**ใบงานที่ 6**

**เรื่อง Calculate 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
#include <math.h> //use power
#define MaxStack 40 //Set Max Operator Stack
char postfix1[80] ={"AB+C-d/"}; //Assign INFIX
float ValPostfix[80]; //Keep value of Postfix here
float ValOperandST[MaxStack]; //Operator stack size
int SP = 0; //Initial SP=0
```

**ส่วนที่ 1 เป็นที่ประการเรียกใช้ library และประการตัวแปร ในโค้ดดังนี้**

-Libraly ที่ใช้มี 4 ตัวดังนี้

<stdio.h> คือ ใช้สำหรับฟังก์ชัน printf() ที่ใช้แสดงผลบนหน้าจอ

<conio.h> คือใช้สำหรับฟังก์ชัน getch() ที่รอรับอักขระที่ผู้ใช้ป้อนจากแป้นพิมพ์โดยไม่แสดงผลบนหน้าจอ

<string.h> คือ ใช้สำหรับฟังก์ชันที่เกี่ยวข้องกับการจัดการข้อความ เช่น strlen() ซึ่งใช้ในการหาความยาวของสตริง

<math.h> คือ ใช้สำหรับฟังก์ชันทางคณิตศาสตร์ เช่น power() ซึ่งใช้ในการคำนวณเลขยกกำลัง

-ตัวแปรมี 5 ตัวดังนี้

MaxStack คือ ค่าคงที่ที่กำหนดขนาดสูงสุดของ stack ตัวดำเนินการ

postfix1 คือ สตริงที่ใช้เก็บนิพจน์รูปแบบ Postfix

ValPostfix คือ อาร์เรย์ที่ใช้เก็บค่าของนิพจน์รูปแบบ Postfix

ValOperandST คือ อาร์เรย์stack ตัวดำเนินการที่ใช้เก็บค่าตัวดำเนินการ

SP คือ ตัวแปรที่ใช้เก็บตำแหน่งปัจจุบันใน stack ตัวดำเนินการ

**ส่วนที่ 2 push**

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

**ส่วนที่ 2 ฟังก์ชัน void push(float ValOperand) เป็นการเพิ่มค่าของตัว Operand เข้าสู่ stack และรับพารามิเตอร์ ValOperand ที่เป็นเลขทศนิยม**

1. if (SP == MaxStack) คือ เช็คว่า stack เต็มหรือไม่ โดยเปรียบเทียบค่าของตัวแปร SP กับค่าคงที่ MaxStack

2. printf("ERROR STACK OVER FLOW!!!...\n");คือถ้า stack เต็มจะพิมพ์ข้อความแจ้งเตือน "ERROR STACK OVER FLOW!!!..." บนหน้าจอ

3. SP = SP + 1; คือ เพิ่มค่าของตัวแปร SP เพื่อเป็นการเลื่อนตำแหน่งสำหรับการเก็บใน stack ที่ช่องถัดไป

4. ValOperandST[SP] = ValOperand; คือ กำหนดค่า ValOperand ลงในช่องข้อมูลของ stack ที่ตำแหน่ง SP

ส่วนที่ 3 pop

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

**ส่วนที่ 3 ฟังก์ชัน float pop() เป็น ใช้ในการดึงค่าตัว Operand จาก stack และส่งคืนค่าที่เป็นตัวเลขทศนิยม (float)**

1. float ValOperand คือ ประกาศตัวแปรใหม่ประเภท float เอาไว้เก็บค่าของ Operand

2. if (SP != 0) คือ เช็คว่า stack ว่างหรือไม่ โดยเปรียบเทียบค่าของตัวแปร SP กับ 0

3. ValOperand = ValOperandST[SP]; คือ ถ้า stack ไม่ว่าง จะเก็บค่าตัว Operand จากสตริง ValOperandST ที่ตำแหน่ง SP ในตัวแปร ValOperand

4. SP--; คือ ลดค่าของตัวแปร SP เพื่อเป็นการเลื่อนตำแหน่งสำหรับการดึงค่าใน stack ตัวดำเนินการไปที่ช่องก่อนหน้า

5. return (ValOperand); คือ ส่งคืนค่าที่ดึงมาจาก stack ตัวดำเนินการ

6. printf("\nERROR STACK UNDER FLOW!!!...\n"); คือ ถ้า stack ว่าง จะพิมพ์ข้อความแจ้งเตือน "ERROR STACK UNDER FLOW!!!..." บนหน้าจอ

## ส่วนที่ 4 CalPostfix

**ส่วนที่ 4** ฟังชั้น void CalPostfix(char postfix[80]) เป็นการคำนวณค่าของนิพจน์ที่เป็นรูปแบบ Postfix

### ส่วนที่ 4.1

```
void CalPostfix(char postfix[80])
{
 float pop1,pop2,value;
 int i,len;
 char ch;
 len = strlen(postfix);
 printf("Postfix = %s\n",postfix);
 for (i=0;i<=len-1;i++) //Assign data to OPERAND
 {
 ch=postfix[i]; //Split Character for assign data
 if (strchr("+-* /^", ch)==0) //Check Is OPERAND?
 {
 printf("\nAssign Number to %c : ",ch);
 scanf("%f",&ValPostfix[i]); //Read data from KBD and direct to Value of OPERAND in Array
 }
 }
```

**ส่วนที่ 4.1 อธิบายดังนี้**

1. ประกาศตัวแปรที่ใช้ในการคำนวณดังนี้

   -pop1, pop2 คือ ตัวแปรที่ใช้ในการเก็บค่าตัวดำเนินการที่ดึงออกจาก stack ตัวดำเนินการ

   -value คือ ตัวแปรที่ใช้เก็บผลลัพธ์การคำนวณ

   -I, len คือ ตัวแปรที่ใช้ในการวนลูป

   -ch คือ ตัวแปรที่ใช้ในการเก็บอักขระที่ถูกแยกออกมาจากนิพจน์

2. len = strlen(postfix) คือ นับความยาวของนิพจน์ Postfix ที่รับเข้ามา
3. printf(“Postfix = %s\n”, postfix); คือ พิมพ์ข้อความแสดงค่าของนิพจน์ Postfix ที่รับเข้ามา
4. for (I = 0; I <= len – 1; i++) คือ วนลูปตามความยาวของนิพจน์ Postfix เพื่อดำเนินการกับแต่ละตัวอักขระในนิพจน์
5. ch = postfix[i]; คือ กำหนดค่าตัวแปร ch เป็นอักขระที่ตำแหน่ง I ในนิพจน์ Postfix
6. if (strchr(“+-* /^”, ch) == 0) คือ เช็คว่าอักขระ ch ไม่ใช่ตัวดำเนินการ (Operator) โดยใช้ฟังก์ชัน strchr() เพื่อค้นหาตัวดำเนินการที่เป็น “+-* /^” ถ้าไม่เป็นตัวดำเนินการจะเข้าสู่เงื่อนไขนี้

   6.1 printf(“\nAssign Number to %c : “, ch); คือ พิมพ์ข้อความ “Assign Number to “ พร้อมกับแสดงอักขระ ch ที่ต้องกำหนดค่าตัวดำเนินการ

   6.2 scanf(“%f”, &ValPostfix[i]); คือ อ่านค่าจากผู้ใช้จากแป้นพิมพ์และเก็บค่าที่ผู้ใช้ป้อนลงในตัวแปร ValPostfix[i] ซึ่งเป็นตำแหน่ง I ในอาร์เรย์ ValPostfix ซึ่งใช้ในการเก็บค่าตัวดำเนินการ

**ส่วนที่ 4.2**

```
for (i=0;i<=len-1;i++) //Calculate Value of POSTFIX
{
ch=postfix[i]; //Split Character for prepare to STACK
if (strchr("+-* /^", ch)==0) //Check Is OPERAND?
{
push(ValPostfix[i]); //push value of OPERAND to Stack
}
else
{
pop1=pop(); //Pop 1st
pop2=pop(); //pop 2nd
switch (ch)
{
case '+' : value=pop2+pop1; //Calculate
push(value); //Push value to Stack
break;
case '-' : value=pop2-pop1;
push(value);
break;
case '*' : value=pop2*pop1;
push(value);
break;
case '/' : value=pop2/pop1;
push(value);
break;
case '^' : value=pow(pop2,pop1);
push(value);
break;
}
} //End IF
} //End IF
printf("\nANS = %f",pop()); //Last value is ANSWER
} //End Fn.
```

**ส่วนที่ 4.2 อธิบายดังนี้**

1. for (i = 0; i <= len - 1; i++) คือ วนลูปตามความยาวของนิพจน์ Postfix เพื่อดำเนินการกับแต่ละตัวอักขระในนิพจน์
2. ch = postfix[i]; คือ กำหนดค่าตัวแปร ch เป็นอักขระที่ตำแหน่ง i ในนิพจน์ Postfix
3. if (strchr(“+-* /^”, ch) == 0) คือ เช็คว่าอักขระ ch ไม่ใช่ตัวดำเนินการ (Operator) โดยใช้ฟังก์ชัน strchr() เพื่อค้นหาตัวดำเนินการที่เป็น “+-* /^” ถ้าไม่เป็นตัวดำเนินการจะเข้าสู่เงื่อนไขนี้
   3.1 push(ValPostfix[i]);คือเรียกใช้ฟังก์ชัน push เพื่อเพิ่มค่าตัวดำเนินการที่เก็บอยู่ในตำแหน่ง i ของอาร์เรย์ ValPostfix เข้าสู่ stack ตัวดำเนินการ
4. else คือ ถ้าอักขระเป็นตัวดำเนินการ (Operator) จะเข้าสู่เงื่อนไขนี้
   4.1 pop1 = pop(); คือ เรียกใช้ฟังก์ชัน pop เพื่อดึงค่าตัวดำเนินการออกมาและเก็บไว้ในตัวแปร pop1
   4.2 pop2 = pop(); คือ เรียกใช้ฟังก์ชัน pop อีกครั้งเพื่อดึงค่าตัวดำเนินการออกมาและเก็บไว้ในตัวแปร pop2
   4.3 switch คือ เพื่อตรวจสอบ ch และคำนวณผลลัพธ์ตามตัวดำเนินการนั้น ๆ:

หากตัวดำเนินการเป็น + จะทำการบวกค่า pop1 กับ pop2 และเก็บผลลัพธ์ในตัวแปร value จากนั้นเรียกใช้ฟังก์ชัน push เพื่อเก็บผลลัพธ์ลงใน stack ตัวดำเนินการ ในกรณีอื่น ๆ เช่น -, *, /, ^ จะดำเนินการเช่นเดียวกับตัวอย่างข้างต้น

5. printf("\nANS = %f", pop()); คือ พิมพ์ค่าคำตอบ (ANSWER) ซึ่งเป็นผลลัพธ์สุดท้ายที่ถูกดึงออกมาจาก stack ตัวดำเนินการ โดยใช้ฟังก์ชัน pop

**ส่วนที่ 5 main**

```
int main()
{
 printf("POSTFIX CALCULATION PROGRAM\n");
 printf("============================\n");
 CalPostfix(postfix1);
 getch();
 return(0);

 } //End Main
```

**ส่วนที่ 5 ฟังก์ชัน int main() เป็นฟังก์ชันหลักหรือฟังก์ชันที่โปรแกรมจะเริ่มต้นทำงาน**

1. printf("POSTFIX CALCULATION PROGRAM\n"); คือ พิมพ์ข้อความ "POSTFIX CALCULATION PROGRAM" บนหน้าจอเพื่อแสดงชื่อโปรแกรม
2. CalPostfix(postfix1); คือ เรียกใช้ฟังก์ชัน CalPostfix เพื่อคำนวณค่าของนิพจน์ Postfix โดยส่งนิพจน์ที่จะคำนวณเป็นอาร์กิวเมนต์ ในที่นี้คือ postfix1 ซึ่งเป็นตัวแปรที่ถูกกำหนดค่าไว้ก่อนหน้านี้
3. getch(); คือ รอรับการกดปุ่มจากแป้นพิมพ์เพื่อทำให้โปรแกรมหยุดทำงานและรอให้ผู้ใช้ดูผลลัพธ์
4. return (0); คือ ส่งคืนค่า 0 เพื่อแสดงว่าโปรแกรมสิ้นสุดการทำงานอย่างปกติ

แสดงผลการรันโปรแกรม พร้อมอธิบายการทำงาน

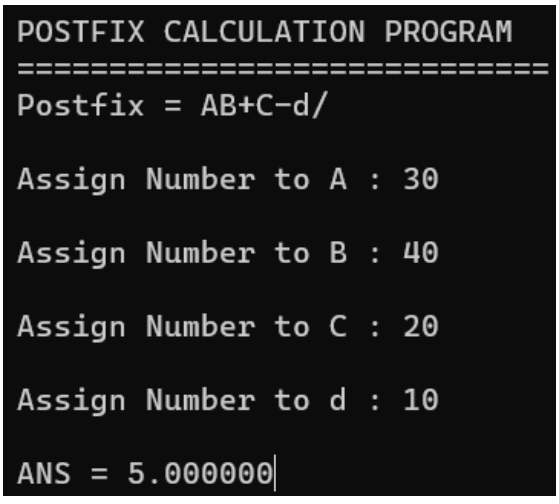
1. โปรแกรมเริ่มต้นที่ฟังก์ชัน main แสดงข้อความเริ่มต้น "POSTFIX CALCULATION PROGRAM" และเส้นขั้น "=============================="
2. เรียกใช้ฟังก์ชัน CalPostfix เพื่อคำนวณค่าของนิพจน์ Postfix โดยใช้นิพจน์ AB+C-d/ ที่ถูกกำหนดในตัวแปร postfix1
3. ในฟังก์ชัน CalPostfix จะมีขั้นตอนดังนี้
   3.1 แสดงค่าของนิพจน์ Postfix ที่รับเข้ามา "Postfix = AB+C-d/"
   3.2 รับค่าตัวดำเนินการ A, B, C, d จากผู้ใช้ โดยแสดงข้อความ "Assign Number to [ตัวดำเนินการ] : " และใช้ scanf เพื่อรับค่าจากผู้ใช้และเก็บในตัวแปร ValPostfix

3.3 วนลูปตามความยาวของนิพจน์ Postfix เพื่อคำนวณค่า

3.3.1 ถ้าตัวอักขระไม่ใช่ตัวดำเนินการ จะนำค่าตัวดำเนินการไปเก็บใน stack ตัวดำเนินการโดยใช้ฟังก์ชัน push

3.3.2 ถ้าตัวอักขระเป็นตัวดำเนินการ จะดึงค่าตัวดำเนินการออกมาจาก stack ตัวดำเนินการ 2 ค่า และดำเนินการคำนวณผลลัพธ์ตามตัวดำเนินการนั้น ๆ โดยใช้คำสั่ง switch และเก็บผลลัพธ์ในสแต็กตัวดำเนินการด้วยฟังก์ชัน push

3.4 แสดงคำตอบ (ANSWER) ที่คำนวณได้จาก stack ตัวดำเนินการโดยใช้ฟังก์ชัน pop



**สรุปผลการทดลอง**

* ให้ระบุข้อมูลสรุปผลการดำเนินงาน สิ่งที่ได้เรียนรู้ ปัญหา และวิธีการแก้ไข

**สรุปผล** : โปรแกรมนี้เป็นโปรแกรมคำนวณค่าของนิพจน์ที่เป็นรูปแบบ Postfix โดยผู้ใช้จะกรอกค่าตัวดำเนินการและโปรแกรมจะคำนวณค่าของนิพจน์โดยใช้หลัก push, pop ของ stack

**ตอบคำถามท้ายการทดลอง**

* ตอบคำถามท้ายการทดลอง (ถ้ามี)

.....

.....

.....

.....

**สื่อ / เอกสารอ้างอิง**

* ให้ระบุที่มาของข้อมูลที่ใช้ประกอบการทดลอง เช่น หนังสือ เว็บไซต์ youtube เป็นต้น

-ในเอกสารประกอบการสอนสัปดาห์ที่ 4 ของอาจารย์ปิยพล ยืนยงสถาวร