

第4章 循环结构

4.1 教学要点

本章通过典型程序解析,介绍循环结构程序设计的基本思想和实现方法,以及典型算法,使学生理解 C 语言所提供的三种循环语句即 `for`、`while` 和 `do-while` 语句的执行机制,并能综合运用编写循环结构类的程序。

4.1 节通过引例“用格里高利公式求 π 的近似值”和示例“统计学生成绩”,针对没有显性给出循环次数的情况(分别对比例 2-8 和例 3-3),介绍如何确定循环条件,并引入 `while` 语句。教师在讲授时,应重点说明根据具体问题,确定循环条件和循环体的基本思路,并针对 3 种常见的循环控制方式(计数控制、计算值控制和输入值控制),选择合适的循环语句实现。还要介绍 `while` 语句的执行流程和使用方法,通过与 `for` 语句的比较,加深对循环语句的理解,使学生能使用 `for/while` 语句进行循环程序设计。

4.2 节通过案例“统计一个整数的位数”,详细介绍 `do-while` 语句的执行流程以及使用方法,教师在讲授时,应详细介绍 `do-while` 语句的执行流程、循环条件以及循环体,并通过与 `for`、`while` 语句的比较,加深理解 `do-while` 语句,使学生不仅能使用 `do-while` 语句进行循环程序的设计,并且能合理选择循环语句。

4.3 节通过案例“判断素数”,详细介绍循环结构中 `break` 和 `continue` 语句的功能以及执行流程,教师在讲授时,应详细介绍判断素数的算法以及 `break` 和 `continue` 语句的执行流程,使学生能在循环结构中正确使用 `break` 和 `continue` 语句。

4.4 节通过案例“求 $1!+2!+3!+\cdots+100!$ ”,详细介绍嵌套循环的程序设计方法,教师在讲授时,应详细嵌套循环的执行流程(过程),使学生能正确使用嵌套循环进行程序设计。

4.5 节综合介绍循环结构的程序设计,涉及到多个典型的算法,教师在讲授时,重点应放在问题的分析、算法分析以及选择合理的循环语句上。使学生充分理解程序设计的思想与方法。

讲授学时: 4 学时, 实验学时同讲授学时。

本章的知识能力结构图见图 4.1。

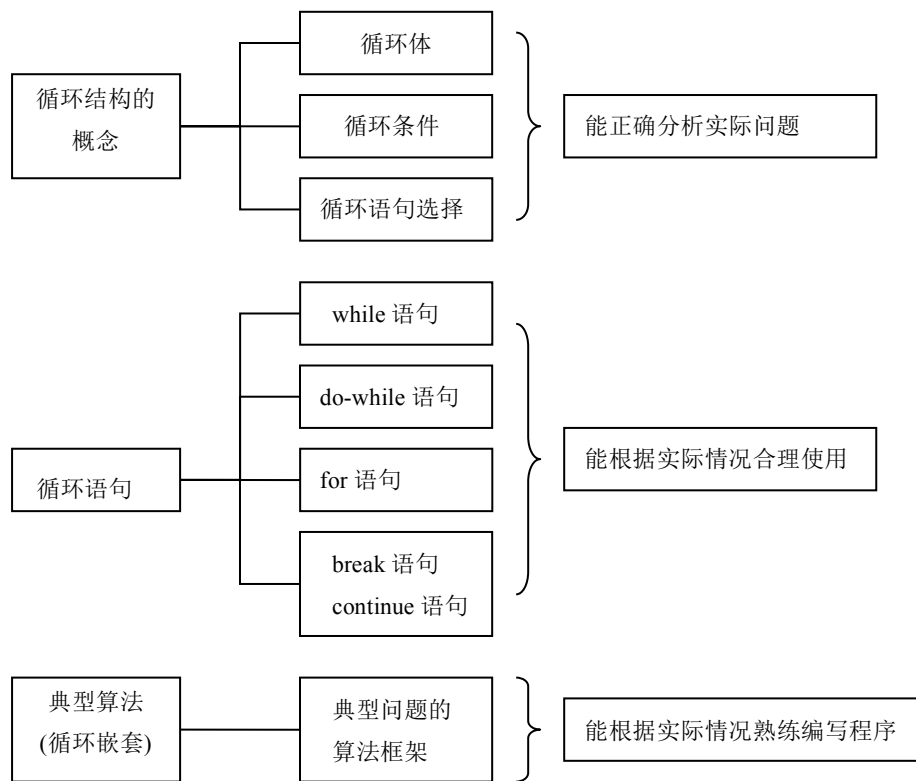
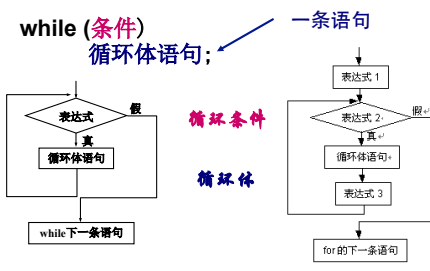


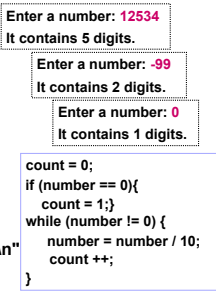
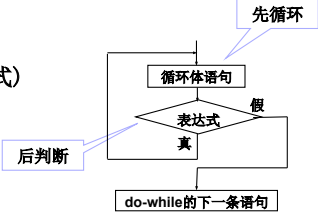
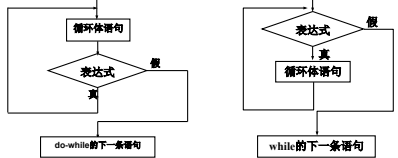
图 4.1 知识能力结构图

4.2 讲稿

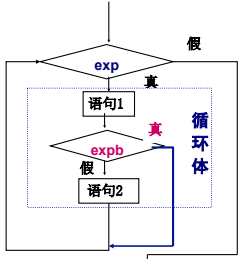

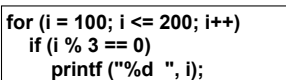
1	<p>第四章 循环结构</p> <p>4.1 用格雷戈里公式求π的近似值 (while语句)</p> <p>4.2 统计一个整数的位数 (do-while语句)</p> <p>4.3 判断素数 (break 和 continue 语句)</p> <p>4.4 求$1! + 2! + \dots + 100!$ (循环嵌套)</p> <p>4.5 循环结构程序设计</p>	本章分 4 节。
2	<p>本章要点</p> <ul style="list-style-type: none"> 什么是循环? 为什么要使用循环? 如何实现循环? 实现循环时, 如何确定循环条件和循环体? 怎样使用while和do-while语句实现次数不确定的循环? while和do-while语句有什么不同? 如何使用break语句处理多循环条件? 如何实现多重循环? 	提出本章的学习要点。
3	<p>4.1 用格里高利公式求π的近似值</p> <p>例4-1 使用格雷戈里公式求π的近似值, 要求精确到最后一项的绝对值小于10^{-4}。</p> $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$ <p>4.1.1 程序解析</p> <p>4.1.2 while语句</p>	<p>借助引例提出要解决的问题, 并比较与例 2-8 的异同, 简要分析, 提供思路, 聚焦在如何确定循环条件。</p> <p>可运行例 4-1 程序, 让学生感受场景。</p> <p>本节介绍为解决这个问题所编写的程序和涉及到的语言知识。</p>
4	<p>4.1.1 程序解析—求π的近似值</p> <p>例4-1 使用格雷戈里公式求π的近似值, 要求精确到最后一项的绝对值小于10^{-4}。</p> <p>例2-8 求$1-1/3+1/5-\dots$ 的前n项和</p> <pre> flag = 1; denominator = 1; sum = 0; item = 1; while (fabs(item) >= 0.0001) { for (i = 4; i <= n; i++) { item = flag * 1.0 / denominator; sum = sum + item; flag = -flag; denominator = denominator + 2; } } </pre> <p>循环结束条件: $item < 0.0001$</p> <p>循环条件: $item \geq 0.0001$ $\text{fabs}(item) \geq 0.0001$</p>	<p>对比分析例 2-8 源程序, 共同点在于循环体不变, 差异在于本例并没有显性给出循环次数, 但给出了循环的结束条件。通过具体分析示例的处理流程, 提出问题:</p> <p>(1) 循环条件是什么?</p> <p>(2) 如何确保循环的初始条件为真? 使循环能正常启动?</p> <p>(3) 程序如何实现?</p> <p>提醒: 学习就是在已有的基础上(循环体不变), 从已知到未知(循环条件变化), 不断探索深化的过程, 要扎实走好每一步。</p>

5	<p>例4-1源程序—求π的近似值</p> <pre> #include <math.h> int main (void) { int denominator, flag; double item, pi; flag = 1; denominator = 1; item = 1.0; pi = 0; while (fabs (item) >= 0.0001) { item = flag * 1.0 / denominator; pi = pi + item; flag = -flag; denominator = denominator + 2; } pi = pi * 4; printf ("pi = %f\n", pi); return 0; } </pre> <p>pi = 3.141613</p> <p>item = 0.0 ?</p> <p>fabs (item) < 0.0001?</p>	<p>展示、运行例 4-1 程序，并解读程序：</p> <p>(1) 程序运行过程：当 item 的值尚未足够小的时候，计算 item，并累加至 pi，周而复始，直到循环条件为假，即 item 已足够小，不再满足其值≥ 0.0001。</p> <p>提醒：循环结束时，最后一次累加至 pi 的 item 之值恰好小于 0.0001，满足题目要求。</p> <p>(2) 简要介绍 while 语句的执行流程。</p> <p>(3) 说明变量初值、循环条件、循环体、循环体语句先后顺序等之间的逻辑关系。</p> <p>设问，以加深对循环条件的理解：</p> <p>(1) 若 item 初值为 0，运行结果？</p> <p>(2) 若循环条件由\geq改成$<$，运行结果？</p> <p>解答：循环初始条件为假，一次也不执行。</p>
6	<p>4.1.2 while 语句</p>  <p>图 4-2 for 语句的执行流程</p>	<p>从例 4-1 分析得知，累加求 π 的循环次数事先不确定，需要在循环累加的过程中反复测试 item 值来决定是否继续循环，即“当条件成立时要做循环”，从而引出 while 语句。</p> <p>说明：</p> <p>(1) while 语句的一般形式；</p> <p>(2) while 语句的执行流程，强调先判断条件；</p> <p>(3) while 语句与 for 语句流程的比较。</p>
7	<p>while 语句说明</p> <p>while 语句和 for 语句 都是在循环前先判断条件</p> <p>for (表达式1; 表达式2; 表达式3)</p> <p>循环体语句</p> <pre> 表达式1; while (表达式2) { for的循环体语句; 表达式3; } </pre> <p>把 for 语句改写成 while 语句</p>	<p>继续对比 while 语句与 for 语句，重点说明两者可互换。应针对具体问题，选择合适的循环语句实现。</p>
8	<p>while 和 for 的比较</p> <pre> sum = 0; for (i = 1; i <= 10; i++){ sum = sum + i; } sum = 0; i = 1; /* 循环变量赋初值 */ while (i <= 10){ /* 循环条件 */ sum = sum + i; i++; /* 循环变量的改变 */ } </pre> <p>循环体</p>	<p>继续讨论 while 语句与 for 语句，举例说明两者可互换，建议让学生练习。</p> <p>强调 while 语句的循环体内必须有能改变循环条件的语句（如 i++），以避免死循环。</p>

9	<h3>统计学生的成绩</h3> <p>例4-2 从键盘输入一批学生的成绩，计算平均成绩，并统计不及格学生的人数。</p> <p>例3-3 输入一个正整数n，再输入n个学生的成绩，计算平均分，并统计不及格成绩的个数。</p> <pre>total = 0; count = 0; for (i = 1; i <= n; i++){ scanf ("%lf", &grade); total = total + grade; if (grade < 60){ count++; } }</pre> <p>如何确定循环条件?</p>	<p>例 4-2 介绍如何根据输入值确定循环条件。可通过运行例 4-2 程序，让学生感受场景（程序运行停不下来，无法结束）。</p> <p>对比分析例 3-3 源程序，共同点在于循环体基本不变，差异在于，本例同样没有显性给出循环次数（与例 4-1 相似），问题也同样聚焦在如何确定循环条件。</p>
10	<h3>分析-统计成绩</h3> <p>从键盘输入一批学生的成绩，计算平均成绩，并统计不及格学生的人数。</p> <p>■ 确定循环条件</p> <ul style="list-style-type: none"> □ 不知道输入数据的个数，无法事先确定循环次数； □ 用一个特殊的数据作为正常输入数据的结束标志，比如选用一个负数作为结束标志。 <p>循环结束条件: <code>grade < 0</code> 循环条件: <code>grade >= 0</code></p>	<p>通过用一个特殊的值作为正常输入数据的结束标志，解决系列输入数据何时结束的问题，这也是解决此类问题的常用方法。即给出循环的结束条件，进而确定循环条件。</p> <p>再次提醒： 如果题目描述的是循环的结束条件，必须将其转换为循环条件。这是 C 语言中循环语句的语法要求。</p>
11	<h3>程序段-统计成绩</h3> <pre>total = 0; count = 0; for (i = 1; i <= n; i++){ scanf ("%lf", &grade); total = total + grade; if (grade < 60){ count++; } }</pre> <pre>num = 0; total = 0; count = 0; scanf ("%lf", &grade); while (grade >= 0) { scanf ("%lf", &grade); total = total + grade; num++; if (grade < 60){ count++; } scanf ("%lf", &grade); }</pre>	<p>具体分析示例的处理流程, 在例 3-3 源程序上修改, 边修改边运行, 渐进式提出问题:</p> <ol style="list-style-type: none"> (1) 循环条件是什么? 已解决 (2) 如何确保循环的初始条件为真? 使循环能正常启动? 正常运转? <p>如果参照例 4-1, while 前 grade 赋值 1 (<code>>=0</code> 皆可), 循环正常启动, 但伪数据也累加至 total。原因在于输入的数据先参加运算, 再判断其值是否满足循环条件。例如: 输入 90 50 -2, total 值: 138 (90+50-2) 输入 -1, total 值: -1</p> <p>解答: 将 while 循环体的输入语句移到最后, 并在 while 前输入第 1 个数。</p> <ol style="list-style-type: none"> (3) 如何求平均值? 增加变量 num 计数
12	<h3>例4-2 源程序-统计成绩</h3> <pre>#include <stdio.h> int main (void) { int count, num; double grade, total; num = 0; total = 0; count=0; printf ("Enter grades: \n"); scanf ("%lf", &grade); /* 输入第1个数*/ while (grade >= 0) { /* 输入负数, 循环结束 */ total = total + grade; num++; if (grade < 60){ count++; } scanf ("%lf", &grade); } if(num != 0) { printf("Grade average is %.2f\n", total/num); printf("Number of failures is %d\n", count); } else{ printf("Grade average is 0\n"); } return 0; }</pre> <p>Enter grades: 67 88 73 54 82 -1 Grade average is 72.80 Number of failures is 1</p>	<p>展示、运行程序，分析程序结构。</p> <p>设问：</p> <ol style="list-style-type: none"> (1) 为什么 while 循环前要先输入第一个数据? (2) 循环体中语句的先后顺序可以任意调整吗? (3) 循环结束后, 为什么要用 if 语句判断?

17	<p>例4-3 源程序-统计整数位数</p> <pre> int main (void) { int count, number; printf ("Enter a number: "); scanf ("%d", &number); if (number < 0){ number = -number; } count = 0; do { number = number / 10; count ++; } while (number != 0); printf ("It contains %d digits.\n", count); return 0; } </pre> 	<p>展示、运行例 4-3 程序，并解读程序：</p> <p>(1) 程序运行过程：从右向左，每划去一个数字($number = number/10$)，计数器 count 增 1，直到所有数字全部被划去 $number == 0$，此时 count 的值就是该整数的位数。</p> <p>(2) 简要介绍 do-while 语句的执行流程。</p> <p>(3) 对负数的处理：-99 与 99 的位数是一样的，故将负数改为正数($number = -number$)</p> <p>设问，本例如何用 while 语句实现？以加深对 do-while/while 的理解。</p> <p>解答：对 0 单独处理。</p>
18	<p>4.2.2 do - while 语句</p> <pre> do { 循环体语句 } while (表达式) </pre> 	<p>从例 4-3 分析得知，先划去一个数字并计数，再判断是否所有数字全部被划去了。即“先执行循环体，再判断条件，当条件成立时继续执行循环体”，从而引出 do-while 语句。</p> <p>说明：</p> <p>(1) do-while 语句的一般形式；</p> <p>(2) do-while 语句的执行流程，强调先循环后判断条件。</p>
19	<p>while 和 do-while 的比较</p> <ul style="list-style-type: none"> while: 先判别条件，再决定是否循环； do-while: 先至少循环一次，然后再根据条件决定是否继续循环。 	<p>对比 do-while 语句与 while 语句，说明：</p> <p>(1) 执行流程的不同；</p> <p>(2) 结合例 4-3，分析循环体的执行次数：当循环的初始条件为假时，do-while 执行 1 次，while 执行 0 次；</p> <p>(3) 两者可互换。</p> <p>应针对具体问题，选择合适的循环语句实现。</p>
20	<p>4.3 判断素数</p> <p>例4-4 输入一个正整数m，判断它是否为素数。素数就是只能被1和自身整除的正整数，1不是素数，2是素数。</p> <p>4.3.1 程序解析</p> <p>4.3.2 break 语句 和 continue 语句</p>	<p>借助引例提出要解决的问题，简要分析，提供思路，即如果只能被 1 和自身整除，则为素数；否则不是素数，这就需要在取值范围内检测能否整除，这是循环问题，应聚焦于实现的算法。</p> <p>可运行例 4-4 程序，让学生感受场景。</p> <p>本节介绍为解决这个问题所编写的程序和涉及到的语言知识。</p>

21	<div><h3>4.3.1 程序解析—判断素数</h3><p>算法：除了 1 和 m，不能被其它数整除。</p><pre>for (i = 2; i <= m/2; i++) if (m % i == 0) break; if (i > m/2) printf ("yes\n") else printf ("no\n");</pre><table><tr><td>m</td><td>%2</td><td>%3</td><td>%4</td><td>%5</td><td>%(m-1)</td></tr><tr><td>不是素数 </td><td>=0</td><td>=0</td><td></td><td></td><td></td></tr><tr><td>是素数 &&</td><td>!=0</td><td>!=0</td><td></td><td></td><td></td></tr></table><p>m不可能被大于 m/2 的数整除 i 取值 [2, m-1]、[2, m/2]、[2, \sqrt{m}]</p></div>	m	%2	%3	%4	%5	%(m-1)	不是素数	=0	=0				是素数 &&	!=0	!=0				<p>算法解析：</p> <p>设 i 的取值范围为[2~m-1]，则：</p> <p>(1) 若 m 是素数，则该范围内的每一个数都必须被测试且满足 $m\%i \neq 0$，即循环测试每一个数，直到所有的数都被测试且不能整除；</p> <p>(2) 一旦遇到能被该范围内的某一个数整除，即 $m\%i == 0$，则该数一定不是素数，此时其余的数不必再继续检测，应中止循环，此时使用 break 语句提前中止循环。</p> <p>If 语句用于区分循环是正常结束(所有的数都被测试且不能整除)，故 m 是素数；还是遇到能整除的数，用 break 提前中止循环，故 m 不是素数。</p>
m	%2	%3	%4	%5	%(m-1)															
不是素数	=0	=0																		
是素数 &&	!=0	!=0																		
22	<div><h3>例4-4 源程序1-判断素数</h3><pre>int main (void) { int i, m; printf ("Enter a number: "); scanf ("%d", &m); for (i = 2; i <= m/2; i++){ if (m % i == 0){ 循环条件? break; } 循环的结束条件? } if (i > m/2 && m != 1){ /* 1不是素数 */ printf ("%d is a prime number! \n", m); } else{ printf ("No!\n"); } }</pre><div><p>Enter a number: 9 No!</p><p>Enter a number: 11 11 is a prime number!</p><p>Enter a number: 1 No!</p><p>Enter a number: 2 2 is a prime number!</p></div></div>	<p>展示、运行例 4-4 程序，并解读程序：</p> <p>(1) 循环条件： $i \leq m/2$ 且 $m\%i \neq 0$</p> <p>(2) 循环结束条件： $i > m/2$ 或者 $m\%i == 0$</p> <p>$i > m/2$：说明 for 循环正常结束； $m\%i == 0$：说明 break 强行终止了循环</p> <p>(3) for 循环结束后必须判断是怎么结束的，即是正常结束还是强行终止，以此来判断是否是素数。</p>																		
23	<div><h3>4.3.2 break 语句</h3><pre>while (exp){ 语句1 if (expb){ break; } 语句2 }</pre><p>当循环有多个出口时：</p><ul style="list-style-type: none">共同表示循环条件区分结束条件</div>	<p>介绍 break 语句的执行流程。重点说明：</p> <p>(1) 如何理解共同表示循环条件？ exp 为真 且 expb 为假；</p> <p>(2) 如何理解循环结束条件？ exp 为假 或 expb 为真；</p> <p>(3) 如何区分循环的结束条件？ 用 if 语句。</p> <p>特别提醒： while 表达式 exp：表示循环条件 if 表达式 expb：表示循环结束条件</p>																		
24	<div><h3>练习-输出结果是什么？</h3><div><pre>for (i = 2; i <= m/2; i++){ if (m%i == 0){ printf ("No!\n"); break; } } printf ("Yes!");</pre><p>Enter a number: 9 No! Yes!</p></div><div><pre>for (i = 2; i <= m/2; i++){ if (m % i == 0) { break; } } if (i > m/2) {printf ("Yes!");} else { printf ("No!\n"); }</pre><p>Enter a number: 9 Yes! No! Yes!</p></div></div>	<p>常见错误解析，建议学生练习。</p>																		


25	<p>continue 语句</p> <pre>while (exp){ 语句1 if (expb) { continue; } 语句2 }</pre>  <p>跳过continue后面的语句，继续下一次循环</p>	<p>介绍 continue 语句的执行流程。</p> <p>提醒：</p> <p>当 expb 为真时，跳过 continue 后面的语句，即提前结束本次循环，重新开始下一次循环。</p>
26	<p>break和continue</p> <pre># include <stdio.h> int main (void) { char c; int i; for (i = 0; i < 10; i++) { c = getchar (); if (c == '\n') continue; putchar (c); } }</pre> 	<p>举例说明 break 与 continue 语句的不同，建议单步运行程序。</p>
27	<p>break和continue</p> <p>输出100~200之间所有能被3整除的数</p> <pre>for (i = 100; i <= 200; i++) { if (i % 3 != 0) continue; printf ("%d ", i); }</pre> 	<p>举例说明 continue 语句的功能，建议单步运行程序。</p> <p>设问：</p> <p>如果将 continue 语句换成 break 语句，程序的运行结果是什么？</p> <p>解答：没有输出。</p>
28	<p>例4-5-1 简单的猜数游戏，最多允许猜7次。</p> <pre># include <stdio.h> int main (void) { int count = 0, flag, mynumber, yournumber; mynumber = 38; /* 计算机指定被猜的数 */ flag = 0; /* flag: 为0表示没猜中，为1表示猜中了 */ for (count = 1; count <= 7; count++) { printf ("Enter your number: "); scanf ("%d", &yournumber); if (yournumber == mynumber) { printf ("Lucky You!\n"); flag = 1; break; /* 已猜中，游戏结束，终止循环 */ } else if (yournumber > mynumber) printf ("Too big!\n"); else printf ("Too small!\n"); } if (flag == 0) printf ("Game Over!\n"); return 0; }</pre>	<p>改写例 3-1 的猜数游戏，增加没猜中可反复猜数的功能，最多猜 7 次。解读程序：</p> <p>(1) 定义变量 flag 表示猜数状态，1 表示猜中；0 表示尚未猜中，不做结论。即：</p> <p>flag=1: yournumber == mynuber, 终止循环 flag=0: yournumber != mynuber, 继续循环</p> <p>(2) 将 flag 的初值置 0。考虑到猜中则终止循环，没猜中则继续循环，flag 初值应选择能使循环继续的值 0。</p> <p>(3) 循环条件：count<=7 且 flag==0；</p> <p>(4) 循环结束条件：count>7 或者 flag==1。 count>7: 此时 flag==0，说明始终没猜中，for 循环正常结束，结束后显示信息； flag==1: 猜中，显示信息，break 终止循环。</p>



29	<p>例4-4源程序2-判断素数</p> <pre> int main (void) { int i, m; printf ("Enter a number: "); scanf ("%d", &m); for (i = 2; i <= m/2; i++){ if (m % i == 0) { break; } } if (i > m/2 && m != 1) { printf ("Yes\n"); } else{ printf ("No\n"); } } </pre> <pre> int main (void) { int i, flag = 1, m; printf ("Enter a number: "); scanf ("%d", &m); if (m == 1) flag = 0; for (i = 2; i <= m/2; i++) { if (m % i == 0){ flag = 0; break; } } if (flag == 1){ printf ("Yes\n"); } else{ printf ("No\n"); } } </pre>	<p>例 4-4 求素数的另一种实现,使用 flag 表示判断素数的状态。解读程序:</p> <p>(1) 变量 flag 表示判断素数的状态, 0 表示不是素数; 1 表示尚未遇到能整除该数的数, 假设其为素数, 不做结论。即: flag=0: m%i == 0, 终止循环 flag=1: m%i != 0, 继续循环</p> <p>(2) 将 flag 的初值置 1。考虑到不是素数则终止循环, 否则继续循环, flag 初值应选择能使循环继续的值 1, 即假设其为素数。</p> <p>(3) 循环条件: i <= m/2 且 flag==1。</p> <p>(4) 循环结束条件: i > m/2 或者 flag==0。 i>m/2: 此时 flag==1, 说明始终没遇到能整除该数的数, for 循环正常结束, m 是素数; flag==0: 遇到能整除该数的数, m 不是素数, break 强行终止了循环。</p>
30	<p>例4-5-2 简单的猜数游戏, 最多允许猜7次。</p> <pre> #include <stdlib.h> #include <time.h> int main (void) { int count = 0, flag = 0, mynumber, yournumber; srand (time(0)); mynumber = rand () % 100 + 1; /* 随机产生一个1~100之间的被猜数 */ while (count < 7){ printf ("Enter your number: "); scanf ("%d", &yournumber); count++; if (yournumber == mynumber) { printf ("Lucky You!\n"); flag = 1; break; } else if (yournumber > mynumber) printf ("Too big!\n"); else printf ("Too small!\n"); } if (flag == 0) printf ("Game Over!\n"); return 0; } </pre>	<p>继续讨论猜数游戏。</p> <p>介绍由计算机随机产生一个 1~100 之间的被猜数的方法。</p> <p>本节小结:</p> <p>(1) 算法: 判断素数, 猜数</p> <p>(2) 变量 flag 的引入, 表示循环继续与否的状态, 如是否猜中、是否为素数的状态;</p> <p>(3) 多出口循环的设计方法:</p> <p>共同表示循环条件: 循环语句+break 区分循环的结束条件: if</p>
31	<p>4.4 求 1! + 2! + ... + 100!</p> <pre> sum = 0; for (i = 1; i <= 100; i++){ item = i ! sum = sum + item; } </pre> <p>4.4.1 程序解析 调用函数 fact(i) 计算 i 的阶乘</p> <p>4.4.2 嵌套循环 用循环计算 i 的阶乘</p>	<p>引导学生分析题目: 这是一个累加求和的问题。</p> <p>先给出解决此类问题的一般框架结构, 从而给出两种不同的编程方法: 函数、循环嵌套。</p>
32	<pre> #include <stdio.h> double fact (int n); int main (void) { int i; double sum; sum = 0; for (i = 1; i <= 100; i++){ sum = sum + fact (i); } printf ("1! + 2! + 3! + ... + 100! = %e\n", sum); return 0; } double fact (int n) { int i; double result = 1; for (i = 1; i <= n; i++){ result = result * i; } return result ; } </pre> <p>4.4.1 程序解析 求 1! + 2! + ... + 100!</p> <p>1! + 2! + ... + 100! = 9.426900e+157</p>	<p>展示、运行例 4-6 程序, 用定义和调用 fact() 函数方法实现。</p> <p>解读程序, 特别说明累加求和循环体内内的第 i 项数据由 fact()函数计算, 其中 for 变量 i 即作为 for 循环的控制变量又同时参与循环体内的计算, 即作为 fact 函数的参数。</p> <p>考虑: 是否可以不调用函数的方法实现呢?</p>

33	<div> <div>4.4.2 嵌套循环</div> <pre> sum = 0; for (i = 1; i <= 100; i++){ item = i ! sum = sum + item; } </pre> <pre> sum = 0; for (i = 1; i <= 100; i++) { item = 1; for (j = 1; j <= i; j++){ item = item * j; } sum = sum + item; } </pre> </div>	<p>前一种方法是用函数来计算第 i 项的值，然后进行累加，另一方面我们已经知道求 i 的阶乘也可用 for 循环实现，故提出了循环的嵌套。</p>
34	<div> <div>例4-7 源程序</div> <pre> #include <stdio.h> int main(void) { int i, j; double item, sum; /* item 存放阶乘 */ sum = 0; for (i = 1; i <= 100; i++) { item = 1; /* 每次求阶乘都从1开始 */ for (j = 1; j <= i; j++){ /* 内层循环算出 item = i! */ item = item * j; } sum = sum + item; } printf ("1! + 2! + 3! + ... + 100! = %e\n", sum); } </pre> </div>	<p>展示、运行例 4-7 程序，用嵌套循环方法实现。</p> <p>解读程序，特别说明累加求和循环体内内的第 i 项数据 item 由内部循环计算，即内循环专门计算 i 的阶乘值。</p>
35	<div> <div>讨论-内层循环的初始化</div> <div> <div>求1! + 2! + ... + 100!</div> <pre> sum = 0; for (i = 1; i <= 100; i++) { item = 1; for (j = 1; j <= i; j++){ item = item * j; } sum = sum + item; } </pre> </div> <div> <pre> item = 1; sum = 0; for(i = 1; i <= 100; i++){ for (j = 1; j <= i; j++){ item = item * j; } sum = sum + item; } </pre> </div> <div>求1!+ 1!*2! + + 1!*2!*.....*100!</div> </div>	<p>重点说明一下几点：</p> <p>循环的控制变量不同；</p> <p>内、外层循环的变量初始值不同，特别时内层循环的初始化语句所放的位置，放置地方不同将对程序功能产生极大的影响;2个程序段的区别；</p>
36	<div> <div>分析嵌套循环的执行过程</div> <pre> sum = 0; for(i = 1; i <= 100; i++) { item = 1; for (j = 1; j <= i; j++) item = item * j; sum = sum + item; } </pre> <ul style="list-style-type: none"> ■ 外层循环变量 i 的每个值 内层循环变量 j 变化一个轮次； ■ 内外层循环变量的名字不能相同 分别用 i 和 j </div>	<p>分析嵌套循环的执行过程。</p> <p>分析计算下列各语句的执行次数：</p> <p>item=1; //100 次</p> <p>item=item*j; //1+2+3+...+100 次</p> <p>sum=sum+item; //100 次</p>

37	<p>练习-运行结果是什么？</p> <pre>for (i = 1; i <= 100; i++) for (j = 1; j <= i; j++) printf ("%d %d\n", i, j);</pre> <table border="1"> <tbody> <tr> <td>i = 1</td><td>j = 1</td><td>输出 1 1 (第 1 次输出)</td></tr> <tr> <td rowspan="2">i = 2</td><td>j = 1</td><td>输出 2 1 (第 2 次输出)</td></tr> <tr> <td>j = 2</td><td>输出 2 2 (第 3 次输出)</td></tr> <tr> <td colspan="3">.....</td></tr> <tr> <td rowspan="4">i = 100</td><td>j = 1</td><td>输出 100 1 (第 4951 次输出)</td></tr> <tr> <td>j = 2</td><td>输出 100 2 (第 4952 次输出)</td></tr> <tr> <td>.....</td><td></td></tr> <tr> <td>j = 100</td><td>输出 100 100 (第 5050 次输出)</td></tr> </tbody> </table>	i = 1	j = 1	输出 1 1 (第 1 次输出)	i = 2	j = 1	输出 2 1 (第 2 次输出)	j = 2	输出 2 2 (第 3 次输出)			i = 100	j = 1	输出 100 1 (第 4951 次输出)	j = 2	输出 100 2 (第 4952 次输出)		j = 100	输出 100 100 (第 5050 次输出)	<p>以 ppt 中程序段为例，深入说明二重循环的执行过程，必要时可单步执行。</p> <p>说明对于每一个 i 值，内层循环控制变量都必须从 1 开始执行，并分析循环体的执行次数。</p>
i = 1	j = 1	输出 1 1 (第 1 次输出)																				
i = 2	j = 1	输出 2 1 (第 2 次输出)																				
	j = 2	输出 2 2 (第 3 次输出)																				
.....																						
i = 100	j = 1	输出 100 1 (第 4951 次输出)																				
	j = 2	输出 100 2 (第 4952 次输出)																				
																					
	j = 100	输出 100 100 (第 5050 次输出)																				
38	<p>4.5 循环结构程序设计</p> <ul style="list-style-type: none"> ■ 循环程序的实现要点 <ul style="list-style-type: none"> □ 归纳出哪些操作需要反复执行？ 循环体 □ 这些操作在什么情况下重复执行？ 循环条件 ■ 常见的循环控制方式 <ul style="list-style-type: none"> □ 计数控制、计算值控制、输入值控制 □ 多重控制（计数控制+计算值控制，.....） ■ 选用合适的循环语句 <pre>for while do-while</pre> 	<p>本节介绍循环结构程序设计思想、方法及算法。</p> <p>实现要点：</p> <p>哪些操作需反复执行？即循环体；</p> <p>什么情况下重复执行？即循环条件；</p> <p>有哪些常见的循环控制方式？</p> <p>适合使用什么循环语句？</p> <p>用什么方法实现？即算法；</p>																				
39	<p>常见的循环控制方式</p> <ul style="list-style-type: none"> ■ 计数控制： 第2.4节, 例4-6, 4-7, 4-11, 4-12 <pre>sum = 0; for (i = 1; i <= n; i++){ sum = sum + fact(i); }</pre> ■ 计算值控制： 例4-1, 4-3, 4-9 <pre>item = 1.0; while (fabs (item) >= 0.0001) { item = flag * 1.0 / denominator; }</pre> ■ 输入值控制为主： 例4-2, 4-8 <pre>scanf ("%lf", &grade); while (grade >= 0) { scanf ("%lf", &grade); }</pre> ■ 计数控制+计算值控制： 例4-4, 4-5, 4-10 	<p>小结常见的循环控制方式及对应的例题</p>																				
40	<p>循环语句的选择</p> <ul style="list-style-type: none"> ■ 主要考虑因素-循环控制方式 <ul style="list-style-type: none"> □ 事先给定循环次数，首选 for □ 通过其他条件控制循环，考虑 while或 do-while <pre>if (循环次数已知) 使用 for 语句 else /* 循环次数未知 */ if (循环条件在进入循环时明确) 使用 while 语句 else /* 循环条件需要在循环体中明确 */ 使用 do-while 语句</pre>	<p>至此已介绍三种不同的循环语句，即 for、while、do-while，说明一般情况下三种循环语句的选择方法，但需强调这不是规则，通常意义下三种循环语句可通用。</p>																				

41	<p>例4-8 输入一批学生的成绩，求最高分(for)</p> <pre>#include <stdio.h> int main(void) { int i, mark, max, n; printf("Enter n: "); scanf("%d", &n); printf("Enter %d marks: ", n); scanf("%d", &mark); /* 读入第一个成绩 */ max = mark; /* 假设第一个成绩是最高分 */ for (i = 1; i < n; i++) { scanf("%d", &mark); if (max < mark) max = mark; } printf("Max = %d\n", max); return 0; }</pre> <p>Enter n: 5 Enter 5 marks: 67 88 73 54 82 Max = 88</p> <p>mark → max mark → max</p> <p>Enter n: 0</p>	<p>例 4-8 重点要解决的是：在若干个数据中求最大（小）值的问题。分数据个数已定、数据个数不定两种情形。</p> <p>ppt 首先给出的是数据个数确定的情形，故可采取用 for 循环实现。</p> <p>需重点说明求最大值 max 的方法，变量 max 的作用是始终存放当前的最大值。</p> <p>类推到求最小值。</p>
42	<p>例4-8 输入一批学生的成绩，求最高分(while)</p> <pre>#include <stdio.h> int main (void) { int mark, max; printf("Enter marks:"); scanf ("%d", &mark); /* 读入第一个成绩 */ max = mark; /* 假设第一个成绩最高分 */ while (mark >= 0){ if (max < mark) max = mark ; scanf ("%d", &mark); }; printf ("Max = %d\n", max); return 0; }</pre> <p>Enter marks: 67 88 73 54 82 -1 Max = 88</p> <p>Enter marks: -1</p>	<p>本 ppt 给出的是数据个数不确定的情形，用输入一个特殊值作为结束标志，故可采取用 while 循环实现。</p> <p>循环前先读入一个数据，作为当前的最大值 max 的初值，并作为 while 循环判断的依据，一旦输入一个负数，循环结束。</p> <p>引导学生考虑前 2 个程序是否有缺陷？可运行测试。</p>
43	<p>例4-8 输入一批学生的成绩，求最高分(do-while)</p> <pre>#include <stdio.h> int main (void) { int mark, max; max = -1; /* 给max赋一个小初值*/ printf ("Enter marks: "); do { scanf ("%d", &mark); if (max < mark) max = mark; } while (mark >= 0); printf ("Max = %d\n ", max); }</pre> <p>Enter marks: 67 88 73 54 82 -1 Max = 88</p> <p>Enter marks: -1</p>	<p>介绍第 3 种方法，用 do-while 循环实现。运行程序，并进行解读。</p> <p>考虑：当输入数据时第一个就输入-1，程序运行结果是什么？为什么？</p>
44	<p>例4-9 逆序问题。将一个正整数逆序输出</p> <p>确定：循环条件和循环体(循环不变式)</p> <p>12345 5 4 3 2 1</p> <p>12345 % 10 = 5 12345 / 10 = 1234 1234 % 10 = 4 1234 / 10 = 123 123 % 10 = 3 123 / 10 = 12 12 % 10 = 2 12 / 10 = 1 1 % 10 = 1 1 / 10 = 0 结束</p> <p>循环不变式 x%10 x=x/10 scanf ("%d", &x); 循环结束条件 x==0 while (x != 0){ digit = x %10; x = x/10; printf ("%d ", digit); }</p> <p>用do-while实现?</p>	<p>例 4-9 重点要解决的是：对一个数据进行逐位分拆的问题。也可分为逆序和正序分拆两种情形。</p> <p>本例已逆向分拆为例。</p> <p>需说明：</p> <p>逆向分拆的方法；</p> <p>循环条件是什么？</p> <p>循环体是什么？</p> <p>程序结构怎么搭？用 while 循环还是 do-while 循环？有和区别？</p>

45	<p>例4-10 求100以内的全部素数，每行输出10个</p> <pre>for (m = 2; m <= 100; m++) if (m是素数) printf("%d", m);</pre> <div><pre>n = sqrt(m); for (i = 2; i <= n; i++) if (m % i == 0) break; if(i > n) printf ("yes\n") else printf ("no\n");</pre></div> <pre>for (m = 2; m <= 100; m++){ n = sqrt(m); for (i = 2; i <= n; i++) if (m % i == 0) break; if (i > n) printf ("%d", m) }</pre>	<p>例 4-10 需要解决的是：用嵌套循环方式找出某个范围内的所有素数以及如何实现每行几个的问题。</p> <p>先实现找出所有的素数，不考虑分组输出，可逐步细化介绍此程序段。</p>																																				
46	<div><pre>#include <stdio.h> #include <math.h> int main(void) { int count, i, m, n; count = 0; for (m = 2; m <= 100; m++){ n = sqrt(m); for (i = 2; i <= n; i++) if (m % i == 0) break; if (i > n){ /* 如果m是素数 */ printf ("%6d", m); count++; /* 每行10个的处理 */ if (count %10 == 0) printf ("\n"); } } }</pre></div> <p>例4-10 源程序</p> <table><tr><td>2</td><td>3</td><td>5</td><td>7</td><td>11</td><td>13</td><td>17</td><td>19</td><td>23</td><td>29</td></tr><tr><td>31</td><td>37</td><td>41</td><td>43</td><td>47</td><td>53</td><td>59</td><td>61</td><td>67</td><td>71</td></tr><tr><td>73</td><td>79</td><td>83</td><td>89</td><td>97</td><td></td><td></td><td></td><td></td><td></td></tr></table>	2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97						<p>展示、运行程序，说明一下几个问题：</p> <p>内外层循环的变量初始值；</p> <p>外层循环的循环体是什么？</p> <p>内层循环的循环体是什么？</p> <p>变量 count 的作用是什么？</p> <p>如何实现每行输出 10 个？</p> <p>注意：每行输出几个是初学者容易出错的地方。</p>						
2	3	5	7	11	13	17	19	23	29																													
31	37	41	43	47	53	59	61	67	71																													
73	79	83	89	97																																		
47	<p>例4-11 求Fibonacci序列：1,1,2,3,5,8,13,...</p> <table><tr><td>1</td><td>1</td><td>2</td><td>3</td><td>5</td><td>8</td><td>13</td><td>.....</td><td>x1 = x2 = 1;</td></tr><tr><td>x1</td><td>x2</td><td>x</td><td></td><td></td><td></td><td></td><td></td><td>x = x1 + x2;</td></tr><tr><td></td><td>x1</td><td>x2</td><td>x</td><td></td><td></td><td></td><td></td><td>x1 = x2;</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>x2 = x;</td></tr></table> <div><pre>x1 = 1; x2 = 1; printf ("%6d%6d", x1, x2); /* 输出头两项 */ for (i = 1; i <= 8; i++){ /* 循环输出后8项 */ x = x1 + x2; /* 计算新项 */ printf("%6d", x); x1 = x2; /* 更新x1和x2 */ x2 = x; }</pre></div>	1	1	2	3	5	8	13	x1 = x2 = 1;	x1	x2	x						x = x1 + x2;		x1	x2	x					x1 = x2;									x2 = x;	<p>例 4-11 用斐波那契数列为例，说明数列问题如何推算？</p> <p>引导学生分析问题，归纳出需要重复执行的是什么？重复执行的条件是什么？怎么搭程序框架？</p> <p>这是经典的算法例子。</p>
1	1	2	3	5	8	13	x1 = x2 = 1;																														
x1	x2	x						x = x1 + x2;																														
	x1	x2	x					x1 = x2;																														
								x2 = x;																														
48	<div></div> <p>例4-12古典算术问题—搬砖头</p> <p>某地需要搬运砖块，已知男人一人搬3块，女人一人搬2块，小孩两人搬一块。 问用45人正好搬45块砖，有多少种搬法？</p> <pre>for (men = 0; men <= 45; men++) for (women = 0; women <= 45; women++) for (child = 0; child <= 45; child++) if ((men+women+child==45) && (men*3+women*2+child*0.5==45)) printf("men=%d women=%d child=%d\n", men, women, child); }</pre>	<p>例 4-12 以搬转头为例，说明穷举法的算法方法。</p> <p>引导学生分析问题，男人、女人和小孩的人数取值范围均在 0~45 之间，只要能满足总人数为 45 人且所搬的砖为 45 块就是一种搬法，显然要对男人、女人和小孩三个人数的各种可能一一测试，来找出满足条件的人员组合，可采用三重循环。</p> <p>这也是经典的算法例子。</p>																																				

49	 <p>例4-12 源程序(2)</p> <pre> for (men = 0; men <= 15; men++) for (women = 0; women <= 22; women++){ child = 45 - women - men; if (men * 3 + women * 2 + child * 0.5 == 45) printf("men=%d women=%d child=%d\n", men, women, child); } </pre> <p style="text-align: right;">比较循环次数</p> <pre> for (men = 0; men <= 45; men++) for (women = 0; women <= 45; women++) for (child = 0; child <= 45; child++) if ((men+women+child==45) && (men*3+women*2+child*0.5==45)) printf("men=%d women=%d child=%d\n", men, women, child); </pre>	<p>引导学生考虑：上述三重循环的循环体（if 语句）共执行了多少次？是否可减少循环次数？</p> <p>说明如何把三重循环改成二重循环？并比较循环体的执行次数，分析减少的方法。</p>
50	 <p>本章总结</p> <ul style="list-style-type: none"> ■ 循环结构以及 <ul style="list-style-type: none"> □ while do-wh ■ break与cont ■ 嵌套循环 ■ 综合程序设计（循环结构） ■ 常用算法 <ul style="list-style-type: none"> □ 例题：求π、拆分整数、求素数、猜数、求最值、求fibonacci □ 习题：求水仙花数、求最大公约数/最小公倍数、求ddd <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <ul style="list-style-type: none"> • 理解 while和 do-while的执行机制； • 掌握 break 和 continue 的作用方式； • 掌握嵌套循环的执行机制与设计方法； • 能熟练循环语句编写循环结构类的程序； • 熟练掌握常用算法 </div>	<p>归纳总结本章的各个重要知识点。</p>

4.3 练习与习题参考答案

4.3.1 练习参考答案

练习 4-1 在例 4-1 程序中，如果对 item 赋初值 0，运行结果是什么？为什么？如果将精度改为 10^{-3} ，运行结果有变化吗？为什么？

解答：

如果对 item 赋初值 0，则程序运行结果是 $\pi=0$ ，因为 item 为 0 时不满足 while 循环的执行条件，即不执行循环，故 π 值为 0。如果将精度改为 10^{-3} ，运行结果会有变化，因为精度改变意味着 while 循环的条件改变，精度变大使得 while 循环次数减少，必然影响到 π 的值。

练习 4-2 运行例 4-2 程序时，如果将最后一个输入数据改为-2，运行结果有变化吗？如果第一个输入数据是 -1，运行结果是什么？为什么？

解答：

如果将最后一个输入数据改为-2，运行结果没有变化，因为最后一个负数是一结束标志，不进行统计，故任意一个负数都可以。如果第一个输入数据是-1，运行结果是：Grade average is 0，因为第一个输入就是-1，则 while 循环条件不成立，不执行循环体。

练习 4-3 序列求和 ($1-1/4+1/7-1/10+1/13-1/16+\cdots$)：输入一个正实数 eps，计算序列 $1-1/4+1/7-1/10+1/13-1/16+\cdots$ 的值，直到最后一项的绝对值不大于给定精度 eps

(保留 6 位小数)。试编写相应程序。

提醒：教材上此题对最后一项取值的描述为“精确到最后一项的绝对值小于 `eps`”。下次印刷时将修改一致。

解答：

```
#include <stdio.h>
#include <math.h>
int main (void)
{
    int flag;
    double denominator, eps, item, sum;

    flag = 1; denominator = 1 ; item = 1.0; sum = 0;
    scanf("%lf", &eps);
    while (fabs (item) > eps) {
        sum = sum + item;
        flag = -flag;
        denominator = denominator + 3;
        item = flag * 1.0 / denominator;
    }
    sum = sum + item;
    printf ("sum = %.6f\n", sum);

    return 0;
}
```

练习 4-4 如果将例 4-3 程序中的 `do-while` 语句改为下列 `while` 语句，会影响程序的功能吗？为什么？再增加一条什么语句，就可以实现同样的功能？

```
while (number != 0){
    number = number / 10;
    count ++;
}
```

解答：

会有影响，因为当输入数据 `number` 为 0 时，上述 `while` 循环将不执行，`count` 值仍为 0，故输出为 0，不符合题目要求。可增加一条 `if` 语句来解决上面的问题，在 `while` 循环前加上语句 “`if (number == 0) count = 1;` ”。

练习 4-5 例 4-4 程序中的第 9~15 行可以用下列 `for` 语句替代吗？为什么？

```
for (i = 2; i <= m/2; i++)
    if (m%i == 0) printf ("No!\n");
    else printf ("%d is prime number!\n", m);
```

解答：

不能代替，因为只用一个数来除不足以判断该数是否是素数。

练习 4-6 猜数字游戏：先输入 2 个不超过 100 的正整数，分别是被猜数 `mynumber` 和猜测的

最大次数 n ，再输入你所猜的数 `yournumber`，与被猜数 `mynumber` 进行比较，若相等，显示猜中；若不等，显示与被猜数的大小关系，最多允许猜 n 次。如果 1 次就猜出该数，提示“Bingo!”；如果 3 次以内猜到该数，则提示“Lucky You!”；如果超过 3 次但不超过 n 次猜到该数，则提示“Good Guess!”；如果超过 n 次都没有猜到，则提示“Game Over”。如果在到达 n 次之前，用户输入了一个负数，也输出“Game Over”，并结束程序。试编写相应程序。

解答：

源程序 1

```
#include <stdio.h>

int main (void)
{
    int count = 0, flag = 0, mynumber, n, yournumber;

    scanf ("%d%d", &mynumber, &n);
    if (n > 100 || mynumber > 100)
        printf ("Invalid number!\n");
    else{
        while (count < n) {
            scanf ("%d", &yournumber);
            count ++;
            if (yournumber < 0) {
                break;
            }
            if (yournumber == mynumber) {
                flag = 1;
                break;
            } else if (yournumber > mynumber ) {
                printf ("Too big\n");
            } else {
                printf ("Too small\n");
            }
        }
        if (flag == 0) {
            printf ("Game Over\n");
        } else if (count == 1) {
            printf ("Bingo!\n");
        } else if (count <= 3) {
            printf ("Lucky You!\n");
        } else {
            printf ("Good Guess!\n");
        }
    }

    return 0;
}
```

```
}
```

源程序 2

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

```
int main (void)
```

```
{
```

```
    int count = 0, flag = 0, mynumber, n, yournumber;
```

```
    scanf ("%d%d", &mynumber, &n);
```

```
    if (n > 100 || mynumber > 100)
```

```
        printf ("Invalid number!\n");
```

```
    else{
```

```
        scanf ("%d", &yournumber);
```

```
        while (yournumber >= 0) {
```

```
            count ++;
```

```
            if (count > n) {
```

```
                break;
```

```
            }
```

```
            if (yournumber == mynumber) {
```

```
                flag = 1;
```

```
                break;
```

```
            }
```

```
            else if (yournumber > mynumber ){
```

```
                printf ("Too big\n");
```

```
            }
```

```
            else{
```

```
                printf ("Too small\n");
```

```
            }
```

```
            scanf ("%d", &yournumber);
```

```
        }
```

```
    if (flag == 0) {
```

```
        printf ("Game Over\n");
```

```
    }else if (count == 1) {
```

```
        printf ("Bingo!\n");
```

```
    }else if (count <= 3) {
```

```
        printf ("Lucky You!\n");
```

```
    }else {
```

```
        printf ("Good Guess!\n");
```

```
    }
```

```
}
```

```
    return 0;
```

```
}
```

练习 4-7 求 e 的值：输入 1 个正整数 n，计算下式求出 e 的值（保留 8 位小数），要求使用嵌套循环。

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

解答：

```
#include <stdio.h>
int main(void)
{
    int i, j, n;
    double e, product;

    scanf("%d", &n);
    e = 1;
    for (i = 1; i <= n; i++){
        product = 1;
        for (j = 1; j <= i; j++)
            product = product * j;
        e = e + 1.0 / product;
    }
    printf("%.8f\n", e);

    return 0;
}
```

练习 4-8 运行例 4-8 的源程序 1 时，如果先输入 0，即输入数据个数 n=0，表示不再输入任何成绩，运行结果是什么？如何修改程序以应对这种情况？

解答：

当输入数据个数 n=0 时，由于在 for 循环外首先得输入一个成绩，故程序仍将等待输入第一个人的成绩，这与输入个数 n=0 矛盾，可修改如下：

```
#include <stdio.h>
int main (void)
{
    int i, mark, max, n;

    printf ("Enter n: ");
    scanf ("%d", &n);
    if (n > 0) {
        printf ("Enter %d marks: ", n);
        scanf ("%d", &mark);          /* 读入第一个成绩 */
        max = mark;                   /* 假设第一个成绩是最高分 */
        for (i = 1; i < n; i++) {
            scanf ("%d", &mark);
            if (max < mark)
```

```

        max = mark;
    }
    printf ("Max = %d\n", max);
}

return 0;
}

```

练习 4-9 运行例 4-8 的源程序 2 时, 如果输入的第一个数就是负数, 表示不再输入任何成绩, 运行结果是什么? 如何修改程序以应对这种情况?

解答:

当输入的第一个数就是负数时, 将不执行 while 循环, 程序运行结果就是输出所输入的负数。修改思路同上题。

练习 4-10 找出最小值: 输入一个正整数 n, 再输入 n 个整数, 找出其中的最小值。试编写相应程序。

解答:

```

#include <stdio.h>
int main (void)
{
    int i, min, n, x;

    scanf ("%d", &n);
    scanf ("%d", &x);
    min = x;
    for (i = 1; i < n; i++){
        scanf ("%d", &x);
        if (min > x) min = x;
    }
    printf ("min = %d\n", min);

    return 0;
}

```

练习 4-11 统计素数并求和: 输入 2 个正整数 m 和 n ($1 \leq m \leq n \leq 500$), 统计并输出 m 和 n 之间素数的个数以及这些素数的和。素数就是只能被 1 和自身整除的正整数, 1 不是素数, 2 是素数。试编写相应程序。

解答:

```

#include <stdio.h>
#include <math.h>
int main (void)
{
    int count = 0, flag, i, j, k, m, n, sum = 0;

```

```

scanf ("%d%d", &m, &n);
if (m >= 1 && n <= 500 && m <= n){
    for (k = m; k <= n; k++){
        if (k == 1) {
            flag = 0;
        } else {
            flag = 1;
        }
        j = sqrt (k);
        for (i = 2; i <= j; i++){
            if (k % i == 0){
                flag = 0;
                break;
            }
        }
        if (flag == 1){
            count++;
            sum = sum + k;
        }
    }
}
printf("%d %d\n", count, sum);

return 0;
}

```

4.3.2 习题参考答案

一. 选择题

1	2	3	4	5	6	7	8
A	C	D	A	B	B	C	A

二. 填空题

- 13
- **
- t*10
- 12
- (1) k=6,m=2
(2) k=6,m=9
(3) k=3,m=6
- 62ufd!

三. 程序设计题

1. 求奇数和。输入一批正整数（以零或负数为结束标志），求其中的奇数和。试编写相应

程序。

解答：

```
#include <stdio.h>

int main (void)
{
    int sum, x;

    sum = 0;
    scanf ("%d", &x);
    while (x > 0){
        if (x % 2 != 0){
            sum = sum + x;
        }
        scanf ("%d", &x);
    }
    printf ("%d\n", sum);

    return 0;
}
```

2. 展开式求和。输入一个实数 x ，计算并输出下式的值，直到最后一项的绝对值小于 0.00001（保留 4 位小数）。要求定义和调用函数 fact(n) 计算 n 的阶乘，可以调用 pow() 函数求幂。试编写相应程序。

$$s = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

提醒：教材上此题的计算公式见下式，且计算结果要求保留 2 位小数。下次印刷时将修改一致。

$$s = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

解答：

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

int main (void)
{
    int i;
    double item, s, x;
    double fact (int n);

    scanf ("%le", &x);
    s = 1;
    item = x;
    i = 1;
    while (fabs (item) >= 0.00001) {
```

```

        item = pow (x, i) / fact(i);
        s = s+ item;
        i++;
    }
    printf ("%%.4f\n", s);

    return 0;
}

```

```

double fact (int n)
{
    int i;
    double result = 1;

    for(i = 1; i <= n; i++){
        result = result * i;
    }

    return result;
}

```

3. 求序列前 n 项和 ($2/1+3/2+5/3+8/5+\dots$)。输入一个正整数 n ，输出 $2/1+3/2+5/3+8/5+\dots$ 的前 n 项之和，保留 2 位小数。该序列从第 2 项起，每一项的分子是前一项分子与分母的和，分母是前一项的分子。试编写相应程序。

解答：

```

#include <stdio.h>
int main (void)
{
    int i, n;
    double denominator, numerator, sum, temp;

    scanf ("%d", &n);
    numerator = 2;
    denominator = 1;
    sum = 0;
    for (i = 1; i <= n; i++){
        sum = sum + numerator / denominator;
        temp = numerator;
        numerator = numerator + denominator;
        denominator = temp;
    }
    printf ("%%.2f\n", sum);

    return 0;
}

```

```
}
```

4. 求序列前 n 项和 ($a+aa+aaa+aa\cdots a$)。输入两个正整数 a 和 n ，求 $a+aa+aaa+aa\cdots a$ (n 个 a) 之和。例如，输入 2 和 3，输出 246 ($2 + 22 + 222$)。试编写相应程序。

解答：

```
#include <stdio.h>

int main (void)
{
    int a, i, n, sn, tn;

    scanf("%d%d", &a, &n);
    tn = 0;
    sn = 0;
    for (i = 1; i <= n; i++){
        tn = tn * 10 + a;
        sn = sn + tn;
    }
    printf("s = %d\n", sn);

    return 0;
}
```

5. 换硬币。将一笔零钱（大于 8 分，小于 1 元，精确到分）换成 5 分、2 分和 1 分的硬币，每种硬币至少有一枚。输入金额，问有几种换法？针对每一种换法，输出各种面额硬币的数量和硬币的总数量。试编写相应程序。

解答：

```
#include <stdio.h>

int main(void)
{
    int count, fen1, fen2, fen5, x;

    scanf("%d", &x);
    count = 0;
    for (fen5 = (x - 3) / 5; fen5 > 0; fen5--){
        for (fen2 = (x - fen5 * 5 - 1) / 2; fen2 > 0; fen2--){
            fen1 = x - 5 * fen5 - 2 * fen2;
            if (fen1 > 0){
                count++;
                printf("fen5:%d, fen2:%d, fen1:%d, total:%d\n", fen5, fen2, fen1, fen1 + fen2 + fen5);
            }
        }
    }

    printf("count = %d\n", count);
}
```



```
    return 0;
}
```

6. 输出水仙花数。输入一个正整数 n ($3 \leq n \leq 7$), 输出所有的 n 位水仙花数。水仙花数是指一个 n 位正整数, 它的各位数字的 n 次幂之和等于它本身。例如 153 的各位数字的立方和是 $1^3+5^3+3^3=153$ 。试编写相应程序。

提示: 定义函数 `ipow(x, n)` 计算 x 的 n 次幂, 返回值为整型。

解答:

```
#include <stdio.h>

int ipow (int x, int n)
{
    int i, p = 1;

    for (i = 1; i <= n; i++) {
        p = p * x;
    }

    return p;
}

int main (void)
{
    int a, b, digit, i, n, number, s;

    scanf ("%d", &n);
    a = ipow (10, n-1);
    b = a*10;
    for (i = a; i < b; i++) {
        s = 0;
        number = i;
        while (number != 0){
            digit = number % 10;
            number = number / 10;
            s = s + ipow (digit, n);
        }
        if (i == s){
            printf ("%d\n", s);
        }
    }

    return 0;
}
```

7. 求最大公约数和最小公倍数。输入两个正整数 m 和 n ($m \leq 1000, n \leq 1000$), 求其最大公约数和最小公倍数。试编写相应程序。

解答:

```
#include <stdio.h>

int main (void)
{
    int j, k, m, n; /* j 表示最小公倍数, k 表示最大公约数 */

    scanf ("%d%d", &m, &n);
    j = m;
    while (j % n != 0){
        j = j + m;
    }
    k = (m * n) / j;
    printf ("%d %d\n", k, j);

    return 0;
}
```

8. 高空坠球。皮球从 $height$ (米) 高度自由落下, 触地后反弹到原高度的一半, 再落下, 再反弹, …… , 如此反复。问皮球在第 n 次落地时, 在空中一共经过多少距离? 第 n 次反弹的高度是多少? 输出保留 1 位小数。试编写相应程序。

解答:

```
#include <stdio.h>

int main (void)
{
    double h, distance;
    int height, i, n;

    scanf ("%d%d", &height, &n);
    if (height == 0 || n == 0) {
        printf ("0.0 0.0\n");
    }
    else {
        distance = height;
        h = 0.5 * height;
        for (i = 1; i < n; i++) {
            distance = distance + 2 * h;
            h = 0.5 * h;
        }
        printf ("%0.1f %0.1f\n", distance, h);
    }

    return 0;
}
```

```
}
```

9. 打印菱形“星号*”图案。输入一个正整数 n (n 为奇数)，打印一个高度为 n 的“*”菱形图案。例如，当 n 为 7 时，打印出以下图案。试编写相应程序。

```
  *
 * * *
* * * * *
* * * * * *
 * * * * *
  * * *
    *
```

解答：

源程序 1

```
#include <stdio.h>
int main (void)
{
    int i,j, n;

    scanf ("%d", &n);
    for (i = 1; i <= n/2 + 1; i++){
        for (j = 2 *(n/2 + 1 - i); j > 0; j--){
            printf (" ");
        }
        for (j = 1; j <= 2 * i - 1; j++){
            printf ("* ");
        }
        printf ("\n");
    }
    for (i = 1; i <= n/2; i++){
        for (j = 1; j <= 2 * i; j++){
            printf (" ");
        }
        for (j = 1; j <= 2 * (n/2 + 1 - i) - 1; j++){
            printf ("* ");
        }
        printf ("\n");
    }

    return 0;
}
```

源程序 2

```
#include <stdio.h>
int main (void)
```

```

{
    int i, j, n;

    scanf ("%d", &n);
    for (i = 1; i <= n/2 + 1; i++){
        for (j = 2 *(n/2 + 1 - i); j > 0; j--){
            printf (" ");
        }
        for (j = 1; j <= 2 * i - 1; j++){
            printf ("* ");
        }
        printf ("\n");
    }
    for (i = n/2; i >= 1; i--){
        for (j = 2 *(n/2 + 1 - i); j > 0; j--){
            printf (" ");
        }
        for (j = 1; j <= 2 * i - 1; j++){
            printf ("* ");
        }
        printf ("\n");
    }

    return 0;
}

```

10. 猴子吃桃问题。一只猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个；第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半加一个。到第 n 天早上想再吃时，见只剩下一个桃子了。问：第一天共摘了多少个桃子？试编写相应程序。（提示：采取逆向思维的方法，从后往前推断。）

解答 1：假设第 1 天摘的桃子个数是偶数，PTA 上的题目以此为前提。

```

#include <stdio.h>
int main (void)
{
    int i, n, peach;

    scanf ("%d", &n);
    peach = 1;
    for (i = 1; i < n; i++){
        peach = 2 * (peach + 1);
    }
    printf ("%d\n", peach);

    return 0;
}

```

```
}
```

解答 2: 假设第 1 天摘的桃子个数是奇数。

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

```
int main (void)
```

```
{
```

```
    int i, n, peach;
```

```
    scanf ("%d", &n);
```

```
    peach = 1;
```

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

```
        peach = 2 * peach + 1;
```

```
    }
```

```
    printf ("%d\n", peach);
```

```
    return 0;
```

```
}
```

11. 兔子繁衍问题。一对兔子，从出生后第 3 个月起每个月都生一对兔子。小兔子长到第 3 个月后每个月又生一对兔子。假如兔子都不死，请问第 1 个月出生的一对兔子，至少需要繁衍到第几个月时兔子总数才可以达到 n 对？输入一个不超过 10000 的正整数 n ，输出兔子总数达到 n 最少需要的月数。试编写相应程序。

解答：分析：

月 month	1	2	3	4	5
1 月龄兔子对数 n_1	1	0	1	1	2
2 月龄兔子对数 n_2	0	1	0	1	1
≥ 3 月龄兔子对数 n_3	0	0	1	1	2
兔子总数 n	1	1	2	3	5

源程序：

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

```
int main (void)
```

```
{
```

```
    int n, n1, n2, n3, month;
```

```
    scanf ("%d", &n);
```

```
    month = 1;
```

```
    n1 = 1;
```

```
    n2 = n3 = 0;
```

```
    while ((n1+n2+n3) < n) {
```

```
        month ++;
```

```
        n3 = n3 + n2;
```

```
        n2 = n1;
```

```
        n1 = n3;
```

```
    }
```

```

printf ("%d\n", month);

return 0;
}

```

4.4 实验指导教材参考答案

4.4.1 基本循环语句的使用

一、调试示例

最大公约数和最小公倍数：输入两个正整数 m 和 n ，输出它们的最大公约数和最小公倍数。

解答：参见习题程序设计第 7 题。

二、基础编程题

(1) 求奇数和：读入一批正整数（以零或负数为结束标志），求其中的奇数和。请使用 `while` 语句实现循环。

解答：参见习题程序设计第 1 题。

(2) 求最小值：输入一个正整数 n ，再输入 n 个整数，输出最小值。

解答：参见练习 4-10。

(3) 求整数的位数以及各位数字之和：输入一个整数，求它的位数以及各位数字之和。例如，123 的位数是 3，各位数字之和是 6。

解答：

```

#include <stdio.h>
int main (void)
{
    int count, number, sum;

    scanf ("%d", & number);
    if (number < 0) {
        number = - number;
    }
    count = 0;
    sum = 0;
    while (number != 0){
        sum = sum + number % 10;
        number = number / 10;
        count ++;
    }
    printf ("%d %d\n", count, sum);
}

```

```

    return 0;
}

```

(4) 韩信点兵：在中国数学史上，广泛流传着一个“韩信点兵”的故事：韩信是汉高祖刘邦手下的大将，他英勇善战，智谋超群，为汉朝建立了卓越的功劳。据说韩信的数学水平也非常高超，他在点兵的时候，为了知道有多少兵，同时又能保住军事机密，便让士兵排队报数：

按从 1 至 5 报数，记下最末一个士兵报的数为 1；
 再按从 1 至 6 报数，记下最末一个士兵报的数为 5；
 再按从 1 至 7 报数，记下最末一个士兵报的数为 4；
 最后按从 1 至 11 报数，最末一个士兵报的数为 10；
 你知道韩信至少有多少兵？

解答：

```

#include <stdio.h>
int main (void)
{
    int n;

    n = 0;
    while (n % 5 != 1 || n % 6 != 5 || n % 7 != 4 || n % 11 != 10){
        n++;
    }
    printf ("%d\n", n);

    return 0;
}

```

(5) 求序列前 N 项和：输入一个正整数 n，输出 $2/1 + 3/2 + 5/3 + 8/5 + \dots$ 的前 n 项之和（该序列从第二项起，每一项的分子是前一项分子与分母的和，分母是前一项的分子），保留 2 位小数。

解答：参见习题程序设计第 3 题。

(6) 求 $a + aa + aaa + aa\cdots a$ ：输入两个正整数 a 和 n，求 $a + aa + aaa + aa\cdots a$ (n 个 a) 之和。试编写相应程序。

解答：参见习题程序设计第 4 题。

三、改错题

序列求和：输入一个正实数 eps，计算并输出下式的值，直到最后一项的绝对值不大于于 eps（保留 6 位小数）。请使用 do-while 语句实现循环。（源程序 error04_2.cpp）

$$s = 1 - \frac{1}{4} + \frac{1}{7} - \frac{1}{10} + \frac{1}{13} - \frac{1}{16} + \dots$$

提醒：教材上此题对最后一项取值的描述为“精确到最后一项的绝对值小于 eps”。下次印刷时将修改一致。

解答 1:

```
#include <stdio.h>
#include <math.h>
int main (void)
{
    int flag;
    double denominator, eps, item, sum;

    flag = 1; denominator = 1 ; item = 1.0; sum = 0;
    scanf("%lf", &eps);
    do{
        item = flag * 1.0 / denominator;
        sum = sum + item;
        flag = -flag;
        denominator = denominator + 3;
    }while (fabs (item) > eps);
    printf ("sum = %.6f\n", sum);

    return 0;
}
```

解答 2: 用 while 实现, 参见练习 4-3。

四、拓展编程题

(1) 猜数字游戏: 输入一个 100 以内的正整数, 用户再输入一个数对其进行猜测, 需要你编写程序自动对其进行比较, 并提示大了 (“Too big”), 还是小了 (“Too small”), 相等表示猜到了。如果猜到, 则结束程序。程序还要求统计猜的次数, 如果 1 次猜出该数, 提示 “Bingo!”; 如果 3 次以内猜到该数, 则提示 “Lucky You!”; 如果超过 3 次但是在 N (>3) 次以内 (包括第 N 次) 猜到该数, 则提示 “Good Guess!”; 如果超过 N 次都没有猜到, 则提示 “Game Over”, 并结束程序。如果在到达 N 次之前, 用户输入了一个负数, 也输出 “Game Over”, 并结束程序。试编写相应程序。

解答: 参见练习 4-6。

(2) 兔子繁衍问题: 一对兔子, 从出生后第 3 个月起每个月都生一对兔子。小兔子长到第 3 个月后每个月又生一对兔子。假如兔子都不死, 请问第 1 个月出生的一对兔子, 至少需要繁衍到第几个月时兔子总数才可以达到 N 对? 试编写相应程序。

解答: 参见习题程序设计第 11 题。

(3) 高空坠球: 皮球从 height 米的高度自由落下, 触地后反弹到原高度的一半, 再落下, 再反弹, 如此反复。皮球在第 n 次落地时, 在空中经过的路程是多少米? 第 n 次反弹的高度是多少? 试编写相应程序, 输出保留 1 位小数。

提醒: 教材上此题的描述为 “皮球在第 n 次反弹落地时, 在空中经过的路程是多少米?”。

下次印刷时将修改一致。

输入输出示例改为:

10.2

20.0 2.5

解答：参见习题程序设计第 8 题。

(4) 黑洞数：黑洞数也称为陷阱数，又称“Kaprekar 问题”，是一类具有奇特转换特性的数。任何一个数字不全相同的三位数，经有限次“重排求差”操作，总会得到 495。最后所得的 495 即为三位黑洞数。所谓“重排求差”操作即组成该数的数字重排后的最大数减去重排后的最小数。(6174 为四位黑洞数)

例如，对三位数 207：

第 1 次重排求差得： $720 - 027 = 693$ ；

第 2 次重排求差得： $963 - 369 = 594$ ；

第 3 次重排求差得： $954 - 459 = 495$ ；

以后会停留在 495 这一黑洞数。如果三位数的 3 个数字全相同，一次转换后即为 0。

试求出任意输入三位数重排求差的过程。

解答：

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

```
int main (void)
```

```
{
```

```
    int a, b, c, i, max, min, number, t;
```

```
    scanf("%d", &number);
```

```
    i = 1;
```

```
    do {
```

```
        a = number / 100;
```

```
        b = (number / 10) % 10;
```

```
        c = number % 10;
```

```
        if (a < b) {
```

```
            t = a; a = b; b = t;
```

```
        }
```

```
        if (a < c) {
```

```
            t = a; a = c; c = t;
```

```
        }
```

```
        if (b < c) {
```

```
            t = b; b = c; c = t;
```

```
        }
```

```
        max = a * 100 + b * 10 + c;
```

```
        min = c * 100 + b * 10 + a;
```

```
        number = max - min;
```

```
        printf ("%d: %d - %d = %d\n", i, max, min, number);
```

```
        i++;
```

```
    } while (number != 495);
```

```
    return 0;
```

```
}
```

4.4.2 嵌套循环

一、调试示例

求 e ：输入一个正整数 n ，计算下式的和（保留 4 位小数），要求使用嵌套循环。

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

解答：参见练习 4-6。

提醒：PTA 题目集中，要求输出结果保留 8 位小数。

二、基础编程错

(1) 用两种方法求 e ：输入一个正整数 n ，用两种方法分别计算下式的和（保留 4 位小数）。

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

① 使用一重循环，不使用自定义函数。

② 定义和调用函数 `fact(n)` 计算 n 的阶乘。

提醒：求 e 可以采用 3 种方法编程，即一重循环（源程序 1）、使用函数（源程序 2）和嵌套循环（调试示例），在 PTA 题目集中，只体现为 1 道题目，不再区分实现途径。

解答：

源程序 1

```
#include <stdio.h>
int main(void)
{
    int i, n;
    double e, product;

    scanf("%d", &n);
    e = 1;
    product = 1;
    for (i = 1; i <= n; i++){
        product = product * i;
        e = e + 1.0 / product;
    }
    printf("%.8f\n", e);

    return 0;
}
```

(2)

```
#include <stdio.h>
double fact (int n);
int main(void)
{
```

```

int i, n;
double e;

scanf("%d", &n);
e = 1;
for (i = 1; i <= n ; i++){
    e = e + 1.0 / fact (i);
}
printf("%.8f\n", e);

return 0;
}
double fact (int n)
{
    double product;
    int i;

    product = 1;
    for (i = 1; i <= n; i++){
        product = product * i;
    }

    return product;
}

```

(2) 验证哥德巴赫猜想：任何一个大于 2 的偶数均可表示为两个素数之和。例如 $4=2+2$ ， $6=3+3$ ， $8=3+5$ ， \dots ， $18=5+13$ 。要求将输入的一个偶数表示成两个素数之和。

提醒：教材上此题对哥德巴赫猜想的描述为“任何一个大于等于 6 的偶数均可表示为两个素数之和”。下次印刷时将修改一致。

分析：(1) 在一行中按照格式 “ $N = p + q$ ” 输出 N 的素数分解，其中 $p \leq q$ 均为素数。又因这样的分解不唯一（例如 24 还可以分解为 $7+17$ ），要求必须输出所有解中 p 最小的解。

(2) 定义变量 flag 标识素数，flag 为 1: p 和 q 皆为素数；flag 为 0: p 或 q 不是素数。

解答：

```

#include <stdio.h>
#include <math.h>
int main (void)
{
    int flag, i, j, k, p, q, m, n;
    scanf ("%d", &n);
    if (n == 4) {
        printf("4 = 2 + 2\n");
    }
    else{
        for (p = 3; p <= n/2; p = p + 2) {

```

```

        flag = 1;
        m = (int)sqrt(p);
        for (k = 2; k <= m; k++) {
            if (p % k == 0) {
                flag = 0;
                break;
            }
        }
        if (flag == 1) {
            q = n - p;
            m = (int)sqrt(q);
            for (k = 2; k <= m; k++) {
                if (q % k == 0) {
                    flag = 0;
                    break;
                }
            }
        }
        if (flag == 1){
            printf ("%d = %d + %d", n, p, q);
            break;
        }
    }

    return 0;
}

```

(3) 换硬币：将一笔零钱（大于 8 分，小于 1 元，精确到分）换算成 1 分、2 分和 5 分的硬币组合。输入金额，输出共有多少种换法？要求硬币面值按 5 分、2 分、1 分顺序，各类硬币数量依次从大到小的顺序，输出各种换法。

解答：参见习题程序设计第 5 题。

(4) 水仙花数：输入两个正整数 m 和 n ($m \geq 1, n \leq 1000$)，输出 m 到 n 之间的所有水仙花数。水仙花数是指各位数字的立方和等于其自身的数。例如 153 的各位数字的立方和是 $1^3 + 5^3 + 3^3 = 153$ 。试编写相应程序。

提醒：此题与 PTA 题目集的题目描述有所不同，PTA 题目集中题目的描述及解答参见习题程序设计第 6 题。

解答：

```

#include <stdio.h>

int main (void)
{
    int digit, i, m, n, number, sum;

```

```

printf ("Input m:");
scanf ("%d", &m);
printf ("Input n:");
scanf ("%d", &n);
for (i = m; i <= n; i++){
    number = i;
    sum = 0;
    while (number != 0){
        digit = number % 10;
        number = number / 10;
        sum = sum + digit * digit * digit;
    }
    if (sum == i) {
        printf("%d\n", i);
    }
}

return 0;
}

```

(5) 输出三角形字符阵列图形：输入一个正整数 n ($n < 7$)，输出 n 行由大写字母 A 开始构成的三角形字符阵列图形。

解答：

```

#include <stdio.h>
int main (void)
{
    int i, j, n;
    char ch;

    ch = 'A';
    scanf ("%d", &n);
    for (i = n; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
            printf ("%c ", ch);
            ch++;
        }
        printf("\n");
    }

    return 0;
}

```

三、改错题

找完数：找出 200 以内的所有完数，并输出其因子。一个数若恰好等于它的各因子之和，

即称其为完数，例如， $6 = 1 + 2 + 3$ ，其中 1、2、3 为因子，6 为因子和。

提醒：此题与 PTA 题目集的题目略有不同，PTA 题目需要输入查找完数的范围。

解答（PTA）：

```
#include <stdio.h>
#include <math.h>
int main()
{
    int flag, i, j, m, n, s;

    scanf ("%d %d", &m, &n);
    flag = 0;
    if (m == 1) {
        printf ("1 = 1\n");
        m++;
        flag = 1;
    }
    for (i = m; i <= n; i++) {
        s = 0;
        for (j = 1; j <= i / 2; j++) {
            if (i % j == 0) {
                s = s + j;
            }
        }
        if (i == s) {
            flag = 1;
            printf ("%d = 1", i);
            for (j = 2; j <= i / 2; j++){
                if (i % j == 0) {
                    printf (" + %d", j);
                }
            }
            printf("\n");
        }
    }
    if (flag == 0) {
        printf("None\n");
    }

    return 0;
}
```

四、拓展编程题

(1) 从高位开始逐位输出一个整数的各位数字：输入一个整数，从高位开始逐位分割并输出它的各位数字。

解答:

```
#include <stdio.h>
int main (void)
{
    int digit, n, temp, pow;
    scanf ("%d", &n);
    if (n < 0) {
        n = - n;
    }
    temp = n;
    pow = 1;
    while (temp > 10) {
        pow = pow * 10;
        temp = temp / 10;
    }
    while ( pow >= 1 ) {
        digit = n / pow;
        n = n % pow;
        pow = pow / 10;
        printf ("%d ", digit);
    }
    printf ("\n");

    return 0;
}
```

(2) 梅森数: 形如 $2^n - 1$ 的素数称为梅森数 (Mersenne Number)。例如 $2^2 - 1 = 3$ 、 $2^3 - 1 = 7$ 都是梅森数。1722 年, 双目失明的瑞士数学大师欧拉证明了 $2^{31} - 1 = 2147483647$ 是一个素数, 堪称当时世界上“已知最大素数”的一个记录。输入一个正整数 $n(n < 20)$, 编程输出所有不超过 $2^n - 1$ 的梅森数。

解答:

```
#include <stdio.h>
#include <math.h>
int main (void)
{
    int flag, i, j, k, n, number, prime;

    scanf ("%d", &n);
    flag = 0;
    for (i = 2; i <= n; i++) {
        number = (int) pow (2, i) - 1;
        k = (int) sqrt (number);
        prime = 1;
        for (j = 2; j <= k; j++) {
```

```
        if (number % j == 0) {
            prime = 0;
            break;
        }
    }
    if (prime != 0) {
        printf ("%d\n", number);
        flag = 1;
    }
}
if (flag == 0) {
    printf("None\n");
}

return 0;
}
```