C++语言程序设计

尹 燕 编著

C++YUYAN
CHENGXU SHEJI

电子科技大学出版社
University of Electronic Science and Technology of China Press

C++语言程序设计

尹 燕 编著

C++YUYAN
CHENGXU SHEJI

电子科技大学出版社
University of Electronic Science and Technology of China Press

·成都·

图书在版编目（CIP）数据

C++ 语言程序设计 / 尹燕编著. — 成都：电子科技大学出版社，2018.3

ISBN 978-7-5647-5886-8

Ⅰ. ①C… Ⅱ. ①尹… Ⅲ. ①C 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2018）第 051556 号

C++ 语言程序设计

尹 燕 编著

策划编辑 罗 雅

责任编辑 刘 凡

出版发行 电子科技大学出版社

成都市一环路东一段 159 号电子信息产业大厦九楼 邮编：610051

主 页 www.uestcp.com.cn

服务电话 028-83203399

邮购电话 028-83201495

印 刷 成都市火炬印务有限公司

成品尺寸 185mm×260mm

印 张 11.75

字 数 300 千字

版 次 2018 年 3 月第一版

印 次 2018 年 3 月第一次印刷

书 号 ISBN 978-7-5647-5886-8

定 价 38.00 元

前 言

在信息科学技术中，计算机硬件及通信设施是载体，计算机软件是核心。软件是人类知识的固化，是知识经济的基本表征，软件已成为信息时代的新型"物理设施"。人类抽象的经验、知识正逐步由软件予以精确地体现。在信息时代，软件是信息化的核心，国民经济和国防建设、社会发展、人民生活都离不开软件，软件无处不在。软件产业是增长快速的朝阳产业，是具有高附加值、高投入高产出、无污染、低能耗的绿色产业。软件产业的发展将推动知识经济的进程，促进从注重量的增长向注重质的提高方向发展。软件产业是关系到国家经济安全和文化安全，体现国家综合实力，决定21世纪国际竞争地位的战略性产业。

作为计算机软件的主要表现形式，计算机程序不同于其他程序（如音乐会程序），是由计算机来执行的。这就使得计算机程序的编制（程序设计）不能完全以人的思维模式和习惯来进行，它往往要受到计算机解决问题的方式和特点的限制。除此之外，要设计出解决各种问题的程序，程序设计者往往还需要了解与问题领域有关的知识。

从程序设计的发展历史来看，程序设计经历了从低级语言到高级语言、从以编码为中心到面向软件生存周期的软件工程、从过程式到面向对象的发展过程。这一过程体现了人们对程序设计活动的不断认识和改进的过程，特别是从过程式程序设计到面向对象程序设计的发展，体现了人们对以自然的方式来描述和解决问题的需求，它使得解题过程更接近于人的思维方式。

本书所讲述的内容既有对C语言重要内容的学习，又有C++语言的主要内容，并且以讲解面向对象的三大特性（封装性、继承性和多态性）为重点，系统地介绍了C++语言编程所用的全部内容。

本书内容安排由浅入深，讲解方法通俗易懂，以丰富的例题讲解操作方法、验证语法规则，例题针对性强，读者通过学会一个例题，可以掌握一个概念、一种操作方法或一种编程技巧。

本书可作为理工科高校计算机和软件工程或相关专业《C++语言程序设计》的教材，也可作为课程设计、创新实训、毕业设计和创新实践环节的实用指导书，同时也可供其他有关专业人员参考。

全书共分8章。

第1章 C++语言概述，阐述C++语言发展历史与技术特点和C++程序的开发过程。

第2章 数据类型、运算符和表达式。介绍基本数据类型、运算符及表达式、数据类型转换等。

第3章 C++的流程控制，介绍了过程化程序设计中三种基本流程，顺序结构、选择结构和循环结构。

第4章 数组，介绍了数组的结构，以及字符数组的定义和应用。

第5章 指针和引用，介绍了指针和指针变量，内存地址的描述以及动态存储分配等问题。

第6章 函数，重点阐述了函数的定义、调用规则和递归函数的使用。

第7章 类和对象，重点阐述了面向对象的设计思想，类和对象的基本概念，三大特征。

第8章 继承性和派生类，介绍了继承和派生类的思想以及其表示过程。

第9章 输入/输出操作，介绍了C++中的输入流(cin)和输出流(cout)的用法。

第10章 模板，介绍了函数模版和类模板的用法和规则。

第11章 异常处理，阐述了C++异常处理机制和基于断言的程序调试方法。

本书由华东交通大学尹燕担任主编；华东交通大学钟小妹和黄兆华担任副主编；具体编写分工如下：本书由尹燕拟订编写了本书的大纲和目录；尹燕编写第1章、第4章、第5章、第6章、第7章、第8章、第9章；钟小妹编写第2章、第3章、第10章、第11章；黄兆华编写第9章、第10章和第11章。本书由尹燕和钟小妹负责统稿、审校、排版等工作。同时感谢华东交通大学软件学院领导给予的支持和鼓励，感谢华东交通大学教材著作出版基金委员会的基金资助。

在本书的编写过程中，查阅了大量有关C++语言程序设计的文献资料，在此对文献资料的作者表示感谢。尽管作者为本书编写付出了很大努力，并希望本书能成为一部精品，但限于作者水平，加之时间仓促，书中难免有疏漏和不妥之处，敬请广大读者批评指正。

编著者

2018年1月

目 录

第1章 C++语言概述

1.1 计算机语言与程序

人类语言是人与人之间交流信息的工具，而计算机语言是人与计算机之间交流信息的工具。用计算机解决问题时，人们必须首先将解决问题的方法和步骤按照一定的规则和序列用计算机语言描述出来，形成计算机程序，然后让计算机自动执行程序，完成相应功能，解决指定的问题。

1.1.1 机器语言与程序

机器语言是第一代计算机语言。任何信息在计算机内部都是采用二进制代码表示的，指挥计算机完成一个基本操作的指令（称为机器指令）也是由二进制代码表示的，每一条机器指令的格式和含义都是计算机硬件设计者规定的，并按照这个规定制造硬件。一个计算机系统全部机器指令的总和称为指令系统，它就是机器语言。用机器语言编制的程序形式如下所示。

```
0000  0100  0001  0010
0000  0100  1100  1010
1000  1010  0110  0001
.....
```

每一行都是一条机器指令，代表一个具体的操作。机器语言程序能直接在计算机上运行，且运行速度快、效率高，但必须由专业人员编写。机器语言程序紧密依赖于硬件，程序的可移植性差。所谓移植，是指在一种计算机系统下编写的程序经过修改可以在另一种计算机系统中运行，并且运行结果一样。改动越少，可移植性越好；改动越多，可移植性越差。

1.1.2 汇编语言与程序

机器语言是由二进制代码构成的，难以记忆和读写，用它编写程序比较困难。于是计算机工作者发明了汇编语言，用来代替机器语言编写程序。汇编语言是一种符号语言，它用一个有意义的英文单词缩写来代替一条机器指令，如用 ADD 表示加法，用 SUB 表示减法。英文单词缩写被称为助记符，每一个助记符代表一条机器指令，所有指令的助记符集合就是汇编语言。用汇编语言编写的程序形式如下所示。

```
MOV   AL   12D   //表示将十进制数 12 送往累加器 AL
SUB   AL   18D   //表示从累加器 AL 中减去十进制数 18
```

```
HLT                    //表示停止执行程序
```

汇编语言改善了程序的可读性和可记忆性，使编程者在编写程序时稍微轻松了一点。但是汇编语言程序不能在计算机中直接运行，必须把它翻译成相应的机器语言程序才能运行。将汇编语言程序翻译成机器语言程序的过程称为汇编。汇编过程是计算机运行汇编程序自动完成的，如图1-1所示。汇编语言是第二代计算机语言。

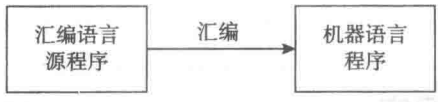

图1-1　汇编过程

1.1.3　高级语言与程序

机器语言和汇编语言都是面向机器的语言，统称为低级语言。它们受特定计算机指令系统的限制，通用性较差，一般只适用于专业人员。非专业人员若想学习使用低级语言编写程序比较困难，为解决这一问题，计算机工作者发明了高级程序设计语言，简称高级语言。高级语言是第三代计算机语言。高级语言用类似于人类自然语言和数学语言的方式描述问题、编写程序。例如，用C++语言编写的程序片段如下所示。

```
int  a,b,c;      //定义变量a、b和c
cin>>a>>b;       //输入变量a、b的值
c=a+b;           //将变量a、b的值相加，结果赋给变量c
cout<<c;         //输出变量c的值
```

该程序片段的功能见每条语句后面的说明。用高级语言编写程序时，编程者不需要考虑具体的计算机硬件系统的内部结构，即不需要考虑计算机的指令系统，而只要告诉计算机“做什么”即可。至于计算机“怎么做”，即用什么机器指令去完成，不需要编程者考虑。

高级语言程序也无法在计算机中直接运行。若要运行高级语言程序，首先必须将它翻译成机器语言目标程序，这个翻译的过程称为编译，编译是由“编译程序”（也称为“编译器”）完成的。然后由“连接程序"将目标程序与系统提供的标准函数的库程序连接，生成可执行程序。可执行程序可以在计算机中运行。编译、连接过程如图1-2所示。“编译程序”和“连接程序”属于计算机系统软件。

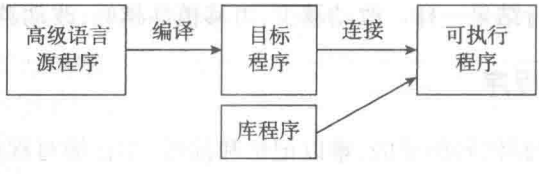

图1-2　编译、连接过程

高级语言不仅易学易用、通用性强，而且具有良好的可移植性。如果想把高级语言程序移植到另一个计算机系统中，只需对源程序稍加改动甚至不改动，使用目标计算机系统的编译程序将高级语言源程序重新编译即可。不同的计算机系统具有不同的编译程序。

目前世界上有数百种高级语言，应用于不同领域，而C++作为其中的优秀语言得到了广泛的使用。

1.2　C ++的发展历史与技术特点

Visual C ++这套功能强大的 Windows 应用程序开发系统是从 Microsoft C/C ++演化而来的,从 Microsoft C/C ++8.0 开始改称为 Visual C ++(简称 VC ++1.0),发展到本书所讲的 Visual C ++6.0,该版本发行至今一直被广泛地用于大大小小的项目开发。目前最新的版本是 Visual C ++2015。Visual C ++6.0 有一套集成开发工具,包括各种编辑器、编译工具、集成调试器等,其主要技术特点有两个。

(1)Visual C ++支持面向对象编程技术,这一技术包装了 Windows 内在的复杂的运行机制,使 Windows 编程变得简单易学。

提到面向对象编程技术,就不得不提 MFC。MFC 是由微软的 AFX 小组开发的,MFC 提供了一整套用于 Windows 应用程序开发的类。1992 年 3 月,MFC 的第一个版本随同 Microsoft C/C ++7.0 诞生了,可见 MFC 比 Visual C ++还要早,也许正是 MFC 的诞生促使 Microsoft C/C ++演变成 Visual C ++。

(2)Visual C ++支持可视化编程,这正是 Visual 名字的由来。Visual C ++提供了 App Wizard 和 Class Wizard 这两个功能强大的可视化编程向导,App Wizard 负责建立应用程序的框架,Class Wizard 则在框架的基础上负责给应用程序添加代码。

同 Microsoft VisualStudio 家族中的 Visual Basic 相比,Visual C ++功能更加强大,开发人员对程序的控制更强,生成的代码效率更高,程序运行的速度更快,但要求开发人员所做的工作也就更多。从使用语言的角度来说,Visual C ++比 Visual Basic 低级,正如同 C 语言与 Basic 语言的关系,因此 Visual C ++适合于专业的软件开发人员使用。

1.3　C ++语言

1.3.1　概述

C ++语言是从 C 语言发展而来。C 语言是一个支持过程式程序设计的高级程序设计语言,它是由贝尔实验室的 Dennis Ritchie 为编写 UNIX 操作系统而设计的一种系统程序设计语言,由于受到 UNIX 被广泛使用的影响,C 语言逐渐成为一种被普遍地应用于各种类型应用程序书写的程序设计语言。C 语言既有高级语言的优点,如提供了类型机制和结构化流程控制成分,同时它又有低级语言(如汇编语言)才具有的一些描述能力,如数据位操作和内存地址操作等。C 语言是一种编译型语言,与其他高级语言相比,C 语言还具有简洁、灵活、高效等特点。

C ++语言是贝尔实验室的 Bjame Stroustrup 为能进行面向对象程序设计而设计的一种高级程序设计语言,它保留了 C 语言的所有成分和特点,并在 C 语言的基础上增加了支持面向对象程序设计的语言成分。另外,C ++也吸收了早期的一些面向对象语言(如 Simula 和 Smalltalk)中许多好的思想。C ++语言既支持过程式程序设计,又支持面向对象程序设计,

它属于一种混合语言。

C ++ 语言也是一种编译型语言，与 C 语言一样，C ++ 语言的主要特点是灵活和高效。C ++ 的灵活性体现在它对使用者的限制较少，易于发挥使用者的创造性；C ++ 的高效体现在它提供的一些语言成分（如指针等）能产生高效的代码，并且，C ++ 很少对程序做运行时刻的合法性检查（如数组下标越界等），从而减少了程序运行时的开销。

从某种程度上讲，C ++ 的特点又是它的缺点，即它对使用者的要求较高。C ++ 的灵活性导致了 C ++ 语言不易把握，特别是对一些使用不当易导致错误的用法不加限制。对于程序设计的初学者来说，他们往往不知道何时使用何种语言成分来解决何种问题，而对语言的使用不当将会造成不良后果。C ++ 的高效则是把保证程序正确运行的责任交给了程序设计者，程序设计者必须对各种可能导致程序运行错误的因素进行仔细考虑和处置。当然，保证程序的正确性是程序设计者义不容辞的责任，不过，这要花费程序设计者很多的精力，增加了他们的负担。而在有些语言中，对可能造成不良后果的一些做法（如指针运算）就根本不提供，从而减少了产生不正确程序的可能性，当然，这是以牺牲灵活性和效率为代价的。

关于 C ++ 语言，对较好地掌握了基本程序设计思想和技术的程序设计“高手”来说，C ++ 是一个很好的语言；而对刚刚从事程序设计的“新手”而言，C ++ 是一个非常糟糕的语言，即评价 C ++ 应该评价使用 C ++ 的人的程序设计素质。因此，只有较好地掌握了程序设计基本思想和技术的人才能很好地使用 C ++ 语言来进行程序设计。

C ++ 是一种使用广泛，支持基本的程序设计思想、概念和技术的程序设计语言，与其他程序设计语言相比，用 C ++ 语言编写程序的效率高，并且它支持多种程序设计范式。本书在介绍 C ++ 语言时，主要强调 C ++ 对基本程序设计思想的支持，而对 C ++ 的一些不利于养成良好编程习惯的做法不予重点介绍，尤其是对属于 C ++ 语言“文化”范畴的内容不予过分强调。

国际标准化组织（ISO）于 1998 年为 C ++ 制定了国际标准（C ++98），并于 2011 年对它进行了修订和扩充，得到新的国际标准（C ++11）。本书中大部分关于 C ++ 的描述是按 C ++11 国际标准给出的，有些描述则是基于微软公司的 Visual C ++ 给出的，而该 C ++ 的实现并没有完全按照国际标准来实施，请读者在阅读本书时注意这一点。

1.3.2 C ++ 程序的构成

逻辑上，一个 C ++ 程序由一些程序实体的定义构成，这些程序实体主要包括：常量、变量、函数、对象以及数据类型（包括类）等。常量和变量是程序所处理的数据；函数是程序对数据的加工过程，这个过程由语句序列来给出；对象是数据以及数据加工的封装体；数据类型用于描述数据的特征，其中的类是把数据及其操作作为一个整体来描述对象。根据在程序中定义位置的不同，数据可分为全局数据、函数的局部数据以及类的成员数据；函数可分为全局函数和类的成员函数；对象可分为全局对象、函数的局部对象以及类的成员对象。每个 C ++ 程序必须有且仅有一个名字为 main 的全局函数，称为主函数，程序从全局函数 main 开始执行。

物理上，一个 C ++ 程序可以存放在一个或多个文件（称为源文件）中，每个源文件包含

一些程序实体的定义,其中有且仅有一个源文件中包含全局函数 main。

下面给出一个简单的 C ++ 程序,它从键盘输入两个数,然后计算这两个数的和,最后把计算结果输出到显示器。

```
//This  is  a  simple  C ++  program
#include  <iostream>  //对 C ++ 标准库中定义的程序实体进行声明
Using  namespace  std;  //使用标准库的命名空间 std
Int  main()  //主函数
{  double  x,y;  //定义两个局部变量 x 和 y
cout << "Enter  two  numbers:";  //通过标准库中定义的对象 cout 往显示器输出提示信息
cin >>x >>y;  //通过标准库中定义的对象 cin 从键盘输入数据给变量 x 和 y
Double z;  //定义一个局部变量 z
z = x + y;  //把 x 和 y 的值相加,结果保存到变量 z 中
cout << x << "+" << y << "=" << z << endl;  //输出计算结果
Return  0;  //主函数返回,程序结束
}
```

上述程序的运行过程如下所示(加下画线的部分为键盘输入的信息,"↙"表示回车符)。

```
Enter  two  numbers:7.2  9.3↙
7.2 +9.3 =16.5
```

1.3.3　C ++ 语言的词法

一个语言包括语法、语义和语用三个方面。这里需要进一步说明的是,语言的语法又包括词法与句法。词法是指语言的构词规则,句法是指由词构成句子(程序)的规则。

1. 字符集

任何一个语言都是由一些基本符号构成的,这些基本符号的集合就构成了相应语言的字符集(Symbol Set)。C ++ 的字符集由 52 个大小写英文字母、10 个数字以及一些特殊符号构成。

(1)大小写英文字母包括:

a ~ z,A ~ Z

(2)数字包括:

0 ~ 9

(3)特殊字符包括:

```
!  #%  ^  &  *  _  -  +  =  ~  <  >  /  \  |  .  ,  :  ;  ?
'  "  (  )  [  ]  {  }  空格  横向制表  纵向制表  换页  换行
```

2. 单词及词法规则

单词(Word)由字符集中的字符按照一定规则构成的具有一定意义的最小语法单位。C ++ 的单词包括:标识符、关键字、字面常量、操作符以及标点符号等。

(1)标识符。

标识符(Identifier)是由大小写英文字母、数字以及下画线(_)所构成的字符序列,第一

个字符不能是数字，如 student、student_name、x_1、_name1 等都是合法的标识符。

标识符通常用作程序实体的名字，程序实体包括：常量、变量、函数、对象、类型（包括类）、标号等。在 C++ 程序中，使用一个标识符前必须要对该标识符进行声明，指出该标识符用于标识何种程序实体。

另外，在使用标识符时应注意以下几点。

①大小写字母有区别，如 abc、Abc 与 ABC 是不同的标识符。

②后面介绍的关键字不能作为用户自定义的标识符，它们有特殊的作用。

③具体实现（编译程序）可能会限制标识符的长度。

④以两个下画线开头或以一个下画线后跟一个大写字母开头的标识符往往在 C++ 语言内部实现中用到（如作为标准库中的全局实体的名字），程序中尽量不要用这些标识符作为自定义的实体的名字。

⑤对不同种类的程序实体最好采用不同风格的标识符，以提高程序的易读性。

关于标识符的风格，不同的编程人员会有不同的命名习惯。下面是本书所采用的书写风格：

①对于符号常量，采用全部大写的标识符，如圆周率 PI；

②对于自定义的类型，采用英语单词的第一个字母大写，如 StudentType；

③对于变量/对象和函数，采用小写字母，如 student、print。

（2）关键字。

关键字（Keyword）是指语言预定义的标识符，它们有固定的含义和作用，在程序中不能用作其他目的。表 1-1 列出了 C++ 中的关键字。

表 1-1　C++ 关键字

asm	auto	bool	break
case	catch	char	class
const	const_cast	contlnue	default
delete	do	double	dynamic_cast
else	enum	explicit	export
extem	false	float	for
friend	goto	if	inline
int	long	mutable	namespace
new	operator	private	protected
public	register	reinterpret_cast	retum
short	signed	sizeof	static
static_cast	struct	switch	template
this	throw	true	try
typedef	typeid	typename	union
unsigned	using	virtual	void
volatile	wchart	while	

除了表 1 -1 列出的关键字外,每个 C ++的实现可能还规定了一些额外的关键字,在使用 C ++时必须参考相应的语言实现参考手册。

(3)字面常量。

字面常量(或称直接量 Literal)是指在程序中直接写出的常量值。如 128、3.14、'A'(字符)、"abcd"(字符串)等。

(4)操作符。

操作符(Operator)用于描述对数据的操作。因为大部分操作符是对数据进行运算,所以又把操作符称为运算符。如 + 、 - 、 * 、/ 、= 、> 、< 、= = 、! = 、> = 、< = 、||、&& 等。

(5)标点符号。

标点符号(Punctuation)起到某些语法、语义上的作用,如逗号、分号、冒号、括号等。

在 C ++程序的书写上,上述的单词有时需要用空白符(White Space Character)把它们分开,使得它们在形式上成为独立的单位。这里的空白符是指空格符、制表符、回车符和注释,其中,注释(Comment)是为了方便对程序的理解而加在程序中的说明性文字信息。C ++提供了以下两种书写注释的方法。

①单行注释:从符号"//"开始到本行结束。

②多行注释:以符号"/*"开始到符号"*/"结束。

注释用于解释程序,它们不构成可执行程序的一部分,编译程序在编译时将忽略程序的注释部分(多行注释被看成是一个空格)。

另外,一个单词如果在一行中写不下(如一个很长的字符串),则可以把它分成几行来写,这时,需要在每一行(最后一行除外)的后面加上一个续行符(Continuation Character)。续行符由一个反斜杠(\)后面紧跟一个回车构成。

3. 语法的形式描述

在有些场合下需要对语言的语法规则进行没有歧义的精确描述,如语言的定义文本等。对一个语言的语法进行精确的描述往往需要采用另一个语法和语义比较简单的语言来完成,相对于被描述的语言而言,该语言称为元语言(Meta Language)。较常用的被用于描述程序设计语言语法的元语言是一种称为 BNF(Backus Normal Form,或 Backus - Naur Form)的描述语言。例如,C ++标识符的构成规则可用 BNF 描述成以下形式。

```
<标识符>    ::= <非数字字符> | <标识符> <非数字字符> | <标识符> <数字字符>
<非数字字符>    ::= _|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|
a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
<数字字符>    ::=0|1|2|3|4|5|6|7|8|9
```

其中,"<""> ""|"和"::="称为元语言符号,它们不属于被描述的语言。"::="表示"定义为";"|"表示"或者";"<标识符>""<非数字字符>"以及"<数字字符>"称为元语言变量,它们代表被描述语言中的语法实体。另外,BNF 也存在一些扩充形式,例如,在一些扩充的 BNF 中增加了:方括号"[]"用于表示其中的内容可有可无,花括号"{}"用于表示其中的内容可以重复出现多次,等等。

由于本书并不介绍 C ++的语言定义文本,并且用 BNF 精确描述的内容有时理解起来

显得比较费劲,所以,在本书中没有采用严格的形式化方法来描述所用到的C++语言成分的语法,而是采用了一种容易理解的混合形式(如简化了的BNF形式加上自然语言)来对C++的语法进行描述。

1.3.4 C++程序的运行步骤

要使得一个C++程序能在计算机上运行,一般要遵循如图1-3所示的步骤。

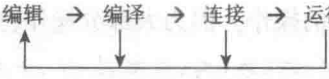

图1-3 C++程序的运行步骤

1. 编辑

编辑(Edit)是指利用某个具有文字处理功能的编辑程序(Editor,如Windows平台上的写字板、记事本、Word等)把C++程序输入计算机中,并作为源程序(Source Code)以纯文本格式保存到外存(如硬盘等)的源文件中。C++源文件的文件名通常为:*.cpp和*.h。

2. 编译

编译(Compile)是指利用某个C++编泽程序对保存在外存中的C++源程序进行翻译,编译结果作为目标程序(Object Code)保存到外存的目标文件中。目标文件的文件名通常为:*.obj。如果一个C++程序由多个源文件构成,则每个源文件可以单独编译,从而生成单独的目标文件。

C++的编译程序中包含了一个编译预处理程序(Preprocessor),用于在编译前对C++源程序中的一些编译预处理命令进行处理,这些编译预处理命令不是C++程序所要完成的功能,而是C++的编译预处理程序在编译之前要做的事情。

3. 连接

一个C++程序可以放在多个源文件中,而每个源文件是分别编译到各自的目标文件的,因此,为了得到一个完整的可执行程序,必须通过一个连接程序(Linker)把这些目标文件以及程序中用到的一些预定义的标准功能所在的目标文件(通常称为库文件)连接(Linking)起来,作为一个可执行程序(Executable Code)保存到外存的可执行文件中。在Windows平台上,可执行文件的文件名通常为:*.exe。

4. 运行

运行(Execute或Run)是指通过操作系统提供的应用程序执行机制,把某个可执行文件中的可执行程序装入内存,让程序运行起来。

在上述的编译、连接和运行过程中都有可能发现程序出错。例如:编译程序在编译时发现源程序中存在语法错误;连接程序在连接目标文件时发现使用的外部实体不存在(外部引用错);程序运行结果与预期的不一致,等等。这些错误都将使得该执行过程返回到前面的阶段并修改错误,然后从改正错误的阶段重新开始执行过程,这个过程可能要重复多次,直到程序产生正确结果为止。

由于上述的C++程序的运行步骤比较麻烦,所以,一些集成的C++程序开发环境相继

出现，如 Visual C ++ 、Turbo C ++ 、C ++ Builder、Dev-C ++ 等。在这些集成开发环境中，运行 C ++ 程序所需要完成的所有操作被集成在一个系统中，从而给操作带来方便。

1.4　C ++ 程序的开发过程

开发一个 C ++ 程序的过程通常包括编辑、编译、连接、运行和调试等步骤。目前有许多软件产品可以帮助完成 C ++ 程序的开发。例如，在 Windows 平台下有 Microsoft 公司的 Visual C ++ 和 Borland 公司的 C ++ Builder；在 Linux 平台下有 GUN 的 GCC 和 GDB 等。本节将以 Microsoft 公司的 Visual C ++6.0（简称 VC6）为工具来介绍 C ++ 程序的开发过程。

1.4.1　编辑

编辑是 C ++ 程序开发过程的第一步，它主要包括程序文本的输入和修改。任何一种文本编辑器都可以完成这项工作。

在 VC6 集成开发环境中，用户可以使用编辑窗口来进行 C ++ 程序的编辑工作。VC6 的编辑窗口是专门为编辑 C ++ 程序而设计的，它提供了包括语法亮色、调用提示、自动缩进、查找和替换等在内的一系列功能，使用起来十分方便。

当用户完成了 C ++ 程序的编辑时，应将输入的程序文本保存为以 .cpp 为扩展名的文件（保存 C ++ 头文件时应以 .h 为扩展名）。

1.4.2　编译

C ++ 是一种高级程序设计语言，它的语法规则与汇编语言和机器语言相比更接近人类自然语言（如英语）的习惯。然而，计算机能够“看”懂的唯一语言是机器指令。因此，要让计算机“看”懂一个 C ++ 程序，就必须使用一种叫做“编译器”的工具，将这个 C ++ 程序“翻译”成机器指令。

编译器所做的工作实际上是一种由高级语言到机器指令的等价变换。用户提供给编译器的输入信息称为源程序代码，它是用某种高级语言编写的程序文本。编译器对源程序代码进行一系列处理后最终产生的输出结果称为目标代码，它是某种计算机的机器指令，并且在功能上与源程序代码完全等价。保存源程序代码和目标代码的文件分别称为源程序文件和目标文件。由源程序文件到目标文件的转换过程就称为编译。

实际上，在进行编译之前还要完成一项称为“预处理”的工作。它的目的是根据程序中的预处理命令对源程序代码做出相应的处理，或为编译器提供一些提示信息。程序中的注释在预处理过程中被删除掉，因此注释中的内容除了方便阅读之外，不会对程序运行产生任何影响。

在 VC6 集成开发环境中，用户可以使用编译（Compile）命令将一个以 .cpp 为扩展名的 C ++ 源程序文件转换成一个以 .obj 为扩展名的目标文件。如果一个 C ++ 程序由多个源程序文件组成，应将它们分别进行编译形成多个目标文件。

1.4.3 连接

要将编译器产生的目标代码变成可执行程序还需要最后一个步骤——连接。连接工作是由“连接器”完成的,它将编译后产生的一个或多个目标文件与程序中用到的库文件连接起来,形成一个可以在操作系统中直接运行的可执行程序(如 Windows 下的.exe 文件)。

在 VC6 集成开发环境中,用户可以使用生成(Build)命令来进行目标文件的连接工作。另外,在源程序文件没有被编译的情况下直接使用此命令,可以使编译和连接工作一起完成。

1.4.4 运行和调试

如果源程序文件中存在着语法错误或连接错误,编译器和连接器会分别给出相关的错误信息。这时需要重新编辑源程序文件,改正其中的错误,再一次进行编译和连接,重复此过程直到错误信息不再出现为止。

在编译和连接工作成功地完成之后可以运行得到的可执行程序,观察程序是否符合期望的运行结果。在 VC6 集成开发环境中,用户可以使用执行(Execute)命令来运行程序。

如果程序的运行结果不是所期望的结果,说明源程序文件中存在着语义错误。这时,需要使用调试器对可执行程序进行跟踪调试来查找错误发生的原因。在 VC6 集成开发环境中,用户可以很方便地进入调试状态,对程序进行设置断点、单步执行、观察变量等操作。

一般地,开发一个 C++程序的整体流程可用图 1-4 表示。

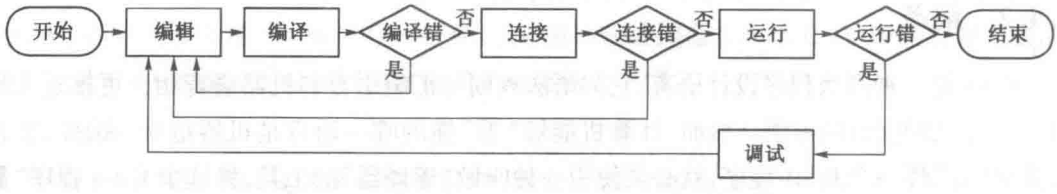

图 1-4 C++程序的开发流程