

实验四 C++对C的扩展（动态存储与链表）

一、 问题描述

1. 实验目的：

掌握“C++对C扩展”中涉及的若干基本概念和特性，并能够应用于程序编写。

掌握验证性实验的基本方法和过程(认知、实验、总结)。

2. 实验内容：

分别编写一段测试代码来回答任务书中的相关问题（每一个问题，用一个工程文件，同时需要记录相应的调试过程），具体问题请参考“实验任务说明05.doc”；

调试的过程；（动态调试的相关截图，比如 设置断点、查看当前变量值等）；

编译出来的可执行程序单独放在一个目录下（bin/exe/debug目录下，同时附上输入数据说明和输出结果）

3. 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

二、 实验过程

1. 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

string movieTitle定义一个字符串movieTitle。然后用字符串常量“Gone with Wind”直接赋值给string对象，并使用标准输出函数符来输出string对象。

输出结果：Movie is Gone with Wind



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）。

字符数组赋值时，“Gone with Wind”在数组中对应存储‘G’、‘o’、‘n’、‘e’……的信息。而数组名movieone是一个指针常量。它是不可改变地址的指针，但可以对它所指向的内容进行修改，即对储存在字符数组中的元素进行访问和修改。

字符型指针变量赋值时，指针变量movietwo可以指向一个字符串常量，但其实指针变量movietwo是没有字符串具体存储空间的。同时movietwo相当于一个常量指针，我们无法对字符串常量“Gone with Wind”修改。

而movieTitle是一个string对象，可以将字符串常量“Gone with Wind”赋值给它。

```

#include <string>
#include <iostream>
using namespace std;
int main() {
    string movieTitle;
    char movieone[30];
    strcpy( movieone, "Gone with Wind"); //(*)字符数组
    cout << "Movie one is " << movieone << endl;
    const char* movietwo;
    movietwo= "Gone with Wind"; //(*)字符型指针变量
    cout << "Movie two is " << movietwo << endl;
    movieTitle = "Gone with Wind"; //(*)
    cout << "Movie is " << movieTitle << endl;
    return 0;
}

```

Microsoft Visual Studio 调试控制台

```

Movie one is Gone with Wind
Movie two is Gone with Wind
Movie is Gone with Wind

D:\XPfile\学习资料\年级分类\大二
要在调试停止时自动关闭控制台，请
按任意键关闭此窗口. . .

```

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。

输入“Gone with Wind”和“ Gone with Wind”，输出结果都是Gone，这是因为string类型的输入操作符具有以下特性：它会读取并忽略开头所有的空白字符（空格、换行、制表符），并且读取字符直至再次遇到空白字符，读取终止。所以程序忽略了第二个输入Gone前的空格，同时读取字符直到再次遇到Gone后面的空格，读取终止，最终读到的movieTitle只有Gone。

(0) 实验代码

```

#include <iostream>
using namespace std;
int main() {
    string movieTitle;
    char movieone[30];
    //strcpy( movieone, "Gone with Wind"); //(*)字符数组
    //cout << "Movie one is " << movieone << endl;
    const char* movietwo;
    movietwo= "Gone with Wind"; //(*)字符型指针变量
    //cout << "Movie two is " << movietwo << endl;
    //movieTitle = "Gone with Wind"; //(*)string对象
    cin >> movieTitle;
    cout << "Movie is " << movieTitle << endl;
    return 0;
}

```

(1) 第一次输出结果



Microsoft Visual Studio 调试控制台

```

Gone with Wind
Movie is Gone

D:\XPfile\学习资料\年级分类\大
要在调试停止时自动关闭控制台，
按任意键关闭此窗口。 . . .

```

(2) 第二次输出结果



Microsoft Visual Studio 调试控制台

```

Gone with Wind
Movie is Gone

D:\XPfile\学习资料\年级分类\大
要在调试停止时自动关闭控制台，
按任意键关闭此窗口。 . . .

```

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。

- (1) gets的输入分割符只有回车，因此gets是能够读入空格的。而scanf不可以。
- (2) 此外，scanf和gets对待缓冲区里的回车符也是完全不同的。scanf在读取非空白符之前会忽略回车，读取之后如果遇到空白字符会停止输入，将其留着缓冲区里。gets只要一遇到回车就输入结束，并把这个回车从缓冲区里移走。
- (3) gets可以读取空回车，但是scanf不能。

(4) gets从标准输入设备读字符串函数。可以无限读取，不会判断上限，以回车结束读取，所以程序员应该确保buffer的空间足够大，以便在执行读操作时不发生溢出。

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。

gets的输入分割符只有回车，因此gets是能够读入空格的。而cin不可以。所以最终可以成功输出Movie is Gone with Wind。

(1) 实验代码

```
#include <iostream>
using namespace std;
int main() {
    string movieTitle;
    char movieone[30];
    //strcpy( movieone, "Gone with Wind"); //(*)字符数组
    //cout << "Movie one is " << movieone << endl;
    const char* movietwo;
    movietwo= "Gone with Wind"; //(*)字符型指针变量
    //cout << "Movie two is " << movietwo << endl;
    //movieTitle = "Gone with Wind"; //(*)string对象
    //cin >> movieTitle;
    getline(cin, movieTitle);
    cout << "Movie is " << movieTitle << endl;
    return 0;
}
```

(2) 实验结果



The screenshot shows the Microsoft Visual Studio debug console. The title bar reads "Microsoft Visual Studio 调试控制台". The console output is as follows:

```
Gone with Wind
Movie is Gone with Wind

D:\XPfile\学习资料\年级分类\大
要在调试停止时自动关闭控制台，
按任意键关闭此窗口. . .
```

6 查找下cin.getline()的使用，并与5做对比。

(1) cin.getline(字符指针(char*),字符个数N(int),结束符(char)): 此函数一次读取多个字符(包括空白字符)，直到读满N-1个，或者遇到指定的结束符为止(默认的是以'\n'结束)。

(2) `getline`: 与`cin.getline`功能类似, 但是参数不一样, 也属于两个不同的流, 是两个不一样的函数。`getline`接受的字符串长度不受限制。

(3) `scanf()`: 当遇到回车, 空格和`tab`键会自动在字符串后面添加`'\0'`, 但是回车, 空格和`tab`键仍会留在输入的缓冲区中。

① 实验代码

```
#define _CRT_SECURE_NO_WARNINGS
#include <string>
#include <iostream>
using namespace std;

int main() {
    string movieTitle;
    char movieone[30];
    strcpy(movieone, "Gone with Wind"); //(*) 字符数组
    cin.getline(movieone, 14, '\n');
    cout << "Movie one is " << movieone << endl;
    //const char* movietwo;
    //movietwo= "Gone with Wind"; //(*) 字符型指针变量
    //cout << "Movie two is " << movietwo << endl;
    //movieTitle = "Gone with Wind"; //(*) string对象
    //cin >> movieTitle;
    //getline(cin, movieTitle);
    //cout << "Movie is " << movieTitle << endl;
    return 0;
}
```

② 实验结果



7 将第3小题和第5小题的, 两种string的输入方式做对比

输入“Gone with Wind”和“ Gone with Wind”, 输出结果都是Gone, 这是因为string类型的输入操作符会读取并忽略开头所有的空白字符(空格、换行、制表符), 并且读取字符直至再次遇到空白字符, 读取终止。所以程序忽略了第二个输入Gone前的空格, 同时读取字符直到再次遇到Gone后面

的空格，读取终止，最终读到的movieTitle只有Gone。

而gets的输入分割符只有回车，因此gets是能够读入空格的。所以最终可以成功输出Movie is Gone with Wind。

1.2 分析以下代码，并回答问题

```
#include <string>
#include <iostream>
using namespace std;
int main() {
    string set1 = "XYX";    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

string对象可以保持!=,<,<=,>,>= 这些操作符惯有含义，这里set1小于set2，set1也不等于set3，所以这两个判断语句输出的结果都为0。



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）

string对象可以直接用关系运算符判断大小，而字符数组或字符指针不可以，可以使用strcmp函数对它们进行比较。比如set3和set4，由于set3小于set4，最后的输出结果为负数。

（1）实验代码

```

#include <iostream>
using namespace std;
int main() {
    string set1 = "XYX";    string set2 = "XYZ";
    char set3[10] = "XYX"; char set4[10] = "XZZ";
    cout << strcmp(set3, set4)<<endl;
    //cout << (set1 > set2) << endl;  /**
    //cout << (set1 == set3) << endl;  /**
    return 0;
}

```

(2) 实验结果



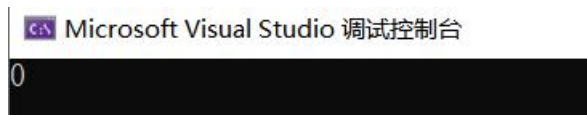
Microsoft Visual Studio 调试控制台

-1

D:\XPfile\学习资料\年级分类\大
要在调试停止时自动关闭控制台，
按任意键关闭此窗口。 . . .

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

输出0表示set3等于set1为false, 这条指令在VS中可以编译通过, 但实际上存在风险。



Microsoft Visual Studio 调试控制台

0

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

前者是比较set1字符串与set2字符串是否相等, 后者是比较set1字符串大小与set2字符串大小是否相等。由于两字符串不相等, 故第一次输出0, 由于两字符串长度相等, 故第二次输出1。

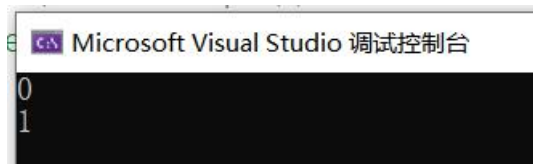
(1) 实验代码


```

#include <iostream>
using namespace std;
int main() {
    string set1 = "XYX";    string set2 = "XYZ";
    char set3[10] = "XYX";  char set4[10] = "XZZ";
    cout << (set1 == set2) << endl;
    cout << (set1.size() == set2.size()) << endl;
    //cout << (set3 == set1) << endl;
    //cout << strcmp(set3, set4)<<endl;
    //cout << (set1 > set2) << endl;  /**
    //cout << (set1 == set3) << endl;  /**
    return 0;
}

```

(2) 实验结果



1.3 分析以下代码，并回答问题

```

#include <iostream>
#include <string.h>
using namespace std;

int main() {
    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

对字符串使用`str1+str2`，可以将`str1`,`str2`连接成一个新字符串“ABCDEF”，而字符串数组存储的字符串，无法直接使用`+`，需要用`strcat()`。`string`中的`+=`可以将字符串附加到`string`对象的末尾，即`str3+=“GHI”`，最终将输出“ABCDEFghi”。

使用`+`时，左右操作数必须至少有一个是`string`类型；这是实现`+`运算符重载的需要。



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。

(1) 实验代码

```
#include <string.h>
using namespace std;

int main() {
    //string str1, str2, str3;
    //str1 = "ABC"; str2 = "DEF";
    //str3 = str1 + str2;
    //cout << str3 << endl;
    //str3 += "GHI";
    //cout << str3 << endl;
    char strone[10]{ "ABC" };
    char strtwo[10]{ "DEF" };
    strcat(strone, strtwo);
    cout << strone << endl;
    strcat(strone, "GHI");
    cout << strone << endl;
}
```

(2) 说明

字符数组存储的字符串，无法直接使用+，需要用strcat()。除此之外，为了防止堆栈溢出，应该在初始化char数组时就将其大小设为10或其他更大的大小。

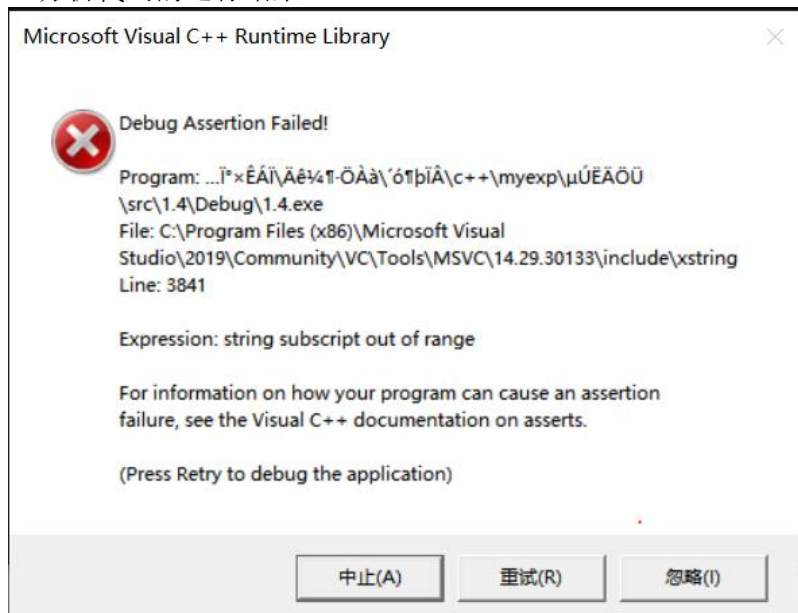
1.4 分析以下代码，并回答问题

```

#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



运行代码，VS出现“string subscript out of range”，字符串边界溢出的报错。

2 分析（*）语句存在的问题

程序中，对string set1未初始化，如果要对set1进行操作，需对s进行初始化，给其分配相应的存储空间，才可以通过下标访问里面的元素。

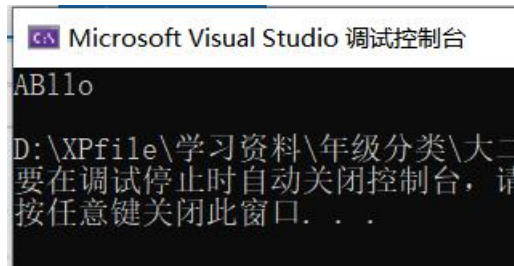
3 代码修改

给string set1先进行初始化，程序成功运行。

```

#include <string>
#include <iostream>
using namespace std;
int main() {
    string set1="Hello";
    set1[0] = 'A'; set1[1] = 'B';    /*
    cout << set1 << endl;
    return 0;
}

```



1.5 分析以下代码，并回答问题

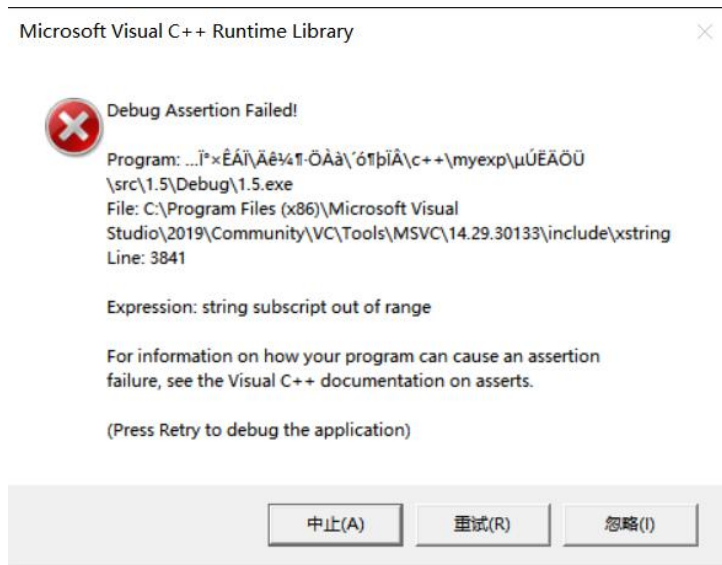
```

#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

运行代码，VS出现“string subscript out of range”，字符串边界溢出的报错。



2 分析(*)语句存在的问题

像数组一样，字符串是以0为索引开始的，这意味着有效的字符索引是0到size-1，而访问str1[-2]处的字符是未定义行为，所以会报错。

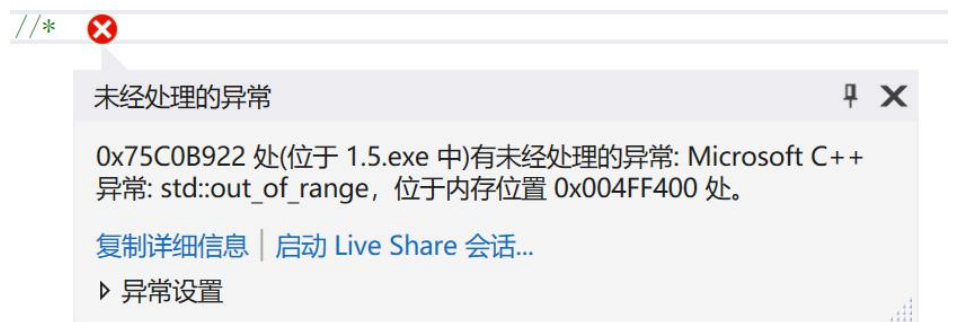
3 若把(*)语句修改为str1.at(-2)；有什么不变化？分析原因

程序抛出std::out_of_range异常，用at方式访问存在访问越界问题，这和C++容器类实现有关。对于std::map::at使用有如下说明：

Access element // 访问元素

Returns a reference to the mapped value of the element identified with key k.
// 返回元素键为k的映射值的引用，即Key为k的元素的对应value值。

If k does not match the key of any element in the container, the function throws an out_of_range exception. // 如果容器中没有匹配的k键，该函数将抛出一个out_of_range异常



2. 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.h>
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=x-3)
18	currency.insert(x, ", ");
19	currency.insert(0, " RMB");
20	}

1 根据题目要求，填空

2 对应第9条语句，重点分析第13条语句

C++的字符串是类string的对象，类对象是允许引用的，所以C++的字符串对象是可以引用传递的。故在第9条语句中，通过将input的引用传入RMBFormat函数中，在RMBFormat函数中对input字符串进行修改，最终在main函数中输出修改后的input字符串。

3. 问答题

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 = "Hel23llo"; int count = 0;
for (string::size_type index = 0; s[index] != s.size(); ++index) { // (*)
    char ch = s[index];
    if (ch >= '0' && ch <= '9') count++;
}
```

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？

`size()` 这个函数返回的类型不是整形，而是 `size_type` 类型的，有些机器上的 `int` 变量的表示范围太小，甚至无法存储实际并不长的 `string` 对象。如有 16 位 `int` 型的机器上，`int` 类型变量最大只能表示 32767 个字符的 `string` 对象。而能容纳一个文件内容的 `string` 对象轻易就能超过这个数字，因此，为了避免溢出，保存一个 `string` 对象的 `size` 的最安全的方法就是使用标准库类型 `string::size_type()`。

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？

像数组一样，字符串是以 0 为索引开始的，这意味着有效的字符索引是 0 到 `size-1`，而不是 0 到 `size`。如果使用 `index <= s.size()`，C++11 之前，访问索引 `s.size()` 处的字符是未定义行为，在 C++11 以及之后将访问字符串的 ‘\0’ 终止符。（技术上讲，C++ 字符串不是 ‘\0’ 终止符，但它们包含一个 ‘\0’ 终止符以与 C 兼容）。而使用 `index != s.size()` 在所有 C++ 版本中都是安全的。

除此之外，C++ 遍历容器时，通常使用 `!=` 而非 `<=` 进行判断，是因为这种编程风格在标准库 (STL) 提供的所有容器都有效。所有 STL 容器的迭代器都定义了 `==` 和 `!=` 运算符，但是大多没有定义 `<` 运算符，考虑到代码的通用性，尽可能使用 `!=` 运算符。

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

如上所述，如果使用 `s[index] != '\0'`，在 C++11 之前，访问索引 `s.size()` 处的字符是未定义的行为，在 C++11 以及之后将访问字符串的 ‘\0’ 终止符。使用 `index != s.size()`，在所有 C++ 版本中都是安全的。

C++ 遍历容器时，通常使用迭代器而非下标，这是因为这种编程风格在标准库 (STL) 提供的所有容器都有效。考虑到代码的通用性，尽可能使用使用迭代器操作容器。

上述代码中，string的长度没有发生改变。现在：

```
string s1 = "Hel23l4l5o";
string new1;
for (string::size_type index = 0; index != s1.size(); ++index) {    // (*)
    char ch = s1[index];
    if (ch >= '0' && ch <= '9')
        new1 += 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<=s1.size();++index) 有什么问题？

如果将语句修改为index<=s1.size()，由于字符串以0为索引开始的，这意味着有效的字符索引是0到size-1，而不是0到size。如果使用index <= s.size(),C++11 之前,访问索引s.size() 处的字符是未定义行为,在C++11 以及之后将访问字符串的‘\0’终止符。

3 能否将(*)中的语句修改为for (string::size_type index=0;s1[index]!=' \0' ;++index) ？

尽量不要，如上题所述，C++标准中未规定需要\0作为字符串结尾。编译器在实现时既可以在结尾加\0，也可以不加。（因编译器不同）所以用这个条件判断会存在风险。

4.程序题

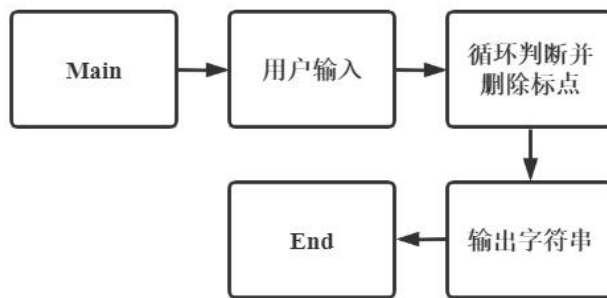
4.1 编写程序，采用**字符数组**存储字符串，从字符串中去除标点符号。要求输入到程序的字符串必须含有标点符号，输出结果则是去掉标点符号的字符串。

(0) 设计思路：

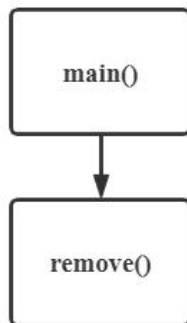
输入到程序的字符串必须含有标点符号，调用cin.getline(text, 100)函数读取用户输入的字符串，存储在字符串数组text中，将字符串数组text在子函数remove中处理，remove函数通过引用获取字符串数组，并利用 for 循环遍历该字符串。在每次循环中，都会调用 ispunct 来检查字符是否为标点符号。如果字符串时标调符号，则进行删除。返回输出结果则是去掉标点符号的字符串。

注意,C++数组作为引用，一定是写出这个数组的大小的 因为不同于指针，数组是带大小信息的。

(1) 主程序模块：



(2) 函数调用关系:



(3) 具体设计:

1、remove函数

```

//采用字符数组存储字符串，从字符串中去除标点符号
#include <iostream>
#include <string>
using namespace std;

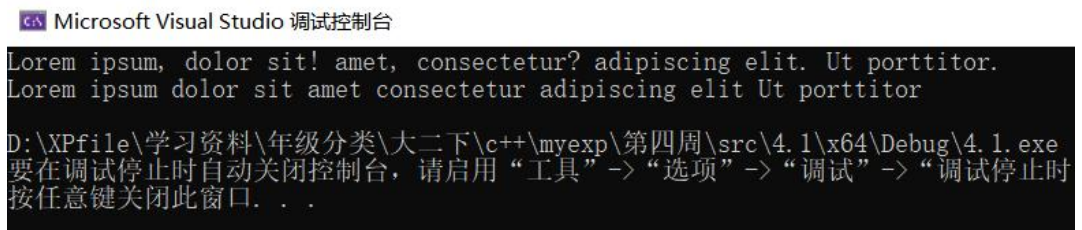
void remove(char(& s)[100]) {
    for (int i = 0, len = strlen(s); i < len; i++) {
        if (ispunct(s[i])) {
            for (int j = i; j < len; j++) {
                s[j] = s[j + 1];
            }
            len = strlen(s);
        }
    }
}
  
```

2、main函数

```
int main() {
    char text[100];
    cin.getline(text, 100); //"Lorem ipsum, dolor sit! amet, consectetur? adipiscing elit. Ut porttitor."
    remove(text);
    cout << text << endl;
    return 0;
}
```

(4) 实验结果：

将输入字符串中的标点符号删除。

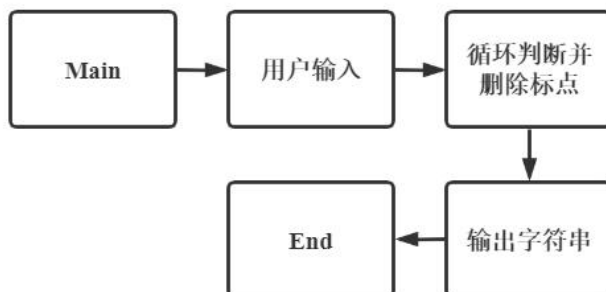


4.2 编写程序（参考第三题），从string对象中去掉标点符号。要求输入到程序的字符串必须含有标点符号，输出结果则是去掉标点符号的字符串。

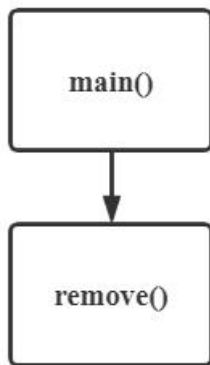
(1) 设计思路：

输入到程序的字符串必须含有标点符号，调用getline(cin,text)函数读取用户输入的一行字符串，将字符串text在子函数remove中处理，使用迭代方法从字符串中删除标点符号，remove函数通过引用获取字符串，并利用 for 循环遍历该字符串。在每次迭代中，都会调用 ispunct 来检查字符是否为标点符号。在每个匹配条件下，都会为 len 变量分配字符串的大小，因为原始字符串对象已被 erase 函数修改，并且循环需要更新计数。返回输出结果则是去掉标点符号的字符串。

(1) 主程序模块：



(2) 函数调用关系:



(3) 具体设计:

1、remove函数

```
#include <iostream>
#include <string>
using namespace std;

void remove(string& s) {
    for (int i = 0, len = s.size(); i < len; i++) {
        if (ispunct(s[i])) {
            s.erase(i--, 1);
            len = s.size();
        }
    }
}
```

2、main函数

```
int main() {
    string text;
    getline(cin, text); // "Lorem ipsum, dolor sit! amet, consectetur? adipiscing elit. Ut porttitor."
    remove(text);
    cout << text << endl;
    return 0;
}
```

(4) 实验结果:

将输入字符串中的标点符号删除。

```
Microsoft Visual Studio 调试控制台
Lorem ipsum, dolor sit! amet, consectetur? adipiscing elit. Ut porttitor.
Lorem ipsum dolor sit amet consectetur adipiscing elit Ut porttitor
```

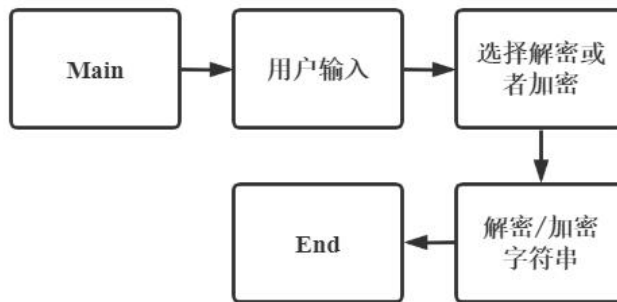
4.3 采用string编写程序，将输入的一行字符以加密的形式输出（写一个加密函数），然后将其解密（写一个解密函数），解密的字符序列与输入的正文进行比较，吻合时输出解密的正文，否则输出解密失败。

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。

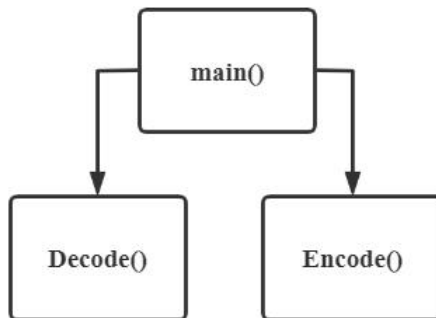
(1) 设计思路：

采用string编写程序，将输入的一行字符以加密的形式输出（通过加密函数Encode），然后将其解密（通过解密函数Decode）。

(1) 主程序模块：



(2) 函数调用关系：



(3) 具体设计：

1、编码函数Encode()

```
#include <iostream>
#include <string>
using namespace std;
```

```
void Encode(string& input) {
    for (auto &ch:input){
        ch += 8;
    }
}
```

2、解码函数Decode()

```
void Decode(string& output) {
    for (auto &ch : output) {
        ch -= 8;
    }
}
```

3、main函数

```
int main() {
    int order;
    cout << "请输入操作：编码（1） 解码（2） 退出（0）" << endl;
    cin >> order;
    while(order!=0){
        if (order == 1) {
            string input;
            cout << "请输入要编码的字符串" << endl;
            cin>>input;
            Encode(input);//编码
            cout << "编码为" <<input<< endl<<endl;
        }
        else if (order == 2) {
            string output;
            cout << "请输入要解码的字符串" << endl;
            cin>>output;
            Decode(output);//编码
            cout << "解码为" << output << endl << endl;
        }
        cout << "请输入操作：编码（1） 解码（2） 退出（0）" << endl;
        cin >>order;
    }
}
```

(4) 实验结果:

成功实现编码解码。用户输入ILOVEYOU，得到编码QTW^MaW]，将QTW^MaW]进行解码，得到ILOVEYOU。

```
Microsoft Visual Studio 调试控制台
请输入操作：编码（1） 解码（2） 退出（0）
1
请输入要编码的字符串
ILOVEYOU
编码为QTW^MaW]

请输入操作：编码（1） 解码（2） 退出（0）
2
请输入要解码的字符串
QTW^MaW]
解码为ILOVEYOU

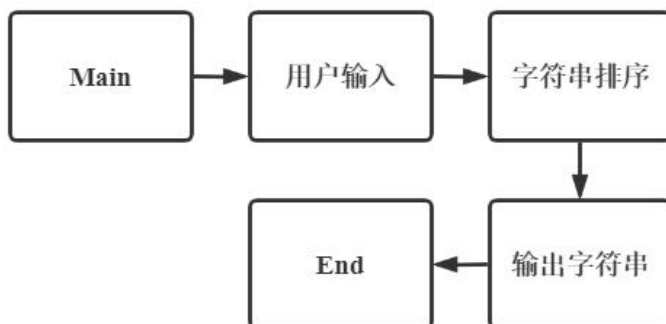
请输入操作：编码（1） 解码（2） 退出（0）
0
```

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。

(1) 设计思路:

从键盘上输入5个字符串，存储在字符串数组a中，使用sort函数将它们按由小到大的顺序排列。

(2) 主程序模块:



(3) 具体设计:

main函数:

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;

int main() {
    string a[5];
    cout << "请输入五个待排序字符串: " << endl;
    for (int i = 0; i < 5; i++) {
        cin >> a[i];
    }
    sort(a, a + 5);
    cout << "排序完成: " << endl;
    for (int i = 0; i < 5; i++) {
        cout << a[i]<<endl;
    }
    return 0;
}
```

(4) 实验结果:

成功实现字符串的排序。



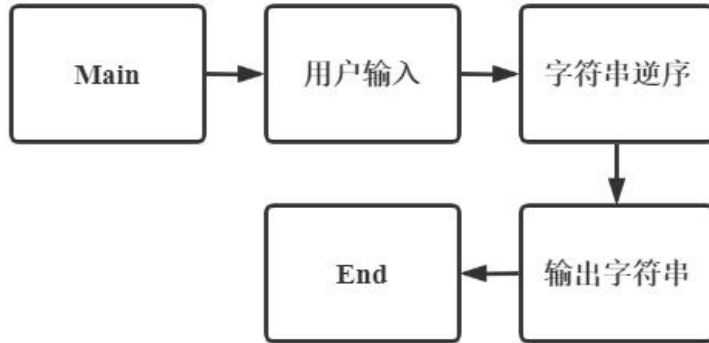
```
Microsoft Visual Studio 调试控制台
请输入五个待排序字符串:
Banana
Apple
Cherry
Strawberry
Peach
排序完成:
Apple
Banana
Cherry
Peach
Strawberry
```

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。

(1) 设计思路:

用户输入一个字符串，直接使用C++标准类库函数reverse，将其中的字符按逆序输出。

(2) 主程序模块：



(3) 具体设计：

main函数：

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int main() {
    string str;
    cin >> str;
    reverse(str.begin(), str.end());
    cout << str << endl; return 0;
}
```

(4) 实验结果：

成功实现字符串的逆序输出。

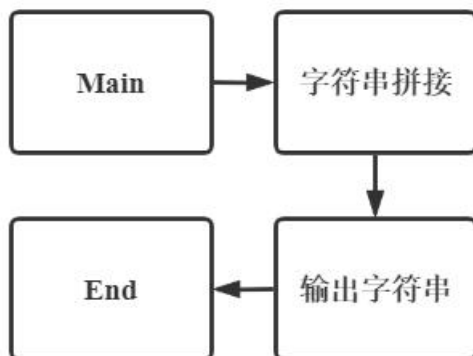


4.6 编写程序，将4个string对象分别初始化为This, is, Xiamen 和University，然后在这些字符串之间添加空格，再显示整个句子。

(1) 设计思路：

利用+操作符将4个string对象与空格拼接成一个新的字符串并输出。注意使用+时，左右操作数必须至少有一个是string类型，这是实现+运算符重载的需要。

(2) 主程序模块：



(3) 具体设计：

main函数：

```
#include <iostream>
#include <string.h>
using namespace std;

int main() {
    string str1 = "This"; string str2 = "is";
    string str3 = "Xiamen"; string str4 = "University";
    cout << str1 + ' ' + str2 + ' ' + str3 + ' ' + str4 << endl;
}
```

(4) 实验结果:

 Microsoft Visual Studio 调试控制台

This is Xiamen University

三、 附录

源程序文件项目清单: 1.1 1.2 1.3 1.4 1.5 3.1 4.1 4.2 4.3 4.5

4.6