โครงงานฝึกทักษะการเขียนโปรแกรม ครั้งที่ 3
โปรแกรมแปลงนิพจน์ Infix to Postfix
ด้วยเครื่องมือ Stack

นายจิรเมธ พั่วพันธ์
รหัสนิสิต 60160157

เสนอ
ผู้ช่วยศาสตราจารย์นวลศรี เด่นวัฒนา
อาจารย์พจน์สพร แซ่ลิ้ม

โครงงานนี้เป็นส่วนหนึ่งของ
รายวิชา 88823359 โครงสร้างข้อมูลและขั้นตอนวิธีสำหรับวิศวกรรมซอฟต์แวร์
ภาคการศึกษาที่ 1 ปีการศึกษา 2561 สาขาวิชาวิศวกรรมซอฟต์แวร์
คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

สารบัญ

เรื่อง หน้า

สารบัญรูปภาพ

ภาพที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

ส่วนที่ 1
การวิเคราะห์โจทย์ปัญหา

1.1 โจทย์ปัญหา

โปรแกรมสำหรับแปลงนิพจน์ Infix to Postfix ด้วยเครื่องมือ Stack Linked List

ความสามารถของโปรแกรม

1. ผู้ใช้งานสามารถแปลงนิพจน์จาก Infix to Postfix ได้

1.2 วิเคราะห์แนวทางแก้ไขปัญหา

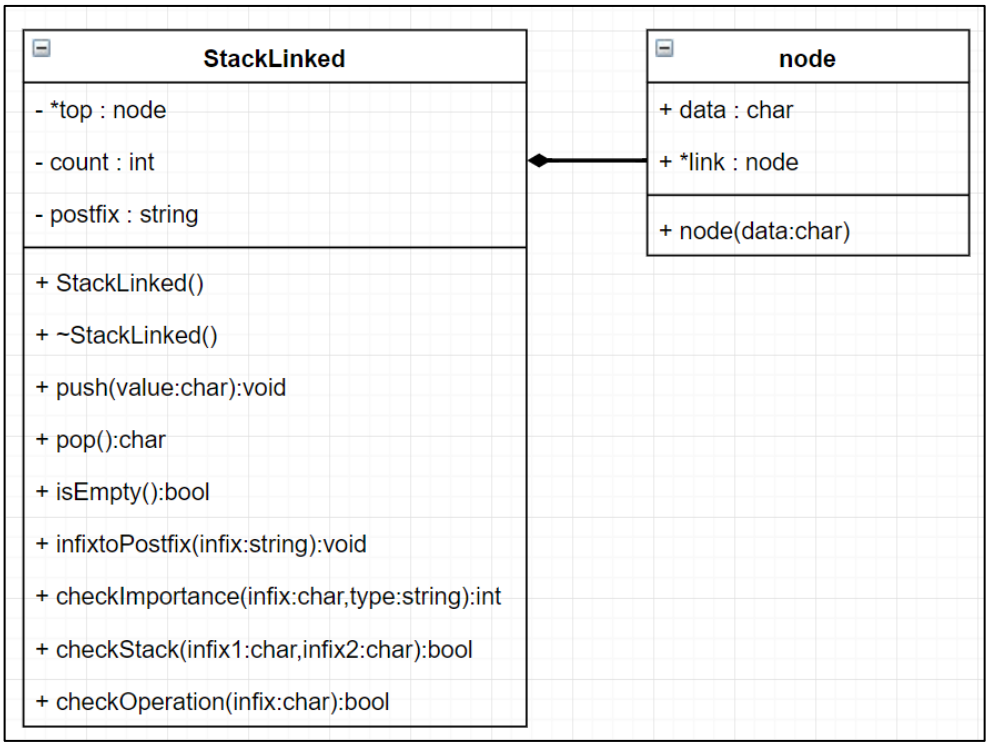
จากโจทย์วิเคราะห์ได้ว่าควรจะมีการสร้างคลาสจำนวน 2 คลาส ในคลาสที่หนึ่ง เป็นคลาส StackLinked ใช้จัดการชุดของข้อมูลที่เรียงต่อกัน และจัดลำดับการเก็บข้อมูลแบบเข้าทีหลังและนำออกก่อน ซึ่งควรมี Attribute (ตัวแปร) ที่เป็นชนิด string เพื่อเก็บข้อมูลที่เป็นอักขระ และชนิด Int เพื่อเก็บข้อมูลที่เป็นจำนวนเต็ม ในคลาสที่สอง เป็นคลาส node จะประกอบไปด้วย Attribute (ตัวแปร) ที่เป็นชนิด char ข้อมูลที่เป็นอักขระ และpointer 1ตัว ใช้เป็นตัวชี้ไปยัง node ถัดไป

1.3 การออกแบบ

จากการวิเคราะห์นำมาสร้างแผนภาพ ทำให้ได้แผนภาพคลาส ดังหัวข้อที่ 1.3.1

1.3.1 แผนภาพคลาส (Class Diagram)

แผนภาพคลาสแสดงได้ดังภาพที่ 1-1 แผนภาพคลาสของโครงงาน รวมถึงการแสดงค่าต่างๆ ในคลาสดังตารางที่ 1-1 ตารางที่ 1-2 แสดงส่วนประกอบของคลาส StackLinked และ node



ภาพที่ 1-1 แผนภาพคลาสของโครงงาน

ตารางที่ 1-1 แสดงส่วนประกอบของคลาส StackLinked

ข้อมูล	คำอธิบาย
Attribute	
* top: node	ตัวชี้สำหรับชี้ที่อยู่ของข้อมูลข้างบนสุด
postfix : string	ตัวแปรสำหรับเก็บค่า Postfix
int : count	ตัวแปรใช้นับจำนวนข้อมูลที่เก็บไว้ทั้งหมด
Method	
StackLinked ()	ค่า Constructor
~ StackLinked ()	ค่า Deconstructor
push(value:char) : void	เป็น Method สำหรับเพิ่มข้อมูลลง Stack
pop() : char	เป็น Method สำหรับนำข้อมูลบนสุดไปใช้ และลบออกจาก Stack
isEmpty() : bool	เป็น Method สำหรับเช็ค Stack ว่างหรือไม่
infixtoPostfix(infix:string) : void	เป็น Method สำหรับแปลงค่า infix
checkImportance(infix:char,type:string) : int	เป็น Method สำหรับเช็คลำดับความสำคัญของตัวดำเนินการ
checkStack(infix1:char, infix2:char) : bool	เป็น Method สำหรับเช็คตัวดำเนินการที่นำเข้ามามีลำดับความสำคัญน้อยกว่า
checkOperation(infix:char) : bool	เป็น Method สำหรับเช็คว่าเป็นตัวดำเนินการ

ตารางที่ 1-2 แสดงส่วนประกอบของคลาส node

ข้อมูล	คำอธิบาย
Attribute	
* link: node	ตัวชี้สำหรับชี้ที่อยู่ของข้อมูลลำดับถัดไป
data : char	ตัวแปรสำหรับเก็บข้อมูลที่เป็นอักขระ
Method	
node (data:char)	ค่า Constructor

1.4 รหัสเทียม

รหัสเทียมจะแสดงเขียนแยกตาม Method หรือ Module ของโปรแกรม

1.4.1 มอดูลการแสดงข้อมูลเมนู

อัลกอริทึม	มอดูลการแสดงเมนูของโปรแกรม
ข้อมูลนำเข้า	ไม่มี
ผลลัพธ์	แสดงเมนู

1. ประกาศตัวแปรชื่อ choice
2. ลูป
   2.1. แสดงเมนู
   2.2. เลือกเมนูในช่วงที่กำหนด
   2.3. ถ้าเลือกเมนูในช่วงที่กำหนด จบลูป
   2.4. ถ้าไม่ กลับไปที่ 2.1
3. ทำงานตามเมนูที่ระบุ
4. จบการทำงาน

1.4.2 มอดูลการแปลงนิพจน์ (Infix to Postfix)

อัลกอริทึม	มอดูลการแปลงนิพจน์
ข้อมูลนำเข้า	รับค่า Infix จากแป้นพิมพ์
ผลลัพธ์	แสดง Postfix ออกทางหน้าจอ

1. กำหนดค่า postfix ให้เป็นค่าว่าง
2. ประกาศตัวแปร i เป็นชนิดจำนวนเต็ม และกำหนดค่าให้เท่ากับ 0
3. เข้าลูปทำซ้ำ โดยมีเงื่อนไข i น้อยกว่า ความยาวอักขระของ infix
   3.1 เงื่อนไข เรียกใช้ฟังก์ชัน checkOperation โดยมีพารามิเตอร์คือ infix ตำแหน่งอาเรย์ช่องที่ i
   3.2 หากเป็นจริง
      3.2.1 เงื่อนไข infix ตำแหน่งอาเรย์ช่องที่ i มีค่าเท่ากับ “ (”
         3.2.1.1 เรียกใช้ฟังก์ชัน push โดยมีพารามิเตอร์คือ infix ตำแหน่งอาเรย์ช่องที่ i
      3.2.2 เงื่อนไข infix ตำแหน่งอาเรย์ช่องที่ i มีค่าเท่ากับ “) ”
         3.2.2.1 เข้าลูป เงื่อนไข count ไม่เท่ากับ 0 และ top ชี้ที่ data ไม่เท่ากับ “ (”
            • กำหนดค่าให้ postfix มีข้อมูลต่อเพิ่มจากการเรียกใช้ฟังก์ชัน pop
         3.2.2.2 เรียกใช้ฟังก์ชัน pop

3.2.3 เงื่อนไข Else

3.2.3.1 เข้าลูป เงื่อนไข count ไม่เท่ากับ 0 และ เรียกใช้ฟังก์ชัน checkStack โดยมีพารามิเตอร์คือ infix ตำแหน่งอาเรย์ช่องที่ I และ top ชี้ที่ data

• กำหนดค่าให้ postfix มีข้อมูลต่อเพิ่มจากการเรียกใช้ฟังก์ชัน pop

3.2.3.2 เรียกใช้ฟังก์ชัน push โดยมีพารามิเตอร์คือ infix ตำแหน่งอาเรย์ช่องที่ i

3.3 หากเป็นเท็จ

3.3.1 กำหนดค่าให้ postfix มีข้อมูลต่อเพิ่มจาก infix ตำแหน่งอาเรย์ช่องที่ i

4. เข้าลูป เงื่อนไข count ไม่เท่ากับ 0

4.1 กำหนดค่าให้ postfix มีข้อมูลต่อเพิ่มจากการเรียกใช้ฟังก์ชัน pop

5. แสดงค่า postfix ออกทางหน้าจอ

6. จบการทำงาน

1.4.3 มอดูลเปรียบเทียบค่าของตัวดำเนินการ (checkStack)

อัลกอริทึม	มอดูลเปรียบเทียบค่าของตัวดำเนินการ
ข้อมูลนำเข้า	infix1 และ infix2 ที่ส่งมาเป็นพารามิเตอร์
ผลลัพธ์	คืนค่า true หรือ flase

1. เงื่อนไข เรียกฟังก์ชัน checkImportance โดยมีพารามิเตอร์คือ infix1 และ ค่า “out” เปรียบเทียบน้อยกว่า ฟังก์ชัน checkImportance โดยมีพารามิเตอร์คือ infix2 และ ค่า “in”

1.1 น้อยกว่าจริง คืนค่า true

1.2 น้อยกว่าเท็จ คืนค่า false

2. จบการทำงาน

1.4.4 มอดูลเช็คค่าความสำคัญของตัวดำเนินการ (checkImportance)

อัลกอริทึม	มอดูลเช็คความสำคัญของตัวดำเนินการ
ข้อมูลนำเข้า	infix1 และ type ที่ส่งมาเป็นพารามิเตอร์
ผลลัพธ์	คืนค่าความสำคัญเป็นจำนวนเต็ม

1. เงื่อนไข type มีค่าเท่ากับ “out”
   1.1 เงื่อนไข infix มีค่าเท่ากับ “+” หรือ “-”
      1.1.1 คืนค่า 1
   1.2 เงื่อนไข infix มีค่าเท่ากับ “/” หรือ “*”
      1.2.1 คืนค่า 3
   1.3 เงื่อนไข infix มีค่าเท่ากับ “^”
      1.3.1 คืนค่า 6
   1.4 เงื่อนไข infix มีค่าเท่ากับ “(”
      1.4.1 คืนค่า 7
2. เงื่อนไข type มีค่าเท่ากับ “out”
   2.1 เงื่อนไข infix มีค่าเท่ากับ “+” หรือ “-”
      2.1.1 คืนค่า 2
   2.2 เงื่อนไข infix มีค่าเท่ากับ “/” หรือ “*”
      2.2.1 คืนค่า 4
   2.3 เงื่อนไข infix มีค่าเท่ากับ “^”
      2.3.1 คืนค่า 5
   2.4 เงื่อนไข infix มีค่าเท่ากับ “(”
      2.4.1 คืนค่า 0
3. จบการทำงาน

1.4.5 มอดูลเช็คตัวดำเนินการ (checkOperation)

อัลกอริทึม	มอดูลเช็คตัวดำเนินการ
ข้อมูลนำเข้า	infix ที่ส่งมาเป็นพารามิเตอร์
ผลลัพธ์	คืนค่า true หรือ flase

1. เงื่อนไข infix มีค่าเท่ากับ บวก หรือ ลบ หรือ คูณ หรือ หาร หรือ ยกกำลัง หรือ วงเล็บเปิด หรือ วงเล็บปิด
   1.1 เป็นจริง คืนค่า true
   1.2 เป็นเท็จ คืนค่า false
2. จบการทำงาน

ส่วนที่ 2
รายละเอียดของผลลัพธ์

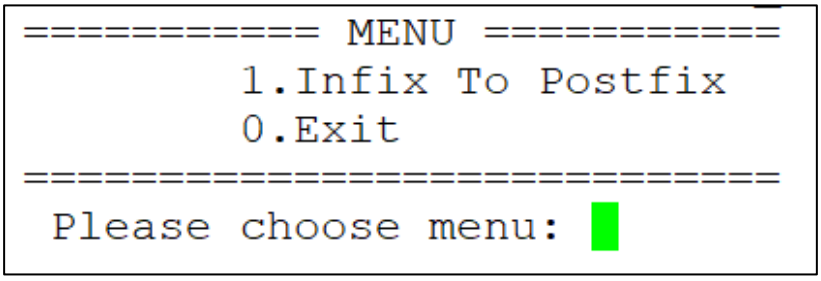
2.1 หน้าจอระบบ

2.1.1 การแสดงผลของเมนูระบบ

เมื่อเริ่มเปิดโปรแกรม โปรแกรมจะแสดงข้อมูลเมนูระบบ ดังภาพที่ 2 – 1 โดย การใช้งานคือ ให้ระบุเลขตัวเลือก 0 หรือ 1 โดยแต่ละตัวเลขที่การทำงานที่แตกต่างการ ดังนี้

0. Exit คือ ออกจากโปรแกรม

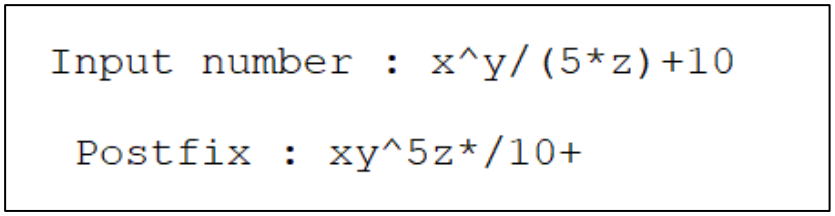
1. Infix To Postfix คือ การแปลง Infix to Postfix



ภาพที่ 2-1 ส่วนการแสดงผลของเมนูระบบ

2.1.2 การแปลง Infix to Postfix

เมื่อเข้าสู่เมนูที่ 1 โปรแกรมจะแสดงผล ดังภาพที่ 2 – 2 โดย การใช้งานคือ ให้ระบุนิพจน์ที่ต้องการแปลงที่ Input number :



ภาพที่ 2-2 ส่วนการแปลง Infix to Postfix

เมื่อทำการแปลง Infix สำเร็จ จะได้ Postfix ที่แปลงเรียบร้อย ดังภาพที่ 2 – 2 ตรงที่ Postfix :

ส่วนที่ 3
การวิเคราะห์และประเมินผลลัพธ์

3.1 ประสิทธิภาพของโปรแกรม

เป็นส่วนการวิเคราะห์ข้อมูล ที่ได้จากผลลัพธ์ของโปรแกรม ซึ่งประกอบไปด้วยมอดูลการแปลงนิพจน์ โดยมีการวิเคราะห์ได้ดังนี้

3.1.1 มอดูลการแปลงนิพจน์

ในส่วนมอดูลการแปลงนิพจน์ มีลักษณะการทำงานตามโปรแกรมที่แสดงอยู่ ดังภาพที่ 3-1 และภาพที่ 3-2 และทำการวิเคราะห์ได้ดังตารางที่ 3 – 1 และ ตารางที่ 3 – 2

```
for(int i=0; i<infix.length(); i++) {
        if(checkOperation(infix[i])) {
                if(infix[i]=='(') {
                        push(infix[i]);
                } else if(infix[i]==')') {
                        while(!isEmpty() && top->data != '(') {
                                postfix += pop();
                        }
                        pop();
                } else {
                        while(!isEmpty() && checkStack(infix[i],top->data)) {
                                postfix += pop();
                        }
                        push(infix[i]);
                }
        } else {
                postfix += infix[i];
        }
}
```

ภาพที่ 3-1 โปรแกรมส่วนของมอดูลการแปลงนิพจน์

ตัวอย่างนิพจน์เช่น A+B

ตารางที่ 3-1 แสดงการวิเคราะห์ประสิทธิภาพของการทำงานมอดูลการแปลงนิพจน์

ค่า i	เงื่อนไข I < infix.length()	เงื่อนไข checkOperation(infix[i]	แสดงผลภายในเงื่อนไข checkOperation(infix[i]	เงื่อนไข else	แสดงผลภายในเงื่อนไข else
1	/	/	X	/	/
2	/	/	/	X	X
3	/	/	X	/	/
4	/	X	X	X	X
จำนวนครั้งที่ทำ	4	3	1	2	2

สรุปได้ว่า Big-O ของการทำงานมอดูลนี้คือ BigO(n)

```
while(!isEmpty()) {
        postfix += pop();
}
```

ภาพที่ 3-2 โปรแกรมส่วนของมอดูลการแปลงนิพจน์

กำหนดให้ค่า count = 3

ตารางที่ 3-2 แสดงการวิเคราะห์ประสิทธิภาพของการทำงานมอดูลการแปลงนิพจน์

ค่า count	เงื่อนไข !isEmpty()	แสดงผลภายในเงื่อนไข
3	/	/
2	/	/
1	/	/
0	/	X
จำนวนครั้งที่ทำ	4	3

สรุปได้ว่า Big-O ของการทำงานมอดูลนี้คือ (n) + (n – 1) = 2n – 1 = Big O(n)

3.2 สรุปผลลัพธ์

จากการวิเคราะห์การทำงานของโปรแกรม พบว่ามอดูลทุกมอดูลมีประสิทธิภาพเท่ากันในโปรแกรม เนื่องจาก การทำงานมีเพียงการวนเพื่อเช็คเงื่อนไขเท่านั้น โปรแกรมจึงมีประสิทธิภาพที่ Big O(n)