

第十二次作业实验报告

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：字符串排序

实验目的

定义一个指向字符串的指针数组，用一个函数完成N个不等长字符串的输入，使得指针数组元素依次指向每一个输入的字符串。设计一个完成N个字符串按升序的排序函数（在排序过程中，要求只交换指向字符串的指针，不交换字符串）。在主函数中实现对排序后的字符串的输出。假设已知字符串的最大为80字节，根据实际输入的字符串长度来分配存储空间。

说明：自行设计输入样例并给出结果截图，要求在实验报告中给出算法思路。

练习使用指针数组和 `malloc` 函数

实验内容

分析

题目要求实现一个任意长度的指针数组，每个元素又指向不等长度的字符串，示意如下：

```
1 | p[0]="four"
2 | p[1]="scores"
3 | ...
```

可以通过构建二重指针，先指向n个指针，再由这n个指针指向n个不等长度字符串。

用代码实现为：

```
char **pp = (char**)malloc(n * sizeof(char*));
```

或者

```
char **pp = new char*[n];
```

之后再给n个指针分配空间即可

代码

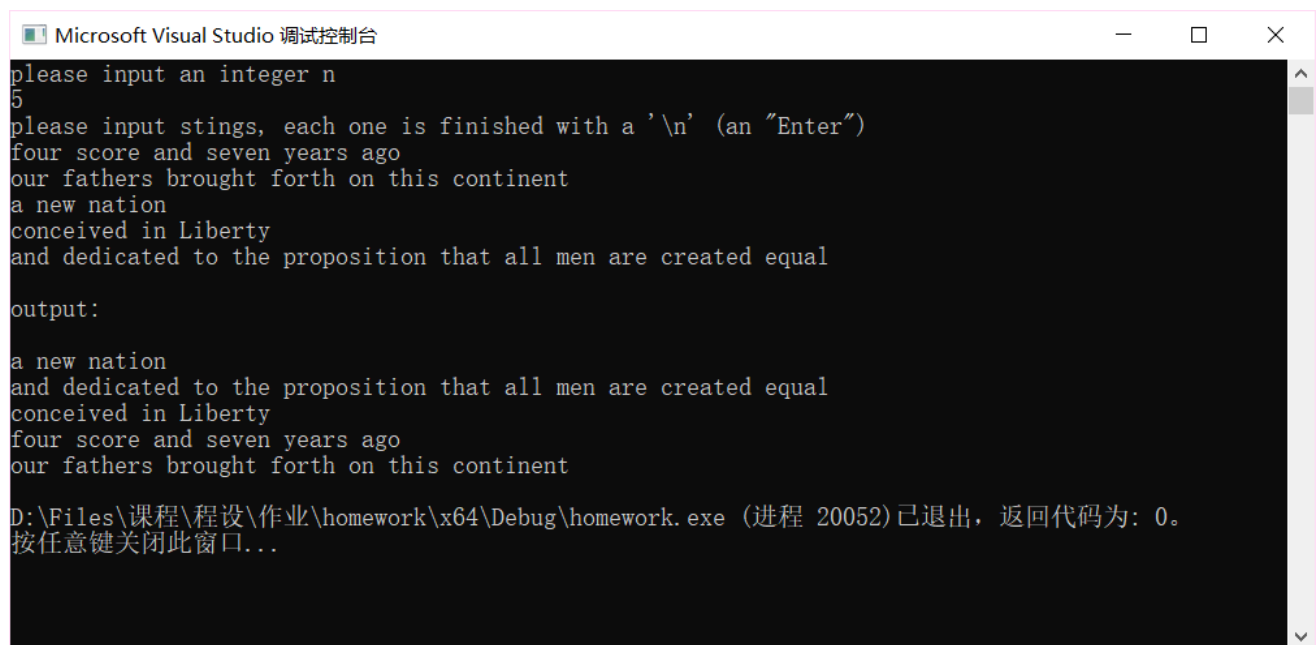
```
1  #include <iostream>
2  #include <malloc.h>
3  using namespace std;
4
5  void input(char **p, int n);
6  void sort(char **p, int n);
7
8  int main() {
9      int n;
10     cout << "please input an integer n\n";
11     cin >> n;
12     //char **pp = new char*[n];
13     char **pp = (char**)malloc(n * sizeof(char*)); //申请n个存字符指针的空间
14     cout << "please input stings, each one is finished with a '\\n' (an
15     \"Enter\\")\\n";
16     input(pp, n);
17     sort(pp, n);
18     for (int i = 0; i < n; ++i) {
19         cout << *(pp + i) << endl;
20         free(*(pp + i));
21     }
22     free(pp); //释放内存
23     return 0;
24 }
25
26 void input(char **pstr, int n) {
27     char str[80];
28     char *p = &str[0];
29     for (int i = 0; i < n; ++i) {
30         cin.ignore();
31         cin.get(str, 81, '\\n'); //输入整行的字符，读取空格
32         //cin >> str; 空格就中断
33         /*(pstr + i) = new char[strlen(str) + 1];
34         *(pstr + i) = (char*)malloc((strlen(str) + 1) * sizeof(char)); //申请相应长度的
35         空间存放字符串
36         int j = 0;
37         for (j; j < strlen(str); ++j) {
38             (*(pstr + i) + j) = *(p + j);
39         }
39     }
```

```

38     *(*pstr + i) + j) = '\0';
39     }
40 }
41
42 void sort(char **p, int n) {
43     //冒泡排序
44     for (int i = 0; i < n - 1; ++i) {
45
46         for (int j = 0; j < n - 1 - i; ++j) {
47             if (strcmp(*(p + j), *(p + j + 1)) > 0) {
48                 char *temp = NULL;
49                 temp = *(p + j);
50                 *(p + j) = *(p + j + 1);
51                 *(p + j + 1) = temp;
52             }
53         }
54     }
55 }

```

结果



```

Microsoft Visual Studio 调试控制台
please input an integer n
5
please input strings, each one is finished with a '\n' (an "Enter")
four score and seven years ago
our fathers brought forth on this continent
a new nation
conceived in Liberty
and dedicated to the proposition that all men are created equal

output:
a new nation
and dedicated to the proposition that all men are created equal
conceived in Liberty
four score and seven years ago
our fathers brought forth on this continent

D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 20052) 已退出, 返回代码为: 0。
按任意键关闭此窗口...

```

分析总结

字符串排序采用 `strcmp()` 函数，可以十分容易地返回以ASCII码为基准的“字典序”，先是大写字母，再是小写字母。

一个巧妙的存储多个不同长度字符串的方式是构建二重指针，这样交换存放的n个指针时不会影响到字符串存放的地址，且实现了不浪费空间。

第二题：方阵对角和

实验目的

编写程序，求一个 $N \times N$ 方阵的第 i 条对角线的元素之和。其中，方阵的第 i 条对角线定义如下：

$N \times N$ 方阵 A 的第 n 条对角线之和为 $S(n) = \sum_{i=1}^N a_{i \otimes n}$ ，其中，

$$i \otimes n = \begin{cases} i + n, & i + n \leq N \\ (i + n) \% N, & i + n > N \end{cases}$$

方阵

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$$

其中

$a_{11}, a_{22}, \cdots, a_{NN}$ 称为第 0 对角线

$a_{12}, a_{23}, \cdots, a_{N-1 N}, a_{N1}$ 称为第 1 对角线

依次类推

要求：

- 方阵大小固定为 10×10
- 方阵元素为 $A = [[0, 1, \dots, 9], [10, 11, \dots, 19], \dots, [90, 91, \dots, 99]]$
- 使用指针编程

说明：在 `main` 函数中遍历给出第 $i(0, 1, \dots, 9)$ 对角角线的计算结果，要求在实验报告中给出算法思路。

练习使用二维数组的指针

实验内容

分析

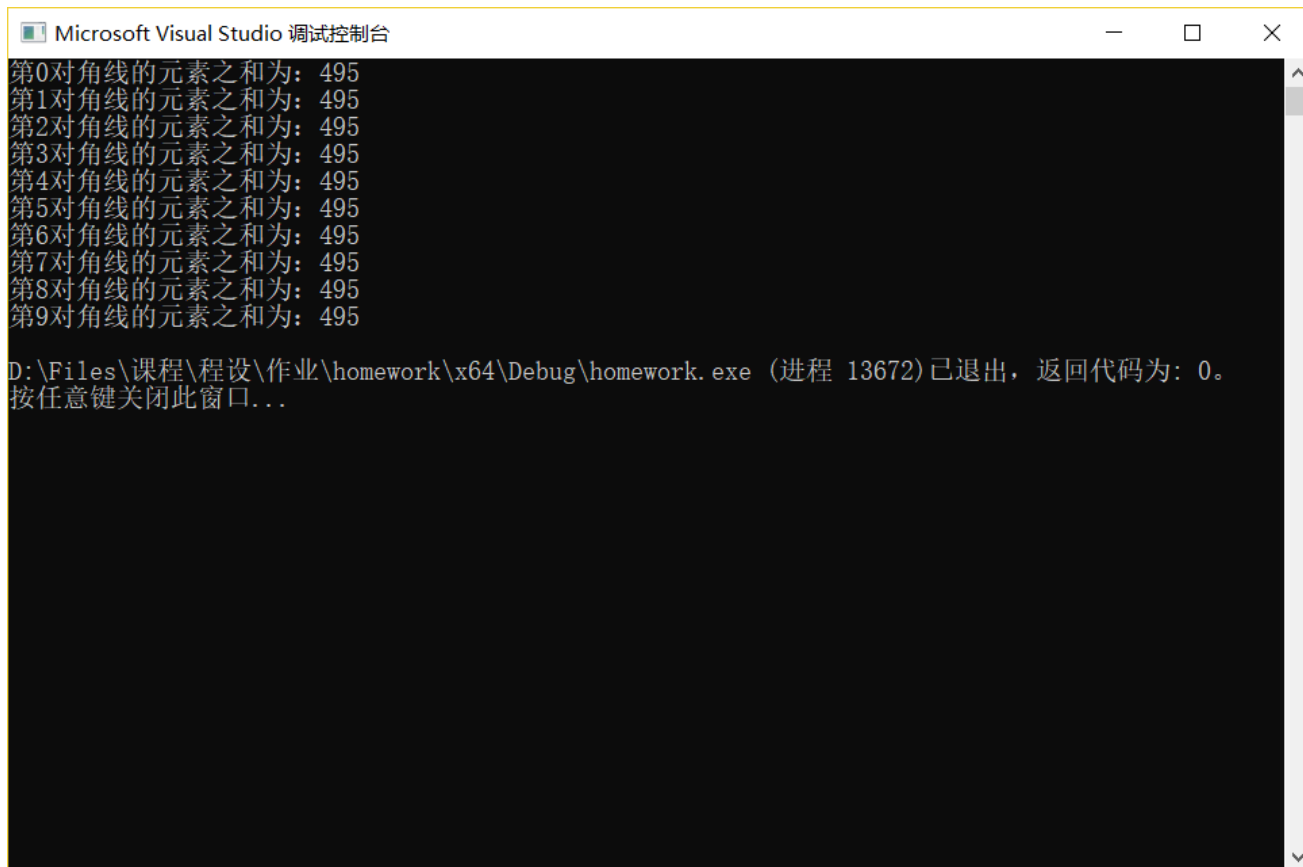
定义行指针变量 `(*p)[10]`，然后类似二维数组的进行操作即可

代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int A[10][10];
6      int (*p)[10] = A;
7
8      for (int i = 0; i < 10; ++i) {
9          for (int j = 0; j < 10; ++j) {
10             (*(p + i) + j) = 10 * i + j;
11         }
12     }
13
14     for (int i = 0; i < 10; ++i) {
15         int sum = 0;
16         for (int j = 0; j < 10; ++j) {
17             if (i + j >= 10) sum += (*(p + j) + (i + j) % 10);
18             else sum += (*(p + j) + i + j);
19         }
20     }
21 }
```

```
19     }
20     cout << "第" << i << "对角线的元素之和为: " << sum << endl;
21 }
22
23 return 0;
24 }
```

结果



```
Microsoft Visual Studio 调试控制台
第0对角线的元素之和为: 495
第1对角线的元素之和为: 495
第2对角线的元素之和为: 495
第3对角线的元素之和为: 495
第4对角线的元素之和为: 495
第5对角线的元素之和为: 495
第6对角线的元素之和为: 495
第7对角线的元素之和为: 495
第8对角线的元素之和为: 495
第9对角线的元素之和为: 495
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 13672) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

第三题：Emoji

实验目的

通常一个基于转义符的emoji表情输入由以下三部分构成：转义符 + 表情名称 + 终止符

以新浪微博为例，当微博正文读取到一个转义符“\”时，它与终止符“\”之间的文字将作为表情名称在表情库中进行搜索，如果存在匹配表情，则输出显示。注意，如果在一段语句中存在多个转义符和一个终止符，那么以离终止符最近的一个转义符作为表情的起始标志。

要求：编写程序，在输入的一句文字中，输出被转义符括起来的表情名称文字。输入：首先输入转义符，然后输入终止符，二者均为临时指定的任意半角标点符号，随后输入一行任意的文字，由英文、数字和符号组成。文字最长不超过140个字符。输出：被转义符括起来的表情名称文字，如有多个表情名称，则分行输出。

输入输出样例例如下：

```

1  start: *
2  end: #
3  input:
4  Time for lunch. *greedy# Hope a big meal.
5
6  output:
7  greedy
8
9  // 多个转义符以距离终止符最近的一个为准:
10 input: *happy*smile#
11 output:
12 smile
13
14 start: [
15 end: ]
16 input:
17 Emm... You use [Grin] instead of [Smile] when you are really happy in wechat.
18 output:
19 Grin
20 Smile

```

说明：自行设计输入测试样例（需要涵盖上述两个样例）并给出结果截图，要求在实验报告中给出算法思路。

练习使用指针寻找字符串中的特定字符

实验内容

分析

用两个指针，第一个遍历字符串寻找转义符，找到后用第二个指针从其后遍历寻找终止符，遍历过程中如遇到转义符，改变第一个指针即可。

代码

```

1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main() {
6      char str[141]; //140+'\\0'
7      char start, end, *pstart, *pend;
8      bool found = false;
9      cout << "please input:\\nstart:";
10     cin >> start;
11     cout << "end:";
12     cin >> end;
13     cout << "input:\\n";
14
15     cin.ignore(); //忽略前面输入的换行符
16     cin.get(str, 141, '\\n'); //输入带空格的一句话
17     pstart = &str[0];
18     cout << "output:\\n";

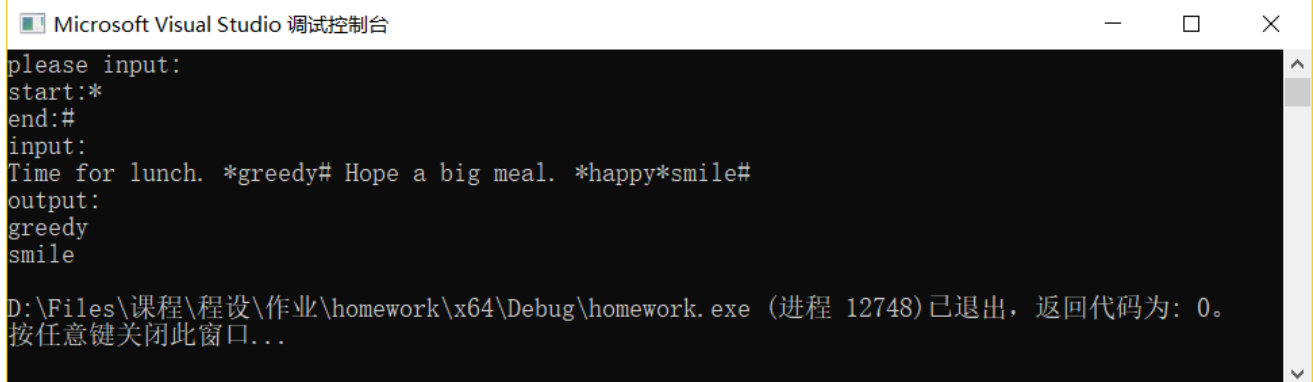
```

```

19 //寻找表情符号
20 while (pstart < &str[strlen(str) - 1]) {
21     if (*pstart == start) {
22         for (pend = pstart + 1; pend < &str[strlen(str) - 1]; pend++) {
23             if (*pend == start)
24                 pstart = pend;
25             if (*pend == end) {
26                 found = true;
27                 //输出“表情”文字
28                 for (pstart += 1; pstart < pend; pstart++) {
29                     cout << *pstart;
30                 }
31                 cout << endl;
32                 pstart = pend + 1; //检索这个表情之后的转义符
33                 break;
34             }
35         }
36     }
37     ++pstart;
38 }
39 if (found == false) cout << "FOUND NOTHING NO EMOJI\n";
40
41 return 0;
42 }

```

结果



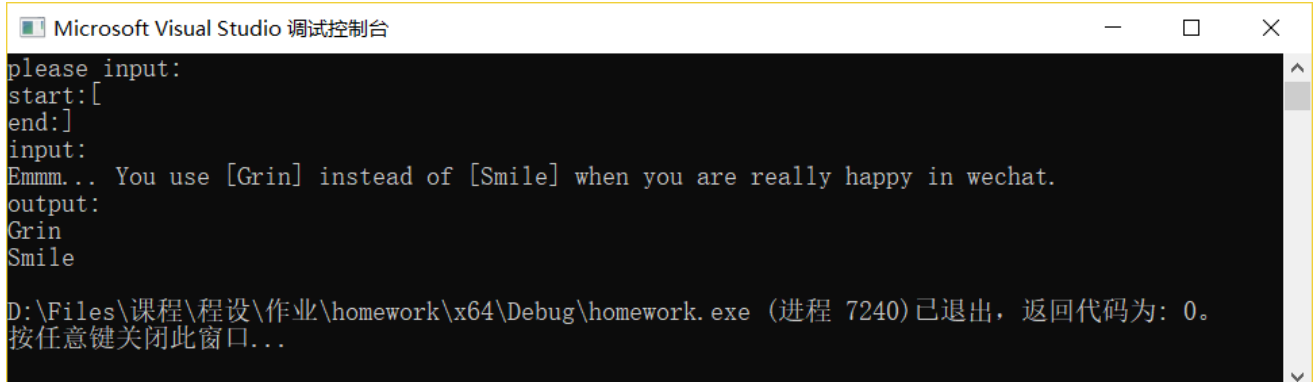
Microsoft Visual Studio 调试控制台

```

please input:
start:*
end:#
input:
Time for lunch. *greedy# Hope a big meal. *happy*smile#
output:
greedy
smile

D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 12748) 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:
start:[
end:]
input:
Emmm... You use [Grin] instead of [Smile] when you are really happy in wechat.
output:
Grin
Smile

D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 7240) 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

start:[
end:]
input:
tes[ttes[ttest]tes]t
output:
ttest

D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进
程 8692) 已退出，返回代码为：0。
按任意键关闭此窗口...
```

分析总结

记住更改指向转义符的指针指向正确的位置即可

选做题1：Zig-Zag扫描

实验内容

分析

从上到下，从左到右，依照二维数组给每个“格子”标号为 ij ，当 $i + j$ 为2的倍数时，向斜上方扫描；为奇数时向斜下方扫描

可得流程见代码

对于输入矩阵数据，可类似第一题进行空间的申请

代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n;
6      cout << "please input an integer bigger than 0:\n";
7      cin >> n;
8      int **matrix = (int**)malloc(n * sizeof(int*));
9      for (int i = 0; i < n; ++i) {
10         *(matrix + i) = (int*)malloc(n * sizeof(int));
11         for (int j = 0; j < n; ++j) {
12             cin >> (*(matrix + i) + j);
13         }
14     }
15 }
```



```

16     int i = 0, j = 0;
17     while (1) {
18         cout << (*(matrix + i) + j) << " ";
19         if (i == n - 1 && j == n - 1) break;
20         if ((i + j) % 2 == 0) {
21             if (i > 0) {
22                 --i;
23                 if (j < n - 1) ++j;
24                 else i += 2;
25             }
26             else {
27                 if (j < n - 1) ++j;
28                 else ++i;
29             }
30         }
31         else {
32             if (j > 0) {
33                 --j;
34                 if (i < n - 1) ++i;
35                 else j += 2;
36             }
37             else {
38                 if (i < n - 1) ++i;
39                 else ++j;
40             }
41         }
42     }
43
44     for (int i = 0; i < n; ++i) {
45         free(*(matrix + i));
46     }
47     free(matrix);
48
49     return 0;
50 }

```

结果

```

选择Microsoft Visual Studio 调试控制台
please input an integer bigger than 0:
5
please input entries in a matrix which is n*n:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
1 2 6 11 7 3 4 8 12 16 21 17 13 9 5 10 14 18 22 23 19 15 20 24 25
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 21632) 已退出，返回代码为：0。
按任意键关闭此窗口...

```

选做题2

答案: ttestFFINALEEXAM