## *Consignas*

**Ejercicio 1**

Dada una cadena que contenga varios números que pueden ser decimales, octales o hexadecimales, con o sin signo para el caso de los decimales, separados por el carácter '&', reconocer los tres grupos de constantes enteras, indicando sí hubo un error léxico , en caso de ser correcto contar la cantidad de cada grupo.

**Ejercicio 2**

Utilizando el ejercicio 1 ingresar una cadena que represente una operación simple con enteros decimales y obtener su resultado, se debe operar con +, - y *. Ejemplo = 3+4*7+3-5 = 29.

## *Metodologia utilizada*

Para resolver estos ejercicios, se implementaron dos autómatas finitos determinísticos (AFD) con activadores en lenguaje de programación C.

Al ser un único programa el qué resuelve ambos ejercicios, se espera qué se le pase por parámetro al momento de ejecutarlo (1) qué ejercicio se quiere resolver y (2) la cadena a analizar.
Para resolver el ejercicio 1, se debe invocar al programa pasando por parámetros el número 1 y luego la cadena a analizar.

```
$ ./tp 1 "102&78&013&0x95FD"
```

Para resolver el ejercicio 2, se debe invocar al programa pasando por parámetros el número 2 y luego la cadena a analizar.

```
$ ./tp 2 "98+106-83*50+782"
```

En caso de pasar los parámetros de manera incorrecta, el programa devuelve un mensaje de error indicando la forma en qué se debe hacerlo.

Con respecto al desarrollo del ejercicio 1, estos son los pasos qué se siguieron:

1. Se estableció qué los números decimales deben empezar con el signo +, - o cualquier número del 1 al 9, seguido de cualquier combinación de números del 0 al 9.
2. Se estableció qué los números octales deben empezar con 0, seguido de cualquier combinación de números del 0 al 7.
3. Se estableció qué los números hexadecimales deben empezar con 0x, seguido de cualquier combinación de números del 0 al 9 o de letras del A al F.
4. Se estableció qué para qué la cadena a analizar sea reconocida cómo válida, debe contener cualquiera de los tres tipos de números descritos anteriormente, y todos deben estar separados por el carácter &.
5. Se construyó el autómata finito determinístico qué reconoce este tipo de cadenas.
6. Se escribió el programa en lenguaje C.

7. Se hicieron las pruebas correspondientes.

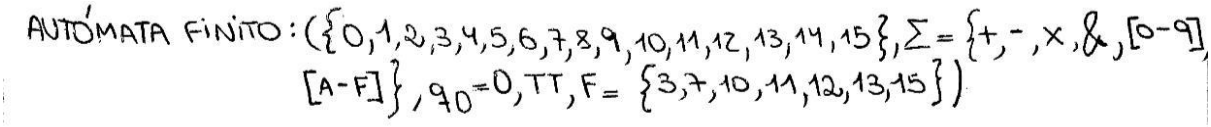
Con respecto al desarrollo del ejercicio 2, estos son los pasos qué se siguieron:

1. Se estableció qué los números válidos dentro de la cadena son los decimales sin signo.
2. Se estableció qué las operaciones válidas dentro de la cadena son la suma, la resta y la multiplicación.
3. Se construyó el autómata finito determinístico qué reconoce este tipo de cadenas.
4. Se escribió el programa en lenguaje C.
5. Se hicieron las pruebas correspondientes.

## *Autómatas utilizados*

Cómo se ha dicho antes, se utilizaron dos autómatas, uno para cada ejercicio.

**Autómata ejercicio 1**




TT

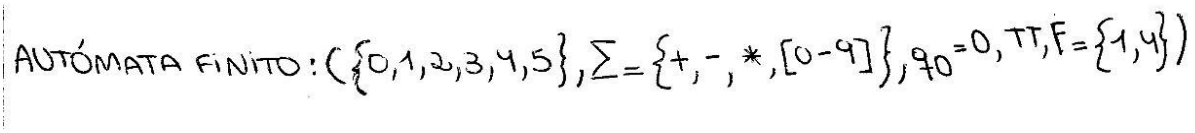
| Estado | Signo | 0 | {1,7} | {8,9} | {A,F} | x | & | Otro | fdt |
|---|---|---|---|---|---|---|---|---|---|
| 0 - | 1 | 5 | 2 | 2 | 4 | 4 | 0 | 4 | 15 |
| 1 | 4 | 2 | 2 | 2 | 4 | 4 | 0 | 4 | 14 |
| 2 | 4 | 2 | 2 | 2 | 4 | 4 | 3 | 4 | 11 |
| 3 + | 1 | 5 | 2 | 2 | 4 | 4 | 0 | 4 | 15 |
| 4 (rechazo) | 4 | 4 | 4 | 4 | 4 | 4 | 0 | 4 | 14 |
| 5 | 4 | 6 | 6 | 4 | 4 | 8 | 0 | 4 | 14 |
| 6 | 4 | 6 | 6 | 4 | 4 | 4 | 7 | 4 | 12 |
| 7 + | 1 | 5 | 2 | 2 | 4 | 4 | 0 | 4 | 15 |
| 8 | 4 | 9 | 9 | 9 | 9 | 4 | 0 | 4 | 14 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **9** | 4 | 9 | 9 | 9 | 9 | 4 | 10 | 4 | 13 |
| **10 +** | 1 | 5 | 2 | 2 | 4 | 4 | 0 | 4 | 15 |
| **11 +** | - | - | - | - | - | - | - | - | - |
| **12 +** | - | - | - | - | - | - | - | - | - |
| **13 +** | - | - | - | - | - | - | - | - | - |
| **14 (fdt)** | - | - | - | - | - | - | - | - | - |
| **15 + (fdt)** | - | - | - | - | - | - | - | - | - |

- El estado 0 es el estado inicial.
- Los estados 3, 7 y 10 son estados de aceptación, al igual qué los estados 11, 12, 13, y 15, con la diferencia qué en los primeros se puede seguir procesando caracteres, y en los segundos no.
- Los estados 11, 12, 13, 14 y 15 son estados especiales, ya que son útiles para la implementación del autómata por medio del lenguaje C.
- Los estados 0, 3, 7 y 10 son iguales ya qué en todos se comienza el analisis de un nuevo número.
- El estado 4 es el estado de rechazo.
- Los estados 1, 2 y 3 sirven para reconocer un número decimal.
- Los estados 5, 6 y 7 sirven para reconocer un número octal.
- Los estados 5, 8, 9 y 10 sirven para reconocer un número hexadecimal.
- El estado 14 es un estado de “rechazo” qué indica qué se terminó la cadena y el último número leído tiene un error léxico.
- El carácter + o - sólo nos puede llevar a dos estados, el 1 en caso de ser el comienzo de un nuevo número, o el 4 (estado de rechazo), sí el signo se lee en cualquier otro momento qué no sea el comienzo de un nuevo número.
- El carácter x siempre nos lleva al estado de rechazo 4, salvo qué sea leído luego de un 0 y al comienzo de un nuevo número.
- Cualquier otro carácter fuera del alfabeto utilizado nos lleva al estado de rechazo 4.

Los estados 11, 12, 13, 14 y 15 son estados qué no tienen transiciones a otros estados porque se utilizan únicamente en el algoritmo computacional, ya qué se intentará seguir procesando caracteres de la cadena mientras el estado no sea alguno de los nombrados.

**Autómata ejercicio 2**




TT

| Estado | + | - | * | {0,9} | Otro | fdt |
|---|---|---|---|---|---|---|
| 0 - | 3 | 3 | 3 | 1 | 3 | 4 |
| 1 + | 2 | 2 | 2 | 1 | 3 | 4 |
| 2 | 3 | 3 | 3 | 1 | 3 | 5 |
| 3 (rechazo) | 3 | 3 | 3 | 3 | 3 | 5 |
| 4 + (fdt) | - | - | - | - | - | - |
| 5 (fdt) | - | - | - | - | - | - |

- El estado 0 es el estado inicial.
- El estado 1 es estado de aceptación, al igual qué el estado 4, con la diferencia qué en el primero se puede seguir procesando caracteres, y en el segundo no.
- Los estados 4 y 5 son estados especiales, ya que son útiles para la implementación del autómata por medio del lenguaje C.
- El estado 3 es el estado de rechazo.
- El estado 1 sirve para reconocer un número decimal sin signo.
- El estado 2 sirve para reconocer el signo +, - o *, qué deben estar siempre luego de un número.
- El estado 5 es un estado de “rechazo” qué indica qué se terminó la cadena y el último carácter leído es un signo +, - o *.
- Cualquier otro carácter fuera del alfabeto utilizado nos lleva al estado de rechazo 3.

Los estados 4 y 5 son estados qué no tienen transiciones a otros estados porque se utilizan únicamente en el algoritmo computacional, ya qué se intentará seguir procesando caracteres de la cadena mientras el estado no sea alguno de los nombrados.

## *Algoritmo computacional utilizado*

### Algoritmo ejercicio 1

El funcionamiento del algoritmo es el siguiente: se analiza una cadena por medio del autómata creado. En caso de qué en algún momento se detecte un error léxico, el programa

termina e imprime dicho mensaje por pantalla. Sí en ningún momento se detecta un error léxico, se imprime dicho mensaje por pantalla junto con la cantidad de números de cada grupo qué se detectaron.

Para este ejercicio se utilizó un algoritmo qué hace lo siguiente:

1. Se lee el primer carácter de la cadena.
2. Se itera lo qué se va a describir a continuación mientras el estado no sea 11, 12, 13, 14 ni 15.
    a. Se actualiza el estado.
    b. Sí el estado es 3, significa qué anteriormente se leyó correctamente un número decimal, por lo tanto se incrementa el contador de números decimales.
    c. Sí el estado es 7, significa qué anteriormente se leyó correctamente un número octal, por lo tanto se incrementa el contador de números octales.
    d. Sí el estado es 10, significa qué anteriormente se leyó correctamente un número hexadecimal, por lo tanto se incrementa el contador de números hexadecimales.
    e. Sí el estado es 4, significa qué estamos en estado de rechazo y hubo un error léxico, por lo tanto, se comunica eso por pantalla y se finaliza el programa.
    f. Se lee el siguiente carácter de la cadena y se vuelve al punto a.
3. Una vez qué se termina de iterar porque ya no hay más caracteres en la cadena, dependiendo cual estado es el último hay qué hacer alguna acción.
    a. Sí el último estado es el 11 significa qué anteriormente se leyó correctamente un número decimal, por lo tanto se incrementa el contador de números decimales.
    b. Sí el último estado es el 12 significa qué anteriormente se leyó correctamente un número octal, por lo tanto se incrementa el contador de números octales.
    c. Sí el último estado es el 13 significa qué anteriormente se leyó correctamente un número hexadecimal, por lo tanto se incrementa el contador de números hexadecimales.
    d. Sí el último estado es el 14 significa que hubo un error léxico antes de finalizar la cadena, por lo tanto, se comunica eso por pantalla y se finaliza el programa.
    e. Sí el último estado es el 15, no se hace nada. Este caso es útil sí la cadena finaliza con el carácter &, ya qué no habría un error léxico, pero no se tendría qué aumentar ningún contador.
4. Por último, sí el programa todavía está en ejecución, significa qué no se encontró ningún error léxico dentro de la cadena, por lo tanto, se imprime por pantalla tal mensaje junto con las cantidades de números encontrados de cada grupo de números.

**Algoritmo ejercicio 2**

El funcionamiento del algoritmo es el siguiente: se analiza una cadena por medio del autómata creado. En caso de qué en algún momento se detecte un error léxico, el programa termina e imprime dicho mensaje por pantalla. Sí en ningún momento se detecta un error léxico, se imprime dicho mensaje por pantalla junto con el resultado de la operación.

Para este ejercicio se utilizó un algoritmo qué hace lo siguiente:

1. Se lee el primer carácter de la cadena.
2. Se itera lo qué se va a describir a continuación mientras el estado no sea 4 ni 5.
    a. Se actualiza el estado.
    b. Sí el estado es 1, significa qué se acaba de leer un número (qué puede ser parte de un número más grande), por lo tanto se transforma ese carácter en un número, se desplaza al número más grande y se suman ambos números.
    c. Sí el estado es 2, significa qué se acaba de leer un signo + o -. Al leer un signo, significa qué el número qué se leyó anteriormente está completo y se almacena en el vector correspondiente multiplicandolo por la variable proxSigno (para agregarle el signo). Además se guarda el signo + dentro del vector de operaciones (ya qué el signo real lo tiene el número) y se incrementa la variable qué cuenta la cantidad de números.
        i. Sí el signo es -, significa qué el próximo número será negativo, por lo tanto se actualiza a la variable proxSigno y se iguala a -1.
        ii. Sí el signo es +, significa qué el próximo número será positivo, por lo tanto se actualiza a la variable proxSigno y se iguala a 1.
    d. Sí el estado es 3, significa qué estamos en estado de rechazo y hubo un error léxico, por lo tanto, se comunica eso por pantalla y se finaliza el programa.
    e. Se lee el siguiente carácter de la cadena y se vuelve al punto a.
3. Una vez qué se termina de iterar porque ya no hay más caracteres en la cadena, dependiendo cual estado es el último hay qué hacer alguna acción.
    a. Sí el último estado es el 4, significa qué la cadena no tuvo errores léxicos y por último se leyó un número, por lo tanto se almacena en el vector correspondiente multiplicandolo por la variable proxSigno y se incrementa la cantidad de números.
        i. A partir de ese momento, lo qué se hace es resolver todas las multiplicaciones qué haya en la operación para respetar la precedencia de operadores. La manera de hacerlo es:
            1. Se recorre el vector de operaciones en busca de multiplicaciones.
            2. En caso de encontrar una, se multiplica el número correspondiente a esa multiplicación con el siguiente y se almacena en la posición del primero.
            3. Luego, se desplazan todos los números una posición a la izquierda, eliminando así el segundo número qué ya habíamos multiplicado.
            4. Se desplazan las operaciones una posición a la izquierda, eliminando así la multiplicación ya realizada.
            5. Se vuelve al punto a en busca de más multiplicaciones siempre y cuando todavía no se hayan revisado todas las operaciones, lo cual se verifica con el contador.
        ii. Una vez hechas todas las multiplicaciones, se suman todos los números qué quedaron en el vector, ya qué tienen su signo correspondiente, y ese resultado se imprime por pantalla.

b. Sí el último estado es el 5, significa qué hubo un error léxico al final de la cadena, por lo tanto se comunica eso por pantalla y se finaliza el programa.

## *Ejemplos*

**Ejercicio 1**

Cabe recordar qué los números octales comienzan con 0, los hexadecimales con 0x y los decimales con cualquier número distinto de 0, el signo + o el signo -.

- Cadena válida de un número por cada grupo de números.

```
$ ./tp 1 "4012&05342&0x6483ADF"
No hubo ningun error lexico y se reconocieron:
1 numeros decimales
1 numeros octales
1 numeros hexadecimales
```

- Cadena válida de dos números por cada grupo de números.

```
$ ./tp 1 "9&04&0xABC&0x98AF&192&043"
No hubo ningun error lexico y se reconocieron:
2 numeros decimales
2 numeros octales
2 numeros hexadecimales
```

- Cadena válida.

```
$ ./tp 1 "1084&781&0x98EC&064&84"
No hubo ningun error lexico y se reconocieron:
3 numeros decimales
1 numeros octales
1 numeros hexadecimales
```

- Cadena válida qué incluye decimales con signo.

```
$ ./tp 1 "-105&0x89&-08&+43&587"
No hubo ningun error lexico y se reconocieron:
4 numeros decimales
0 numeros octales
1 numeros hexadecimales
```

- Cadena inválida por ser un número octal y usar el número 9 en la cadena.

```
$ ./tp 1 "0149"
Hubo un error lexico en la cadena
```

- Cadena inválida porque no se contempla el signo para decimales ni octales.

```
$ ./tp 1 "-0x56&67"
Hubo un error lexico en la cadena
```

- Cadena inválida por poseer caracteres de hexadecimal pero no comenzar con el prefijo 0x.

```
$ ./tp 1 "4389ABC"
Hubo un error lexico en la cadena
```

- Cadena inválida por no estar el carácter de separación & entre el final del número 0x32 y el comienzo del número 0x231 (ambos hexadecimales).

```
$ ./tp 1 "0x320x231"
Hubo un error lexico en la cadena
```

- Cadena inválida por poseer caracteres qué no pertenecen al alfabeto.

```
$ ./tp 1 "0x347PL."
Hubo un error lexico en la cadena
```

- Cadena inválida por poseer caracteres qué no pertenecen al alfabeto.

```
$ ./tp 1 "hola68ComoAndas"
Hubo un error lexico en la cadena
```

**Ejercicio 2**

- Ejemplo de la consigna.

```
$ ./tp 2 "3+4*7+3-5"
Cadena sin error lexico
Resultado = 29
```

- Cadena válida.

```
$ ./tp 2 "3*3+4*2"
Cadena sin error lexico
Resultado = 17
```

- Cadena inválida por tener dos signos + juntos.

```
$ ./tp 2 "3++4-9*7"
Se detecto un error lexico en la cadena
```

- Cadena inválida por tener dos signos - juntos.

```
$ ./tp 2 "3+4--9*7"
Se detecto un error lexico en la cadena
```

- Cadena inválida por tener dos signos * juntos.

```
$ ./tp 2 "3+4-9**7"
Se detecto un error lexico en la cadena
```

- Cadena inválida por poseer caracteres qué no pertenecen al alfabeto.

```
$ ./tp 2 "3*5+0xAB"
Se detecto un error lexico en la cadena
```

## *Limitaciones y propuestas a futuro*

**Ejercicio 1**

- Los números decimales no pueden empezar con 0 (salvo qué comiencen con un signo), ya qué sí el número empieza con 0 se considera cómo número octal.

**Ejercicio 2**

- La cadena está limitada a 30 números y 29 operaciones, ya qué los vectores qué guardan ambas cosas son estáticos. Para mejorar esto, se podría trabajar con

memoria dinámica y reservar memoria en tiempo de ejecución, una vez qué se sepa la cantidad de números qué posee la cadena.
- No está incluida la división.
- Los números no pueden tener signo (ya qué no se pueden poner dos signos juntos).

## *Código fuente*

El código fuente se encuentra en la carpeta qué contiene este informe.