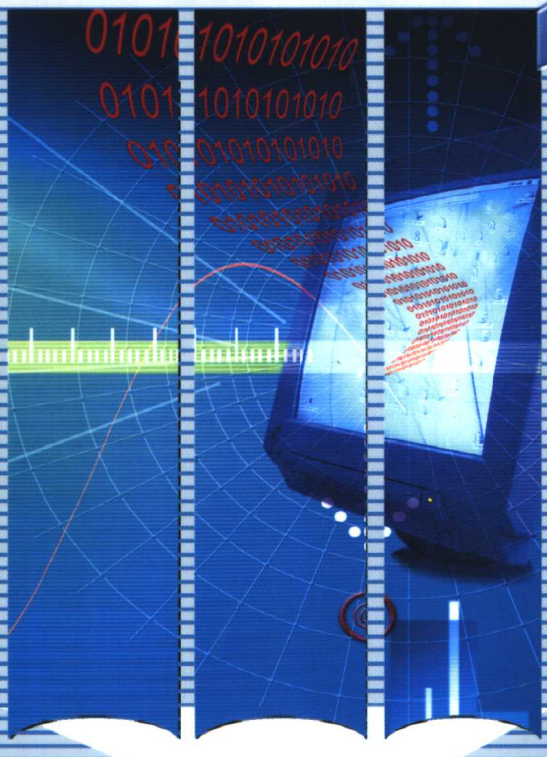
计算机基础知识复习指南与题解系列丛书

C语言
复习指南与题解

李俊杰 编著

全面提升水平的应试辅导书

- 涵盖知识重点
- 精选典型实例
- 题解详尽透彻



清华大学出版社

计算机基础知识复习指南与题解系列丛书

C语言复习指南与题解

李俊杰　编著

清华大学出版社
北　京

内 容 简 介

本书通过介绍C语言的基本概念和大量的习题让读者掌握C语言。为满足广大读者学习C语言程序设计知识、熟练掌握编程技巧、备考和应考的需要，本书系统介绍了C语言的基本语法，包括数据类型及运算规律、基本语句、结构控制、数组和函数等，并介绍了编译预处理、指针、结构体、共用体、位运算和文件操作等较深层次的内容。本书根据当前学生的学习特点，紧扣C语言教学大纲的要求精心编写。

在本书的编写过程中，充分考虑了计算机等级考试的性质和学生学习及应试的特点，列出了各章的重点、难点及主要内容，其中绝大部分习题和例题都是平常学习C语言的一些典型例子。其目的是帮助读者在学习过程中把握重点，掌握典型题目，有的放矢地学习，以便在各类考试中发挥出应有水平，取得满意效果。

本书适合开设C语言课程的高等院校学生使用，也可作为C语言自学者及备考者的参考用书。

图书在版编目(CIP)数据

C语言复习指南与题解/李俊杰编著.—北京：清华大学出版社，2003

(计算机基础知识复习指南与题解系列丛书)

ISBN 7-302-07380-5

Ⅰ.C… Ⅱ.李… Ⅲ.C语言—程序设计—高等学校—自学参考资料 Ⅳ.TP312

中国版本图书馆CIP数据核字(2003)第091029号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：张雪峰

文稿编辑：许瑛琪

封面设计：陈刘源

印 刷 者：北京密云胶印厂

装 订 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：23.75　**字数**：561千字

版　　次：2003年11月第1版　2003年11月第1次印刷

书　　号：ISBN 7-302-07380-5/TP·5355

印　　数：1～5000

定　　价：29.00元

前　言

C 语言是目前应用最为广泛的计算机高级程序设计语言之一。它短小精悍，功能强大，目标程序效率高，可移植性好，既具有高级语言的优点，又有低级语言的许多特点。因此，C 语言特别适合编写系统软件。目前最流行的操作系统几乎都是用 C 语言写的，它几乎可以实现其他语言所能实现的所有功能。现在，C 语言已不仅为计算机专业工作者所使用，而且为广大计算机应用人员(包括大量非计算机专业出身的人们)所喜爱和使用。学习 C 语言已成为广大计算机应用人员和青年学生的迫切要求。

由于 C 语言牵涉到的概念比较复杂，规则繁多，使用灵活但容易出错，不少初学者反映 C 语言易学难精，往往学过之后感觉学的只是些皮毛，因此迫切希望有一本能够对 C 语言进行深入讲解的复习指导书。本书针对这些特点精心策划，准确定位；而且内容新颖，概念清晰，体系合理，例题丰富，逻辑性强，文字流畅，通俗易懂。

本书的主要特点有以下几个方面：

详细列出了 C 语言的概念，并对一些重要的概念做了进一步说明。对于一些基本理论力求解释详细，易于读者理解，能够看过之后一目了然。

针对一些典型的例题进行深入的讲解，在各个例题中贯穿了每章的知识要点，便于对内容的理解和掌握，同时在例题中又培养学生良好的编程能力，能够通过例题举一反三。

在每章的最后都给出了一些习题，习题内容紧扣本章的知识要点，又不失灵活性，使读者能够通过做一定量的习题来深刻掌握 C 语言的精髓。每道习题都给出了答案便于读者自我检查。

本书由李俊杰老师编写，参加编写的还有赵子慧、秦瑞丽、苏志英、侯旅、师贵钦、陈洁、王为治等。在写作过程中由于时间仓促，作者水平有限，难免存在不妥之处，恳请各位读者不吝指正。

目　录

第1章 C语言入门

C语言是在20世纪70年代初问世的。1978年由美国电话电报公司(AT&T)贝尔实验室正式发表了C语言。同时由B.W.Kernighan和D.M.Ritchit合著了著名的《THE C PROGRAMMING LANGUAGE》一书。通常简称为《K&R》，也有人称之为《K&R》标准。但是，在《K&R》中并没有定义一个完整的标准C语言，后来由美国国家标准学会在此基础上制定了一个C语言标准，于1983年发表。通常称之为ANSI C。最初的C语言只是为描述和实现UNIX操作系统提供的一种语言而设计的，后来对C语言又多次做改进，现在C语言已经风靡全世界，成为世界上应用最广泛的几种计算机高级语言之一。

1.1 基本概念

C语言是一种结构化语言。它层次清晰，便于按模块化方式组织程序，易于调试和维护。C语言的表现能力和处理能力极强。它不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构，还具有效率高，可移植性强等特点。因此广泛地移植到了各种类型操作平台上，从而形成了多种版本的C语言。

1.1.1 C语言

C语言是在B语言的基础上发展起来的广泛流行的、很有发展前途的国际性计算机高级语言。它适合于作为系统描述语言，即可用来写系统软件，也可用来写应用软件。

1.1.2 C语法单位

词法符号是程序设计语言中由若干字符组成的有意义的最小语法单位。按照词法符号在程序中的作用，可以分为：关键字、标识符、分隔符、运算符、标点符号和常量。

1. 关键字

关键字是由系统预定义的词法符号，由系统保留定义权，不允许用户重新定义。在C语言中，这些关键字有特定的含义，并在程序中有着不同的用途。学习C语言，必须了解这些关键字的意义和用法。表1.1列出了一些基本关键字。

表1.1 基本关键字

auto	break	case	char	const
continue	default	do	double	else

续表

enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

注意： C语言中所有关键字均为小写。

2. 标识符

在程序中使用的变量名、函数名、标号等统称为标识符。除库函数的函数名由系统定义外，其余都由用户自定义。C语言规定：标识符只能是字母(A～Z，a～z)、数字(0～9)、下划线(_)组成的字符串，并且其第1个字符必须是字母或下划线。

以下标识符是合法的：

```
a_x, x3, BOOK_1, sum5
```

以下标识符是非法的：

```
3s      以数字开头
s*T     出现非法字符*
-3x     以减号开头
bowy-1  出现非法字符-(减号)
```

在使用标识符时还必须注意以下几点：

- 标准C语言不限制标识符的长度，但它受各种版本的C语言编译系统限制，同时也受到具体计算机的限制。例如在某版本C语言中规定标识符前八位有效，当两个标识符前八位相同时，则被认为是同一个标识符。
- 在标识符中，大小写是有区别的。例如BOOK和book是两个不同的标识符。
- 标识符虽然可由程序员随意定义，但标识符是用于标识某个量的符号。因此，命名应尽量有相应的意义，以便阅读理解，作到“顾名思义”。

3. 运算符

运算符是表示运算的词法符号，C语言有非常丰富的运算符。C语言的运算符按功能可以分为算术运算符、逻辑运算符、关系运算符、位运算符、赋值运算符、递增递减运算符、地址运算符、逗号运算符和sizeof运算符等；按运算符需要的操作数的数目可分为单目运算符、双目运算符和三目运算符。运算符是C语言中非常重要的部分，将在第3章中详细讨论。

4. 分隔符

分隔符的作用是分隔其他的词法符号。C语言中的分隔符包括空格符、制表符(Tab键)、换行符和注释。

在C语言的源代码中，词法符号之间可以用一个或多个分隔符作为间隔，例如，在连

续出现的关键字和标识符之间，必须有分隔符进行分隔。即使在不需分隔符也能识别词法符号的情况下，也可以通过分隔符的恰当运用，使代码的外观格式更为清晰易读。

尽管 C 语言对源代码的书写格式没有强制性规定，但好的程序书写习惯不仅有助于程序的阅读，也可以帮助分析程序中的语法错误。

建议采用缩进的格式书写代码，基本原则为：尽量在一行中书写一条语句；同一层次上的语句应当左端对齐书写；子语句应当比上层语句的左端缩进若干字符书写；适当地运用空格，使代码的形式尽可能清晰易读。

为了便于阅读和理解程序，可以在某一程序段或某语句的前面或后面加注释，解释该程序段或语句的功能。C 语言采用“/*”和“*/”作为注释的起始和结束标志，从“/*”到“*/”之间的所有文字都被视作注释。如下所示：

```
/*   下面程序段是判断数据是否服从正态分布     */
/*   This program is an example.  */
```

注释在编译过程的词法分析阶段将从源代码中过滤掉，因此其语法作用与分隔符相同。在一个程序中，适当的注释不仅有助于编程者记忆各个模块的功能和符号的意义，同时还可以帮助其他阅读者理解源程序。但是，太多的或不必要的注释也可能引起程序的混乱。

对于初学者来说，从一开始编写代码就注意代码的格式，养成良好的程序书写风格和习惯，合理使用注释和其他分隔符，在以后的程序设计实践中将会受益匪浅。

5. 标点符号

C 语言中的标点符号有逗号、分号、冒号、花括号、圆括号。标点符号的作用与分隔符相似，但用法非常严格，什么地方使用什么标点符号，有明确的语法规定。有些标点符号同时也作运算符用，当符号在表达式中出现时，是运算符；在其他地方出现时，是标点符号。

- 逗号(，)

  在变量定义语句中，用于分隔变量名，如：

  ```
  int x, y;
  ```

  在函数的参数表中，用于分隔函数参数，如：

  ```
  int F( int a, int b );
   F( x, y );
  ```

  在 C 语言中，逗号同时也是一个运算符，当逗号出现在表达式中时，即作为逗号运算符，如：(i=0,j=0);

- 分号(；)

  用于表达式语句的结束；如：

  ```
  x=3; y=4;;
  ```

  在 for 循环语句中，分隔括号内的 3 个表达式，如：for (i=0; i<9; i++)

  如果单独出现，构成一条空语句。

- 冒号(：)

用于语句标号的结束，如：start:_;

用于 switch ... case 语句中 case 块中，例如：

```
switch( ch ){
case 'a':_
…
case 'b':_
```

在条件运算符“?:”中的冒号则是条件运算符的一部分，而不是标点符号。

- 花括号 {}

用做枚举类型定义的开始和结束；

用做数组初始值定义的开始和结束；

用做函数体的开始和结束；

用做语句块的开始和结束；

用做结构类型定义块的开始和结束；

- 圆括号 ()

用在函数头定义中，作为参数表的开始和结束；

用在流程控制语句中，作为条件表达式的开始和结束；

用在函数调用中，作为参数表的开始和结束；

当圆括号出现在表达式中时，则是括号运算符，而不是标点符号。

1.1.3 C 程序

C 程序是一组计算机能识别和执行的指令。每一条指令使计算机执行特定的操作。程序可用高级语言编写。用高级语言编写的程序称为源程序。从根本上说，计算机只能识别和执行由 0 和 1 组成的二进制指令，而不能识别和执行用高级语言编写的程序指令。要使计算机能够执行高级语言源程序，必须先用一种称为“编译程序”的软件，把源程序编译成二进制的目标程序，然后将该目标程序与库函数和其他目标程序链接起来，形成可执行的目标程序。

程序= 数据结构 + 算法

数据结构：数据既是程序加工操作的原料，又是程序加工操作的结果。因此数据是程序的中心，即没有数据也就没有对数据的操作，对数据的正确描述和合理构造是程序的基础要素。

算法：对数据进行什么操作，如何操作就是算法，它是程序的关键要素，是程序的灵魂。

1.1.4 程序的实现

一个程序的实现通常可以用下面的公式来表达：

程序(实现)= 程序概念 + 程序设计方法 + 语言工具和环境

语言工具和环境：程序的实现工具是某种语言标准版本，如 C 语言、BASIC 语言、PASCAL 语言、FORTRAN 语言等。任何一种语言标准都会在不同公司所提供的使用环境中被具体化，例如 Turbo C、Visual C、Boland C 提供了 C 语言的不同使用环境。

程序设计方法：设计程序所遵循的方法，与实现程序的语言工具和环境无关，例如，结构化面向过程的程序设计方法、面向对象的程序设计方法。

1.1.5 数据

凡是能够被计算机接收、识别和处理的一切符号称为数据。“1234”是数据，一张图片也可以做数据，一支歌，一串有确切含意的字符串，如“北京”、“WTO”也是数据。根据数据不同属性，数据又可以划分为不同类型，如能够进行算术运算的叫做数值型数据；能够说明事务某个特征的一个串字符叫字符型数据，如人民大会堂、UFO。

1.1.6 指令

在计算机中，指令和命令是同义语(尽管有时候两者有一点差异)。它指示计算机进行什么操作和操作什么。进行什么操作的符号叫做操作码，操作对象叫操作数。所以一条指令包含操作码和操作数两上部分。如

```
sum=1+2;
```

这是一条 C 程序指令，该指令省略了操作码，但包括了加法运算和赋值运算两步操作，1、sum、2 是指令的操作对象。

1.1.7 编译系统和解释系统

无论是高级程序设计语言还是专用程序设计语言，都不能被计算机系统直接识别，用这些语言所编写出的程序代码称为源程序，源程序需要通过预先设计好的专用程序进行转换，转换为计算机系统可以识别的机器指令，然后才能交由计算机系统执行。这种转换程序与汇编程序不同，由于高级语言与人类自然语言更为接近，因此，往往高级语言中的一条语句可能会对应于数条或数十条机器指令，而且绝不是简单的一一对应关系。

最初，这种转换程序是由计算机系统的生产厂家开发并提供给使用者，随着系统复杂度的提高和产业的细化，现在通常由专业的软件开发厂商来为各种计算机系统开发转换程序，而且在功能上也已经不再仅仅局限于对源程序的转换，而是将程序开发过程中所需要的各项功能集成在一个软件平台上提供给使用者。

转换程序最常见的工作机制有编译型和解释型两种，这两种工作机制的区别类似于外语的书面翻译和现场翻译的区别。编译型转换程序是将源程序进行整体的分析和转换，生成机器代码程序，称为目标程序。目标程序可以直接被计算机系统识别和执行。解释型转换程序则是对源程序进行逐句地分析和转换，每转换一条语句，即执行一条语句。

编译型语言需要进行整体的分析和转换(称为编译过程)，生成目标程序后才能执行，转换速度较慢，而且当源程序发生修改后，必须重新编译，才能再次执行，在执行过程中

不能停止和修改源程序，不便于对程序进行调试。但是编译型语言是对整个程序进行分析，可以针对整个程序进行更深入的优化处理，而且只需经过一次编译，生成目标程序，以后就可以直接执行目标程序。目标程序可以直接被计算机系统识别和执行，因此执行速度也比较快。

解释型语言并不生成整体的目标程序，而是对源程序进行逐条解释和逐条执行，因此不能进行全局性的优化，执行速度相对也较慢，而且每次执行都需要重新解释。但是解释型语言的转换程序由于是按语句进行解释，其难度比对整个程序进行分析要小，更容易实现，而且解释型的程序是边解释边执行，可以在执行过程中随时停下来修改源程序，然后继续执行，这对于程序的调试是非常有利的。

现行的很多开发系统往往兼有编译型和解释型的特点，在程序开发过程中，可以以解释的方式执行程序，以便于对程序进行调试，当程序开发完毕后，再通过编译生成最终的目标程序。

1.2 基本理论

了解了C语言的特点和C程序的特性，才能学会编写好的C程序，而编写的C程序是否正确，还要通过上机来检验。

1.2.1 C语言的特点

C语言之所以现在能蓬勃发展，并具有强大的生命力，在于它具有不同于其他语言的优点。

C语言的主要特点如下：

- C语言简洁、紧凑，使用方便、灵活，共有32个关键字，9种控制语句，程序书写迅速自由，主要用小写字母表示，压缩了一些不必要的成分。
- 运算符丰富。共有34种运算符。
- 数据结构丰富，具有现代语言的各种数据结构。
- 具有结构化的控制语句。
- 语法限制不太严格，程序设计自由度大。
- C语言允许直接访问物理地址，能进行位(bit)操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。
- 生成目标代码质量高，程序执行效率高。
- 用C语言写的程序可移植性好(与汇编语言比)。

1.2.2 C程序的特性

C程序具有如下特性：

- C程序是由函数构成的。一个C源程序至少包括一个函数(main函数)，也可以包

括一个 main 函数和若干其他函数。因此，函数是 C 程序的基本单位。被调用的函数可以是系统提供的库函数(如 printf 和 scanf 函数)，也可以是用户根据需要自己编制设计的函数(例如第 7 章中的【例 12】中的 max 函数)。C 的函数相当于其他语言中的子程序。用函数来实现特定的功能。可以说 C 是函数式的语言。程序全部工作都是由函数来完成的。C 的这种特性很容易实现模块化。

- 一个函数由两部分组成：
  - 函数的说明部分。包括函数名、函数类型、函数属性、函数参数(形参)名、形式参数类型；
  - 函数体，即函数说明部分下面的大括弧{...}内的部分。如果一个函数内有多个大括弧，则最外层的一对{}为函数体的范围。

    函数体一般包括：

    变量定义。如【例 1.3】中 main 函数中的 int a,b,c;。

    执行部分。由若干个语句组成。

  当然，在某些情况下也可以没有变量定义部分(例如【例 1】)。甚至可以既无变量定义也无执行部分，如：

```
dump()
{ }
```

  它是一个空函数，什么也不干，但这是合法的。
- 一个 C 程序总是从 main 函数开始执行的，而不论 main 函数在整个程序中的位置如何。main 函数可以放在程序最前头，也可以放在程序最后，或在一些函数之前在另一些函数之后。
- C 程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行上。C 程序没有行号，也不象 FORTRAN 或 COBOL 那样严格规定书写格式(语句必须从某一列开始书写)。
- 每个语句和数据定义的最后必须有一个分号。分号是 C 语句的必要组成部分。例如：c=a+b; 分号不可少。即使是程序中最后一个语句也应包含分号(这是和 Pascal 语言不同的)。
- C 语言本身没有输入输出语句。输入和输出的操作是由库函数 scanf 和 printf 等函数来完成的。C 对输入输出实行函数化。
- 可以用/*……*/对 C 程序中的任何部分作注释，以增加程序的可读性。

1.2.3　算法的特性

在 C 语言中，通常有以下几种算法特性：

- 有穷性

  一个算法通过若干步骤即可实现预定目标，算法的每一步都可以在合理的时间内完成。
- 确定性

算法中的每一条指令都必须有确切含义，同样的步骤、同样的条件，结果也是相同的。

(3) 输入和输出

计算机工作大体由3步组成：

输入数据 → **处理数据** → **输出结果**

计算机工作的3个步骤具有不可逆的方向性。再简单、复杂的计算机都是这样的，也就是说一个算法再简单，也要有输入、输出、处理3个步骤。

算法是可以描述的，描述算法的常用工具是伪代码和流程图。

所谓伪代码是指用自然语言和指令语句(不要求绝对正确的语句)结合起来描述算法的一种方法。这种方法与画流程图相比省时省力，转换为程序容易，但是清晰度与层次性不如流程图。伪代码描述算法没有统一的规定，写出来只要自己或别人看懂即可。

1.2.4 算法的表示方法

C语言的常用的算法的表示方法有如下几种：

- 用自然语言表示
- 用流程图表示
- 用N-S流程图表示
- 用伪代码表示
- 用计算机语言表示

1.2.5 C源程序的书写规范

从书写清晰，便于阅读，理解，维护的角度出发，在书写程序时 应遵循以下规则：

- 一个说明或一个语句占一行。
- 用{} 括起来的部分，通常表示了程序的某一层次结构。{}一般与该结构语句的第一个字母对齐，并单独占一行。
- 低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写。以便看起来更加清晰，增加程序的可读性。在编程时应力求遵循这些规则，以养成良好的编程风格。

1.2.6 程序设计方法

程序设计方法就是人们在开发程序时所采用的解题策略，一个好的程序设计方法是提高程序质量和程序开发效率的基本手段。

1. 结构化程序设计方法

结构化程序设计方法是荷兰学者迪克特拉(Dikctra)等人提出来的，其要点包括：

- 编写的程序易读易理解。

- 参加设计人员必须按照同一规划、同一原则进行程序设计。
- 顺序结构、选择结构和循环结构是大而复杂程序的三大基本结构。
- 采取“自顶向下，逐步细化”和模块化的设计方法。

自顶向下是指从解题的题目开始向下进行分解，例如，确定题目的最终要求、确定题目运行的环境平台、选择一种合适的计算机语言，按照题目的内容结构划分为若干个子题目(模块)，对子题目进行算法描述，逐步细化到基本结构为止，即把大问题分解为小问题，直到小问题很容易解决。

模块化从要解决的问题出发，将问题按照功能划分为若干个模块，采用自顶向下逐步细化的一种方法，即把大问题分解成若干个功能相对独立的一个功能，这个功能可以是子系统或一个子程序，如学籍管理信息系统至少包括：输入、统计、查询、打印 4 个子系统，每个子系统都是一个模块。查询模块又可划分为按姓名查询、按学号查询、按身份证号查询等子系统。模块具有相对的独立性，模块化便于组织分工，尤其是便于消除因程序的一个错误而影响程序全局的现象，对提高编程和调试效率有很大帮助。

2. 面向对象程序设计

面向对象的思想最初出现于挪威斯陆大学和挪威计算中心共同研制的 Simula67 语言中，其后，随着位于美国加利福尼亚的 Xerox 研究中心推出的 Smalltalk-76 和 Smalltalk-80 语言，面向对象的程序设计方法得到比较完整的实现，由此面向对象的程序设计方法得到迅速的发展。

面向对象是当前计算机界关心的重点，它是 20 世纪 90 年代软件开发方法的主流。面向对象的概念和应用已经超越了程序设计和软件开发，扩展到很宽的范围，如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术和人工智能等领域。一些新的工程概念及其实现，如并行工程、综合集成工程等也需要面向对象的支持。所以面向对象是程序设计的新风范，它是软件开发的一种新方法，也是一种新技术。

面向对象的程序设计方法要求语言必须具备抽象、封装、继承和多态性 3 个方面的特殊性。面向对象就是将世界看成是由一些彼此相关并能相互通信的实体(即对象)来实现的。程序中的对象映象现实世界中的对象。

1.2.7　程序上机步骤

在实际应用中使用高级语言进行程序设计的一般步骤如图 1.1 所示。

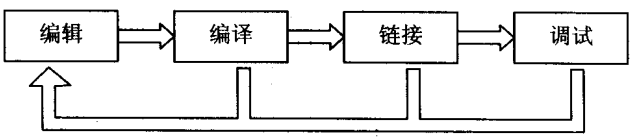


图 1.1　程序设计的一般步骤

编辑是程序员通过编辑软件录入源程序的过程。通常程序员要先利用某种文字编辑软件录入源程序代码，生成源程序文件。

源程序文件录入后，就可以使用编译程序对源程序文件进行编译，如果源文件中有语

法错误，无法生成目标文件，则返回编辑过程，程序员根据编译程序所提供的编译错误信息对源文件进行修改，并再次编译，直到编译通过，成功生成目标文件。

目标文件已经是二进制的机器代码，但在实际应用当中，往往目标文件还不能直接执行。在实际的程序设计中，经常需要使用到一些基本功能，例如，输入输出、各种常用计算等，但是这些功能有的对于一般程序员来说难以自行开发，有的使用频率很高，因此编译系统通常将这些功能预先编制好，以程序库的形式提供给程序员使用，如果程序中引用到了这些功能，就需要将这些功能的代码嵌入到待开发的程序中。

另一方面，待开发的程序可能不只由一个源文件组成，而是由多个源文件构成的，每个源文件是整个程序中的一个局部，经过编译后，每个源文件生成一个不完整的目标文件，将这些目标文件拼在一起才能构成整个的目标程序。

链接过程就是对组成整个程序的各个目标文件、程序库进行拼装，生成最终的目标程序的过程。如果在链接过程中发现错误，例如某个被引用的子程序在所有的目标文件中都没有找到，或者某个子程序同时出现在两个以上的目标文件中等，无法生成最终的、完整的、可执行的目标程序，可能需要再次回到编辑过程中，对源程序进行修改、再编译、再链接，直到成功地生成了可执行的目标程序。

经过编译和链接的过程，并不意味着程序的开发过程完成了。编译和链接过程的成功只是意味着在源程序中没有语法上的错误，而并不意味着程序在逻辑上也是正确的。因此还需要对所生成的目标程序进行试运行，观察程序的运行是否达到了预期的目标，是否能够实现预定的功能。如果发现程序不能按照预期的设想工作，就需要对程序的运行过程和程序逻辑进行分析，找出发生错误的原因，这个过程称为调试过程。找到错误之后，又需要回到编辑过程，对源文件进行再修改，再编译、再链接和再调试。

1.3 典型例题

【例 1.1】

设 $f_1()$至 $f_n()$代表用户定义的函数，则下列程序结构(　)是不正确的。

A.

```
main()
{ 程序段 }
f1()
{ 程序段 }
  …
fn()
{ 程序段 }
```

B.

```
f1()
  { 程序段 }
    …
  fn()
  { 程序段 }
 main()
  { 程序段 }
```

C.

```
  f1()
{ 程序段 }
main()
{ 程序段 }
…
```

D.

```
  f1()
  {{ 程序段 }
  main()
  { 程序段 }}
  …
```