Data Structures Lab

Experiment 8:
Infix to Postfix

Project By:

Mohammed Rabeeh
Roll No: 35
TVE18CS036

Contents

1 Aim

Write a program to convert an infix expression to postfix using stack

2 Problem Description

A postfix expression is one in which the operators are placed after the operands. For example,

$$A + B = AB+$$

An infix expression is converted to a postfix expression for easy processing.

3 Algorithm

1. Read each character from left to right until end of infix expression.
2. If stack is empty, push to stack.
3. If character is '(', push to stack.
4. If character is an operand, append to output expression.
5. Else if character is operator,
6. If precedence of stack[top] ¡ operator, push operator to stack.
7. Else, pop stack[top] until precedence of stack[top] ¡ operator, append operator to output expression.
8. If character is ')', pop stack[top] and append to output expression until stack[top] = '(' and then pop stack[top].

4 Program Code

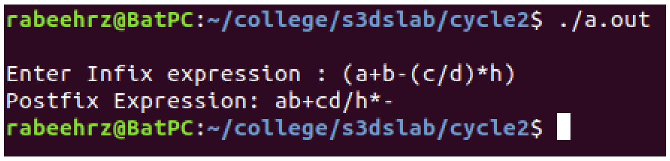
```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>

#define SIZE 100

char stack[SIZE];
int top = -1;


void push(char item)
{
   if(top >= SIZE-1)
   {
      printf("\nStack Overflow.");
   }
   else
   {
      top = top+1;
      stack[top] = item;
   }
}

char pop()
{
   char item ;

   if(top <0)
   {
      printf("stack under flow: invalid infix expression");
      getchar();
      exit(1);
   }
   else
   {
      item = stack[top];
```

```
    top = top-1;
    return(item);
  }
}


int is_operator(char symbol)
{
  if(symbol == '^' || symbol == '*' || symbol == '/' || symbol ==
     '+' || symbol =='-')
  {
    return 1;
  }
  else
  {
  return 0;
  }
}


int precedence(char symbol)
{
  if(symbol == '^')
  {
    return(3);
  }
  else if(symbol == '*' || symbol == '/')
  {
    return(2);
  }
  else if(symbol == '+' || symbol == '-')
  {
    return(1);
  }
  else
  {
    return(0);
  }
}
```

```
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
   int i, j;
   char item;
   char x;

   push('(');
   strcat(infix_exp,")");

   i=0;
   j=0;
   item=infix_exp[i];

   while(item != '\0')
   {
      if(item == '(')
      {
         push(item);
      }
      else if( isdigit(item) || isalpha(item))
      {
         postfix_exp[j] = item;
         j++;
      }
      else if(is_operator(item) == 1)
      {
         x=pop();
         while(is_operator(x) == 1 && precedence(x)>=
             precedence(item))
         {
            postfix_exp[j] = x;
            j++;
            x = pop();
         }
         push(x);

         push(item);
      }
      else if(item == ')')
      {
         x = pop();
         while(x != '(')
         {
            postfix_exp[j] = x;
            j++;
            x = pop();
         }
      }
      else
      {
         printf("\nInvalid infix Expression.\n");
         getchar();
         exit(1);
      }
      i++;


      item = infix_exp[i];
   }
   if(top>0)
   {
      printf("\nInvalid infix Expression.\n");
      getchar();
      exit(1);
   }
   if(top>0)
   {
      printf("\nInvalid infix Expression.\n");
      getchar();
      exit(1);
   }


   postfix_exp[j] = '\0';

}

int main()
{
   char infix[SIZE], postfix[SIZE];
```

```
    printf("\nEnter Infix expression : ");
    gets(infix);
    InfixToPostfix(infix,postfix);
    printf("Postfix Expression: ");
    puts(postfix);
    return 0;
}
```

5 Output



```
rabeehrz@BatPC:~/college/s3dslab/cycle2$ ./a.out

Enter Infix expression : (a+b-(c/d)*h)
Postfix Expression: ab+cd/h*-
rabeehrz@BatPC:~/college/s3dslab/cycle2$
```

6 Result

A C program to convert the given input infix expression to postfix output expression was tested ant the output was verified.