Sesión 03**: Analizador sintáctico de una calculadora simple con flex**

## I. OBJETIVOS

- Reconocer las operaciones de suma, resta, multiplicación y división
- Utilizar las funcionalidades básicas de flex para definir expresiones de suma, resta multiplicación y división.
- Crear una aplicación que resuelva sumas, restas, multiplicaciones y divisiones.

## II. TEMAS A TRATAR

- Flex
- Análisis léxico de operaciones

## III. ANÁLISIS DEL ALGORITMO

Para reconocer una operación en nuestra calculadora, debemos entender qué sintaxis reconoceremos. El alcance de nuestra primera calculadora serán las operaciones de:

- Suma
- Resta
- Multiplicación
- División

Nuestra primera calculadora reconocerá entonces las siguientes expresiones:

- número + número
- número – número
- número * número
- número / número

Nuestra definición de reglas deberá reconocer:

| **Token (componente léxico)** | **Lexema** | **Patrón** |
|---|---|---|
| Operador | +, -, *, / | [+-*/] |
| Dígito | 0,1,2,3,4,5,6,7,8,9 | [0-9] |
| Número | 3, 3.5, 9, 4.21 | [Dígito]+(\.[Dígito]+)? |

```
DIGIT [0-9]
NUM {DIGIT}+(\.{DIGIT}+)?
OP [*/+-]
```

Las variables que debemos guardar son dos operadores (float) y la operación (char):

- float operador1
- float operador2
- char operacion

```
char operacion;
float operador1=0,operador2=0,respuesta=0;
```

El orden en el que reconoceremos la sintaxis de nuestra calculadora será:

1. Reconocer inicialmente un número que será el operador1.
2. Luego reconoceremos la operación.
3. Y luego debemos reconocer otro número que será el operador2.

Para indicar el orden correcto, podemos usar flags de estado, como tenemos dos operadores (númeos), para saber en qué variable guardaremos el valor, vamos a tener dos flags, una para el operador1 y outra para el operador2. Las flags pueden tener los siguientes valores:

| Flag | Estado | Descripción |
|---|---|---|
| Flag_operador1 | 0 | Estado inicial, indica que el operador 1 aún no se ha guardado y el número que se reconozca deberá guardarse en la variable operador1. |
| | 1 | Estado listo, indica que el operador1 ya tiene un valor y el siguiente número reconocido deberá guardarse en el operador2. |
| Flag_operador2 | -1 | Estado inicial, indica que aún no se ha reconocido ningún número y el número reconocido deberá guardarse en el operador1. |
| | 0 | Estado en espera, indica que ya se reconoció el primer número (operador1) y también se reconoció el caracter de la operación. |
| | 1 | Estado listo, indica que el valor operador2 ya se ha guardado |

Las reglas serían entonces:

```
{NUM} {
    if(flag_operador1==0){
        operador1=atof(yytext);
        flag_operador1=1;
    }
    else if(flag_operador2==0){
        operador2=atof(yytext);
        flag_operador2=1;
    }

    if((flag_operador1==1) && (flag_operador2==1)) {
        evaluate();
        flag_operador1=0;
        flag_operador2=0;
```

```
    }
}

{OP} {
    operacion=(char) *yytext;
    flag_operador2=0;
}
```

Solo nos faltaria evaluar las expresiones, para lo cual debemos generar una función de evaluación que definiremos en la sección de código de usuario. La respuesta se almacenará en la variable respuesta:

```
void evaluate(){
    switch(operacion) {
        case '+':
            respuesta=operador1+operador2;
            break;

        case '-':
            respuesta=operador1-operador2;
            break;

        case '*':
            respuesta=operador1*operador2;
            break;


        default:
            printf("No se reconoce la operacion");
            break;
    }
    printf("La respuesta es = %0.2f \n",respuesta);
}
```

Ya que hemos modificado la sección de código de usuario, vamos a redefinir la función main y yywrap así:

```

int yywrap(){}

int main(){
    printf(“Calculadora simple \nIngrese la expresion a analizar: ”);
    yylex();
    return 0;
}
```

Finalmente la sección de definiciones deberá quedar así:

```
%{
int flag_operador1=0,flag_operador2=-1;
char operacion;
```

```
float operador1=0,operador2=0,respuesta=0;
void evaluate();

%}


DIGIT [0-9]
NUM {DIGIT}+(\.{DIGIT}+)?
OP [*/+-]
```

## IV. ACTIVIDADES

Realiza las siguientes actividades:

1. Reconocer la división
2. Reconocer la operación módulo (%)
3. Reconocer la operación potencia (^)
4. Reconocer los errores e imprimir “No se reconoce la operación”.
5. Reconocer los espacios entre la operación.

## VII. Ejercicios

1. Reconocer como entradas textos en archivos (Hasta ahora nuestro input ha sido solo el ingreso por teclado desde la terminal).
2. Hacer un informe definiendo símbolos terminales y no terminales.
3. Hacer una gramática con símbolos terminales y no terminales de una calculadora.

## VII. Bibliografía y referencias

1. https://es.qwe.wiki/wiki/Flex_(lexical_analyser_generator)
2. https://docplayer.es/46330679-Lex-flex-generacion-de-analizador-lexico-p-1.html
3. http://web.archive.org/web/20130627190411/
4. http://wwwdi.ujaen.es:80/~nacho/Flex.htm