

IMPLEMENT INFIX TO POSTFIX CONVERSION USING STACK IN C

DATA STRUCTURE & ALGORITHM PROJECT

Guided by Mrs. SONAM MODI

MEMBER NAME

ENROLLMENT NUMBER:- 216400307127

NAME:- SOLANKI TRUSHEN

ENROLLMENT NUMBER:- 216400307129

NAME:- PITRODA RUDRA

ENROLLMENT NUMBER:- 216400307125

NAME:- RATHOD ROHIT

ENROLLMENT NUMBER:- 216400307118

NAME:- BHAVSAR DAKSH

INDEX

SR NO.	TOPIC NAME	PAGE NO.
1	General	3
2	Infix to Postfix convert	4
3	Advantages of Postfix expression	5
4	Algorithm of convert Infix to Postfix	6
5	Infix to Postfix conversion	7
6	Code screenshot	8-9
7	Output screenshot	10

General

- One of the applications of Stack is in the conversion of arithmetic expressions in high-level programming languages into machine readable form. As our computer system can only understand and work on a binary language, it assumes that an arithmetic operation can take place in two operands only e.g., $A+B$, $C*D$, D/A etc. But in our usual form an arithmetic expression may consist of more than one operator and two operands e.g. $(A+B)*C(D/(J+D))$.
- These complex arithmetic operations can be converted into polish notation using stacks which then can be executed in two operands and an operator form.

Infix & Postfix

❑ Infix Expression

- It follows the scheme of **<operand><operator><operand>** i.e. an **<operator>** is preceded and succeeded by an **<operand>**. Such an expression is termed infix expression. E.g., **A+B**

❑ Postfix Expression

- It follows the scheme of **<operand><operand><operator>** i.e. an **<operator>** is succeeded by both the **<operand>**. E.g., **AB+**

Advantage of Postfix Expression

□ Advantage of Postfix Expression over Infix Expression

- An infix expression is difficult for the machine to know and keep track of precedence of operators. On the other hand, a postfix expression itself determines the precedence of operators (as the placement of operators in a postfix expression depends upon its precedence). Therefore, for the machine it is easier to carry out a postfix expression than an infix expression.

Algorithm to convert Infix To Postfix

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

1. Push “(“ onto Stack, and add “)” to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered ,then:
 1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
 2. Add operator to Stack.
[End of If]
6. If a right parenthesis is encountered ,then:
 1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.
 2. Remove the left Parenthesis.
[End of If]
[End of If]

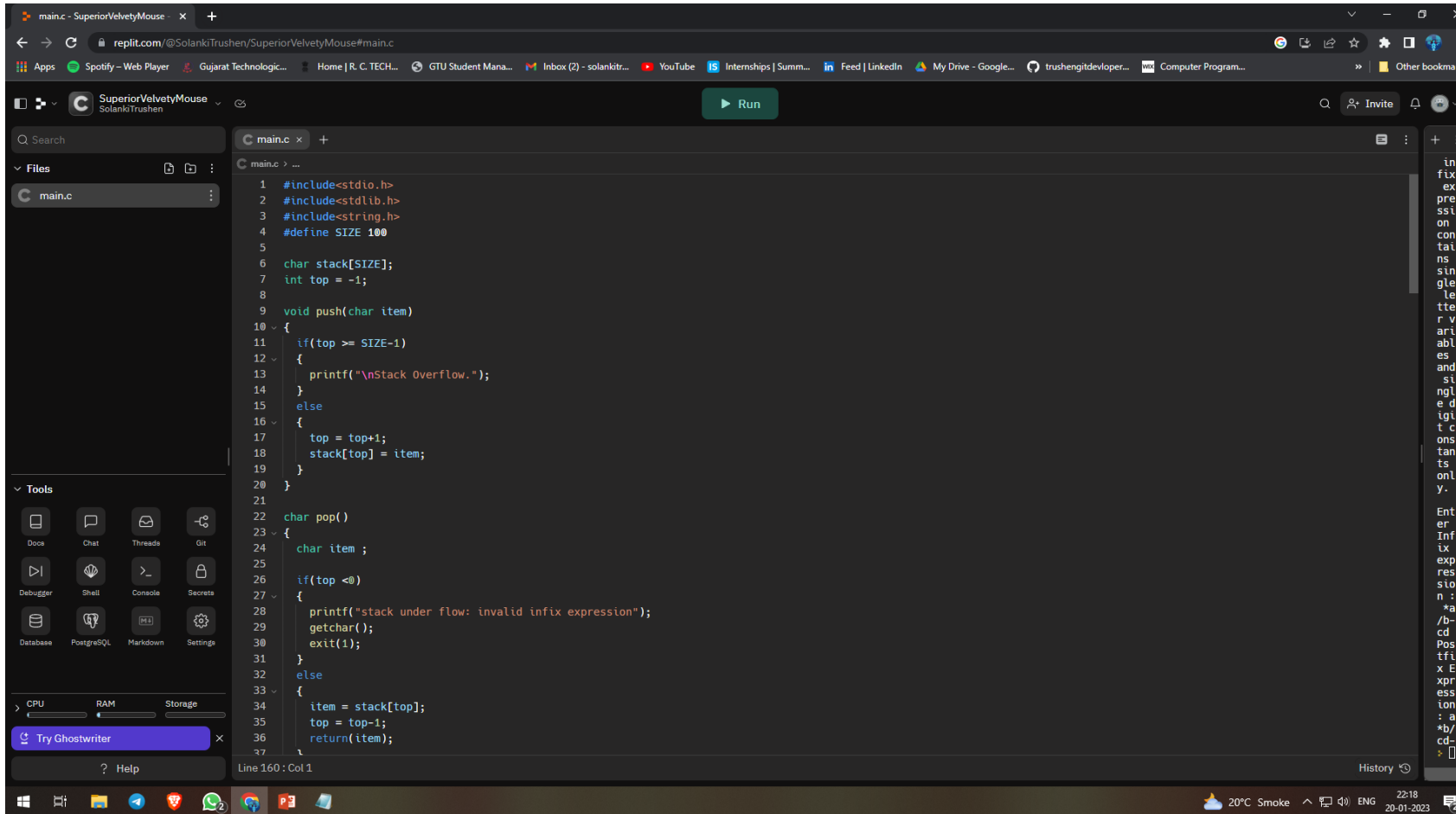
Infix to Postfix conversion

Infix Expression: $A + (B * C - (D / E ^ F) * G) * H$, where $^$ is an exponential operator.

Symbol	Scanned	STACK	Postfix Expression	Description
1.		(Start
2.	A	(A	
3.	+	(+	A	
4.	((+ (A	
5.	B	(+ (AB	
6.	*	(+ (*	AB	
7.	C	(+ (*	ABC	
8.	-	(+ (-	ABC*	'*' is at higher precedence than '-'
9.	((+ (- (ABC*	
10.	D	(+ (- (ABC*D	
11.	/	(+ (- /	ABC*D	
12.	E	(+ (- /	ABC*DE	
13.	^	(+ (- / ^	ABC*DE	
14.	F	(+ (- / ^	ABC*DEF	
15.)	(+ (-	ABC*DEF^/	Pop from top on Stack , that's why '^' Come first
16.	*	(+ (- *	ABC*DEF^/	
17.	G	(+ (- *	ABC*DEF^/G	
18.)	(+	ABC*DEF^/G*-	Pop from top on Stack , that's why '^' Come first
19.	*	(+ *	ABC*DEF^/G*-	
20.	H	(+ *	ABC*DEF^/G*-H	
21.)	Empty	ABC*DEF^/G*-H*+	END

Resultant Postfix Expression: $ABC*DEF^/G*-H*+$

Code screen shot



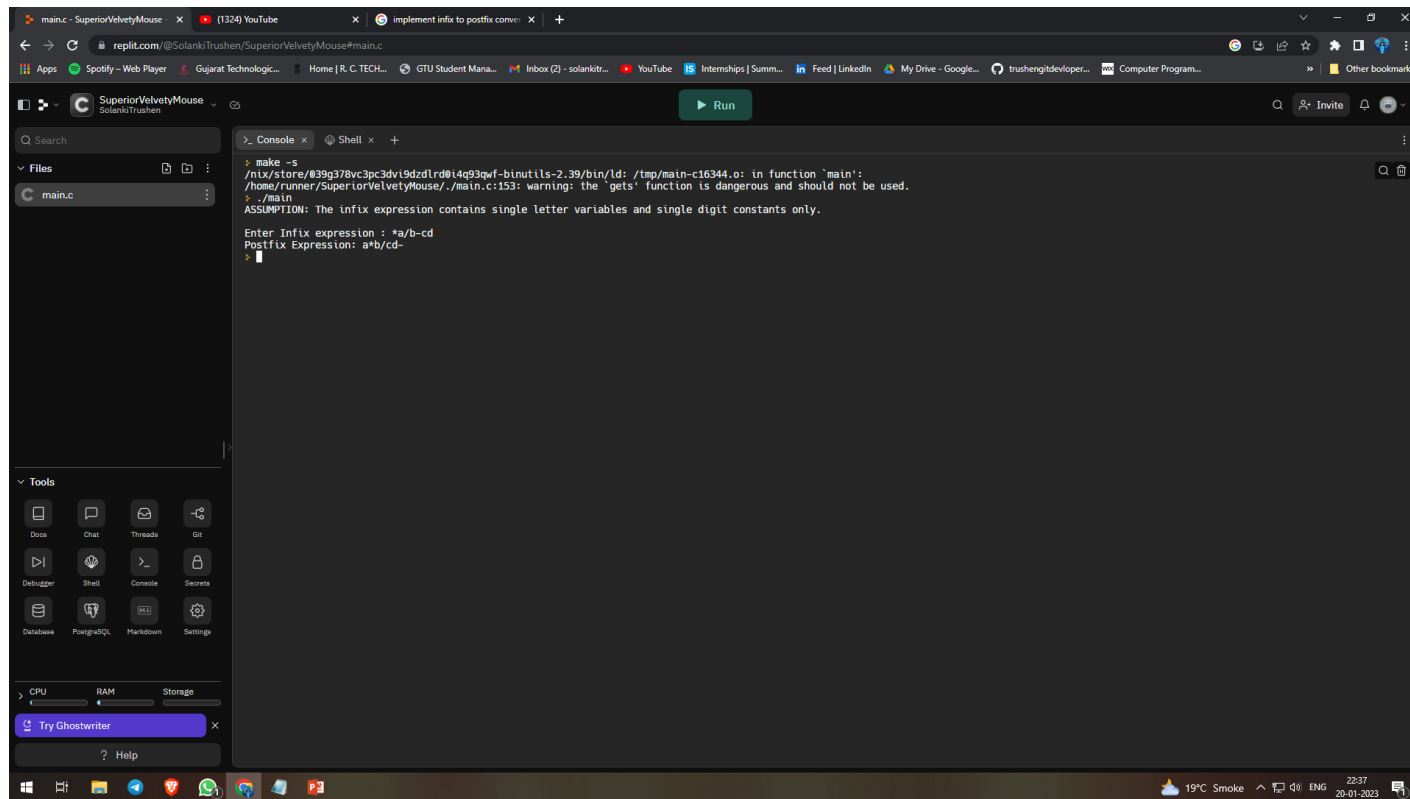
```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #define SIZE 100
5
6 char stack[SIZE];
7 int top = -1;
8
9 void push(char item)
10 {
11     if(top >= SIZE-1)
12     {
13         printf("\nStack Overflow.");
14     }
15     else
16     {
17         top = top+1;
18         stack[top] = item;
19     }
20 }
21
22 char pop()
23 {
24     char item ;
25
26     if(top <=0)
27     {
28         printf("stack under flow: invalid infix expression");
29         getchar();
30         exit(1);
31     }
32     else
33     {
34         item = stack[top];
35         top = top-1;
36         return(item);
37     }
38 }
```

infix expression notation: a/b+c*d


```
1 int is_operator(char symbol)
2 {
3     if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-')
4     {
5         return 1;
6     }
7     else
8     {
9         return 0;
10    }
11 }
12
13 int precedence(char symbol)
14 {
15     if(symbol == '^')
16     {
17         return(3);
18     }
19     else if(symbol == '*' || symbol == '/')
20     {
21         return(2);
22     }
23     else if(symbol == '+' || symbol == '-')
24     {
25         return(1);
26     }
27     else
28     {
29         return(0);
30     }
31 }
32
33 void infixToPostfix(char infix_exp[], char postfix_exp[])
34 {
35     int i, j;
36     char item;
```

```
37 {
38     int i, j;
39     char item;
40     char x;
41
42     push('^');
43     strcat(infix_exp, "");
44
45     i=0;
46     j=0;
47     item=infix_exp[i];
48
49     while(item != '\0')
50     {
51         if(item == '(')
52         {
53             push(item);
54         }
55         else if (isdigit(item) || isalpha(item))
56         {
57             postfix_exp[j] = item;
58             j++;
59         }
60         else if(is_operator(item) == 1)
61         {
62             x=pop();
63             while(is_operator(x) == 1 && precedence(x)>= precedence(item))
64             {
65                 postfix_exp[j] = x;
66                 j++;
67                 x = pop();
68             }
69             push(x);
70
71             postfix_exp[j] = item;
72             j++;
73         }
74     }
75 }
```

Output screen shot



The screenshot shows a web-based IDE interface. The top browser tab is titled "main.c - SuperiorVelvetyMouse". The address bar shows the URL "replit.com/@SolankiTrushen/SuperiorVelvetyMouse#main.c". The IDE has a dark theme. On the left, there is a sidebar with a search bar, a file explorer showing "main.c", and a tools panel with icons for Docs, Chat, Threads, Git, Debugger, Shell, Console, Secrets, Database, PostgreSQL, Markdown, and Settings. The main area is a console window titled "Console" with a "Run" button. The console output shows the execution of a C program. The first line is "make -s". The second line is a warning from the linker: "/nix/store/03g378vc3pc3dvi9dzdld0l4g93qwf-binutils-2.39/bin/ld: /tmp/main-ci6344.o: in function 'main': /home/runner/SuperiorVelvetyMouse/./main.c:153: warning: the 'gets' function is dangerous and should not be used." The third line is the program's output: "ASSUMPTION: The infix expression contains single letter variables and single digit constants only." The fourth line is a prompt: "Enter Infix expression : *a/b-cd". The fifth line is the program's output: "Postfix Expression: a*b/cd-". The bottom status bar shows the system tray with icons for CPU, RAM, Storage, a weather icon (19°C), a smoke icon, a language icon (ENG), and a clock (22:37, 20-01-2023).

```
make -s
/nix/store/03g378vc3pc3dvi9dzdld0l4g93qwf-binutils-2.39/bin/ld: /tmp/main-ci6344.o: in function 'main':
/home/runner/SuperiorVelvetyMouse/./main.c:153: warning: the 'gets' function is dangerous and should not be used.
./main
ASSUMPTION: The infix expression contains single letter variables and single digit constants only.
Enter Infix expression : *a/b-cd
Postfix Expression: a*b/cd-
```