

## Laboratorio 2 - Teoría de la Computación

### EJERCICIOS PROPUESTOS

1. Reconocer la multiplicación(\*)
2. Reconocer la división(/)
3. Reconocer la operación del módulo(%)
4. Reconocer la operación potencia(^)
5. Reconocer los errores e imprimir “No se reconoce la operación”
6. **EXTRA - BONUS:** No es parte de la calificación pero si agrega puntos. Se solicita poder abarcar todas las operaciones en un solo archivo, es decir tener la calculadora en un solo archivo, aplicando estructuras condicionales para que ejecute la operación según el operador recibido.

Se tomó como base las variables vistas ya en clase, agregando 2 variables más y una función para calcular la potencia de las 2 entradas.

```
int flag_operador1=0, flag_operador2=0, flag_operador=0;
void evaluate();
float operador1=0, operador2=0, respuesta=0;
char * operator;
int potencia(int a , int b);
```

Se añadió otra expresión: OPERADOR, que identifica los operadores de la entrada, además de identificar solo uno de estos, y se usaron las mismas expresiones vistas en clase:

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

Se definió la siguiente funcionalidad para el patrón OPERADOR:

```
{OPERATOR} {
    if(flag_operador == 0)
    {
        operator = yytext;
        printf("La operacion es: %c \n", *operator);
        flag_operador = 1;
    }
}
```

Verificando que su bandera no se haya activado aún y actuando asignando el mencionado operador en la variable operator y activando la bandera.

La función potencia, tiene una implementación sencilla:

```

int potencia(int a , int b)
{
    if(b == 0)
    {
        return 1;
    }
    int result = 1;
    for(int i=0 ; i<b ; i++)
    {
        result *= a;
    }
    return result;
}

```

Luego en la función evaluate, se usó la estructura de control switch de la siguiente forma:

```

switch(*operator)
{
    case '+':
        printf("Suma\n");
        resultado = operador1 + operador2;
        break;
    case '-':
        printf("Resta\n");
        resultado = operador1 - operador2;
        break;
    case '*':
        printf("Multiplicacion\n");
        resultado = operador1 * operador2;
        break;
    case '/':
        printf("Division\n");
        resultado = operador1 / operador2;
        break;
    case '%':
        printf("Modulo\n");
        int operadorEntero1 = operador1;
        int operadorEntero2 = operador2;
        resultado = operadorEntero1%operadorEntero2;
        break;
    case '^':
        printf("Potencia\n");
        resultado = potencia(operador1 , operador2);
        break;
}

```

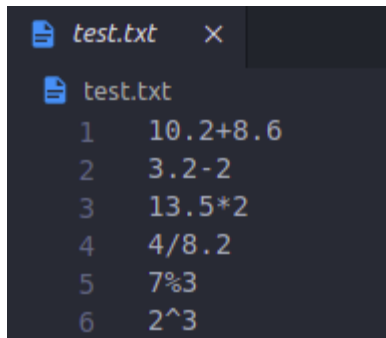
Actuando según cada uno de los operadores, cuando se trata de divisiones por cero, se actúa controlando la salida posteriormente, y para el módulo, se convierten las

entradas en valores enteros, mientras que para la potencia se hace el llamado a la función potencia, para obtener un resultado medianamente correcto. Para finalmente dar solución al problema mostrando la operación y su respectivo resultado, en caso de tener una solución factible.

## CUESTIONARIO

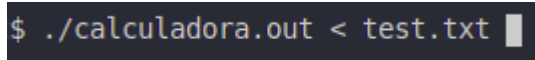
### 1. Reconocer como entradas textos en archivos (Hasta ahora nuestro input ha sido solo el ingreso por teclado desde la terminal).

Usando el siguiente archivo, con operaciones de prueba, para ejecutar el programa teniendo como entrada este archivo de texto, podemos hacer uso de la manipulación de I/O, de forma que el programa reciba el archivo como entrada:

A screenshot of a text editor window titled 'test.txt'. The window contains a list of six arithmetic operations, each preceded by a line number from 1 to 6. The operations are: 1. 10.2+8.6, 2. 3.2-2, 3. 13.5\*2, 4. 4/8.2, 5. 7%3, and 6. 2^3.

```
test.txt
1  10.2+8.6
2  3.2-2
3  13.5*2
4  4/8.2
5  7%3
6  2^3
```

Por ejemplo ejecutando el siguiente comando en la terminal:

A terminal window showing a command prompt with the command './calculadora.out < test.txt' entered. The prompt is '\$' and there is a cursor at the end of the command.

```
$ ./calculadora.out < test.txt
```

Tendremos la siguiente salida con la calculadora que se implementó:

```

michael@msac:~/Escritorio/TCLab/lab2$ ./calculadora.out < test.txt
operador 1: 10.20
operador 2: 8.60
Suma
El resultado de operar 10.20 + 8.60 = 18.80
-----

operador 1: 3.20
operador 2: 2.00
Resta
El resultado de operar 3.20 - 2.00 = 1.20
-----

operador 1: 13.50
operador 2: 2.00
Multiplicacion
El resultado de operar 13.50 * 2.00 = 27.00
-----

operador 1: 4.00
operador 2: 8.20
Division
El resultado de operar 4.00 / 8.20 = 0.49
-----

operador 1: 7.00
operador 2: 3.00
Modulo
El resultado de operar 7.00 % 3.00 = 1.00
-----

operador 1: 2.00
operador 2: 3.00
Potencia
El resultado de operar 2.00 ^ 3.00 = 8.00
-----

michael@msac:~/Escritorio/TCLab/lab2$ █

```

## 2. Definir símbolos terminales y no terminales de una calculadora

Usualmente se hace uso del carácter: \$ para identificar el final de una cadena, en muchos algoritmos, sobretodo en los de búsqueda de texto o patrones.

Claro que dependiendo del lenguaje se recomienda usar un carácter que sea el menos usado en su alfabeto.

Mientras que se suele tomar al resto de signos como no terminales.

Para una calculadora sería útil utilizar alguna letra o símbolo que no tenga efecto en una operación matemática, en el caso de esta calculadora, también puede ser el símbolo \$, ya que usualmente no representa alguna operación en calculadoras.