**Collage : Vishwakarma Institute of Technology**

**Course Name : Data Structure in C**

**Name : Vedika Vikas Sontakke**

**Roll no : 37**

**PRN NO 12220206**

Assignment No – 3 : Implement a stack for following expression conversion.

1. Infix to Prefix

Program :

#include<stdio.h>

#include<string.h>

// operator stack

void pushopr(char);

char popopr();

char peepopr();

void displayopr();

//output stack

void pushout(char);

void displayout();

int getpriority(char);

char opr[25] = {'\0'};

char out[25] = {'\0'};

int topopr = -1;

int topout = -1;

void main()

{

   char infix[25] = {'\0'} , ele , pop\_ele;

   printf("enter infix expression : \n");

   scanf("%s",infix);

   printf("infix expression is : %s\n" , infix);

   int i = strlen(infix)-1;

   while(i >= 0)

   {

     ele = infix[i];

     if(ele == ')') pushopr(ele);

     else if(ele == '('){

        while(peepopr() != ')'){

            pop\_ele = popopr();

            pushout(pop\_ele);

        }

        popopr();

     }

     else if(ele == '^' || ele == '\*' || ele == '/' || ele == '+' || ele == '-')

     {

         if(topopr >= 0) {

            while(getpriority(peepopr()) > getpriority(ele) && topopr != -1) {

                pop\_ele = popopr();

                pushout(pop\_ele);

            }

         }

         pushopr(ele);

     }

     else

         pushout(ele);

     displayopr();

     displayout();

     i--;

   }

   if(topopr != -1){

     while(topopr != -1) {

        pop\_ele = popopr();

        pushout(pop\_ele);

     }

   }

   printf("\nprefix expression is : %s \n", strrev(out));

}

void pushopr(char ele){

    if(topopr == 24) printf("\n operator stack is full\n");

    else opr[++topopr] = ele;

}

char popopr()

{

    if(topopr != -1) return opr[topopr--];

}

char peepopr()

{

   if(topopr != -1) return opr[topopr];

}

void displayopr()

{

    printf("\noperator stack is :");

    for(int i=0 ; i<=topopr ; i++) printf("| %c ", opr[i]);

}

int getpriority(char ele)

{

   switch(ele)

   {

     case '^' : return 3;

     case '\*' :

     case '/' : return 2;

     case '+' :

     case '-' : return 1;

   }

   return -1;

}

void pushout(char ele)

{

    if(topout == 24) printf("output stack is full\n");

    else out[++topout] = ele;

}

void displayout()

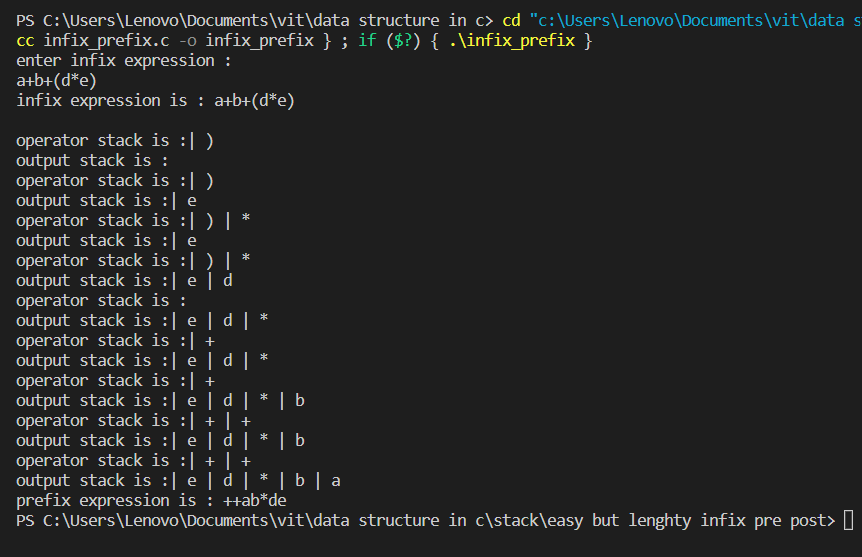
{

    printf("\noutput stack is :");

    for(int i=0 ; i<=topout ; i++)  printf("| %c ", out[i]);

}

Output :



1. Infix to postfix

Program :

#include<stdio.h>

// operator stack

void pushopr(char);

char popopr();

char peepopr();

void displayopr();

//output stack

void pushout(char);

void displayout();

int getpriority(char);

char opr[25] = {'\0'};

char out[25] = {'\0'};

int topopr = -1;

int topout = -1;

void main()

{

   char infix[25] = {'\0'} , ele , pop\_ele;

   int i=0;

   printf("enter infix expression : \n");

   scanf("%s",infix);

   printf("infix expression is : %s\n" , infix);

   while(infix[i] != '\0')

   {

     ele = infix[i];

     if(ele == '(') pushopr(ele);

     else if(ele == ')'){

        while(peepopr() != '('){

            pop\_ele = popopr();

            pushout(pop\_ele);

        }

        popopr();

     }

     else if(ele == '^' || ele == '\*' || ele == '/' || ele == '+' || ele == '-')

     {

         if(topopr >= 0) {

            while(getpriority(peepopr()) >= getpriority(ele)) {

                pop\_ele = popopr();

                pushout(pop\_ele);

            }

         }

         pushopr(ele);

     }

     else

         pushout(ele);

     displayopr();

     displayout();

     i++;

   }

   if(topopr != -1){

     while(topopr != -1) {

        pop\_ele = popopr();

        pushout(pop\_ele);

     }

   }

   printf("\n Postfix expression is : %s \n", out);

}

void pushopr(char ele){

    if(topopr == 24) printf("\n operator stack is full\n");

    else opr[++topopr] = ele;

}

char popopr()

{

    if(topopr != -1) return opr[topopr--];

}

char peepopr()

{

   if(topopr != -1) return opr[topopr];

}

void displayopr()

{

    printf("\noperator stack is :");

    for(int i=0 ; i<=topopr ; i++) printf("| %c ", opr[i]);

}

int getpriority(char ele)

{

   switch(ele)

   {

     case '^' : return 3;

     case '\*' :

     case '/' : return 2;

     case '+' :

     case '-' : return 1;

   }

   return -1;

}

void pushout(char ele)

{

    if(topout == 24) printf("output stack is full\n");

    else out[++topout] = ele;

}

void displayout()

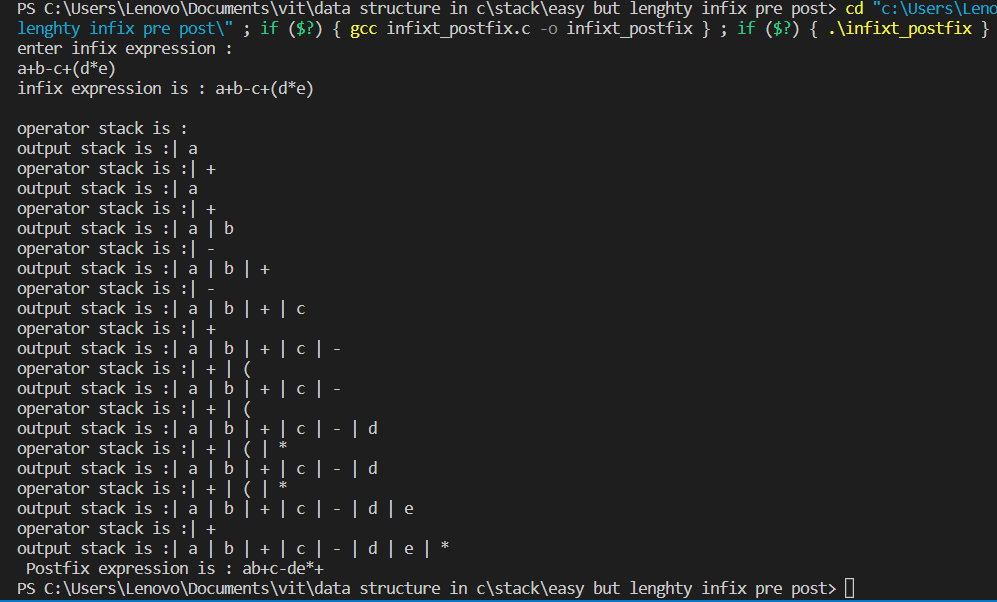
{

    printf("\noutput stack is :");

    for(int i=0 ; i<=topout ; i++)  printf("| %c ", out[i]);

}

Output :



1. Prefix to Infix

Program :

#include<stdio.h>

#include<string.h>

#include<ctype.h>

char stack[25] = {'\0'};

int top = -1;

char ch;

void push(char);

char pop();

void display();

void main()

{

   char prefix[25] = {'\0'} , ele;

   int j ;

   printf("enter prefix expression : \n");

   scanf("%s",prefix);

   printf("prefix expression is : %s\n" , prefix);

   int i = strlen(prefix)-1;

   while(i>=0)

   {

      ele = prefix[i];

       if((isalnum(ele)))

         {

            push(ele);

            push(' ');

            printf("\n ele = %c push to the stack",ele);

         }

         else

         {

             printf("\n operator = %c" , ele);

             for(j= top-1 ; j>=0 ; j--)

             {

                if(stack[j]==' ')

                {

                    stack[j] = ele;

                    break;

                }

             }

         }

    display();

    printf("\n-------------------------------------------------------------------\n");

      i--;

   }

    printf("\n Infix expression is %s ", strrev(stack));

}

void push(char ele)

{

    if(top == 24) printf("stack is overflow\n");

    else stack[++top] = ele;

}

char pop()

{

    if(top == -1) printf("stack is empty\n");

    else return stack[top--];

}

void display()

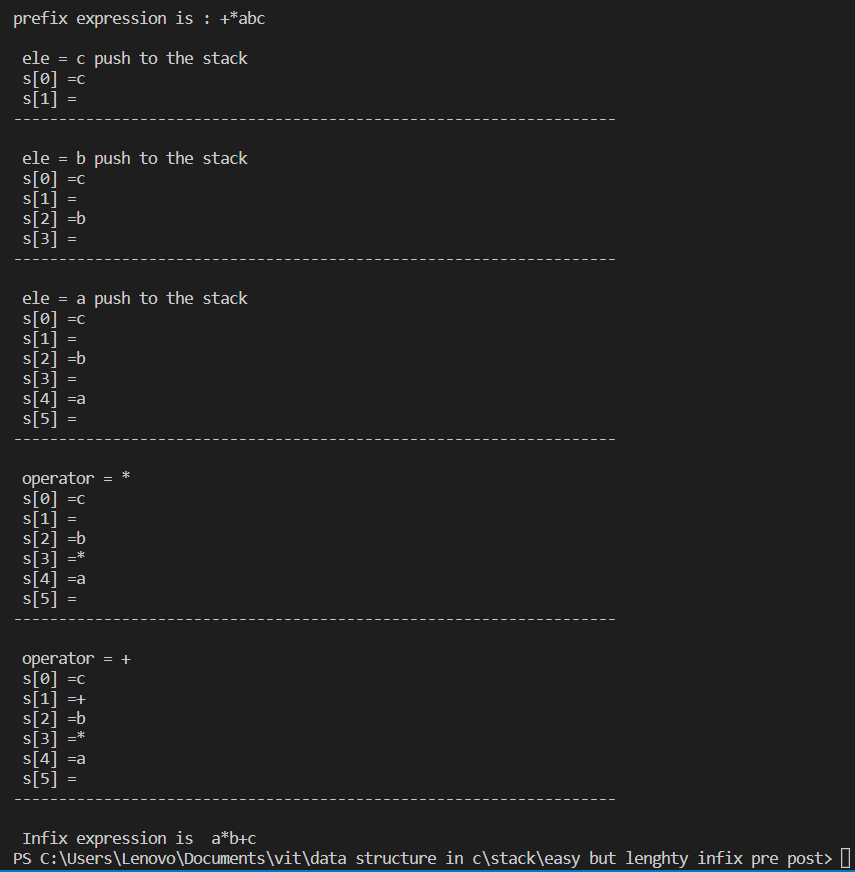
{

    int i;

    for(int i=0 ; i<=top ; i++) printf("\n s[%d] =%c", i, stack[i]);

}

Output :



1. Postfix to Infix

Program :

#include<stdio.h>

#include<string.h>

#include<ctype.h>

void push(char);

char pop();

void display();

char stack[25]={'\0'};

int top = -1;

char ch;

int main()

{

    char postfix[25] = {'\0'} , ele;

    int i=0 , j ;

   printf("enter postfix expression : \n");

   scanf("%s",postfix);

   printf("postfix expression is : %s\n" , postfix);

    while(i < strlen(postfix))

    {

         ele = postfix[i];

         if((isalnum(ele)))

         {

            push(ele);

            push(' ');

            printf("\n ele = %c push to the stack",ele);

         }

         else

         {

             printf("\n operator = %c" , ele);

             for(j= top-1 ; j>=0 ; j--)

             {

                if(stack[j]==' ')

                {

                    stack[j] = ele;

                    break;

                }

             }

         }

         display();

         printf("\n-------------------------------------------------------------------\n");

         i++;

    }

    printf("infix expression is %s", stack);

}

void push(char ele)

{

    if(top == 24) printf("stack is overflow\n");

    else stack[++top] = ele;

}

char pop()

{

    if(top == -1) printf("stack is empty\n");

    else return stack[top--];

}

void display()

{

    int i;

    for(int i=0 ; i<=top ; i++) printf("\n s[%d] =%c", i, stack[i]);

}

Output :

