



รายงาน

เรื่อง โปรแกรม Calculate Postfix by assigned in variable

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง โปรแกรม Calculate Postfix by assigned in variable

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาควิชา วิทยาการคอมพิวเตอร์ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในการอธิบายโค้ดของโปรแกรม Calculate Postfix by assigned in variable รวมถึงอธิบายหลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable และอธิบายผลลัพธ์การใช้งานโปรแกรม Calculate Postfix by assigned in variable

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม Calculate Postfix by assigned in variable หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 01/08/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม Calculate Postfix by assigned in variable พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable	5
ผลลัพธ์การใช้งานโปรแกรม Calculate Postfix by assigned in variable	13
บรรณานุกรม	14

โค้ดของโปรแกรม Calculate Postfix by assigned in variable พร้อมคำอธิบาย

```

/*
Program calculate Postfix by assigned in variable.
=====
*/
#include <stdio.h> //ใช้ printf()
#include <conio.h> //ใช้ getch()
#include <string.h> //ใช้ฟังก์ชัน string
#include <math.h> //ใช้ power
#define MaxStack 40 //กำหนดขนาดสูงสุดของ stack
char postfix1[80] ={"AB+C-d/"}; //กำหนดค่า INFIX
float ValPostfix[80]; //เก็บค่าของ Postfix ไว้ที่นี่
float ValOperandST[MaxStack]; //ขนาดของ stack สำหรับตัวดำเนินการ
int SP = 0; //กำหนดค่าเริ่มต้นของ SP=0
void push(float ValOperand) //ฟังก์ชัน PUSH
{
    if(SP == MaxStack) //ตรวจสอบว่า stack เต็มหรือไม่
    {
        printf("ERROR STACK OVER FLOW!!!...\n");
    }
    else
    {
        SP=SP+1; //เพิ่มค่า SP
        ValOperandST[SP]=ValOperand; //ใส่ข้อมูลเข้าไปใน stack
    }
}

```

โค้ดของโปรแกรม Calculate Postfix by assigned in variable พร้อมคำอธิบาย (ต่อ)

```
float pop() //ฟังก์ชัน POP
{
    float ValOperand;
    if (SP != 0) //ตรวจสอบว่า stack ไม่ว่างหรือไม่
    {
        ValOperand=ValOperandST[SP]; //รับข้อมูลจาก stack

        SP--; //ลดค่า SP

        return (ValOperand); //ส่งคืนข้อมูล
    }
    else
        printf("\nERROR STACK UNDER FLOW!!!...\n");
}

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++) //กำหนดข้อมูลให้กับ OPERAND
    {
        ch=postfix[i]; //แยกตัวอักษรเพื่อกำหนดข้อมูล

        if (strchr("+-* /^", ch)==0) //ตรวจสอบว่าเป็น OPERAND หรือไม่
        {
            printf("\nAssign Number to %c : ",ch);

            scanf("%f",&ValPostfix[i]); //อ่านข้อมูลจากคีย์บอร์ดและกำหนดค่าให้กับ OPERAND ใน
            array โดยตรง
        }
    }
}
```

โค้ดของโปรแกรม Calculate Postfix by assigned in variable พร้อมคำอธิบาย (ต่อ)

```

for (i=0;i<=len-1;i++) //คำนวณค่าของ POSTFIX
{
    ch=postfix[i]; //แยกตัวอักษรเพื่อเตรียมเข้า stack

    if (strchr("+-* /^", ch)==0) //ตรวจสอบว่าเป็น OPERAND หรือไม่
    {
        push(ValPostfix[i]); //push ค่า OPERAND เข้า stack
    }
    else
    {
        pop1=pop(); //pop ครั้งที่ 1

        pop2=pop(); //pop ครั้งที่ 2

        switch (ch)
        {
            case '+': value=pop2+pop1; //คำนวณ

                push(value); //push ค่าเข้า 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;

        }

    } //จบ IF

} //จบ IF

printf("\nANS = %f",pop()); //ค่าสุดท้ายคือคำตอบ

} //จบฟังก์ชัน

```

โค้ดของโปรแกรม Calculate Postfix by assigned in variable พร้อมคำอธิบาย (ต่อ)

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


หลักการการทำงานของโปรแกรม Calculate Postfix by assigned in variable

1. การนำเข้าไลบรารีและการกำหนดค่าตัวแปร

```

/*
Program calculate Postfix by assigned in variable.
=====
*/
#include <stdio.h> //ใช้ printf()
#include <conio.h> //ใช้ getch()
#include <string.h> //ใช้ฟังก์ชัน string
#include <math.h> //ใช้ power
#define MaxStack 40 //กำหนดขนาดสูงสุดของ stack
char postfix1[80] = {"AB+C-d/"}; //กำหนดค่า INFIX
float ValPostfix[80]; //เก็บค่าของ Postfix ไว้ที่นี่
float ValOperandST[MaxStack]; //ขนาดของ stack สำหรับตัวดำเนินการ
int SP = 0; //กำหนดค่าเริ่มต้นของ SP=0

```

ในส่วนของการนำเข้าไลบรารีและการกำหนดค่าตัวแปร ได้ทำการนำเข้าไลบรารีที่จำเป็น ได้แก่ `stdio.h` สำหรับการพิมพ์ข้อความ, `conio.h` สำหรับการรอรับการกดปุ่ม, และ `string.h` สำหรับการจัดการสตริง, `math.h` สำหรับการใช้ฟังก์ชันทางคณิตศาสตร์ เป็นฟังก์ชันในส่วนการยกกำลัง จากนั้นกำหนดขนาดสูงสุดของสแตกเป็น 40 และนิพจน์อินฟิกซ์ที่ต้องการคำนวณ รวมถึงตัวแปร `ValPostfix` สำหรับการเก็บค่าของการดำเนินการ Postfix พร้อมตัวแปร `ValOperandST` สำหรับกำหนดขนาดในการเก็บโอเปอเรเตอร์ในสแตก และตัวชี้สแตก `SP` ที่เริ่มต้นเป็น 0 จากการอธิบายข้างต้น สามารถสรุปรายละเอียดได้ ดังนี้

- `stdio.h` ใช้สำหรับฟังก์ชันการรับ-ส่งข้อมูลผ่านคอนโซล เช่น `printf()`
- `conio.h` ใช้สำหรับฟังก์ชัน `getch()` ซึ่งรอการกดปุ่มจากผู้ใช้ก่อนปิดโปรแกรม
- `string.h` ใช้สำหรับการจัดการสตริง เช่น `strlen()` และ `strchr()`
- `math.h` ใช้สำหรับฟังก์ชันทางคณิตศาสตร์ เช่น `pow()` เพื่อยกกำลัง
- `MaxStack` กำหนดขนาดสูงสุดของ stack เป็น 40

- postfix1 เป็น array ที่เก็บโพสต์ฟิกซ์นิพจน์ที่กำหนดไว้ล่วงหน้า
- ValPostfix เป็น array ที่เก็บค่าตัวเลขของโพสต์ฟิกซ์นิพจน์
- ValOperandST เป็น stack สำหรับเก็บค่าตัวดำเนินการ
- SP (Stack Pointer) เป็นตัวชี้ตำแหน่งใน stack ที่กำหนดค่าเริ่มต้นเป็น 0

หลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable (ต่อ)

2. ฟังก์ชัน Push

```
void push(float ValOperand) //ฟังก์ชัน PUSH
{
    if (SP == MaxStack) //ตรวจสอบว่า stack เต็มหรือไม่
    {
        printf("ERROR STACK OVER FLOW!!!...\n");
    }
    else
    {
        SP=SP+1; //เพิ่มค่า SP
        ValOperandST[SP]=ValOperand; //ใส่ข้อมูลเข้าไปใน stack
    }
}
```

ฟังก์ชันนี้ใช้ในการดัน (push) โอเปอเรเตอร์ลงในสแต็ก โดยตรวจสอบว่าสแต็กเต็มหรือไม่ หากเต็มจะแสดงข้อผิดพลาด หากไม่เต็ม จะเพิ่มค่า SP ขึ้นหนึ่งและเก็บค่า ValOperand ลงในตำแหน่งใหม่ของ stack จากการอธิบายข้างต้น สามารถสรุปรายละเอียดได้ ดังนี้

- ตรวจสอบว่าสแต็กเต็มหรือไม่ (SP == MaxStack)
- หากเต็มจะแสดงข้อความ "ERROR STACK OVER FLOW!!!"
- หากไม่เต็ม จะเพิ่มค่า SP ขึ้นหนึ่งและเก็บค่า ValOperand ลงในตำแหน่งใหม่ของ stack

หลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable (ต่อ)

3. ฟังก์ชัน Pop

```
float pop() //ฟังก์ชัน POP
{
    float ValOperand;
    if (SP != 0) //ตรวจสอบว่า stack ไม่ว่างหรือไม่
    {
        ValOperand=ValOperandST[SP]; //รับข้อมูลจาก stack

        SP--; //ลดค่า SP

        return (ValOperand); //ส่งคืนข้อมูล
    }
    else
        printf("\nERROR STACK UNDER FLOW!!!!...\n");
}
```

ฟังก์ชันนี้ใช้ในการดึง (pop) โอเปอเรเตอร์ออกจากสแต็ก โดยตรวจสอบว่าสแต็กไม่ว่างเปล่า หากไม่ว่างเปล่า จะดึงโอเปอเรเตอร์ออกและลดค่า SP จากนั้นคืนค่าโอเปอเรเตอร์ที่ดึงออกมา หากสแต็กว่างเปล่า จะแสดงข้อผิดพลาด จากการอธิบายข้างต้น สามารถสรุปรายละเอียดได้ ดังนี้

- ตรวจสอบว่าสแต็กไม่ว่าง (SP != 0)
- หากไม่ว่าง จะดึงค่าจากตำแหน่งปัจจุบันของ stack มาเก็บใน ValOperand ลดค่า SP ลงหนึ่งและส่งคืนค่า ValOperand
- หากว่าง จะแสดงข้อความ "ERROR STACK UNDER FLOW!!!"

หลักการการทำงานของโปรแกรม Calculate Postfix by assigned in variable (ต่อ)

4. ฟังก์ชัน CalPostfix

```
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++) //กำหนดข้อมูลให้กับ OPERAND
    {
        ch=postfix[i]; //แยกตัวอักษรเพื่อกำหนดข้อมูล
        if (strchr("+-* /^", ch)==0) //ตรวจสอบว่าเป็น OPERAND หรือไม่
        {
            printf("\nAssign Number to %c : ", ch);
            scanf("%f", &ValPostfix[i]); //อ่านข้อมูลจากคีย์บอร์ดและกำหนดค่าให้กับ OPERAND ใน
            array โดยตรง
        }
    }
    for (i=0; i<=len-1; i++) //คำนวณค่าของ POSTFIX
    {
        ch=postfix[i]; //แยกตัวอักษรเพื่อเตรียมเข้า stack
        if (strchr("+-* /^", ch)==0) //ตรวจสอบว่าเป็น OPERAND หรือไม่
        {
            push(ValPostfix[i]); //push ค่า OPERAND เข้า stack
        }
        else
        {
            pop1=pop(); //pop ครั้งที่ 1
            pop2=pop(); //pop ครั้งที่ 2
```

หลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable (ต่อ)

4. ฟังก์ชัน CalPostfix (ต่อ)

```

        switch (ch)
        {
            case '+' : value=pop2+pop1; //คำนวณ

                push(value); //push ค่าเข้า 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;
        }

    } //จบ IF

} //จบ IF

printf("\nANS = %f",pop()); //ค่าสุดท้ายคือคำตอบ

} //จบฟังก์ชัน

```

ฟังก์ชัน CalPostfix ทำหน้าที่คำนวณค่าโพสต์ฟิกซ์นิพจน์:

- ประกาศตัวแปร pop1, pop2, และ value สำหรับเก็บค่าชั่วคราว
- คำนวณความยาวของโพสต์ฟิกซ์นิพจน์
- พิมพ์โพสต์ฟิกซ์นิพจน์
- วนลูปแรกกำหนดค่าให้กับตัวดำเนินการ (operand):

- แยกตัวอักษรแต่ละตัวจากนิพจน์
- หากเป็น operand (ไม่ใช่เครื่องหมาย $+$, $-$, $*$, $/$, $^$) จะอ่านค่าจากคีย์บอร์ดและเก็บใน ValPostfix
- วนลูปที่สองคำนวณค่าโพสต์ฟิกซ์นิพจน์:
 - แยกตัวอักษรแต่ละตัวจากนิพจน์
 - หากเป็น operand จะ push ค่าเข้า stack
 - หากเป็น operator จะ pop ค่าออกจาก stack สองครั้ง^๕และทำการคำนวณตาม operator นั้นๆ แล้ว push ค่าผลลัพธ์กลับเข้า stack
- แสดงผลลัพธ์สุดท้ายที่ pop ออกจาก stack

หลักการทำงานของโปรแกรม Calculate Postfix by assigned in variable (ต่อ)

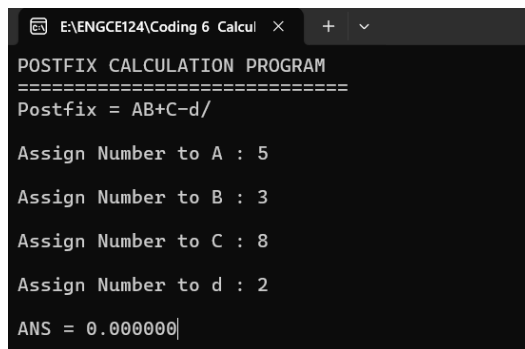
5. ฟังก์ชัน main

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

ฟังก์ชัน main เป็นฟังก์ชันหลักของโปรแกรมที่ทำการเรียกใช้ฟังก์ชัน CalPostfix เพื่อคำนวณนิพจน์อินฟิกซ์ที่กำหนดไว้ จากนั้นรอรับการกดปุ่มเพื่อสิ้นสุด จากการอธิบายข้างต้น สามารถสรุปรายละเอียดได้ ดังนี้

- พิมพ์ข้อความเริ่มต้นของโปรแกรม
- เรียกใช้ฟังก์ชัน CalPostfix โดยส่งนิพจน์โพสต์ฟิกซ์ที่กำหนดไว้ล่วงหน้า (postfix1)
- รอการกดปุ่มจากผู้ใช้ก่อนจบโปรแกรม

ผลลัพธ์การใช้งานโปรแกรม Calculate Postfix by assigned in variable



```

E:\ENGCE124\Coding 6 Calcul >
POSTFIX CALCULATION PROGRAM
=====
Postfix = AB+C-d/

Assign Number to A : 5
Assign Number to B : 3
Assign Number to C : 8
Assign Number to d : 2

ANS = 0.000000
  
```

จากผลลัพธ์โปรแกรมจะคำนวณโปสตรฟิกชันนิพจน์โดยใช้ stack เพื่อเก็บค่าตัวดำเนินการและผลลัพธ์ชั่วคราวตามลำดับการดำเนินการทางคณิตศาสตร์ โดยจะอ่านตัวอักษรแต่ละตัวในโปสตรฟิกชันนิพจน์ โดยถ้าหากเป็นตัวแปร (A, B, C, d) จะ push ค่าตัวแปรลงใน stack แต่ถ้าหากเป็นเครื่องหมายดำเนินการ (+, -, *, /, ^) จะ pop ค่าสองค่าจาก stack ทำการคำนวณตามเครื่องหมายนั้น และ push ผลลัพธ์กลับลงใน stack ซึ่งตัวอย่างการคำนวณสำหรับนิพจน์ $AB+C-d/$ กับค่าที่ผู้ใช้ป้อน ($A = 5, B = 3, C = 8, d = 2$) เป็นดังนี้:

1. จากนิพจน์: $AB+C-d/$
2. กำหนดให้ $A = 5$ และ $B = 3$
3. นำตัวเลข 5 และ 3 แทนลงในสมการจะได้ $5\ 3 + C-d/$
4. นำ $5 + 3 = 8$ (ผลลัพธ์จากการดำเนินการ +)
5. นำผลลัพธ์ที่ได้เท่ากับ 8 ใส่ลงในสแตกจะได้ $8\ C-d/$
6. กำหนดให้ $C = 8$
7. นำตัวเลข 8 แทนลงในสมการจะได้ $8\ 8 - d/$
8. นำ $8 - 8 = 0$ (ผลลัพธ์จากการดำเนินการ -)
9. นำผลลัพธ์ที่ได้เท่ากับ 8 ใส่ลงในสแตกจะได้ $0\ d/$
10. กำหนดให้ $d = 2$
11. นำตัวเลข 2 แทนลงในสมการจะได้ $0\ 2 /$
12. นำ $0 / 2 = 0$ (ผลลัพธ์จากการดำเนินการ /)

ผลลัพธ์ที่ได้จะมีค่าเท่ากับ 0 ซึ่งในโปรแกรมจะแสดงผล $ANS = 0.000000$ โดยสรุปโปรแกรมนี้ทำการคำนวณค่าโปสตรฟิกชันนิพจน์ที่กำหนดไว้ล่วงหน้า ($AB+C-d/$) โดยการใช้ stack เพื่อเก็บค่าตัวดำเนินการและผลลัพธ์ชั่วคราวในขณะที่ทำการคำนวณตามลำดับการดำเนินการทางคณิตศาสตร์. โปรแกรมจะขอให้ผู้ใช้ป้อนค่าตัวแปรที่เกี่ยวข้อง และแสดงผลลัพธ์สุดท้ายของการคำนวณบนหน้าจอ.

บรรณานุกรม

ChatGPT. (-). Calculate Postfix Expressions Using Stack in C Programming. สืบค้น 1 สิงหาคม 2567,
จาก <https://chatgpt.com/>