



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESTRUCTURAS DE DATOS

PRÁCTICA 1

PILAS

HERNÁNDEZ CASTELLANOS CÉSAR URIEL

MARTÍNEZ ISLAS MAURICIO JOEL

1CV7

18/02/2017

LUNA BENOSO BENJAMIN



INDICE

Resumen.....	3
Introducción.....	3
Experimentación y resultados.	5
Pseudocódigo.....	14
Validador de expresiones.	14
Evaluación de una expresión en notación postfija	16
Conclusiones.....	18
Anexo.	18
Bibliografía.	21

Resumen.

En el presente reporte se muestra la documentación de la práctica, cuya función principal es la de evaluar una expresión postfija así como la de desarrollar un verificador sintáctico.

Introducción.

Una computadora es una máquina con la capacidad de manipular un gran volumen de información. Por tanto, es de suma importancia comprender los conceptos de organización y manipulación de la información que se realiza mediante las estructuras de datos que no es más que la forma de organizar de una manera particular los datos en una computadora con el propósito de ser utilizado de manera eficiente.

Variadas estructuras de datos se adecuan para diferentes situaciones, algunas otras son altamente especializadas para ciertas tareas.

Las pilas y las colas son dos de las estructuras de datos más utilizadas, debido a su amplio ámbito de aplicación. En esta práctica se estudiarán las pilas, su utilización y su implementación básica en el lenguaje C.

Uno de los conceptos más fundamentales y útiles en las ciencias de la computación es el de la pila.

Una pila es un conjunto ordenado de elementos, en la cual únicamente se pueden agregar y eliminar elementos de ella por un extremo, a la cual llamaremos tope, también es una estructura de tipo LIFO que traducido del inglés significa último en entrar primero en salir.

El método de la pila fue propuesto en 1955 por el alemán Friedrich L. Bauer para posteriormente ser patentada, recibió en 1988 el premio "IEEE Computer Society Pioneer Award" por su aportación.

En cada momento solamente se tiene acceso al último elemento que entró en la pila denominado TOS (Top of Stack), la operación que permite el acceso a este elemento es la de desapilar, permitiendo que el siguiente elemento sea denominado como el nuevo TOS.

Es fácil confundir entre lo que es una pila y un arreglo, pero son totalmente diferentes, a diferencia del arreglo, la definición de pila considera la inserción y eliminación de elementos, por lo que podemos considerar a la pila como un objeto dinámico que se encuentra en constante cambio.

Esta estructura es aplicada en una gran multitud de supuestos en informática, debido a su gran simplicidad y gran capacidad de dar respuesta a diferentes procesos.

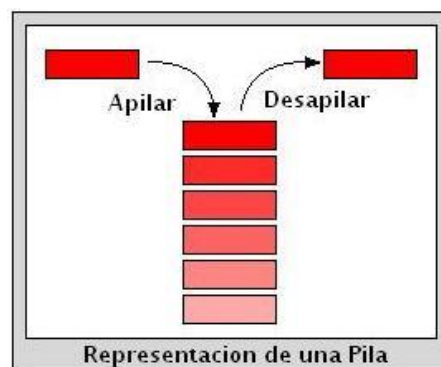
Usualmente las pilas son empleados en las siguientes situaciones:

- Evaluación de expresiones en notación postfija.
- Como reconocedores sintácticos.
- Implementación de recursividad.

Operaciones fundamentales de una pila.

- Tamaño
- Apilar
- Desapilar
- Leer último
- Vacía
- Llena

Es posible implementar una pila de dos maneras, por medio de una matriz o una lista enlazada, la primera será la que desarrollaremos a lo largo de la práctica.



Representación de una pila.

Notación postfija

Es un método algebraico alternativo de introducción de datos.

Se refiere a que el operador ocupa la posición después de los operandos sus características principales son: el orden de los operando se conserva igual que en una expresión infija equivalente.

Su principio es el de evaluar datos directamente cuando se introducen y manejarlos dentro de una estructura LIFO, lo que optimiza los procesos.

Ejemplo: 123^{*+} es 7

Experimentación y resultados.

La práctica consiste en llevar a cabo dos aplicaciones, las cuales tienen como propósito el de calcular una expresión postfija dada que suponemos como cierta, la segunda aplicación tiene como objetivo el verificar si un fichero se encuentra bien escrito en cada una de sus líneas, esto es verificar si tanto los corchetes, llaves, paréntesis se encuentran balanceados.

Acceso al sistema.

El ingreso a la aplicación se realizará desde un entorno de desarrollo integrado (Dev c++).

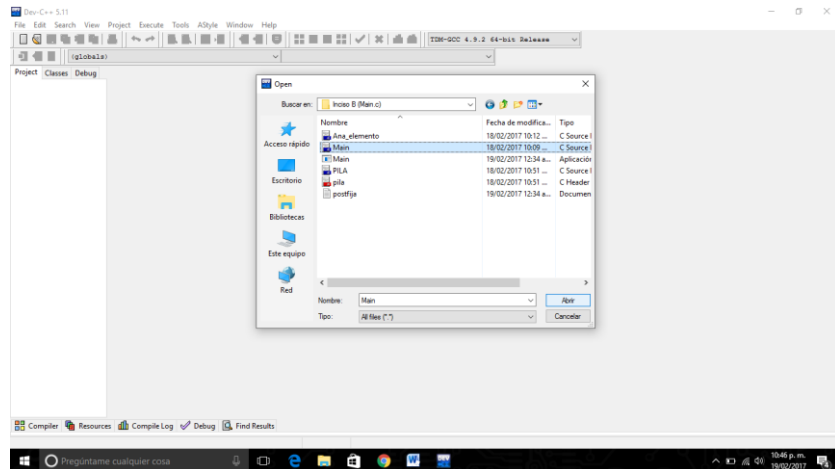


Acceso al IDE.

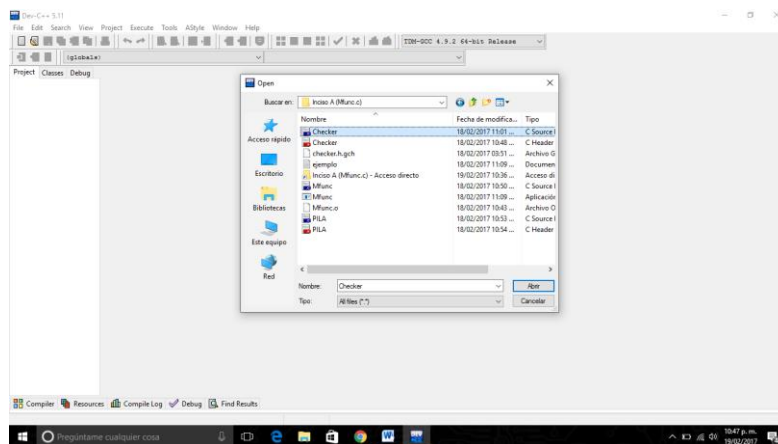
Una vez ingresado en el IDE abriremos dos archivos, para que nuestra aplicación pueda ejecutarse, por lo cual nos iremos a la esquina superior izquierda y daremos las siguientes instrucciones.

File -> Open

1. Abrir: Inciso B (Main.c)/Main.c
2. Abrir: Inciso A (Mfunc.c)/Mfunc.c

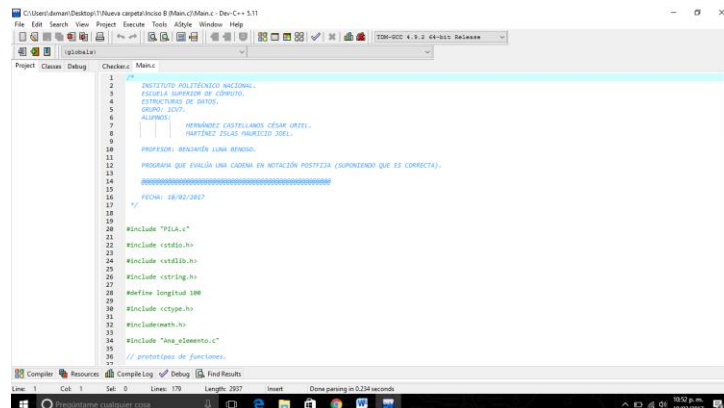


Apertura del Main.c

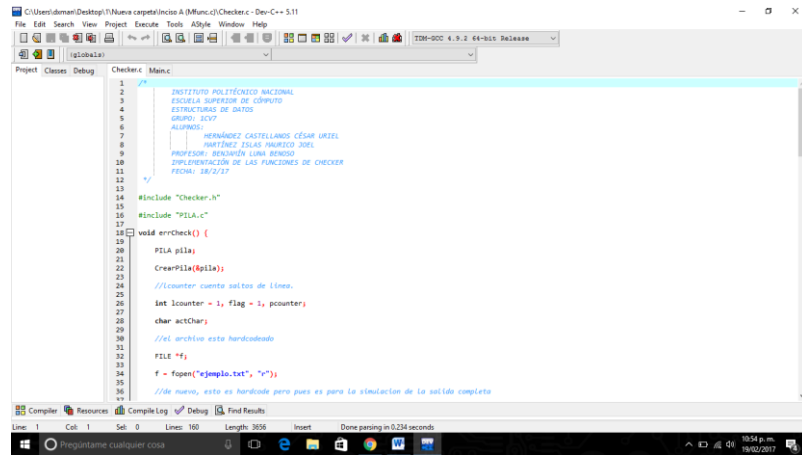


Apertura del Mfunc.c

Una vez abiertos los archivos, observaremos parte del código correspondiente a la práctica, con su respectiva cabecera.

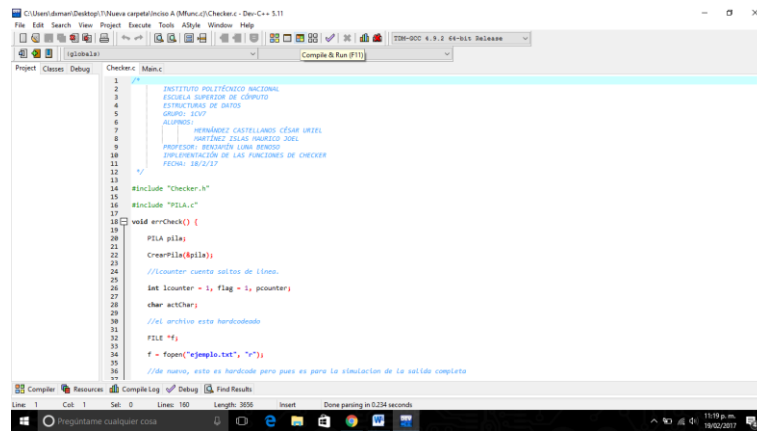


Código de Main.c

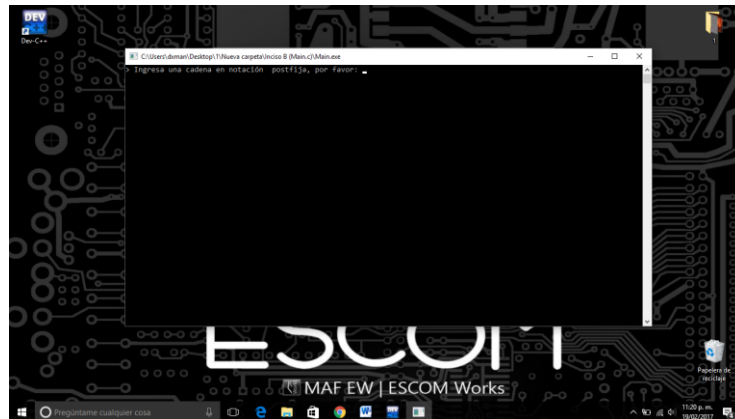


Código del Mfunc.c

En la parte central del entorno de desarrollo integrado observaremos diversos botones, el que nos interesa en esta ocasión es el de (Compile & Run) o bien F11, para ejecutar nuestra aplicación.

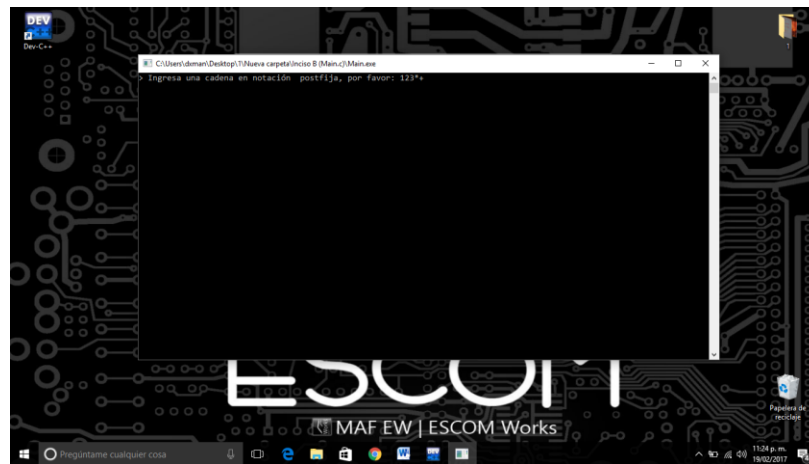


Compilar y ejecutar un programa.



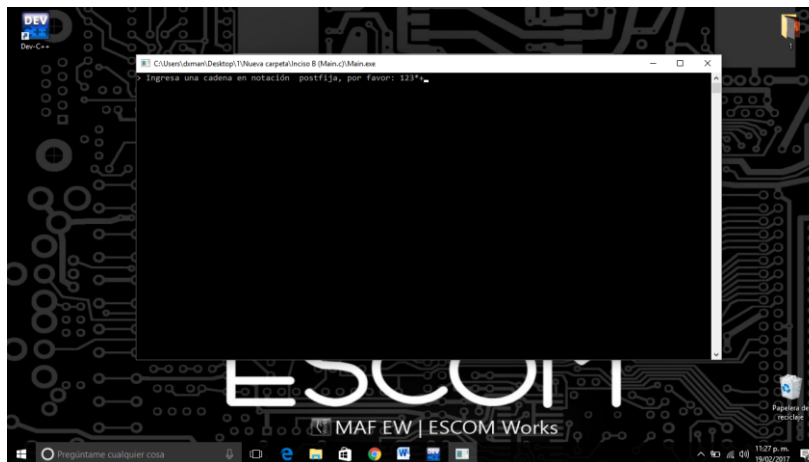
Ejecución del programa.

Nos percatamos que la consola nos solicita una expresión en notación postfija la cual ya supone como cierta.



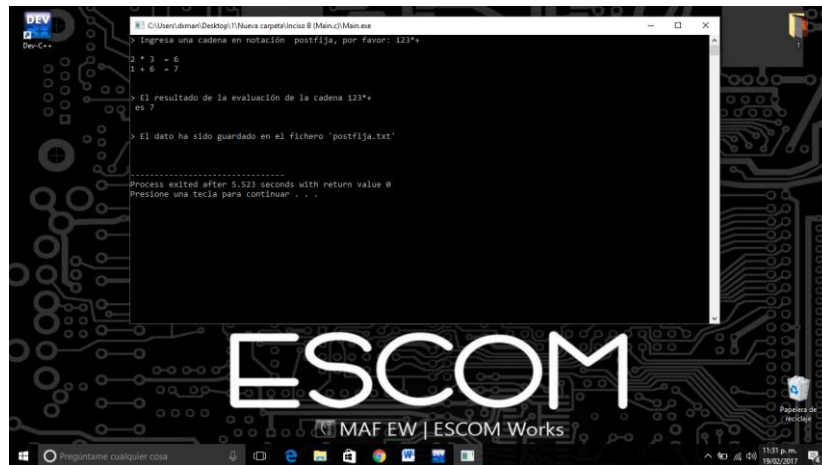
Petición de una expresión en notación postfija.

En el primer caso probaremos con la siguiente expresión $(123*+)$. lo cual tiene que brindarnos un resultado de 7.

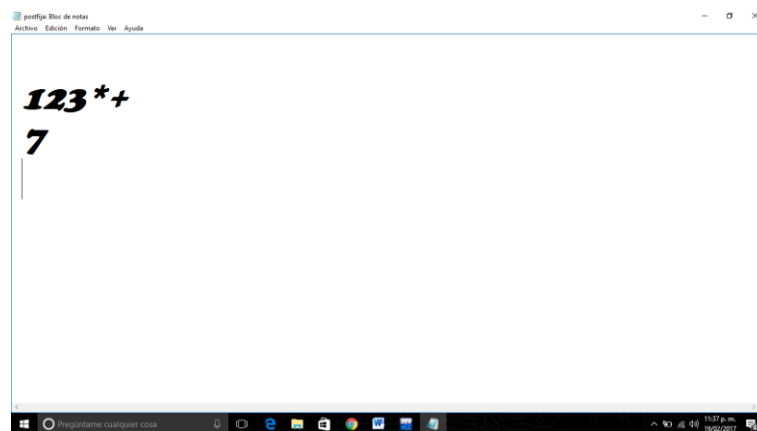


Ingreso de una expresión en notación postfija.

Ingresaremos la expresión sin espacio alguno y con la correcta sintaxis de una expresión en notación postfija, en caso que el usuario ingrese una expresión incorrecta este simplemente terminará su ejecución.



Podemos corroborar que el resultado deseado ha sido reproducido por la aplicación, además de que el resultado ha sido guardado en el fichero “postfija.txt”

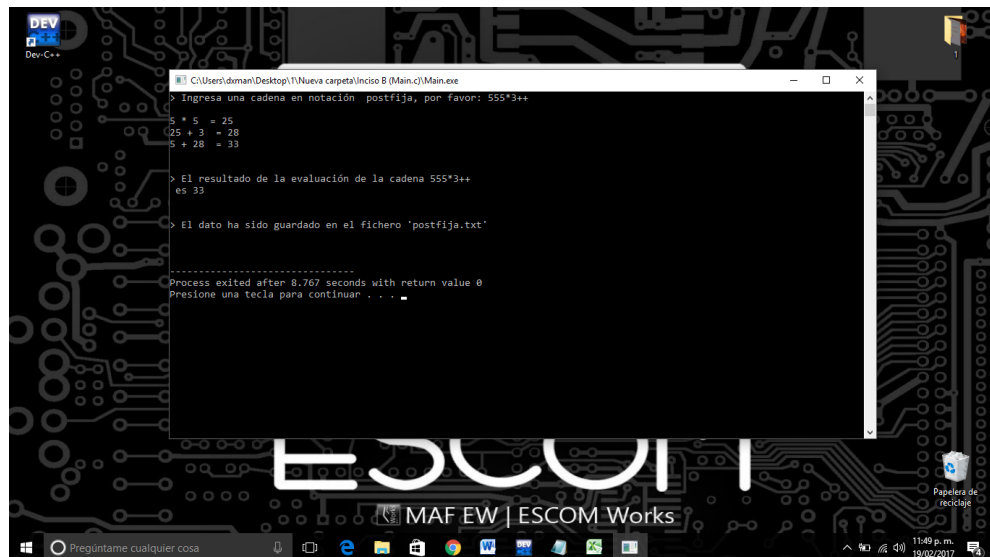


Fichero “postfija.txt”

Comprobación de 123*+

EXPRESIÓN	OPERANDO 1	OPERANDO	OPERANDO 2	PILA
1				1
2				12
3				123
*	2	*	3	16
+	1	+	6	7

En segunda instancia ingresaremos $555*3++$



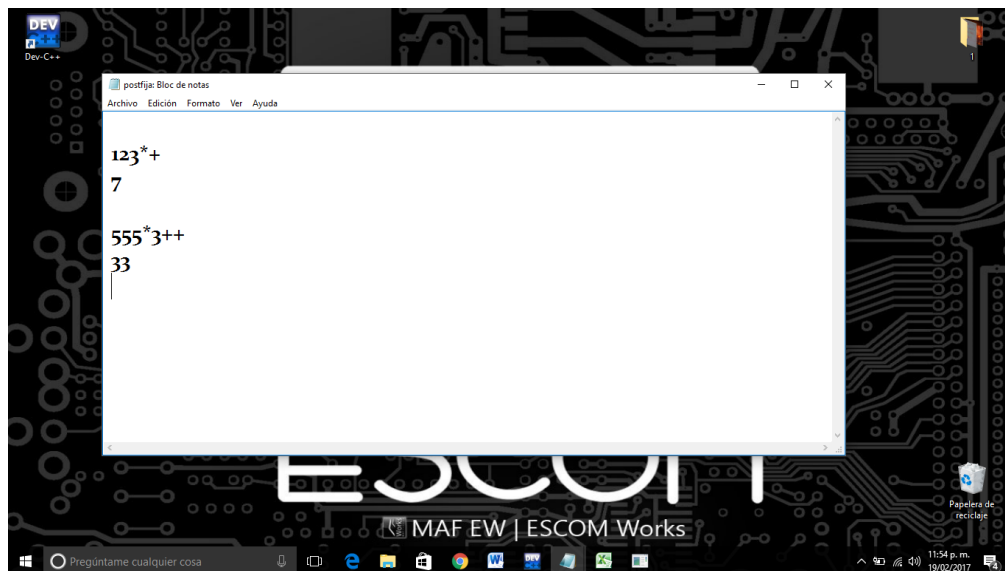
```
C:\Users\doman\Desktop\1\Nueva carpeta\Inicio 8 (Main.c)\Main.exe
> Ingresa una cadena en notación postfija, por favor: 555*3++
5 * 5 = 25
25 + 3 = 28
5 + 28 = 33

> El resultado de la evaluación de la cadena 555*3++
es 33

> El dato ha sido guardado en el fichero 'postfija.txt'

-----
Process exited after 8.767 seconds with return value 0
Presione una tecla para continuar . . .
```

El resultado arrojado por la aplicación es 33, abriremos el fichero “postfija.txt” para verificar que nuestro resultado ha sido almacenado.



Fichero “postfija.txt”

Procederemos a verificar el resultado obtenido por la aplicación, mediante el algoritmo visto en clase.

Comprobación de $555*3++$

EXPRESIÓN	OPERANDO 1	OPERANDO	OPERANDO 2	PILA
5				5
5				55
5				555
*	5	*	5	5[25]
3				5[25]3
+	25	+	3	5[28]
+	28	+	5	33

Nos situamos en el archivo Mfunc.c, para comprobar su funcionamiento.

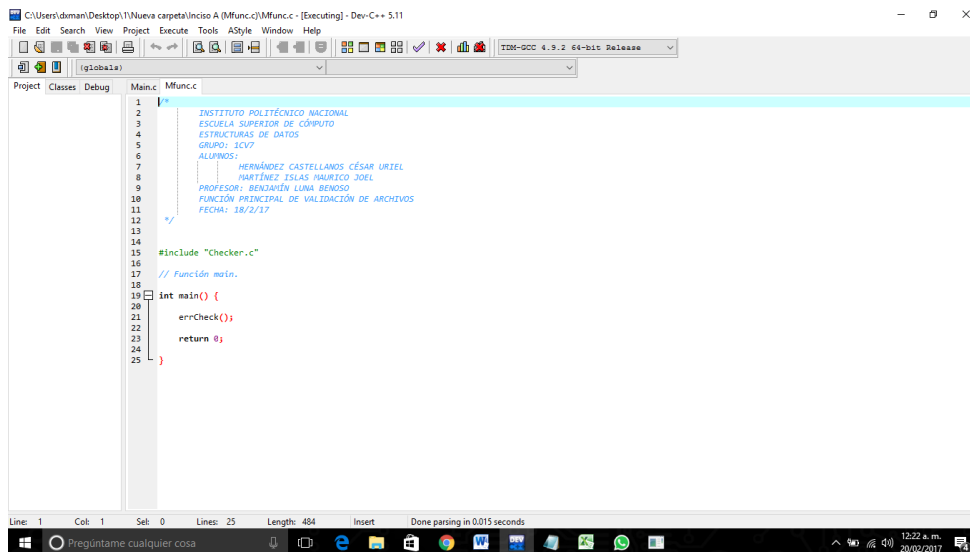
```

1  /*
2  INSTITUTO POLITÉCNICO NACIONAL
3  ESCUELA SUPERIOR DE CÓMPUTO
4  ESTRUCTURAS DE DATOS
5  GRUPO: 1CV7
6  ALUMNOS:
7  HERNÁNDEZ CASTELLANOS CÉSAR URIEL
8  MARTÍNEZ ISLAS MAURICIO JOEL
9  PROFESOR: BENJAMÍN LUNA BENOSO
10 FUNCIÓN PRINCIPAL DE VALIDACIÓN DE ARCHIVOS
11 FECHA: 18/2/17
12 */
13
14
15 #include "Checker.c"
16
17 // Función main.
18
19 int main()
20 {
21     errCheck();
22     return 0;
23 }

```

Archivo Mfunc.c

En la parte central del entorno de desarrollo integrado observaremos diversos botones, el que nos interesa en esta ocasión es el de (Compile & Run) o bien F11, para ejecutar nuestra aplicación.



Archivo Mfunc.c

Procederemos a probar las expresiones erróneas del fichero “ejemplo.txt”

```
(({{{}})000)){{{  
[asdfghj]
```

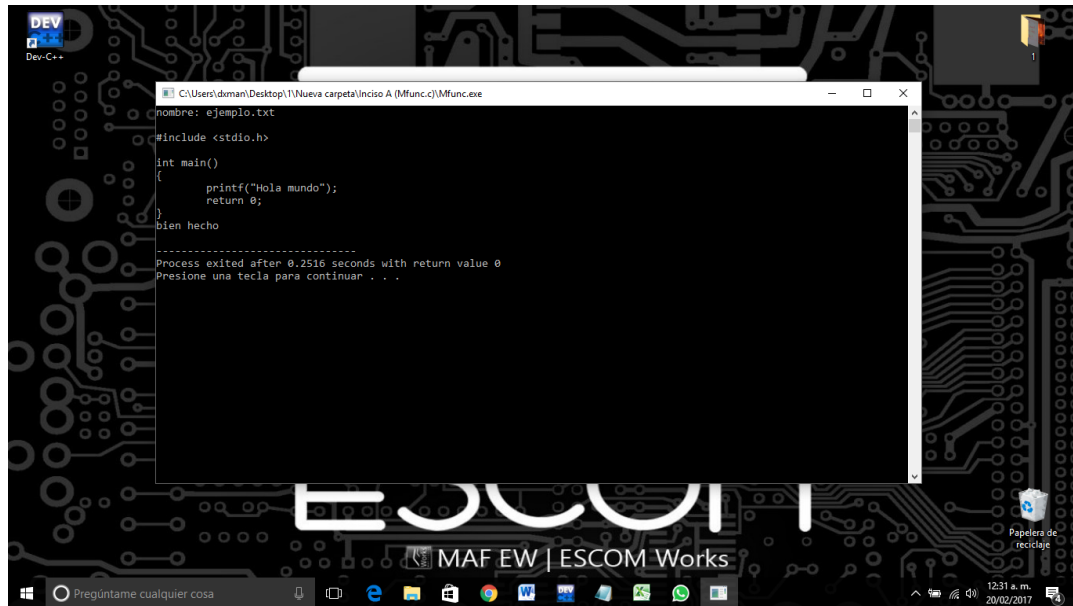
{ aqui diria que falta una llave aqui, por que es la linea del ultimo simbolo en la pila
}}}



Ejecución de Mfunc.c

Podemos observar que cuatro llaves se abren en el principio del archivo, sin embargo en la línea siete haría falta una llevar, por lo que la aplicación marca un error en dicha línea.

Ahora ingresaremos al archivo un código que se encuentre bien elaborado.



La aplicación ha determinado que el código no contiene ningún error, por lo que le despliega el mensaje “bien hecho”.

```
#include <stdio.h>
```

```
int main()
{
    printf("Hola mundo");
    return 0;
}
```

Pseudocódigo

Validador de expresiones.

```
Inicio principal
pila
crear pila
contLineas, band, contP,
actChar
abrir archivo f
Mientras !fin de archivo y bandera ==1
    actChar = siguiente caracter de archivo
    si actChar=='\n' entonces
        contLineas++
    FinSi
    si actChar=='(' o actChar=='[' o actChar=='{' o actChar=='<' entonces
        insertar actChar en pila
        contLineas = contP
    FinSi
    si actChar==')' o actChar==']' o actChar=='}' o actChar=='>' entonces
        si !pilavacia entonces
            //valida es una funcion que imprime el error si los caracteres no
            corresponden, regresa entero
            si valida(cimapila, actChar) entonces
                imprimir "posible linea de error: " + contLineas
                band=0
            Sino
                quitarpila
            FinSi
        Sino
            imprimir "posibe linea de error: " + contLineas
            imprimir "posible error: sobra un " + actChar
            band=0
        FinSi
    FinSi
FinMientras
si !pilavacia y band ==1 entonces
    Imprimir "posible linea de error: " + contP
    valida(cimapila, actChar)
sino
    imprimir "no hay errores"
FinSi
Fin principal
```

Entero validate (carácter l, carácter r)

Inicio función

Entero flag -> 0

Si (l es igual a () entonces

Flag -> (r -> ') ? 1:0

Si no -> Si (l es igual a '[') entonces

Flag -> (r ->]) ? 1:0

Si no -> Si (l es igual a '{ ') entonces

Flag -> (r -> '}') ? 1:0

Si no -> Si (l es igual a <)

Flag = (r -> '>') ? 1:0

Fin Si

Si (flag es igual a cero)

Si (l es igual a '(')

Imprimir Posible error falta un ')'

Si no -> Si (l es igual a '[')

Imprimir Posible error falta un ']'

Si no -> Si (l es igual a '{')

Imprimir Posible error falta un '}'

Si no -> Si (l es igual a <)

Imprimir Posible error falta un '>'

Fin Si

Regresar flag

Fin función
Fin validate

Evaluación de una expresión en notación postfija

Inicio principal

pil
cadena
o
op1, op2, final->0
i-> 0, count-> 0, resultado
crear pila (pil)
cadena -> regresar(cadena)

Inicio para i->0 i< longitud de cadena i -> i+1

Sí (cadena en i) es un dígito, entonces

Insertarenpila(pil, convertir_caracter_a_entero(cadena en i))

Sino

o-> cadena en i
op2 -> quitarpila(pil)
op1-> quitarpila(pil)
final -> validar (o, op1, op2)
imprimir op1,o, op2, final
insertarpila(pil, final)

FinSi

Fin para

Imprimir El resultado de la evaluación de la cadena es + final

Ingresar_a_txt(cadena, final)

Imprimir El dato ha sido guardado en el fichero "postfija.txt"

Fin principal

vacio ingresar_a_txt (carácter cadena, entero resul)

Inicio función

archivo doc
cadena salto
salto -> \n


```
doc -> abrir el fichero postfija.txt en modo a+
imprimir_en_archivo(salto, doc)
imprimir_en_archivo(cadena, doc)
imprimir_en_archivo(resul, doc)
imprimir_en_archivo(salto, doc)
cerrar archivo
```

```
Fin función
Fin ingresar_a_txt
```

```
Entero validar(carácter o, entero op2, entero op2)
```

```
Inicio función
Entero resultado->0
Según o hacer
Caso +
    Resultado -> op1 + op2
Fin caso +
Caso -
    Resultado -> op1-op2
Fin caso -
Caso *
    Resultado -> op1 * op2
Fin caso *
Caso /
    Resultado -> op1/op2
Fin caso /
Caso ^
    Resultado -> op1 ^ op2
Fin caso ^
Fin según
Regresar resultado
Fin función
Fin validar
```

```
Entero convertir_a_caracter_a_entero(carácter caracter)
```

```
Inicio función
Carácter cadena
Entero contador -> 0
Entero i
Cadena -> Asignar una longitud de 100
Si (cadena es nula)
    Salir
FinSi
Imprimir Ingresa una cadena en notación postfija, porfavor.
Cadena -> leer cadena
Regresar cadena
```

Fin función
Fin convertir_a_caracter_a_entero

Conclusiones.

Hernández Castellanos César Uriel:

Las pilas son uno de los conceptos más fundamentales y útiles en la computación, debido a que nos ayudan a simplificar ciertas operaciones de programación, además de poseer un amplio campo de aplicación.

En la práctica vimos algunas de las aplicaciones que tiene la pila en nuestro compilador, como lo es el verificador de sintaxis, que nos permitió saber el como un compilador trabaja de manera interna o la evaluación de expresiones matemáticas, todo esto mediante pilas.

Martínez Islas Mauricio Joel:

El algoritmo de validación de símbolos para texto no quedó muy bien adaptado al archivo y requiere de mejoras. La manera en la que lanza los errores se ve muy sucio al igual que la función de validar, lamentablemente no se pudo trabajar más ésta parte y se tuvo que dejar así, pero se tiene que mejorar.

Anexo.

1. Mediante el algoritmo visto en clase de validación de símbolos (), [], <>, {}, mostrar si las siguientes cadenas con válidas o no.

i) ([{((())])

EXPRESIÓN	SÍMBOLO 1	SÍMBOLO 2	PILA
((
[([
]	[]	(
{			{{
({{(
({{((
)	()	{{(
({{(((
)	()	{{((
)	()	{{(
]	{]	*

No corresponden los símbolos

ii) $\{((())[])\}$

EXPRESIÓN	SÍMBOLO 1	SÍMBOLO 2	PILA
{			{
({{
({{{
)	()	{{
({{{
)	()	{{
)	()	{
[{[
[{[[
]	[]	{[
]	[]	{
}	{	}	

Salió todo bien.

iii) $([])\}$

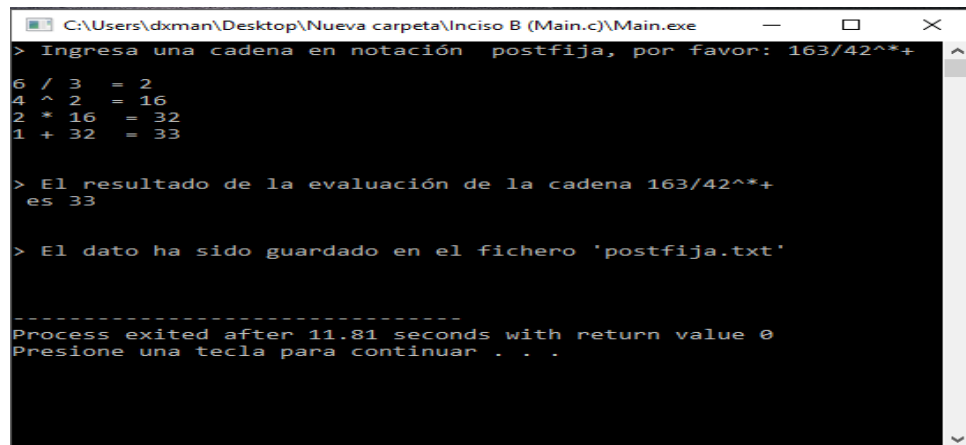
EXPRESIÓN	SÍMBOLO 1	SÍMBOLO 2	PILA
((
[([
]	[]	(
)	()	
}		}	*

El programa lanza error por símbolo extra.

2. Mediante el algoritmo visto en clase evaluar una cadena en notación postfija, mostrar el resultado de $163/42^{*}+$

EXPRESIÓN	OPERANDO 1	OPERANDO	OPERANDO 2	PILA
1				1
6				16
3				163
/	6	/	3	12
4				124
2				1242
^	4	^	2	12[16]
*	2	*	16	1[32]
+	1	+	32	33

Ejecución en el programa.



```
C:\Users\dxman\Desktop\Nueva carpeta\Inciso B (Main.c)\Main.exe
> Ingresa una cadena en notación postfija, por favor: 163/42^*+
6 / 3 = 2
4 ^ 2 = 16
2 * 16 = 32
1 + 32 = 33

> El resultado de la evaluación de la cadena 163/42^*+
es 33

> El dato ha sido guardado en el fichero 'postfija.txt'

-----
Process exited after 11.81 seconds with return value 0
Presione una tecla para continuar . . .
```

Bibliografía.

- [1]L. Nyhoff, *TADs, estructuras de datos y resolución de problemas con C++*, 1st ed. Madrid: Pearson/Prentice Hall, 2006.

- [2]Y. Langsam, M. Augenstein and A. Tenenbaum, *Estructuras de datos con C y C++*, 1st ed. Pearson Prentice Hall, 2000.

- [3]"Programación. Tema 4: Pilas y Colas", *Eduardo Quevedo, Raquel López y Aaron Asencio*, 2017. [Online]. Available: chrome-extension://oemmndcbldboiebfnladdacbfmadadm/http://www.iuma.ulpgc.es/users/jmiranda/docencia/programacion/Tema4_ne.pdf. [Accessed: 20- Feb- 2017].

- [4]"Friedrich L. Bauer • IEEE Computer Society", *Computer.org*, 2017. [Online]. Available: <https://www.computer.org/web/awards/pioneer-friedrich-bauer>. [Accessed: 20- Feb- 2017].

- [5]"About Awards • IEEE Computer Society", *Computer.org*, 2017. [Online]. Available: <https://www.computer.org/web/awards/about-awards;jsessionid=a8cd31dfc305d79167577a08122b>. [Accessed: 20- Feb- 2017].

- [6]"Neptalium » Notación Polaca Inversa", *Neptalium.com*, 2017. [Online]. Available: <http://www.neptalium.com/blog/tag/notacion-polaca-inversa/>. [Accessed: 20- Feb- 2017].