



软件工程师 宝典系列

C++

程序设计

李 钢 张战军 于秋生 等编著

- 提纲挈领，萃取C++语言语法要旨
- 知识、实例、练习行文，实用、实效
- 语言间横向对比，拓展编程技能
- 轻松实战操作演练，培植个性能力
- 编译、连接、调试深度剖析，简明、经典
- 近**150**个实例精解剖析，让你轻松步入顶尖软件工程师殿堂



科学出版社
www.sciencep.com

软件工程师
宝典系列

C++

程序设计

李 钢 张战军 于秋生 等编著

内 容 简 介

这是《软件工程师宝典》系列图书之一，本书是学习 C++ 语言的实用参考工具书，比较全面地讲解了 C++ 的基础知识及实例应用。

从面向对象的基本概念讲起，首先对面向对象编程思想有一个直观的了解，然后讲解 C++ 的基本语法、数据结构和基本程序结构。此外，本书还讲述了 C++ 中流和基本输入/输出流的知识。学习 C++，最重要的就是理解基本概念，例如面向对象、数据类型、数组、指针、链表、函数等。本书结合大量实例，分别讲述了这些重要的知识点。每部分都有详细的开发实例，章末附录若干习题，书后附有习题参考答案，便于初学者学习、实践。实用性、可操作性强，能够有效地提升读者的开发技能和经验。

书中有些实例来自实际项目，读者可以参考使用。本书适合有一定编程经验的 C++ 初学者，以及想要在 C++ 编程经验上得到快速提高的编程人员，同时也是社会培训班选择的理想教材。

本书部分实例源代码可免费从 www.bhp.com.cn 下载。

图书在版编目 (CIP) 数据

C++ 程序设计 / 李钢 张战军 于秋生等编著. —北京: 科学出版社, 2008

(软件工程师宝典系列)

ISBN 978-7-03-022794-2

I. C++... II. ①李... ②张... ③于... III. C++ 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2008) 第 124093 号

责任编辑: 但明天 / 责任校对: 高 雅
责任印刷: 媛 明 / 封面设计: 久久度设计

科 学 出 版 社 出 版

北京东黄城根北街 16 号

邮政编码: 100717

<http://www.sciencep.com>

北京媛明印刷厂印刷

科学出版社发行 各地新华书店经销

*

2008 年 10 月第 一 版 开本: 787×1092 1/16
2008 年 10 月第一次印刷 印张: 22 1/4
印数: 1—3000 字数: 512 772

定价: 32.00 元

总 序

计算机硬件技术的发展日新月异，CPU 几乎是按照摩尔定律进行快速更新的，随着时间的推移，其运算速度呈级数级地增长。同样，计算机软件技术也在突飞猛进地发展，无论是操作系统还是应用软件。

操作系统的开发是从第一代软件工程师在计算机硬件基础上进行的最低层的二进制编码开始的，到后来逐步发展到用户不需要熟悉低层计算机指令即可进行操作的 DOS 系统，再发展到所见即所得的 Windows 桌面操作环境，以及功能强大的 Linux、UNIX 系统。经过多年的发展，操作系统的功能及作用已经发生了根本性的变化。

同样，运行在操作系统之上的各种应用软件也在发生着根本性的变化，从以前需要对硬件有深刻了解才能编好程序的汇编语言开始，到后来广泛使用的高级程序语言 Basic、C、Pascal、FORTRAN 等，再到现在被广泛使用的 Visual Basic、Visual C++、Delphi、Java 语言及各种 Web 语言等，软件的操作越来越方便，功能却越来越强大，从而使软件工程师的编程变得越来越简单，不过要了解的知识也空前庞大。

随着信息社会的到来及对无纸办公需求的增加，现代社会对软件工程师的需求大量增加，仅我国软件工程师的缺口就在数十万以上。软件人才的培养决定了信息社会实现的程度及社会发展的速度，所以适时、合理地大力培养一些优秀的软件开发人才，对于我国信息化产业的发展必将起到举足轻重的作用。

正是基于这样的计算机软件发展背景及信息化发展需求的考虑，我们经过精心策划、周密设计，组织最优秀的一些作者，编著出版了一套常用计算机软件及操作系统的系列图书，希望使之成为软件人才培养及推动信息化产业发展的“宝典”。这套图书起点低，使读者入门快。同时每本书的内容都很实用，案例丰富简练，与基础知识一一对应。图书按照软件人才培养的规律，尽量使内容讲解由易到难、深入浅出。

该套丛书一改过去基础类图书中重说教、轻操作、过于注重理论及概念讲解的弊端，以一种全新的边操作、边熟悉、边学习的方式吸引读者深入学习下去。每本书在精心挑选并巧妙设计大量案例的基础上，将基础知识的讲解融合到案例的练习中，二者相辅相成，结合紧密。

通过案例的操作与分析，可以加深对基础性概念的理解，并以大量的上机操作来巩固所需要掌握的基本知识点。在内容安排上步步为营、循序渐进，在将基础知识学扎实、学透彻的前提下，使读者的软件编程技术也能够迅速得到提高，使他们的技术水平能够尽快地更上一层楼。

本系列图书面向的读者群体非常广，而对读者的要求又非常低，他们只要对计算机的基本操作熟悉，就能够学好本系列图书中的每一本，从而可以快速掌握对应的每一种软件。读者可以是大学、中专院校的在校学生，各种计算机培训班的学员，社会各个机构中需要培

训的在岗人员，以及所有对软件技术有兴趣的计算机爱好者，本系列图书都将成为他们计算机软件学习道路上最好的良师益友，最得心应手的“宝典”，最能够提高他们的学习效率及促进技术进步的法宝。

我们非常希望本系列图书及时出版，能够为所有使用该系列图书的读者奉献上一套精美丰盛的“计算机软件学习”满汉全席。

丛书编委会

前 言

C++是目前世界上应用最为广泛的计算机高级编程语言之一。C++是一种高效实用的程序设计语言，使用它既可以设计面向过程的程序，也可以设计面向对象的程序。面向对象是C++的一大特色，它实现了类的封装、继承和多态，这些特性使得C++的代码具有高度的可维护性和重用性。

由于C++与C语言的特殊关系，使得学过C语言的读者看到C++的代码非常亲切，因为C++的程序结构和C语言的程序结构是一致的。本书是基于读者没有学习过C语言的假设条件下编写的，但是对于学习过C语言的读者同样适用。

学习程序设计，最重要的一点就是要学习编程的思想，本书的第3章简单介绍了面向对象的思想，这是所有面向对象程序设计语言的基础。从第4章起，分别讲述了C++面向对象的基础知识，并结合具体实例讲述了C++中的对象、类、构造函数、析构函数、继承等重要概念，希望读者多加用功，真正理解面向对象的思想，多上机实践。

在本书的最后一章，讲述了Visual C++开发环境。

本书是C++程序设计的入门书籍，不会着力于深度分析C++语言的特性，但会结合实例讲述面向对象的基本概念、C++的基本概念，力求使读者能够通过阅读本书初步了解C++程序设计语言，并能利用C++语言解决简单的问题。

本书的每一章开始都列举出本章学习的重点，并且每章后面都有针对该章内容的练习题，特别是程序题，读者一定要上机亲自实践。

学习程序设计要多读、多练。多读别人的程序，多上机练习。预祝各位读者能顺利地学好C++的基础知识，并为以后深入学习C++打好基础。

本书编写过程中，赵磊、唐兵、刘旭、范荣、杜江、张俊岭、张江涛、冉林仓、刘伟、梁斌、张海霞、于丙超、范翠丽、薛年喜、杨小勇、刘秋红等编写了部分章节的内容，在此一并表示感谢。

由于编者水平有限，书中难免有不当之处，敬请广大读者批评指正。

编 者

目 录

第 1 章 C++语言基础.....	1
1.1 C/C++概述.....	1
1.2 C++程序的基本结构.....	2
1.3 C++的基本数据类型.....	4
1.4 标识符.....	7
1.5 关键字.....	8
1.6 运算符与表达式.....	8
1.7 常量.....	11
1.8 变量.....	12
1.9 枚举类型.....	13
1.10 上机操作.....	15
1.10.1 经典的 Hello World 程序.....	15
1.10.2 常量与变量练习.....	16
1.10.3 运算符与表达式练习.....	18
1.11 Visual C++ 6.0 开发环境简介.....	20
1.11.1 Visual C++ 6.0 中文版开发环境.....	20
1.11.2 利用 Visual C++ 6.0 开发 Hello World 程序.....	20
1.12 小结.....	22
1.13 习题.....	23
第 2 章 C++面向对象基础.....	24
2.1 面向对象的思想.....	24
2.2 类和对象.....	25
2.2.1 对象.....	25
2.2.2 类.....	25
2.3 构造函数.....	29
2.4 析构函数.....	34
2.5 继承与派生.....	36
2.6 友元.....	39
2.7 上机操作.....	41
2.7.1 重载构造函数.....	41
2.7.2 利用析构函数判断数值大小.....	42
2.8 小结.....	44

2.9 习题.....	44
第3章 C++程序的基本控制结构	48
3.1 程序结构知识.....	48
3.2 顺序结构.....	48
3.3 选择结构.....	50
3.3.1 if 结构.....	50
3.3.2 switch 结构.....	54
3.4 条件运算符?:.....	58
3.5 循环结构.....	60
3.5.1 while 语句.....	60
3.5.2 do...while 语句.....	63
3.5.3 for 语句.....	67
3.6 循环嵌套.....	71
3.7 break 语句.....	72
3.8 continue 语句.....	74
3.9 goto 语句.....	75
3.10 exit()和 abort()函数.....	77
3.11 对4种循环的比较.....	80
3.12 枚举类型.....	80
3.13 上机操作.....	81
3.14 小结.....	85
3.15 习题.....	85
第4章 C++函数	88
4.1 主函数.....	88
4.2 函数的定义.....	89
4.3 局部变量.....	93
4.4 全局变量.....	95
4.5 变量的存储类型.....	99
4.6 函数的调用.....	106
4.7 函数的递归调用.....	108
4.8 函数的嵌套调用.....	112
4.9 内联函数.....	115
4.10 函数重载.....	118
4.11 上机操作.....	122
4.12 小结.....	127
4.13 习题.....	128
第5章 存储类型、生命周期和头文件	131

5.1	外部存储类型.....	131
5.2	静态存储类型.....	132
5.3	作用域与可见性.....	133
5.4	头文件.....	134
5.5	小结.....	136
5.6	习题.....	136
第 6 章	数组.....	137
6.1	数组的定义.....	137
6.2	访问数组元素.....	138
6.3	数组的初始化.....	141
6.4	一维数组的定义和使用.....	143
6.5	二维数组的定义和使用.....	147
6.6	字符数组.....	149
6.7	字符串.....	151
6.8	常用字符串处理函数.....	153
6.9	上机操作.....	161
6.9.1	冒泡法排序.....	161
6.9.2	矩阵加法.....	163
6.9.3	用顺序法查找指定数值.....	164
6.9.4	用二分查找法查找数组中的值.....	167
6.9.5	统计学生考试成绩并给出评价.....	169
6.9.6	区分字符数组中的大小写字母.....	171
6.10	小结.....	172
6.11	习题.....	172
第 7 章	指针.....	176
7.1	指针的概念.....	176
7.2	指针变量.....	178
7.3	指针变量赋值.....	180
7.4	指针运算.....	182
7.5	指针数组与指向数组的指针.....	185
7.5.1	指针数组.....	186
7.5.2	指向数组的指针.....	186
7.6	字符串指针与指向字符串的指针.....	187
7.7	指针函数.....	191
7.8	main 函数的参数.....	193
7.9	main 函数的返回值类型.....	195
7.10	上机操作.....	196

7.11 小结.....	199
7.12 习题.....	199
第8章 结构体、链表与联合	203
8.1 结构体的概念.....	203
8.2 结构体的定义和使用.....	204
8.3 使用结构体变量.....	206
8.4 结构体的初始化.....	206
8.5 结构体数组.....	208
8.6 链表的概念.....	210
8.7 节点.....	210
8.8 创建单链表.....	211
8.9 向链表插入节点.....	213
8.10 删除链表中的节点.....	216
8.11 联合的概念.....	219
8.12 引用联合变量.....	221
8.13 上机操作.....	222
利用结构体录入及输出学生信息.....	222
8.14 小结.....	225
8.15 习题.....	225
第9章 输入/输出流.....	230
9.1 输入/输出流概述.....	230
9.2 cout 流.....	231
9.3 cin 流.....	231
9.4 上机操作.....	232
9.5 小结.....	233
9.6 习题.....	233
习题答案.....	235
第1章答案.....	235
第2章答案.....	236
第3章答案.....	237
第4章答案.....	238
第5章答案.....	240
第6章答案.....	242
第7章答案.....	242
第8章答案.....	244
第9章答案.....	246
附录 A 综合练习.....	247

第 1 章 C++ 语言基础

本章学习目标

- ◆ C++程序的基本结构
- ◆ C++的基本数据结构
- ◆ C++的标识符
- ◆ C++的关键字

任何一种编程语言都有其语言基础知识，包括程序的基本结构、数据类型、关键字、标识符等。任何程序都是按照其语言基础组织起来的代码序列。本章将讲述 C++程序的基本结构，重要的 C++基本数据结构、标识符和关键字等知识。

1.1 C/C++概述

本节先来讲 C/C++的异同。C++是 C 的一个超集，在 20 世纪 80 年代，贝尔实验室的本贾尼·斯特劳斯特鲁普（Bjarne Stroustrup）博士及其同事为了克服 C 语言的一些固有缺点（比如类型检查机制较弱），在 C 语言的基础上对其进行了改进和扩充，引入了类的概念，那时的 C++被称为带类的 C（C With Classes）。其后，本贾尼·斯特劳斯特鲁普博士及其同事不断把新的特性增加进来，例如虚函数(virtual function)、运算符重载（operator overloading）、多重继承（multiple inheritance）等。1998 年国际标准化组织 ISO（International Standards Organization）正式发布了 C++语言的国际标准 ISO/IEC:98-14882，随后各软件厂商推出的 C++编译器都支持该标准，并进行了不同的拓展。

C++支持面向对象的程序设计方法，特别是在大型的软件开发项目中，C++在软件的可重用性、可扩展性、可维护性、可靠性方面等都具有巨大的优势。然而，C++与 C 语言的设计原则是不一样的，C 是 C++的子集，是面向过程的程序设计语言，而 C++则是 C 的超集，是面向对象的程序设计语言。在 Effective C++第三版中，C++被描述成由 4 个子语言组成：

- C 子语言。
- 面向对象的 C++。
- 泛型编程语言。C++拥有强大的模板功能，使其能在编译期完成许多工作，从而大大提高运行期效率。
- STL。C++标准模板库，其安全性和规范性非常强大。

可以认为，C++是一门独立的程序设计语言，学习 C++时可以完全不用学习 C 语言，

并且在大多数场合，C++完全可以代替 C 语言。然而，在需要直接对硬件进行操作（比如编写硬件驱动程序），我们仍然需要使用 C 语言。

1.2 C++程序的基本结构

下面先来看一个例子，通过这个例子再来分析 C++程序的基本结构。

实例 1-1 的程序代码如下，为方便表达，我们在每一个代码行的开头都加了行号，这些行号在实际编程中是没有的。

 实例 1-1 在屏幕上输出字符

```
1: #include <iostream.h>
2: void myFunction();

3: int a=1,b=2;
4: main()
5: {
6:     cout<<" Function main:a="<<a<<"\n";
7:     cout<<" Function main:b="<<b<<"\n";
8:     cout<<" Function main:a+b="<<a+b<<"\n";
9:     myFunction();
10: }

11: void myFunction()
12: {
13:     int c=3;
14:     cout<<" Function myFunction:a="<<a<<"\n";
15:     cout<<" Function myFunction:b="<<b<<"\n";
16:     cout<<" Function myFunction:c="<<c<<"\n";
17:     cout<<" Function myFunction:a+b-c="<<a+b-c<<"\n";
18: }
```

下面来分析一下这个程序的结构，其中涉及一些名词，不甚理解也没有关系，本书以后的章节中会陆续对这些名词做出解释。

- 处理命令 实例 1-1 的第 1 行是一条预处理命令，其作用是将头文件 `iostream.h` 包含进程序中，以实现程序的基本输入/输出操作。
- 函数 C++程序是由若干个文件组成的，每个文件又由若干个函数组成，函数与函数间是平行的，函数间可以调用。在一个程序中，必须有一个主函数 `main()`。实例 1-1 的第 2 行声明了一个自定义函数 `myFunction()`，从程序的第 11 行到第 18 行是这个自定义函数的函数体，第 9 行调用这个函数。第 4 行到第 10 行是主函数体，这是每个程序所必需的。
- 输入/输出 实现与程序内部进行信息交流，几乎每个 C++程序都会用到输入/输出。实例 1-1 的第 6~8 行以及第 14~17 行的作用就是在屏幕上输出相关的字符。
- 变量 几乎所有 C++程序都会用到变量，变量在使用前必须先声明。实例 1-1 的

第3行和第13行就分别声明了3个变量a、b和c，并为它们赋予初始值a=1、b=2、c=3。第6、第7、第14和第15行直接将这些变量的值输出到屏幕上。第8和第17行还对这些变量进行了简单运算，并把运算结果输出到屏幕上。

- 语句 语句是组成程序的基本单元，上面提到的函数就是由若干条语句组成的，C++程序中的语句以分号作为结束符。语句有表达式语句、空语句、复合语句、分支语句、循环语句等。
- 表达式 由常量、变量、函数和运算符组合起来的式子。表达式具有一个值及其类型，它们等于计算表达式所得结果的值和类型。单个常量、变量、函数可以看作表达式的特例。实例1-1的第8和第17行的a+b和a+b-c就是两个表达式。

程序的运行结果如下：

```
Function main:a=1
Function main:b=2
Function main:a+b=3
Function myFunction:a=1
Function myFunction:b=2
Function myFunction:c=3
Function myFunction:a+b-c=0
```

上面分析了一个典型的C++程序的基本结构，下面再给出一个例子，请读者自行分析这个C++程序的基本结构。



实例1-2 通过函数调用，求两个数值加、减、乘、除的值

```
#include <iostream.h>

void funAdd();
void funSub();
void funMul();
void funDiv();

int a=2,b=4;
main(){
    cout<<"=====Start===== "<<"\n";
    cout<<"a = "<<a<<" , b = "<<b<<"\n";
    funAdd();
    funSub();
    funMul();
    funDiv();
    cout<<"=====End===== "<<"\n";
    return 0;
}

void funAdd(){
    cout<<"a + b = "<<a+b<<"\n";
}

void funSub(){
```

```

    cout<<"b - a = "<<b-a<<"\n";
}

void funMul(){
    cout<<"a * b = "<<a*b<<"\n";
}

void funDiv(){
    cout<<"b / a = "<<b/a<<"\n";
}

```

程序的运行结果是：

```

=====Start=====
a = 2,b = 4
a + b = 6
b - a = 2
a * b = 8
b / a = 2
=====End=====

```

实例 1-2 的基本语法和程序结构与实例 1-1 类似，请读者自行分析该程序的结构。另外，有兴趣的读者可试着修改这个例程中的自定义函数 `funDiv()`，例如将 `b/a` 变成 `a/b` 是什么效果，至于原因我们会在以后的章节中讲到。

提示：为了使读者能够在运行本书的例子时清晰了解程序的内容，本书的每一个例子都会在输出部分详细描述本例的功能。

1.3 C++的基本数据类型

所谓数据类型就是按照被定义变量的性质、表现形式以及占存储空间的多少来划分。在 C++ 中，每个变量在使用之前都必须定义其数据类型。

基本数据类型最主要的特点是：其值不可以再分解为其他类型。也就是说，基本数据类型是自我说明的。

C++ 的基本数据类型包括：

- 整型（int） 用于定义整数对象。
- 字符型（char） 用于定义字符数据。
- 浮点型（float, double） 用于定义实数，包括单精度和双精度类型。
- 空类型（void） 这是一个比较特殊的类型，void 描述了一个空集，变量不能被定义成 void 类型，主要用来声明没有返回值的函数。另外，void 也用来声明未确定类型或任意数据类型的指针。

在 C++ 中，可以用一些修饰符来对数据类型做进一步的修饰，包括：

- short 短类型，例如 short int 表示短整型。
- long 长类型，例如 long int 表示长整型。

- **signed** 有符号类型，例如 **signed short int** 表示有符号短整型。
- **unsigned** 无符号型，例如 **unsigned short int** 表示无符号短整型。

表 1-1 为 C++ 部分基本数据类型的内存空间占用情况和取值范围。

表 1-1 C++ 基本数据类型

基本数据类型分类	标识方法	内存占用（字节）	取值范围
整型	int	4	-2147483648~2147483647
短整型	short int	2	-32768~32767
有符号短整型	signed short int	2	-32768~32767
无符号短整型	unsigned short int	2	0~65535
长整型	long int	4	-2147483648~2147483647
有符号长整型	signed long int	4	-2147483648~2147483647
无符号长整型	unsigned long int	4	0~4294967295
单精度浮点型	float	4	$-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$
双精度浮点型	double	8	$-1.7 \times 10^{308} \sim 1.7 \times 10^{308}$
字符型	char	1	-128~127
有符号字符型	signed char	1	-128~127
无符号字符型	unsigned char	1	0~255

下面我们举个例子来说明不同的数据类型的不同取值范围。

实例 1-3 不同数据类型不同取值范围

```
#include <iostream.h>

void myFunShort();
void myFunInt();
void myFunStart();

void myFunShortInt(){
    short int a,b,c;
    a = 32766;
    b = a + 1;
    c = a + 2;
    cout<<" b = "<<b<<"\n";
    cout<<" c = "<<c<<"\n";
}

void myFunInt(){
    int a,b,c;
    a = 32767;
    b = a + 1;
    c = a + 2;
    cout<<" b = "<<b<<"\n";
    cout<<" c = "<<c<<"\n";
}
```

```
void myFunStart(){
    cout<<"*****"<<"\n";
}
```

```
main(){
    myFunStart();
    myFunShortInt();
    cout<<"\n";
    myFunInt();
    myFunStart();
}
```

程序运行结果如下:

```
*****
b = 32767
c = -32768

b = 32768
c = 32769
*****
```

上面的例子可以说明以下几个问题:

(1) 不同的数据类型具有不同的取值范围。在自定义函数 `myFunShortInt` 中, 我们定义了 `short int` 类型的变量 `a`、`b` 和 `c`, 由于 `short int` 类型的取值范围是 $-32\,768 \sim 32\,767$, 因此我们给变量 `a` 赋值为 `32\,766`, 然后把变量 `b` 的值设为 `a+1`, 变量 `c` 的值设为 `a+2`。通过程序运行的结果可以看到, 变量 `b` 的值是 `32\,767`, 并没有超过 `short int` 的取值范围, 而变量 `c` 的值应该是 `32\,769`, 但这个值已经超出了 `short int` 的取值范围, 所以实际得到变量 `c` 的值并不是 `32\,769`, 而是 `-32\,768`。

在自定义函数 `myFunInt` 中, 我们定义了 `int` 型的变量 `a`、`b` 和 `c`, 由于 `int` 的取值范围是 $-2\,147\,483\,648 \sim 2\,147\,483\,647$, 给变量 `a`、`b`、`c` 赋予与 `myFunShortInt` 相同的值, 所以这个函数计算变量 `c` 的值就是 `32\,769`, 证明在这个函数中, 变量 `c` 并没有超出 `int` 的取值范围。

(2) 主函数并没有做具体的计算工作, 而是调用不同功能的函数。在一个程序中, 绝大部分具体目标的实现都是通过不同的功能函数来实现的, 主函数一般不做这类工作。

(3) 变量具有作用域, 正如在函数 `myFunShortInt` 和 `myFunInt` 中都定义了变量 `a`、`b` 和 `c`, 并且在这两个函数中这些变量的类型不一致, 通过程序运行的结果可以看到, 它们之间互不影响, 也就是说 3 个变量都只在自己所在函数中有效, 离开所在函数就不存在了, 这种变量称为局部变量。在实例 1-1 和实例 1-2 中, 我们也定义了变量 `a` 和 `b`, 但这两个变量在程序文件的主函数和所有自定义函数中均有效, 这种变量称为全局变量。

下面再来看一个较为复杂的例子。

实例 1-4 根据输入数值判断要调用的函数

```
#include <iostream.h>

void myFun1();
```

```

void myFun2();
void myFunStar();

short int a,b;
void myFun1(){
    cout<<" a - b = "<<a-b<<"\n";
}

void myFun2(){
    cout<<" b - a = "<<b-a<<"\n";
}

void myFunStar(){
    cout<<"*****"<<"\n";
}

main(){
    cout<<" a = ";
    cin>>a;
    cout<<" b = ";
    cin>>b;
    myFunStar();
    if (a > b)
        myFun1();
    else
        myFun2();
    myFunStar();
}

```

程序运行结果如下：

```

a = 1
b = 2
*****
b - a = 1
*****

```

这个例子相对于之前看到的程序比较复杂，因为它增加了一个条件判断语句 `if...else`，这个语句的作用是判断表达式的条件是否成立，如果条件成立则执行 `if` 后面的语句，否则执行 `else` 后面的语句。

关于 `if...else` 结构，我们会在后面的章节中详细讲解，现在只须知道它的意思即可。另外，这个例子还使用了 `cin` 来读取用户从键盘输入的数值。读者请自行运行这个程序，并分析程序运行的结果。

1.4 标识符

所谓标识符，就是指程序员所起的名字，用来标识变量、常量、函数、数组、类、对象等。C++中有效的标识符构成规则如下：