InfixToPostfix

จัดทำโดย

นายวรดร พรมอนันต์

รหัส 67543206020-9

SEC 1

เสนอ

อาจารย์นุรักษ์ ไชยศรี

ใบงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตรวิศวกรรมศาสตร์บัณฑิต

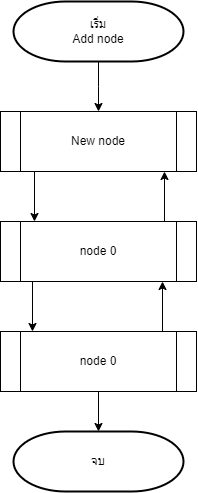
สาขาวิชาวิศวกรรมไฟฟ้า (วิศวกรรมคอมพิวเตอร์)

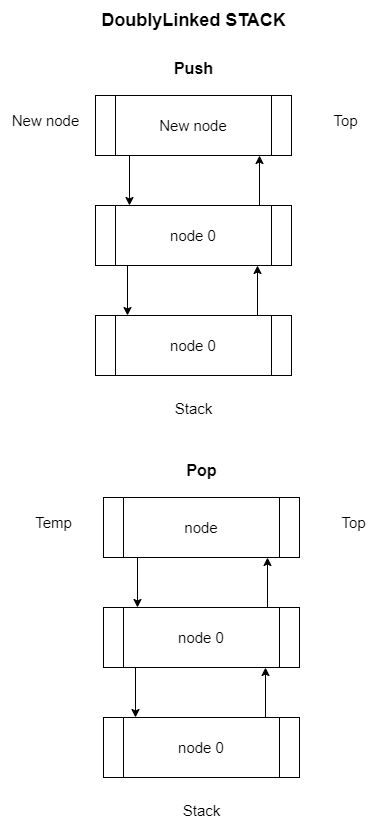
คณะ วิศวกรรมศาสตร์

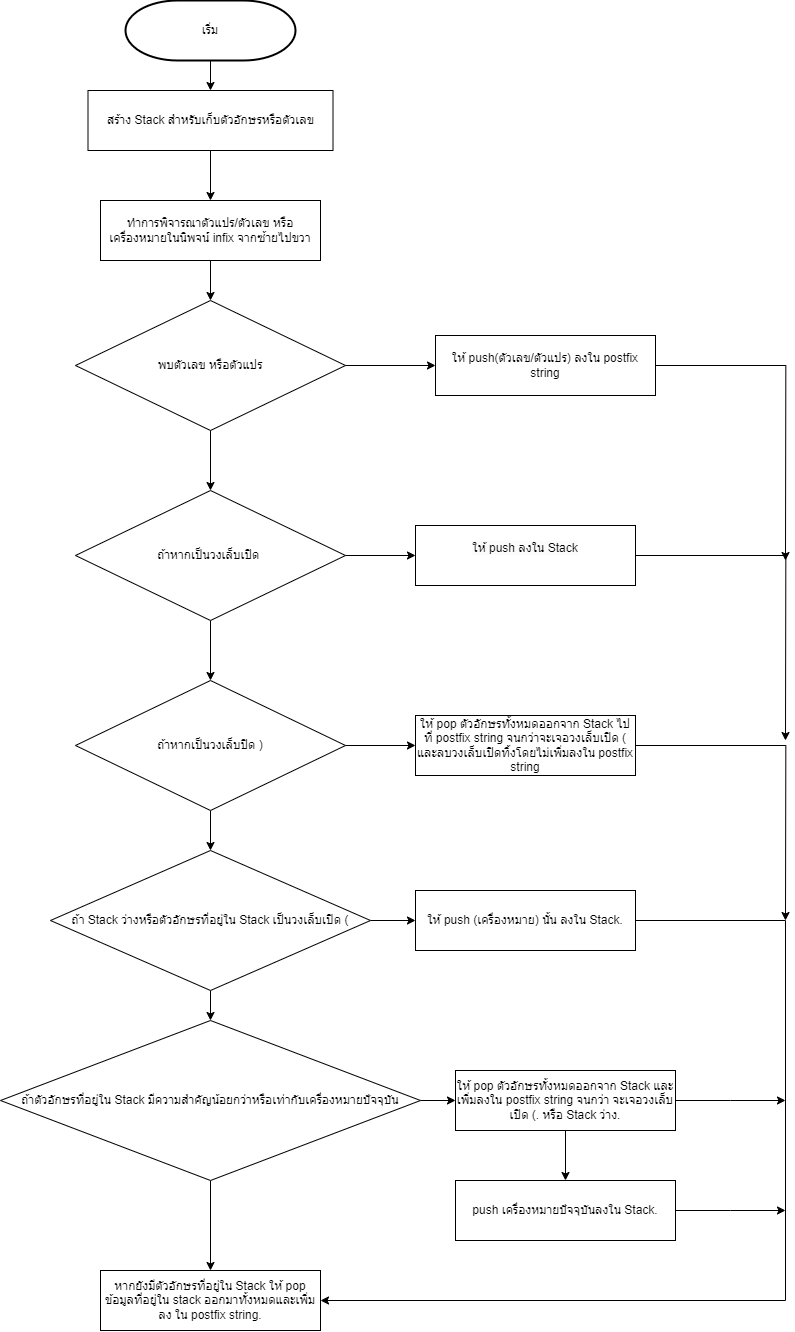
มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา เชียงใหม่  
ภาคเรียนที่ 1 ปีการศึกษา 2568

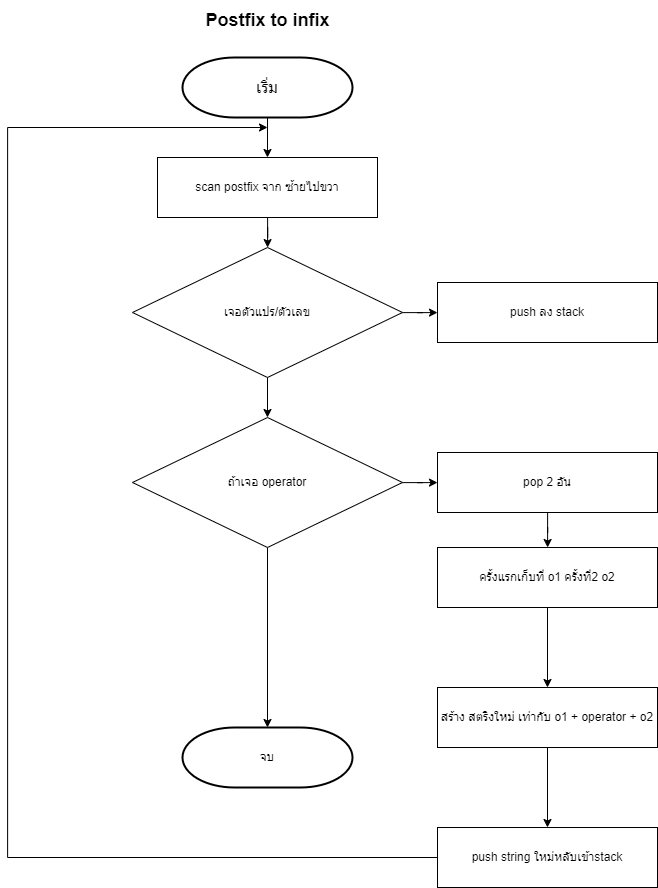
Flowchat :

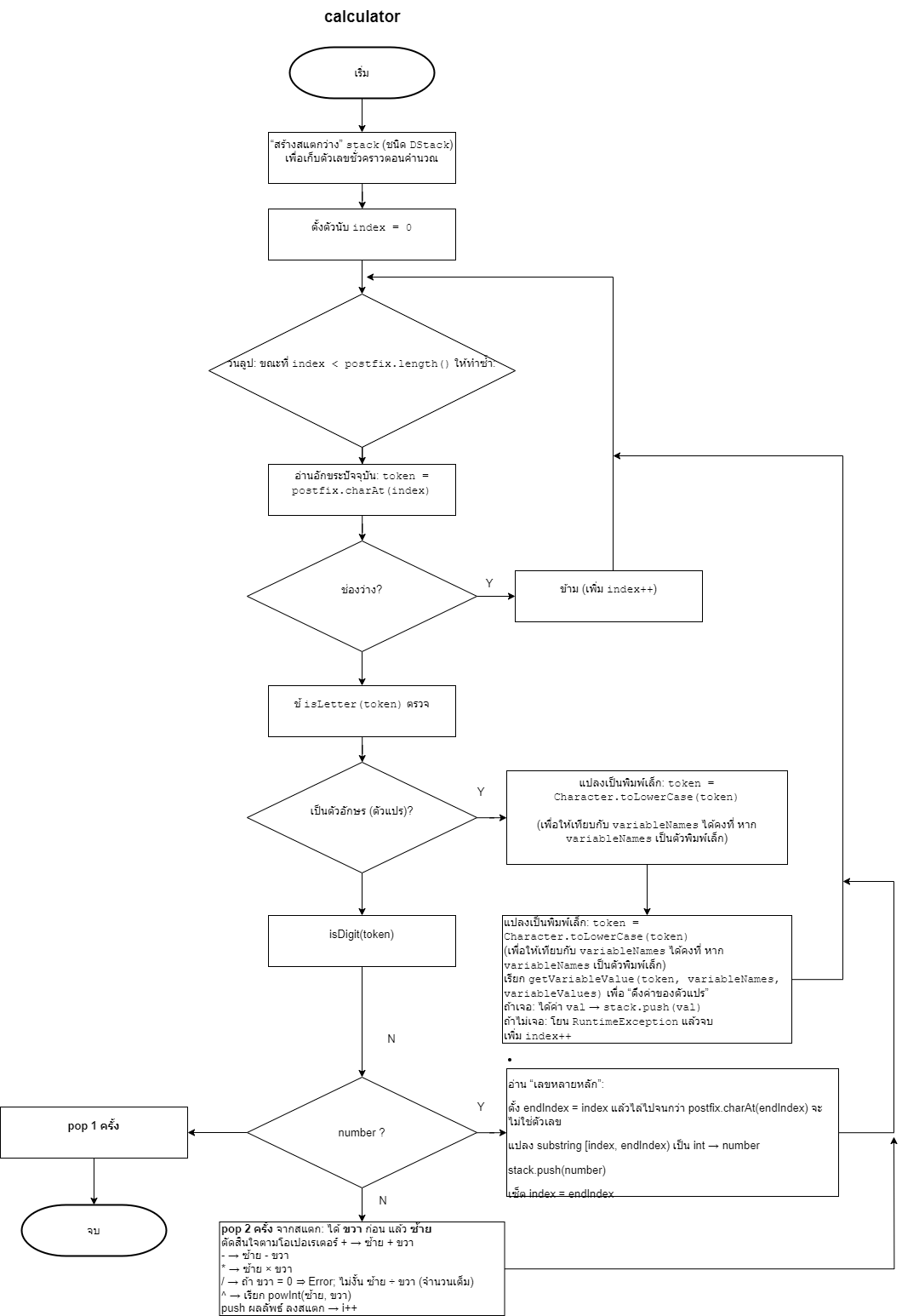
Add node DoublyLinked:











**InfixToPostfixStack.java :**

public class InfixToPostfixStack {

public static void main(String[] args) {

DStack st = new DStack() ;

calcu cl = new calcu() ;

// String Exp = "A\*(B+C)/D" ;

// String Exp = "A+B+C" ;

// String Exp = "A+B\*c" ;

// String Exp = "a+b\*(c^d-e)^(f+g\*h)-i" ;

// String Exp = "A+B)\*C" ;

String Exp = "A^b+c\*d" ;

st.InTopost(Exp) ;

System.out.println("Exp = " + Exp ) ;

String post = st.PostTOIn();

char[] vars = {'a','b','c','d','e','f','g','h','i'};

int[] vals = { 2 , 3 , 2 , 3 , 1 , 4 , 2 , 1 , 5 };

int ans = cl.evaluatePostfixWithVariables(post, vars, vals);

System.out.println("Result = " + ans);

}

}

**DNode.java :**   
public class DNode {

DNode Llink, Rlink ;

int info ;

DNode(){

this(0);

}

DNode(int data){

info = data;

}

}

**DList.java :**

public class DList {

DNode head,tail,travel ;

int count ;

void append( int data ) {

DNode newnode = new DNode(data) ;

if ( count == 0 ) {

head = newnode ;

tail = newnode ;

}else{

tail.Rlink = newnode ;

newnode.Llink = tail ;

tail = newnode ;

}

count++ ;

}

String getString( ) {

String strPostfix = "" ;

travel = head ;

while ( travel != null ) {

strPostfix = strPostfix + ( char )travel.info ; // ต่อ string

travel = travel.Rlink ; /\* เท่ากับข้อมูลทางขวา \*/

}

return strPostfix ;

}

void ShowAll() {

travel = head ;

while ( travel != null ) {

System.out.print((char)travel.info) ;

travel = travel.Rlink ;

}

System.out.println("");

}

String getAllNode() {

travel = head ;

String node = "";

for ( int i = 0 ; i < count ; i++ ) {

node = node + (char)travel.info ;

travel = travel.Rlink ;

}

return node ;

}

DNode getNode( int index ) {

if (index < 0 || index >= count) {

return null ;

}

DNode result = head ;

for( int i = 0 ; i < index ; i++ ){

result = result.Rlink ; /\* เท่ากับข้อมูลทางซ้ายของโหนด \*/

}

return result ;

}

String getStringAt(int index) {

DNode node = getNode(index);

if (node != null) {

return String.valueOf((char)node.info);

} else {

return null;

}

}

void clear (){

tail = null ;

head = null ;

travel = null ;

count = 0 ;

}

}

**DStack.java:**

public class DStack {

DList DLL = new DList() ;

int count ;

DNode top, temp ;

void push( int item ) {

DNode newnode = new DNode() ;

newnode.info = item ;

if ( count == 0 ) {

top = newnode ;

}else{

top.Rlink = newnode ;

newnode.Llink = top ;

top = top.Rlink ;

}

count++ ;

}//end push

DNode pop() {

if (!isEmpty()) {

if ( size() == 1 ) {

temp = top;

top = null ;

} else{

temp = top ;

top = top.Llink ;

top.Rlink = null ;

temp.Llink = null ;

}

count-- ;

}else{

System.out.println("stack is Empty");

temp.info = -1 ;

}

return temp ;

}

boolean isEmpty(){ return count == 0 ; }

int size(){ return count ; } // วัดsize

static int Precedence\_Operator\_input( char operators ){

if ( operators == '^' ) {

return 4 ;

}else if (operators == '\*' || operators == '%' || operators == '/' ) {

return 2 ;

}else if (operators == '+' || operators == '-') {

return 1;

}else{

return -1 ;

}//end if

}//end function PrecOperatorinput

static int Precedence\_Operator\_Stack( char operators ){

if (operators == '^' ) {

return 3 ;

}else if (operators == '\*' || operators == '%' || operators == '/' ) {

return 2 ;

}else if (operators == '+' || operators == '-') {

return 1;

}else{

return 0 ;

}//end if

}//end function PrecOperatorStack

void InTopost(String proposition) {

System.out.printf("%-8s | %-16s | %-8s%n", "Symbol", "Stack", "Postfix String");

System.out.println("-----------------------------------------------");

for( int i = 0 ; i < proposition.length() ; i++ ){

char patiant = proposition.charAt(i) ;

char CheckOperand = Operandcheck( patiant ) ;

int CheckOperetor = Operetorcheck( patiant ) ;

if (CheckOperand != '\0') { //ไม่ใช่ Operetor

DLL.append(CheckOperand) ;

}else if (CheckOperand == '\0') {

if (CheckOperetor == 40) { //ASCII 40 = (

push(patiant) ;

}else if ( patiant == 41) { //ASCII 41 = )

while ( !isEmpty()&&(char)top.info != 40) { /\*ถ้าไม่ว่างและไม่เท่ากับ ( \*/

DLL.append(pop().info) ; /\*pop to DLL(postfix) \*/

}

if( !isEmpty() && (char)top.info == 40){ /\*ถ้าไม่ว่างและเท่ากับ ( \*/

pop() ; /\*ทิ้ง \*/

}

}else{

/\* ถ้าเป็น +-/^\* \*/

while (!isEmpty() && Precedence\_Operator\_input(patiant) <= Precedence\_Operator\_Stack((char)top.info)) {

DLL.append(pop().info);

}

push(patiant) ;

}

}

printAll(String.valueOf(patiant),DLL.getString() ) ;

}//end for

while ( !isEmpty() ) {

DLL.append(pop().info) ;

}

printAll("", DLL.getString());

}

char Operandcheck( int Infix ) { /\*check Operand \*/

if ( Infix >= 65 && Infix <= 90 ||

Infix >= 97 && Infix <= 122 ||

Infix >= 48 && Infix <= 57) {

return (char)Infix ;

}else{

// System.out.println("0 Find Oparetor");

return '\0' ;

}

}

int Operetorcheck( int Infix ) {

if ( Infix >= 33 && Infix <= 47 ||

Infix >= 58 && Infix <= 63 ||

Infix >= 91 && Infix <= 96||

Infix >= 123 && Infix <= 126)

{

// System.out.println("1 Find Oparand") ;

return Infix ;

}else{

// System.out.println("0 Find Oparetor");

return -1 ;

}

}

void printAll(String symbol, String postfix) {

String stackStr = "";

DNode p = top;

while (p != null) {

stackStr = (char)p.info + stackStr; // ไล่จากล่างขึ้นบน

p = p.Llink;

}

System.out.printf("%-8s | %-16s | %-8s%n",

symbol, stackStr, postfix);

}

String PostTOIn(){

System.out.print("Postfix = " ) ;

DLL.ShowAll() ;

PostfixToInfix pti = new PostfixToInfix() ;

String post = DLL.getAllNode() ;

// System.out.println("Post = " + post) ;

pti.PostToIn(post) ;

return post ;

}

}

classStringStack.java :

public class classStringStack {

String info ;

classStringStack next ;

classStringStack( String data ) {

info = data ;

}

}

**StrStack.java :**  
public class StrStack {

classStringStack st = new classStringStack(null);

classStringStack top ;

boolean isEmpty() {

return top == null ;

}

void push ( String str ) {

classStringStack data = new classStringStack( str );

data.next = top ;

top = data ;

}

String pop() {

if( isEmpty() ){

return null ;

}

String str = top.info;

top = top.next;

return str ;

} }

**PostfixToInfix.java :**  
public class PostfixToInfix {

void PostToIn( String proposition) {

// DList Dou = new DList() ;

StrStack st = new StrStack() ;

for( int i = 0 ; i < proposition.length() ; i++ ){ /\*Postfix จากซ้ายไปขวา \*/

char chack = proposition.charAt(i) ;

if((chack >= 'A' && chack <= 'Z') || (chack >= 'a' && chack <= 'z' ) || (chack >= '0' && chack <= '9') ){

st.push(String.valueOf(chack)) ;

} else {

String op2 = st.pop() ;

String op1 = st.pop() ;

st.push("("+op1+chack+op2+")") ;

} }

String infix = st.pop();

System.out.println("Infix = " + infix);

} }

**calcu.java :**

public class calcu {

static boolean isDigit(char ch){ return ch >= '0' && ch <= '9'; } //check number

static boolean isLetter(char ch){ return (ch >= 'A' && ch <= 'Z') //check operand

|| (ch >= 'a' && ch <= 'z'); }

static int getVariableValue(char variable, char[] variableNames, int[] variableValues){

for (int i = 0 ; i < variableNames.length ; i++)

if (variableNames[i] == variable) /\* เปลี่ยนเทียบแต่ละตัว \*/

return variableValues[i];

throw new RuntimeException("do not find veriable" + variable);

}

int powInt(int base, int exponent){

if (exponent < 0) throw new RuntimeException("ERROR pow ^ - ");

int result = 1;

while (exponent > 0){

if ((exponent & 1) == 1)

result \*= base ;

base \*= base ;

exponent >>= 1 ;

}

return result;

}

int evaluatePostfixWithVariables(String postfix, char[] variableNames, int[] variableValues){

DStack stack = new DStack() ;

for (int index = 0 ; index < postfix.length(); ){

char token = postfix.charAt(index) ; //รับมาทีละตัว

// ข้ามช่องว่าง (รองรับตัวเลขหลายหลักคั่นด้วย space)

if (token == ' ' || token == '\t'){ index++; continue; }

if (isLetter(token)){

token = Character.toLowerCase(token) ; /\*แปลตัวใหญ่เป็นเล็ก \*/

stack.push(getVariableValue(token, variableNames, variableValues)) ;

index++;

} else if (isDigit(token)){

// อ่านเลขหลายหลัก

int endIndex = index ;

while (endIndex < postfix.length() && isDigit(postfix.charAt(endIndex)))

endIndex++ ;

int number = Integer.parseInt(postfix.substring(index, endIndex)) ;

stack.push(number) ;

index = endIndex ;

} else {

/\* เจอ operator pop 2 ตัว\*/

int rightOperand = stack.pop().info ;

int leftOperand = stack.pop().info ;

int result ;

if (token == '+') result = leftOperand + rightOperand;

else if (token == '-') result = leftOperand - rightOperand;

else if (token == '\*') result = leftOperand \* rightOperand;

else if (token == '/'){

if (rightOperand == 0)

throw new ArithmeticException("หารด้วยศูนย์");

result = leftOperand / rightOperand; // แบ่งจำนวนเต็ม

}

else if (token == '^')

result = powInt(leftOperand, rightOperand) ;

else

throw new RuntimeException("operator error: " + token) ;

stack.push(result) ;

index++;

}//end if

}//end for

return stack.pop().info;

}//end mathod

}//end class

Output :

Symbol | Stack | Postfix String

-----------------------------------------------

A | | A

\* | \* | A

( | \*( | A

B | \*( | AB

+ | \*(+ | AB

C | \*(+ | ABC

) | \* | ABC+

/ | / | ABC+\*

D | / | ABC+\*D

| | ABC+\*D/

Exp = A\*(B+C)/D

Postfix = ABC+\*D/

Infix = ((A\*(B+C))/D)

Result = 3

Symbol | Stack | Postfix String

-----------------------------------------------

A | | A

^ | ^ | A

b | ^ | Ab

+ | + | Ab^

c | + | Ab^c

\* | +\* | Ab^c

d | +\* | Ab^cd

| | Ab^cd\*+

Exp = A^b+c\*d

Postfix = Ab^cd\*+

Infix = ((A^b)+(c\*d))

Result = 14