DATA STRUCTURES AND ALGORITHMS
โครงสร้างข้อมูลและขั้นตอนวิธี

โปรแกรมสำหรับการแปลงนิพจน์แบบ infix เป็น postfix และ postfix ไปเป็น infix

เสนอ
อาจารย์ ดรสุภาพร บุญฤทธิ์.
523231 โครงสร้างข้อมูลและขั้นตอนวิธี
ภาคเรียนที่ 2562/2
มหาวิทยาลัยเทคโนโลยีสุรนารี

จัดทำโดย

นาย กิตติศักดิ์	เพชรแหน	B6015909	
นางสาว อภิชญา	ตาดี	B6102579	
นางสาว มนัสชนก	ศรีเครือดง	B6102647	
นางสาว พิชชาพร	ยอดคีรี	B6131203	

วิธีการเปิดไฟล์เพื่อใส่สมการ

Infix to Postfix

-เข้าไปที่โฟลเดอร์ **InfixToPostfix**

-เลือกไฟล์ **main.exe**

-ใส่สมการที่ต้องการในหน้าต่างที่เด้งขึ้นมา

→ **InfixToPostfix -> main.exe**

Postfix to Infix

- เข้าไปที่โฟลเดอร์ **PostfixToInfix**

- เข้าไปที่โฟลเดอร์ **Project**

- เลือกไฟล์ **PostfixToInfix.exe**

→ **PostfixToInfix -> Project -> PostfixToInfix.exe**

INFIX TO POSTFIX

```
#include <bits/stdc++.h>
#include <string>
typedef struct nd{
   char c;
   struct nd *next;
} node;
node *top = NULL;
```

← เป็นการประกาศโครงสร้างของโน๊ต

```
void push(char ch)
{
   node *n = (node*)malloc(sizeof(node));
   n->next = top;
   top = n;
   n->c = ch;
}
```

← เป็นการเพิ่มโน๊ตใน stack

```
char pop()
{
   char p;
   node *n;
   n = top;
   top = top->next;
   p = n->c;
   free(n);
   return p;
}
```

← การนำโน๊ตออกจาก stack

```
char stacktop()
{
   if(top == NULL )
      return NULL;
   else
      return top->c;
}
```

← เป็นฟังก์ชันที่เช็คตัวบนของ stack

```
int checkPriority(char c)
   int priority;
   if(c=='^')
      priority = 3;
   else if(c=='*'||c=='/')
      priority = 2;
   else if(c=='+'||c=='-')
      priority = 1;
   else
      priority = -1;
   return priority;
```

← เช็คลำดับความสำคัญของแต่ละเครื่องหมาย

```
void checkOperator(char ch)
{
   if(stacktop() == NULL )
      push(ch);
   else
   {
      if(checkPriority(ch)<=checkPriority(stacktop()))
      {
         while(checkPriority(ch)<=checkPriority(stacktop())&&
stacktop != NULL)
         {
            printf("%c ",pop());
         }
      }
      push(ch);
   }
}
```

เช็คเครื่องหมายที่จะเข้า stack ว่ามีความสำคัญกว่าตัวบนสุดของ stack หรือไม่ หากมากกว่า จะใส่เพิ่มและอยู่ที่ตำแหน่งบนสุด แต่หากมีความสำคัญน้อยว่าจะนำตัวที่มีค่ามากกว่าตัวที่โดนเพิ่มออกนอก stack จนกว่าตัวที่อยู่ใน stack จะมีค่าน้อยว่าตัวเพิ่มเข้าไปใหม่

```
bool isBasicOperator(char c)
{
   if(c == '+' || c == '-' ||c == '*' ||c == '/' ||c == '^' ||c == '(' ||c == ')'||c
== '.')
      return true;
   else
      return false;
}
bool isOperator(char c)
{
   if(c == '+' || c == '-' ||c == '*' ||c == '/' ||c == '^' ||c == '(' ||c == ')')
      return true;
   else
      return false;
```

เช็คเพื่อหาเครื่องหมายโดยจะรวมการเช็คจุดเข้าไปด้วยเพื่อกรณีที่ใช้เป็นทศนิยม

```
void infixToPostfix(char s[])
{
   char str[100]="",stck[50]="",str2[50]="";
   int num=1,x=0,index=0,index2=0;
   char p;

printf("\n=============================================================================================
===============\n\n");
   printf("%-5s%-10s%-20s%-40s\n\n","Step","Symbol","Stack","Output");
   int i=0;
   while(i<strlen(s))
   {
      if(!isOperator(s[i]))
      {
         while(!isOperator(s[i])&&i+1<=strlen(s))
         {
            str2[index2++] = s[i++];
         }
         str2[index2++] = ' ';
         strcat(str,str2);
         x+=index2;
         printf("%-5d%-10s%-20s%-50s\n",num++,str2,stck,str);
         while(index2>0)
         {
            str2[index2--] = '\0';
         }
         str2[0] = '\0';
      }
      if(s[i] == '(')
      {
         push('(');
         stck[index++] = '(';
         stck[index++] = ' ';
         printf("%-5d%-10c%-20s%-50s\n",num++,s[i++],stck,str);
      }
      else if(s[i] == ')')
      {
         while(stacktop() != NULL && stacktop() != '(')
         {
            p = pop();
            str[x++] = p;
            str[x++] = ' ';
            stck[index--] = '\0';
            stck[index--] = '\0';
         }
         if(stacktop() == '(')
         {
            p = pop();
            stck[index--] = '\0';
            stck[index--] = '\0';
            stck[index] = '\0';
         }
         printf("%-5d%-10c%-20s%-50s\n",num++,s[i++],stck,str);
```

```
      }
      else
      {
         if(i<strlen(s))
         {
            if(stacktop() == NULL || (checkPriority(s[i]) > checkPriority(stacktop())))
            {
               push(s[i]);
               stck[index++] = s[i];
               stck[index++] = ' ';
            }

            else
            {
               while((stacktop() != NULL) && (checkPriority(s[i]) <= checkPriority(stacktop())))
               {
                  p = pop();
                  str[x++] = p;
                  str[x++] = ' ';
                  stck[index--] = '\0';
                  stck[index--] = '\0';
               }
               push(s[i]);
               stck[index++] = s[i];
               stck[index++] = ' ';
            }
            printf("%-5d%-10c%-20s%-50s\n",num++,s[i++],stck,str);
         }
      }
   }
   while(stacktop() != NULL)
   {
      p = pop();
      str[x++] = p;
      str[x++] = ' ';
      stck[index--] = '\0';
      stck[index--] = '\0';
   }
   stck[index] = '\0';
   printf("%-5d%-10c%-20s%-50s\n\n",num++,' ',stck,str);
printf("=================================================================================================
=============\n\n");
   printf("Postfix Equation : %s\n\n",str);

printf("\n=================================================================================================
===============\n\n");
```

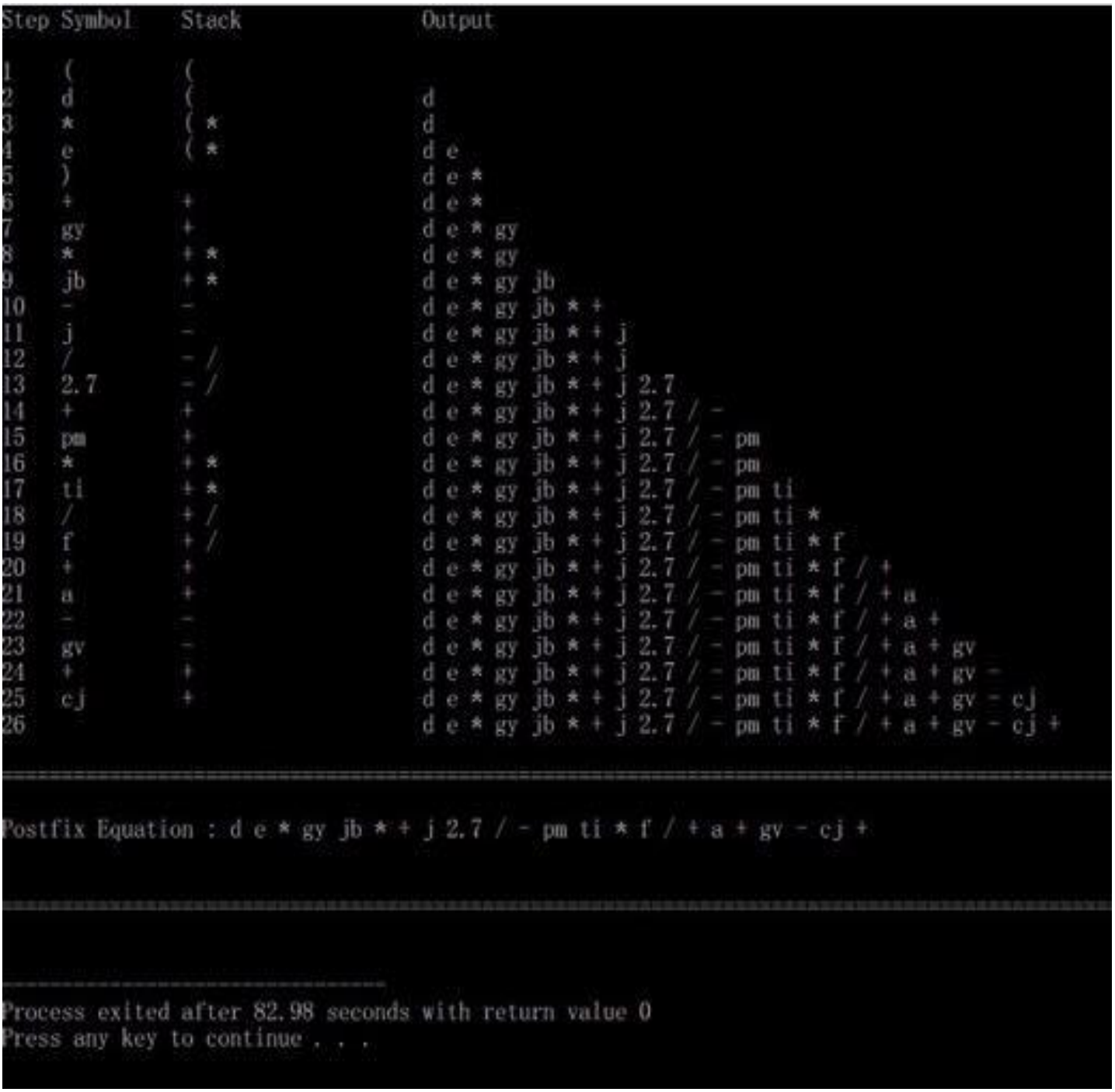
หากเจอตัวอักษรหรือตัวเลขให้ปริ้นได้เลย หากเจอเครื่องหมายจะต้องเช็คลำดับความสำคัญก่อน หากเจอวงเล็บเปิดสามารถใส่ใน stack ได้เลย แต่หากเจอวงเล็บปิด จะนำออกจาก stack จนกว่าจะเจอวงเล็บเปิด

```
while(stacktop() != NULL)
    {
        p = pop();
        str[x++] = p;
        str[x++] = ' ';
        stck[index--] = '\0';
        stck[index--] = '\0';
    }
    stck[index] = '\0';
    printf("%-5d%-10c%-20s%-50s\n\n",num++,' ',stck,str);

printf("================================================
================================================
==============\n\n");
    printf("Postfix Equation : %s\n\n",str);

printf("\n================================================
================================================
==============\n\n");
}
```

เป็นคำสั่งที่จะนำเครื่องหมายที่เหลืออยู่ใน stack ออกมานอก stack ให้หมด

```
nt main()
{
    char infixString[100];

printf("================================================
=== Infix to Postfix
================================================\n\n");
    printf("Enter an equation : ");
    gets(infixString);
    char c = checkString(infixString);
    if(c=='y')
        infixToPostfix(infixString);
    else
    {
        printf("%c not basic operand\n",c);
    }
    return 0;
}
```

เป็นส่วนรับข้อมูล string เพื่อนำไปแปลงข้อมูลจาก infix เป็น postfix



ผลลัพธ์ infix to postfix

POSTFIX TO INFIX

```
package PostfixToInfix;
import java.util.Scanner;
public class MainClass {
    public static void main(String[] args) {
        Scanner sc = new
Scanner(System.in);//.useDelimiter("\n");
        int n;
        String str= null;
        System.out.println("---------------------------------
Postfix to Infix---------------------------------");
        System.out.println("1.Postfix to Infix (answer in
character)");
        System.out.println("2.Postfix to Infix (answer in real
number)");
        System.out.println("----------------------------------------
----------------------------------------");
        do{
            System.out.print("select a number (1 or 2) : ");
            n = Integer.parseInt(sc.nextLine()) ;
        }while(n<1||n>2);
        System.out.println("----------------------------------------
----------------------------------------");
```

เลือกคำสั่งว่าจะใส่ข้อมูลเป็น Character หรือ Real Number

```
        System.out.println("Sample input equation : \"23 5.0 +
3 * 48\" ");
        System.out.println("Use space to split");
        System.out.println("----------------------------------------
----------------------------------------");
        System.out.print("\nenter an equation : ");
        str = sc.nextLine();
        System.out.println();
        System.out.println("----------------------------------------
----------------------------------------");
        new PostfixToInfix(str,n);
    }
}
```

รับค่า **string** เพื่อนำไปแปลงค่าจาก **postfix to infix** โดยให้เลือกประเภทคำตอบ ตอบได้ทั้ง **character** และ **real number**

```
public class PostfixToInfix extends Stack {

    private String str;
    private String stck[] = new String[40];
    private int index = 0;
    private int n;
    private String[] t;

    public PostfixToInfix() {
    }
```

class PostfixToInfix มีการประกาศตัวแปร String str, int index, int n, String[] t, String stck[]

```
public PostfixToInfix(String str, int n) {
    this.str = str;
    this.n = n;
    t = str.split(Pattern.quote(" "));
    if (this.n == 1) {
        FindCharacterAnswer();
    } else if (this.n == 2) {
        FindRealnumberAnswer();
    }
}
```

มี **constructor** PostfixToInfix เพื่อกำหนดค่าให้ตัวแปร และหาก n = 1 จะได้คำตอบเป็นตัวอักษร แต่หาก n = 2 จะได้คำตอบเป็นตัวเลข

```
public void FindCharacterAnswer() {
    System.out.printf("%-10s%-15s%-60s\n", "Step", "Symbol", "Stack");
    System.out.println("----------------------------------------------------------------------------");
    for (int i = 0; i < t.length; i++) {
        if (isBasicOperator(t[i])) {
            String s = "(";
            String s1 = pop();
            stck[index--] = "";
            String s2 = pop();
            stck[index--] = "";
            s += s2;
            s += t[i];
            s += s1;
            s += ")";
            push(s);
            stck[index++] = s;
        } else {
            push(t[i]);
            stck[index++] = t[i];
        }
        System.out.printf("%-10d%-15s", i + 1, t[i]);
        for (int j = 0; j < index; j++) {
            System.out.print(stck[j] + " ");
        }
        System.out.println();
    }
    System.out.println("----------------------------------------------------------------------------");
    System.out.print("Infix equation is ");
    System.out.println(pop());
    System.out.println("----------------------------------------------------------------------------");
}
```

เป็นการหาคำตอบที่เป็นตัวอักษร โดยที่หากเจอตัวอักษรจะนำไปใส่ใน **stack** แต่หากเจอเครื่องหมายจะนำสองตัวบนสุดใน **stack** ออกมาแล้วทำการแทรกเครื่องหมายไปตรงกลาง แล้วใส่วงเล็บครบทั้งหมดไว้แล้วนำทั้งหมดใส่กลับเข้าไปใน **stack** เหมือนเดิม

```
public void FindRealnumberAnswer() {
      System.out.printf("%-10s%-15s%-60s\n", "Step",
"Symbol", "Stack");
      System.out.println("----------------------------------------
----------------------------------------");
      for (int i = 0; i < t.length; i++) {
         if (isBasicOperator(t[i])) {
            double result = 0;
            double n2 = Double.valueOf(popD());
            stck[index--] = "";
            double n1 = Double.valueOf(popD());
            stck[index--] = "";
            if (t[i].equals("+")) {
               result = n1 + n2;
            } else if (t[i].equals("-")) {
               result = n1 - n2;
            } else if (t[i].equals("*")) {
               result = n1 * n2;
            } else if (t[i].equals("/")) {
               result = n1 / n2;
            } else if (t[i].equals("^")) {
               result = Math.pow(n1, n2);
            }
            pushD(result);
            stck[index++] = ""+result;
         }
         else {
            pushD(Double.valueOf(t[i]));
            stck[index++] = t[i];
         }
         System.out.printf("%-10d%-15s", i + 1, t[i]);
         for (int j = 0; j < index; j++) {
            System.out.print(stck[j] + " ");
         }
         System.out.println();
      }
      System.out.println("----------------------------------------
----------------------------------------");
      System.out.print("Result is ");
      System.out.println(popD());
      System.out.println("----------------------------------------
----------------------------------------");
   }
```

ตรวจสอบหาตัวเลขหากเจอจะทำไปใส่ใน **stack** หากเจอเครื่องหมายจะนำสองตัวบนใน **stack** ออกมา แล้วนำไปทำการคำนวณค่าตามเครื่องหมายที่ตรวจเจอ หากคำนวณเสร็จจะนำตัวเลขกลับเข้าไปใน **stack**

```java
public boolean isAlpha(char c) {
    if (c >= 'a' && c <= 'z') {
        return true;
    } else if (c >= 'A' && c <= 'Z') {
        return true;
    } else {
        return false;
    }
}
```

← เช็คว่าเป็นตัวอักษรหรือไม่

```java
public boolean isDigit(char c) {
    if (c >= '0' && c <= '9') {
        return true;
    } else {
        return false;
    }
}
```

← เช็คว่าเป็นตัวเลขหรือไม่

```java
public boolean checkOperator(char c) {
    if (c == '+' || c == '-' || c == '*' || c == '/' || c == '^' || c == '.') {
        return true;
    } else {
        return false;
    }
}
```

← เช็คว่าเป็นเครื่องหมายหรือไม่เพื่อในกรณีมีเลขทศนิยม

```java
public boolean isBasicOperator(String s) {
    if (s.equals("+")) {
        return true;
    } else if (s.equals("-")) {
        return true;
    } else if (s.equals("*")) {
        return true;
    } else if (s.equals("/")) {
        return true;
    } else if (s.equals("^")) {
        return true;
    } else {
        return false;
    }
}
```

← เช็คว่าเป็นเครื่องหมายหรือไม่

```
public void push(String str) {
        Node n = new Node();
        n.next = top;
        top = n;
        n.data = str;

    }
```

เป็นการนำค่าใน **stack**

```
public String pop() {
        Node n;
        n = top;
        top = top.next;
        return n.data;
    }
```

เป็นการนำตัวอักษรจาก **stack**

```
public double popD() {
        NodeD n;
        n = topD;
        topD = topD.next;
        return n.data;
    }

}
```

เป็นการนำตัวเลขออกจาก **stack**

```
run:
----------------------------------Postfix to Infix----------------------------------
1.Postfix to Infix (answer in character)
2.Postfix to Infix (answer in real number)
------------------------------------------------------------------------------------
select a number (1 or 2) : 1
------------------------------------------------------------------------------------
Sample input equation : "23 5.0 + 3 * 48"
Use space to split
------------------------------------------------------------------------------------

enter an equation : A B C * D / +

------------------------------------------------------------------------------------
Step      Symbol         Stack
------------------------------------------------------------------------------------
1         A              A
2         B              A B
3         C              A B C
4         *              A (B*C)
5         D              A (B*C) D
6         /              A ((B*C)/D)
7         +              (A+((B*C)/D))
------------------------------------------------------------------------------------
Infix equation is (A+((B*C)/D))
------------------------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 27 seconds)
```

```
run:
----------------------------------Postfix to Infix----------------------------------
1.Postfix to Infix (answer in character)
2.Postfix to Infix (answer in real number)
------------------------------------------------------------------------------------
select a number (1 or 2) : 2
------------------------------------------------------------------------------------
Sample input equation : "23 5.0 + 3 * 48"
Use space to split
------------------------------------------------------------------------------------

enter an equation : 7 3 5 * 5 / +

------------------------------------------------------------------------------------
Step      Symbol         Stack
------------------------------------------------------------------------------------
1         7              7
2         3              7 3
3         5              7 3 5
4         *              7 15.0
5         5              7 15.0 5
6         /              7 3.0
7         +              10.0
------------------------------------------------------------------------------------
Result is 10.0
------------------------------------------------------------------------------------
BUILD SUCCESSFUL (total time: 14 seconds)
```