**实验报告5：标准库类型之string**

**黄勖 22920212204392**

背景：C++提供了字符型常量、字符型变量和字符串常量，但没有提供字符串变量类型。

目前，处理字符串主要通过三种途径：

1、字符数组：有空间

2、字符型指针

3、string：有空间，且功能强大

#include <string>

#include <iostream>

using namespace std;

int main() {

string movieTitle;

cin >> movieTitle;

cout<<"Movie is"<<movieTitle<<endl;

return 0;

}

一、程序验证题：

1.1 分析以下代码，并回答问题

#include <string>

#include <iostream>

using namespace std;

int main() {

string movieTitle;

**movieTitle=”Gone with Wind”; //(\*)**

cout<<”Movie is”<<movieTitle<<endl;

return 0;

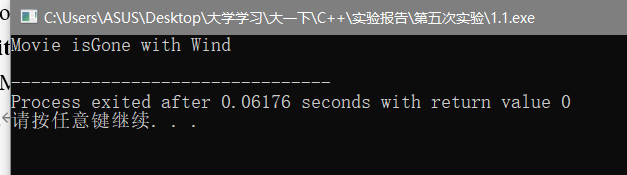
}

1 分析语句功能，输出结果

**[分析]代码语句定义了string变量movieTitle，内容为”Gone with Wind”，最后程序将”Movie is”和movieTitle的内容合并输出**

**[输出结果]** **Movie isGone with Wind**

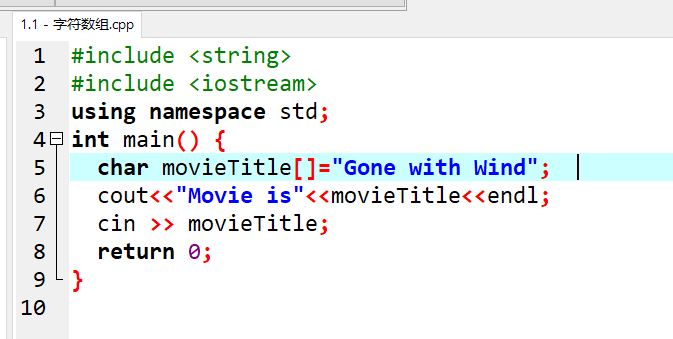
**[实验验证]**



2 请根据上述程序功能，分别使用字符数组和指针变量完成上述功能，描述三者实现上的不同（请特别分析（\*）赋值语句的执行）。

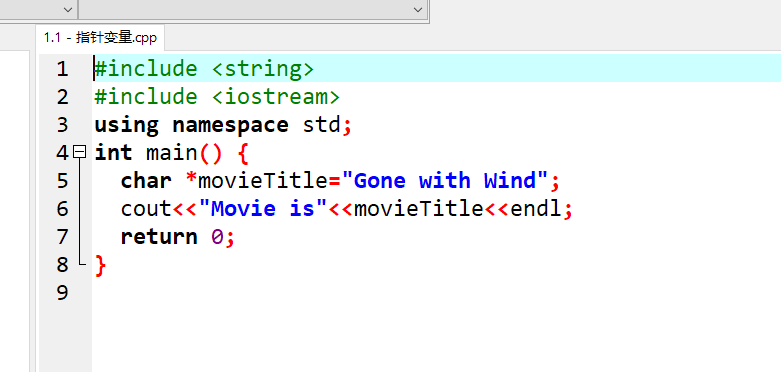
**[实验编写]**

1. **字符数组**



**分析：该赋值语句中将”Gone with Wind”分别一个字符一个字符存在movieTitle数组中；**

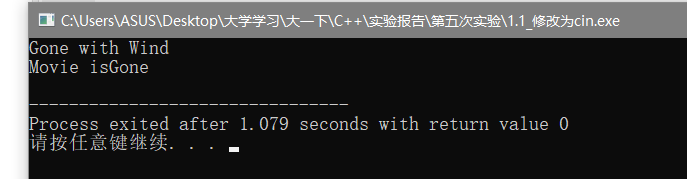
1. **指针变量**



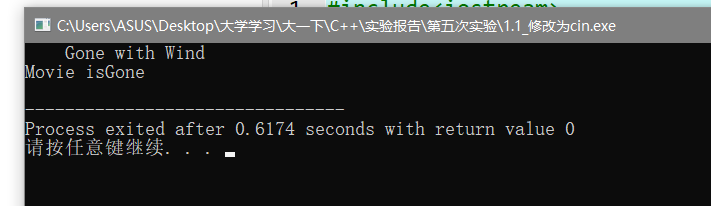
**分析：该赋值语句中将”Gone with Wind”作为字符串常量存在指针movieTitle中**

3 将(\*)语句修改成cin>>movieTitle; ，并且输入值为”Gone with Wind”和” Gone with Wind”【前面带有空格】，请关注输出结果。若输出结果与输入信息不一致，请分析原因。

**[实验] 输入”Gone with Wind”**



**输入” Gone with Wind”**



**结果：输出结果相同**

**原理：cin以空格为结束符 两次输入都只在缓冲区读入了“Gone“字符**

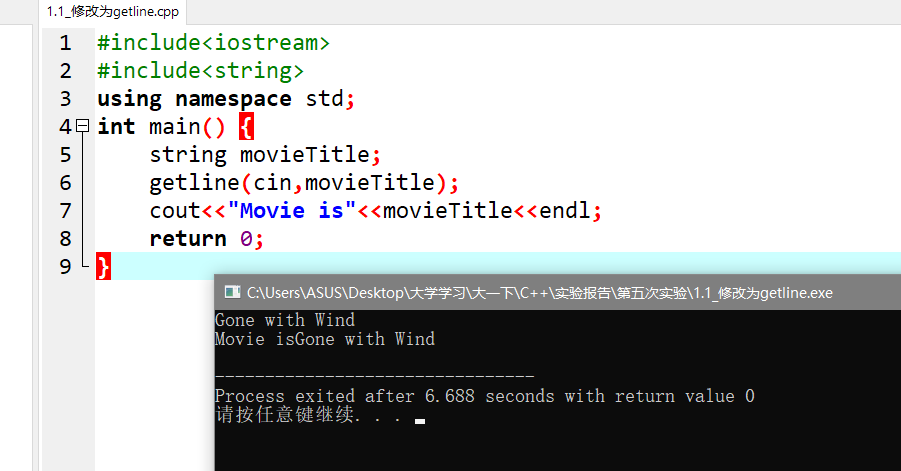
4 对比，C语言中scanf和gets两个函数在读取字符串信息上的不同。

**[解答] scanf不能接受空格、制表符Tab、回车等；对末尾回车符的处理：把回车符保留在缓存中。**

**而gets能够接受空格、制表符Tab和回车等；对末尾回车符的处理：接收回车，但把回车替换为\0.**

5 将(\*)语句修改成getline(cin,movieTitle); , 并且输入值为”Gone with Wind”，请关注输出结果。

**[实验结果]**



6 查找下cin.getline()的使用，并与5做对比。

**[cin.getline()的使用]**

1、cin.getline()实际上有三个参数，**cin.getline(接收字符串的变量,接收字符个数,结束字符)**  
2、当第三个参数省略时，系统默认为'\0'  
**[对比]cin.getline()属于istream流，而getline()属于string流，是不一样的两个函数**

**cin.getline()函数是处理数组字符串的，其原型为cin.getline(char \* , int)，第一个参数为一个char指针，第二个参数为数组字符串长度。**

**getline(cin,str)函数是处理string类的函数。第二个参数为string类型的变量。**

**[注意]当同时使用 cin>> , getline() 时，需要注意的是，在cin>>输入流完成之后，getline()之前，需要通过**

**str="\n";**

**getline(cin,str);**

**的方式将回车符作为输入流cin以清除缓存，如果不这样做的话，在控制台上就不会出现getline()的输入提示，而直接跳过，因为程序默认地将之前的变量作为输入流。**

7 将第3小题和第5小题的，两种string的输入方式做对比

**[对比]**

**cin>>**

根据cin>>sth 中sth的变量类型读取数据，这里变量类型可以为int，float,char,char\*,string等诸多类型。这一输入操作，在遇到结束符（Space、Tab、Enter）就结束，且对于结束符，并不保存到变量中。注意：最后一个enter也在缓冲区。

**cin.getline(字符数组名，接收长度，结束符)**

其中结束符意味着遇到该符号结束字符串读取,默认为enter，读取的字符个数最多为（长度 - 1），**因为最后一个为'\0'**。要注意的是，cin.getline()操作遇到结束符停止读取，但并不会将结束符从缓冲区丢弃。

1.2分析以下代码，并回答问题

#include <string>

#include <iostream>

using namespace std;

int main() {

string set1=”XYY”; string set2=”XYZ”;

char set3[10]=”XYX”;

**cout<<(set1>set2)<<endl; //\***

**cout<<(set1==set3)<<endl; //\*\***

return 0;

}

1 分析语句功能，输出结果

**[分析] string set1=”XYY”; string set2=”XYZ”;**

**char set3[10]=”XYX”;**

**该部分定义了三个字符串的内容和类型**

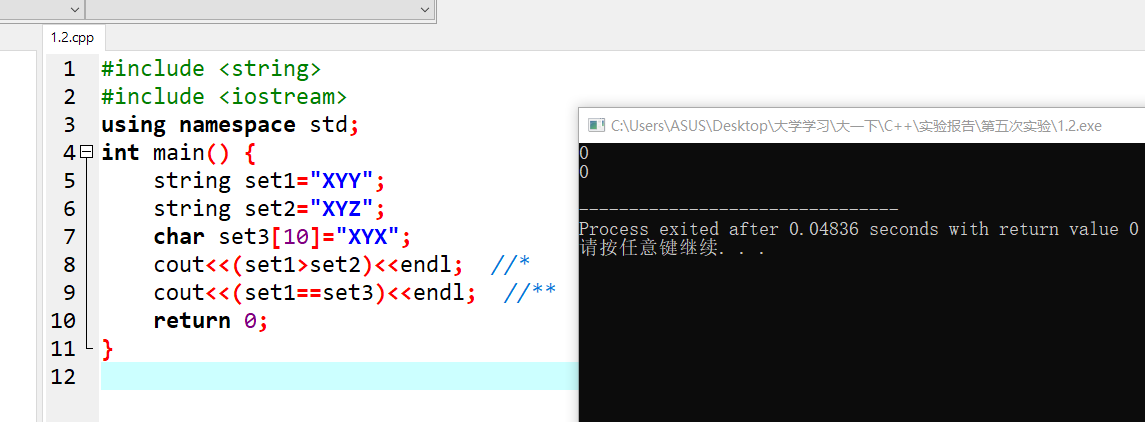
**cout<<(set1>set2)<<endl; //\***

**该句判断set1和set2的大小关系 并输出**

**cout<<(set1==set3)<<endl; //\*\***

**该句判断set1是否与set3相等 并输出**

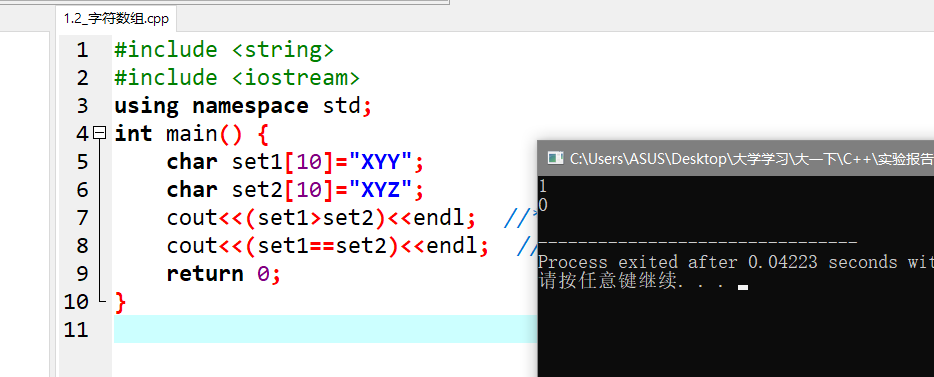
**[实验输出结果]**

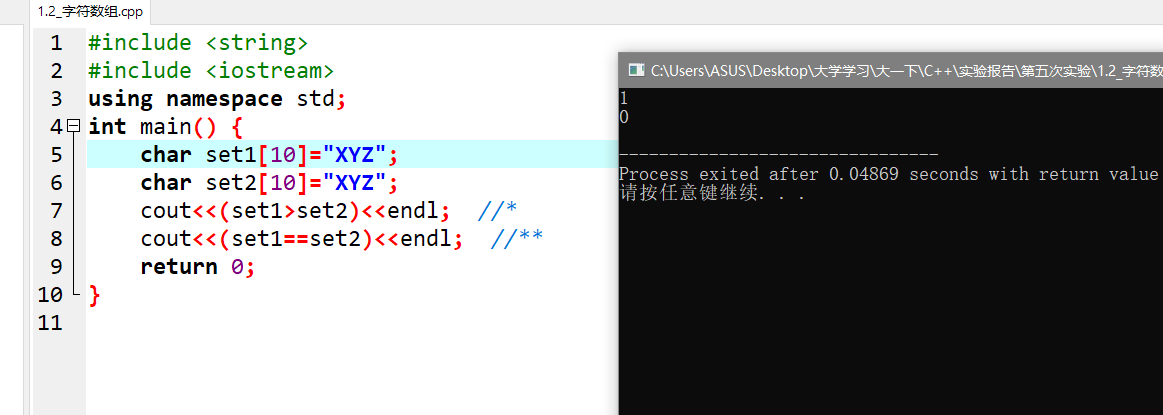


2 请根据上述程序功能，使用字符数组（或字符指针）完成上述功能，描述两者实现上的不同（重点分析关系运算符是否能直接应用于字符数组模式）

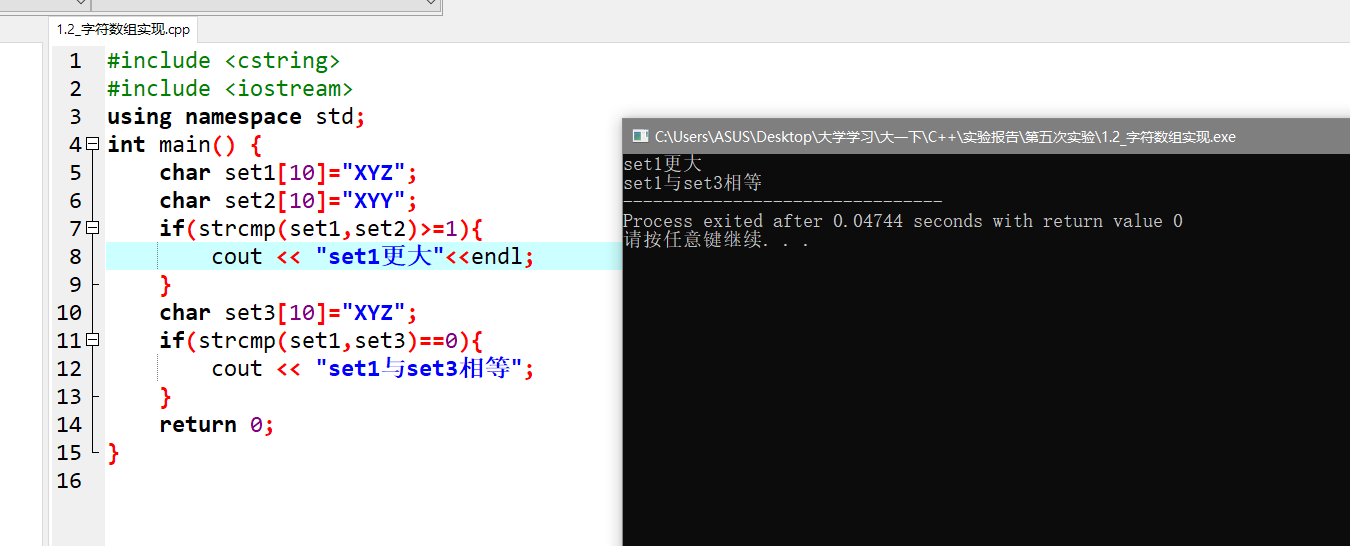
**[实验结果]**

**选择字符数组完成。**





**实际实现：**



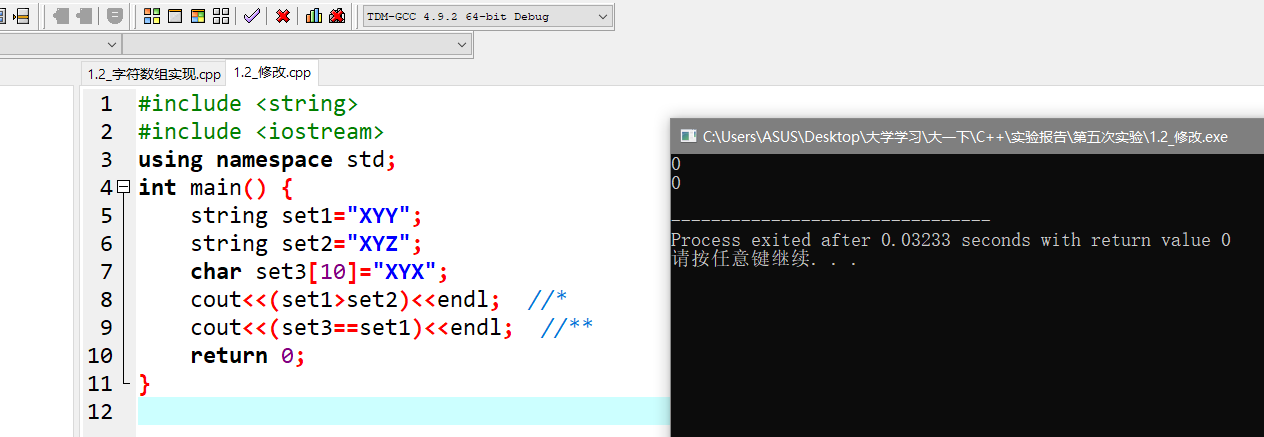
**[不同]**

**在本次实验中用strcmp才能实现正确的比较。**

**[分析]**

**关系运算符不能直接应用于字符数组模式 char数组直接进行比较的话比较的是数组的地址相对大小，这样比较并不能实现实际的目的。**

3 将（\*\*）语句修改成：cout<<(set3==set1)<<endl; 看看结果？



**结果不变**

4 比较cout<<(set1==set2)<<endl; 和 cout<<(set1.size( )==set2.size( )<<endl; 有什么区别？

**[分析]前者比较的是字符串的内容是否相等，后者比较的字符串的长度是否相等。**

1.3分析以下代码，并回答问题

string str1,str2,str3;

str1=”ABC”; str2=”DEF”;

str3=str1+str2;

cout<<str3<<endl;

str3+=”GHI”;

cout<<str3<<endl;

1 分析片段的输出结果

**[分析]**

**str1=”ABC”; str2=”DEF”;**

**定义了两个字符串的内容**

**str3=str1+str2;**

**将两个字符串拼接在一起并赋值给str1**

**cout<<str3<<endl;**

**输出str3的结果**

**str3+=”GHI”;**

**将“GHI”的内容凭借在str3的结尾**

**cout<<str3<<endl;**

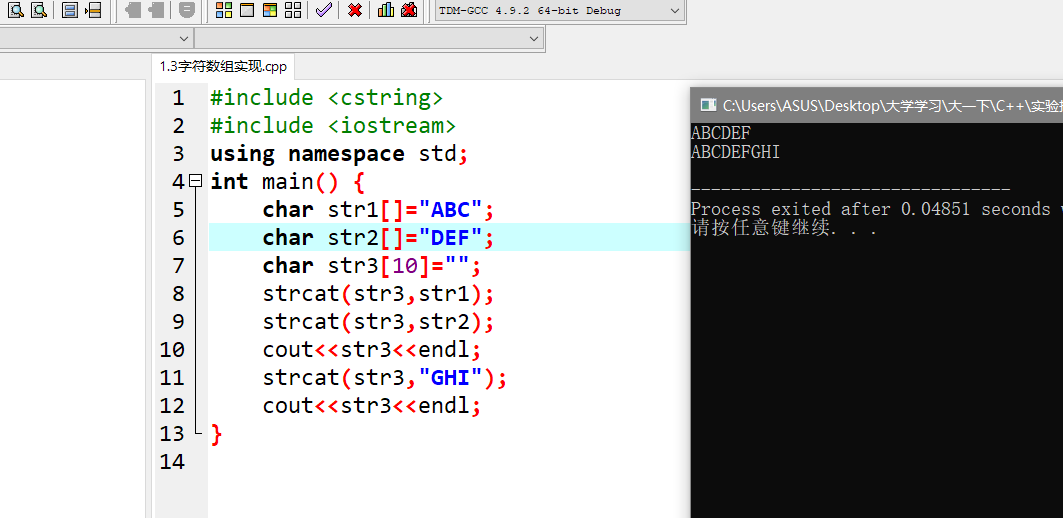
**再次输出结果**

**[实验结果]**



2 请根据上述程序功能，使用字符数组（或字符指针）完成上述功能。

**使用了字符数组来完成相同功能。**



1.4分析以下代码，并回答问题

#include <string>

#include <iostream>

using namespace std;

int main() {

string set1;

set1[0]=’A’; set1[1]=’B’; **//\***

**cout<<set1<<endl;**

return 0;

}

1 分析代码的运行结果

**[分析]**

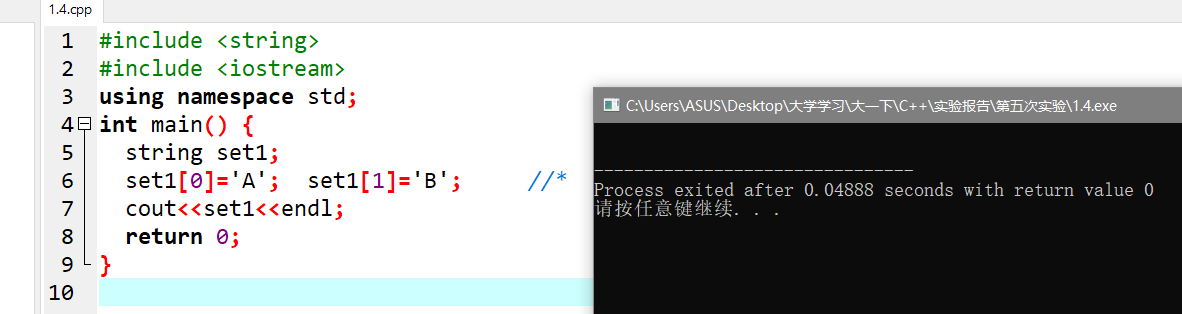
set1[0]=’A’; set1[1]=’B’;

**代码直接对string下标赋值**

**cout<<set1<<endl;**

**输出set1内容**

**[实验结果]**



2 分析（\*）语句存在的问题

**[分析]string类型为动态分配空间，下标长度未知，故变量不能直接对未占用空间的下标进行赋值，本句代码存在错误。**

1.5分析以下代码，并回答问题

#include <string>

#include <iostream>

using namespace std;

int main() {

string str1=”China”;

str1[0]=’c’;

**cout<<str1[-2]; //\***

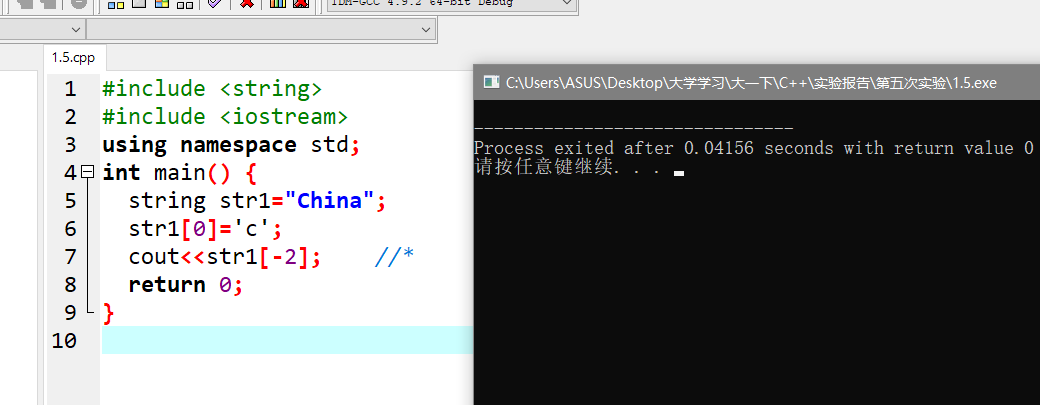
return 0;

}

1 分析代码的运行结果

**[分析] cout<<str1[-2];输出一个不存在的值 故无输出**

**[实验结果]**

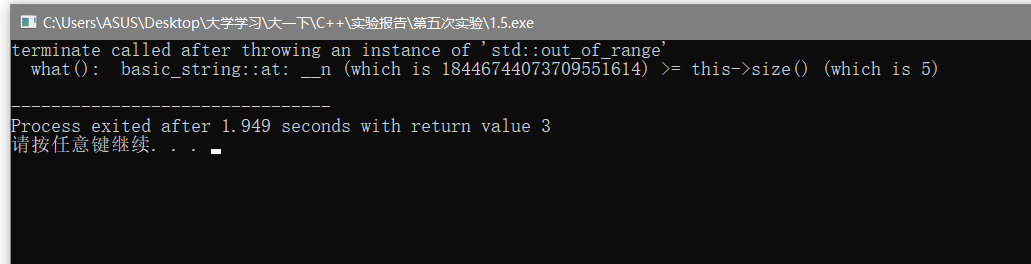


2 分析（\*）语句存在的问题

**[问题] cout<<str1[-2];输出一个不存在的值**

3 若把（\*）语句修改为str1.at(-2); 有什么不变化？分析原因

**[分析]语句语法有误 仍然不输出 无变化**



二、程序填空题

在金融行业中，输出人民币的格式为RMB1 234 567.89形式，即数量的前面加上一个人民币符号，并在数值的适当位置采用逗号分开。编写一个函数实现人民币的“格式化”输出。

|  |  |
| --- | --- |
| 1 | #include <iostream> |
| 2 | #include <string> |
| 3 | using namespace std; |
| 4 | void RMBFormat(string &currency); |
| 5 | int main() { |
| 6 | string input; |
| 7 | cout<<”按照nnnn.nn格式输入人民币的数量：”; |
| 8 | cin>>input; |
| 9 | RMBFormat(input); |
| 10 | cout<<”格式化结果：”<<input<<endl; |
| 11 | return 0; |
| 12 | } |
| 13 | void RMBFormat(string &currency) { |
| 14 | int dp; |
| 15 | dp=currency.find(‘.’)// 查找其中的点 |
| 16 | if (dp>3) |
| 17 | for (int x=dp-3;x>0;x-=3) |
| 18 | currency.insert(x,”,”); |
| 19 | currency.insert(0,”RMB”); |
| 20 | } |

1 根据题目要求，填空

**填空如上**

2 对应第9条语句，重点分析第13条语句

**[分析]第9条语句将input字符串传递给RMBFormat参数 从而让函数格式化字符串；在第13条语句中 currency 参数以引用的方式传入 从而能够在函数体内改变外部变量的值，从而改变输入的input字符串的字符。**

三、问答题

背景：字符串的操作中很注重：合久必分的思维。即，对字符串的操作往往在循环的支持下，实现对单个字符的处理。其中，如**何判定循环结束的条件和循环变量如何设置就是关键。**在string中也不例外。

string s=”He123llo”; int cout=0;

for (string::size\_type index=0;index!=s.size();++index) { //（\*）

ch=s[index];

if (ch>=’0’&&ch<=’9’) cout++;

}

请回答：

1 若将（\*）中的语句修改为for (int index=0;index!=s.size();++index) 有什么不好？

**[回答]当字符串过长时，用int类型变量可能无法表示全部的字符，造成index下标溢出，从而无法完全遍历整个string变量**

2 若将（\*）中的语句修改为for (string::size\_type index=0;index<=s.size();++index) 有什么问题？

**[回答] index!=s.size()条件最后循环不会访问到index==s.size()时的循环；但index<=s.size();会访问到index==s.size()，通常string的最后一个下标存储的是结束符，并无遍历的实际意义，故不需要等于号；**

**在遍历过程中size\_type为unsigned类型（通常为int范围的两倍） 可以进行比较运算，所以在访问到index==s.size()之前整个遍历过程不存在问题。**

3 能否将（\*）中的语句修改为for (string::size\_type index=0;s[index]!=’\0’;++index) ?

**[回答]可以修改，但不建议。在字符串存储正常时，结束符为’\0’，此时可以正常遍历，但当字符串存在多个’\0’或字符串不存在’\0’时遍历会出现错误，故不建议如此修改。**

上述代码中，string的长度没有发生改变。现在：

string s=”He123l4l5o”;

string new;

for (string::size\_type index=0;index!=s.size();++index) { //（\*）

ch=s[index];

if (ch>=’0’&&ch<=’9’)

new+=ch;

}

在这段代码中，请重新回答问题2和3。

2 若将（\*）中的语句修改为for (string::size\_type index=0;index<=s.size();++index) 有什么问题？

**[回答] index!=s.size()条件最后循环不会访问到index==s.size()时的循环；但index<=s.size();会访问到index==s.size()，通常string的最后一个下标存储的是结束符，并无遍历的实际意义，故不需要等于号；**

**在遍历过程中size\_type为unsigned类型（通常为int范围的两倍） 可以进行比较运算，所以在访问到index==s.size()之前整个遍历过程不存在问题。**

3 能否将（\*）中的语句修改为for (string::size\_type index=0;s[index]!=’\0’;++index) ?

**[回答]可以修改，但不建议。在字符串存储正常时，结束符为’\0’，此时可以正常遍历，但当字符串存在多个’\0’或字符串不存在’\0’时遍历会出现错误，故不建议如此修改。**

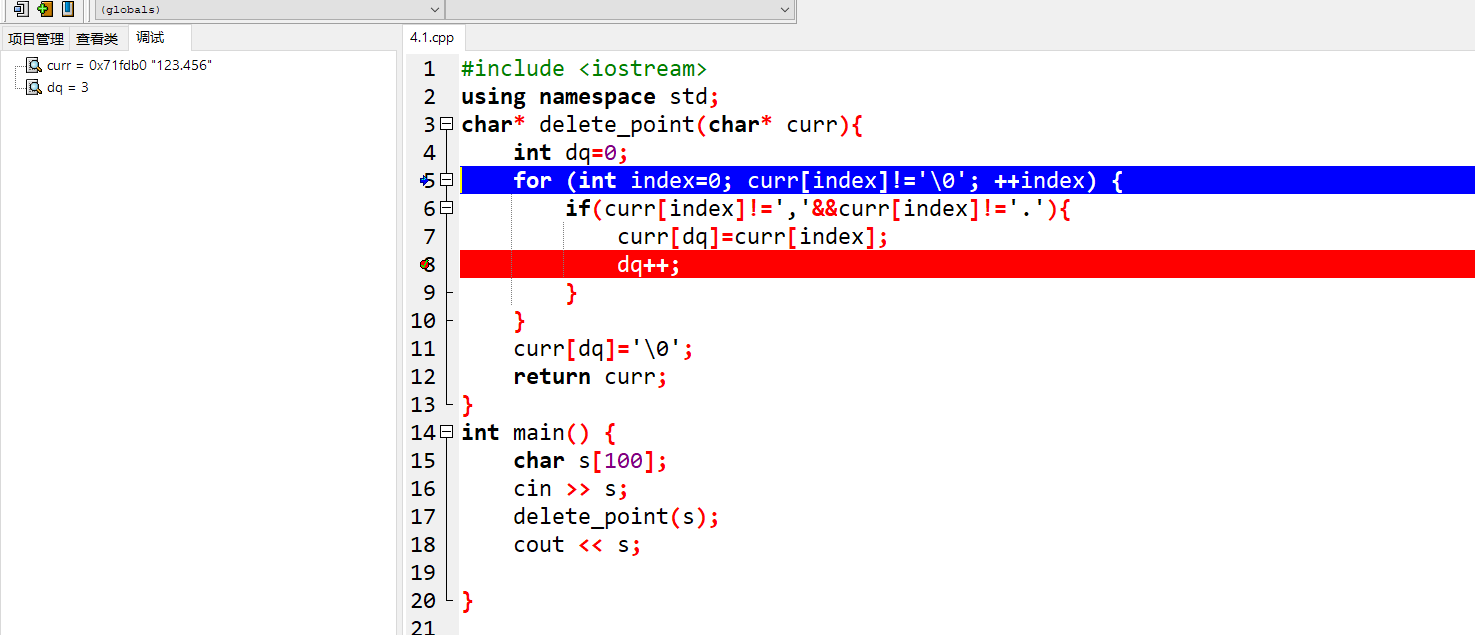
四、程序题

4.1 编写程序，采用字符数组存储字符串，从字符串中去除标点符号。

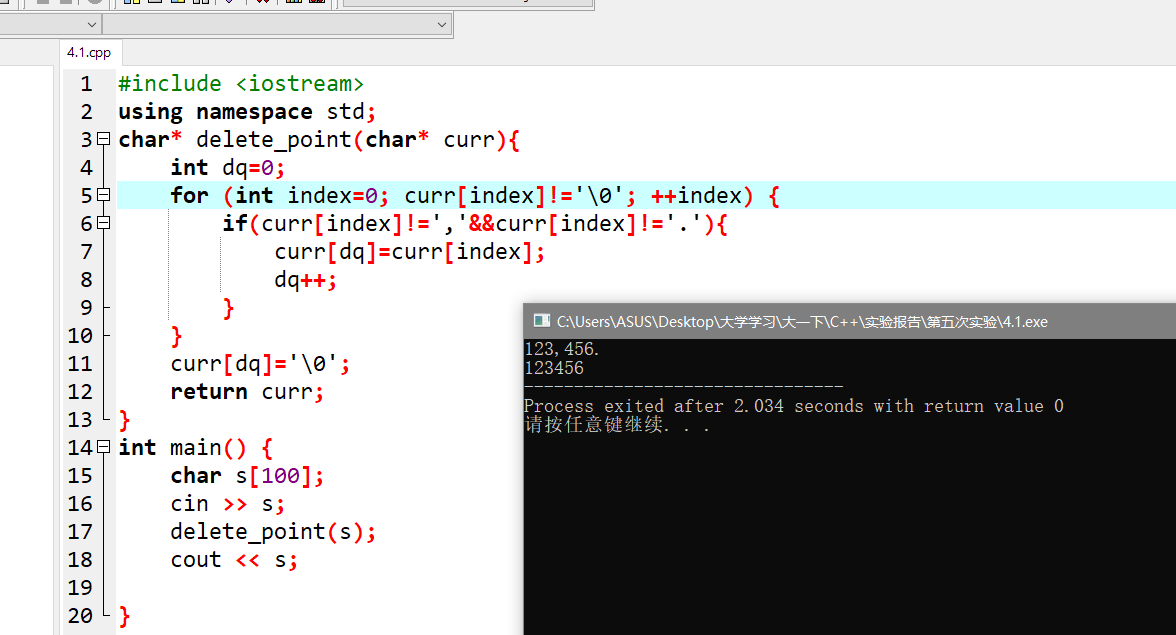
要求输入到程序的字符串必须含有标点符号，且在子函数中处理，返回输出结果则是去掉标点符号的字符串。

[代码]见4.1.cpp

[调试过程]



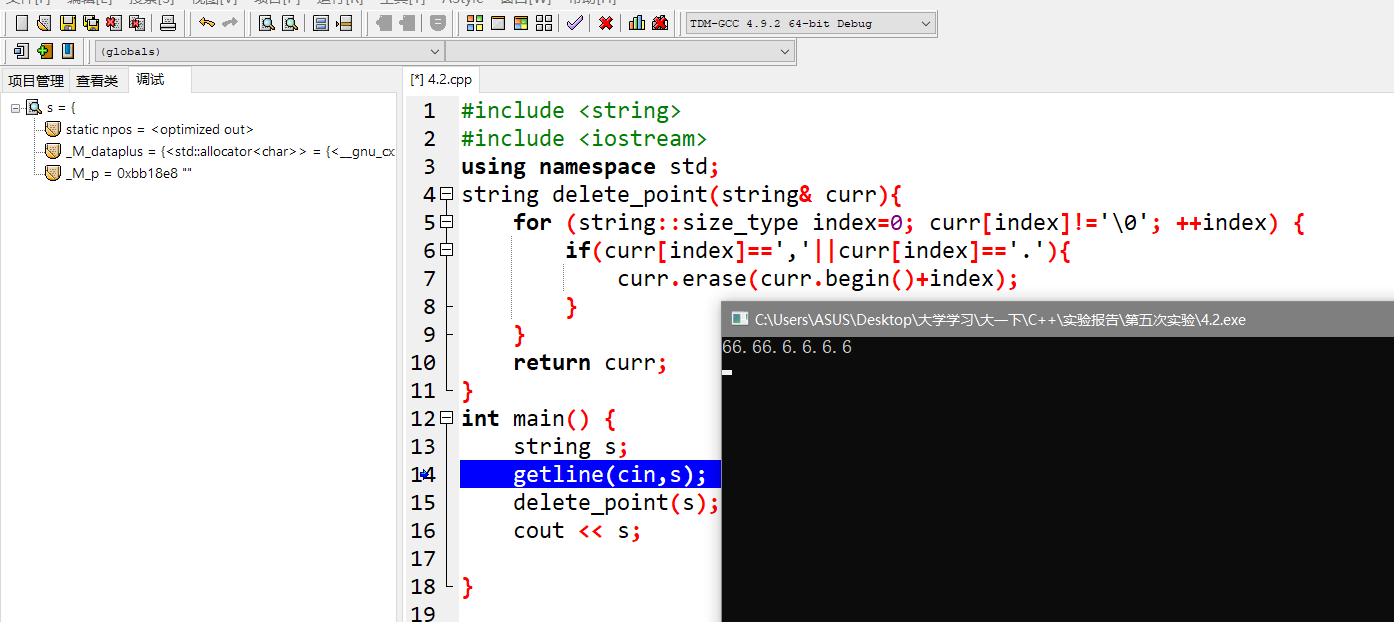
[实验结果]



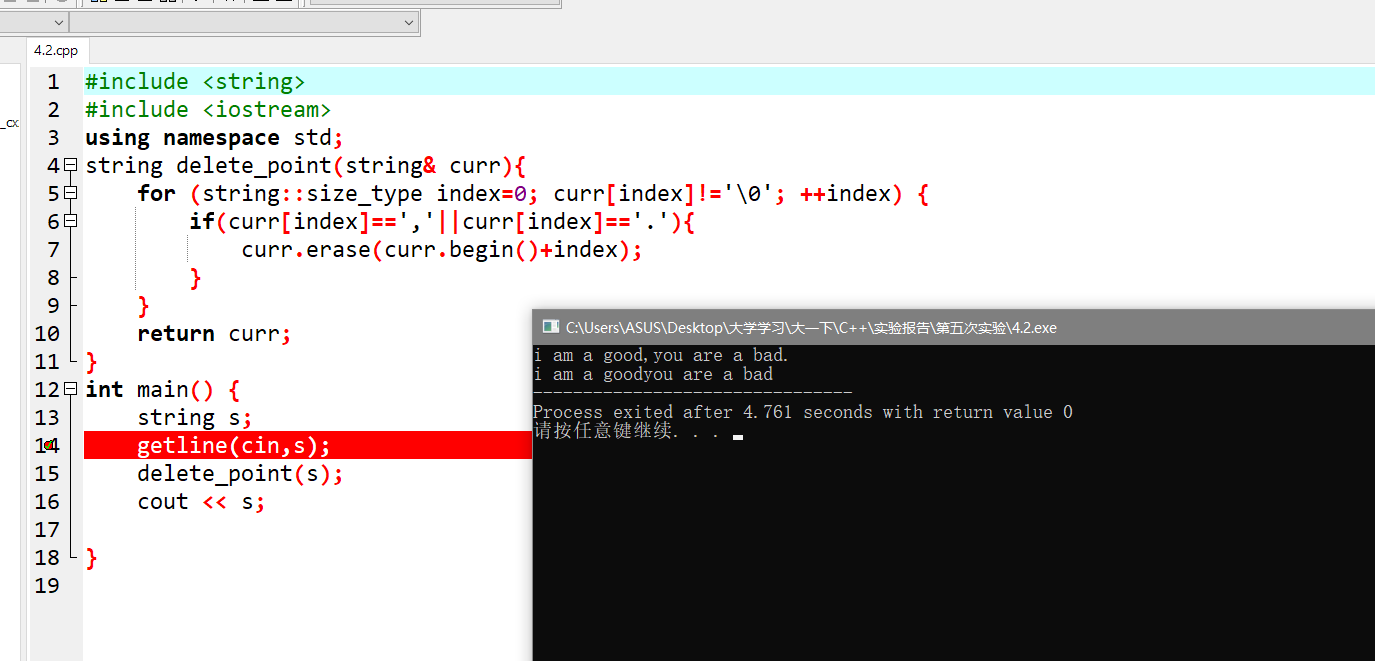
4.2 编写程序（参考第三题），从string对象中去掉标点符号。要求输入到程序的字符串必须含有标点符号，输出结果则是去掉标点符号的字符串。

[代码]见4.2.cpp

[调试过程]



[实验结果]

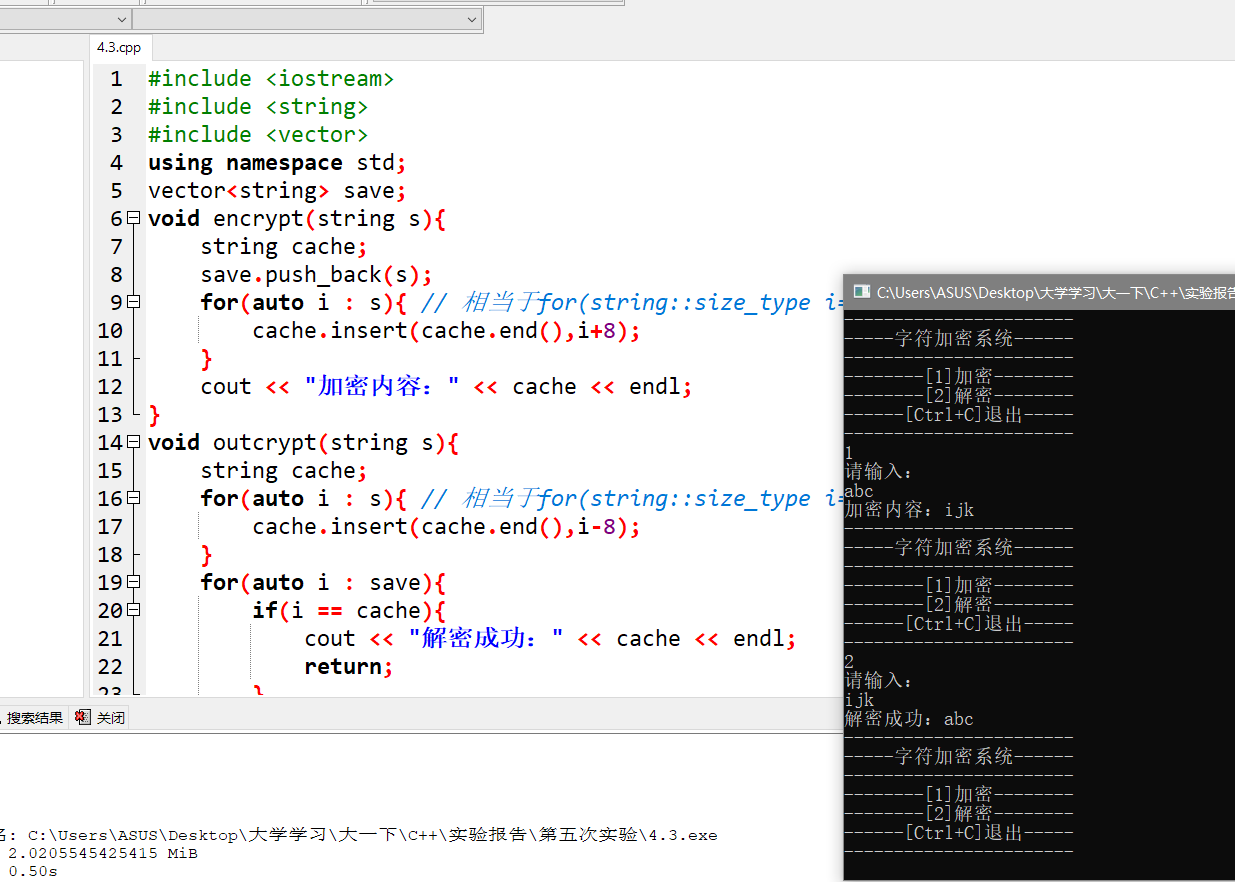


4.3采用string编写程序，将输入的一行字符以加密的形式输出（写一个加密函数），然后将其解密（写一个解密函数），解密的字符序列与输入的正文进行比较，吻合时输出解密的正文，否则输出解密失败。

注意：加密原则，将每个字符的ASCII码加上8；解密与加密的顺序相反。

[代码]见4.3.cpp

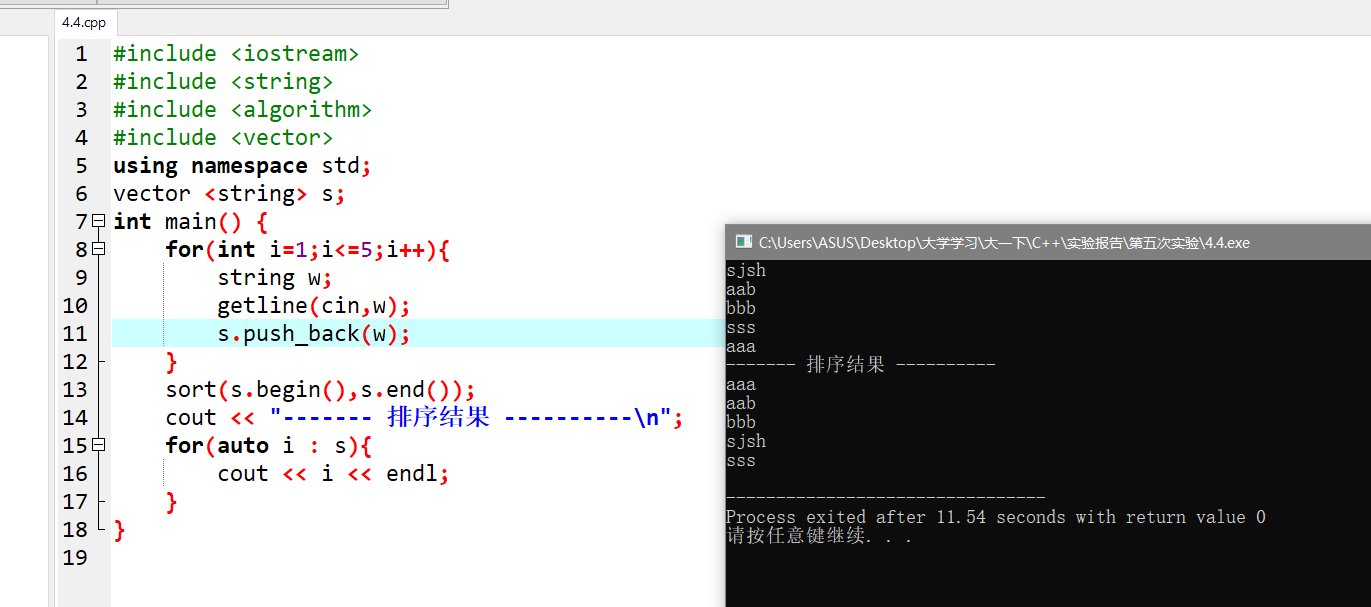
[实验结果]



4.4 编写程序，从键盘上输入5个字符串，要求将它们按由小到大的顺序排列，用string方法。

[代码]见4.4.cpp

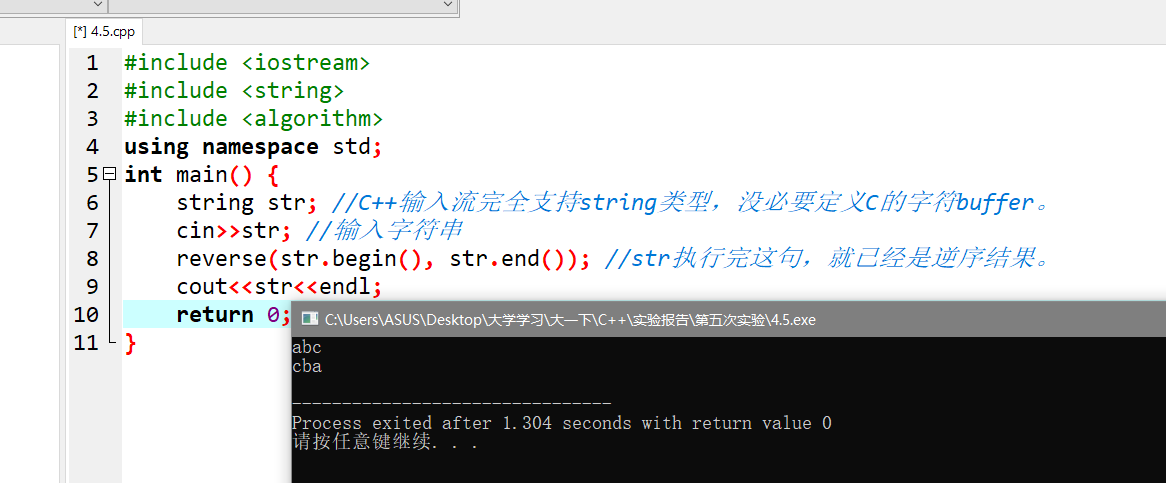
[实验结果]



4.5 编写程序，输入一个字符串，将其中的字符按逆序输出（写一个逆序函数）。如输入LIGHT，输出THGIL。要求使用string方法。

[代码]见4.5.cpp

[实验结果]



4.6 编写程序，将4个string对象分别初始化为This, is, Xiamen 和University，然后在这些字符串之间添加空格，再显示整个句子。

[代码]见4.6.cpp

[实验结果]

