

10/8/24

(5)

Week 02

program and algorithm to convert infix to postfix

1) Scan the given expression from left to right.

2) If the scanned symbol is

(i) operand \rightarrow Place it on the postfix

(ii) left parenthesis \rightarrow push it onto stack

(iii) right parenthesis \rightarrow pop the stack and place them on postfix until meet a left parenthesis.

(iv) Operator \rightarrow if stack is empty or top of stack is a left parenthesis push the operator onto stack.

else: until top of stack is having higher precedence compared to the scanned symbol pop the stack and place them on postfix

\rightarrow Push the operator onto the stack

3) Until (stack becomes empty) pop the stack content and place on postfix.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

struct stack {

int top;

char data [SIZE];

};

typedef struct stack STACK;

void push (STACK *s, char item) {

if (s->top < SIZE - 1) {

s->data[++(s->top)] = item;

} else {

printf("stack overflow\n");

}

}

char pop (STACK *s) {

if (s->top != -1) {

return s->data[(s->top)--];

} else {

printf("stack underflow\n");

return '\0';

}

}


```
int preced (char symbol) {
```

```
switch (symbol) {
```

```
case '^': return 5;
```

```
case '*':
```

```
case '/': return 3;
```

```
case '+':
```

```
case '-': return 1;
```

```
default: return 0;
```

```
}
```

```
}
```

```
void infixpostfix (char infix [20], STACK *S) {
```

```
char postfix [20], symbol, temp;
```

```
int i, j = 0;
```

```
for (i = 0; infix [i] != '\0'; i++) {
```

```
symbol = infix [i];
```

```
if (isalnum (symbol)) {
```

```
postfix [j++] = symbol;
```

```
} else {
```

```
switch (symbol) {
```

```
case '(':
```

```
push (S, symbol);
```

```
break;
```

```
case ')':
```

```
while (S->top != -1 && (temp = pop (S)) != '(')
```

```
{
```

```
postfix [j++] = temp;
```

```
}
```

```
break;
```


case '+';

case '-';

case '*';

case '/';

case '^';

while (s → top != -1 && precd(s → data[s → top])
≥ precd(symbol)) {

postfix[j++] = pop(s); }

← push(s, symbol);

break;

}

}

}

while (s → top != -1) {

postfix[j++] = pop(s);

}

postfix[j] = '\0';

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

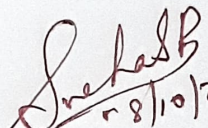
}


```
int main() {  
    char infix [20];  
    STACK S;  
    S.top = -1;  
    printf ("Enter infix expression");  
    scanf ("%s", infix);  
    infixpostfix (infix, &S);  
    return 0;  
}
```

Output :

Enter infix expression : $A + (B + C * D)^E$

Postfix expression is : $ABCD * ^E +$


r 8/10/24