

資結作業 HW4&5

班級:資訊三丙

學號:D0745765

姓名:楊耀寧

中置轉前置

題目定義

我們先創造 2 個字串陣列，一個是 `infix`、另一個是 `prefix`。大小皆為 `MAX`(預設為 100)，分別儲存輸入的中置表達式，以及轉換後的前置表達式。我們再創造一個堆疊，大小也為 `MAX`，用來儲存運算子，並使用 `top` 變數來記錄目前頂端的位置。

演算法

我們從中置式的尾端讀到前端來進行轉換。若遇到運算元直接存進 `prefix`。若遇到運算子，則讓她跟 `stack` 中的運算子做比較，要是 `stack` 中的運算值優先權大於要進入的運算子優先權，則把 `stack` 中的運算子拿出來，重複比對直到要進入的運算子優先權不在小於 `stack` 中的，則把要進去的運算子放進 `stack` 中。若遇到左括號，則將 `stack` 中的內容拿出，直到遇到又括號時停下來，並將又括號也拿出來，但不輸出。

原始程式碼

```
C:\Users\User\Downloads\中置轉前置.c - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(R) 工具(T) AStyle 視窗(W) 求助(H)
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #define MAX 100
5
6 int *stack;
7
8 int create(int* top){//創造陣列
9     int i,n;
10
11     stack = (int*)malloc(sizeof(int)*MAX);
12
13     for (i = 0 ; i < MAX ; i++){
14         stack[i]=0;
15     }
16     *top = -1;
17 }
18 int isFull(int* top){//判斷陣列是否為滿
19
20     if (*top >= MAX)
21         return 1;
22     else
23         return 0;
24 }
25 int isEmpty(int* top){//判斷陣列是否為空
26
27     if (*top == -1)
```

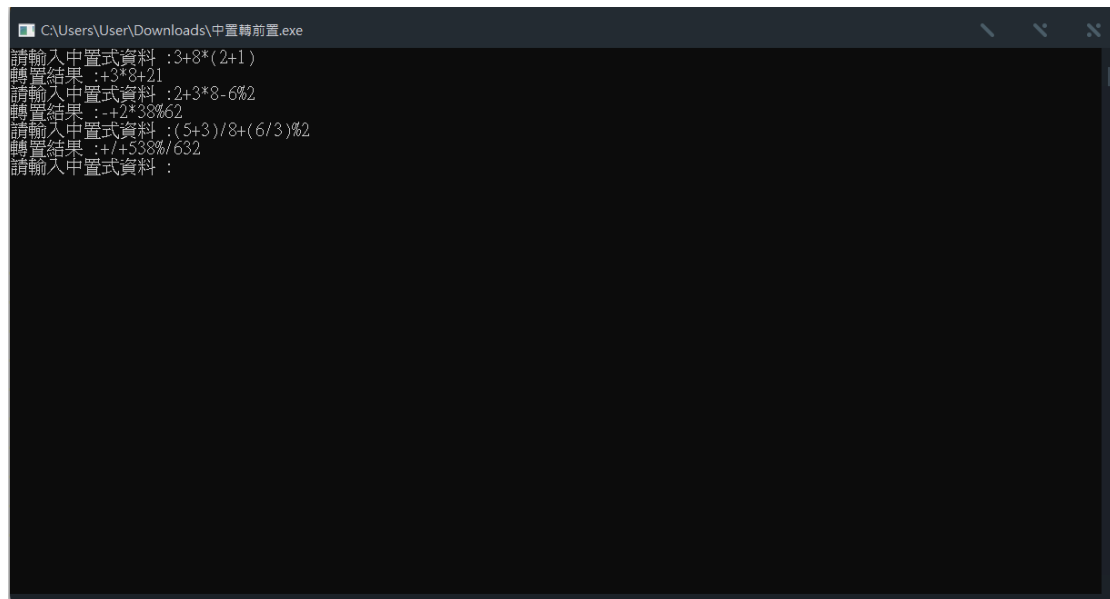
```
28         return 1;
29     else
30         return 0;
31 }
32 }
33 int priority(char op) { //比較各運算子的優先順序
34     switch(op) {
35         case '+': case '-' : return 12 ;
36         case '*': case '/' : case '%' : return 13;
37         case '^': return 14 ;
38         default : return 0 ;
39     }
40 }
41
42 void push(int* top,int item)//將運算子存入stack
43 {
44     if (isFull(&(*top)))
45         printf("堆疊已滿!\n");
46     else
47         stack[++(*top)] = item;
48 }
49
50
51 char pop(int* top)//將運算子從stack取出
52 {
53     if(isEmpty(top))
54
```

```
C:\Users\User\Downloads\中置轉前置.c - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(R) 工具(T) AStyle 視窗(W) 求助(H)
中置轉前置.c 中置轉前置.c [C] 中置轉前置.c
55     printf("堆疊已空!\n");
56     else
57         return stack[( *top)--];
58 }
59
60 void InToPostfix(char infix[], char postfix[])
61 {
62     int i, j, top = -1;
63     create(&top);
64     for(i = strlen(infix) - 1, j = 0; i >= 0; i--)
65         switch(infix[i])
66         {
67             case ')': // 運算子堆疊
68                 /* 1. push infix[i] 至 stack 中 */
69                 push(&top, infix[i]);
70                 break;
71             case '+': case '-': case '*': case '/': case '^': case '%':
72                 while(priority(stack[top]) > priority(infix[i])) { // 比較優先權
73                     /* 2. 將 stack 中值 pop 出 存入 postfix[j++] 中 */
74                     postfix[j++] = pop(&top);
75                 }
76                 /* 2. push infix[i] 至 stack 中 */ // 將該運算子加入堆疊
77                 push(&top, infix[i]);
78                 break;
79             case '(':
80                 while(stack[top] != ')') { // 遇 ) 輸出至 (
81
```

```
82                 /* 3. 將 stack 中值 pop 出 存入 postfix[j++] 中 */
83                 postfix[j++] = pop(&top);
84             }
85             /* 將 stack 中值 pop 出 */ // 不輸出 (
86             pop(&top);
87             break;
88             default: // 運算元直接輸出
89                 postfix[j++] = infix[i];
90         }
91     while(top > -1) {
92         postfix[j++] = pop(&top);
93     } // 將堆疊內所有運算元輸出
```

```
C:\Users\User\Downloads\中置轉前置.c - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(R) 工具(T) AStyle 視窗(W) 求助(H)
中置轉前置.c 中置轉前置.c [C] 中置轉前置.c
94     /* 5. 將 stack 中值 pop 出 存入 postfix[j++] 中 */
95
96 }
97 int main(void)
98 {
99     int n;
100     char infix[MAX], postfix[MAX];
101
102     while(1) {
103         for(n = 0; n < MAX; n++) // 陣列初始化
104             infix[n] = '\0';
105         postfix[n] = '\0';
106
107         printf("請輸入中置式資料 :");
108         scanf("%s", infix);
109         if(strlen(infix) < 3) { // 當輸入資料無法計算時跳出迴圈
110             printf("Error");
111             break;
112         }
113
114         InToPostfix(infix, postfix); // 轉前置的副函式
115         printf("轉置結果 :");
116         for(n = strlen(postfix) - 1; n >= 0; n--) // 顯示資料
117             printf("%c", postfix[n]);
118         printf("\n");
119     }
120 }
```

執行結果



```
C:\Users\User\Downloads\中置轉前置.exe
請輸入中置式資料 :3+8*(2+1)
轉置結果 :+3*8+21
請輸入中置式資料 :2+3*8-6%2
轉置結果 : -+2*38%62
請輸入中置式資料 : (5+3)/8+(6/3)%2
轉置結果 : +/+538%/632
請輸入中置式資料 :
```

時間複雜度

假設輸入的中置式長度為 n ，則他的時間複雜度為 $O(n)$ 。

心得

這題在實作上比較沒有困難點，主要是從中置轉後置的演算法做點改變就達到了預想的效果，這題讓我更加了解了前置運算式的規則。

前置求值

題目定義

我們先創造 2 個資料結構，一個是 **prefix**，用來儲存使用者輸入的前置表示式，另一個則是 **stack**，用來儲存運算元及運算後的結果。

演算法

因為是前置式所以我們從後往前讀取，若遇到運算原則丟進 **stack**，若遇到運算子則將 **stack** 中拿出兩個，做完運算後再把結果丟進 **stack**。一直重複以上動作，直到前置運算式讀取完畢。從 **stack** 中拿出一個整數，即為此前置式之值

原始程式碼

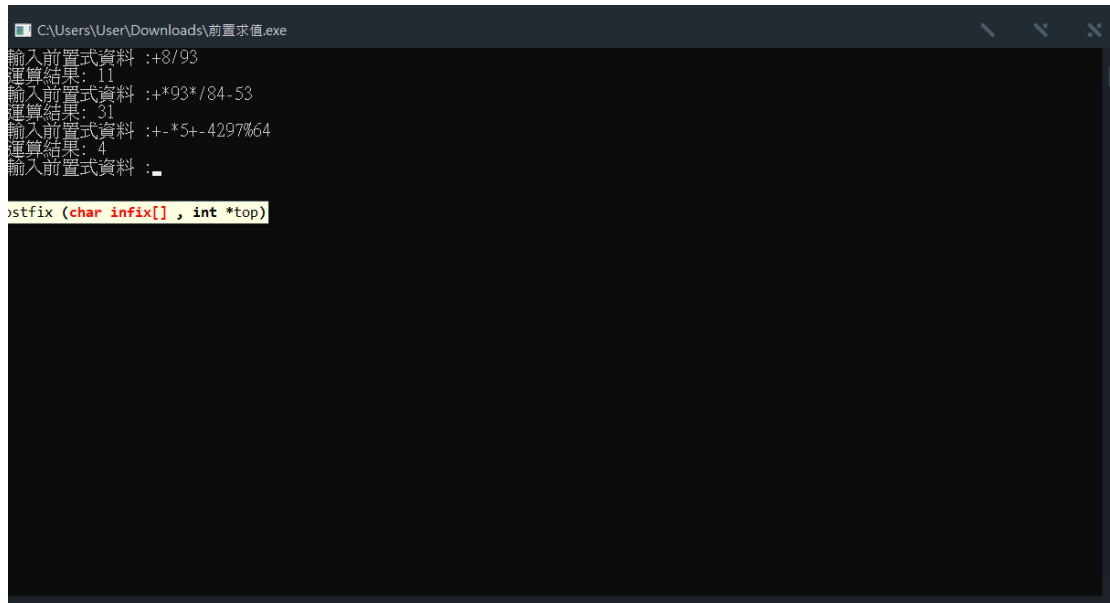
```
C:\Users\User\Downloads\前置作業c - [Executing] - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(O) 工具(T) AStyle 視窗(W) 求助(H)
gdbgui
前置作業c - [前置作業c] 前置作業c
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #define MAX 100
5
6 int *stack;
7
8 void create(){//創造陣列
9     int i,n;
10
11     stack = (int*)malloc(sizeof(int)*MAX);
12
13     for (i = 0 ; i < MAX ; i++){//建立初始值
14         stack[i] = 0;
15     }
16 }
```

```
C:\Users\User\Downloads\前置作業c - [Executing] - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(O) 工具(T) AStyle 視窗(W) 求助(H)
gdbgui
前置作業c - [前置作業c] 前置作業c
17 int isFull(int* top){//判斷陣列是否已滿
18
19     if (*top >= MAX)
20         return 1;
21     else
22         return 0;
23 }
24 int isEmpty(int* top){//判斷陣列是否為空
25
26     if (*top == -1)
27         return 1;
28     else
29         return 0;
30 }
31 }
```

```
C:\Users\User\Downloads\前置作業c - [Executing] - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(O) 工具(T) AStyle 視窗(W) 求助(H)
gdbgui
前置作業c - [前置作業c] 前置作業c
32 void push(int* top,int item)//將資料存入陣列
33 {
34     if (isFull(top))
35         printf("堆疊已滿!\n");
36     else
37         stack[++(*top)] = item;
38 }
39
40
41 int pop(int* top)//將資料提出
42 {
43     if(isEmpty(top))
44         printf("堆疊已空!\n");
45     else{
46         return stack[(--*top)];
47     }
48 }
49 }
```

```
C:\Users\User\Downloads\前置作業c - [Executing] - Dev-C++ 5.11
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(O) 工具(T) AStyle 視窗(W) 求助(H)
gdbgui
前置作業c - [前置作業c] 前置作業c
50
51 void InToPostfix(char infix[] , int *top)
52 {
53     int i, j , k , nth;
54     create();
55     int op1,op2;
56     for(i = strlen(infix) - 1, j = 0; i >= 0; i--) { //從陣列最右邊開始做判斷
57         switch(infix[i])
58         {
59             case '+': case '-': case '*': case '/': case '^': //決定做何種運算
60                 op1 = pop(top); //pop(top)-'0'??
61                 op2 = pop(top);
62
63                 if(infix[i] == '+') //---'A+B'
64                     push(top,op1 + op2);
65                 else if(infix[i] == '-') //---'A-B'
66                     push(top,op1 - op2);
67                 else if(infix[i] == '*') //---'A*B'
68                     push(top,op1 * op2);
69                 else if(infix[i] == '/') //---'A/B'
70                     push(top,op1 / op2);
71                 else if(infix[i] == '^') //---'A%B'
```


執行結果



```
C:\Users\User\Downloads\前置求值.exe
輸入前置式資料 :+3/93
運算結果: 11
輸入前置式資料 :+*93*/84-53
運算結果: 31
輸入前置式資料 :+-*5+-4297%64
運算結果: 4
輸入前置式資料 :
stfix(char infix[], int *top)
```

時間複雜度

假設輸入的前置式長度為 n ，則他的時間複雜度為 $O(n)$ 。

心得

這題在實作上比上題困難，主要是從後置求值的演算法做點改變就達到了預想的效果，但我依舊思考了很久才想到該如何實作。這題讓我更加了解了前置運算式的規則。