

# 第十次作业实验报告

---

## 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：IP转换

---

### 实验目的

在网络编程时，经常需要把IP地址转换为计算机内部的整型数来处理。C++系统提供的 `atoi()` 就是实现该功能。参考该函数，编写另一个函数（如 `aton()` ），其功能是将输入的IPv4地址（字符串，例如 166.111.64.89）字符串转换为无符号整数输出。注：`aton()` 函数应返回无符号十进制整型数，然后在 `main` 函数中将该返回值转换为32位二进制数输出。

要求：

- 输入参数：`str`，输入字符串
- 返回值：转换结果，若 `str` 无法转换成整数，返回0
- 函数申明：`unsigned int aton(const char str[])`；例如：输入"166.111.64.89"，`aton` 函数返回值应为2792308825 ( $166 * 2^{24} + 111 * 2^{16} + 64 * 2^8 + 89 = 2792308825$ )，最后在 `main` 中转 为10100110011011110100000001011001 输出。

字符数组的应用

### 实验内容

## 分析

遍历一个字符数组识别 '.' 和 '\0' 便可以将一个合法的IPv4地址的四个部分分开，进而转换成所需的无符号整型数。

再采用数学上的短除法获得二进制输出即可。

考虑到短除法操作获得的二进制数是逆序的，只要逆序输出即可得到正常的二进制数。

## 代码

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  unsigned int aton(const char str[]);
6  bool bcheckStr(const char str[]); //检查输入的是否只有数字和'.'
7
8  int main() {
9      char str[16];
10     unsigned int temp;
11     int result[32];
12     cout << "please input an IP (according to IPv4)\n";
13     cin >> str;
14     if (bcheckStr(str)) {
15         temp = aton(str);
16         int i = 0;
17         //用数组存储转换为2进制时的余数
18         do {
19             result[i] = temp % 2;
20             temp /= 2;
21             ++i;
22         } while (temp != 0);
23         //倒序输出，转换为标准形式的二进制数
24         for (--i; i >= 0; --i) {
25             cout << result[i];
26         }
27     }
28     else {
29         cout << "0\n";
30     }
31     return 0;
32 }
33
34 unsigned int aton(const char str[]) {
35     unsigned int result;
36     int num[4] = { 0 };
37     for (int i = 0, j = 0; ++i) {
38         if (str[i] == '\0') break;
39         else if (str[i] < 46 || str[i] == 47 || str[i] > 57) {
40
41         }
42         else if (str[i] == '.') {
43             ++j;
44         }
```

```

45         else {
46             num[j] = 10 * num[j] + (str[i] - 48);
47             if (num[j] > 255) {
48                 return 0;
49             }
50         }
51     }
52     result = 256 * 256 * 256 * num[0] + 256 * 256 * num[1] + 256 * num[2] + num[3];
53     return result;
54 }
55
56 bool bCheckStr(const char str[]) {
57     for (int i = 0;; ++i) {
58         if (str[i] == '\0') break;
59         else if (str[i] < 46 || str[i] == 47 || str[i] > 57) {
60             return 0;
61         }
62     }
63     return 1;
64 }

```

## 结果

```

Microsoft Visual Studio 调试控制台
please input an IP (according to IPv4)
166.111.64.89
10100110011011110100000001011001
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 21728) 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 an IP (according to IPv4)
1666.46.46.8
0
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 31228) 已退出，返回代码为: 0。
按任意键关闭此窗口...

```

## 分析总结

采用上述代码输出二进制的方式不利于保存和再调用，应有更好方式，还有改进空间。

## 第二题：元音翻转

### 实验目的

对于一个由a-z（小写）组成的字符串，将其中的元音反转，而辅音不反转。如对于字符串"hello"，替换后的字符为"holle"。（注：元音为a, e, i, o, u）要求：输入一个由a-z（小写）组成的字符串，输出反转后的字符串。例如：  提示：本题通过字符串数组和循环分支能够解决。参考算法：首先遍历字符串，将其中所有的元音存入字符串数组中，之后再次遍历字符串，将其中所有的元音反序替换。

## 实验内容

### 分析

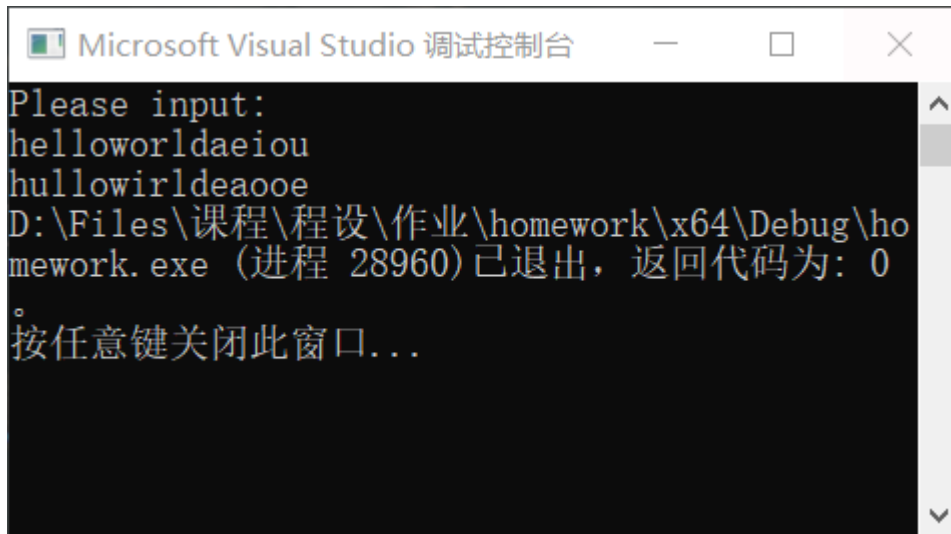
需要定义一个足够长的字符数组用于存储输入的字符串，一个足够长的字符数组用于存储元音和它们所在的位置。

先遍历一次输入的字符串，记录所有的元音和它们的位置信息，然后通过循环更改字符串相应位置的元音即可。

### 代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char str[1024] = { 0 }, vowel[1024][2];
6      int i = 0, j = 0;
7      cout << "Please input:\n";
8      while ((str[i] = getchar()) != '\n') {
9          if (str[i] < 97 || str[i]>122) {
10             cout << "Input Error";
11             return 0;
12         }
13         if (str[i] == 97 || str[i] == 101 || str[i] == 105 || str[i] == 111 ||
str[i] == 117) {
14             vowel[j][0] = i;
15             vowel[j++][1] = str[i];
16         }
17         i++;
18     }
19
20     for (int k = 0; k < j; ++k) {
21         str[vowel[k][0]] = vowel[j - k - 1][1];
22     }
23     for (int k = 0; k < i; ++k) {
24         cout << str[k];
25     }
26
27     return 0;
28 }
```

### 结果



## 分析总结

采用字符数组的一个问题是长度必须初始化，可能可以考虑修改为 `string` 类，这样或许就不需要指明长度。

## 第三题：字母排序

### 实验目的

输入一段由英文字母(区分大小写)组成的字符串，将其按ASCII码从大到小顺序输出。要求：输入一个由英文字母（区分大小写）组成的字符串，输出符合要求的字符串。例如：input: aggregate output: trgggeaa  
参考算法：本题通过字符串数组、循环分支能够求解。算法可以为冒泡排序等排序算法。若同学不会冒泡排序，则根据出现的字符仅在英文字母范围，遍历法寻找字符串中从z到A的字母，也可以求得。

仍然是字符数组的练习

### 实验内容

#### 分析

需要一个足够长的用于存储字母的字符数组，和一个将字母重排的方程。

根据ASCII嘛进行排序，所以任何一个排序算法都可以，以下采用快速排序。

#### 代码

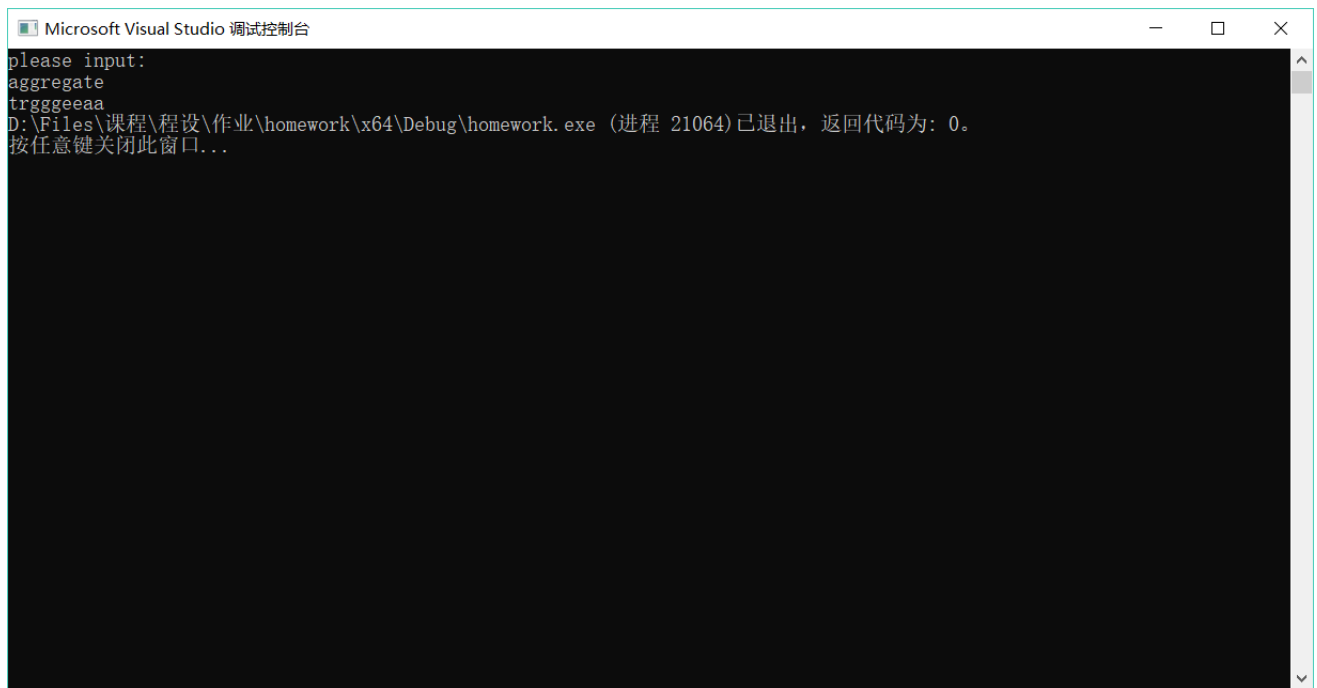
```
1  #include <iostream>
2  using namespace std;
3
4  void sort(char cStr[], int start, int end);
5
6  int main() {
7      char str[1024];
8
9      cout << "please input:\n";
10     int i = 0;
```

```

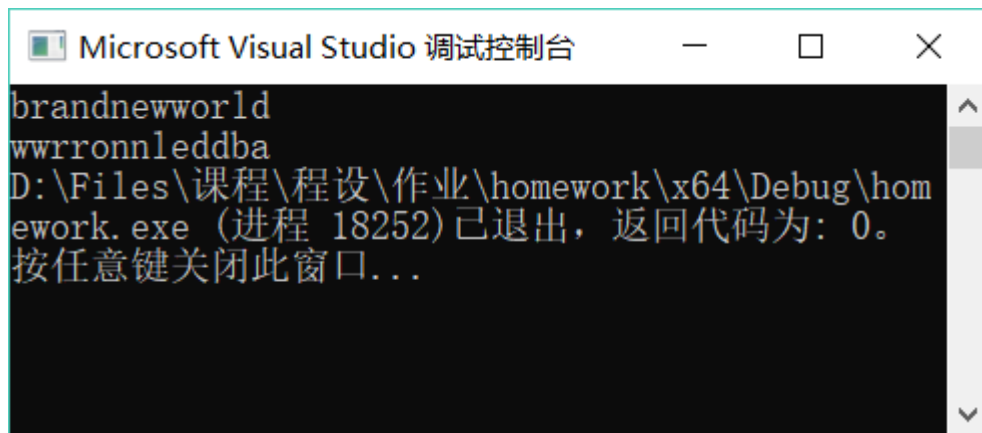
11     while((str[i]=getchar())!='\n'){
12         if (str[i] < 65 || str[i]>122 || (str[i] > 90 && str[i] < 97)) {
13             cout << "Input Error";
14             return 0;
15         }
16         i++;
17     }
18
19     sort(str, 0, i - 1);
20     for (int j = 0; j < i; ++j) {
21         cout << str[j];
22     }
23     return 0;
24 }
25
26 void sort(char cStr[], int start, int end) {
27     if (start >= end) return;
28     int i = start, j = end, temp = cStr[j];
29     while (i < j) {
30         while (i < j && cStr[i] >= temp) {
31             ++i;
32         }
33         cStr[j] = cStr[i];
34         while (i < j && cStr[j] <= temp) {
35             --j;
36         }
37         cStr[i] = cStr[j];
38     }
39     cStr[j] = temp;
40     sort(cStr, start, i - 1);
41     sort(cStr, i + 1, end);
42 }

```

## 结果



```
Microsoft Visual Studio 调试控制台
please input:
aggregate
trgggeaa
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 21064) 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
brandnewworld
wwrronnleddba
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 18252) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

## 分析总结

快速排序的思路很巧，在一般情形下相比冒泡排序，可以大幅减少无用的操作。