Postfix Expression

Postfix expression คืออะไร

Postfix Expression เป็นวิธีการแสดงค่าคณิตศาสตร์โดยที่ตัวดำเนินการ (operators) ตามหลัง operand โดยไม่ต้องใช้วงเล็บหรือคำสั่งพิเศษในการกำหนดลำดับการดำเนินการ ซึ่งคำนวณใน Postfix Expression สามารถทำได้โดยอ่านจากซ้ายไปขวา และไม่ต้องกังวลเรื่องลำดับการดำเนินการ

Operators and Operand

Operator ใน Java นั้นก็คือสิ่งที่เอาไว้จัดการกับตัวแปรต่างๆ เช่น การ บวก ลบ คูณ หาร หรือการทำสิ่งต่างๆ

"Operand" หมายถึง ค่าหรือตัวแปรที่ถูกใช้ในการดำเนินการคำนวณตามรูปแบบ postfix โดยใส่ Operand ก่อนดำเนินการและตามด้วยตัวดำเนินการ (operator)

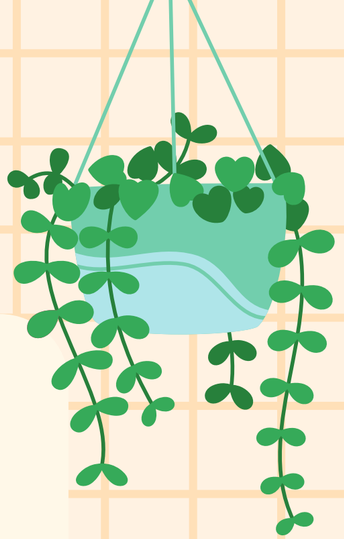
การใช้ operand แบบ postfix มักเป็นกรณีที่ใช้ในการคำนวณทางคณิตศาสตร์หรือการคำนวณทางคอมพิวเตอร์ที่อาศัยลำดับการดำเนินการในรูปแบบนี้เป็นส่วนหนึ่งของกระบวนการคำนวณ.

Arithmetic Operators

ก็คือ Operator ทางคณิตศาสตร์ เช่น การบวก ลบ คูณ หาร

Operator	Description	Example
+	การบวก	x = 5+5
-	การลบ	x = 10-5
*	การคูณ	y = 2*2
/	การหาร	y = 4/2
%มอด	การหารแล้วเอาเศษ	z = 11%2 จะได้ 1
++	การเพิ่มค่าของจำนวนเต็ม 1	x = x++;
--	การลดค่าของจำนวนเต็ม 1	y = y--;

Precedence



Precedence เป็นคุณสมบัติที่กำหนดลำดับความสำคัญของตัวดำเนินการ (operators) ในการคำนวณ Postfix Expression ในภาษา Java แบบ postfix โดยที่ตัวดำเนินการที่มี Precedence สูงกว่าจะถูกดำเนินการก่อนต่อจากตัวดำเนินการที่มี Precedence ต่ำกว่า

ตัวอย่างของ Precedence ในภาษา Java แบบ postfix มีดังนี้:

- Precedence สูงสุด: ตัวดำเนินการทางคณิตศาสตร์ เช่น *, /, %
- Precedence กลาง: ตัวดำเนินการทางคณิตศาสตร์ เช่น +, -
- Precedence ต่ำสุด: ตัวดำเนินการทางตรรกะ เช่น <, >, ==

Associativity

"Associativity" เป็นคุณสมบัติที่กำหนดลำดับการดำเนินการของตัวดำเนินการ (operators) เมื่อมีตัวดำเนินการหลายตัวในนิพจน์เดียว โดยวิธีที่ตัวดำเนินการจัดลำดับตัวเองเมื่อมีตัวดำเนินการที่มี Precedence เท่ากันหรือใกล้เคียงกัน ใน Postfix Expression, ความสัมพันธ์จะช่วยให้ระบุว่าตัวดำเนินการในนิพจน์จะถูกดำเนินการตามลำดับไหน

ขั้นตอนการคำนวณ Postfix expression

ในการคำนวณค่า Postfix ที่แปลงมาแล้ว จะนำมาทำการคำนวณ โดยใช้โครงสร้างสแตกช่วยอีกเช่นกัน ขั้นตอนการดำเนินการมีดังนี้

1. อ่านข้อมูลเป็นนิพจน์ Postfix จากซ้ายไปขวา ทีละตัว
2. ถ้าเป็น Operand ให้ทำการ push Operand นั้นลงในสแตก แล้วกลับไปอ่านตัวถัดไป
3. ถ้าเป็น Operator ให้ทำการ pop ค่าจากสแตก 2 ค่า โดยตัวแรกจะเป็น Operand ตัวที่ 2 และตัวที่ 1 ตามลำดับ
4. ทำการคำนวณและ ทำการ push ผลลัพธ์ที่ได้จากการคำนวณ ในข้อ 3 ลงสแตก
5. ถ้าข้อมูลยังอ่านไม่หมดให้กลับไปทำ ข้อ 1 ใหม่

วิธีการคำนวณ Postfix expression

นิพจน์postfixที่จะนำมาคำนวณ คือ 5 3 2 * + 4 - 5 +

5	3	2	*	+	4	-	5	+
		2						
	3	3	6		4		5	
5	5	5	5	11	11	7	7	12

ดังนั้น ผลลัพธีจากการคำนวณ คือ 12

Algorithmของระบบ

```java
import java.util.Scanner; //เป็นการ Import Liberry Scanner เข้ามาใช้ในการรับค่า จากผู้ใช้งาน

public class Stack { //สร้าง Class Stack เพื่อใช้ในการทำงานในรูปแบบ Stack
    private int maxSize;   //ประกาศตัวแปร maxSize ชนิด Integer เป็นแบบ Private เพื่อใช้ตัวแปรนี้ได้แค่ ในClass
    private int top;       //ประกาศตัวแปร top ชนิด Integer
    private int[] stackArray; //ประกาศตัวแปร stackArray ชนิด Integer ในรูปแบบ Array

    public Stack(int size) { //เป็น Method Stack แบบ Public และมีการนำค่าเข้ามาใช้ทำงานใน Method
        maxSize = size; //ให้ ค่าsize ที่รับมาเข้ามาเก็บใน maxSize เพื่อกำหนดขนาดของ Stack
        stackArray = new int[maxSize];//ให้  stackArray เก็บค่าใหม่ โดยมีขนาดของ Array = maxSize
        top = -1;  //ให้ค่า top -1 เพื่อให้ Stack เป็นค่า null
    }

    public void push(int item) {//เป็น Method push แบบ Public และMethod เป็นการรับค่าเข้ามาเก็บใน Stack
        if (isFull()) { // ถ้า ค่าใน Stack เต็ม
            System.out.println("Stack is full. Cannot push " + item); // จะทำการ Output ข้อความดังกล่าวให้ผู้ใช้ทราบ
        } else { //แต่ถ้าไม่เต็ม
            stackArray[++top] = item; //จะทำการให้เก็บ ค่า item ลง stackArray ในตำแหน่งที่ topมีการเพิ่มค่า
        }
    }

    public int pop() {//เป็น Method pop แบบ Public และMethod เป็นการนำค่าบนสุดออกจาก Stack
        if (isEmpty()) { // ถ้า ค่าใน Stack ว่าง
            System.out.println("Stack is empty. Cannot pop."); // จะทำการ Output ข้อความดังกล่าวให้ผู้ใช้ทราบ
            return -1;  //ให้ทำการ -1 เพื่อให้ระบบรู้ว่า Error
        } else { //แต่ถ้าไม่ว่าง
            return stackArray[top--]; //จะทำการ ส่งค่า StackArray โดยตำแหน่ง top จะลดลง 1 ค่า
        }
    }
```

Algorithmของระบบ

```
31      public boolean isEmpty() { //เป็น Function isEmpty แบบ Public และจะมีการ return ค่าเป็นชนิด boolean ใช้เพื่อตรวจสอบว่าเป็นค่าว่างไหม
32          return (top == -1);// จะ return true เมื่อ top == -1 หรือ เป็นค่าว่าง
33      }
34
35      public boolean isFull() { //เป็น Function isFull แบบ Public และจะมีการ return ค่าเป็นชนิด boolean ใช้เพื่อตรวจสอบว่าค่าใน Stackว่าเต็มไหม
36          return (top == maxSize - 1);// จะ return true เมื่อ top == maxSize -1
37      }
38
39      public static void main(String[] args) { // เป็น main หลักในการทำงาน
40          Scanner scan = new Scanner(System.in); //กำหนดให้ตัวแปร scan ที่ไว้สำหรับการเรียกใช้  class ของ Scanner
41          String PostfixEps; //กำหนดตัวแปร PostfixEps ในรูปแบบ String เพื่อใช้ในการเก็บค่าจากผู้ใช้
42          System.out.print("Enter Postfix Expresstion : "); // จะทำการ Output ข้อความดังกล่าวให้ผู้ใช้ทราบ
43          PostfixEps = scan.nextLine(); //ให้ PostfixEps รับค่าจากผู้ใช้
44
45          int lenPost = PostfixEps.length(); //ให้ lenPost เก็ขนาดของPostfixEps เป็น Integer
46          Stack stack = new Stack(lenPost); //กำหนดให้ตัวแปร stack ที่ไว้สำหรับเรียกใช้ class ของ Stack
47
48          for(int i = 0; i<lenPost;i++) { //จะทำการวนลูป เมื่อ i มีค่าน้อยกว่าค่า lenPost โดยเริ่มต้นให้ค่า i=0
49              char P = PostfixEps.charAt(i);//ให้ Pชนิด Charเก็บค่า อักษรจากข้อความที่รับมาจากผู้ใช้
50              if(Character.isDigit(P)) {//ถ้าค่าPเป็น
51                  stack.push(Integer.parseInt(String.valueOf(P)));//แปลงCharเป็นStringแล้วแปลงเป็นInt แล้วนำค่าก็บลงstack
52              }else if (P == ' '){//แต่ถ้าเจอเว้นวรรคหรือค่าว่าง ให้ข้ามไปไม่สนใจ
53
54              }else{//ถ้าไม่ใช่ตัวเลขและค่าว่าง
55                  int Operand2 = stack.pop();//ให้Operand2 เก็บค่า ที่ดึงค่าบนสุดออกมาจากแสตก
56                  int Operand1 = stack.pop();//ให้Operand1 เก็บค่า ที่ดึงค่าบนสุดออกมาจากแสตก
57
```

Algorithmของระบบ

```
58                  switch (P) {//รับค่าPมาใช้ในการเลือกCase
59                  case '+'://ถ้าค่าPเป็น+
60                      stack.push(Operand1 + Operand2);//ให้Operand1 + Operand2 ล้วเก็บค่าลงสแตก
61                      break;//หยุดการทำงาน
62                  case '-'://ถ้าค่าPเป็น-
63                      stack.push(Operand1 - Operand2);//ให้Operand1 - Operand2 ล้วเก็บค่าลงสแตก
64                      break;//หยุดการทำงาน
65                  case '*'://ถ้าค่าPเป็น*
66                      stack.push(Operand1 * Operand2);//ให้Operand1 * Operand2 ล้วเก็บค่าลงสแตก
67                      break;//หยุดการทำงาน
68                  case '/'://ถ้าค่าPเป็น/
69                      stack.push(Operand1 / Operand2);//ให้Operand1 / Operand2 ล้วเก็บค่าลงสแตก
70                      break;//หยุดการทำงาน
71                  case '^'://ถ้าค่าPเป็น^
72                      stack.push((int)(Math.pow(Operand1, Operand2)));//ให้Operand1 ยกกำลัง Operand2 ล้วเก็บค่าลงสแตก
73                      break;//หยุดการทำงาน
74                  }
75
76              }
77
78              }
79          while (!stack.isEmpty()) {//ถ้าในstackไม่ว่างให้ทำการวนลูป
80              int ElementofStack = stack.pop();//ให้ดึงค่าจากstackมาเก็บในElementofStack
81              System.out.print("\nResult of Postfix Expression : " + ElementofStack);//ให้แสดงค่าที่อยู่ในstack
82          }
83      }
84  }
```

ตัวอย่างการทำงานของระบบ

```
Enter Postfix Expresstion : 5 3 2 * + 4 - 5 +

Result of Postfix Expression : 12
```

รายชื่อสมาชิก

1.นาย สิทธินนท์ สิงห์พะเนา เลขที่ 5

2.นาย ศุภวัทน์ แบบขุนทด เลขที่ 9

3.นาย สุทธิพงษ์ พูลสวัสดิ์ เลขที่ 16

4.นางสาว จารุกัญญ์ ทองหล่อ เลขที่ 17

5.นาย ภูริต เจนสาริกิจ เลขที่ 18

6.นางสาว รลินทิพย์ สกุลอินทร์ เลขที่ 24

Thank You