PRÁCTICA 2

Abel Eduardo Robles Lázaro

13 DE MARZO DE 2023

UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Desarrollo:

Construyendo sobre la base que fue la práctica 1, la primera adición fue la variable global "cadenaPF" cuyo objetivo era almacenar la construcción de la cadena postfijo en un string:

```
string cadenaEntrada = "";
string cadenaPF = "";
char analisis;
size_t indice = 0;
```

Además de agregar la línea para imprimir el resultado dentro del main:

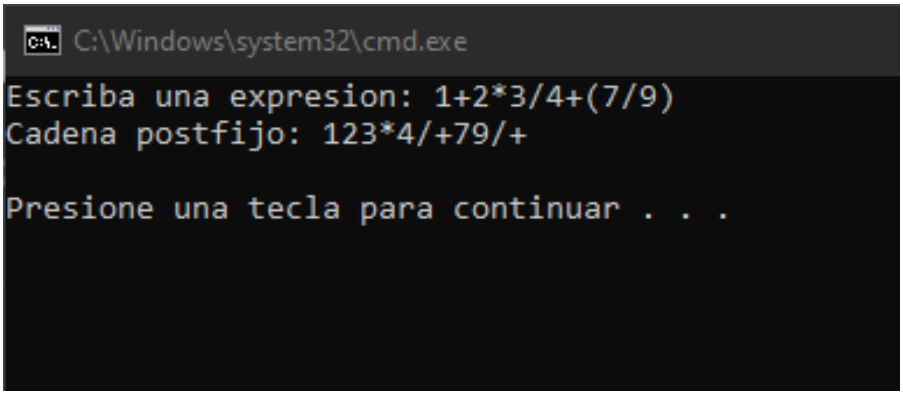
```
cout << "Cadena postfijo: " << cadenaPF << endl;
```

Dentro de cada función agregué una variable char que almacenara el token de análisis, el cuál se agregaría a la cadena postfijo antes de terminar con su función y después de haber llamado a otras funciones:

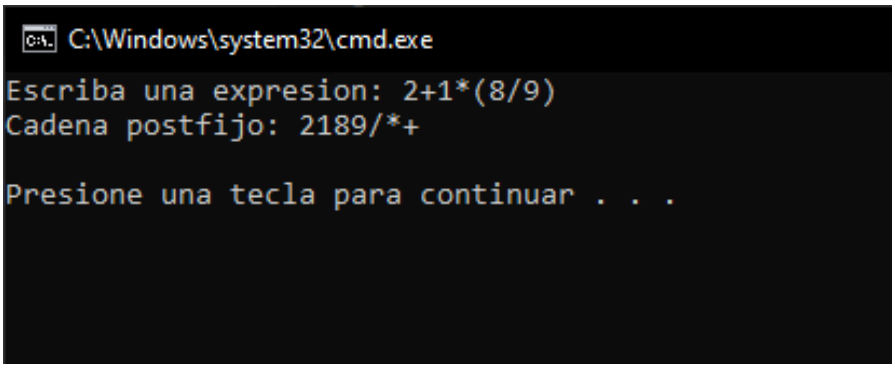
```
void expr_()
{
    char pf;
    while (true)
    {
        analisis = cadenaEntrada[indice];
        pf = analisis;
        switch (analisis)
        {
        case '+':
            coincidir('+');
            term();
            cadenaPF += pf;
            continue;

        case '-':
            coincidir('-');
            term();
            cadenaPF += pf;
            continue;
        default:
            return;
        }
    }
}
```

El resultado fue el siguiente:



Omití el proceso de guardado para los paréntesis porque me informaron que en notación postfija no son impresos



Código fuente:

Main.cpp:

```
#include <iostream>
#include <string>

using namespace std;

string cadenaEntrada = "";
string cadenaPF = "";
char analisis;
size_t indice = 0;

void coincidir(char);

void expr();
void expr_();

void term();
void term_();

void factor();
void digito();

int main()
{
    indice = 0;
    cout << "Escriba una expresion: ";
    cin >> cadenaEntrada;
    cadenaEntrada += "!";
    expr();
    if (cadenaEntrada[indice] != '!')
    {
        cerr << "Hay un error en la cadena" << endl;
        exit(1);
    }

    // cout << "Cadena valida" << endl;
    cout << "Cadena postfijo: " << cadenaPF << endl;

    return 0;
}

void coincidir(char curr)
{
    if (analisis != curr)
    {
        cerr << "Hay un error en '" << curr << "'" << endl;
        exit(1);
    }
    indice++;
}

void expr()
{
```

```
    term();
    expr_();
}

void expr_()
{
    char pf;
    while (true)
    {
        analisis = cadenaEntrada[indice];
        pf = analisis;
        switch (analisis)
        {
        case '+':
            coincidir('+');
            term();
            cadenaPF += pf;
            continue;

        case '-':
            coincidir('-');
            term();
            cadenaPF += pf;
            continue;
        default:
            return;
        }
    }
}

void term()
{
    factor();
    term_();
}

void term_()
{
    char pf;
    while (true)
    {
        analisis = cadenaEntrada[indice];
        pf = analisis;
        switch (analisis)
        {
        case '*':
            coincidir('*');
            factor();
            cadenaPF += pf;
            continue;

        case '/':
            coincidir('/');
            factor();
            cadenaPF += pf;
```

```
            continue;

        default:
            return;
        }
    }
}

void factor()
{
    analisis = cadenaEntrada[indice];
    // char pf = analisis;
    switch (analisis)
    {
    case '(':
        coincidir('(');
        // cadenaPF += '(';
        expr();
        coincidir(')');
        // cadenaPF += ')';
        break;
    default:
        digito();
        // cadenaPF += pf;
        break;
    }
}

void digito()
{
    analisis = cadenaEntrada[indice];
    char pf = analisis;
    if (isdigit(analisis))
    {
        coincidir(analisis);
        cadenaPF += pf;
        return;
    }
    cerr << "Se introdujo un digito invalido" << endl;
    exit(1);
}
```

Conclusiones:

El desarrollo de una función para cada regla de producción me llevó por el camino correcto para poder conseguir la práctica 1 y 2, y eventualmente también la impresión de la cadena en postfijo.