字符串程序的综合分析——子串的数组版简单检索

2022 年 8 月 16 日

1. 阅读程序，回答相应问题。

```
#include <stdio.h>
#include <string.h>

/*子串检索*/

int main()
{
   char s1[1024], s2[1024];
   printf("Input a string to be searched:\n"); // 被搜查的字符串
   gets(s1);
   printf("Input a string to be located:\n"); // 待检索的字符串
   gets(s2);
   len1 = strlen(s1); len2 = strlen(s2);
   for(int i = 0; i < len1 - len2 + 1; i++) // 恰好检查完所有可能的子串
   {
      int j;
      for(j = 0; j < len2; j++) // 恰好扫描到被检索子串的末端
      {
         // s1[i + j]表示s1串中相应位置的字符，s2[j]表示s2串中相应位置的字符
         if(s1[i + j] != s2[j]) // 失配
         {
            break; // 本轮无需继续检查
         }
      }
      if(j == strlen(s2)) // 扫完整个s2，表示已经找到
      {
         printf("the first occurence of %s is at %d\n", s2, i);
```

```
28          return 0;
29        }
30      }
31      printf("%s not found\n", s2);
32      return 0;
33  }
```

(a) 试分别就以下两种情况，分析以上程序的运行，并写出输出结果。

i. 输入 `s1` 为 “The goose is good.”，输入 `s2` 为 “good”；

ii. 输入 `s1` 为 “Then she got a sheep.”，输入 `s2` 为 “shed”。

(b) 假设用户输入 `s1` 为 “The goose is good.”，输入 `s2` 为 “good”。计算第 14 行的 `i++`、第 17 行的 `j++` 和第 20 行的 `s1[i + j] != s2[j]` 分别被执行或判断的次数。

(c) 假设用户输入 `s1` 为 “Then she got a sheep.”，输入 `s2` 为 “shed”。计算第 14 行的 `i++`、第 17 行的 `j++` 和第 20 行的 `s1[i + j] != s2[j]` 分别被执行或判断的次数。