

第五次作业实验报告

实验环境

以下所有实验都处于这一环境中

操作系统

Windows 10 家庭中文版 64位 版本10.0.17134.345

硬件

CPU Intel Core i7-8750H

RAM 8GB

IDE

Microsoft Visual Studio Community 2017 VisualStudio.15.Release/15.8.5+28010.2036

Visual C++ 2017 00369-60000-00001-AA380

第一题：三角形判断

实验目的

编写程序，输入3个整数，判断它们是否能够构成三角形，若能构成三角形，则输出三角形的类型，若为等边三角形，输出 `Equilateral triangle`，若为等腰三角形输出 `Isosceles triangle`，若为一般三角形（非等边，非等腰）输出 `Triangle`，否则输出 `Not triangle`。

要求：输入：三个整数 x 、 y 、 z ，用空格隔开。输出：判断结果。

实验内容

分析

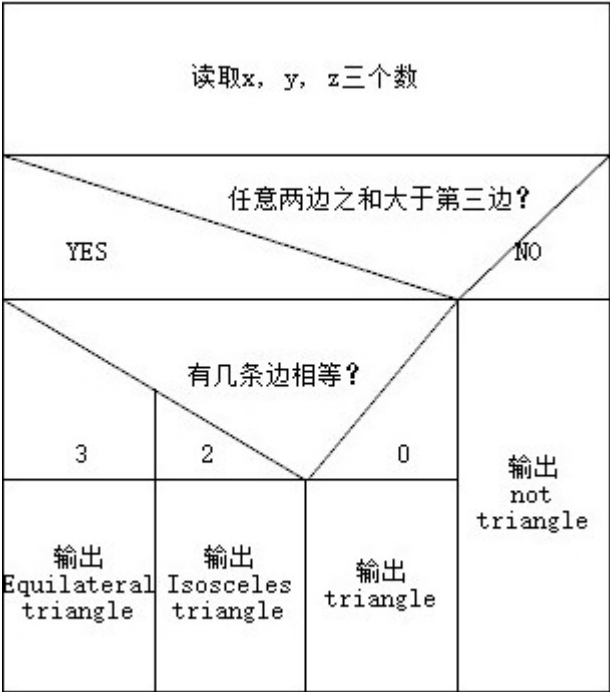
读入输入的三个数后，可以由数学知识知道三角形满足以下条件：

$$\begin{aligned} & \text{任意两边之和} > \text{第三边} \\ & \text{(或者写为)} \\ & \text{任意两边之差} < \text{第三边} \end{aligned}$$

如果不等式成立，则输入的三个数可以组成三角形；如果有一个不成立，则输入的三个数不可以组成三角形。

由等边三角形和等腰三角形的定义，只要判断输入的三个数中有几个相等即可。

所以逻辑可以写成如下N-S图



用代码来表示

第一个条件 `if(x + y > z && y + z > x && x + z > y)`

第二个条件

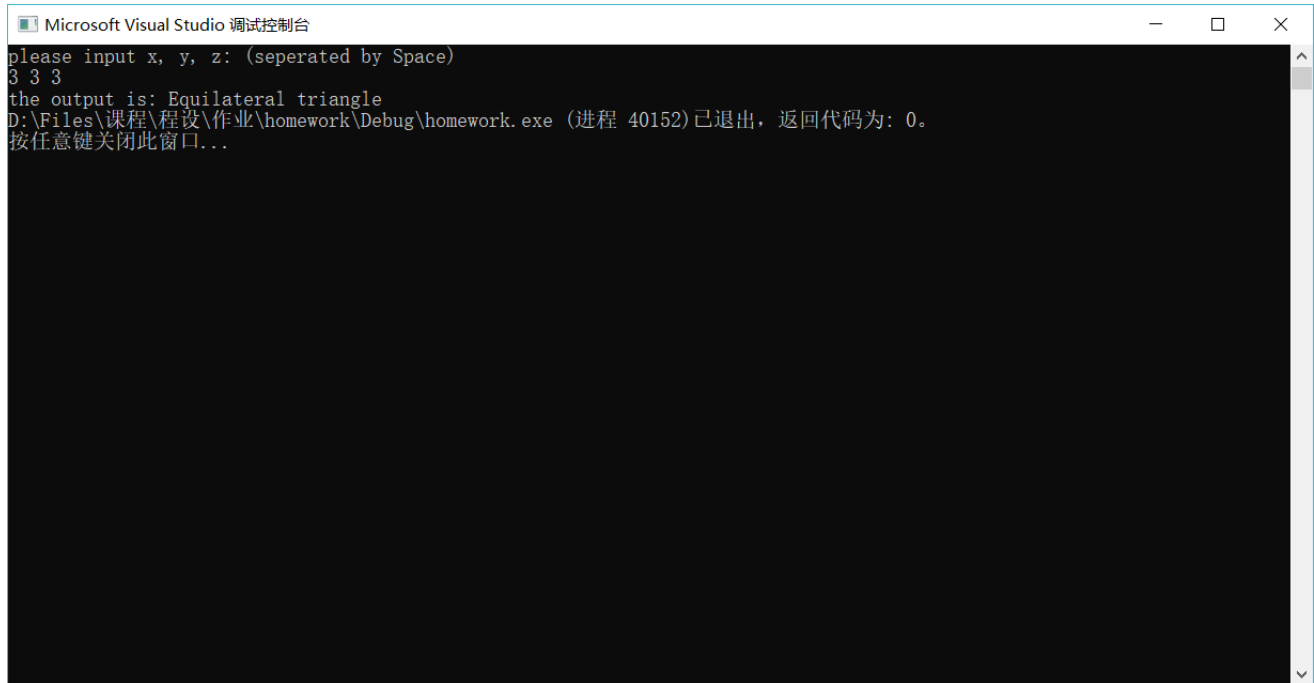
```
1 switch((x == y) + (x == z) + 1){
2     case 1://三条边互不相等
3     case 2://仅有两条边相等，等腰三角形
4     case 3://三条边都相等，等边三角形
5 }
```

代码

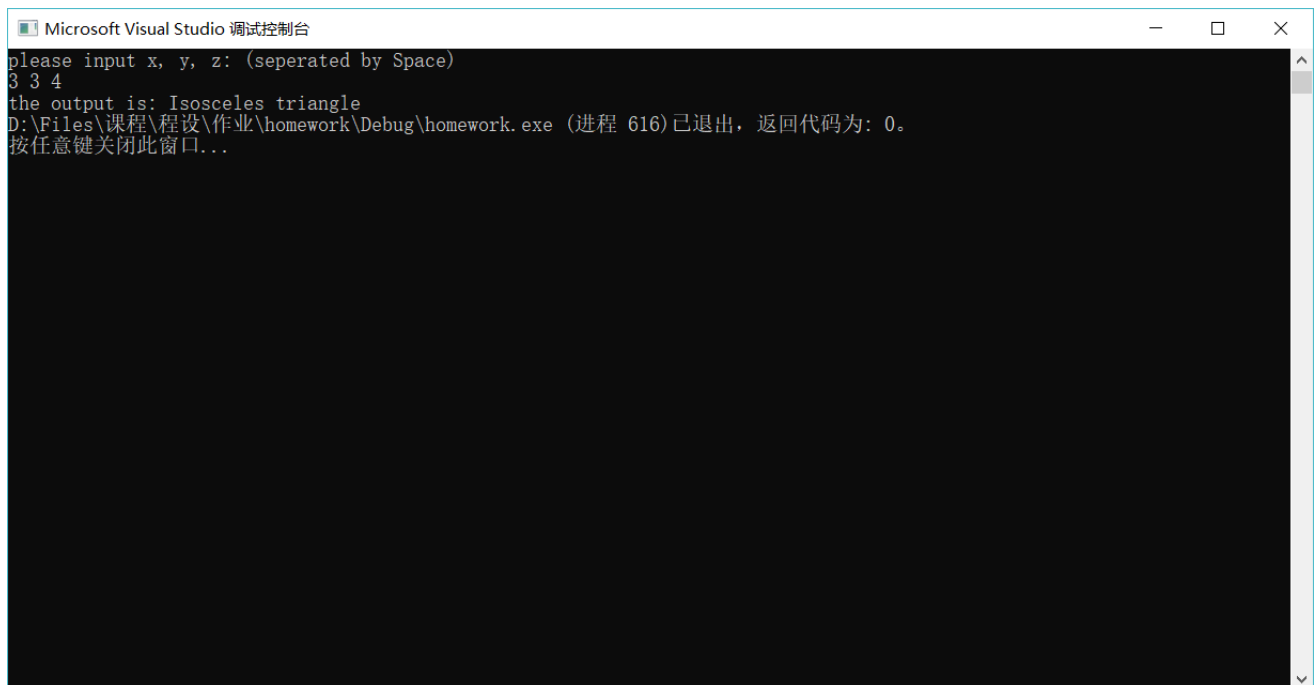
```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int x, y, z;
6     cout << "please input x, y, z: (seperated by Space)\n";
7     cin >> x >> y >> z;
8     cout << "the output is: ";
9     if ((x + y > z) && (y + z > x) && (x + z > y)){
10         switch((x == y) + (x == z) + 1){
11             case 1: cout << "Triangle"; break; //三条边互不相等
12             case 2: cout << "Isosceles triangle"; break; //仅有两条边相等，等腰三角形
13             case 3: cout << "Equilateral triangle"; break; //三条边都相等，等边三角形
14         }
15     }
16     else
```

```
17         cout << "Not triangle";  
18  
19     return 0;  
20 }
```

结果



```
Microsoft Visual Studio 调试控制台  
please input x, y, z: (seperated by Space)  
3 3 3  
the output is: Equilateral triangle  
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 40152) 已退出, 返回代码为: 0。  
按任意键关闭此窗口...
```



```
Microsoft Visual Studio 调试控制台  
please input x, y, z: (seperated by Space)  
3 3 4  
the output is: Isosceles triangle  
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 616) 已退出, 返回代码为: 0。  
按任意键关闭此窗口...
```

成功实现

实验总结体会

第一次尝试用N-S图阐明自己的思路，十分有助于理清每一步应该干什么。不过目前绘制N-S图的方法效率有点低。

此题简单利用三角形的几何性质便可以得出一个判别三个数是否能组成三角形的条件句，再通过 两条边是否相等 这个条件句的真假分别为1、0，可以利用switch语句进行分类输出结果。

值得一提的是，如果用 `scanf()` 进行数据读取的话，`%d` 之间的空格数量需要注意一下，而且不能用其他符号。

第二题：有理数计算

实验目的

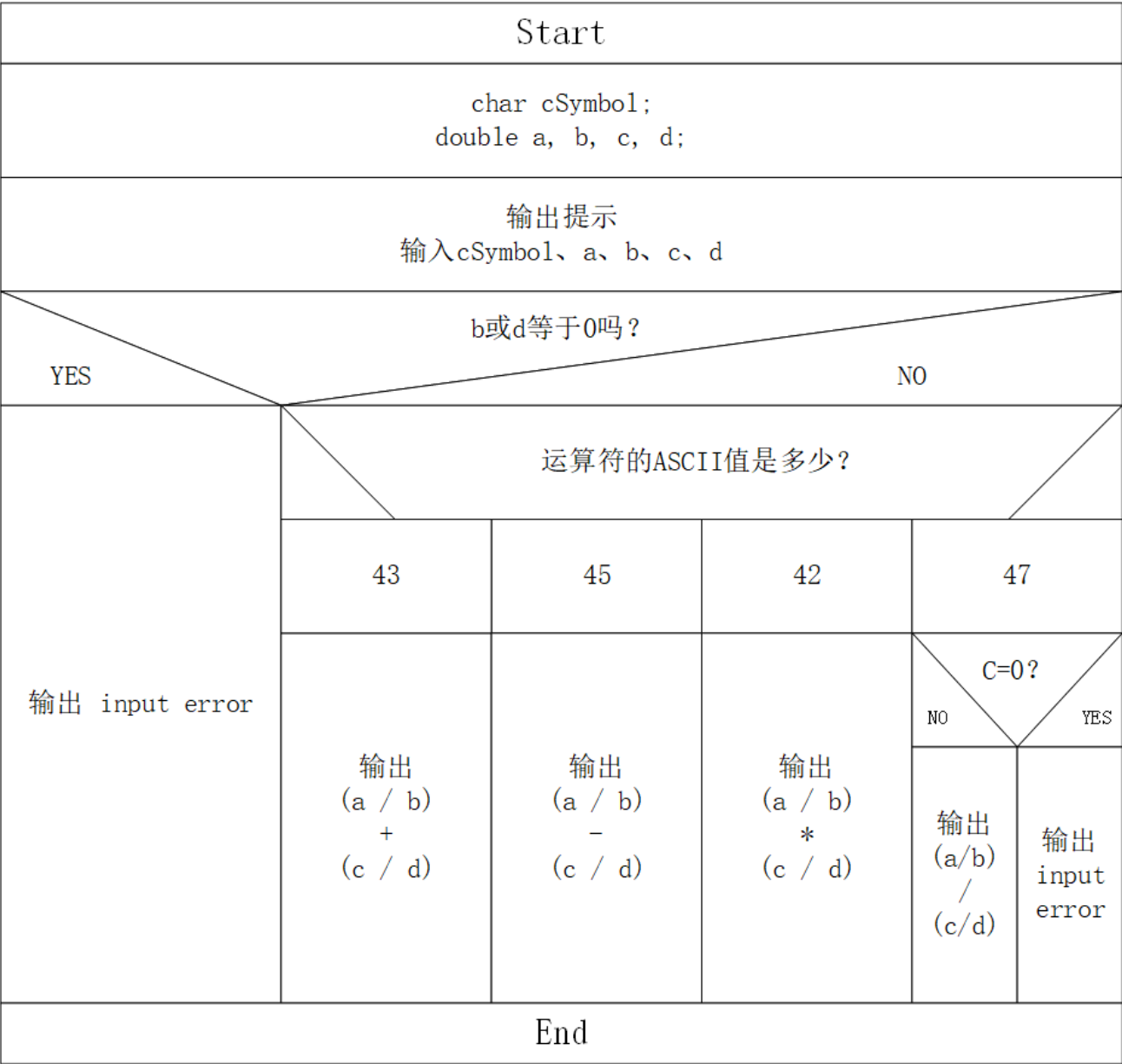
有理数计算问题：依次输入运算符@（@为+、-、*、/四种运算之一）和整数a、b、c、d，计算并输出表达式 $\frac{a}{b} @ \frac{c}{d}$ 结果的值。仅给出计算结果，不需要显示表达式。当输入的b或d出现0时，输出错误提示input error；当输入运算符为/且输入c值为0时，同样提示input error。要求：输入：运算符@和整数a、b、c、d，用空格隔开。输出：计算结果或input error。

实验内容

分析

由于需要读取运算符@，故需要一个字符变量存储运算符并根据运算符执行不同的操作。可以利用 `switch` 语句和运算符的ASCII码实现。

大致逻辑如下图所示：

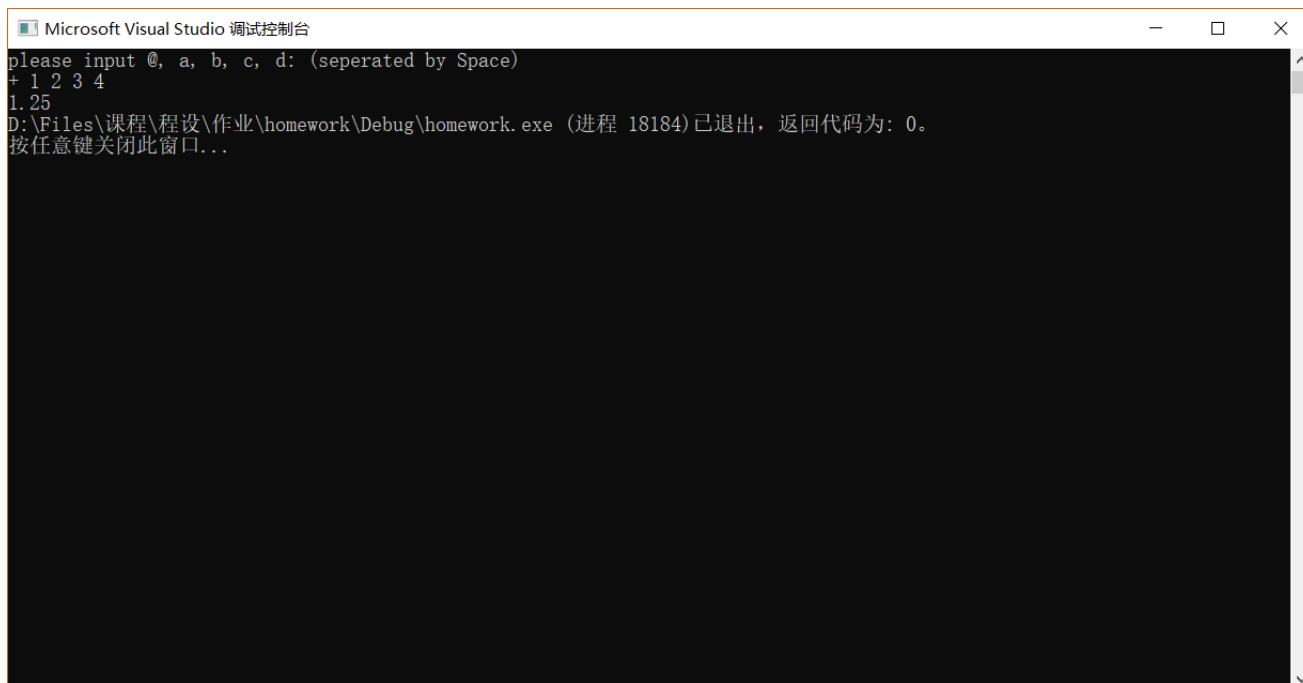


代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char cSymbol;
6      double a, b, c, d;
7      cout << "please input @, a, b, c, d: (seperated by Space)\n";
8      cin >> cSymbol >> a >> b >> c >> d;
9      if ((b == 0) || (d == 0))cout << "input error";
10     switch (cSymbol) {
11         case 43: cout << (a / b) + (c / d); break;
12         case 45: cout << (a / b) - (c / d); break;
13         case 42: cout << (a / b) * (c / d); break;
14         case 47: {
15             if (c == 0) {
16                 cout << "input error";
```

```
17         break;
18     }
19     else {
20         cout << (a / b) / (c / d);
21         break;
22     }
23 }
24 default: cout << "input error";
25 }
26
27 return 0;
28 }
```

结果



```
Microsoft Visual Studio 调试控制台
please input @, a, b, c, d: (seperated by Space)
+ 1 2 3 4
1.25
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 18184) 已退出，返回代码为：0。
按任意键关闭此窗口...
```

实验总结体会

在这道题中，`switch` 语句可以有效利用运算符的ASCII码实现分类输出，简化了条件判断的步骤。

虽然题目说明中a、b、c、d四个数字为整数（整型），但是由于输出的结果应该为实型，所以直接在声明变量时用double类型可以避免之后忘记变量类型转换的步骤。

第三题：数字分离

实验目的

给出一个不多于5位的正整数。要求：

- 求出它是几位数。
- 分别打印出每一位数字。
- 按照逆向打印各位数字。例如，原数为321应输出123。

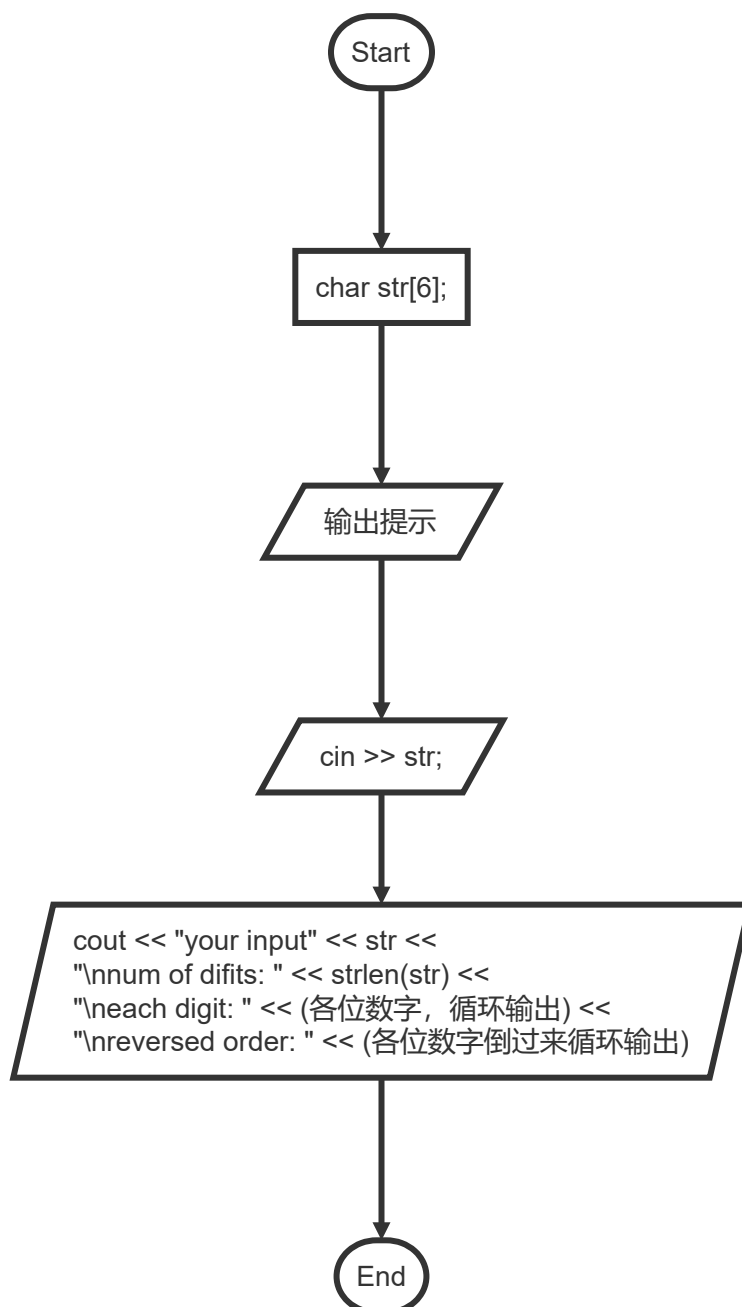
- 最少设计10个不不同数据来测试运行行行结果。

实验内容

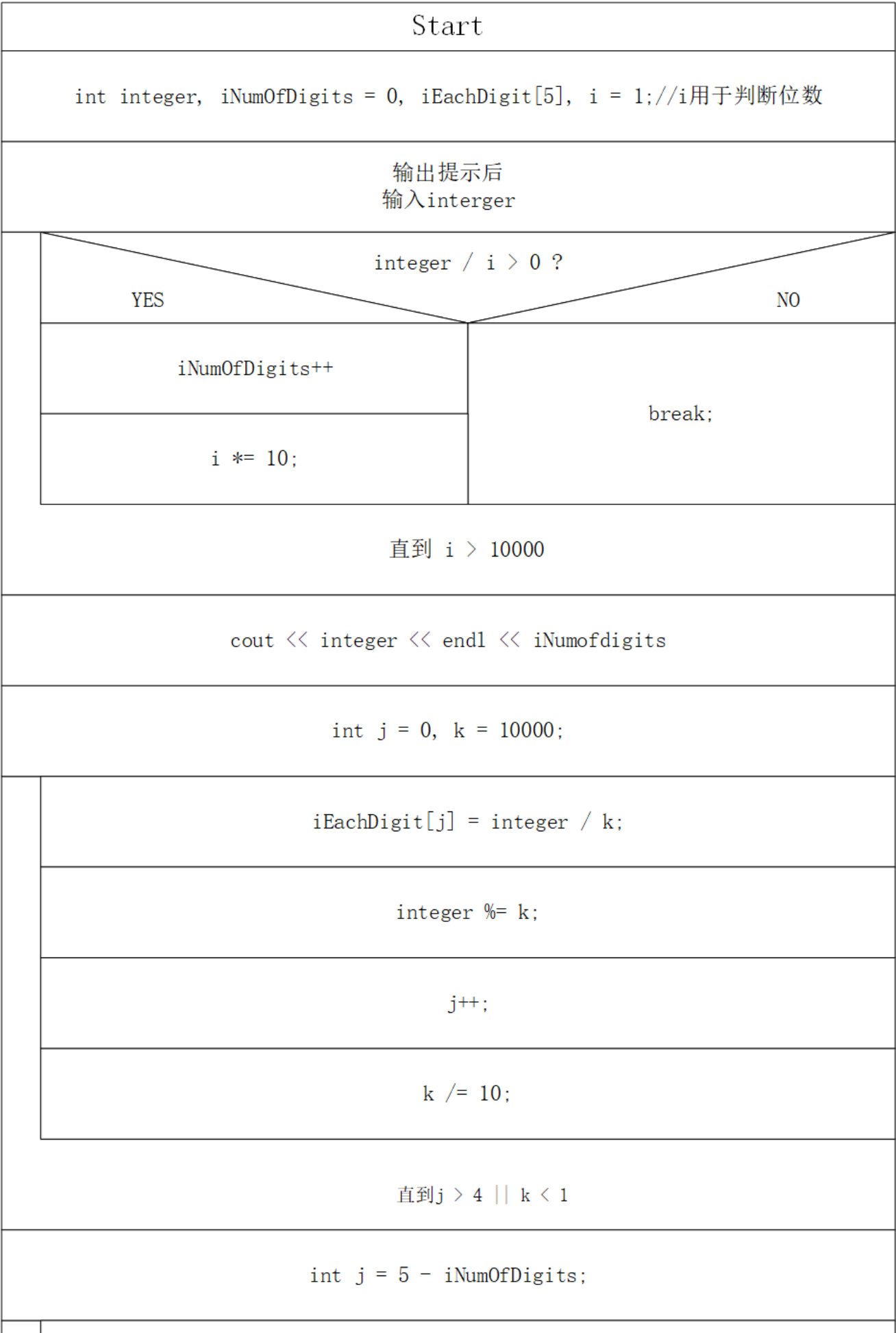
分析

考虑到要求出输入数字的位数，可以采用整型数逐步对10000、1000、100、10进行除法并验证结果是否小于0得出，也可以用 `string` 类型进行存储后通过调用函数得出长度，还可以用字符数组进行存储后用 `strlen()` 获取其长度。

由于需要分别打印每一位数字以及逆向打印各位数字，采用 `string` 类型或者字符数组相比整型变量更为方便，只需采用如下流程图的操作即可



而对于用整型数进行存储和读取，可以通过设定多个变量分别存储个十百千万位上的数字，再分别调用。逻辑如下图：



	cout << iEachDigit[j]
	j++;
	直到j > 4
	int j = 4;
	cout << iEachDigit[j]
	j--;
	直到j < 5 - iNumOfDigits
	End

代码

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int integer, iNumOfDigits = 0, iEachDigit[5], i = 1;
6      cout << "please input an integer:\n";
7      cin >> integer;
8      for (i; i < 10001; i++) {
9          if (integer / i > 0) {
10             iNumOfDigits++;
11             i *= 10;
12         }
13         else break;
14     }
15     cout << "\nyour input: " << integer;
16     cout << ",\nnum of difits: " << iNumOfDigits;
17     for (int j = 0, k = 10000; j < 5, k > 0; j++) {
18         iEachDigit[j] = integer / k;
19         integer %= k;
20         k /= 10;
21     }
22     cout << "\neach digit: ";
23     for (int j = 5 - iNumOfDigits; j < 5; j++) {

```

```
24     cout << iEachDigit[j] << ", ";
25 }
26 cout << "\nreversed order: ";
27 for (int j = 4; j >= 5 - iNumOfDigits; j--) {
28     cout << iEachDigit[j] << ", ";
29 }
30
31 return 0;
32 }
```

结果

```
D:\Files\课程\...  -  □  ×

please input an integer:
12345

your input: 12345,
num of difits: 4
each digit: 2, 3, 4, 5,
reversed order: 5, 4, 3, 2,
please input an integer:
1234

your input: 1234,
num of difits: 4
each digit: 1, 2, 3, 4,
reversed order: 4, 3, 2, 1,
please input an integer:
124

your input: 124,
num of difits: 3
each digit: 1, 2, 4,
reversed order: 4, 2, 1,
please input an integer:
12

your input: 12,
num of difits: 2
each digit: 1, 2,
reversed order: 2, 1,
please input an integer:
1

your input: 1,
num of difits: 1
each digit: 1,
reversed order: 1,
please input an integer:
```

```
D:\Files\课程\...  —  □  ×

98765

your input: 98765,
num of difits: 4
each digit: 8, 7, 6, 5,
reversed order: 5, 6, 7, 8,
please input an integer:
8765

your input: 8765,
num of difits: 4
each digit: 8, 7, 6, 5,
reversed order: 5, 6, 7, 8,
please input an integer:
765

your input: 765,
num of difits: 3
each digit: 7, 6, 5,
reversed order: 5, 6, 7,
please input an integer:
65

your input: 65,
num of difits: 2
each digit: 6, 5,
reversed order: 5, 6,
please input an integer:
4538

your input: 4538,
num of difits: 4
each digit: 4, 5, 3, 8,
reversed order: 8, 3, 5, 4,
please input an integer:
```

实验总结体会

本题主要是锻炼学习者利用除法和求余运算得出输入数字的位数，并进行相应处理。

当然，本题不需要用到数组，但是用数组之后可以利用循环达到减少代码量的作用，而且对于不同位数的分类操作也“隐藏”进了 `for (int j = 5 - iNumOfDigits; j < 5; j++)` 和 `for (int j = 4; j >= 5 - iNumOfDigits; j--)` 之中。

如果采用分类的做法，伪代码大致如下：

```
1  switch (iNumOfDigits){
2      case 1:
3          int a1 = integer;
4          cout << a1; //顺序输出和逆序输出皆如左侧
5      case 2:
6          int a1 = integer / 10, a2 = integer % 10;
7          cout << a1 << ", " << a2; //顺序输出
8          cout << a2 << ", " << a1; //逆序输出
9      case 3:
10         ...
11     case 4:
12         ...
13     case 5:
14         ...
15 }
```

这样操作的麻烦之处在于编写 `switch` 语句块的时候大量的重复操作所带来的错误率的上升。相比之下，就编程实现目的的复杂度而言，个人认为采用循环和数组的方案较优，采用字符串的方案更优。