7

Mechi Multiple Campus

(Tribhuvan University)
Bhadrapur, Jhapa

Lab Report of

Data Structures and Algorithm (CACS-201)

Conversion of Expression

Faculty of Humanities & Social Sciences

Tribhuvan University

Kritipur, Nepal

Submitted By

Name: Santosh Bhandari

Roll No: 58

Submitted To

Mechi Multiple Campus
Department of Bachelor in Computer Application
Bhadrapur, Jhapa, Nepal

Introduction to the Mathematical Expression

A mathematical expression includes operands and the operator.

Types of Mathematical Expression

i) Infix Expression

In infix expression, the operator is placed in between the operands. It is the most common mathematical expression used.

Example :- a+b

ii) postfix Expression

In postfix expression, the operator is placed after the operands.

Example :- ab+

iii) Prefix Expression

In prefix expression, the operator is placed before the operands.

Example :- +ab

Algorithm to Convert Infix Expression to Postfix

① START

② PUSH left paranthesis "(" onto STACK and add right paranthesis ")" at the end of Q (Infix Expression)

③ Scan each element of Q from left to right and repeat 4 to 7 until the STACK is empty.

④ If an Operand is encountered, then add it to p (postfix expression)

⑤ If a left paranthesis is encountered, then push it to STACK.

⑥ If an operator ⊗ is encountered, then :-

a) pop all the operators at the TOP of the stack which has higher or same precedence than ⊗ and add into p.

b) PUSH the Operator ⊗ onto STACK.

⑦ If a right paranthesis ")" is encountered, then

a) Repeatedly pop all the operators at the top of the stack until left paranthesis "(" is encountered and add in p.

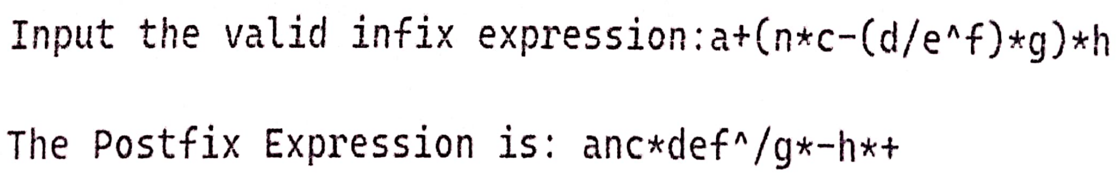
b) Remove the left parantheses "(" from stack.

⑧ EXIT

Program Code

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
void push(char);
char pop(void);
int isoperator(char);
int isprecedence(char);
void InfixToPostfix(char[], char[]);
char stack[50];
int tos=-1;
void push(char data){
        tos++;
        stack[tos]=data;
}
char pop(){
        char data;
        data=stack[tos];
        tos--;
        return(data);
}
int isoperator(char symbol){
        if(symbol == '^' || symbol == '$' || symbol == '*' || symbol == '/' || symbol == '+'
        || symbol == '-')
                return(1);
        else
                return(0);
}
int precedence(char symbol){
        if(symbol == '^' || symbol == '$')
                return(3);
        if(symbol == '*' || symbol == '/')
                return(2);
        if(symbol == '+' || symbol == '-')
                return(1);
        else
                return(0);
}
void InfixToPostfix(char infix[], char postfix[]){
        int i=0,j=0;
        char scan_el,x,op;
        push('(');
        strcat(infix,")");
        scan_el=infix[i];
        while(scan_el != '\0'){
                if(isalpha(scan_el)){
                        postfix[j]=scan_el;
                        j++;
                }else if(scan_el == '('){
                        push(scan_el);
                }else if(isoperator(scan_el)==1){
                        op=pop();
                        while(isoperator(op)==1 && precedence(op)>=
                        precedence(scan_el)){
                                postfix[j]=op;
                                j++;
                                op=pop();
                        }
                        push(op);
                        push(scan_el);
                }else if(scan_el==')'){
                        x=pop();
                        while(x != '('){
                                postfix[j]=x;
                                j++;
                                x=pop();
                        }
                }else{
                        printf("\nInvalid infix expression");
                        exit(0);
                }
                i++;
                scan_el=infix[i];
        }
        if(tos>0){
                printf("\nInvalid infix expression");
                exit(0);
        }
        postfix[j]='\0';
}
void main(){
        char postfix[50], infix[50];
        printf("\nInput the valid infix expression:");
        gets(infix);
        InfixToPostfix(infix,postfix);
        printf("\nThe Postfix Expression is: ");
        puts(postfix);
}
```

Output of the Program



```
Input the valid infix expression:a+(n*c-(d/e^f)*g)*h

The Postfix Expression is: anc*def^/g*-h*+
```

Algorithm to Evaluate postfix Expression

The following algorithm calculates the result of a mathematical expression (P written in Postfix notation)

①START

②Add a right parenthesis ")" at the end of P.

③Scan all the elements of P from left to right individually and repeat steps 4 and 5 Until ")" is encountered.

④If an operand is encountered, PUSH it to STACK.

⑤If an operator $\otimes$ is encountered, then

a) pop the top two elements of STACK, where A is the top element and B is the next to top element.

b) Evaluate $B \otimes A$

c) PUSH the result back to STACK

⑥Set the result as the element at the top of the STACK.

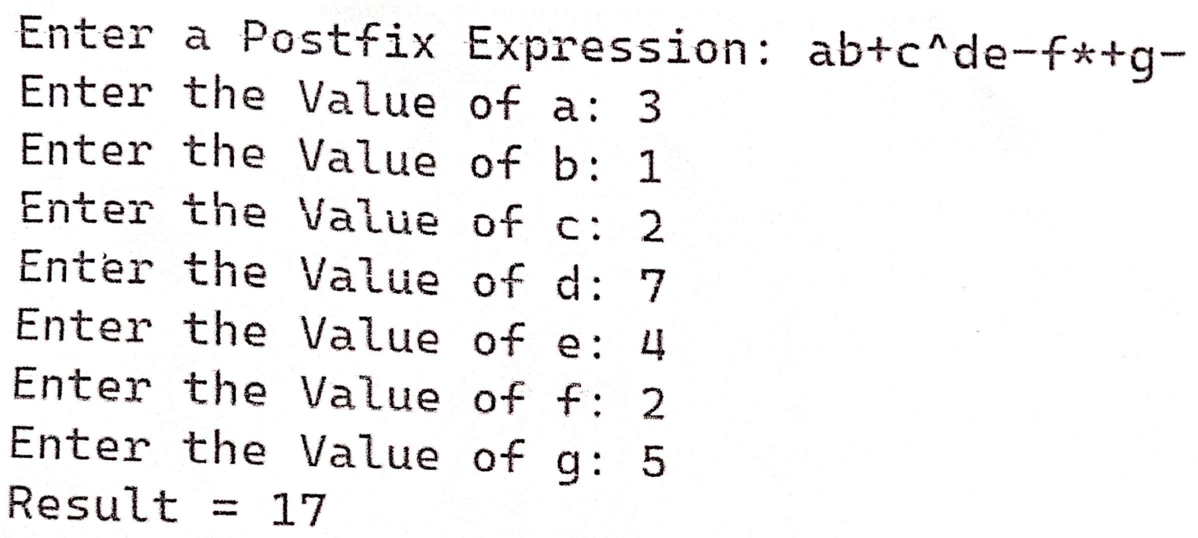
⑦EXIT

Program Code

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
void push(int);
int pop();
int stack[50],tos=-1;
void main(){
        char postfix[50],ch;
        int i,a,b,c,value,result,len;
        printf("Enter a Postfix Expression: ");
        gets(postfix);
        len=strlen(postfix);
        for(i=0;i<len;i++){
                ch=postfix[i];
                if(isalpha(ch)){
                        printf("Enter the Value of %c: ",ch);
                        scanf("%d",&value);
                        push(value);
                }
                else{
                        a=pop();
                        b=pop();
                        switch(ch){
                                case '+':
                                        c=b+a;
                                        break;
                                case '-':
                                        c=b-a;
                                        break;
                                case '*':
                                        c=b*a;
                                        break;
                                case '/':
                                        c=b/a;
                                        break;
                                case '$':
                                case '^':
                                        c=pow(b,a);
                                        break;
                        }
                        push(c);
                }
        }
        printf("Result = %d",pop());
}
int pop(){
        int data;
```

```
            data = stack[tos];
            tos--;
            return data;
    }
    void push(int data){
            tos++;
             stack[tos]=data;
    }
```

Output of the Program



```
Enter a Postfix Expression: ab+c^de-f*+g-
Enter the Value of a: 3
Enter the Value of b: 1
Enter the Value of c: 2
Enter the Value of d: 7
Enter the Value of e: 4
Enter the Value of f: 2
Enter the Value of g: 5
Result = 17
```

Conclusion

A mathematical expression includes operator and the operands. In mathematical expression, if the operator is between the operands, it is known as Infix expression. If the operator is placed after the operands, it is known as postfix and if the operator is placed before the operands, it is known as prefix.