Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Práctica 4

Alumno:

Ramírez Cotonieto Luis Fernando.

Grupo: 1CV3

Profesor: Yaxkin Flores Mendoza

CDMX, México 22/Diciembre/2020

Practica 4

Estructura de Datos

1. Introducción

Para nuestro problema una expresión matemática será un medio que permite indicar el orden en que se deben realizar una serie de operaciones para obtener un resultado. Las operaciones se indican mediante operadores, que en nuestro caso representan las operaciones suma, resta, producto, división e igualdad, todas ellas operaciones binarias (necesitan exactamente dos argumentos para poderse evaluar).

Existen varias notaciones para representar expresiones matemáticas, que se diferencian en el orden en que se escriben los argumentos (operandos) de los operadores. Las más relevantes son:

- Notación infija: La notación habitual. El orden es primer operando, operador, segundo operando.
- Notación prefija: El orden es operador, primer operando, segundo operando.
- Notación postfija: El orden es primer operando, segundo operando, operador.
- Notación funcional: Se escribe el operador/función y despues, entre paréntesis, los operadores separados por comas.

La notación infija tiene el problema de que en expresiones con más de un operador existe ambiguedad sobre cual es el orden de evaluación. Por ejemplo, la expresión 8/4/2 se puede interpretar como (8/4)/2 o bien como 8/(4/2). Las otras notaciones no sufren este problema. Para resolver estas ambiguedades, se añaden unas reglas denominadas orden de precedencia de operadores. Cuando dos operadores compiten por el mismo operando (en el ejemplo anterior, el primer y el segundo operador de división se disputan el operando 4) gana el el operador (se evalúa primero) con mayor precedencia, o a igualdad de precedencia, el operador situado más a la izquierda. Las reglas de precedencia habituales son que los operadores división y producto tienen igual precedencia y ganan al resto de operadores, y que los operadores suma y resta tienen igual precedencia y ganan al operador igualdad. Para este programa se utilizará la expresión infija y posfija para darle haver evaluaciones sobre operaciones dadas en un documento externo.

2. Código

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "pila.h"

int precedencia(char opc){

  int pre;

  if(opc == '+' || opc =='-')
      pre = 1;
      else if(opc == '*' || opc == '/')
          pre = 2;
          else if(opc == '^')
              pre = 3;
              else
                pre = 4;

  return pre;
}
```

```
22
23 void verificar_operador(pila* pila, Info inf, int *cont){
24
25   int pre_act,pre_pil,ncont;
26   Info inf_aux;
27   char aux;
28
29   if (empty(*pila))
30     push(pila,inf);
31
32   else{
33     inf_aux = top(*pila);
34     pre_act = precedencia(inf.caracter);
35     pre_pil = precedencia(inf_aux.caracter);
36
37     if(pre_act == 4)
38       {
39         while (!empty(*pila))
40         {
41           printf("%c",inf_aux.caracter);
42           pop(pila);
43           if(empty(*pila))
44             break;
45           else{
46             inf_aux = top(*pila);
47             printf("%c",inf_aux.caracter);
48             pop(pila);
49           }
50         }
51
52       }else if (pre_act == pre_pil)
53       {
54         printf("%c",inf_aux.caracter);
55         pop(pila);
56         push(pila,inf);
57         inf_aux = top(*pila);
58
59       }else if(pre_act > pre_pil){
60           push(pila,inf);
61       }
62       else if(pre_act < pre_pil){
63         while (!empty(*pila))
64         {
65           printf("%c",inf_aux.caracter);
66           pop(pila);
67           if(empty(*pila))
68             break;
69           else{
70             inf_aux = top(*pila);
71             pop(pila);
72           }
73         }
74
75         push(pila, inf);
76       }
77       else
78         printf("Incorrecto\n");
79
80   }
81 }
82
```

```
83  void  cambiar_postfijo(char* op){
84    char carac =' ';
85    int cont = 0,i=0;
86    pila P;
87    Info inf;
88
89    crearpila(&P);
90
91    do
92    {
93      carac = op[i];
94        if(isdigit(carac)){
95          printf("%c",carac);
96          if(cont == 2){
97            cont = 0;
98            inf = top(P);
99            pop(&P);
100           if(empty(P))
101             ++cont;
102
103         }
104       }
105       else{
106         inf.caracter = carac;
107         verificar_operador(&P,inf,&cont);
108       }
109       ++i;
110   }while(carac!= '\n');
111 }
112
113 int main(int argc, char *argv[]) {
114
115   FILE* operaciones = fopen("Operaciones.txt","r+");
116   char* operacion;
117
118   do{
119     fgets(operacion,20,operaciones);
120     printf("\nOperacion: %s",operacion);
121     printf("Notacion postfija:");
122     cambiar_postfijo(operacion);
123     printf("\n");
124   }while(!feof(operaciones));
125
126   fclose(operaciones);
127
128   return 0;
129 }
```

3. Extensión de Funcionalidades

- Podríamos crear una calculadora más compleja que pueda evaluar integrales y derivadas, inclusive ecuaciones diferenciales por diversos metodos.
- Podriamos realizar una visualización paso a paso para poder entender de mejor manera como se mueven las notaciones al evaluar cada expresión.

4. Programas de se emplea una pila

- Gestión de ventanas en Windows o Linux (cuando cerramos una ventana siempre recuperamos la que teníamos detrás).
- Editores de texto u otras herramientas, donde el usuario puede deshacer los cambios mediante la operación "deshacer", la cual extrae el estado del texto o cualquier elemento, antes del último cambio realizado.

5. Capturas de Pantalla

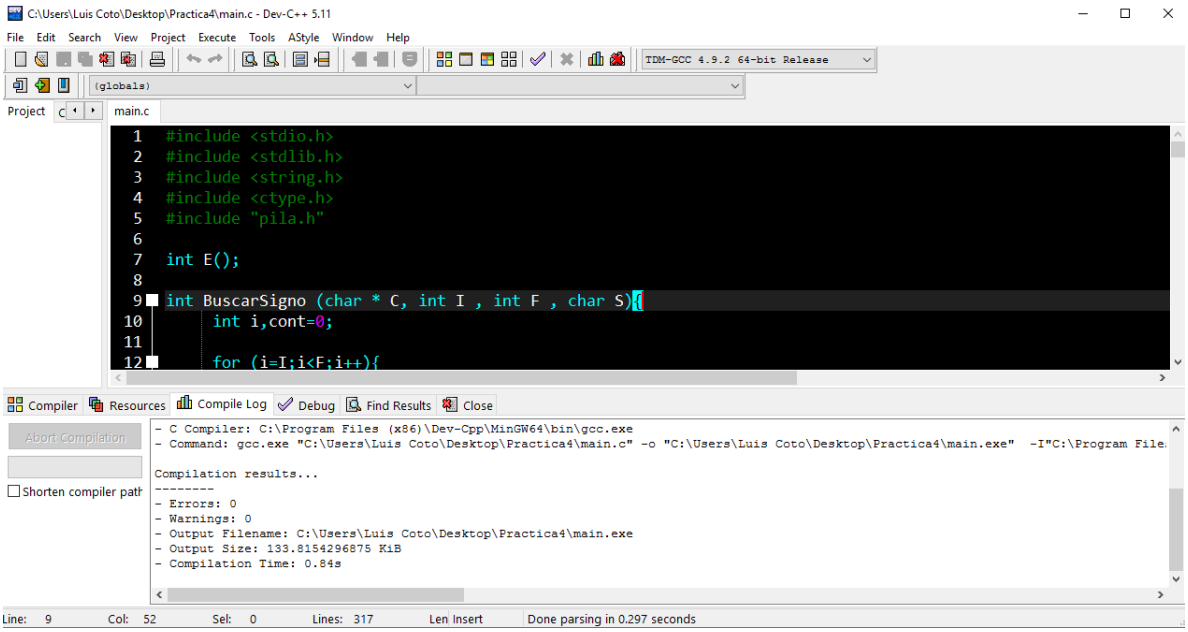

Figura 1: Pantalla de DevC

```
Compiling single file...
--------
- Filename: C:\Users\Luis Coto\Desktop\Practica4\main.c
- Compiler Name: TDM-GCC 4.9.2 64-bit Release

Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "C:\Users\Luis Coto\Desktop\Practica4\main.c" -o "C:\Users\Luis Coto\Desktop\Practica4\main.exe"  -I"C:\Program File
```

Figura 2: Diagnostico de ejecución

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Luis Coto\Desktop\Practica4\main.exe
- Output Size: 133.8154296875 KiB
- Compilation Time: 1.16s
```

Figura 3: Diagnostico de ejecución

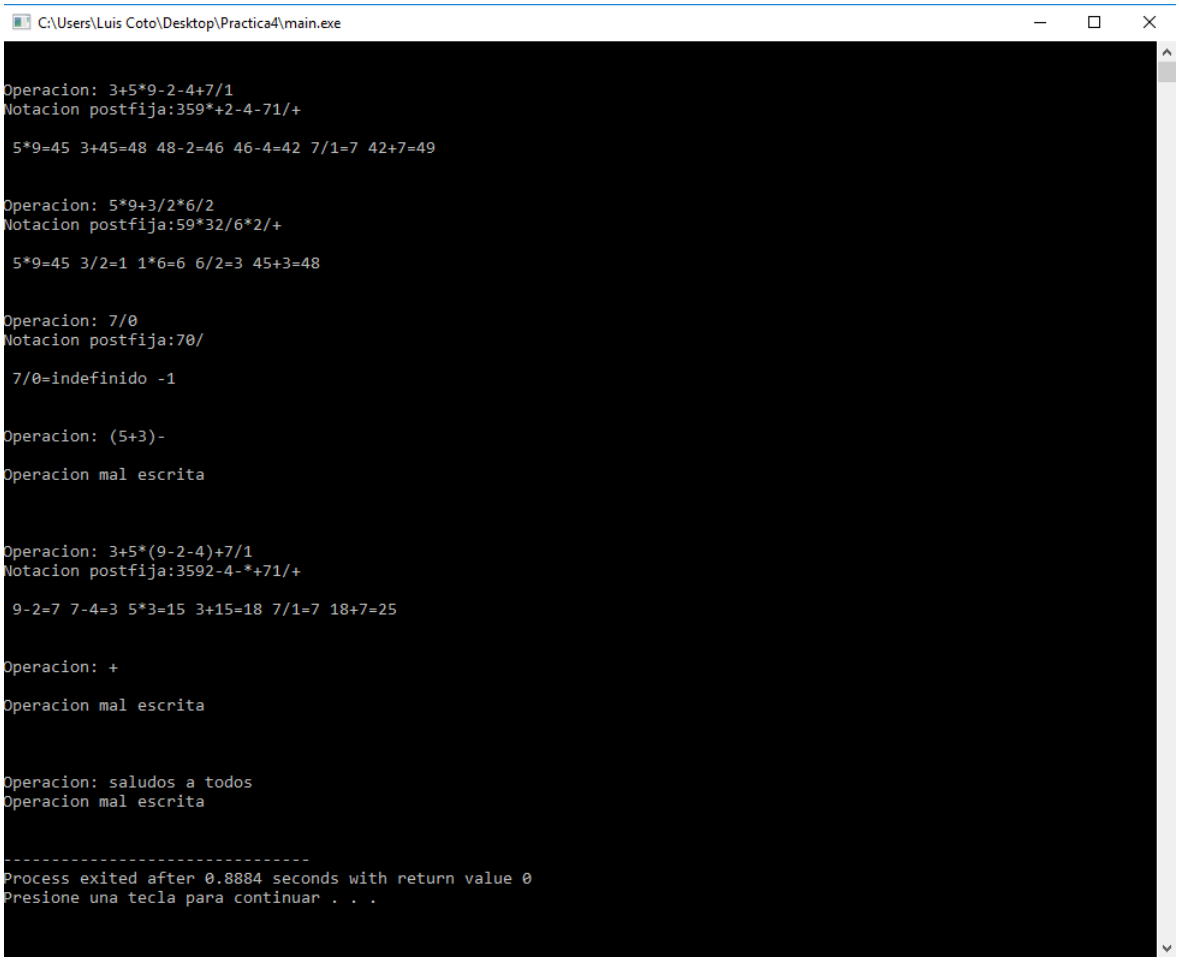

Figura 4: Pantalla completa de la terminal

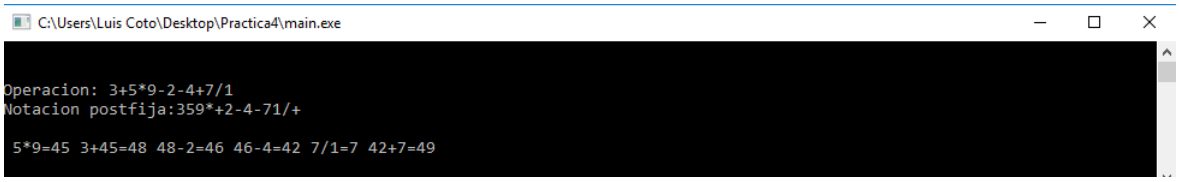

Figura 5: Operación 1

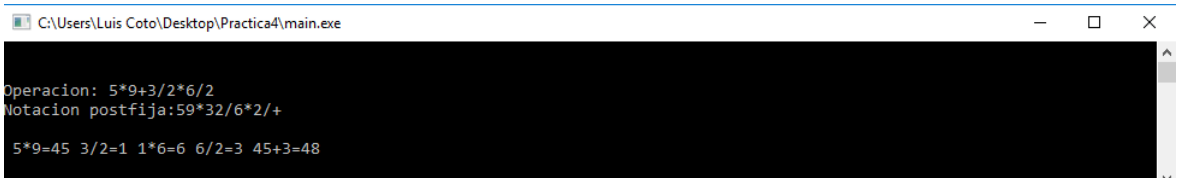

Figura 6: Operación 2

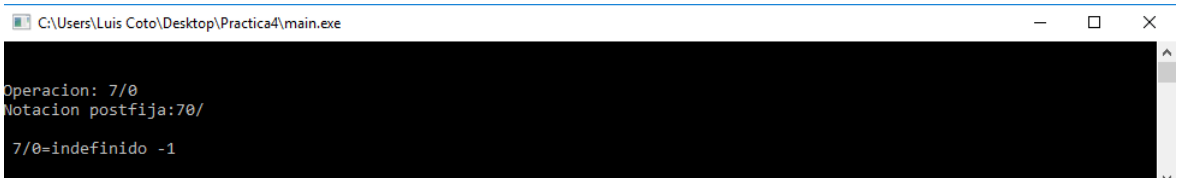

Figura 7: Operación 3

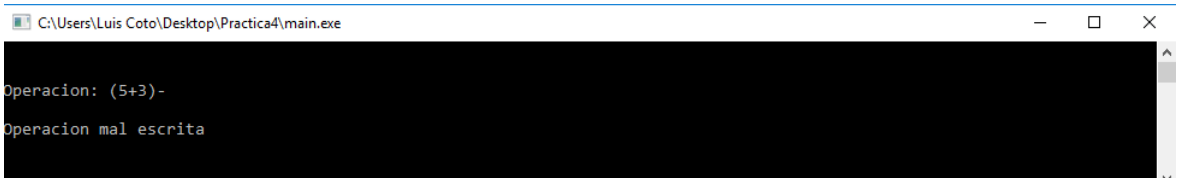

Figura 8: Operación 4

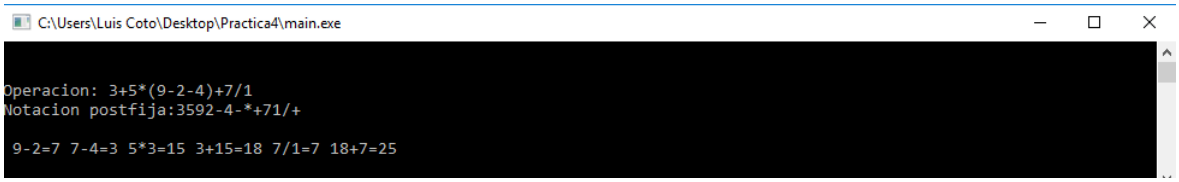

Figura 9: Operación 5

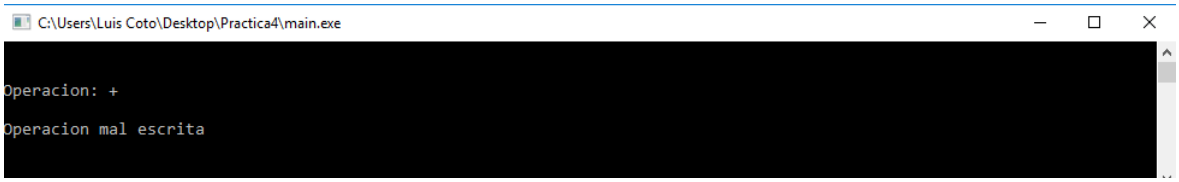

Figura 10: Operación 6

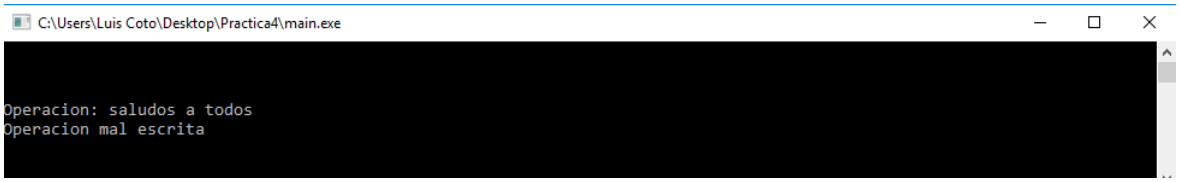

Figura 11: Operación 7