รายงาน

INFIX - POSTFIX

จัดทำโดย

นายธนรัก ชุ่มสวัสดิ์

รหัสศึกษา 68543210018-6

อาจารย์ผู้สอน

นายปิยพล ยืนยงสถาวร

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Data Structures and Algorithms

สาขาวิชาวิศวกรรมซอฟต์แวร์ มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา

ภาคเรียนที่ 2 ปีการศึกษา 2568

Code Program Stack

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h> // ต้องใช้สำหรับฟังก์ชัน pow (ยกกำลัง)

#define MAX 100

char opStack[MAX];
int opTop = -1;

float valStack[MAX];
int valTop = -1;

void pushOp(char ch) {
    if (opTop >= MAX - 1) printf("Stack Overflow\n");
    else opStack[++opTop] = ch;
}

char popOp() {
    if (opTop == -1) return -1;
    else return opStack[opTop--];
}

void pushVal(float val) {
    if (valTop >= MAX - 1) printf("Val Stack Overflow\n");
    else valStack[++valTop] = val;
}

float popVal() {
    if (valTop == -1) return 0;
    else return valStack[valTop--];
}

int isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^');
}

int isOperand(char ch) {
```

```
    return (isalpha(ch));
}

int precedenceIP(char op) {
    switch(op) {
        case '+': case '-': return 1;
        case '*': case '/': return 2;
        case '^': return 4;
        case '(': return 4;
        default: return 0;
    }
}

int precedenceST(char op) {
    switch(op) {
        case '+': case '-': return 1;
        case '*': case '/': return 2;
        case '^': return 3;
        case '(': return 0;
        default: return 0;
    }
}

void infixToPostfix() {
    char infix[MAX], postfix[MAX];
    int i = 0, j = 0;
    char ch;

    opTop = -1;

    printf("\n--- 1. INFIX TO POSTFIX ---\n");
    printf("Enter Infix Expression (e.g., A+B*C): ");
    scanf("%s", infix);

    int len = strlen(infix);
    for (i = 0; i < len; i++) {
        ch = infix[i];

        if (isOperand(ch) || isdigit(ch)) {
```

```
            postfix[j++] = ch;
        } else if (ch == '(') {
            pushOp(ch);
        } else if (ch == ')') {
            while (opTop != -1 && opStack[opTop] != '(') {
                postfix[j++] = popOp();
            }
            popOp();
        } else if (isOperator(ch)) {
            while (opTop != -1 && precedenceIP(ch) <= precedenceST(opStack[opTop])) {
                postfix[j++] = popOp();
            }
            pushOp(ch);
        }
    }

    while (opTop != -1) {
        postfix[j++] = popOp();
    }
    postfix[j] = '\0';
    printf("Result Postfix: %s\n\n", postfix);
}

float calculate(float op1, float op2, char operator) {
    switch(operator) {
        case '+': return op1 + op2;
        case '-': return op1 - op2;
        case '*': return op1 * op2;
        case '/': return op1 / op2;
        case '^': return pow(op1, op2);
        default: return 0;
    }
}

void evaluatePostfix() {
    char postfix[MAX];
    int i, len;
    float val, op1, op2, result;
```

```
    valTop = -1;

    printf("\n--- 2. EVALUATE POSTFIX ---\n");
    printf("Enter Postfix Expression (e.g., AB+C*): ");
    scanf("%s", postfix);

    len = strlen(postfix);
    for (i = 0; i < len; i++) {
        char ch = postfix[i];

        if (isOperand(ch)) {
            printf("Enter value for %c: ", ch);
            scanf("%f", &val);
            pushVal(val);
        }
        else if (isdigit(ch)) {
            pushVal((float)(ch - '0'));
        }
        else if (isOperator(ch)) {
            op2 = popVal();
            op1 = popVal();

            result = calculate(op1, op2, ch);
            pushVal(result);
        }
    }

    printf("Final Result: %.2f\n\n", popVal());
}

int main() {
    int choice;
    do {
        printf("==================================\n");
        printf("      DATA STRUCTURE MENU        \n");
        printf("==================================\n");
        printf("1. Convert Infix -> Postfix\n");
        printf("2. Calculate Postfix Result\n");
        printf("0. Exit\n");
```

```
        printf("Select menu: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: infixToPostfix(); break;
            case 2: evaluatePostfix(); break;
            case 0: printf("Exiting...\n"); break;
            default: printf("Invalid choice!\n");
        }
    } while (choice != 0);
    return 0;
}
```

ส่วนที่มีการแก้ไขและเพิ่มเติม

1. แก้ไข Syntax Error ใน Switch Case และเพิ่มเครื่องหมายตกหล่น

ฟังก์ชัน precedenceIP เขียน case '- :ผิดไวยากรณ์ (ขาด single quote ปิด) และขาดการระบุ case '^': ที่ชัดเจนทั้งที่ในทฤษฎีระบุว่ามี โค้ดใหม่จึงแก้ไขเครื่องหมายให้ถูกต้องและเพิ่ม case '^' ให้ครบถ้วนเพื่อให้รองรับการยกกำลังตามทฤษฎี

```
switch(op) {
    case '+': case '-': return 1;
    case '*': case '/': return 2;
    case '^': return 4;
    case '(': return 4;
    default: return 0;
 }
```

2. ปรับปรุงตรรกะ Loop และเงื่อนไข If-Else (Logic Fix)

โค้ดเดิมในเอกสารมีบรรทัด if (SP==0) else ซึ่งผิดหลักภาษา C และการจัดลำดับการเช็กวงเล็บปิด) ทำได้สับสน โค้ดใหม่จึงจัดโครงสร้าง if-else if ใหม่ แยกกรณี Operand, วงเล็บเปิด, วงเล็บปิด และ Operator ออกจากกันให้ชัดเจน เพื่อให้ Stack ทำงานสัมพันธ์กับค่า Priority ได้ถูกต้องตามอัลกอริทึม

```
if (isOperand(ch)) {
        postfix[j++] = ch;
    }
    else if (ch == '(') {
        pushOp(ch);
    }
    else if (ch == ')') {
        while (opTop != -1 && opStack[opTop] != '(') {
            postfix[j++] = popOp();
        }
     }
    else if (isOperator(ch)) {
        while (opTop != -1 && precedenceIP(ch) <= precedenceST(opStack[opTop])) {
            postfix[j++] = popOp();
```

```
        }
        pushOp(ch);
    }
```

3. เปลี่ยนวิธีการรับค่า (Input Method)

โค้ดเดิมกำหนดค่าตายตัวในตัวแปร infix1 ทำให้คำนวณได้แค่โจทย์เดียว โค้ดใหม่เปลี่ยนมาใช้ scanf รับค่าจากคีย์บอร์ด เพื่อให้ผู้ใช้ป้อนสมการทางคณิตศาสตร์ใดๆ ก็ได้ และเพิ่มเมนู do-while ใน main เพื่อให้โปรแกรมรันค้างไว้รอรับคำสั่งใหม่ได้เรื่อยๆ โดยไม่ต้องเปิดปิดโปรแกรมใหม่

```
printf("\n=== INFIX TO POSTFIX CONVERTER ===\n");
 printf("Enter Infix Expression (e.g., A+B*C): ");
 scanf("%s", infix);
```

4. รวมโปรแกรมและสร้างฟังก์ชันคำนวณค่า (Function Integration)

จากเดิมเอกสารแยกโค้ดเป็นส่วนๆ และ Input ของการคำนวณในเอกสาร (หน้า 12) ใช้วิธีวนลูปรับค่าใส่ Array ไว้ก่อน ซึ่งซับซ้อนเกินจำเป็น โค้ดใหม่จึงยุบรวมเป็น 2 เมนูในไฟล์เดียว และแก้ไขวิธีรับค่าในฟังก์ชัน evaluatePostfix โดยเมื่อเจอตัวแปร (เช่น A, B) โปรแกรมจะถามค่าจากผู้ใช้ทันที (Interactive Prompt) แล้ว pushVal ลง Stack ทำให้โค้ดกระชับและเข้าใจง่ายขึ้น

```
        printf("Enter value for %c: ", ch);
        scanf("%f", &val);
        pushVal(val);
```

5. แก้ไขลำดับการ Pop และคำนวณ (Evaluation Logic Fix)

มีการเขียน value=pop2-pop1 ซึ่งอาจทำให้สับสนเรื่องลำดับตัวตั้งตัวลบ โค้ดใหม่ระบุชัดเจนว่า op2 = popVal() คือตัวที่อยู่บนสุด (ตัวกระทำ) และ op1 = popVal() คือตัวถัดมา (ตัวตั้ง) แล้วจึงส่งเข้าฟังก์ชัน calculate(op1, op2, ch) เพื่อให้มั่นใจว่าการลบและการหาร (A−B หรือ A/B) ได้ผลลัพธ์ถูกต้องตามหลักคณิตศาสตร์

```
        op2 = popVal();
        op1 = popVal();
```

```
        result = calculate(op1, op2, ch);
```

อธิบายการทำงานของโปรแกรม

1. main() Function

เมื่อโปรแกรมเริ่มต้น ตัวแปร choice จะรอรับค่าจากการกดเลือกเมนูของผู้ใช้ หากผู้ใช้เลือก '1' โปรแกรมจะเรียกฟังก์ชัน infixToPostfix ให้เริ่มทำงานเพื่อแปลงนิพจน์ แต่ถ้าเลือก '0' ลูปจะจบลงและโปรแกรมจะปิดตัวลง

```
int main() {
    int choice;

    do {
        printf("=====================================\n");
        printf("      INFIX CONVERSION PROGRAM      \n");
        printf("=====================================\n");
        printf("1. Convert Infix -> Postfix\n");
        printf("0. Exit\n");
        printf("Select menu: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                infixToPostfix();
                break;
            case 0:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 0);
```

```
    return 0;
}
```

2. infixToPostfix() Function

ฟังก์ชันนี้จะรับค่าสมการ Infix เข้ามาในตัวแปร infix จากนั้นตัวแปร i จะวนลูปอ่านตัวอักษรทีละตัว หากตัวอักษรนั้นเป็นตัวเลข (Operand) จะถูกส่งไปเก็บต่อท้ายในตัวแปร postfix ทันที แต่ถ้าเป็นเครื่องหมาย (Operator) จะถูกส่งไปเปรียบเทียบค่าความสำคัญผ่านฟังก์ชัน precedenceIP และ precedenceST เพื่อตัดสินใจว่าจะเรียก pushOp หรือ popOp ในการจัดการลำดับก่อนหลัง เมื่อวนลูปครบทุกตัวแปรแล้ว ผลลัพธ์สุดท้ายใน postfix จะถูกแสดงผลออกมา

```
void infixToPostfix() {
    char infix[MAX], postfix[MAX];
    int i = 0, j = 0;
    char ch;

    opTop = -1;

    printf("\n=== INFIX TO POSTFIX CONVERTER ===\n");
    printf("Enter Infix Expression (e.g., A+B*C): ");
    scanf("%s", infix);

    int len = strlen(infix);
    for (i = 0; i < len; i++) {
        ch = infix[i];

        if (isOperand(ch)) {
            postfix[j++] = ch;
        }
        else if (ch == '(') {
```

```
            pushOp(ch);
        }
        else if (ch == ')') {
            while (opTop != -1 && opStack[opTop] != '(') {
                postfix[j++] = popOp();
            }
            popOp();
        }
        else if (isOperator(ch)) {
            while (opTop != -1 && precedenceIP(ch) <= precedenceST(opStack[opTop])) {
                postfix[j++] = popOp();
            }
            pushOp(ch);
        }
    }
    while (opTop != -1) {
        postfix[j++] = popOp();
    }
    postfix[j] = '\0';

    printf("Result Postfix: %s\n\n", postfix);
}
```

3. pushOp(char ch) Function

เมื่อฟังก์ชันหลักต้องการเก็บเครื่องหมายลง Stack ตัวแปร ch จะถูกส่งเข้ามาที่นี่ ฟังก์ชันจะตรวจสอบว่า Stack เต็มหรือไม่ หากไม่เต็ม ตัวแปร opTop จะขยับค่าขึ้น 1 ตำแหน่ง แล้วนำค่า ch ไปบันทึกลงในอาร์เรย์ opStack เพื่อพักไว้รอใช้งานต่อ

```
void pushOp(char ch) {
    if (opTop >= MAX - 1) {
        printf("Stack Overflow\n");
    } else {
        opStack[++opTop] = ch;
    }
}
```

4. popOp() Function

เมื่อฟังก์ชันหลักต้องการดึงเครื่องหมายออกจาก Stack เพื่อนำไปต่อท้ายผลลัพธ์ ฟังก์ชันนี้จะไปดึงค่าจากอาร์เรย์ opStack ณ ตำแหน่ง opTop ปัจจุบันออกมาเก็บในตัวแปรชั่วคราว จากนั้นลดค่า opTop ลง 1 ขั้น แล้วส่งคืน (return) ตัวอักษรนั้นกลับไปยังฟังก์ชัน infixToPostfix

```
char popOp() {
    if (opTop == -1) {
        return -1;
    } else {
        return opStack[opTop--];
    }
}
```

5. isOperator(char ch) Function และ isOperand(char ch) Function

เมื่อตัวอักษร ch ถูกส่งเข้ามา ฟังก์ชันนี้จะตรวจสอบว่าเป็นเครื่องหมายทางคณิตศาสตร์ หรือเป็นตัวเลข/ตัวอักษร แล้วส่งค่าจริง (True/1) หรือ เท็จ (False/0) กลับไปให้ infixToPostfix ใช้ตัดสินใจว่าจะเข้าเงื่อนไข if ข้อไหน

```
int isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^');
}

int isOperand(char ch) {
    return (isalnum(ch));
}
```

6. precedenceIP(char op) Function และ precedenceST(char op) Function

เมื่อมีเครื่องหมายถูกส่งเข้ามาในตัวแปร op ฟังก์ชันจะนำไปเทียบกับตาราง Switch Case (เช่น ถ้าเป็น * ได้ค่า 2, ถ้าเป็น + ได้ค่า 1) แล้วส่งค่าตัวเลขจำนวนเต็ม (Integer) กลับไปยังฟังก์ชัน infixToPostfix เพื่อใช้เปรียบเทียบว่าเครื่องหมายใหม่ที่เข้ามา หรือเครื่องหมายเก่าใน Stack ใครมีอำนาจมากกว่ากัน

```
int precedenceIP(char op) {
    switch(op) {
        case '+': case '-': return 1;
        case '*': case '/': return 2;
        case '^': return 4; // ยกกำลังมีความสำคัญสูงสุดเมื่อเข้ามาใหม่
```

```
        case '(': return 4; // วงเล็บเปิดมีความสำคัญสูงสุดเพื่อให้ Push ลง Stack ได้ทันที
        default: return 0;
    }
}

int precedenceST(char op) {
    switch(op) {
        case '+': case '-': return 1;
        case '*': case '/': return 2;
        case '^': return 3; // เมื่ออยู่ใน Stack ความสำคัญลดลง (เพื่อให้รองรับ Right Associative)
        case '(': return 0; // วงเล็บเปิดใน Stack มีค่าน้อยที่สุด
        default: return 0;
    }
}
```

7. evaluatePostfix()

ฟังก์ชันนี้จะรับนิพจน์ Postfix เข้ามาทีละตัวอักษร หากตัวแปร ch เป็นตัวอักษร (A-Z) โปรแกรมจะหยุดถามผู้ใช้เพื่อขอค่าตัวเลข แล้วส่งค่านั้นเข้าฟังก์ชัน pushVal เพื่อเก็บลง Stack แต่ถ้า ch เป็นเครื่องหมาย (Operator) โปรแกรมจะเรียก popVal สองครั้งเพื่อดึงตัวเลข 2 ตัวล่าสุดออกมาเก็บ

ใน op2 และ op1 จากนั้นส่งทั้งสามค่าไปยัง calculate เพื่อหาผลลัพธ์ และนำผลที่ได้ pushVal กลับลง Stack เพื่อรอคำนวณต่อ

```
void evaluatePostfix() {
    char postfix[MAX];
    int i, len;
    float val, op1, op2, result;

    valTop = -1;

    printf("\n--- 2. EVALUATE POSTFIX ---\n");
    printf("Enter Postfix Expression (e.g., AB+C*): ");
    scanf("%s", postfix);

    len = strlen(postfix);
    for (i = 0; i < len; i++) {
        char ch = postfix[i];

        if (isOperand(ch)) {
            printf("Enter value for %c: ", ch);
            scanf("%f", &val);
            pushVal(val);
        }
        else if (isdigit(ch)) {
            pushVal((float)(ch - '0'));
        }
        else if (isOperator(ch)) {
            op2 = popVal();
            op1 = popVal();

            result = calculate(op1, op2, ch);
            pushVal(result);
        }
    }
```

8. calculate(float op1, float op2, char operator)

เมื่อได้รับตัวเลขสองจำนวนและเครื่องหมาย ฟังก์ชันนี้จะทำหน้าที่เป็นเครื่องคิดเลข โดยใช้ switch-case ตรวจสอบเครื่องหมาย หากเป็น + ก็นำ op1 + op2 หรือหากเป็น ^ ก็ใช้ฟังก์ชัน pow(op1, op2) เมื่อได้ผลลัพธ์แล้วจะส่งค่า (return) กลับไปยัง evaluatePostfix ทันที

```
float calculate(float op1, float op2, char operator) {
    switch(operator) {
        case '+': return op1 + op2;
        case '-': return op1 - op2;
        case '*': return op1 * op2;
        case '/': return op1 / op2;
        case '^': return pow(op1, op2);
        default: return 0;
    }
}
```

9. pushVal(float val) และ popVal()

สองฟังก์ชันนี้ทำหน้าที่จัดการ valStack (Stack สำหรับตัวเลขทศนิยม) เมื่อ pushVal ทำงาน มันจะขยับ valTop ขึ้นและบันทึกค่าตัวเลขลงไป ในทางกลับกัน เมื่อ popVal ทำงาน มันจะดึงตัวเลขล่าสุดออกมาแล้วลด valTop ลง เพื่อส่งค่านั้นไปใช้ในการคำนวณ

```
void pushVal(float val) {
    if (valTop >= MAX - 1) printf("Val Stack Overflow\n");
    else valStack[++valTop] = val;
}

float popVal() {
    if (valTop == -1) return 0;
    else return valStack[valTop--];
}
```

ผลลัพธ์ของโปรแกรม

Convert Infix -> Postfix

```
================================
      DATA STRUCTURE MENU
================================
1. Convert Infix -> Postfix
2. Calculate Postfix Result
0. Exit
Select menu: 1

--- 1. INFIX TO POSTFIX ---
Enter Infix Expression (e.g., A+B*C): ABCD^E*F/*+G-
Result Postfix: ABCDE^F*/*G+-
```

Calculate Postfix Result

```
================================
      DATA STRUCTURE MENU
================================
1. Convert Infix -> Postfix
2. Calculate Postfix Result
0. Exit
Select menu: 2

--- 2. EVALUATE POSTFIX ---
Enter Postfix Expression (e.g., AB+C*): ABCDE^F*/*G+-
Enter value for A: 10
Enter value for B: 20
Enter value for C: 30
Enter value for D: 40
Enter value for E: 50
Enter value for F: 60
Enter value for G: 70
Final Result: -60.00

================================
      DATA STRUCTURE MENU
================================
1. Convert Infix -> Postfix
2. Calculate Postfix Result
0. Exit
Select menu: 0
Exiting...
```