Week2 Batch3

Tuesday, December 07, 2021 10:40 PM

1. Conversion of infix expression to postfix and prefix forms

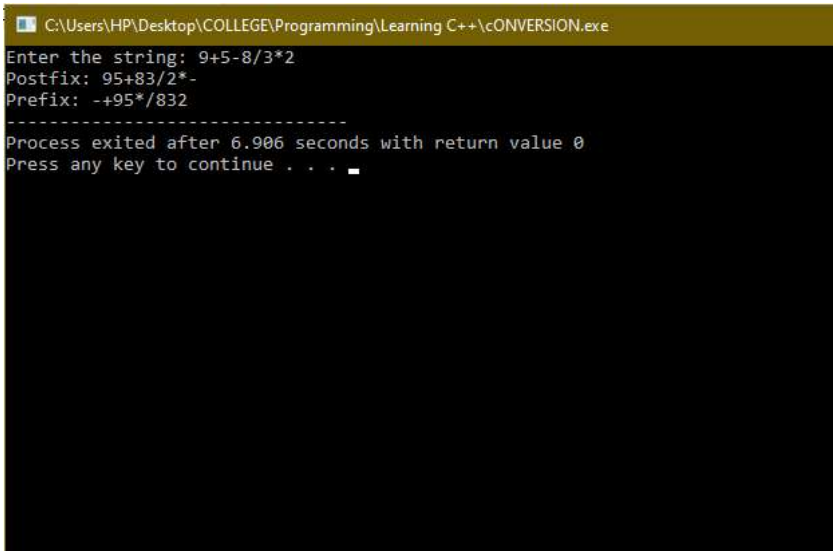
```
#include <iostream>
using namespace std;




int top = -1;
char stack[100];
int max_size = 100;
char answer[100];
int ind;




void push(char c) {
if(top == max_size - 1)
cout << "Stack Overflow";
else {
stack[++top] = c;
}
}




void pop() {
if(top == -1)
cout << "Stack Underflow";
else if(stack[top] == '(')
top--;
else {
answer[ind] = stack[top--];
ind++;
}
}



void decide(char c, int flag) {
if(stack[top] == '(')
push(c);
else {
int priority;
if(c == '^')
priority = 3;
else if (c == '*' || c == '/')
priority = 2;
```

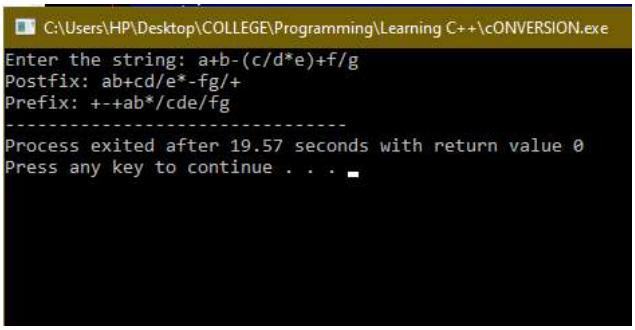
```
else
priority = 1;
int prev;
if(stack[top] == '^')
prev = 3;
else if (stack[top] == '*' || stack[top] == '/')
prev = 2;
else
prev = 1;



if(flag == 0) {
while(priority <= prev) {
if(stack[top] == '(')
break;
if(stack[top] == '^')
prev = 3;
else if (stack[top] == '*' || stack[top] == '/')
prev = 2;
else
prev = 1;
if(priority <= prev)
pop();
}
}
else {
while(priority < prev) {
if(stack[top] == '(')
break;
if(stack[top] == '^')
prev = 3;
else if (stack[top] == '*' || stack[top] == '/')
prev = 2;
else
prev = 1;
if(priority < prev)
pop();
}
}
push(c);
}
}




void pop_till_open() {
while(stack[top] != '(')
pop();
pop();
}




void convert_to_postfix(char infix[100], int flag = 0) {
```

```
ind = 0;
top = -1;
push('(');
int i;
for(i = 0; infix[i] != '\0'; i++);
infix[i] = ' ';
infix[i+1] = ')';
infix[i+2] = '\0';
if(flag == 0)
cout << "Postfix: ";
for(i = 0; infix[i] != '\0'; i++) {
if(infix[i] == '(')
push(infix[i]);
else if (infix[i] == '*' || infix[i] == '/' || infix[i] == '+' || infix[i] == '-' || infix[i] == '^')
decide(infix[i], flag);
else if(infix[i] == ')')
pop_till_open();
else if(infix[i] == ' ')
continue;
else {
answer[ind] = infix[i];
ind++;
}
}
answer[ind] = '\0';
if(flag == 0)
cout << answer;
}
```

```
void convert_to_prefix(char infix[100]) {
// reverse
char reverse[100];
int i, j;
for(i = 0; infix[i] != '\0'; i++);
for(i = i-1, j = 0; i >= 0; i--, j++) {
if(infix[i] == '(')
reverse[j] = ')';
else if(infix[i] == ')')
reverse[j] = '(';
else if (infix[i] == ' ')
continue;
else
reverse[j] = infix[i];
}
reverse[j] = '\0';
// convert to postfix
convert_to_postfix(reverse, 1);
// display answer backward
cout << endl << "Prefix: ";
for(ind = ind - 1; ind > 0; ind--)
cout << answer[ind];
}
```

```
int main() {
char stk[100];
cout << "Enter the string: ";
gets(stk);
convert_to_postfix(stk);
convert_to_prefix(stk);
return 0;
}
```





2. Evaluation of postfix and prefix expressions

```
#include <iostream>
#include <string.h>
using namespace std;
int stack[20];
int top = -1;
void push(int operand)
{
top++;
stack[top] = operand;
}
char pop()
```

```
{
int f;
f = stack[top];
stack[top] = '\0';
top--;
return f;
}
bool isOperand(char c)
{
return (c >= '0' && c <= '9');
}
int PrefixExpression(char prefix[50])
{
int length = strlen(prefix);
for (int i = length - 1; i >= 0; i--)
{
if (prefix[i] == ' ')
continue; if (isOperand(prefix[i]))
{
push(prefix[i] - '0');
}
else
{
int v1 = stack[top];
pop();
int v2 = stack[top];
pop();
switch (prefix[i])
{
case '*':
push(v1 * v2);
break;
case '/':
push(v1 / v2);
break;
case '+':
push(v1 + v2);
break;
case '-':
push(v1 - v2);
break;
}
}
}
return stack[top];
}
int PostfixExpression(char postfix[50])
{
int length = strlen(postfix);
for (int i = 0; i < length; i++)
{
if (postfix[i] == ' ')
continue; if (isOperand(postfix[i]))
{
push(postfix[i] - '0');
} else
{
int v1 = stack[top];
pop();
```

```
int v2 = stack[top];
pop();
switch (postfix[i])
{
case '*':
push(v2 * v1);
break;
case '/':
push(v2 / v1);
break;
case '+':
push(v2 + v1);
break;
case '-':
push(v2 - v1);
break;
}
}
}
return stack[top];
}
int main()
{
int o;
cout<<"1) Evaluate Postfix Expression\n2) Evaluate Prefix Expresion\n"<<endl;
cin>>o;
if(o == 1){
char postfix[50];
cout<<"Input Postfix: "<<endl;
cin>>postfix;
int result = PostfixExpression(postfix);
cout<<"Evaluation: "<<result<<endl;
}
if(o == 2){
char prefix[50];
cout<<"Input Prefix: "<<endl;
cin>>prefix;
int result = PrefixExpression(prefix);
cout<<"Evaluation: "<<result<<endl;
}
}
```