# 2 流程控制实验

## 2.1 实验目的

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容

**2.2.1 程序改错**

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF) {
7. for(i=2,k=x>>1;i<=k;i++)
8. if (!x%i) {
9. flag = 1;
10. break;
11. }
12. if(flag=1) printf("%d是合数", x);
13. else printf("%d不是合数", x);
14. }
15. return 0;
16. }

**解答：**

（1）语法错误修改：

1) 第8行的if (!x%i)，运算优先级错误，正确形式为：

if (!(x%i) )

2) 第12行的if(flag=1) printf("%d是合数", x);，判定等于写成了赋值符号，正确形式为：

if(flag==1) printf("%d是合数", x);

（2）逻辑错误修改：

1）一次循环后，没有给flag赋初值，应该在while循环开始时添加语句：

flag = 0 ;

（3）修改后，源程序清单如下：

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ; // 逻辑错误
9. for(i=2,k=x>>1;i<=k;i++)
10. if (!(x%i) )
11. { // 疑似语法错误
12. flag = 1;
13. break;
14. }
15. if(flag==1) printf("%d是合数\n", x); // 第三处
16. else printf("%d不是合数\n", x);
17. }
18. return 0;
19. }

（4）测试

(a)测试用例

选择的测试用例如表2-1所示：

表2-1 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 2 | 2不是合数 |
| 用例2 | 3 | 3不是合数 |
| 用例3 | 10 | 10是合数 |
| 用例4 | ^Z | 结束 |

(b)对应表2-1测试用例的运行结果如图2-1所示：

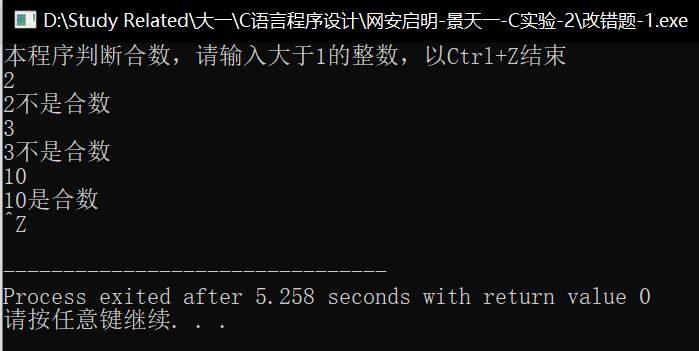


图2-1 实验2-1修改后运行结果截图

**2.2.2 程序修改替换**

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

（2）修改实验2-1程序，将for循环改用do-while循环。

（3）修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

**解答：**

1. 修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句，按照要求修改后的程序如下：

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ;
9. for(i=2,k=x>>1;(i<=k)&&(flag != 1);i++)
10. if (!(x%i) )
11. {
12. flag = 1;
13. }
14. if(flag==1) printf("%d是合数", x);
15. else printf("%d不是合数", x);
16. }
17. return 0;
18. }
19. 修改实验2-1程序，将for循环改用do-while循环。按照要求修改后的程序如下：

/\* 实验2-1改错题程序：合数判断器\*/

1. #include <stdio.h>
2. int main( )
3. {
4. int i, x, k, flag = 0;
5. printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");
6. while (scanf("%d", &x) !=EOF)
7. {
8. flag = 0 ;
9. i = 2 ;
10. k = x>>1 ;
11. do
12. {
13. if (!(x%i) && (x != 2 ) && (x!=0))
14. {
15. flag = 1;
16. }
17. i++ ;
18. }while((i<=k)&&(flag != 1));
20. if(flag==1) printf("%d是合数\n", x);
21. else printf("%d不是合数\n", x);
22. }
23. return 0;
24. }
25. 修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。
26. 解题思路：通过for 循环遍历所有3位数，调用前两题的素数判断函数进行判断，通过对输入数据除以10和100获取第二位，第三位，若判断后均为合数，则输出该数据；遍历所有3位数后结束循环。
27. 修改后程序的源代码如下：
28. #include<stdio.h>
29. int composite(int x)
30. {
31. int i, k, flag = 0;
32. for(i=2,k=x>>1;i<=k;i++)
33. if (!(x%i) )
34. {
35. flag = 1;
36. break;
37. }
38. return flag ;
39. }
40. int main(void)
41. {
42. for (int init=100;init < 1000;init++)
43. {
44. if ((composite(init)==1)&&(composite(init/10)==1)&&(composite(init/100)==1))printf("%d ",init);
45. }
46. return 0 ;
47. }
48. 优化和替换后运行结果如图2-2所示。

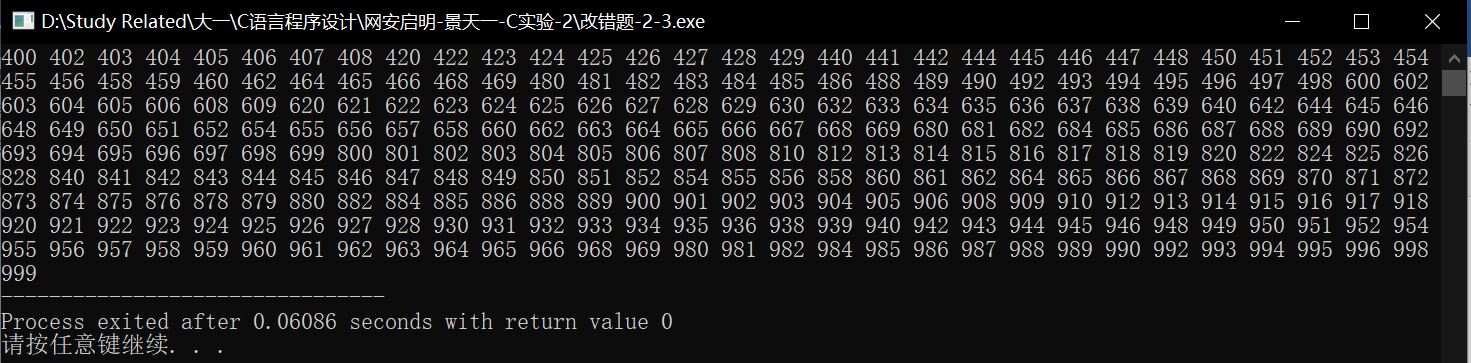
****

图2-2 实验2-2优化替换后运行结果截图

**2.2.3 程序设计**

（1）假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；x＞5000，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

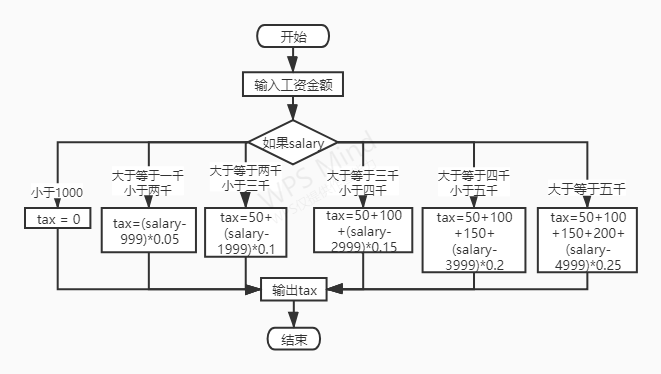
1. 算法流程
2. 程序设计题1 if语句版流程图如图2-3所示：

图2-3 程序设计题1程序流程图

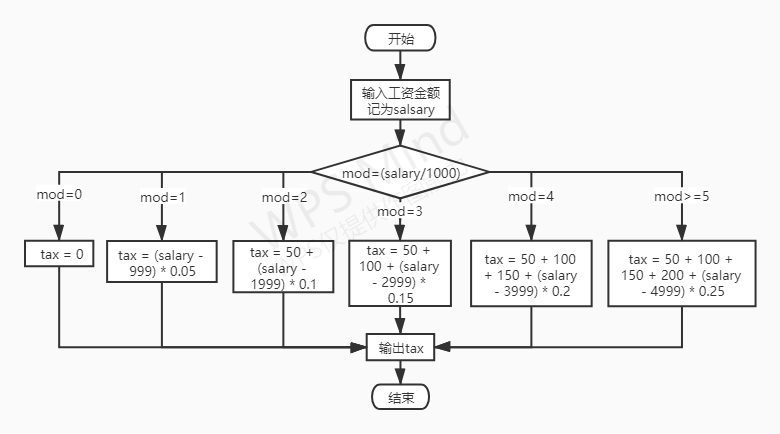
1. 程序设计题1 switch语句版流程图如图2-4所示：

图2-4 程序设计题1程序流程图

2) 源程序清单

1. 程序设计题1 if语句版源程序如下所示：
2. #include<stdio.h>
3. int main(void)
4. {
5. float salary = 0 ;
6. float tax = 0;
7. printf("请输入工资金额：") ;
8. scanf("%f",&salary) ;
10. if (salary < 1000)
11. {
12. tax = 0 ;
13. }
14. else if (salary >= 1000 && salary < 2000)
15. {
16. tax = (salary - 999) \* 0.05 ;
17. }
18. else if (salary >= 2000 && salary < 3000)
19. {
20. tax = 50 + (salary - 1999) \* 0.1 ;
21. }
22. else if (salary >= 3000 && salary < 4000)
23. {
24. tax = 50 + 100 + (salary - 2999) \* 0.15 ;
25. }
26. else if (salary >= 4000 && salary < 5000)
27. {
28. tax = 50 + 100 + 150 + (salary - 3999) \* 0.2 ;
29. }
30. else if (salary >= 5000)
31. {
32. tax = 50 + 100 + 150 + 200 + (salary - 4999) \* 0.25 ;
33. }
35. printf("税金额度为：%.2f\n",tax) ;
36. return 0 ;
37. }
38. 程序设计题1 switch语句版源程序如下所示：
39. #include<stdio.h>
40. int main(void)
41. {
42. float salary = 0 ; // 声明变量
43. float tax = 0;
44. printf("请输入工资金额：") ; // 输入工资金额
45. scanf("%f",&salary) ;
46. int mod=0 ;
47. mod = (salary / 1000) ; // 通过区分除数来判断大小
48. switch(mod)
49. {
50. case 0 : tax = 0 ;break ;
51. case 1 : tax = (salary - 999) \* 0.05 ;break ;
52. case 2 : tax = 50 + (salary - 1999) \* 0.1 ;break ;
53. case 3 : tax = 50 + 100 + (salary - 2999) \* 0.15 ;break ;
54. case 4 : tax = 50 + 100 + 150 + (salary - 3999) \* 0.2 ;break ;
55. default: tax = 50 + 100 + 150 + 200 + (salary - 4999) \* 0.25 ;
56. }
58. printf("税金额度为：%.2f",tax) ;
59. return 0 ;
60. }

3）测试

（a） 测试数据：

选择的测试用例如表2-2所示：

表2-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 900 | 0 |
| 用例2 | 4500 | 400.2 |
| 用例3 | 6000 | 750.25 |

（b） 对应测试数据的运行结果截图

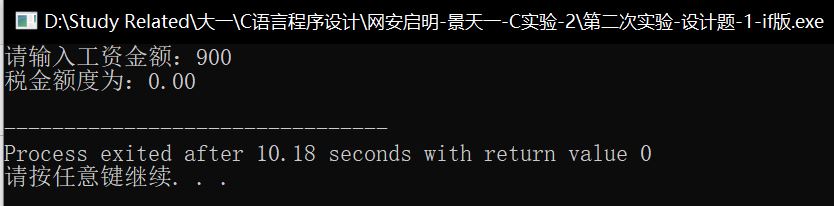
i) 对应测试测试用例1的运行结果如图2-5所示。

图2-5 程序设计题1的测试用例一的运行结果

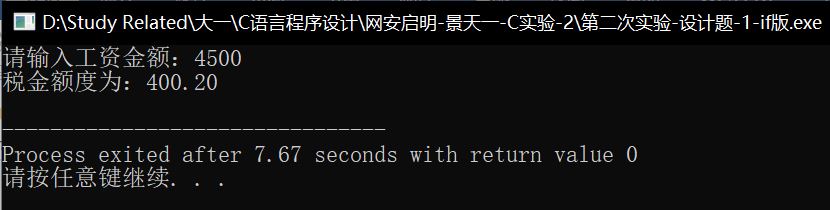
对应测试测试用例2的运行结果如图2-6所示。

图2-6 程序设计题1的测试用例二的运行结果

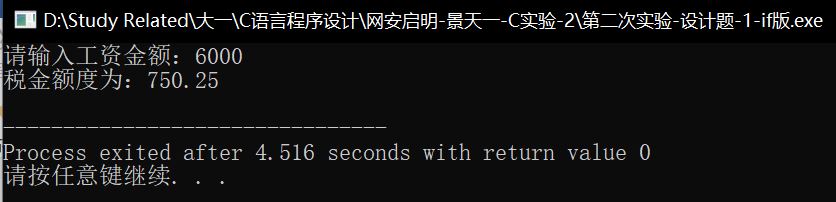
对应测试测试用例3的运行结果如图2-7所示。

图2-7 程序设计题1的测试用例三的运行结果

（2）将输入的正文复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

**解答：**

1) 解答思路：

程序设计题1 switch语句版流程图如图2-8所示：

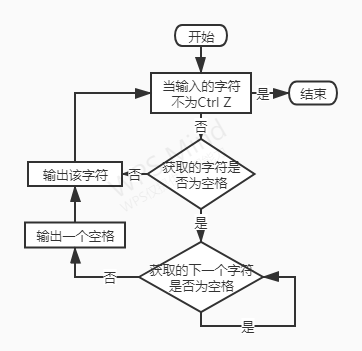


图2-8程序设计题2程序流程图

2)程序清单

1. #include<stdio.h>
2. int main(void)
3. {
4. char ch ; // 声明字符ch；
5. while (scanf("%c",&ch) != EOF) // 设定停止条件：输入ctrl z;
6. {
7. if (ch == ' ') // 若获得的字符为空格，则继续读取直至获取的字符部位空格为止；
8. {
9. while(ch == ' ')
10. {
11. scanf("%c",&ch);
12. }
13. putchar(' '); // 输出一个空格；
14. putchar(ch) ;
15. }
16. else
17. {
18. putchar(ch); // 若获得的字符不为空格，则直接打印字符；
19. }
20. }
21. return 0 ;
22. }

3）测试

（a） 测试数据：

测试数据如表2-2所示。

表2-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | □□□abc□□□12 | □abc□12 |
| 用例2 | □□□□□□□a | □a |
| 用例3 | 1□□□2□□□3 | 1□2□3 |
| 用例4 | ^Z | 结束 |

（b） 对应测试测试用例1，2，3，4的运行结果如图2-9所示。

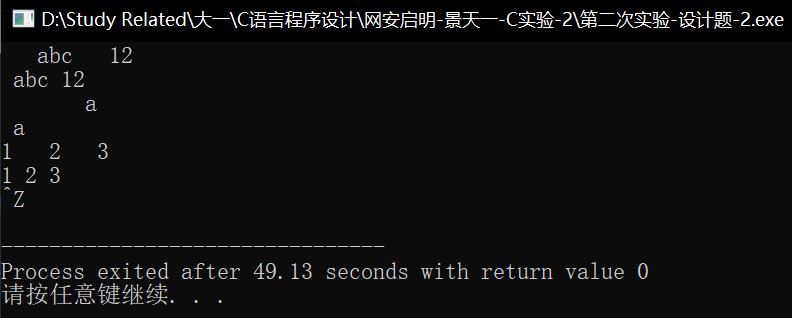


图2-9 程序设计题2的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

第i行第j列位置的数据值可以由组合表示，而的计算如下：

|  |  |
| --- | --- |
|  | (i=0,1,2,…) |
|  | (j=0,1,2,3,…,i) |

根据以上公式，采用顺推法编程，输出金字塔效果的杨辉三角形。特别要注意空格的数目，一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格。

**解答：**

1) 解答思路：

程序设计题3流程图如图2-10所示：

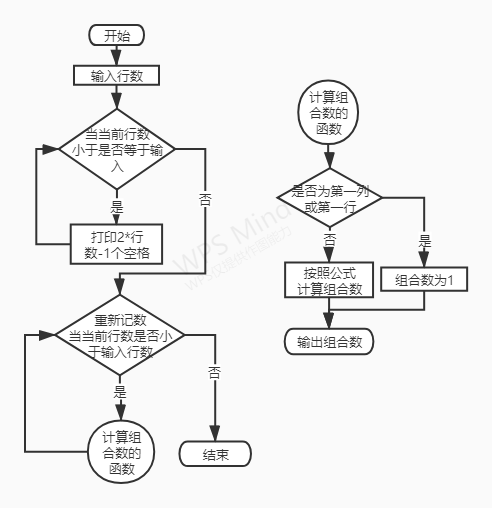


图2-10 程序设计题3程序流程图

1. 源程序清单
2. #include<stdio.h>
3. int factorial(int num) //计算阶乘的函数
4. {
5. int sum =1 ;
6. while(num !=1) // 封装来节省行数
7. {
8. sum = sum \* num ;
9. num-- ;
10. }
11. return sum ;
12. }
13. int number(int i, int j) // 输出第j行第i列的数字
14. {
15. int sum = 1 ;
16. i-- ;
17. j-- ;
18. if (i == 0 || i == j) //若计算第一列或最后一列的数字，则直接输出1
19. sum = 1 ;
20. else
21. {
22. sum = (factorial(j)/(factorial(i)\*factorial(j-i))) ; //调用阶乘函数
23. }
24. return sum ;
25. }
26. int main(void)
27. {
28. int line=0, j=1 ; // line表示最大行数，j表示当前行数
29. printf("请输入行数：") ;
30. scanf("%d",&line) ;
31. line++ ;
32. while (j <= line) // 当前行数达到输出最大行数时停止
33. {
34. // 首先打印左侧空格
35. int count=1 ;
36. while (count < (2\*(line - j + 1) - 1)) // 每行前空格为行数乘2减1
37. {
38. printf(" ") ;
39. count++ ;
40. }
41. // 依次打印数字
42. count = 1 ; // count表示当前列数 j表示当前列数
43. while (count <= j)
44. {
45. int num ;
46. num = number(count,j) ;
47. printf("%-4d",num) ;
48. if (count == j)
49. printf("\n") ;
50. count++ ;
51. }
53. j++ ;
54. }
56. return 0 ;
57. }
58. 测试
59. 测试数据

测试数据如表2-3所示：

表2-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试数据 | 预计输出 |
| 用例1 | 5 |  |
| 用例2 | 9 |  |
| 用例3 | 10 |  |

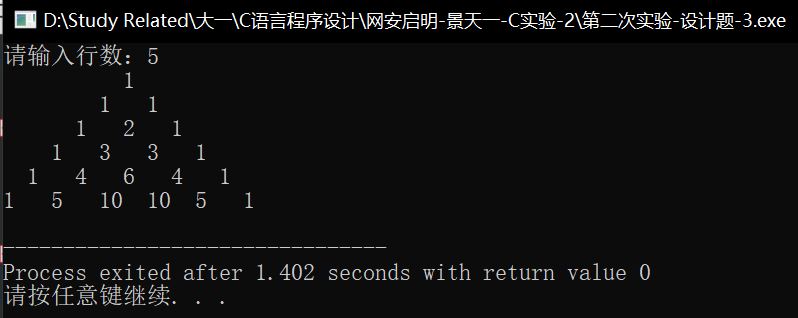
1. 对应测试测试用例1的运行结果如图2-11所示。

图2-11 程序设计题3的测试用例1的运行结果

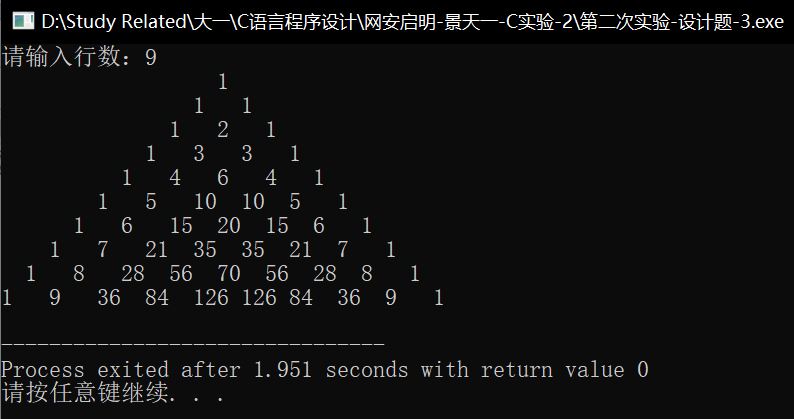
对应测试测试用例2的运行结果如图2-12所示。

图2-12 程序设计题3的测试用例2的运行结果

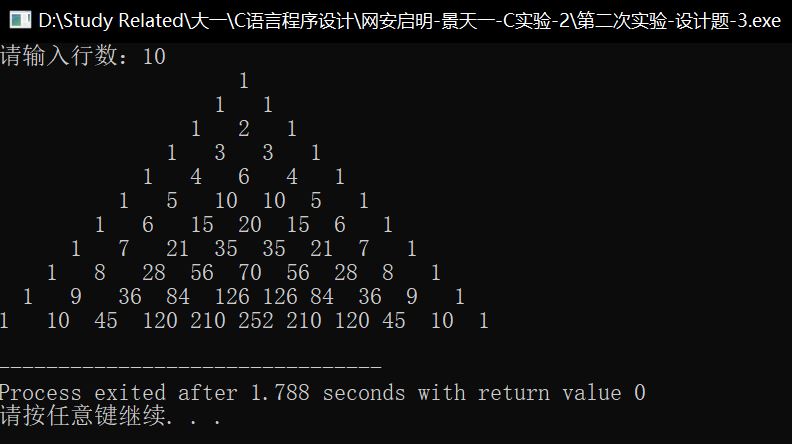
对应测试测试用例3的运行结果如图2-13所示。

图2-13 程序设计题3的测试用例3的运行结果

1. 625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。

**解答：**

* + 1. 解答思路：

程序设计题4流程图如图2-14所示：

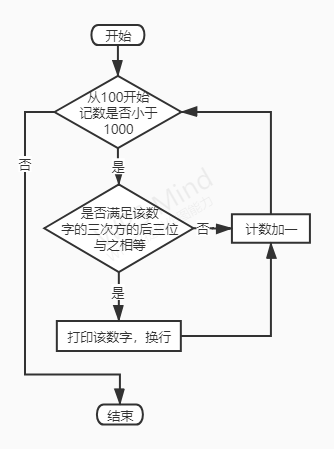


图2-14 程序设计题4程序流程图

1. 源程序清单
2. #include <stdio.h>
3. int main(void)
4. {
5. for (int num = 100;num < 999;num++) // 遍历所有三位数
6. {
7. if ((num == ((num \* num)%1000)) && (num <= 999)) // 题目条件
8. {
9. printf("%d\n", num) ;
10. }
11. }
12. return 0 ;
13. }

3) 测试

预计输出结果：376 625

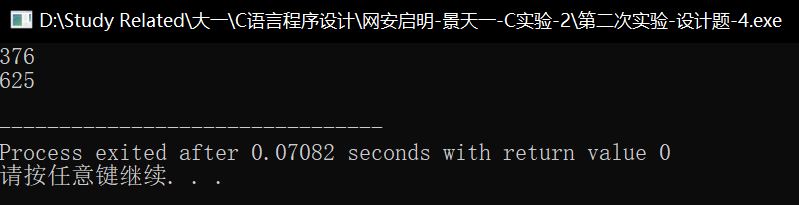
 程序设计题4的运行结果如图2-15所示：

图2-15 程序设计题4的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 2.3 实验小结

在编程设计题1中最初没有理解分段收取税金的含义，导致输出结果与预计不一致，说明每次在写程序时，最好是确保已经完全弄懂题目的目的和含义再进行操作。在编程设计题3中，因为不了解double型占位符的作用导致程序设计卡壳；在查阅资料了解了占位符和左对齐后解决了问题。

另外，在此次实验中，我切实感受到了程序设计时所要注意的一些问题，尤其是程序的可修改性，如果将主要重复的模块封装为函数，将大大节省调试步骤和编写时间。