第 1 部分　C++语言基础

第 1 章　　概 述

1.1　计算机程序设计方法及程序设计语言的发展

计算机是工业化、批量生产的产物，为保证通用化，其硬件功能相对固定，用户需要设计程序改变计算机系统的状态以适合人们的需求。计算机实际上是带有一套指令系统的机器，开发者直接面对的是这套指令系统，利用这套指令系统，开发者可以设计程序，驱动计算机工作，使功能相对固定的机器灵活地满足用户的需要。因此，程序设计是计算机应用的核心。

计算机是信息处理的工具，所谓计算机程序就是对客观世界的问题抽象后，建立合理的模型，最后由计算机程序设计人员利用合适的程序设计语言设计而成的、按人们的要求驱动计算机进行信息处理的指令序列，该指令序列包括数据和操作两部分内容。如图 1-1 所示。

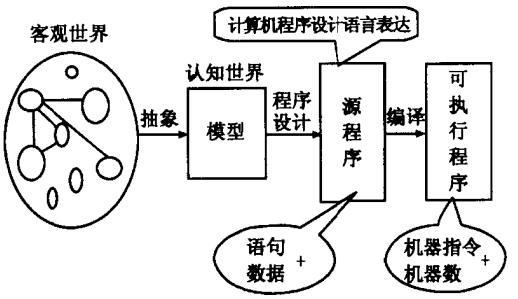

图 1-1　计算机程序设计语言在程序设计中的作用

程序设计语言在程序设计中具有很重要的作用，它既要符合计算机驱动的要求，又要合理地表达、描述客观世界。计算机从本质上来说是具有一定智能构成的机器，它要求对它的驱动准确、并符合其工作规律（比如：顺序执行、顺序存储）；而全面描述客观事物，又要求程序设计语言符合人类思维习惯。因此，寻找合理的计算机程序设计方法和开发高效率的程序设计语言是计算机行业的永恒的课题。从第一台计算机诞生以来，程序设计方法和程序设计语言就在不断发展。

1.1.1　早期发展

在计算机发展的初期，由于计算机硬件条件的限制，运算速度低，存储空间有限，软件的规模也小，这时程序员追求的目标是计算机系统资源的高利用率。这一阶段没有固定的程序设计方法，程序设计是程序员本人很个性化的艺术，强调的是技巧，不太讲究所编写程序的结构，

所开发的程序其他人难以理解，程序的修改也比较困难，计算机的应用领域也主要在科学计算方面。此时，系统软件还不完善，所以软件的设计中还没有系统软件和应用软件的界限，用户所开发的程序往往要兼顾系统管理的一些功能，软件设计有很明显的面向系统硬件平台的特性，无可移植性。软件设计工具主要以机器语言和汇编语言为主。

所谓机器语言，是指由计算机硬件系统可以直接识别的二进制编码组成的语言。机器语言面向硬件系统平台，只适用于开发功能简单、规模较小的软件。用机器语言开发软件，存在通用性差、修改、测试都比较困难的问题。

汇编语言是用比较好理解的英文助记符号代表机器语言的一种软件开发工具，如 mov，jmp，add 等，程序员用汇编语言设计源程序，然后可以利用计算机汇编程序完成对所设计程序的翻译，即将助记符号翻译成二进制机器码，如图 1-2 所示。

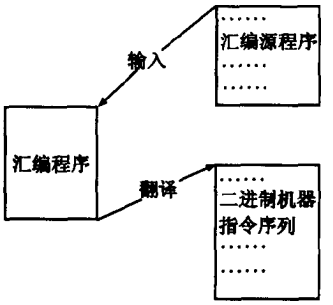

图 1-2　汇编语言源程序的编译

汇编语言的出现使人们摆脱了记忆机器语言的负担，借助于汇编语言，可以使用符合人们记忆表达习惯的语言设计计算机程序。汇编语言为人们设计程序时摆脱机器语言的束缚，使用适合于人类思维的工具进行计算机程序设计开辟了一条途径。

但汇编语言程序设计表达方式与人的正常思维方式仍然有很大的差距，程序员仍然摆脱不了顺序存储、顺序执行的机器束缚，在程序设计时依然要处理硬件工作的所有细节问题，所设计的程序的可读性、可修改性很差。程序某一部分的修改，可以导致整个程序重新编写。这一时期为 20 世纪 40 年代后期至 50 年代中期，软件规模不大，软件设计工作是少数计算机专家们特有的才能。

这时候，人们对客观事物的思想表达与计算机软件之间没有直接的联系，思想的表达必须经过具有专门计算机研究工作经验、具备汇编程序设计技巧的程序员的加工，纳入具体计算机的顺序执行、顺序存储的硬件工作执行范围内，才能够成为软件在计算机上运行。程序与描述客观事物的思想是分离的。

1.1.2　结构化程序设计时期

20 世纪 50 至 60 年代，随着计算机硬件技术的发展，运算速度和存储空间都得到了增长，系统软件也逐渐从应用软件中分离开。所谓系统软件是指管理计算机系统本身的专门软件（如操作系统等）它支持用户对计算机的运行管理，对用户屏蔽计算机运行的底层细节，用户通过通用的人机接口就可以调度计算机进行正常工作。用户也可以在系统软件的基础上开发应用软件，这就为程序设计语言脱离具体的硬件环境，进一步摆脱硬件工作机制的束缚提供了条件。

在这一时期，各种计算机高级语言相继出现，如 FORTRAN、COBOL 和 BASIC 语言等。

高级语言的出现是计算机程序设计语言技术发展的一大进步，它借助于系统软件屏蔽了计算机硬件的细节，提高了语言的抽象层次，程序可以由具有一定逻辑含义、便于理解的语句构成，这使得所设计的程序和程序所描述的具体事物之间可以从逻辑上直接联系起来。

随着软件的规模和复杂程度不断提高，人们也在不断探索新的程序设计方法。 20 世纪 70 年代初期出现了结构化程序设计语言： C 语言和 Pascal 语言。

结构化程序设计的方法是自顶向下、逐步求精的程序设计技术，自顶向下是一种分解问题的技术，逐步求精是指结构化程序的连续分解，最终成为三种基本控制结构（顺序、选择、循环）的组合。其结果是将一个程序最终由若干个过程组成，每个过程完成一个确定的功能。

Pascal 语言是根据结构化程序设计方法开发出来的计算机高级程序设计语言，其特点是它能提炼出程序设计的共同特征并能将这些特征编译成高效的代码。 Pascal 语言是结构化程序设计的有力工具，至今仍然是描述数据结构及其算法的强有力的语言，在计算机教育界广受关注。

C 语言是 1972 年美国贝尔实验室的丹尼思·里奇（Dennis Ritchie）开发的，是在肯·汤普生（K.Thompson）的 B 语言的基础上重新设计的语言。其最初的目标是为了描述和实现 UNIX 操作系统提供一种工作语言。UNIX 操作系统的 90%的代码是由 C 语言编写的。该语言最初是在 DEC 公司的 PDP-II 小型计算机上，在 UNIX 操作系统环境上实现的，随着 UNIX 操作系统的成功和广泛的应用，C 语言成为一种普遍使用的计算机语言。C 语言具有灵活方便、目标代码效率高、可移植性好的特点，一度成为事实上的语言标准，美国国家标准局于 1987 年制定了 C 语言标准，称为 ANSI C。

1.1.3 面向对象的程序设计

时至今日，结构化程序设计的方法仍然在软件业有极大的市场，几乎每一种程序设计语言都支持结构化程序设计的机制，包括 C++语言在内。

面向对象程序设计是建立在结构化程序设计的基础上的。与以往的程序设计方法不同的是，其程序设计是围绕着被操作数据而进行的，而不是围绕着程序本身。因此，面向对象的程序设计的出发点就是为了方便地描述客观世界的事物（即对象）及事物之间的关系。

打一个通俗的比喻，制作课桌，使用面向结构的设计方法，先要能够制作桌子腿、抽屉、台面，最后才能够制作课桌，其制作过程与人们正常思维的过程是相反的；而使用面向对象的设计方法，先进行课桌整体功能的设计、然后再考虑组成课桌的其他部分的制作方法，其制作过程与人类的思维方式相同。

如果说，汇编语言的出现，使人们的程序设计语言与实际的机器语言脱离开来，人们可以使用符合人类记忆习惯的符号进行程序设计；而面向过程的高级程序设计语言又使人们摆脱了在程序设计过程中具体硬件平台的限制，摆脱了程序设计中很多涉及计算机硬件的麻烦、枯燥的细节处理，只关心程序设计过程中的数据合理的描述和算法本身的实现，将软件设计从进行高深研究的实验室的专门人员手中解放出来，成为一个较为大众化的专业技能。因此，面向对象的程序设计语言的出现又给人们提供了符合人类思维习惯、符合人类社会化协作多人工作方式的开发计算机软件的工具，将软件设计变成一个社会化的职业工作，将软件生产变成一种工

业产业。

另外，现代计算机已经从最初的科学计算、字符处理向多媒体应用方面发展，软件的规模和复杂性在不断增加。比如，以个人微机为例，十年前在字符界面 DOS操作系统上进行的软件开发需要一个人在几个月内完成的工作量，在 Windows 图形界面上需要多人去协作开发，开发周期可以持续数年。软件规模和复杂性的增长也在迫使人们寻找新的开发工具，按新的软件开发组织方式进行软件生产，形成一种社会工业产业。在这种背景下，面向对象的程序设计方法和工具应运而生，这是软件产业化发展的必然趋势，面向对象的程序设计方法还在不断发展，它会不断给人们提供更加高效、快捷的软件开发工具。

C++语言是由 AT&T 贝尔实验室的 Bjarne Stroustrup 博士在 1980 年设计的，它是在 C 语言的基础上为支持面向对象的程序设计而研制的程序设计语言，为了支持面向对象的程序设计，该语言引入了类的机制，因此面世时称为带类的 C，1983 年正式命名为 C++程序设计语言。1989 年开始其标准化工作，1994 年制定 ANSI C++标准草案。目前，已经有比较成熟的 C++语言编译器产品，如 Microsoft 公司的 Visual C++和 Borland 公司的 Borland C++。C++语言得到广泛的应用，其软件工具还在不断发展、完善。

计算机程序设计语言和软件开发方法的发展历程如图 1-3 所示。

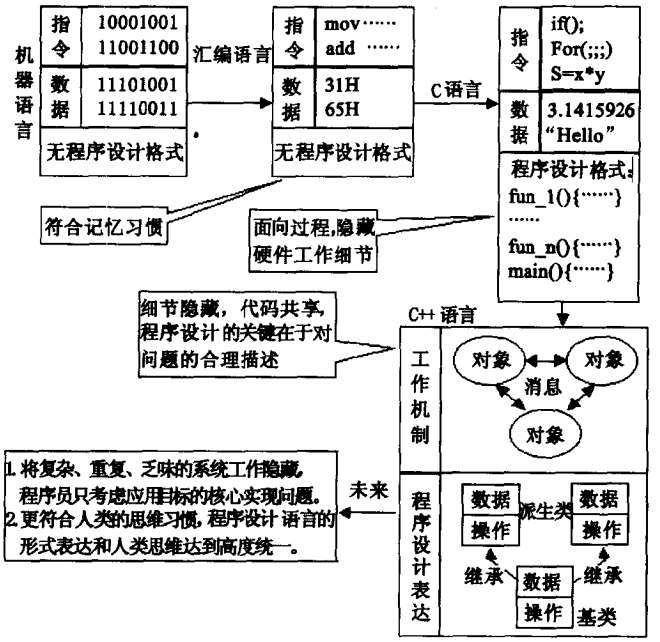

图 1-3　计算机语言和计算机软件开发方法的发展历程

1.2　面向对象程序设计的特点

从程序设计方法的发展可看到，面向对象的程序设计方法注重于对问题的分析和在分析基础上的合理设计。本节主要介绍面向对象程序设计技术中的一些主要概念。

1. 抽象（Data Abstraction）

抽象是指将具体事物一般化的过程，即对具有特定属性及行为特征的对象提取其共性的过程。在抽象过程中，需要研究目标程序要解决的问题，以及组成该问题的概念性实体，找出有利于解决问题的一些特性，从而得到对该问题的抽象结果。

在面向对象的程序设计中，往往需要进行不同级别的抽象过程。对高级抽象，可能会将一个问题抽象为一些特定对象的交互行为；而对低级抽象，这些特定的对象常常可抽象分解为其他更小对象的交互行为。假如要设计一个学校的办公管理系统，可以在高级别上将该办公系统看成是学校的一些不同的系、部门之间的信息交流，在低级别上，又可以看做不同的教研室、科室之间工作的相互配合。

抽象的目的在于代码重用，由于每一层的抽象都会带来不同的可重用代码，因此面对问题经常需要进行不同级别的抽象。

抽象是面向对象程序设计的一个基本特征，类就是对一些问题和概念进行抽象的工具，类从客观世界的一组事物中抽取其共同的属性和行为，对象则是类的实例化，具体化。

在面向对象的程序设计方法中，用对象来描述具体的事物，对象是构成应用系统的基本单位，它由类而来，包括属性和行为，属性就是描述对象特征的数据，行为则是对属性的操作。类和对象的关系如图 1-4 所示。

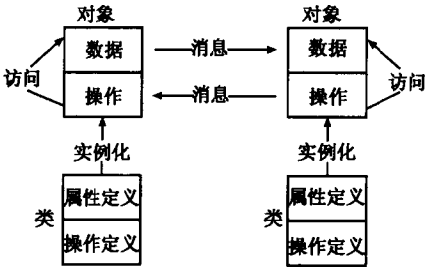

图 1-4　类的实例化及对象之间的交互方式

2. 封装（Encapsulation）

对象（Object）是面向对象程序设计的重要性质之一，从具体的程序设计工具的使用上来讲，对象就是既含有数据（叫做对象的属性——Attribute）又含有对数据操作的代码（Method）的一个逻辑实体。使用对象将数据和操作进行封装，这样，在程序中对数据的存取只能通过对该对象本身的操作来进行，程序的其他部分不能直接访问作用于此对象的数据，因此封装又称为数据隐藏。在面向对象的程序设计中，对象之间的交互只能通过消息来进行，如图 1-4 所示。

从图 1-4 可见，在面向对象程序中，类是对象的抽象，对象是类的实例化。通过定义类来尽可能地将一些关键数据、函数隐藏起来，这样，只有通过用户定义的接口才能访问类中的数据，避免了在程序开发过程中数据的不恰当使用，大大增加了程序设计过程的可靠性。这一点在需要多人协作、共同开发大型软件系统时，具有很重要的意义。

3. 继承（Inheritance）

继承是面向对象程序设计的又一个重要特性，在其他类型的程序设计中，数据类型之间没

有层次关系，但在面向对象的程序中，一个类可以继承另一个类中所有的数据和操作。按照这种程序的组织方法，程序既可以使用类中的对象完成特定的任务，也可以使用基本类提供的功能完成一些带有普遍性质的工作。如图 1-5 所示。

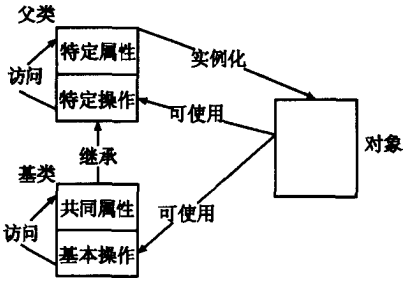

图 1-5　面向对象的继承关系

从图 1-5 可见，被继承的类称为基类，在继承基类的基础上产生的类称为父类，基类可以包含父类中的成员,反之则不行,继承关系具有单向性。在本书 C++程序设计中常常要用到类、实例化、对象、基类、父类等术语，读者现在就应当开始熟悉它们。

4. 多态（Ploymorphism）

同一个消息被不同的对象接受时被解释为不同的含义，从而导致不同的行为的现象称为多态。多态性与面向对象的继承性有关。利用多态性，用户可以通过发送统一的消息，由各个接受者来完成不同的实现细节，即所谓的“同一接口，多种方法”。

多态性的优点在于：减轻了程序设计人员的记忆负担，使程序的设计、修改更加灵活，程序的使用者只需要记住有限的几个接口就可以完成各种所需要的操作，程序中可以用简单的操作完成同一类体系中不同的对象的操作，在 C++ 语言程序中通过虚函数实现多态性。

5. 动态联编(Dynamic Binding)

通过继承性和多态性就可以实现动态联编。动态联编是指如果需要，把程序的某些组成部分在程序运行过程中，根据所接受到的信息连接起来的过程。它与多态性一样，使程序的开发更加灵活。

1.3　面向对象的软件开发过程

1.3.1　软件开发模式

分析、设计、实现是软件开发过程中的三个重要组成部分。分析实际上是对所面对的客观事物进行明确的定义的过程，在分析中要建立描述问题重要特征的模型，无论是对事物本身或事物之间的关系，都必须精确地描述其本来面目，分析过程的重点在于精确、完整地描述客观事物。设计是根据分析得到的模型，建立解决该问题的计算机软件各部分的程序架构的过程，其中还要考虑数据存储、人机界面等与下一步实现有关的问题。而实现则是根据设计方案，生

产软件产品的过程，包括编码、调试、修改、测试、软件发行等过程。

如果通过分析得到对客观事物中的问题的精确描述，通过设计得到了满足所描述问题的程序架构，并实现了按设计架构设计的程序，那么就成功地完成了软件开发。

早期的软件开发模式如图 1-6 所示。

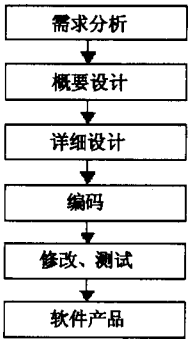

图 1-6　早期的软件开发模式

按照这种开发模式，每个任务都是一个封闭的实体，它们只能够从上一个任务得到输入，然后将结果输入给下一个任务，上一个任务完成以后，才可以进入下一个任务。这种方法的优点是软件开发的阶段性好，但也有很多缺点，开发过程一旦进入下一阶段，以前各个阶段所完成的工作就很难修改，一旦发现某些环节出错需要修改，则相关的工作量就很大，有时甚至导致整个软件重新设计。比如，在编码阶段发现概要设计的错误，就必须从需求分析开始考虑问题的修改。这种开发模式，适合于软件规模较小、功能相对比较简单、参加软件开发的人员不多的情况。

由于软件规模和复杂程度的不断增长，一般在分析阶段很难全面、细致地给出问题的精确模型，显然，上述开发模式已经不能满足软件开发的需要，在这种情况下，面向对象的程序设计方法和程序设计语言工具就应运而生，软件开发进入现代开发模式。如图 1-7 所示。

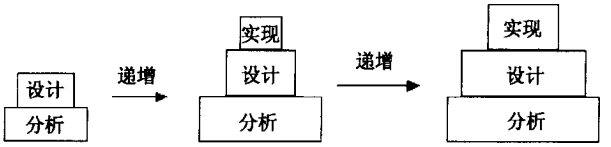

图 1-7　软件开发的现代模式

从图 1-7 可见，在这种模式下，软件开发过程被看成是一个逐步递增的过程。分析、设计、实现三个阶段是并行递增的过程，在设计的任何阶段，如果发现问题，都可以方便地回到前面的阶段去修改。软件开发是一个各阶段逐步增长、逐步完善的过程，但这种开发方式的有效性是受开发思想和开发工具的影响的。

1.3.2　C++语言程序设计中的主要问题

C++语言程序设计实际上就是利用软件开发过程中的分析阶段所形成的对事物描述的模型，建立一个层次明确、清晰的程序架构来解决实际问题。

1. 建立合理的类体系

在分析阶段，所有问题都是参照现实世界的对象来进行抽象的，通过抽象，将现实问题化解为它们各自的组成类。但此时类的建立的重点在于全面、合理地描述现实事物，因此在设计阶段仍然要对这些类仔细研究和分析,从中归纳出一些共同的属性和方法,形成一个基本类(基类)，在基类的基础上，派生继承出整个程序的类体系结构。在类的体系结构建立的过程中可能会有以下两种情况。

第一种情况，在一个类所包含的属性中，有些需要和其他类的相关属性按相同的方式处理，有些需要特殊处理，这样的类就要分解成基类和派生类，由基类提供公共属性和方法，由派生类处理各自的特殊要求。类分解如图 1-8所示。

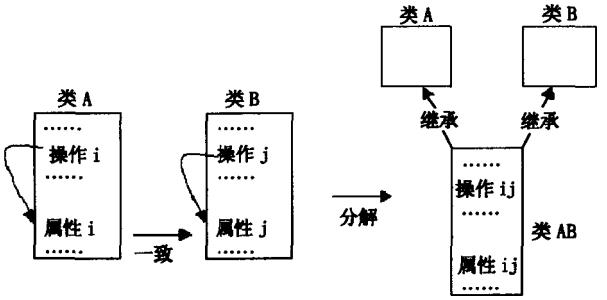

图 1-8　类分解示意图

第二种情况就是有些类中总包含有其他类的属性，这时，就要考虑是否将所包含的属性作为一个公共属性来处理，从而建立该类与另外一个类的逻辑联系。类属性的合成如图 1-9所示。

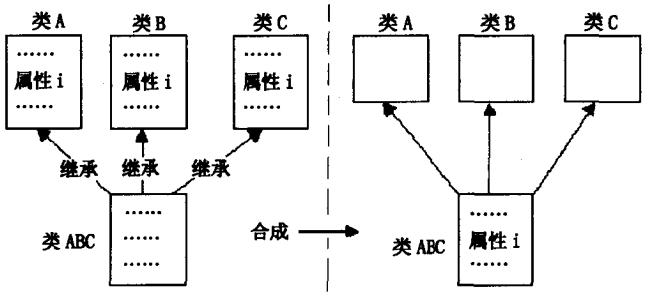

图 1-9　类属性的合成

类体系的设计，是面向对象程序设计的出发点，对今后的软件开发工作至关重要，关系到软件开发的成败。建立合理的类体系结构，就能够在今后的工作中收到事半功倍的效果。

另外，除了从目标问题中抽象建立起来的类以外，软件系统自身可能还需要建立一些内部类，比如，对于那些相互之间没有直接访问权利、但又需要互相通信的类，可以建立接口类实现它们的通信，将访问权限纳入到类管理的范畴，保证数据的安全。

2. 代码重用

采用面向对象程序设计方法的目的之一就是代码重用，为此，1980 年 Bjarne Stroustrup 博

士在其著名的《The C++ Programing Language》中提出组件的概念。所谓组件是一些类所组成的群组。组件有两个功能，每个组件完成一个独立的功能，每个组件都有一个与其他组件的通信接口。比如，将用户界面类划分为一个组件，将软件接口类划分为一个组件，将网络通信类划分成一个组件等。

所有的组件联合起来共同实现软件的功能。组件是类和软件之间的一个中间部件，如图 1-10 所示。根据组件，可以将软件从逻辑上划分成若干个组成部分，以便于由不同的软件开发组或不同的软件公司协同完成其各自不同的任务，在开发组内部或公司内用类实现代码共享，在开发组之间或公司间通过组件实现代码共享，类和组件的划分，使用户在代码共享的同时，具有了对组件内或类内重要数据必要的保护措施，降低了代码重用的出错风险。

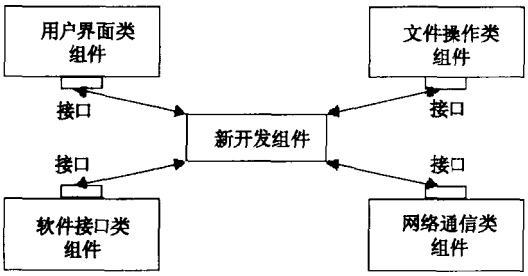

图 1-10　组件概念

已有组件或类一般都是经过程序设计人员精心设计，严格测试过的，设计人员在开发过程中应有意识地尽量采用现有组件、类库，以减低开发过程中的工作量。对于自己设计的类、组件、函数、过程等，应尽量使其具有通用性，为软件代码重用提供资源。目前软件重用还不成熟，但其不断的发展会大大缩短软件开发周期，降低软件开发、测试难度，提高软件的可靠性。

读者在学习和实践面向对象的程序设计技术时，要逐步建立类体系和代码重用的概念，在掌握基本概念、基本工具的基础上，应熟悉各种软件资源，深入理解面向对象程序设计技术。

1.4　C++程序的结构

1.4.1　C++程序的基本结构

每个 C++程序由注释、编译预处理、程序主体三部分组成。

【例 1-1　求两数之积。C++程序基本结构示例。

```
/* 一个完整的 C++语言程序示例    */
/*   功能两数相乘。
/* i—乘数，j—被乘数，sum—积     */
# include<iostream.h>
main()
{
    int i,j;          //说明了两个代表整数的标识符号
```

```
long int sum=0; //说明了一个代表长整数的标识符号
cout<<"Please input two integers, separated by space。"<<endl; //输入提示
cin>>i>>j;      // 输入操作
sum=i*j;
cout<<endl;
cout<<"The result is "<<sum<<endl;
    }
```

分析例 1-1。C++程序的基本结构如下。

1. 注释

注释是设计程序者对所设计的程序做的注解、说明，其目的在于提高程序的可读性。一个好的注释，也是源程序不可缺少的一部分。注释一般有两种，序言式注释和注解性注释。

序言式注释是对一段程序或模块（比如函数）功能的注释，一般位于程序段落或模块的开头，说明该程序段落或模块的名称、用途、所使用的变量等。如果有必要，甚至还可以写上编写者姓名和编写日期等。例 1-1 的程序中，一对符号“ /* ”和“ */ ”之间的部分就是序言式注释。序言式注释可以占用一行或多行。

注解式注释是对程序中关键的语句或难以理解的语句的注释，程序中符号“ // ”之后的部分就是注解式注释。它只能够占一行。

2. 编译预处理

C++语言的编译预处理包括文件包含、宏命令和条件编译命令等。每个以符号“ # ”开头的行，称为编译预处理行，include 命令为文件包含命令，命令行“ #include<iostream.> ”的作用是在程序编译前将文件 iostream.h包含到程序本身所在的文件中来，iostream.h 是 C++系统定义的一个“ 头文件 ”,它设置了 C++语言的输入输出相关环境，如将例 1-1 中 cin，cout 定义为 C++的标准输入、输出设备标识符。有关 C++语言的编译预处理将在本书的其他章节详细介绍。

3. 程序主体

一般C++ 语言程序是由若干个不同的函数按层次结构组织而成,其基本结构如图 1-11 所示。

```
fun_1()              // 函数 fun_1()的定义
 {

}

fun_1()
 {

}
main()               // main()函数的定义
 {

}

fun_n()              //函数 fun_n()的定义
```

图 1-11　C++语言程序的基本结构

整个程序用函数进行逻辑功能模块的划分，每一个函数完成一个特定的功能。函数是 C++语言程序的基本构成单位，是由 C++语言语句组成的。例 1-1 只有一个主函数 main()，main()函数就是由其后花括号中的若干语句组成的。

例 1-1 的执行结果如下。

```
Please input two integers, separated by space。
3 5<CR>
The result is 15
```

一个能够独立运行的完整的 C++程序必须有一个也只能够有一个主函数 main()，main()函数是执行程序的起点。

1.4.2 C++语言的基本语法单位

从程序编译的角度来看，组成 C++语言程序主体的语言要素主要有以下几个。

1. 标识符

标识符是程序员对程序实体进行定义的字符串。在 C++语言中规定，标识符必须由字母或下划线“_”开头，不超过 31 个字符，用来标识用户定义的常量名、变量名、函数名、类名、文件名等。关于标识符详细的用法见第 2 章。

2. 关键字

C++语言中的具有特定含义的专用单词，如 case, for 等。详细的解释见第 2 章。

3. 运算符号和分隔符号

运算符号如例 1-1 中的符号“>>”从 cin 指向的标准输入设备中输入数据，“<<”向 cout 指向的标准输出设备输出数据，另外还有“*”，“/”，“>”，“&&”等运算符号。详见本书其他章节。

分隔符号主要是程序中的空格符号、制表符号和换行符号。

4. 常量

C++语言中的常量包括整型常量、实型常量、浮点常量和字符串常量。

5. 字符串

字符串是双引号之间的字符序列，也是常量的一种，无论字符串本身代表的含义是什么，当被双引号括起来，就只是一个能够被 C++程序引用的字符常量。

1.5 Visual C++ 6.0 的基本使用

1. 软件安装

将 Visual C++ 6.0 光盘插入光盘驱动器，自动启动或者在光盘目录中找 Setup.exe 文件，双

击该文件图标，启动安装后，按提示画面安装 Visual C++ 6.0 软件，安装完毕后，需重新启动计算机才能使用该软件。

2. 启动 Visual C++ 6.0 软件

在程序菜单中启动或双击在桌面上的 Microsoft Visual C++ 6.0 图标就可启动该软件，启动后，系统显示 Visual C++ 6.0应用界面。如图 1-12 所示。

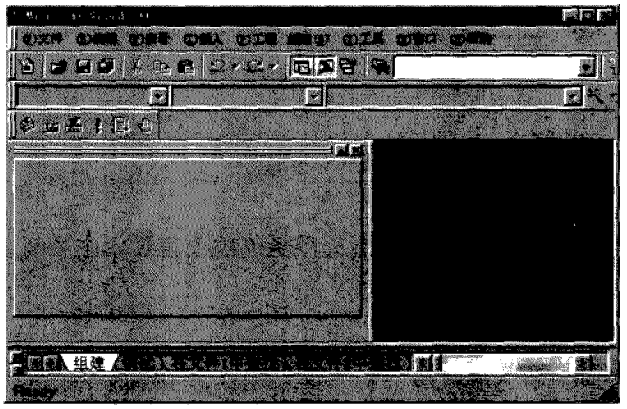

图 1-12 Visual C++ 6.0 应用界面

(1) 从如图 1-12 所示的对话框中选择“文件”（Files）菜单中的“新建”（New）命令，系统显示图 1-13 所示“新建”对话框。

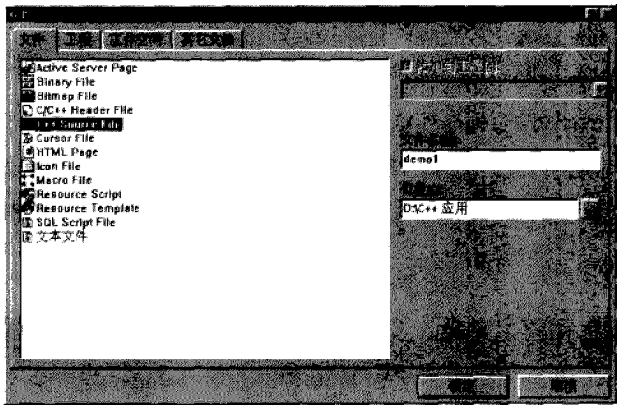

图 1-13 Visual C++ 6.0 “新建”对话框

(2) 在对话框中可以选择“文件”（Files）、“工程”（Projects）、“工作空间”（Workspaces）和“其它文档”（Other Documents）等属性页。在“文件”（Files）属性页中选择‘ C++ Source File ”，按“ OK”键确定以后，显示如图 1-14 所示的 Visual C++ 工作窗口。

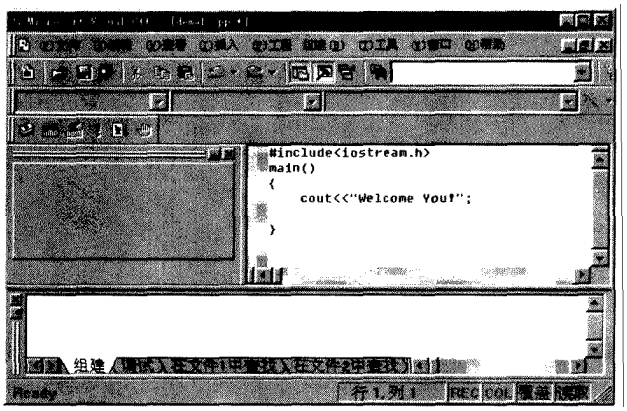

图 1-14　Visual C++ 6.0　工作窗口

(3) 在客户区窗口中输入 C++源程序代码。

(4) 在 C++源程序代码输入完成后，就可以对程序进行编译。如图 1-15 所示，选择“组建”（Build）菜单中的“ ReBuild All ”命令进行 C++源程序的编译，并建立相应的文件。

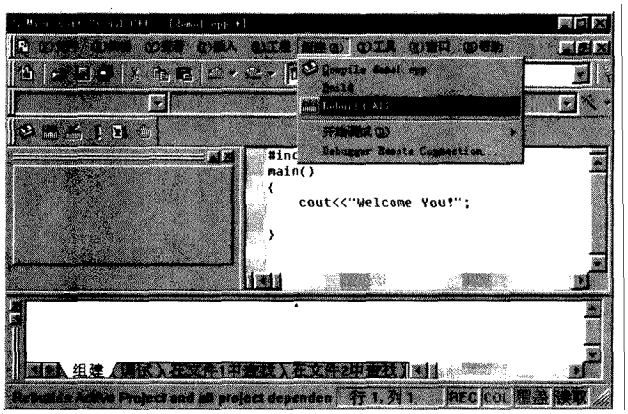

图 1-15　进行程序编译

(5) 如图 1-16 所示，编译的结果将会在客户区窗口输出。

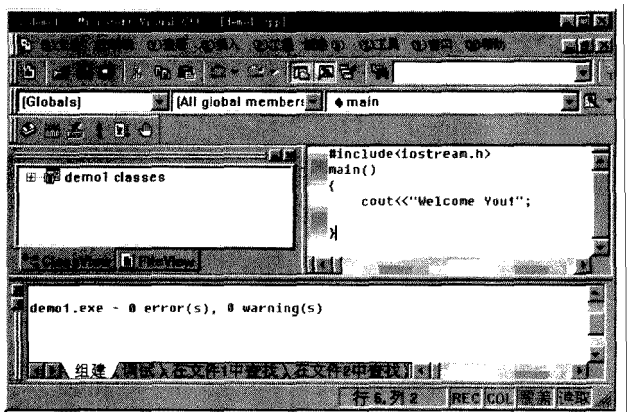

图 1-16　输出编译结果

(8) 如果无错误和警告提示，则可以选择“组建”菜单中的“Execute”命令或按‘Ctrl+F5”组合键执行该程序。若执行的是 MS DOS 程序，Windows 自动切换到 DOS 环境，并在 DOS 窗口列表中列出运行的结果。

Visual C++ 6.0 软件是一个使用比较复杂的可视化程序设计软件，随着课程的学习，请读者参阅相关资料，进一步学习所感兴趣的内容。

习题

1. 叙述计算机程序设计语言的发展历程，并简单叙述面向对象语言的特点。
2. 简单叙述面向对象程序设计的特点。
3. C++如何进行源程序的注释？
4. 简单说明 C++语言有哪些基本的语法要素。
5. 编写程序输出‘Hello 2008”。

第 2 章　C++数据类型与输入输出

C++的数据类型规定了它们的存储表示及可以对它进行的操作运算。 C++语言中，数据处理的基本对象是常量和变量。

操作运算是对数据进行处理。C++语言具有十分强大的运算功能，它既有算术运算、逻辑运算和关系运算等一般的常用运算功能，还可以对数据进行位移操作处理和一些特殊运算。本章介绍数据类型和各种运算的特点，并介绍 C++语言的基本输入输出语句的用法。

2.1　关键字和标识符

关键字和标识符是 C++语言最基本的词法符号，本节对它们加以介绍。

2.1.1　关键字

关键字是 C++语言的保留字,在 C++语言中有特定的含义，不能被重新定义用做其他用途。下面是 C++中的常用关键字。

asm	continue	extern	int	register	switch	unsigned
auto	default	float	long	return	template	virtual
break	delete	for	new	short	this	void
case	do	friend	operator	signed	throw	volatile
char	double	goto	private	sizeof	try	while
class	else	if	protected	static	typedef	
const	enum	inline	public	struct	union	

2.1.2　标识符

标识符是程序员在程序中自己定义的名字，代表变量名、函数名和类型名等。标识符由大小写字母、下划线和 0～9 的数字组成，组成标识符的规则是以字母或下划线开头，其后可跟数字、下划线、零个或多个字母。

标识符的长度原则上可以是任意的，但不同的 C++编译器能识别的最大长度是有限的，编译器自动忽略掉多余的字符。

在 C++编译器中，标示符中的大小写字母是敏感的。编译器可以识别大写和小写字母。如：timer，Timer 和 TIMER 是三个不同的标示符。也就是说，若在程序中有 timer，Timer 和 TIMER 三个标示符，它们代表不同的变量或函数。

标识符应使用有意义的单词或缩写来表明这个标识符的用途，这样就可以使程序具有很好的可读性，为了表达方便，在较长的标识符中，可以使用大小写字母或下划线分隔标识符的单

词，例如：liner_List，lineList，X.point

2.1.3 标点符号

C++语言中的标点符号不表示任何实际的操作。但在程序的某些地方必须使用标点，如分号用于在一条语句之后，表示该语句的结束。 C++中的常用标点符号如表 2-1 所示。

表 2-1 C++中的常用符号

#	( )	{ }	,	;	"	'	:	...

其中 ( ) 和 { } 必须成对使用。C++语言程序中标点符号的使用将会在本书相关章节中结合实例介绍。

2.2 数据类型

数据是对客观现实世界的数学描述，如用字符串表示人名，用“1 ”和‘ 0 ”表 示“ 是 ”与“ 非 ”数据还可以依本身的特点进行分类,不同类型的数据用不同的处理方法,如 整数可以参加算术运算，字符串可以连接、比较等。

编写计算机程序的目的就是要用计算机处理客观世界的现实问题。程序处理的基础就是数据。C++语言中提供了丰富的数据类型和运算。

C++语言提供的数据类型有基本类型、构造类型和指针类型等。

2.2.1 基本数据类型

基本类型包括整形、实型（浮点型包括单精度型、双精度型 ）、字符型等。C++语言的基本数据类型如表 2-2 所示。

表2-2 C++语言的基本数据类型

类 型 名	位 数	表 示 范 围
char	8	−128～+127
signed char	8	−128～+127
unsigned char	8	0～256
int	16	−32 768～32 767
short int	16	−32 768～32 767
signed short int	16	−32 768～32 767
unsigned short int	16	0～65 535
signed int	16	−32 768～32 767
unsigned int	16	0～65 535
long int	32	−2 147 488 648～2 147 488 647
signed long int	32	−2 147 488 648～2 147 488 647
unsigned long int	32	0～4 294 967 295
float	32	3.4×10^{-38}～3.4×10^{38}
double	64	1.7×10^{308}～1.7×10^{308}
long double	80	3.4×10^{-4932}～1.1×10^{4932}
void	0	无值

描述相应类型的关键字是 char，int，float，double 等

字符类型的变量用于保存 ASCII 字符，整型变量用于保存整数，浮点类型和双精度类型的变量用于保存带小数点的数，但双精度类型比浮点类型有更大的表示值域。void 类型在后续章节讨论。

类型前面可以加类型修饰符用于改变基本类型的含义，但它们仍然是基本数据类型。C++的类型修饰符是：signed（有符号的）、unsigned（无符号的）、short（短的）和 long（长的）。修饰符 signed 和 unsigned 可以用于字符类型和整型，short 和 long 可以用于整型，long 还可以用于双精度类型。当用 short、long、signed 或 unsigned 修饰 int 时，关键字 int 可以省略。字符变量不仅可以用于存储 ASCII 字符的代码值，也可以用于存储-128～127 或者 0～255 之间的整数值。

2.2.2 构造类型

构造类型又称为组合类型，它是由基本类型组成的复杂数据类型。构造类型包括结构类型、数组类型、联合型类型、枚举型类型等，含义如表 2-3 所示。

表 2-3 构造数据类型的含义

变量名称	含义
数组	由相同属性数据组成的集合
结构	一种混合的数据结构，它由不同的数据类型构成
联合	类似于结构，也是一种混合的数据结构，但它的成员共用同一段内存
枚举	被命名为整型常量的集合

2.2.3 指针类型

指针类型数据实际上是存储器的地址，用于存储另一数据的地址，在内存中它占一定的存储单元。指针类型数据的引入是 C++语言的特点之一。

2.2.4 空类型

空类型 void 用于说明一个函数不返回任何值。还可以说明指向 void 类型的指针，此时这个指针即可指向各种不同的数据对象。

2.3 常量

常量表示固定的数值或字符值，它的特点是在程序中不进行说明就可以直接使用，但在程序运行过程中不可改变其值。C++语言中的常量类型如下。