Actividad 05. Calculadora

El objetivo de esta actividad es la construcción de un intérprete que actúe como una calculadora de expresiones aritméticas y lógicas.

Las órdenes que recibirá serán instrucciones de asignación, similares a las de cualquier lenguaje imperativo, que asignan un valor numérico o booleano a una variable. El intérprete deberá escribir un mensaje en el que aparezca el nombre de la variable, el valor asignado, el tipo de ese valor y el número de orden de la instrucción.

El intérprete se implementará a partir de un analizador léxico (construido con Lex) y un analizador sintáctico (construido con Bison).

Al diseñar el intérprete, se deben tener en cuenta los siguientes aspectos:

- Las instrucciones de la calculadora serán siempre asignaciones donde, en la parte izquierda, habrá una variable. A continuación, el operador de asignación (**:=**). En la parte derecha habrá una expresión numérica o lógica.

- Las variables solo pueden aparecer en la parte izquierda de la asignación, y se definen según las reglas siguientes:

  - El nombre de la variable empieza siempre por una letra, y luego puede haber letras, números o guiones bajos.

  - La longitud máxima se supone que será menor de 25 caracteres. (No es necesario comprobar que la longitud real del identificador es inferior.)

- Las expresiones numéricas pueden tener un resultado real o entero. Las expresiones lógicas tendrán como resultado los valores lógicos **true** o **false**.

- Los valores reales deben tener un punto, y puede estar vacía la parte entera o la parte decimal (3.14, 2., .5), pero no ambas. También se pueden admitir reales en notación científica (aunque no es obligatorio).

- En las expresiones numéricas pueden aparecer los siguientes operadores aritméticos:

  \+ - * / div %

  - El operador **div** calcula el resultado de la división entera. Los dos operandos deberán ser, por tanto, valores enteros.

  - El operador **/** calculará el resultado de la división exacta, con decimales, independientemente del tipo de los operandos.

  - **div** y **/** tienen la misma prioridad que * y %.

  - El resto de operadores tienen el mismo significado y la misma prioridad que en el lenguaje C/C++.

  - El operador - puede referirse a la operación binaria de resta o a la unaria de cambio de signo.

- Si se intenta dividir entre 0 se cometerá un error semántico y habrá que indicarlo como resultado de la evaluación de la asignación.

- El operador **%** (módulo, resto de la división entera) solo puede aplicarse sobre valores de tipo entero. Si aparece en una expresión cuyos operandos son reales, habrá un error semántico y habrá que indicarlo al evaluar la instrucción correspondiente.

- Para definir expresiones lógicas se podrán utilizar comparadores (operadores para realizar comparaciones) y operadores lógicos.

  - Los siguientes comparadores solo se pueden usar con expresiones aritméticas:
    `<  <=  >  >=`

  - Los siguientes comparadores se pueden usar con expresiones aritméticas y lógicas:
    `==  !=` (distinto)

  - Los operadores lógicos, de menor a mayor prioridad, son:
    `and  or  not`

- El orden de prioridad de los operadores es similar a la prioridad en el lenguaje C/C++. En esta tabla están ordenados por prioridad, de mayor a menor, todos los operadores que se utilizan en esta actividad.

`not`
`-  (cambio de signo)`
`*  /  div  %`
`+  -`
`<  <=  >=  >`
`==  !=`
`and`
`or`

- Pueden utilizarse paréntesis en cualquier expresión. Los paréntesis modifican la prioridad de los operadores.

- Puede haber espacios en blanco entre los datos y las operaciones, en cualquier lugar de la instrucción; pero no dentro de un operador (por ejemplo, **n o, : = , < =** no son correctos).

- Es necesario controlar el tipo de las expresiones definidas (entero, real o lógico), que se indicará en el mensaje de salida.

  Para distinguir entre expresiones enteras y reales se aconseja usar un atributo semántico compuesto, según lo explicado en las sesiones de laboratorio. El uso de banderas para la distinción entre expresiones enteras y reales no es adecuado.

  Una expresión aritmética es de tipo real siempre que en ella aparezca algún número real, aunque el resultado sea un número sin decimales. Por ejemplo, 2.5*2 es una expresión real que vale 5.0.

- Si hay errores en una instrucción el intérprete debe mostrar un mensaje de error y seguir funcionando hasta que la palabra SALIR (escrita con letras mayúsculas o minúsculas) haga que el programa termine.

Ejemplos de funcionamiento

```
LISTO> A := 2 * 3 + 1
Instrucción 1: La variable A, de tipo entero, toma el valor 7
LISTO> Bbbbbb
Error sintáctico en la instrucción 2
LISTO> var := 22*4 <= 100-12 and 5 != 2*2
Instrucción 3: La variable var, de tipo lógico, toma el valor true
LISTO> X := ;(2*55
Error sintáctico en la instrucción 4
LISTO> C := 20 / (4 – 2*2)
Error semántico en la instrucción 5: división por 0
LISTO> Salir
```

Detalles del envío y de la evaluación de la calculadora

Esta actividad debe entregarse **antes del 14 de abril (a las 23:55).**

Puede realizarse individualmente o en pareja. Si se hace en pareja, ambos miembros deberán subir la misma tarea al aula virtual. El nombre de las dos personas deberá aparecer en un comentario dentro del fichero ***makefile***.

Se subirá a la tarea un fichero comprimido con el nombre y apellidos de las personas que hayan hecho la actividad.

Ese fichero comprimido deberá contener, al menos, los ficheros fuente correspondientes al analizador léxico y sintáctico y un fichero ***makefile***.

El fichero ***makefile*** debe estar diseñado para que se compile todo y comience la ejecución del programa calculadora al escribir la orden "***make***".

Evaluación

Para evaluar la actividad se ejecutarán algunas pruebas similares a las que se muestran a continuación.

No es necesario que el mensaje escrito sea idéntico al aquí indicado (aunque sí es aconsejable), pero debe incluir:

- el número de la línea (se considerará válido aunque haya una diferencia de una unidad con lo reflejado en el ejemplo),
- el nombre de la variable,
- el tipo de la variable que, en realidad, es el tipo de la expresión,
- el valor que se asigna a la variable. Si no tiene decimales no importa si éstos aparecen o no, es decir, puede aparecer 7 o 7.0. En el caso de los valores lógicos no debe aparecer un valor numérico, sino **true** o **false**.

En el caso de los errores semánticos no es necesario mostrar el tipo de error semántico que se ha producido.

En el aula virtual hay un fichero de texto, pruebas_enunciado.txt. Debería funcionar la siguiente instrucción para probarlo. (En la consola solo se verán los resultados.)

```
> make < pruebas_enunciado.txt
```

Os aconsejamos que vayáis construyendo vuestro propio fichero de pruebas a medida que realicéis la implementación.

Instrucción	Resultado
`A := 2 * 3 + 1`	Instrucción 1: La variable A, de tipo entero, toma el valor 7
`B := (12.9 – 4.4) / 2`	Instrucción 2: La variable B, de tipo real, toma el valor 4.25
`C := true and not false`	Instrucción 3: La variable C, de tipo lógico, toma el valor true
`cuenta := 3 + 5 * -6 + 10`	Instrucción 4: La variable cuenta, de tipo entero, toma el valor -17
`operacion := 4.3 + 9%2`	Instrucción 5: La variable operación, de tipo real, toma el valor 5.3
`res := true or (5 >=10) == false and (5 != 5)`	Instrucción 6: La variable res, de tipo lógico, toma el valor true
`var_ := 22*4 <= 100-12 and 5 != 4 div 2`	Instrucción 7: La variable var_, de tipo lógico, toma el valor true
`2 + 3 * 6`	Error sintáctico en la instrucción 8
`div := 5 / 2`	Error sintáctico en la instrucción 9
`x := 10 div (6 – 3*2)`	Error semántico en la instrucción 10: división por cero
`resto := 6 / 2.0 % 2`	Error semántico en la instrucción 11: se usa el operador % con operandos reales
`abc : = - 1+2*3-4 div 5`	Error sintáctico en la instrucción 12
`ejemplo := 4 and 5 <= 10`	Error sintáctico en la instrucción 13
`Salir`	Vuelva al terminal

Explicación de los errores sintácticos

Error sintáctico en la instrucción 8: falta la asignación; una expresión sola no es correcta.

Error sintáctico en la instrucción 9: div no es un identificador de variable, sino un operador aritmético.

Error sintáctico en la instrucción 12: El operador de asignación aparece separado por un espacio en blanco.

Error sintáctico en la instrucción 13: Es incorrecto usar un entero como operando de la operación **and** lógica.