最长公共子序列（LCS）

SA20225085 朱志儒

## 实验内容

编程实现最长公共子序列（LCS）算法，并理解其核心思想。

给定两个字符串text1和text2，返回这两个字符串的最长公共子序列的长度。

一个字符串的**子序列**是指这样一个新的字符串：它是由原字符串在不改变字符的相对顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。例如，"ace"是"abcde"的子序列，但"aec"不是"abcde"的子序列。两个字符串的「公共子序列」是这两个字符串所共同拥有的子序列。

若这两个字符串没有公共子序列，则返回 0。

程序输入：

由控制台输入两个字符串text1，text2。

其中：

1 <= text1.length <= 1000

1 <= text2.length <= 1000

输入的字符串只含有小写英文字符（a~z）

程序输出：

控制台打印LCS的长度及相应的序列，如不存在公共子序列则返回0。

示例1：

输入：text1 = “abcde”，text2 = “ace”

输出：LCS：“abc”，长度：3

示例2：

输入：text1 = “abc”，text2 = “def”

输出：0

## 实验目的

通过本实验加深对最长公共子序列（LCS）算法的理解和运用。

## 算法设计思路

1. **LCS最优解的结构特征：**

设序列和，是X和Y的任意一个LCS，则

1. 若且是和的一个LCS；
2. 若Z是和Y的一个LCS；
3. 若 Z是X和的一个LCS；

由此可见，2个序列的最长公共子序列可由（1）（2）（3）算出，（2）（3）的解时对应子问题的最优解。

1. **子问题的递归解：**
2. **数据结构设计：**

//存放最优解值，计算时行优先

//解矩阵，存放构造最优解信息

当构造解时，从b[m, n]出发，上溯至i = 0或j = 0为止，上溯过程中，当b[i,j]包含“↖”时打印出。

1. **伪代码：**

## 源码+注释

1. **void** LCS\_length(**int**\*\* c, **int**\*\* b, string x, string y) {
2. **int** m = x.length() - 1, n = y.length() - 1;
3. **for** (**int** i = 0; i <= m; ++i)
4. c[i][0] = 0;
5. **for** (**int** j = 0; j <= n; ++j)
6. c[0][j] = 0;
7. **for** (**int** i = 1; i <= m; ++i) {
8. **for** (**int** j = 1; j <= n; ++j) {
9. **if** (x[i] == y[j]) {
10. c[i][j] = c[i - 1][j - 1] + 1;
11. b[i][j] = 0;    //指向↖
12. }
13. **else** **if** (c[i - 1][j] >= c[i][j - 1]) {
14. c[i][j] = c[i - 1][j];
15. b[i][j] = 1;    //指向↑
16. }
17. **else** {
18. c[i][j] = c[i][j - 1];
19. b[i][j] = 2;    //指向←
20. }
21. }
22. }
23. }
25. **void** get\_LCS(**int**\*\* b, string x, **int** i, **int** j, string& result) {
26. **if** (i == 0 || j == 0)
27. **return**;
28. **if** (b[i][j] == 0) {         //指向↖
29. get\_LCS(b, x, i - 1, j - 1, result);
30. result.push\_back(x[i]);
31. }
32. **else** **if** (b[i][j] == 1) {    //指向↑
33. get\_LCS(b, x, i - 1, j, result);
34. }
35. **else** {                      //指向←
36. get\_LCS(b, x, i, j - 1, result);
37. }
38. }
40. **int** main() {
41. cout << "请输入text1，text2：";
42. string str1, str2;
43. cin >> str1 >> str2;
44. str1 = ' ' + str1;
45. str2 = ' ' + str2;
46. **int** m = str1.length() - 1, n = str2.length() - 1;
47. //初始化二维数组c
48. **int**\*\* c = **new** **int**\* [m + 1];
49. **for** (**int** i = 0; i <= m; ++i)
50. c[i] = **new** **int**[n + 1];
51. //初始化二维数组b
52. **int**\*\* b = **new** **int**\* [m + 1];
53. **for** (**int** i = 0; i <= m; ++i)
54. b[i] = **new** **int**[n + 1];
55. LCS\_length(c, b, str1, str2);
56. **int** len = c[m][n];
57. **if** (len > 0) {
58. string result;
59. get\_LCS(b, str1, m, n, result);
60. cout << "LCS：" << result << ", ";
61. }
62. cout << "长度：" << len << endl;
63. **return** 0;
64. }

## 算法测试结果

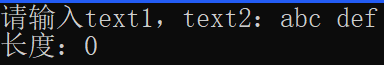
输入：text1 = “abcde”，text2 = “ace”

输出：



输入：text1 = “abc”，text2 = “def”

输出：



## 实验过程中遇到的困难及收获

在本次实验中，我遇到了一个小小的问题：伪代码中的下标是从1开始的，而我实现的C++代码的下标是从0开始的，这导致我在测试程序正确性的时候出现了问题。其实解决方案很简单，只需把声明的数组的长度加1，下标为0的位置不使用，实现算法时下标从1开始。