# 英文排版程序

- 对于段英文文章按指定列数进行排版，满足下列要求：

1.排版后除最后一行外，所有的行都你具有相同的列数，而且左右两端是对齐的。

2.每行行首不能以空格和标点符号开始，每行行尾不为空格。

3.为了简化程序设计假设文本中每个单词间用1个空格分隔；标点后直接是字母；没有回车换行符。标点符号只考虑后文本中的“,”、“.”和“!”

# 短文用例

We have an old musical instrument.It is called a clavichord.It was made in Germany in 1681.Our clavichord is kept in the living-room.It has belonged to our family for a long time.The instrument was bought by my grandfather many years ago.Recently it was damaged by a visitor.She tried to play jazz on it!She struck the keys too hard and two of the strings were broken.My father was shocked.Now we are not allowed to touch it.It is being repaired by a friend of my father's.

# 设计思路：

将文本存入一个字符数组(strText)；读入列宽(numColumn)；如果文本长度(textLen)小于列宽则直接将文本打印输出；否则将起始字符位置(startChar)设为0，终止位置(endChar)设计为numColumn-1，判断endChar处的字符属性：如果为标点则直接文本输出，如果是空格则将endChar-1与startChar之间的字符全部复制到strALine字符数组中，并将startChar设为endChar+1；如果是字符则逐个字符向前查找直到为标点或空格就用前面的办法处理。如果strALine的长度小于numColumn，则将其差用空格平均插入strALine已经有的空格中并将其存入strOutput字符数组，否则直接将strALine存入strOutput打印输出strOutput。

# 算法设计

1. 将文本保存在一个字符数组里**strText**
2. 输入列数为**numColumn**，设每行最后一个字符的位置在strText中的位置为**endChar**=numColumn-1，设起始位置为**startChar=0**。将strText长度存入**textLen**。
3. endChar=startChar+numColumn
4. if(endChar>=textLen的长度)
5. 　　将startChar至textLen间的字符，存入字符数组**strOutput**；startChar=textLen;
6. else
7. 　　分行处理。
8. if(endChar<textLen)
9. 格式化，并存入strOutput字符数组，打印strOutput至屏幕
10. 重复3至9步，直到startChar>=textLen。

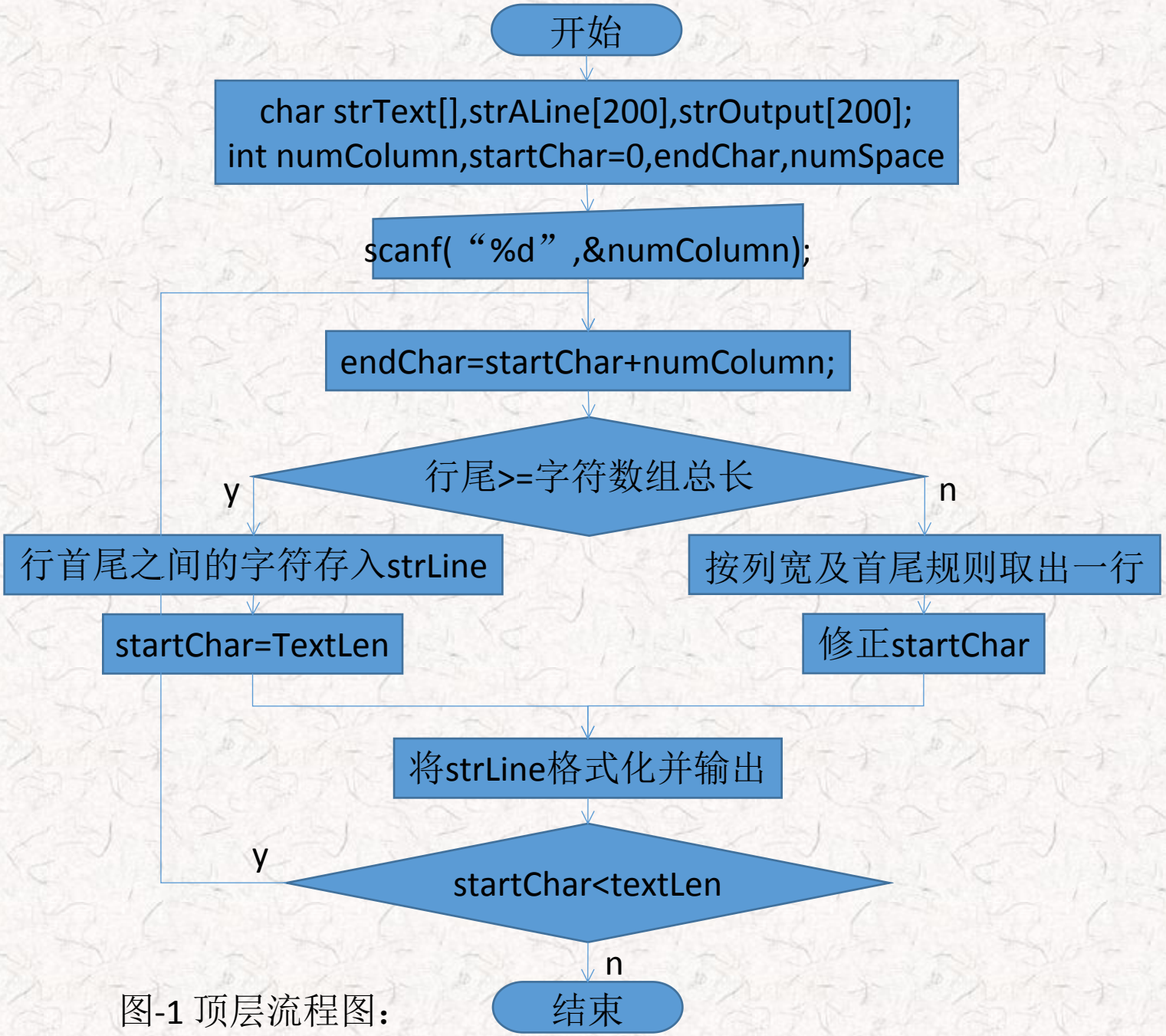
开始
char strText[],strALine[200],strOutput[200];
int numColumn,startChar=0,endChar,numSpace
scanf(“%d”,&numColumn);
endChar=startChar+numColumn;
行尾>=字符数组总长
y
n
行首尾之间的字符存入strLine
按列宽及首尾规则取出一行
startChar=TextLen
修正startChar
将strLine格式化并输出
y
startChar<textLen
n
结束

图-1 顶层流程图：

# 数据流分析

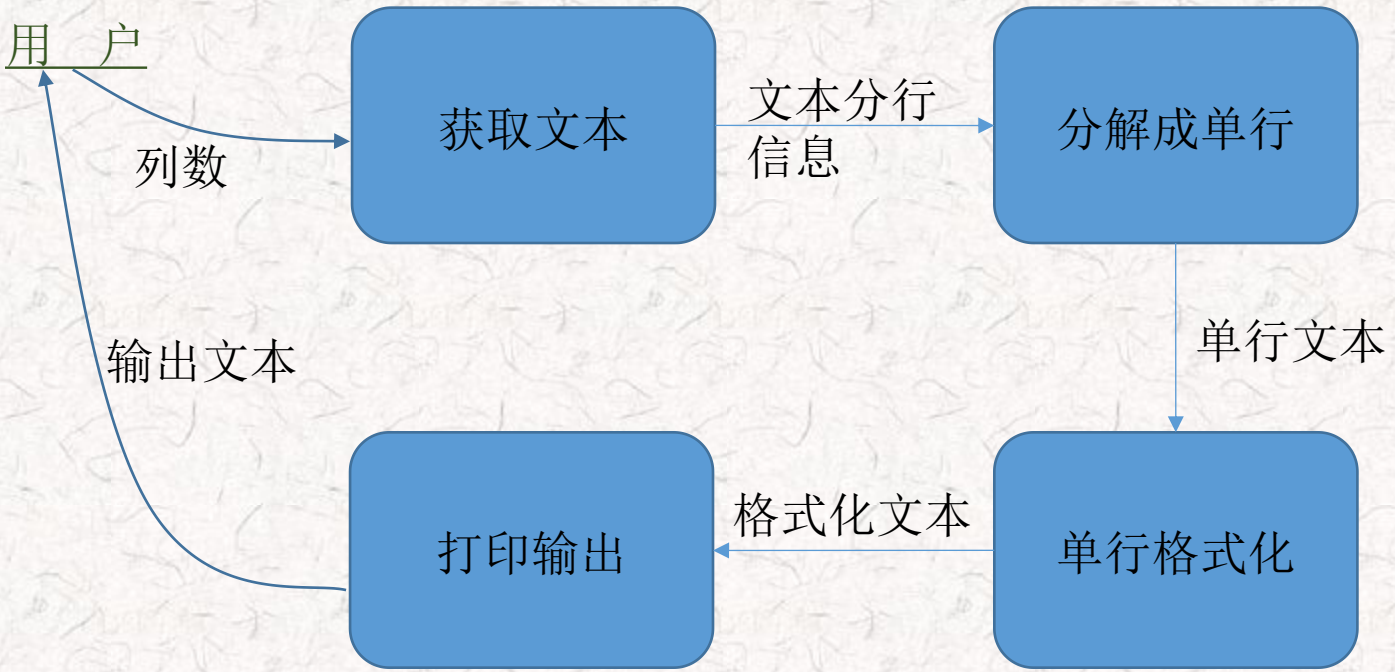


图-2 数据流图

# 数据字典

1. 列数={numColumn(int)}
2. 文本分行信息={startChar(int),endChar(int),strText[](char)}
3. 单行文本={strALine[200](char)}
4. 格式化文本={strOutput[](char)}
5. 输出文本=格式化文本

# 7. 分行处理

7.1 switch(endChar处字符)//分行处理

case 标点：

7.2 将startChar至endChar之间的字符复制到 **strALine**字符数组中。并将startChar设置为endChar+1；

case 空格：

7.3 则将startChar至endChar-1之间的字符复制到**strLine**字符数组中。并将startChar设置为endChar+1(去掉下行行首的空格)；

default：

7.4 行尾为字符的处理getALine();

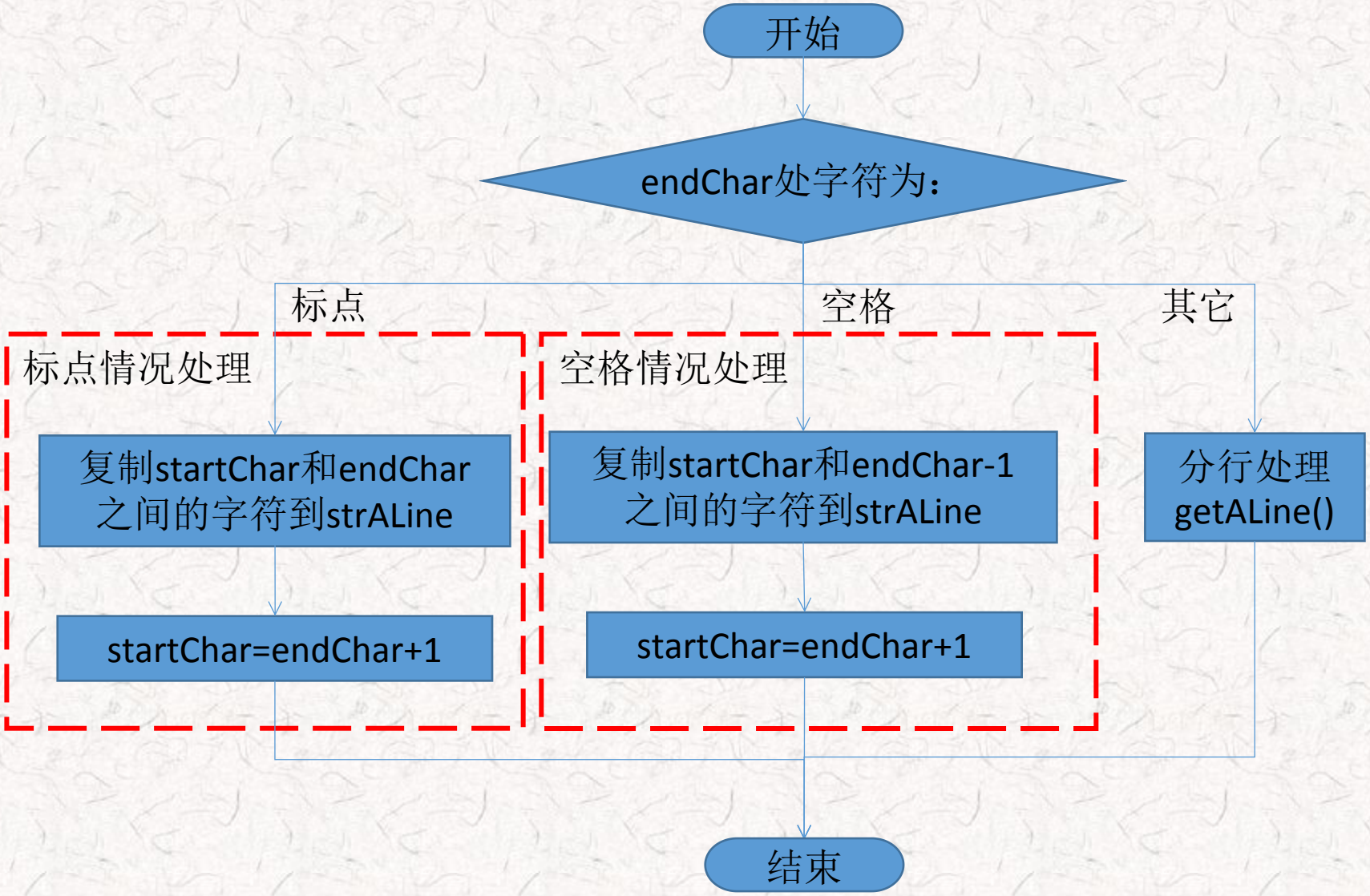


图-3 分行处理模块流程图

# 7.4行尾为字符的处理getALine()

向前查找直到出现标点或空格，将endChar设为该位置，执行7.1。

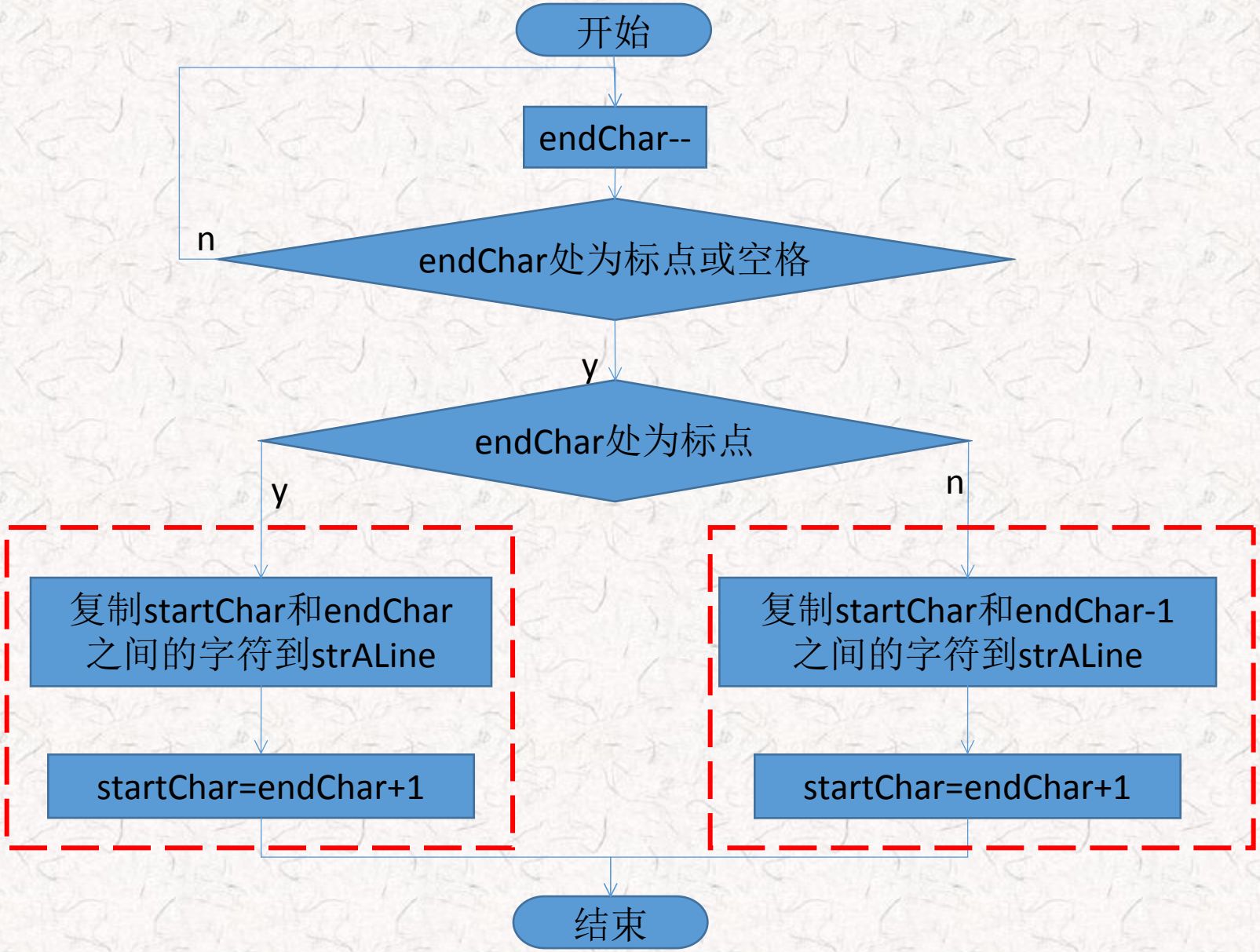


图-4 行尾为字符的处理getALine()

# 9. 格式化及输出处理算法设计

9.1 if(文本长度不于列宽且strALine长度小于列宽)

9.1.1 计算numColumn与strALine的长度差numSubstractLen；

9.1.2 计算strLine中的空格数numSpace；

9.1.3逐个将strALine字符数组中的字符复制到strOutput，在空格处按要求插入若干空格，在最后加\0和\n；

9.2 else 将strALine直接复制到strOutput。

9.3 打印strOutput

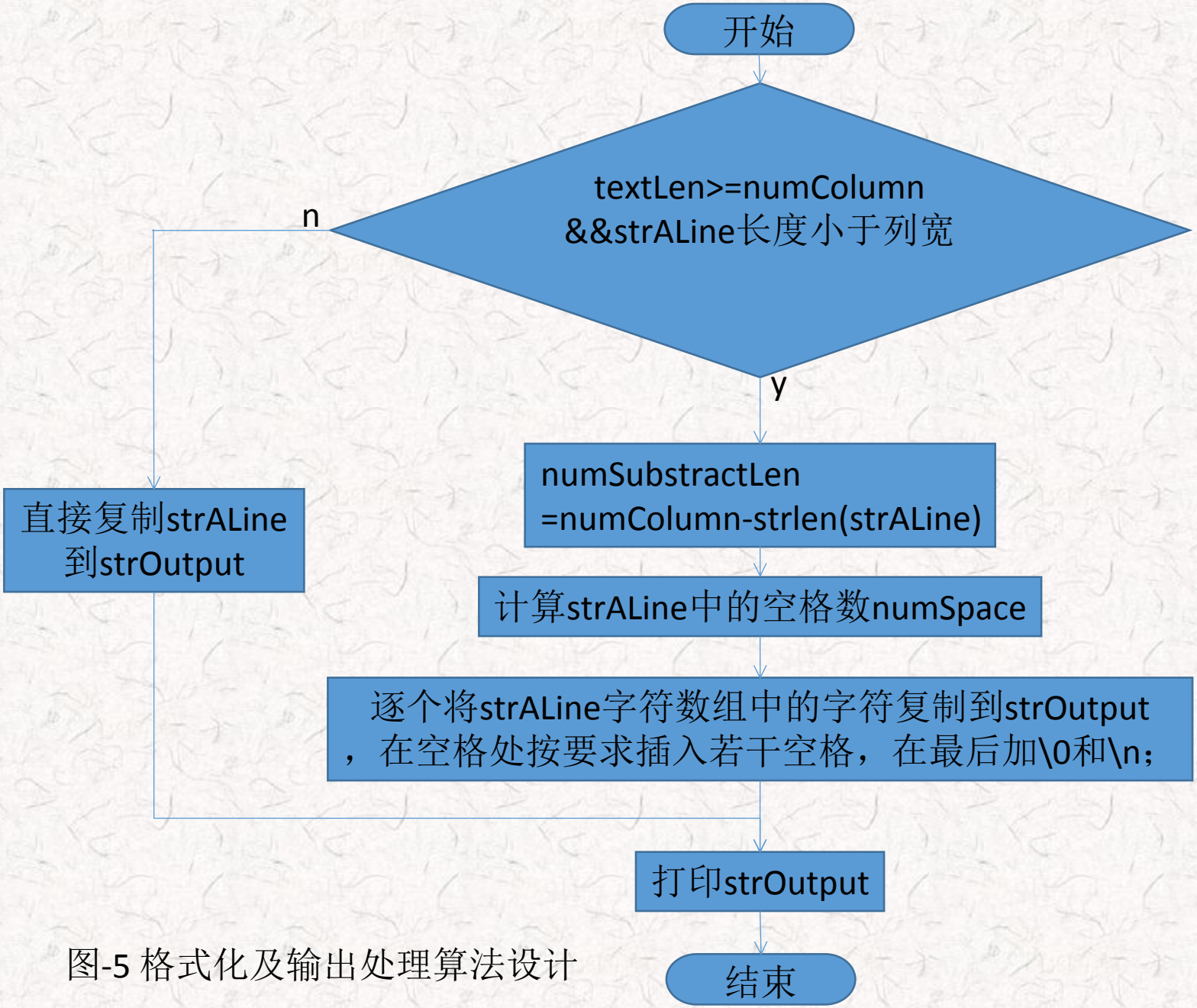


图-5 格式化及输出处理算法设计

# 9.1.3 格式化输出数组

9.1.3.1计算插入每个空格位置的最少空格数numInsert;

9.1.3.2计算插入最少空格数后的余数numInsert1；

9.1.3.3字符数组strOutput[0]全部初始化为空格，设变量int j=0用于记录写入的位置。

```
9.1.3.4for(i=0;i<strlen(strAline);i++){
        if(' '==strALine[i])
          { j=j+numInsert+1;
          if(0<numInsert1)
            { j++;
              numInsert1--;}}
        strOutput[j]=strALine[i];
        j++;}
```

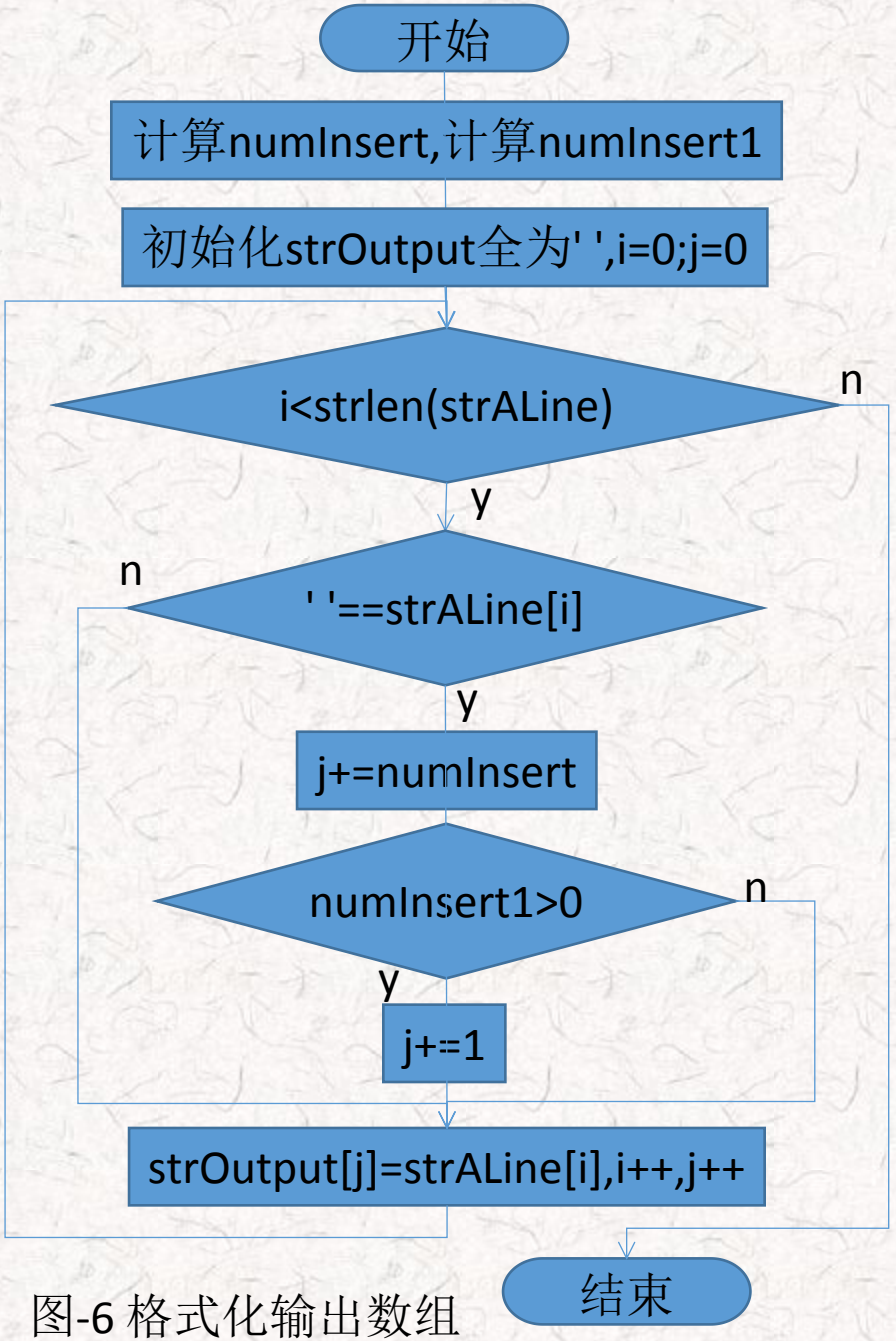
开始
计算numInsert,计算numInsert1
初始化strOutput全为' ',i=0;j=0
i<strlen(strALine)
n
y
' '==strALine[i]
n
y
j+=numInsert
numInsert1>0
n
y
j+=1
strOutput[j]=strALine[i],i++,j++
结束

图-6 格式化输出数组

# 注：插入空格数的计算

numInsert=(strlen(strALine)-numColumn)/numSpace

numInsert1=(strlen(strALine)-numColumn)%numSpace