

```

infostopofix.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins Doxygen Settings Help
Start here X infostopofix.c
1 #include<stdio.h>
2 #include<limits.h>
3 #include<string.h>
4 #define MAX 100
5 #define MIN -(MAX)
6 int top=-1;
7 void push(char c){
8     if(top==MAX-1)
9         printf("stack overflow");
10    return;
11 }
12 stack[++top]=c;
13 }
14 char pop(){
15     if(top==0)
16         printf("stack is underflow");
17     return -1;
18 }
19 return stack[top--];
20 }
21 char peek(){
22     if (top==0) return -1;
23     return stack[top];
24 }
25 int precedence(char op){
26     switch (op){
27     case '+': case '-':
28         return 1;
29     case '*': case '/':
30         return 2;
31     case '^':
32         return 3;
33     case ')':
34         return 0;
35     case '(':
36         return 4;
37     default:
38         return 0;
39 }
40 }
41 int ass(char op){
42     if (op=='+'||op=='-')
43         return 1;
44     return 2;
45 }
46 return 0;
47 void infixtopostfix(char infix[], char postfix[]){
48     int i,k=0;
49     char c;
50     for(i=0;infix[i]!='\0';i++){
51         c=infix[i];
52         if(isalnum(c))
53             postfix[k++]=c;
54         else if(c=='(')
55             push(c);
56         else if(c==')'){
57             while(pop()!='(')
58                 postfix[k++]=pop();
59         }
60         else if(c=='*'||c=='/'||c=='^'){
61             while (top!=0 && (precedence(postfix[top])>precedence(c)) || (precedence(postfix[top])==precedence(c) && ass(c)<ass(postfix[top])) )
62                 postfix[k++]=pop();
63             pop();
64         }
65         else if(c=='-'){
66             while (top!=0 && (precedence(postfix[top])>precedence(c)) || (precedence(postfix[top])==precedence(c) && ass(c)==ass(postfix[top])) )
67                 postfix[k++]=pop();
68             push(c);
69         }
70     }
71     while(top!=0)
72         postfix[k++]=pop();
73     postfix[k]='\0';
74 }
75 int main(){
76     char infix[MAX], Postfix[MAX];
77     printf("Enter a valid parenthesized infix exp:");
78     scanf("%s",infix);
79     infixtopostfix(infix,Postfix);
80     printf("Postfix exp:\n",Postfix);
81     return 0;
82 }
83

```

infotopostfix.c - CodeBlocks 20.03

File Edit View Search Project Build Debug Fortran Tools Tools> Plugins DavyBlocks Settings Help

Start here X infotopostfix X

```
1 #include<stdio.h>
2 #include<ctype.h>
3 #include<string.h>
4 #define MAX 100
5 char stack[MAX];
6 int top=-1;
7 void push(char c){
8     if(top==MAX-1)
9         printf("stack overflow");
10    else
11        stack[++top]=c;
12 }
13
14 char pop(){
15     if(top==-1)
16         printf("stack is underflow");
17     else
18         top--;
19 }
20
21 char peek(){
22     if (top== -1) return -1;
23     return stack[top];
24 }
25
26 int precedence(char op){
27     switch (op)
28     {
29         case '+':
30         case '-':
31             return 1;
32         case '*':
33         case '/':
34             return 2;
35         case '^':
36             return 3;
37         case ')':
38             return 0;
39     }
40     return -1;
41 }
42
43 int sgn(char op){
44     if (op=='-')
45         return 1;
46     return 0;
47 }
48
49 void infotopostfix(char infix[], char postfix[]){
50     int i,k=0;
51     char c;
52     for(i=0;infix[i]!='\0';i++){
53         c=infix[i];
54         if(isalnum(c))
55             postfix[k++]=c;
56         else
57             if(sgn(c)==1)
58                 push(c);
59             else
60                 while(sgn(stack[top])>sgn(c))
61                     postfix[k++]=pop();
62                 push(c);
63     }
64 }
```

D:\chethan\IP\infotopostfix.exe

enter a valid parenthesized infix exp:A\*B+C\*D-E  
postfix exp:AB\*CDE+

Process returned 0 (0x0) execution time : 267.259 s

Press any key to continue.

C/C++ Windows (CR+LF) WINDOWS-1252 Line 54 Col 1 Pos 974 Insert Read/Write default ENG 06-10-2023

DATE: \_\_\_\_\_  
PAGE: \_\_\_\_\_

Infix to postfix conversion

```
#include <iostream.h>
#include <ctype.h>
#include <string.h>
#define Max 100
char stack[Max];
int top = -1;
```

```
void push(char c) {
    if (top == Max - 1) {
        cout("stack overflow\n");
        return;
    }
```

```
    stack[++top] = c;
}
```

```
char pop() {
    if (top == -1) {
        cout("stack underflow\n");
        return -1;
    }
```

```
    return stack[top--];
}
```

```
char peek() {
    if (top == -1) return -1;
    return stack[top];
}
```

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators.

#include <stdio.h>  
#include <ctype.h>

Algorithm :-

- 1) Start
- 2) Initialize an empty stack for operators
- 3) Initialize an empty string postfix
- 4) Scan the infix expression from left to right, one symbol at a time
- 5) If operand is 'c' push to stack.
- 6) If operand is 'y' pop from stack and append it to postfix until '(' open bracket.
- 7) If operator (+,-,\*.)
  - \* If operator have high precedence push to stack
  - \* If operator have lower precedence append it to postfix, pop and print the top
  - \* If incoming operator is equal precedence, then go to associativity laws.
- 8) At the end of the expression, pop and print all operators.
- 9) Stop

6/10

DATE \_\_\_\_\_  
PAGE \_\_\_\_\_

```
else if (c == ')') {
    while (peek() != '(') {
        postfix[k++] = pop();
    }
    pop();
}
else {
    while (top != -1 &&
        ((precedence(peek()) > precedence(c)) ||
         (precedence(peek()) == precedence(c) &&
          associativity(c) == 0))) {
        postfix[k++] = pop();
    }
    push(c);
}
while (top != -1)
{
    postfix[k++] = pop();
}
postfix[k] = '\0';
}

int main() {
    char infix[Max], postfix[Max];
    printf("Enter a valid parenthesized infix expression:");
    scanf("%s", infix);
    infixtopostfix(infix, postfix);
    printf("postfix expression: %s\n", postfix);
    return 0;
}
```

DATE: \_\_\_\_\_  
PAGE: \_\_\_\_\_

```
int precedence (char op) {
    switch (op) {
        case '+':
        case '-':
            return 2;
        case '*':
        case '/':
            return 3;
        case '^':
            return 4;
        default:
            return 0;
    }
}

int associativity (char op) {
    if (op == '^')
        return 1;
    return 0;
}

void infixtopostfix (char infix[], char postfix[]) {
    int i, k = 0;
    char c;
    for (i = 0; infix[i] != '\0'; i++) {
        c = infix[i];
        if (c isalnum(c)) {
            postfix[k++] = c;
        } else if (c == '(') {
            push (c);
        }
    }
}
```

