结构

在图 2-2 所示的结构中，只有文件 second.cpp 包含了当前项目的核心功能代码。在日常应用中，往往一个项目的程序代码会比较复杂，例如经常需要编写多个类和多个函数。为了使项目文件规整有序地排列，需要为文档设置比较合理的安排方法。

下面给读者提供 5 条建议，可以为项目合理地规划好整体的文件结构。

(1)　每个类的声明写在一个头文件中，根据编译器的要求可以加.h 后缀名，也可以不加。这个头文件一般以类的名字命名。为了防止编译器多次包含同一个头文件，头文件总是以下面的框架组织：

```
#ifndef CLASSNMAE_H_
#define CLASSNAME_H_
将类的声明写在这里面
#endif
```

知识精讲

CLASSNAME_H_ 中的 CLASSNAME 就是在这个文件中声明的类名。

(2)　将类的实现放在另一个文件中，取名为 classname.cpp(classname 为用户在类声明文件中声明的类名)。在该文件中的第一行包含类声明的头文件，如：#include "classname"(C++

新标准不支持带.h 的头文件)，然后在此文件中写类的实现代码。一般格式：

```
#include "classname"
```

(3) 与类相似，在编写函数时，总是把函数的声明和一些常数的声明放在一个头文件中，然后把函数的具体实现放在另一个头文件中。

(4) 通常来说，如果在某个源文件中需要引入的头文件很多，或者为了源程序的简洁，可以将头文件的引入功能写在另一个头文件中，然后在源程序的第一行引入这个头文件即可。

(5) 当在文件中需要使用函数和类时，只需引入类和函数声明的头文件，而无须包含实现的文件。

一个中型或大型项目通常会包含很多.cpp 文件和函数文件，例如图 2-3 所示的结构。

图 2-3 复杂 C++项目的结构

2.3 必须遵循的编码规范

编码规范即我们在编写代码时需要遵守的一些规则，在编写 C++程序时需要遵守一些编码规范。好的编码规范，可以提高代码的可读性和可维护性，甚至提高程序的可靠性和可修改性，保证了代码的质量。特别是在团队开发大型项目时，编码规范就成了项目高效运作的要素。

↑扫码看视频

2.3.1　养成良好的编程风格

在编写 C++程序时，需要养成如下所示的编程风格。

- 程序块缩进，要使用 Tab 键缩进，不能和空格键混合使用。
- 函数不要太长，如果太长，建议拆分处理。
- 不要使用太深的 if 嵌套语句，可以使用函数来代替。
- 双目操作符号前后加空格，更加醒目。
- 单目操作符前后不加空格。
- 不要使用太长的语句，如果太长，可以分行处理。
- 每个模板中只有一个类。
- if、while、for、case、default、do 等语句要独占一行。
- 一行不能写多条语句。
- 如果表达式中有多个运算符，要用括号标出优先级。

2.3.2　必须使用的注释

使用注释可以帮助阅读程序，通常用于概括算法、确认变量的用途或者阐明难以理解的代码段。注释并不会增加可执行程序的大小，编译器会忽略所有注释。

在 C++中有两种类型的注释：单行注释和成对注释。

1)　单行注释

单行注释以双斜线(//)开头，行中处于双斜线右边的内容是注释，会被编译器忽略。例如：

```
//计算 m 和 n 的和
z=add(m,n);
```

2)　成对注释

成对注释也叫注释对(/**/)，是从 C 语言继承过来的。这种注释以“/*”开头，以“*/”结尾，编译器把注释对“/**/”之间的内容作为注释。任何允许有制表符、空格或换行符的地方都允许放注释对。注释对可跨越程序的多行，但不是一定要如此。当注释跨越多行时，最好能直观地指明每一行都是注释的一部分。我们的风格是在注释的每一行以星号开始，指明整个范围是多行注释的一部分。例如：

```
/*计算 m 和 n 的和
z 只是一个简单函数而已
*/
z=add(m,n);
```

在 C++程序中通常混用上述两种注释形式，具体说明如下。

- 注释对：一般用于多行解释。
- 双斜线注释：常用于半行或单行的标记。

太多的注释混入程序代码可能会使代码难以理解，最好是将一个注释块放在所解释代码的上方。当改变代码时，注释应与代码保持一致。程序员即使知道系统其他形式的文档

已经过期，还是会信任注释，认为它会是正确的。错误的注释比没有注释更糟，因为它会误导后来者。

在使用注释时必须遵循如下原则。

- 禁止乱用注释。
- 注释必须和被注释内容一致，不能描述与其无关的内容。
- 注释要放在被注释内容的上方或被注释语句的后面。
- 函数头部需要注释，主要包含文件名、作者信息、功能信息和版本信息。
- 注释对不可嵌套：注释对总是以“/*”开始并以“*/”结束。这意味着，一个注释对不能出现在另一个注释对中。由注释对嵌套导致的编译器错误信息容易使人迷惑。

2.3.3 获取 3 个输入数字中的最大数值

化妆可以使美女帅哥们变得更加光彩夺目，对于 C++程序代码来说，化妆也同样重要。通过对代码进行化妆处理后，可以大大提高代码的可视性，能够充分显示出一个程序员的良好素养和编程风格。下面通过一个简单示例的代码，体会一下代码的缩进和注释的艺术，本实例的功能是获取 3 个输入值中的最大数值。

实例 2-1：获取 3 个输入数字中的最大数值

源文件路径：daima\2\2-1

实例文件 second.cpp 的主要实现代码如下。

```
#include "stdafx.h"                          //这是必须设置的头文件
int MaxIn3(int x,int y,int z);               //定义函数 MaxIn3
int main(int argc, char* argv[]){
    int x,y,z;                               //定义 3 个变量

    cout<<"请输入 3 个得分数字";              //显示一段文本
    cin>>x>>y>>z;                            //获取输入的 3 个数字
    cout<<MaxIn3(x,y,z)<<endl;               //调用函数 MaxIn3 处理 3 个数字
    return 0;
}
/*
* 函数名称：MaxIn3
* 参    数：接收三个整型参数
* 返回值  ：无
* 函数功能：找出三个整型数中较大的数
* 作    者：XXX

* 版本号  ：0.0.1
* 修改日期：XXXX.XX.XX
*/
int MaxIn3(int x,int y,int z)  {             //函数 MaxIn3 的具体实现
    int num=0;                               //存放最大数
    //选择最大数
    if (x>y){                              //如果 x 大于 y
        if (x>z)                           //如果 x 大于 z
```

```
            num=x;                          //x 是最大数
        else                                //如果 x 不大于 z
            num=z;                          //则 z 是最大数
    }
    else{                                   //如果 x 不大于 y
        if (y>z)                            //如果 y 大于 z
            num=y;                          //则 y 是最大数
        else                                //如果 y 不大于 z
            num=z;                          //则 z 是最大数
    }
    return num;                             //返回最大数
}
```

在上述代码中，变量、函数、if 语句等内容都是 C++语言语法中的重要知识点，这些知识点将在本书后面的章节中进行详细讲解。通过讲解上述实例，目的不是说明代码的语法或功能实现等知识，而是让读者仔细观察这段代码，分析其中的代码缩进和注释的书写格式。

通过 Visual C++ 6.0 或 Visual Studio 2017 进行调试，运行后将首先提示输入 3 个得分数字，如图 2-4 所示；输入 3 个得分数字并按 Enter 键后，将输出其中最大的得分数字，如图 2-5 所示。在此按 Enter 键退出当前程序。

图 2-4　提示输入 3 个数字

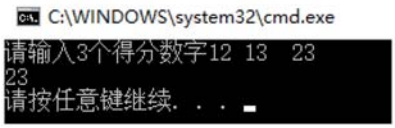

图 2-5　输出其中最大的数字

2.4　输入和输出

在 C++语言程序中，必须通过输入和输出才能实现用户和计算机的交互，实现软件程序的具体功能。其实 C++并没有直接定义进行输入或输出(I/O)的任何语句，而是由标准库(Standard Library)提供，标准库为程序员提供了大量的工具。

↑扫码看视频

2.4.1　标准输入与输出对象

在 C++标准库中定义了四个 I/O 对象，在处理输入功能时使用命名为 cin(读作 see-in)的 istream 类型对象，这个对象也叫作标准输入(standard input)。在处理输出功能时使用命名为 cout(读作 see-out)的 ostream 类型对象，这个对象也称为标准输出(standard output)。在标准库中还定义了另外两个 ostream 对象，分别命名为 cerr 和 clog(分别读作 see-err 和 see-log)。

对象 cerr 又叫作标准错误(standard error)，通常用来输出警告和错误信息给程序的使用者。而 clog 对象用于产生程序执行的一般信息。在一般情况下，系统将这些对象与执行程序的窗口联系起来。这样，当我们从 cin 读入时，数据从执行程序的窗口读入，当写到 cout、cerr 或 clog 时，输出写至同一窗口。运行程序时，大部分操作系统都提供了重定向输入或输出流的方法。利用重定向可以将这些流与所选择的文件联系起来。

2.4.2 一个使用 I/O 库的程序

接下来将通过一个例子助读者深入理解 I/O 库，这是一段实现把两个数字相加的处理代码。我们可以使用 I/O 库来扩充 main 程序，实现由用户给出两个数，然后输出它们的和的功能，主要实现代码如下。

```
#include <iostream>
int main(){
    std::cout << " Enter two numbers: " << std::endl;
    int v1, v2;
    std::cin >> v1 >> v2;
    std::cout << " The sum of " << v1 << " and " << v2
              << " is " << v1 + v2 << std::endl;
    return 0;
}
```

上述代码非常简单，首先在用户屏幕上显示提示语“输入两个数字：”，输入后按 Enter 键，将输出两个数字的和。

1) 第一行

程序的第一行是一个预处理命令，功能是告诉编译器要使用 iostream 库。

```
#include <iostream>
```

尖括号里的名字是一个头文件，这是 C++标准库中的一个内置文件。当程序使用库工具时必须包含相关的头文件。#include 指令必须单独写成一行：头文件名和#include 必须在同一行。通常，#include 指令应出现在任何函数的外部。而且习惯上，程序的所有#include 指示都在文件开头部分出现。

2) 写入到流

main 函数体中第一条语句执行了一个表达式(expression)。在 C++中，一个表达式通常由一个或几个操作数和一个操作符组成。该语句的表达式使用输出操作符(<<)，在标准输出上输出如下所示的提示语。

```
std::cout << "Enter two numbers:" << std::endl;
```

上述代码用了两次输出操作符，每个输出操作符实例都接受两个操作数：左操作数必须是 ostream 对象，右操作数是要输出的值。操作符将其右操作数写到作为其左操作数的 ostream 对象。

在 C++中，每个表达式产生一个结果，通常是将运算符作用到其操作数所产生的值。当操作符是输出操作符时，结果是左操作数的值。也就是说，输出操作返回的值是输出流本身。既然输出操作符返回的是其左操作数，那么我们就可以将输出请求链接在一起。输出提示语的那条语句等价于：

```
(std::cout << "Enter two numbers:") << std::endl;
```

因为(std::cout << "Enter two numbers:")返回其左操作数 std::cout，这条语句等价于下面的代码。

```
std::cout << "Enter two numbers:";
std::cout << std::endl;
```

在上述代码中，endl 是一个特殊值，称为操纵符(manipulator)，当将它写入输出流时，具有输出换行的效果，并刷新与设备相关联的缓冲区(buffer)。通过刷新缓冲区，保证用户立即看到写入到流中的输出。

智慧锦囊

程序员经常在调试过程中插入输出语句，这些语句都应该刷新输出流。忘记刷新输出流可能会造成输出停留在缓冲区中，如果程序崩溃，将会导致程序错误推断崩溃位置。

3)　使用标准库中的名字

细心的读者会注意到，在上述程序中使用的是 std::cout 和 std::endl，而不是 cout 和 endl。前缀 std::表明 cout 和 endl 是定义在命名空间(namespace)std 中的。命名空间使程序员可以避免与库中定义的名字相同引起的命名冲突。因为标准库定义的名字是定义在命名空间中，所以我们可以按自己的意图使用相同的名字。

在标准库中使用命名空间的副作用是，当我们使用标准库中的名字时，必须显式地表达出使用的是命名空间 std 下的名字。std::cout 的写法使用了作用域操作符(scope operator，::操作符)，表示使用的是定义在命名空间 std 中的 cout。

4)　读入流

在输出提示语后会读入用户输入的数据，先定义两个名为 v1 和 v2 的变量(variable)来保存输入：

```
int v1, v2;
```

将这些变量定义为 int 类型，int 类型是一种代表整数值的内置类型。这些变量未初始化(uninitialized)，表示没有赋给它们初始值。这些变量在首次使用时会读入一个值，因此可以没有初始值。下一条语句读取输入：

```
std::cin >> v1 >> v2;
```

输入操作符(>>)行为与输出操作符相似，功能是接受一个 istream 对象作为其左操作数，接受一个对象作为其右操作数，它从 istream 操作数读取数据并保存到右操作数中。像输出操作符一样，输入操作符返回其左操作数作为结果。由于输入操作符返回其左操作数，我们可以将输入请求序列合并成单个语句。换句话说，这个输入操作等价于下面的代码。

```
std::cin >> v1;
std::cin >> v2;
```

输入操作的效果是从标准输入读取两个值，将第一个存放在 v1 中，第二个存放在

v2 中。

5) 完成程序

剩余代码的功能是打印输出结果：

```
std::cout << "The sum of " << v1 << " and " << v2
        << " is " << v1 + v2 << std::endl;
```

上述代码虽然比输出提示语的语句长，但是在概念上没什么区别，功能是将每个操作数输出到标准输出。有趣的是操作数并不都是同一类型的值，有些操作数是字符串字面值。例如下面的代码：

```
"The sum of "
```

其他是不同的 int 值，如 v1、v2 以及对算术表达式 v1 + v2 求值的结果。iostream 库定义了接受全部内置类型的输入输出操作符版本。

知识精讲

在写 C++程序时，大部分出现空格符的地方，可用换行符代替。这条规则的一个例外是字符串字面值中的空格符不能用换行符代替。另一个例外是换行符不允许出现在预处理指示中。

2.4.3 使用 using 声明命名空间

在 C++中提供了简洁的方式来使用命名空间成员，在此介绍一种最安全的声明机制——using 声明。通过使用 using 声明，允许程序员访问命名空间中的名字，而不需要加前缀 namespace_name::。使用 using 声明的语法格式如下：

```
using namespace::name;
```

在使用了 using 声明后，程序员就可以直接引用后面的名字 name，而不需要引用该名字的命名空间。例如在下面的演示代码中，如果没有 using 声明，而直接使用命名空间中的名字的无限定符版本是错误的，尽管有些编译器也许无法检测出这种错误。

```
#include <string>
#include <iostream>
using std::cin;
using std::string;
int main(){
    string s;           //正确：string 现在是一个 std::string
    cin >> s;           //正确：cin 现在是一个 std::cin
    cout << s;          //错误：不使用声明；必须使用全名
    std::cout << s;     //正确：明确使用来自 std 命名空间
}
```

在使用 using 声明命名空间时需要注意，一个 using 声明一次只能作用于一个命名空间成员。using 声明允许程序员在程序中明确指定用到的命名空间中的名字，如果程序员希望使用 std(或其他命名空间)中的几个名字，则必须给出将要用到的每个名字的 using 声明。例

如下面是一个利用 using 声明实现加法功能的演示程序。

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;
int main(){
    cout << "Enter two numbers:" << endl;
    int v1, v2;
    cin >> v1 >> v2;
    cout << "The sum of " << v1
        << " and " << v2
        << " is " << v1 + v2 << endl;
    return 0;
}
```

在上述代码中，对 cin、cout 和 endl 进行 using 声明，就意味着程序员可以省去前缀 std::，直接使用命名空间中的名字，且使得编写的程序更易读。

知识精讲

使用标准库类型的类定义

在有一种情况下，程序员必须总是使用完全限定的标准库名字：在头文件中。理由是头文件的内容会被预处理器复制到程序员编制的程序中。当我们用#include 包含文件时，就好像把头文件中的文本当作我们编写的文件的一部分了。如果在头文件中放置了 using 声明，就相当于在包含该头文件的每个程序中都放置了同一 using 声明，不论该程序是否需要 using 声明。通常，头文件中只定义确实必要的东西，请养成这个好习惯。

2.5　实践案例与上机指导

通过本章的学习，读者基本可以掌握 C++程序基本结构的知识。其实 C++程序基本结构的知识还有很多，这需要读者通过课外渠道来加深学习。下面通过练习操作，以达到巩固学习、拓展提高的目的。

↑扫码看视频

实例 2-2：使用 I/O 库输出文本信息

源文件路径：daima\2\2-2

实例文件 main.cpp 的主要实现代码如下。

```
#include <iostream.h>                    //引入标准库文件
void main(){                             //主函数
int i=0;                                 //定义变量 i
cout << i<< endl;                        //输出 i 的值
cout << "C++语言是一门面向对象的编程语言。" <<endl;
}
```

执行后的效果如图 2-6 所示。

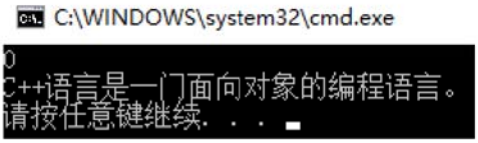

图 2-6　执行效果

2.6　思考与练习

本章详细讲解了 C++程序基本结构的知识，循序渐进地讲解了什么是面向对象、分析C++的程序结构、必须遵循的编码规范、输入和输出等知识。在讲解过程中，通过具体实例介绍了 C++程序基本结构的用法。通过本章的学习，读者应能熟悉 C++程序基本结构的知识，并掌握它们的使用方法和技巧。

1. 选择题

(1) int 是定义(　　)的。

A．变量　　B．函数　　C．类　　D．注释

(2) 下面是合法注释的是(　　)。

A．/*注释*/　　B．/*注释　　C．注释*/

2. 判断对错

(1) 太多的注释混入程序代码可能会使代码难以理解，最好是将一个注释块放在所解释代码的上方。当改变代码时，注释应与代码保持一致。　　(　　)

(2) 在 C++语言程序中，必须通过输入和输出才能实现用户和计算机的交互，实现软件程序的具体功能。　　(　　)

3. 上机练习

(1) 使用标准输入输出。

(2) 使用标准输出流(cout)。