AIM:-Program to convert infix to prefix.

Algorithm

Step 1: First, reverse the infix expression given in the program.

Step 2: Scan the expression from left to right.

Step 3: Whenever the operand arrive, print them.

Step 4: If the operator arrives and the stack is found is empty, then push the incoming operator into the stack.

Stack 5: If the incoming operator has higher precedence or same precedence then replace the operator.

Step 6: When we reach the end the expression, pop, and print all the operators from top and the stack.

Step 7: If the operator is ‘(‘, then push it into the stack.

Step 8: If the operator is ‘(‘, then pop all the operators from the stack file it find ‘)’ openly bracket in the stack.

Step 9: At the end, reverse the output.

Program: -

//Infix to postfix//

#include<stdio.h>//standard input output header file//

#include<stdlib.h>//standard library header file//

#include<ctype.h>//set of functions to classify//

#include<string.h>//string header file//

#define max 100//initializing the max value to the 100//

int top=-1, a[max];//initializing the top value to the -1//

void push(char x)//push function//

{

a[++top]=x;

}

char pop()//pop function//

{

if(top==-1)//checks the condition//

return -1;

else

return a[top--];

}

int prcd(char c)

{

if(c==')')//checks the condition//

return 0;

else if(c=='+'||c=='-')//checks the condition//

return 1;

else if(c=='\*'||c=='/')//checks the condition//

return 2;

}

void strrev(char \*exp)//string reverse function//

{

char temp[50];

int size=strlen(exp);

temp[size--]='\0';

int i=0;

while(exp[i]!='\0')//checks the condition//

{

temp[size]=exp[i];

i++;

size--;

}

strcpy(exp,temp);

}

void infixtoprefix(char infix[max],char prefix[max])//infix to postfix function//

{

char temp,x;

int i=0,j=0;

strrev(infix);

while(infix[i]!='\0')//checks the condition//

{

temp=infix[i];

if(isalnum(temp))//checks the condition//

{

prefix[j++]=temp;

}

else if(temp==')')

push(temp);//push the element in to a stack//

else if(temp=='(')

{

while((x=pop())!=')')

{

prefix[j++]=x;

}

}

else

{

while(prcd(a[top])>=prcd(temp))//checks the condition//

{

prefix[j++]=pop();

}

push(temp);//pushes the temp in to the stack//

}

i++;//increment to next one//

}

while(top!= -1)//checks the condition//

prefix[j++]=pop();

prefix[j]='\0';

strrev(prefix);

}

int main()//main function//

{

char infix[max],prefix[max];

printf("Enter the infix expression\n");//printing statement//

scanf("%s",infix);//store the expression in the infix space//

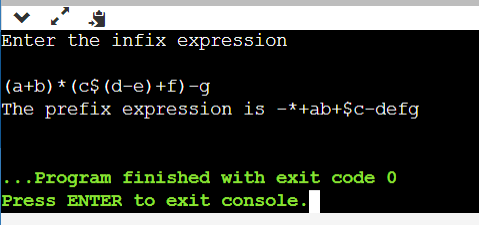
infixtoprefix(infix, prefix);

printf("The prefix expression is %s\n",prefix);//printing the answer//

return 0;

}

Screen shot of the output: -



Link of the program in git hub: -

https://github.com/Koushik123123/datastructures-1/blob/main/20\_12\_22\_1NT21IS109\_program%204.docx