

UNIVERSIDAD DE GRANADA

INGENIERÍA INFORMÁTICA

Entrega 2 - Calculadora en LEX

Autor: JUAN CARLOS RUIZ GARCÍA

Asignatura: Modelos de Computación

11 de diciembre de 2017



Índice

1. Problema planteado	2
1.1. Estructura de datos utilizada	2
1.1.1. variables & variablesU	2
1.1.2. valores & simbolos	2
1.2. Funcionamiento	3
1.3. Restricciones de uso	3
2. Compilación y modos de ejecución	4
2.1. Compilación	4
2.2. Modos de ejecución	5
2.2.1. Fichero de texto	5
2.2.2. Entrada manual	6

1. Problema planteado

El problema planteado, es el desarrollo de una **calculadora utilizando LEX** y apoyandonos en **estructuras de datos** desarrolladas en **C++**.

La calculadora podrá realizar operaciones con **enteros y reales**. Además únicamente permitirá llevar a cabo las operaciones de **suma, resta, multiplicación y división**.

1.1. Estructura de datos utilizada

Para el desarrollo de la calculadora, hemos utilizado la siguiente estructura de datos:

```
vector<double> variables;  
vector<bool> variablesU;  
list<double> valores;  
list<pair<char,int>> simbolos;
```

1.1.1. variables & variablesU

En la calculadora podremos definir hasta **26 variables** para realizar operaciones con ellas, siendo estas de la **'a-z'** en **minúscula**.

- **variables** : vector de doubles utilizado para almacenar los valores de las 26 posibles variables.
- **variablesU** : vector de boolean inicialmente los 26 a false, menos las variables con valor asignado que estarán a true.

1.1.2. valores & simbolos

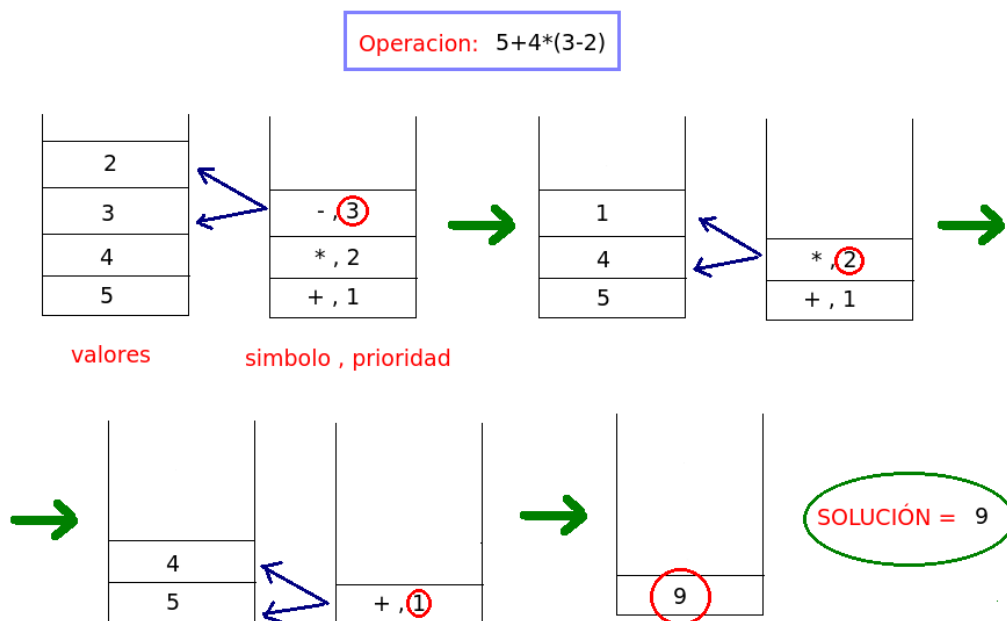
Son dos listas enlazadas utilizadas para:

- **valores** : contiene los números que forman la operación a realizar.
- **simbolos** : contiene los operadores que se van a aplicar a los números del vector valores junto con su prioridad. Dicha prioridad se calcula:
 - Con parentesis (a):
 - '+' y '-' : como **numParentesis*3**
 - '*' y '/' : como **numParentesis*4**
(a) numParentesis inicialmente vale 0. Se incrementa en 1 cuando se encuentra '(' y se decrementa en 1 cuando se encuentra ')'
 - Sin parentesis (b):
 - '+' y '-' : como **prioSumRes**
 - '*' y '/' : como **prioMulDiv**
(b) prioSumRes = 1; prioMulDiv = 2;

1.2. Funcionamiento

En la **lista de valores** se van almacenando los números que se van encontrando en la operación de izquierda a derecha, al igual que en la **lista de símbolos** se van guardando los operadores encontrados junto con su correspondiente prioridad, calculada como se explicó anteriormente.

Un ejemplo de funcionamiento sería el siguiente:



1.3. Restricciones de uso

He de decir, que existen muchísimos casos problemáticos, los cuales habría que controlar para el correcto funcionamiento de la calculadora.

Por mi parte he intentado crear las suficientes reglas que respalden a todos estos casos, pero es posible que no los haya abarcado todos, por lo que no me hago responsable del resultado de las operaciones o de la ejecución del programa, si no se cumplen **TODAS Y CADA UNA** de las restricciones siguientes:

1. Para realizar cualquier operación la sintaxis debe ser **[OPERACION]** ?.

Ejemplos:

- a) MAL: $(-5)*(3+4)$ BIEN: $(-5)*(3+4)$?
- b) MAL: (-55) BIEN: (-55) ?
- c) MAL: $a=a*b$ BIEN: $a=a*b$?

2. Los **numeros negativos** deben ir **siempre entre parentesis**.

Ejemplos:

- a) MAL: $-5*3$? BIEN: $(-5)*3$?
- b) MAL: -10 ? BIEN: (-10) ?
- c) MAL: $a=-5*3$? BIEN: $a=(-5)*3$?

3. No debe existir **ningun espacio** en las operaciones.

Ejemplos:

- a) MAL: $(-5) * 3$? BIEN: $(-5)*3$?
- b) MAL: $a = (-5) * 3$? BIEN: $a=(-5)*3$?

4. Escribir **cualquier cosa** que no sea una operacion, una variable o una asignacion a una variable puede generar un **resultado inesperado**.
5. Puede utilizar un **número ilimitado de parentesis**, siempre y cuando no falte ninguno y respete las restricciones anteriores.
6. No debe **dividir por cero**.

Nota: Es posible que algunas operaciones complejas con muchos parentesis y operadores devuelvan valores erroneos. No he sido capaz de solventar este problema, ya que ocurre solo en casos contados.

2. Compilación y modos de ejecución

2.1. Compilación

Para compilar unicamente debemos ir a la carpeta donde se encuentra el fichero **calculadora.l**, abrir un terminal y ejecutar la orden **make**. Esto generará un ejecutable llamado **calculadora**.

Para eliminar el ejecutable y los restos creados durante la compilación debemos ejecutar la orden **make clean** en un terminal.

```
juanka1995@juanka1995-Desktop ~/GitHub/practicasmc/trabajoLex $ make
rm -f calculadora
rm -f lex.yy.c
lex calculadora.l
g++ -std=c++11 lex.yy.c -o calculadora -ll
juanka1995@juanka1995-Desktop ~/GitHub/practicasmc/trabajoLex $ make clean
rm -f calculadora
rm -f lex.yy.c
juanka1995@juanka1995-Desktop ~/GitHub/practicasmc/trabajoLex $
```

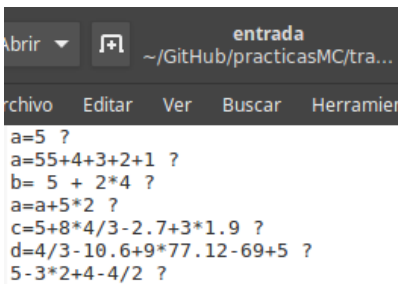
2.2. Modos de ejecución

Tenemos dos formas de utilizar la calculadora, a través de **un fichero de texto** o de **la entrada manual de operaciones**.

2.2.1. Fichero de texto

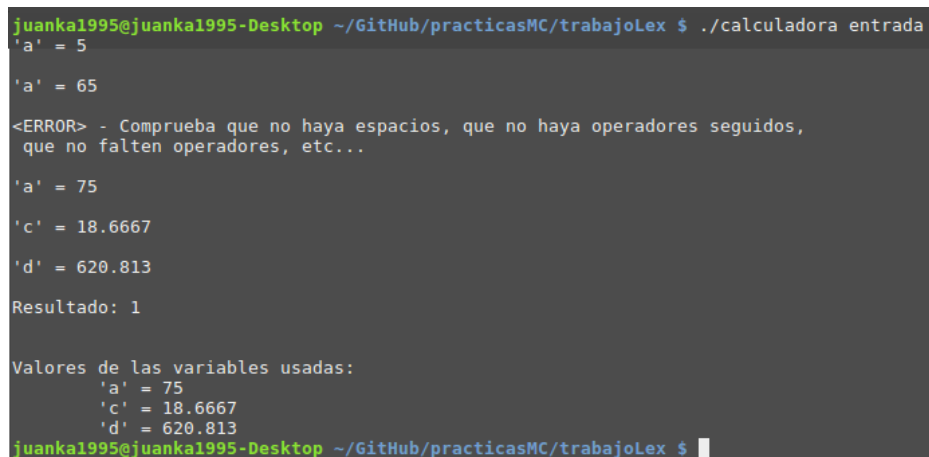
Debemos de crear un fichero con las operaciones que deseamos que nuestra calculadora ejecute. Podemos basarnos en el fichero **entrada** que adjunto en el interior del comprimido.

Seguidamente lanzamos la calculadora pasandole como parametro la ruta de dicho fichero de entrada.



The screenshot shows a text editor window titled 'entrada' with the path '~/.GitHub/practicasMC/tra...'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Buscar', and 'Herramientas'. The content of the file is as follows:

```
a=5 ?  
a=55+4+3+2+1 ?  
b= 5 + 2*4 ?  
a=a+5*2 ?  
c=5+8*4/3-2.7+3*1.9 ?  
d=4/3-10.6+9*77.12-69+5 ?  
5-3*2+4-4/2 ?
```



The screenshot shows a terminal window with the following output:

```
juanka1995@juanka1995-Desktop ~/GitHub/practicasMC/trabajoLex $ ./calculadora entrada  
'a' = 5  
'a' = 65  
<ERROR> - Comprueba que no haya espacios, que no haya operadores seguidos,  
que no falten operadores, etc...  
'a' = 75  
'c' = 18.6667  
'd' = 620.813  
Resultado: 1  
  
Valores de las variables usadas:  
'a' = 75  
'c' = 18.6667  
'd' = 620.813  
juanka1995@juanka1995-Desktop ~/GitHub/practicasMC/trabajoLex $
```

2.2.2. Entrada manual

Simplemente deberemos ejecutar el programa y escribir las operaciones por teclado. Cuando terminemos deberemos pulsar la combinación **CTRL + D** para salir.

```
juanka1995@juanka1995-Desktop ~/GitHub/practicasmc/trabajoLex $ ./calculadora
```

```
a=5 ?  
'a' = 5  
  
a*3 ?  
Resultado: 15  
  
b=a+4*2 ?  
'b' = 13  
  
(-5.2)*3-(4+6*9+2.5/0.5) ?  
Resultado: -78  
  
5.5-5.2 ?  
Resultado: 0.3  
  
c=(5*(4+3*(10/2+3))) ?  
'c' = 140
```