

# 第 1 章 引言

## 1.1 教学要点

本章主要介绍程序与程序设计语言的知识、C 语言的发展历史与特点，以及利用计算机求解问题的基本过程，使学生了解相关的背景知识，对程序和程序设计语言有初步的认识，理解计算机求解问题的基本过程。

1.1 节通过一个示例程序“计算阶乘”使学生对 C 语言程序有一个感性认识。教师在讲授时，可以先运行程序，让学生观察程序的运行过程，再简单介绍 C 程序的构成。

1.2 节简要介绍程序的概念、程序设计语言的功能和语法要素，使学生对程序与程序设计语言有初步的认识，不需要展开。



1.3 节简要介绍 C 语言的发展历史与特点。

1.4 节通过讲解一个示例“求 1~100 间所有偶数的和”，说明利用计算机求解问题的过程，尤其是程序设计的主要过程。

建议教师讲授时，着重讲解 1.1 和 1.4 两节，结合程序演示，使学生对用 C 语言编程解决实际问题的全过程有直观的认识，激发其学习编程的兴趣。宜将 2.1 节并入第一次课的讲授，让学生第 1 次课后就能练习编程，有助于养成每周坚持课内、外上机的良好习惯。对 1.2 和 1.3 两节则简要介绍，也可以考虑将这两节的部分内容放在最后的复习课，帮助学生梳理知识体系，体会 C 语言编程的特点。

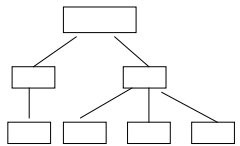
讲授学时：2 学时，实验学时同讲授学时。

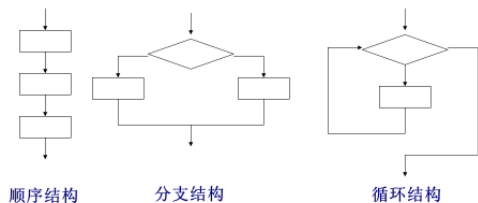
## 1.2 讲稿

1	 <b>Chap 1 引言</b>  1.1 一个C语言程序 1.2 程序与程序设计语言 1.3 C语言的发展历史与特点 1.4 实现问题求解的过程	本章分 4 节。
2	 <b>本章要点</b>  ■ 什么是程序？程序设计语言包含哪些功能？ ■ 程序设计语言在语法上包含哪些内容？ ■ 结构化程序设计有哪些基本的控制结构？ ■ C语言有哪些特点？ ■ C语言程序的基本框架如何？ ■ 形成一个可运行的C语言程序需要经过哪些步骤？ ■ 如何用流程图描述简单的算法？	提出本章的学习要点。

3	<div> <div>1.1 一个C语言程序</div> <div> <div>输入 4</div> <div>输出 24</div> </div> <div> <p>例 1-1 求阶乘问题。输入一个正整数n，输出n!。</p> <pre> #include &lt;stdio.h&gt;           /* 编译预处理命令 */ int main(void)              /* 主函数 */ {     int n;                  /* 变量定义 */     int factorial(int n);    /* 函数声明 */     scanf("%d", &amp;n);        /* 输入一个整数 */     printf("%d\n", factorial(n)); /* 调用函数计算阶乘 */     return 0; } int factorial(int n)        /* 定义计算n!的函数 */ {     int i, fact = 1;     for(i = 1; i &lt;= n; i++) {         fact = fact * i;     }     return fact; } </pre> <div> <div>C程序由函数组成</div> <div>有且只有一个主函数main()</div> </div> </div> </div>	<p>展示第一个 C 语言程序，现场演示编译、连接、运行的全过程，伴以生动的语言描述，吸引学生的兴趣。</p> <p>简要说明 C 语言由函数组成，有且只有一个 main 函数。</p>
4	<div> <div>1.2 程序与程序设计语言</div> <div> <div>■ 程序</div> <div> <div>□ 人们为解决某种问题用计算机可以识别的代码编排的一系列加工步骤。</div> <div>□ 程序的执行过程实际上是对程序所表达的数据进行处理的过程。</div> </div> <div>■ 程序设计语言</div> <div> <div>□ 提供了一种表达数据与处理数据的功能</div> <div>□ 要求程序员按照语言的规范编程</div> </div> </div> </div>	<p>程序表达了要对数据进行处理的过程，运行程序就是实际处理数据。</p> <p>程序设计语言用于编写程序，因此，它必须能表达数据、表达处理过程，同时有自己的规范，即语法。</p> <p>程序员需合法（合乎语法）使用程序设计语言编程。</p>
5	<div> <div>1.2 程序与程序设计语言</div> <div> <div>■ 1.2.1 程序与指令</div> <div>■ 1.2.2 程序设计语言的功能</div> <div>■ 1.2.3 程序设计语言的语法</div> <div>■ 1.2.4 程序的编译与编程环境</div> </div> </div>	<p>本节介绍 4 个问题，其中程序与指令、程序的编译与编程环境举例说明，其余简要说明。</p>
6	<div> <div>1.2.1 程序与指令</div> <div> <div>■ 指令：计算机的一个最基本的功能</div> <div>如实现一次加法运算或实现一次大小的判别</div> <div>■ 计算机的指令系统：计算机所能实现的指令的集合</div> <div>■ 程序：一系列计算机指令的有序组合</div> </div> </div>	<p>第一个问题，介绍程序与指令。</p> <p>解释指令（指令集）和程序。</p> <p>指令集是有限的，但一系列指令的组合却能实现复杂的功能，这就是计算机的奇妙之处。</p>

7	<p><b>程序与指令</b></p> <p>例1-2 编写程序，分别求和与乘积</p> <p>■ 虚拟的计算机指令系统（7条指令）</p> <ul style="list-style-type: none"> <li>指令1: Input X 将当前输入数据存储到内存的X单元</li> <li>指令2: Output X 将内存X单元的数据输出。</li> <li>指令3: Add X Y Z 将内存X单元的数据与Y单元的数据相加并将结果存储到Z单元。</li> <li>指令4: Sub X Y Z 将内存X单元的数据与Y单元的数据相减并将结果存储到Z单元。</li> <li>指令5: BranchEq X Y P 比较X与Y，若相等则程序跳转到P处执行，否则继续执行下一条指令。</li> <li>指令6: Jump P 程序跳转到P处执行。</li> <li>指令7: Set X Y 将内存Y单元的值设为X。</li> </ul>	<p>例如，一个只有 7 条指令的指令系统，如何通过对这些指令的不同组合实现求和、乘积的功能。</p> <p>将这些指令分为 3 类：</p> <ul style="list-style-type: none"> <li>● 输入和输出；</li> <li>● 加、减运算和赋值运算；</li> <li>● 流程控制。</li> </ul>
8	<p><b>程序与指令</b></p> <p>■ 输入3个数A, B和C，求A+B+C的结果</p> <p>Input A;            输入第1个数据到存储单元A中  Input B;  Input C;  Add A B D;        将A、B相加并将结果存在D中  Add C D D;        将C、D相加并将结果存在D中  Output D;         输出D的内容</p>	<p>求 <math>A+B+C</math>，可以通过指令集中 2 类 3 条指令（输入、求和、输出）的组合实现。</p> <p>设问 1：如何求 <math>A-B+C</math>？</p> <p>解答：</p> <p>将 Add A B D;  改为：  Sub A B D;</p>
9	<p><b>程序与指令</b></p> <p>■ 输入A，求A+A+A的结果</p> <p>解1:  Input A;  Add A A D;  Add A D D;  Output D;</p> <p>解2:  Input A;  Set 0 Z;  Add Z A Z;  Add Z A Z;  Add Z A Z;  Output Z;</p>	<p>设问 2：如何求 <math>A+A+A</math>？</p> <p>解 1 延续了求 <math>A+B+C</math> 的思路；</p> <p>解 2 思路：</p> <p>（1）输入一个数到 A 变量  （2）设 <math>Z = 0</math>  （3）重复做 3 遍：<math>Z = Z + A</math>；  （4）输出 Z</p> <p>最后点出 <math>A*3=A+A+A</math>，为后面求 <math>A*B</math> 做铺垫。</p>
10	<p><b>程序与指令</b></p> <p>■ 输入2个数A和B，求<math>A*B</math></p> <p><math>A*B = A+A+\dots+A</math>（B个A相加）</p> <p>◆ 分别输入两个数到A、B两个变量  ◆ 设<math>X = 0, Z = 0</math>  ◆ 当X不等于B时，重复做以下操作：      <math>Z = Z + A</math>;      <math>X = X + 1</math>;  ◆ 输出Z</p> <p>1. Input A;  2. Input B;  3. Set 0 X;  4. Set 0 Z;  5. BranchEq X B 9;  6. Add Z A Z;  7. Add 1 X X;  8. Jump 5;  9. Output Z;</p>	<p>求 <math>A*B</math>：</p> <p>先说明乘法的运算基础是加法（帮助学生回忆小学怎么引入乘法概念），即 <math>A*B=A+A+\dots+A</math>（B 个 A 相加），而计算机的乘法运算可以用加法实现的；再分析程序中用了指令集中 3 类 6 条指令（除减法）实现乘法运算。</p> <p>说明：X 用于计数，每加一个 A 就自增 1，直到 X 的值等于 B，说明已经累加了 B 个 A。</p>

11	<h3>1.2.2 程序设计语言的功能</h3> <ul style="list-style-type: none"> <li>■ 数据表达：表达所要处理的数据</li> <li>■ 流程控制：表达数据处理的流程</li> </ul>	<p>第二个问题，介绍程序设计语言的功能，即数据表达和流程控制。</p>
12	<h3>数据表达</h3> <ul style="list-style-type: none"> <li>■ 数据表达：一般将数据抽象为若干类型</li> <li>■ 数据类型：对某些具有共同特点的数据集合的总称 <ul style="list-style-type: none"> <li>□ 代表的数据（数据类型的定义域）</li> <li>□ 在这些数据上做什么（即操作或称运算）</li> </ul> </li> </ul> <p>例如：整数类型</p> <ul style="list-style-type: none"> <li>■ 包含的数据：{..., -2, -1, 0, 1, 2, ...}</li> <li>■ 作用在整数上的运算：+、-、*、/等</li> </ul>	<p>数据表达：</p> <p>（1）能处理哪些类型的数据？</p> <p>（2）对数据能进行怎样的运算？</p> <p>举例说明，如：</p> <p>C 语言不能直接处理图像类型的数据；</p> <p>C 语言能处理整数，对整数可以进行加、减、乘、除和求余等算术运算；</p> <p>C 语言能处理实数，但是对实数不能进行求余运算。</p>
13	<h3>数据表达</h3> <ul style="list-style-type: none"> <li>■ C语言提供的数据类型 <ul style="list-style-type: none"> <li>□ 基本数据类型：程序设计语言事先定义好，供程序员直接使用，如整型、实型（浮点型）、字符型等。</li> <li>□ 构造类型：由程序员构造，如数组、结构、文件、指针等。</li> </ul> </li> <li>■ 各种数据类型的常量与变量形式 <ul style="list-style-type: none"> <li>□ 常量（常数）与变量</li> </ul> </li> </ul>	<p>如何用有限的语言语法形式表达客观世界中多种多样的数据？</p> <p>基本数据类型+构造类型</p> <p>类比前面提到：虽然指令集是有限的，但一系列指令的组合能实现复杂的功能。</p>
14	<h3>流程控制</h3> <ul style="list-style-type: none"> <li>■ 结构化程序设计方法 <ul style="list-style-type: none"> <li>□ 将复杂程序划分为若干个相互独立的模块</li> <li>□ 模块：一条语句（Statement）、一段程序或一个函数（子程序）等</li> <li>□ 单入口、单出口</li> </ul> </li> </ul> 	<p>流程控制：描述数据处理的过程，即程序的控制过程。</p> <p>结构化程序设计方法：解决复杂问题的方法之一，即分而治之，合而用之。</p>

15	<h3>流程控制</h3> <ul style="list-style-type: none"> <li>任何程序都可以将模块通过3种基本的控制结构进行组合来实现</li> </ul>  <p>顺序结构      分支结构      循环结构</p>	3 种基本控制结构的组合可以解决复杂问题。
16	<h3>流程控制</h3> <ul style="list-style-type: none"> <li>语句级控制：3种基本的控制结构 <ul style="list-style-type: none"> <li>顺序控制结构：自然顺序执行</li> <li>分支控制结构（选择结构）：根据不同的条件来选择所要执行的模块</li> <li>循环控制结构：重复执行某个模块</li> </ul> </li> <li>单位级控制：函数的定义与调用 <ul style="list-style-type: none"> <li>处理复杂问题时，将程序分为若干个相对独立的子程序（函数）</li> </ul> </li> </ul>	单位级控制：程序模块之间的控制（调用、返回），不需展开。
17	<h3>1.2.3 程序设计语言的语法</h3> <ul style="list-style-type: none"> <li>用程序设计语言所写的程序必须符合相应语言的语法 <ul style="list-style-type: none"> <li>源程序（源代码）是一个字符序列，这些字符序列按顺序组成了一系列“单词”，“单词”的组合就形成了语言有意义的语法单位，一些简单语法单位的组合又形成了更复杂的语法单位，最后一组语法单位组合成程序。</li> </ul> </li> </ul>	<p>第三个问题，介绍程序设计语言的语法要素。</p> <p>说明 3 点：</p> <p>（1）程序设计语言有自己的规范（语法）。类比汉语语法、英语语法等。</p> <p>（2）程序员编写的程序必须合法（合乎语法）。</p> <p>（3）示例给出了程序的构成，可以将其比作写文章（字、词、句、段落、文章）。写文章为了表达思想，编程序为了使计算机能理解问题处理过程并解决问题（处理数据）。</p>
18	<h3>程序设计语言的语法</h3> <ul style="list-style-type: none"> <li>C语言的主要“单词” <ol style="list-style-type: none"> <li>标识符: C语言的标识符规定由字母、数字以及下划线组成，且第一个字符必须是字母或下划线。</li> <li>保留字(关键字): 它们是C语言规定的、赋予它们以特定含义、有专门用途的标识符。</li> <li>自定义标识符: 包括在程序中定义的变量名、数据类型名、函数名以及符号常量名。有意义的英文单词</li> <li>常量: 常量是有数据类型的，如，123、12.34</li> <li>运算符: 代表对各种数据类型实际数据对象的运算。如，+（加）、-（减）、*（乘）、/（除）、%（求余）、&gt;（大于）</li> </ol> </li> </ul>	<p>程序设计语言最主要的语法要素之一：“单词”——标识符、常量、运算符等。</p> <p>结合例 1-1 说明，无需展开。</p>

19	<div data-bbox="300 203 820 241" data-label="Image"></div> <h2 data-bbox="331 255 617 293">程序设计语言的语法</h2> <ul style="list-style-type: none"> <li>■ C语言的主要语法单位 <ul style="list-style-type: none"> <li>(1) 表达式: 运算符与运算对象组合就形成了表达式。如, <math>2 + 3 * 4</math></li> <li>(2) 变量定义: 变量也有数据类型, 所以在定义变量时要说明相应变量的类型。如: <code>int i;</code></li> <li>(3) 语句: 语句是程序最基本的执行单位, 程序的功能就是通过对一系列语句的执行来实现的。</li> <li>(4) 函数定义与调用</li> </ul> </li> </ul>	<p>程序设计语言最主要的语法要素之一:</p> <p>由“单词”组成的语法单位—表达式、变量定义、语句、函数等。</p> <p>结合例 1-1 说明, 无需展开。</p>
20	<div data-bbox="300 620 820 658" data-label="Image"></div> <h2 data-bbox="331 672 727 710">1.2.4 程序的编译与编程环境</h2> <ul style="list-style-type: none"> <li>■ 程序的编译 <div data-bbox="363 734 663 775" data-label="Diagram"> <pre> graph LR     A[程序] --&gt; B[编译器]     B --&gt; C[计算机直接能理解的指令序列] </pre> </div> <p>编译器: 对源程序进行词法分析、语法与语义分析, 生成可执行的代码。 直接指出程序中的语法错误</p> <li>■ 编程环境 <p>包括编辑程序 (Edit)、编译 (Compile)、调试 (Debug) 等过程。</p> </li> <li>■ 掌握程序设计语言: 根据语言的语法, 用语言表达数据、实现程序的控制, 并会使用编程环境。</li> </li></ul>	<p>第四个问题, 介绍程序的编译与编程环境。</p> <p>说明 3 点:</p> <p>(1) 编译可类比翻译。</p> <p>(2) 结合例 1-1 说明编程环境和 C 语言上机过程 (下一张 PPT)。</p> <p>(3) 学会编程: 使用一种程序设计语言, 编写符合语法要求的程序, 并在编程环境下编译、连接、运行、调试程序, 以解决实际问题。</p>
21	<div data-bbox="300 1037 820 1075" data-label="Image"></div> <h2 data-bbox="331 1088 553 1126">C 语言上机过程</h2> <div data-bbox="331 1140 799 1382" data-label="Diagram"> <pre> graph LR     A[源程序 test.cpp] --&gt; B[编译]     B --&gt; C[test.obj]     C --&gt; D[连接]     D --&gt; E[可执行代码 test.exe]     E --&gt; F[运行] </pre> </div>	
22	<div data-bbox="300 1453 820 1491" data-label="Image"></div> <h2 data-bbox="331 1505 732 1543">1.3 C语言的发展历史与特点</h2> <ul style="list-style-type: none"> <li>■ 历史 <ul style="list-style-type: none"> <li>□ 1972年: 贝尔实验室的Dennis Ritchie在B语言的基础上设计并实现了C语言。</li> <li>□ 1978年: B.W.Kernighan和D.Ritchie (简称K &amp; R) 合著的《The C Programming Language》是各种C语言版本的基础, 称之为旧标准C语言。</li> <li>□ 1983年: 美国国家标准化协会 (ANSI) 制定了新的C语言标准, 称ANSI C。</li> </ul> </li> </ul>	<p>本节简单介绍即可。利用一些历史资料激发学生的学习兴趣。</p> <p>可结合 UNIX, 介绍 C 语言的发展过程, 并说明 C 语言在业界的地位 (可在网上查找: Tiobe 编程语言排行榜)。</p> <p>也可介绍 C 语言发明人 Ritchie 所获得的荣誉。</p>

23	<div data-bbox="300 203 820 241" data-label="Section-Header"> <h2>C语言的特点</h2> </div> <div data-bbox="344 286 798 560" data-label="List-Group"> <ol style="list-style-type: none"> <li>1. C语言是一种结构化语言</li> <li>2. C语言语句简洁、紧凑，使用方便、灵活 32个关键字，9种控制语句，程序书写形式自由。</li> <li>3. C语言程序易于移植 C语言将与硬件有关的因素从语言主体中分离出来，通过库函数或其他实用程序实现它们。</li> <li>4. C语言有强大的处理能力</li> <li>5. 生成的目标代码质量高，运行效率高</li> <li>6. 数据类型检查不严格，表达式出现二义性，不具备数据越界自动检查功能，运算符的优先级与结合性对初学者难于掌握。</li> </ol> <p>C语言中大小写字母代表不同含义</p> </div> <td data-bbox="831 188 1369 607">说明大小写字母代表不同含义。可将例 1-1 中某个标识符中的小写字母改为大写字母，观察编译器的出错提示。</td>	说明大小写字母代表不同含义。可将例 1-1 中某个标识符中的小写字母改为大写字母，观察编译器的出错提示。
24	<div data-bbox="300 622 820 660" data-label="Section-Header"> <h2>1.4 实现问题求解的过程</h2> </div> <div data-bbox="335 719 798 972" data-label="List-Group"> <p>问题：求1~100间所有偶数的和。</p> <p>1. 问题分析与算法设计</p> <p>求在一定范围内（1~100）、满足一定条件(偶数)的若干整数的和，求累加和。</p> <p>思路：设置一个变量(sum)，其初值为0，然后在1~100的数中(i)寻找偶数，将它们一个一个累加到sum中。</p> <ul style="list-style-type: none"> <li>一步累加：sum = sum + i;</li> <li>重复累加，用循环语句实现，在循环过程中：             <ol style="list-style-type: none"> <li>(1) 判别 i 是不是偶数：用分支控制语句来实现。</li> <li>(2) 对循环次数进行控制：通过 i 值的变化</li> </ol> </li> </ul> </div> <td data-bbox="831 607 1369 1025">本节通过示例“求 1~100 间所有偶数的和”介绍程序设计的主要过程： (1) 问题分析与算法设计； (2) 编辑程序； (3) 编译连接； (4) 运行调试。</td>	本节通过示例“求 1~100 间所有偶数的和”介绍程序设计的主要过程： (1) 问题分析与算法设计； (2) 编辑程序； (3) 编译连接； (4) 运行调试。
25	<div data-bbox="300 1041 820 1079" data-label="Section-Header"> <h2>问题分析与算法设计</h2> </div> <div data-bbox="355 1151 778 1391" data-label="List-Group"> <ul style="list-style-type: none"> <li>思路 —— 确定算法</li> <li>算法：一组明确的解决问题的步骤，它产生结果并可在有限的时间内终止。</li> <li>算法的描述：             <ul style="list-style-type: none"> <li>自然语言</li> <li>伪代码</li> <li>流程图：算法的图形表示法</li> </ul> </li> </ul> </div> <td data-bbox="831 1025 1369 1444">(1) 问题分析与算法设计 分析问题，明确思路，确定算法，并选择适合的方式描述算法。 顺带介绍算法的基本概念、描述方式。</td>	(1) 问题分析与算法设计 分析问题，明确思路，确定算法，并选择适合的方式描述算法。 顺带介绍算法的基本概念、描述方式。
26	<div data-bbox="363 1460 668 1827" data-label="Diagram"> <pre> graph TD     Start([sum = 0 i = 1]) --&gt; LoopCond{i &lt;= 100}     LoopCond -- 假 --&gt; Output([输出 sum])     LoopCond -- 真 --&gt; EvenCond{i 是偶数?}     EvenCond -- 真 --&gt; SumAdd[sum = sum + i]     EvenCond -- 假 --&gt; IncI[i = i + 1]     SumAdd --&gt; IncI     IncI --&gt; LoopCond     </pre> </div>	

27	<h2>编辑程序</h2> <p>2. 编辑程序 生成程序的源文件，C语言源文件的后缀为 <b>.c / .cpp</b></p> <pre>#include &lt;stdio.h&gt; int main(void) {     int i, sum = 0;      for(i = 1; i &lt;= 100; i++){         if (i%2 == 0) {             sum = sum + i;         }         printf("%d", sum);         return 0;     } }</pre>	(2) 编辑程序：现场演示，可以观察文件的保存位置。
28	<h2>程序编译连接</h2> <p>3. 程序编译连接 编辑程序后，用该语言的编译程序对其进行编译，以生成二进制代码表示的目标程序(.obj)，与编程环境提供的库函数进行连接 (Link) 形成可执行的程序(.exe)。</p> <p>编译程序指出<b>语法错误</b></p>	(3) 编译连接：现场演示，观察提示窗口的内容（编译无错误）。
29	<h2>运行与调试</h2> <p>4. 运行与调试 经过编辑、编译、连接，生成执行文件后，就可以在编程环境或操作系统环境中运行该程序。 如果程序运行所产生的结果不是你想要的结果，这是程序的<b>语义错误</b>（<b>逻辑错误</b>）。</p> <p><b>语法错误VS逻辑错误</b></p> <p>调试：在程序中查找错误并修改错误的过程。</p> <p>调试的方法</p> <ul style="list-style-type: none"> <li>■ 设置断点</li> <li>■ 单步跟踪</li> </ul> <p>调试是一个需要耐心和经验的工作，也是程序设计最基本的技能之一。</p>	<p>(4) 运行与调试：现场演示，看运行结果。修改程序，观察语法错误、逻辑错误及改正方法。</p> <p>✧ 语法错误：可以删除一个分号，使编译出现错误提示，改正之，再编译。 类比写文章有错别字。</p> <p>逻辑错误：将 <code>sum=sum+i</code> 改为 <code>sum=sum-i</code>，编译、连接、运行，发现运行结果不符合题目要求。找出错误（调试），改正之，再编译、连接、运行，看结果。类比写文章词不达意。</p>
30	<h2>C语言程序的编辑、编译连接、运行调试步骤示意图</h2> <pre> graph TD     Start([开始]) --&gt; Edit[编辑]     Edit --&gt; Compile[编译]     Compile --&gt; Output{输出?}     Output -- Y --&gt; Link[连接]     Output -- N --&gt; Compile     Link --&gt; Execute[执行]     Execute --&gt; Correct{正确?}     Correct -- Y --&gt; End([结束])     Correct -- N --&gt; Edit     subgraph Library [库函数和库函数]         direction TB         L1[库函数]         L2[库函数]     end     Library -.-&gt; Link   </pre>	<p>✧ 归纳总结 C 语言程序的编辑、编译连接、运行调试步骤的示意图。用本章所学过的顺序、分支以及循环三种流程结构图来表达，达到复习前面所学知识的目的。</p>



## 1.3 习题参考答案

1. 对 C 语言来说，下列标识符中哪些是合法的，哪些是不合法的？

total, \_debug, Large&Tall, Counter1, begin\_

解答：合法标识符：total, \_debug, Counter1；不合法标识符：Large&Tall, begin\_。

2. 改写本章 1.4 节中的流程图 1.2，求 1~100 中能被 6 整除的所有整数的和。

解答：

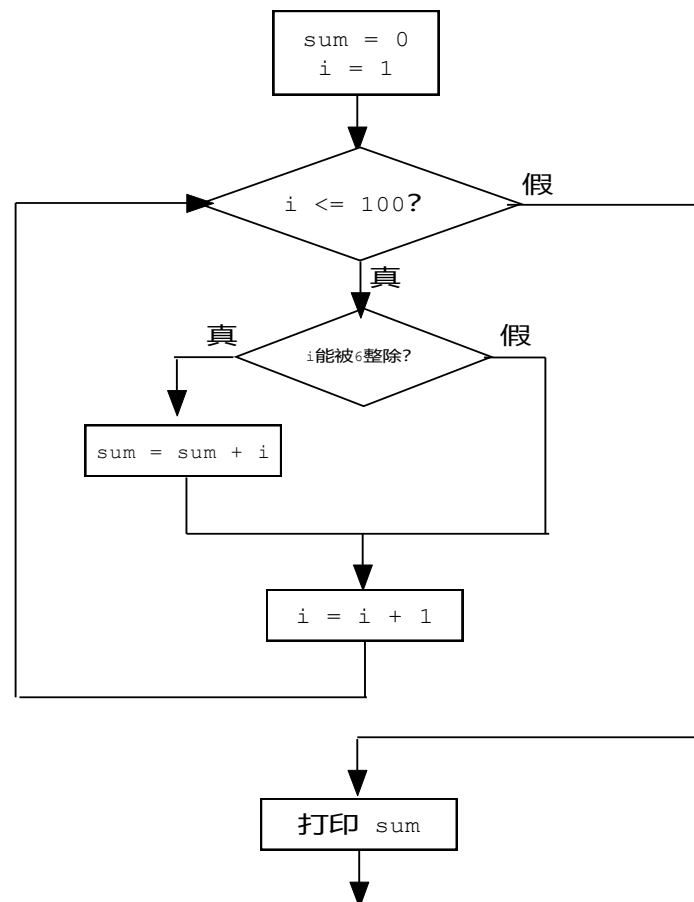


图 1.1 “求 1~100 中能被 6 整除的所有整数的和”的流程图

3. 改写本章 1.4 节中的程序，求 1~100 中能被 6 整除的所有整数的和，并在编程环境中验证该程序的运行结果。

解答：

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int i, sum = 0;
```

```
    for(i = 1; i <= 100; i++){
```

```
        if (i % 6 == 0) {
```

```
            sum = sum + i; }
```

```
}
```

```

printf("%d", sum);
return 0;
}

```

4. 对于给定的整数  $n$  ( $n > 1$ )，请设计一个流程图判别  $n$  是否为一个素数（只能被 1 和自己整除的整数），并分析该流程图中哪些是顺序结构、哪些是分支结构与循环结构。

解答：在流程图中，分支结构和循环结构如图 1.2 所示，自上而下的 2 个实线框和 2 个虚线框组成了顺序结构。

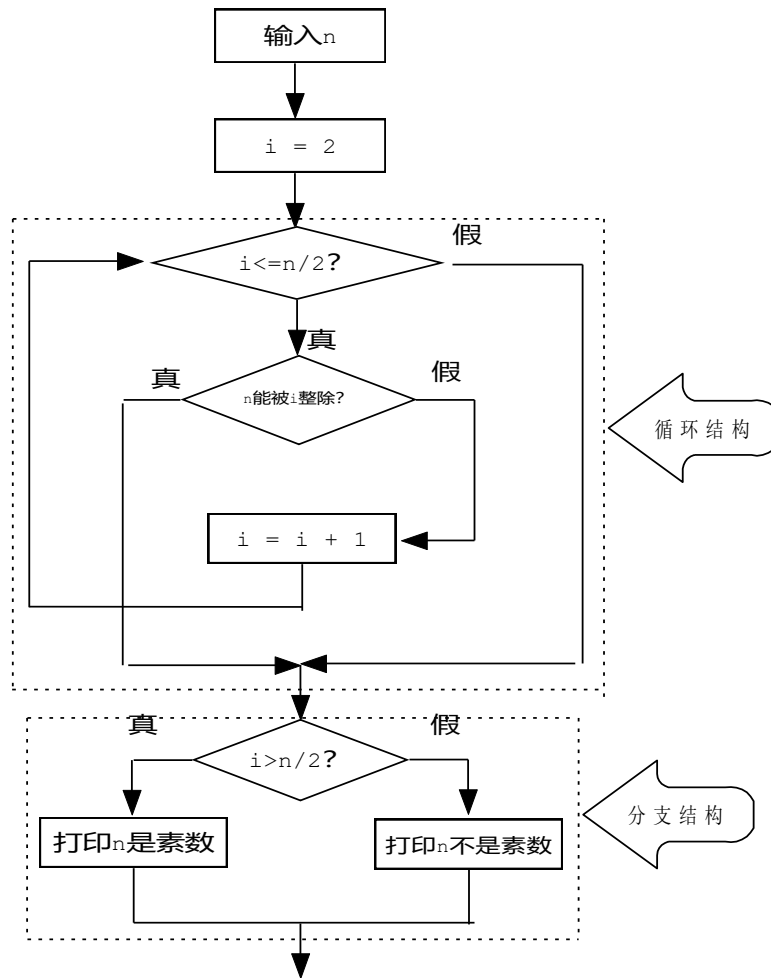


图 1.2 “判别  $n$  是否为素数”的流程图

## 1.4 实验指导教材参考答案

### 一、编程示例

略

### 二、运行示例

略