



ใบงานที่ 5

เรื่อง Convert infix to postfix

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นายกฤษฎา วิทยา

65543206041-7

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

- แสดงโค้ดโปรแกรมเป็นส่วนๆพร้อมทั้งอธิบาย
- แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
- สรุปผลการทดลอง

ลำดับขั้นตอนการทดลอง

```
6  #include <stdio.h>           //use printf()
7  #include <conio.h>           //use getch()
8  #include <string.h>          //use string function
9  #define MaxStack 40         // Set Max Operator Stack
10 char infix1[80] = {"A+B*(C^D*E/F)-G"}; // Assign INFIX
11 char OpSt[MaxStack];        // Operator stack size
12 int SP = 0;                 // Initial SP=0
```

- **#include <stdio.h>:** เป็นส่วนของโค้ดที่ใช้ระบุว่าโปรแกรมต้องการใช้ฟังก์ชันที่เกี่ยวข้องกับการป้อนออกข้อมูล
- **#include <conio.h>:** เป็นส่วนของโค้ดที่ใช้ระบุว่าโปรแกรมต้องการใช้ฟังก์ชันที่เกี่ยวข้องกับการควบคุมอินพุตทางคีย์บอร์ด getch()
- **#include <string.h>:** ใช้เพื่อนำเข้าไฟล์ส่วนของฟังก์ชันที่เกี่ยวข้องกับการจัดการกับสตริง (string) เช่น strlen() เพื่อหาความยาวของสตริง และ strcpy() เพื่อคัดลอกสตริง.
- **#define MaxStack 40:** กำหนดค่าคงที่ MaxStack เป็น 40 ซึ่งเป็นขนาดสูงสุดของสแต็กแบบตัวดำเนินการ (Operator Stack) ที่ใช้ในโปรแกรม.
- **char infix1[80] = {"A+B*(C^D*E/F)-G"}:** กำหนดค่านิพจน์ในรูปแบบ infix ที่ต้องการแปลงเป็นค่านิพจน์แบบ postfix โดยให้ค่าเริ่มต้นเป็น "A+B*(C^D*E/F)-G"
- **char OpSt[MaxStack]:** ประกาศตัวแปร OpSt ให้เป็นอาร์เรย์ขนาด MaxStack ที่ใช้เก็บตัวดำเนินการใน Stack
- **int SP = 0:** ประกาศตัวแปรชนิด int ชื่อ SP (Stack Pointer) และกำหนดค่าเริ่มต้นให้เป็น 0 ซึ่งเป็นตัวชี้ที่ใช้บ่งชี้ตำแหน่งล่าสุดในสแต็กแบบตัวดำเนินการ.

```

14 void push(char oper) { // PUSH Function
15     if (SP == MaxStack) // Check Stack FULL?
16         printf("ERROR STACK OVER FLOW!!! ...\n");
17     else {
18         SP = SP + 1; // Increase SP
19         OpSt[SP] = oper; // Put data into Stack
20     }
21 }

```

push(char oper): เป็นฟังก์ชันที่ใช้ในการเพิ่มข้อมูลลงใน Stack โดยเช็คกว่า Stack เต็มหรือไม่ ถ้าเต็มจะแสดงข้อความว่า "ERROR STACK OVER FLOW!!!" และหาก Stack ยังไม่เต็มจะเพิ่มข้อมูลลงใน Stack และเพิ่มค่า SP (Stack Pointer) ด้วย โดยใช้ตัวแปร OpSt เก็บข้อมูลใน Stack โดย SP คือตำแหน่งล่าสุดที่ว่างใน Stack.

```

23 int pop() { // POP Function
24     char oper;
25     if (SP != 0) { // Check Stack NOT EMPTY?
26         oper = OpSt[SP]; // Get data from Stack
27         SP--; // Decrease SP
28         return (oper); // Return data
29     }
30     else
31         printf("\nERROR STACK UNDER FLOW!!! ...\n");
32 }

```

pop(): เป็นฟังก์ชันที่ใช้ในการลบข้อมูลออกจาก Stack โดยเช็คกว่า Stack ไม่เป็น Empty ถ้าไม่เป็นจะนำข้อมูลจากตำแหน่งล่าสุดใน Stack (OpSt[SP]) และลดค่า SP ลง 1 จากนั้นจะส่งค่าข้อมูลนั้นกลับออกมา

```

34 int precedenceIP(char oper) { // Function for check precedence of input operator
35     switch (oper) {
36         case '+':
37             return (1);
38         case '-':
39             return (1);
40         case '*':
41             return (2);
42         case '/':
43             return (2);
44         case '^':
45             return (4);
46         case '(':
47             return (4);
48     }
49 }

```

precedenceIP(char oper): เป็นฟังก์ชันที่ใช้ในการตรวจสอบลำดับความสำคัญของตัวดำเนินการที่รับเข้ามา (oper) โดยใช้ค่าที่สั่งให้ตามลำดับความสำคัญที่กำหนดในฟังก์ชันนี้

```

51 int precedenceST(char oper) { // Function for check precedence of stack operator
52     switch (oper) {
53         case '+':
54             return (1);
55         case '-':
56             return (1);
57         case '*':
58             return (2);
59         case '/':
60             return (2);
61         case '^':
62             return (3);
63         case '(':
64             return (0);
65     }
66 }

```

precedenceST(char oper): เป็นฟังก์ชันที่ใช้ในการตรวจสอบลำดับความสำคัญของตัวดำเนินการที่อยู่ใน Stack (OpSt) โดยใช้ค่าที่สั่งให้ตามลำดับความสำคัญที่กำหนดในฟังก์ชันนี้

```

68 void infixToPostfix(char infix2[80]) {
69     int i, j, len;
70     char ch, temp;
71     printf("INFIX : %s\n ", infix2); // Show infix
72     len = strlen(infix2); // Find length of infix
73     printf("Infix Length = %d\n", len); // Display length of infix
74     printf("POSTFIX IS : ");
75     for (i = 0; i <= len - 1; i++) // split infix
76     {
77         ch = infix2[i]; // Transfer character in to ch variable
78         if (strchr("+-*/^()", ch) == 0) // Check Is OPERAND?
79             printf("%c", ch); // Out to Postfix
80         else // If OPERATOR do below
81         {
82             if (SP == 0) // Stack empty?
83                 push(ch); // Push any way if Stack empty
84             else if (ch != '(') // If not '(' do below
85             {
86                 if (precedenceIP(ch) > precedenceST(OpSt[SP])) // If precedence input > precedence in stack
87                     push(ch); // Push input operator to Stack
88                 else {
89                     printf("%c", pop()); // Out to Postfix
90                     while (precedenceIP(ch) <= precedenceST(OpSt[SP]) && (SP != 0)) // Do Until precedence input > precedence in stack
91                         printf("%c", pop()); // Out to Postfix
92                     push(ch); // Push operator input to Stack
93                 }
94             }
95             else {
96                 temp = pop(); // Pop operator from Stack
97                 while ((temp != '(')) // Do if not found '('
98                 {
99                     printf("%c", temp); // Out to Postfix
100                     temp = pop(); // Pop again and loop
101                 }
102             }
103         }
104     }
105     j = SP; // Let j for count left Stack
106     for (i = 1; i <= j; i++) // POP all if Input is NULL
107         printf("%c", pop()); // Out to Postfix
108 }

```

infixToPostfix(char infix2[80]): เป็นฟังก์ชันหลักที่ใช้ในการแปลงนิพจน์ Infix เป็น Postfix โดยใช้วิธีการดังนี้:

- รับนิพจน์ Infix เข้ามาในรูปแบบของอาร์กิวเมนต์ infix2
- หาความยาวของ Infix ด้วยฟังก์ชัน strlen()
- แปลง Infix เป็น Postfix โดยตัวอักษรที่ไม่ใช่ตัวดำเนินการจะถูกแสดงออกทางหน้าจอ
- ตัวดำเนินการจะถูกดำเนินการเพิ่มลงใน Stack ตามลำดับความสำคัญ หากตัวดำเนินการใหม่มีความสำคัญมากกว่าตัวดำเนินการที่อยู่ใน Stack จะถูกเพิ่มลงใน Stack โดยใช้ฟังก์ชัน push() แต่หากตัวดำเนินการใหม่มีความสำคัญน้อยกว่าหรือเท่ากับตัวดำเนินการที่อยู่ใน Stack จะทำการลบตัวดำเนินการใน Stack และแสดงตัวดำเนินการนั้นออกทางหน้าจอ จนกว่าจะพบตัวดำเนินการที่มีความสำคัญน้อยกว่าตัวดำเนินการใหม่ แล้วจึงทำการเพิ่มตัวดำเนินการใหม่ลงใน Stack ด้วย push()
- เมื่อตำแหน่งของตัวดำเนินการที่อยู่ใน Stack ถูกหนัยงสุดท้ายแล้ว หรือพบวงเล็บเปิด (จะทำการลบตัวดำเนินการออกจาก Stack และแสดงออกทางหน้าจอ จนกว่าจะพบวงเล็บปิด)
- เมื่อแปลงเสร็จสิ้น จะทำการแสดงผลลัพธ์ในรูปแบบ Postfix ทางหน้าจอ

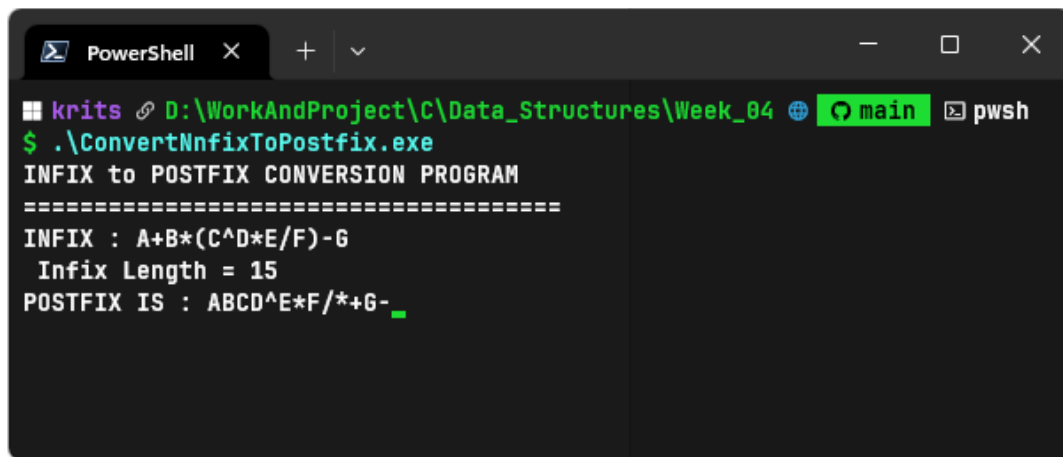
```

110 int main() {
111     printf("INFIX to POSTFIX CONVERSION PROGRAM\n");
112     printf("=====\n");
113     infixToPostfix(infix1);
114     getch();
115     return (0);
116 } // End MAIN

```

main(): เป็นฟังก์ชันหลักของโปรแกรมที่ทำหน้าที่เรียกใช้ฟังก์ชัน infixToPostfix() เพื่อแปลงนิพจน์ Infix (infix1) เป็น Postfix และแสดงผลพร้อมออกทางหน้าจอ

Run Program



The screenshot shows a PowerShell terminal window with the following content:

```

PowerShell
krirts D:\WorkAndProject\C\Data_Structures\Week_04 main pwsh
$ .\ConvertNnfixToPostfix.exe
INFIX to POSTFIX CONVERSION PROGRAM
=====
INFIX : A+B*(C^D*E/F)-G
Infix Length = 15
POSTFIX IS : ABCD^E*F/*+G-

```

สรุปผลการทดลอง

โปรแกรมนี้เป็นโปรแกรมที่ช่วยในการแปลงนิพจน์ทางคณิตศาสตร์จากรูปแบบ Infix เป็นรูปแบบ Postfix เพื่อให้ง่ายต่อการประมวลผลและการวิเคราะห์นิพจน์ในภาษาโปรแกรมหรือระบบอื่น ๆ ที่ต้องการใช้งานนิพจน์ทางคณิตศาสตร์ในรูปแบบ Postfix.