

資料結構實習

11/03 作業報告

Stack 實作 (e, s) segment

班級：資訊二甲

學號：D1109023

姓名：楊孟憲

Contents

1	引言	3
2	題目敘述	4
3	作法	5
4	執行結果	7
5	心得與討論	8

1 引言

Stack 是一個只有一個開口的單向輸入輸出的資料結構，在日常生活中可以使用該資料結構維護許多演算法。插入以及拿取都只需要 $O(1)$ 的時間。以下是利用 LinkedList 實作 Stack 的方法。(以下是使用 C 實作 Stack 的 push/pop)

```
1  /// stack
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  typedef struct Node * NodePtr;
6
7  typedef struct Node {
8      int val;
9      NodePtr next; // Node *
10 } Node;
11
12 void push(NodePtr *head, int val) {
13     NodePtr newNode = (NodePtr)malloc(sizeof(Node));
14     newNode -> val = val;
15     newNode -> next = (*head);
16     *head = newNode;
17     return;
18 }
19
20 void pop(NodePtr *head) {
21     if(*head == NULL) {
22         printf("The Stack is Empty!\n");
23     } else {
24         int topValue = (*head) -> val;
```

```

25     printf("The top value of the stack: %d\n", topValue);
26     NodePtr cur = *head;
27     *head = (*head) -> next;
28     free(cur);
29 }
30 return;
31 }
32
33 int main() {
34     NodePtr head = NULL;
35
36     push(&head, 20);
37     push(&head, 30);
38     pop(&head);
39
40     while(head != NULL) {
41         printf("Val: %d, nextPointer: %p\n", head -> val, head -> next);
42         head = head -> next;
43     }
44     return 0;
45 }

```

2 題目敘述

題意說明：如果在一個字串當中其頭一個字母為 E，最後一個字母為 S，而兩個字母中間不包含任何 E 或 S 字母的話，則稱為 ES 字串。請利用堆疊（Stacks）的原理，撰寫出一個程式，從檔案（input.txt）讀入一段文章或是字串，然後消去所有可能的 ES 字串，使得消去後的字串輸出不包含任何 ES 字串。

3 作法

使用 C++ 的 STL Stack 實作。遍歷字串的每一個字元，當前字元為's' 時，判斷 stack 是否為空並且如果前面有'e' 的話 (cnt > 0)，就不斷地把先前的字元從 stack 拿出來，直到 now == 'e'，並且將 cnt-1，這能夠確保之後的 e, s 區間能被正確的移除。否則，就把's' 放進 stack 裡。

如果當前字元不為's' 的話，就放進 stack 裡，並且判斷如果當前字元為'e' 的話，就 cnt++;

範例程式碼

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define int long long
5 #define pb push_back
6 stack<char> st;
7
8 string get_ans(string ms) {
9     int cnt = 0;
10    for(int i = 0; i < ms.size(); i++) {
11        if(ms[i] == 's') {
12            if(cnt > 0) {
13                while(!st.empty()) {
14                    char now = st.top();
15                    st.pop();
16                    if(now == 'e') {
17                        cnt--;
18                        break;
19                    }
20                }
21            }
22            st.push('s');
23        }
24        else if(ms[i] == 'e') {
25            cnt++;
26        }
27    }
28    return ms;
29 }
```

```

19         }
20     }
21     } else {
22         st.push(ms[i]);
23     }
24     } else {
25         if(ms[i] == 'e') cnt++;
26         st.push(ms[i]);
27     }
28 }
29
30 string ans = "";
31 while(!st.empty()) {
32     ans = st.top() + ans;
33     st.pop();
34 }
35 return ans;
36 }
37
38 void solve() {
39     string ss, ms = "";
40     ifstream ifs("input.txt", ifstream::in);
41
42     if(!ifs.is_open()) {
43         cout << "error\n";
44         return;
45     }
46
47     while(getline(ifs, ss)) {
48         if(ss == "") {
49             cout << get_ans(ms) << "\n";
50             ms = "";
51         }

```

```

52     ms += ss;
53     ms += "\n";
54 }
55 cout << get_ans(ms) << "\n";
56 return;
57 }
58
59 signed main() {
60     solve();
61     return 0;
62 }

```

4 執行結果

輸入輸出結果：

1. 輸入：

```

1 And then there' s the journey itself - retiring to bed as you
  clatter out of a big city and waking up in a new city, or
  even a new country, can create memories to last a lifetime.
2
3 At least that' s the theory - and why the new wave of night
  trains are being touted as one way to replace short or even
  medium-haul flights across Europe and the US.
4
5 So how' s that going?
6
7 Even before their renaissance, night trains could be a
  pleasant, memorable and sometimes economic way to cover long
  distances - but luck has always been a big factor.

```

2. 輸出：

```
1 And then ther the journelf - retiring to b you clatter out of
  a big city and waking up in a new city, or even a new
  country, can create mt a lifetime.
2
3
4 At lt that' s the theory - and why the new wav are being tout
  one way to replachort or Europe and the US.
5
6
7 So how' s that going?
8
9
10 Even before thanc could be a plant, memorablom been a big
   factor.
```

5 心得與討論

這次的作業利用 Stack 的原理來完成，是一個常見的模板題，實作起來沒有什麼問題，這讓我更加了解 Stack 的運作原理。