ใบงานที่ 5

เรื่อง Convert infix to postfix

เสนอ

อาจารย์ ปิยพล ยืนยงสถาวร

จัดทำโดย

นายธีรเดช ประเสริฐวงศ์พนา

65543206016-9

ใบงานนี้เป็นส่วนหนึ่งของรายวิชา โครงสร้างข้อมูลและขั้นตอนวิธี

หลักสูตรวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ประจำภาคที่ 1 ปีการศึกษา 2566

คำสั่ง/คำชี้แจง

* ให้ระบุคำสั่ง/คำชี้แจง/คำอธิบาย ตามที่กำหนดให้

1. สร้างโปรแกรม Convert infix to postfix ตามตัวอย่างในเอกสารประกอบการสอน 4
2. แสดงโค้ดโปรแกรมเป็นส่วนๆ พร้อมอธิบาย
3. แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน
4. สรุปผลการทดลอง

ลำดับขั้นการทดลอง

* ให้ระบุขั้นตอนการทดลอง ผลลัพธ์ที่ได้ โดยใช้รูปภาพประกอบ และอธิบายอย่างละเอียด

แสดงโค้ดโปรแกรมเป็นส่วน ๆ พร้อมอธิบาย

ส่วนที่ 1

```
#include <stdio.h> //use printf()
#include <conio.h> //use getch()
#include <string.h> //use string function
#define MaxStack 40 //Set Max Operator Stack

char infix1[80] ={"A+B*(C^D*E/F)-G"}; //Assign INFIX
char OpSt[MaxStack]; //Operator stack size
int SP = 0; //Initial SP=0
```

ส่วนที่ 1 เป็นที่ประการเรียกใช้ library และประการตัวแปร ในโค้ดดังนี้

-#include <stdio.h> คือ เรียกใช้ไลบรารีสำหรับการแสดงผลทางหน้าจอด้วยฟังก์ชัน printf()

-#include <conio.h> คือ เรียกใช้ไลบรารีสำหรับการรับค่าอักขระด้วยฟังก์ชัน getch()

-#include <string.h> คือ เรียกใช้ไลบรารีสำหรับการจัดการกับสตริงด้วยฟังก์ชันต่างๆ เช่น strlen()

-#define MaxStack 40 คือ กำหนดค่าคงที่ MaxStack เท่ากับ 40 ซึ่งใช้เป็นขนาดของสตรัคเก็ตในการเก็บตัวดำเนินการ

-char infix1[80] = {"A+B*(C^D*E/F)-G"} คือ ประกาศตัวแปร infix1 เป็นอาร์เรย์ของตัวอักษรขนาด 80 ใช้เก็บสมการอินฟิกซ์

-char OpSt[MaxStack] คือ ประกาศตัวแปร OpSt เป็นอาร์เรย์ของตัวอักษรขนาด MaxStack เป็นสตรัคเก็ตในการเก็บตัวดำเนินการ

-int SP = 0 คือ ประกาศตัวแปร SP เป็นจำนวนเต็มและกำหนดค่าเริ่มต้นเป็น 0 เป็นตัวชี้สำหรับบอกตำแหน่งในสตรัคเก็ต

ส่วนที่ 2 push

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

ส่วนที่ 2 ฟังก์ชัน void push(char oper) เป็นฟังก์ชันเพิ่มตัวดำเนินการลงใน stack ที่ใช้เก็บตัวดำเนินการของสมการอินฟิกซ์ที่รับตัวอักษร oper ในพารามิเตอร์

-if (SP == MaxStack) คือ ตรวจสอบว่าสตรัคเก็ตเต็มหรือไม่ โดยเปรียบเทียบค่าของตัวชี้SP กับค่าคงที่ MaxStack ถ้าตรงเงื่อนไขให้ทำดังนี้

printf(“ERROR STACK OVER FLOW!!!...\n”) คือ แสดงข้อความ “ERROR STACK OVER FLOW!!!...” ถ้า stack เต็มและไม่สามารถเพิ่มตัวดำเนินการได้

-else คือ ถ้า stack ยังไม่เต็มให้ทำงานดังนี้

SP = SP + 1 คือ เพิ่มค่าของตัวชี้สตรัคเก็ต SP ขึ้นไป 1 หน่วย

OpSt[SP] = oper คือ ใส่ตัวดำเนินการ oper ใน stack ในตำแหน่งที่ตัวชี้ SP ชี้อยู่

ส่วนที่ 3 pop()

```
int pop() //POP Function
{
 char oper;
 if (SP != 0) //Check Stack NOT EMPTY?
 {
 oper=OpSt[SP]; //Get data from Stack
 SP--; //Decrease SP
 return(oper); //Return data
 }
 else
 printf("\nERROR STACK UNDER FLOW!!!...\n");
}
```

ส่วนที่ 3 ฟังก์ชัน int pop() ใช้ในการเอาตัวดำเนินการออกจากสตรัคเก็ต (stack) ที่ใช้เก็บตัวดำเนินการของสมการอินฟิกซ์

-char oper; คือ ประกาศตัวแปร oper เป็นตัวอักษรที่ใช้เก็บตัวดำเนินการที่จะถูกดึงออกจาก stack

-if (SP != 0) คือ ตรวจสอบว่า stack ไม่ว่าง โดยเช็คค่าของตัวชี้SP ว่าไม่เท่ากับ 0 หรือไม่ถ้าตรงเงื่อนไงให้ทำดังนี้

oper = OpSt[SP]; คือ ดึงตัวดำเนินการที่อยู่ใน stack โดยใช้ตัวชี้SP เป็นตัวกำหนด แล้วเก็บค่าตัวดำเนินการในตัวแปร oper

SP--; คือ ลดค่าของตัวชี้สตรัคเก็ต SP ลง 1 หน่วย เพื่อเลื่อนตำแหน่งชี้สตรัคเก็ตลงไปที่ตำแหน่งก่อนหน้า

return (oper); คือ รีเทิร์นค่าตัวดำเนินการที่ถูกดึงออกจาก stack

-else คือ ถ้า stack ว่างให้ทำดังนี้

printf(“\nERROR STACK UNDER FLOW!!!...\n”); คือ แสดงข้อความ “ERROR STACK UNDER FLOW!!!...” เพื่อแจ้งเตือนว่า stack ว่างและไม่สามารถดึงตัวดำเนินการออกจาก stack ได้

ส่วนที่ 4 precedenceIP

```
int precedenceIP(char oper) //Function for check precedence of input
operator
{
 switch (oper)
 {
 case '+' : return(1);
 case '-' : return(1);
 case '*' : return(2);
 case '/' : return(2);
 case '^' : return(4);
 case '(' : return(4);
 }
}
```

ส่วนที่ 4 ฟังก์ชัน int precedenceIP(char oper) เป็นการตรวจสอบลำดับความสำคัญของตัวดำเนินการที่รับเข้ามา ที่รับตัวอักษร oper ในพารามิเตอร์

-switch (oper) คือ ใช้ในการตรวจสอบค่าของตัวดำเนินการ oper ว่าตรงกับเคสไหน

-case ‘+’: return (1); คือ ถ้า oper เป็น ‘+’ จะรีเทิร์น 1 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘+’ เป็น 1

-case ‘-‘: return (1); คือ ถ้า oper เป็น ‘-‘ จะรีเทิร์น 1 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘-‘ -เป็น 1

-case ‘*’: return (2); คือ ถ้า oper เป็น ‘’ จะรีเทิร์น 2 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘’ เป็น 2

-case ‘/’: return (2); คือ ถ้า oper เป็น ‘/’ จะรีเทิร์น 2 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘/’ เป็น 2

-case ‘^’: return (4); คือ ถ้า oper เป็น ‘^’ จะรีเทิร์น 4 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘^’ เป็น 4

-case ‘(‘: return (4); คือ ถ้า oper เป็น ‘(‘ จะรีเทิร์น 4 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ ‘(‘ เป็น 4

ส่วนที่ 5 precedenceST

```
int precedenceST(char oper) //Function for check precedence of stack
operator
{
 switch (oper)
 {
 case '+' : return(1);
 case '-' : return(1);
 case '*' : return(2);
 case '/' : return(2);
 case '^' : return(3);
 case '(' : return(0);
 }
}
```

ส่วนที่ 5 ฟังก์ชัน int precedenceST(char oper) เป็นการตรวจสอบลำดับความสำคัญของตัวดำเนินการที่อยู่ใน stack รับตัวอักษร oper เป็นพารามิเตอร์

-switch (oper): ใช้ในการตรวจสอบค่าของตัวดำเนินการ oper ว่าตรงกับเคสไหน

-case '+': return (1); คือ ถ้า oper เป็น '+' จะคืนค่า 1 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ '+' เป็น 1

-case '-': return (1); คือ ถ้า oper เป็น '-' จะคืนค่า 1 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ '-' เป็น 1

-case '*': return (2); คือ ถ้า oper เป็น " จะคืนค่า 2 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ " เป็น 2

-case '/': return (2); คือ ถ้า oper เป็น '/' จะคืนค่า 2 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ '/' เป็น 2

-case '^': return (3); คือ ถ้า oper เป็น '^' จะคืนค่า 3 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ '^' เป็น 3

-case '(': return (0); คือ ถ้า oper เป็น '(' จะคืนค่า 0 ซึ่งแสดงถึงลำดับความสำคัญของตัวดำเนินการ '(' เป็น 0

ส่วนที่ 6 infixTOpostfix

```
void infixTOpostfix(char infix2[80])
{
 int i,j, len;
 char ch,temp;
 printf("INFIX : %s\n ",infix2); //Show infix
 len=strlen(infix2); //Find length of infix
 printf("Infix Length = %d \n",len); //Display length of infix
 printf("POSTFIX IS : ");
 for (i=0;i<=len-1;i++) //split infix
 {
 ch=infix2[i]; //Transfer character in to ch variable
 if (strchr("+-* /^()", ch)==0) //Check Is OPERAND?
 printf("%c",ch); //Out to Postfix
 else //If OPERATOR do below
 {
 if (SP==0) //Stack empty?
 push(ch); //Push any way if Stack empty
 else
 if (ch != ')') //If not ')' do below
 {
 if(precedenceIP(ch)>precedenceST(OpSt[SP])) //If
precedence input > precedence in stack
 push(ch); //Push input operator to Stack
 else
 {
 printf("%c",pop()); //Out to Postfix
 while(precedenceIP(ch)<=precedenceST(OpSt[SP]) && (SP != 0)) // Do Until precedence input > precedence in stack
 printf("%c",pop()); //Out to Postfix
 push(ch); //Push operator input to Stack
 }
 }
 else
 {
 temp=pop(); //Pop operator from Stack
 while ((temp != '(') ) // Do if not found '('
 {
 printf("%c",temp); //Out to Postfix
 temp=pop(); //Pop again and loop
 }
 }
 }
 }
 j=SP; //Let j for count left Stack
 for(i=1;i<=j;i++) //POP all if Input is NULL
 printf("%c",pop()); //Out to Postfix
}
```

ส่วนที่ 6 ฟังก์ชัน infixTOpostfix(char infix2[80]) เป็นการแปลงสมการอินฟิกซ์เป็นรูปแบบของสมการโพสต์ฟิกซ์รับสตริง infix2 เป็นพารามิเตอร์

ในกรอบสีแดง

-printf("INFIX : %s\n ", infix2); คือ แสดงข้อความ "INFIX : " ตามด้วยสตริง infix2 ที่รับเข้ามา

-len = strlen(infix2); คือ นับความยาวของสตริง infix2 ด้วยฟังก์ชัน strlen() แล้วเก็บค่าลงในตัวแปร len

-printf("Infix Length = %d \n", len);คือแสดงค่าความยาวของสตริง infix2 ที่ได้จากการนับก่อนหน้านี้

-printf("POSTFIX IS : "); คือ แสดงข้อความ "POSTFIX IS : " เพื่อเตรียมแสดงสมการโพสต์ฟิกซ์

ในกรอบสีเขียว

- for (i = 0; i <= len - 1; i++) คือ ลูป for จะทำการแยกสตริง infix2 ตามตัวอักษรเพื่อดำเนินการในแต่ละตัว ดังนี้

1) ch = infix2[i]; คือ กำหนดตัวแปร ch เป็นตัวอักษรที่ได้จากการแยกสตริง infix2 ในตำแหน่ง i

2) if (strchr("+-* /^()", ch) == 0) คือ ตรวจสอบว่า ch เป็นตัวดำเนินการหรือไม่ โดยใช้ฟังก์ชัน strchr() ซึ่งจะคืนค่าตำแหน่งของตัวอักษรในสตริงถ้าพบ หากไม่พบจะคืนค่า 0
printf("%c", ch) คือ แสดงตัวอักษร ch ในรูปแบบโพสต์ฟิกซ์

3) else คือ ถ้าไม่ใช่ตัวดำเนินการให้printf("%c", ch); แสดงตัวอักษร ch ในรูปแบบโพสต์ฟิกซ์ให้ทำดังนี้

3.1) if (SP == 0) คือ ตรวจสอบว่า stack ว่างหรือไม่ โดยตรวจสอบค่าของตัวชี้SP ว่าเท่ากับ 0 หากเป็นจริงแสดงว่า stack ว่าง push(ch)คือเรียกใช้ฟังก์ชัน push() เพื่อใส่ตัวดำเนินการ ch ลงใน stack

3.2) else คือ ทำดังนี้

3.2.1) if (ch != ')' คือ ตรวจสอบว่า ch ไม่ใช่ตัวปิดวงเล็บ ')' ให้ทำดังนี้

3.2.1.1) if (precedenceIP(ch) > precedenceST(OpSt[SP])) คือตรวจสอบว่าลำดับ ความสำคัญของตัวดำเนินการ ch ที่รับเข้ามามีค่ามากกว่าลำดับความสำคัญของตัวดำเนินการบนสุดใน stack ถ้าตามเงื่อนไขให้
push(ch); คือ เรียกใช้ฟังก์ชัน push() เพื่อใส่ตัวดำเนินการ ch ลงในStack

3.2.1.2) else คือ ถ้าลำดับความสำคัญของตัวดำเนินการ ch ไม่มากกว่าลำดับความสำคัญของตัวดำเนินการบนสุดใน stack ให้ทำดังนี้

1) printf("%c", pop()); คือ แสดงตัวดำเนินการบนสุดใน stack และนำออกจาก stack

2) while (precedenceIP(ch) <= precedenceST(OpSt[SP])
&& (SP !=0)) คือ ลูป while ทำงานตามเงื่อนไข ลำดับความสำคัญของตัวดำเนินการ ch น้อยกว่าหรือเท่ากับลำดับความสำคัญของตัวดำเนินการ บนสุดใน stack และ stack ไม่ว่าง

ให้ลูป printf("%c", pop()); เพื่อ แสดงและนำตัวดำเนินการบนสุดใน stack และนำออกจาก stack

3.2.2) else คือ ถ้า ch เป็นตัวปิดวงเล็บ ')' ให้ทำดังนี้

1) temp = pop(); คือ แยกตัวดำเนินการบนสุดในสตรัคเก็ตออก และเก็บไว้ใน temp

2) while ((temp != '(')) คือ ลูป while ทำงานตามเงื่อนไข คือไม่พบตัวเปิดวงเล็บ '(' ให้ลูป printf("%c", temp); คือ แสดงตัวดำเนินการ temp ในรูปแบบโพสต์ฟิกซ์

3) temp = pop(); คือ แยกตัวดำเนินการบนสุดใน stack ออกและเก็บไว้ใน temp เพื่อทำซ้ำในรอบถัดไป

-j = SP; คือ กำหนดค่าของ j เป็นค่าตัวชี้SP

-for (i = 1; i <= j; i++) คือ ให้ลูป printf("%c", pop()); เพื่อแสดงตัวดำเนินการบนสุดใน stack และนำออกจาก stack

ส่วนที่ 7 main

```
int main ()
{
 printf("INFIX to POSTFIX CONVERSION PROGRAM\n");
 printf("=====================================\n");
 infixTOpostfix(infix1);
 getch();
 return(0);

} //End MAIN
```

ส่วนที่ 7 ฟังก์ชัน int main() เป็นฟังก์ชั่นหลักหรือฟังก์ชั่นที่โปรแกรมจะเริ่มต้นทำงาน

-printf("INFIX to POSTFIX CONVERSION PROGRAM\n"); คือ แสดงข้อความ "INFIX to POSTFIX CONVERSION PROGRAM" เพื่อแสดงชื่อโปรแกรม

-infixTOpostfix(infix1); คือ เรียกใช้ฟังก์ชัน infixTOpostfix() เพื่อแปลงสมการอินฟิกซ์ให้เป็นรูปแบบสมการโพสต์ฟิกซ์ โดยส่งสตริง infix1 เข้าไปเป็นอาร์กิวเมนต์

getch() คือ รอรับการกดปุ่มจากผู้ใช้เพื่อปิดโปรแกรม ฟังก์ชัน getch() จะรอรับค่าปุ่มที่กดและคืนค่ากลับ

แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน

เมื่อผู้ใช้รันโปรแกรม สมการแบบ Infix ที่อยู่ในตัวแปร char infix1[80] ก็จะถูกแปลงให้เป็น สมการแบบ Postfix พร้อมนับจำนวนตัวอักษรที่อยู่ในตัวแปร char infix1[80] ให้ด้วย โดยผ่านฟังก์ชัน infixTOpostfix จากนั้นในฟังก์ชัน infixTOpostfix ก็จะเรียกใช้ ฟังก์ชันย่อยอื่นๆอีก

ผลลัพธ์การรันโปรแกรม

```
INFIX to POSTFIX CONVERSION PROGRAM
=======================================
INFIX : A+B*(C^D*E/F)-G
 Infix Length = 15
POSTFIX IS : ABCD^E*F/*+G-
```

สรุปผลการทดลอง

* ให้ระบุข้อมูลสรุปผลการดำเนินงาน สิ่งที่ได้เรียนรู้ ปัญหา และวิธีการแก้ไข

สรุปผล : โปรแกรมนี้เป็นการแปลงสมการที่เป็นแบบ Infix ให้เป็นสมการแบบ Postfix โดยใช้หลักการของ stack เข้ามาช่วย ด้วยการ push pop ใช้ sp ชี้ตำแหน่งเป็นต้น

ตอบคำถามท้ายการทดลอง

* ตอบคำถามท้ายการทดลอง (ถ้ามี)

.....

.....

.....

.....

สื่อ / เอกสารอ้างอิง

* ให้ระบุที่มาของข้อมูลที่ใช้ประกอบการทดลอง เช่น หนังสือ เว็บไซต์ youtube เป็นต้น

-ในเอกสารประกอบการสอนสัปดาห์ที่ 4 ของอาจารย์ปิยพล ยืนยงสถาวร