Tarea 1: Aritmética básica

Programación 3
Instituto de Computación
Facultad de Ingeniería de la Universidad de la República
Curso 2010
Versión 1.4

1. Objetivos

- Conocer las estructuras básicas del lenguaje C.
- Diseñar e implementar un tipo abstracto de datos de manera m4-dular.
-

comando dado y le devuelve el resultado acorde por pantalla. Un usuario puede ingresar un máximo de 256 caracteres en línea de comandos.

Los comandos son los siguientes:

- Definir variable:
  - Comando: VAR *nom_var* = *valor*
  - Descripción: Define la variable de nombre *nom_var* con valor *valor*. En caso de existir la actualiza.
  - Retorno: OK. *nom_var* agregada.
  - Errores: Se le indica al usuario mediante un mensaje en pantalla en los siguientes casos:
    - Cuando el nombre de la variable es vacío:
      ERROR: Nombre de variable vacio.
    - Se alcanzó la máxima cantidad de variables en el sistema:
      ERROR: Se alcanzo la maxima cantidad de variables.
- Definir expresión:
  - Comando: EXPR *nom_expr* = *expr*
  - Descripción: Define la expresión *expr* con el nombre *nom_expr*.
  - Retorno: OK. *nom_expr* agregada.
  - Errores: Se le indica al usuario mediante un mensaje en pantalla en los siguientes casos:
    - Cuando la expresión no es válida:
      ERROR: La expresion no es valida.
    - Cuando el nombre de la expresión es vacío:
      ERROR: Nombre de expresion vacio.
    - Se alcanzó la máxima cantidad de expresiones en el sistema:
      ERROR: Se alcanzo la maxima cantidad de expresiones.
  - Observación: Asocia la expresión *expr* con el nombre *nom_expr*. Donde *expr* puede ser:
    - Una constante.
    - El nombre de una variable.
    - El nombre de una expresión.
    - expr oper expr: *oper* $\in \{+, -, *, /\}$ Representando las operaciones habituales: suma, resta, multiplicación y división, respectivamente.
    - (expr)
- Evaluar expresión:
  - Comando: EVAL *expr*
  - Descripción: Evalúa la expresión *expr*.
  - Retorno: OK. *res*
    Donde *res* es el resultado de la evaluación.
  - Errores: Se le indica al usuario mediante un mensaje en pantalla en los siguientes casos:

        - Cuando existe una división por cero:
          ERROR: Division por cero.
        - Cuando falta definir una variable o expresión:
          ERROR: Falta definir variable o expresion.
    - Observaciones:
        - La expresión se define igual que en el punto anterior.
        - En el caso que haya más de un error, se muestra el que aparezca más arriba en la lista.
        - El resultado se imprime con dos dígitos después de la coma.

- Persistir datos:
    - Comando: SAVE *prefijo_nombre_archivo*
    - Descripción: Guarda los datos en disco en dos archivos: *prefijo_nombre_archivo*-Exprs.txt y *prefijo_nombre_archivo*-Vars.txt. En el primero se guardan las expresiones y en el segundo las variables. El orden en que se guardan es cronológico a como fueron definidos los nombres.
    - Retorno: OK. Datos guardados.
    - Precondición: *prefijo_nombre_archivo* no contiene espacios.
- Cargar datos:
    - Comando: LOAD *prefijo_nombre_archivo*
    - Descripción: Carga los datos en disco con nombre *prefijo_nombre_archivo*. Los datos existentes en el sistema son remplazados por los nuevos.
    - Retorno: OK. Datos cargados.
    - Precondición: Los archivos existen y tienen la sintaxis correcta.
- Salir del sistema:
    - Comando: SALIR
    - Descripción: Sale del sistema borrando toda la memoria asociada.
    - Retorno: No tiene.
- En cualquier otro caso se debe devolver el mensaje: Comando no reconocido.

3.2. Módulo *lectorExp*

El módulo *lectorExp* contiene la operación:

```
ABExp* leerExpr( const char* expr, bool &error)
```

Retorna un ABexp formado a partir de la expresión expr, y retorna true en error si la expresión ingresada no es una expresión válida (paréntesis no balanceados, operadores consecutivos o falta de parámetros de un operador).

Se proporciona la implementación de este módulo.

3.3. Formato de los archivos

A continuación se definen los formatos de los archivos de persistencia para las variables y expresiones definidas por el usuario.

3.3.1. Variables

Cada línea del archivo representa la definición de una variable, con el siguiente formato:

```
<def var="(NOM)">(VALOR)</def>
```

Donde (NOM) es el nombre de la variable y (VALOR) es el valor.

3.3.2. Expresiones

Cada línea del archivo representa una expresión, con el siguiente formato:

```
<def exp="(NOM)">(EXPR)</def>
```

Donde: (NOM) es el nombre de la expresión y (EXPR) es de la forma:

- Si es un valor:

  ```
  <expr tipo="VAL">(VALOR)</expr>
  ```

  Donde (VALOR) es el valor.

- Si es una referencia:

  ```
  <expr tipo="VAR">(NOMBRE)</expr>
  ```

  Donde (NOMBRE) es el nombre de la variable o expresion que se referencia.

- Si es una operación:

  ```
  <expr tipo="(OPER)">(EXPRIZQ)(EXPRDER)</expr>
  ```

  Donde:

  - (EXPRIZQ) y (EXPRDER) son expresiones del mismo tipo que (EXPR).
  - (OPER) es SUMA, RESTA, MULT o DIV si la operación es una suma, resta, multiplicación o división respectivamente.

3.4. Ejemplo de ejecución

El siguiente es un ejemplo de ejecución del programa.

```
Bienvenido a el mini-sistema:
> VAR X=10
OK. X agregada.
> VAR Y=20
OK. Y agregada.
> EXPR XMASY=X+Y
OK. XMASY agregada.
> EVAL X+Y
OK. 30.00.
> EVAL XMASY
OK. 30.00.
> SAVE PRUEBA
OK. Datos guardados.
> VAR Z=20
OK. Z agregada.
> EVAL Z
OK. 20.00.
> LOAD PRUEBA
OK. Datos cargados.
> EVAL Z
ERROR: Falta definir variable o expresion.
> SALIR
```

3.4.1. Archivo de variables

```
<def var="X">10.000000</def>
<def var="Y">20.000000</def>
```

3.4.2. Archivo de expresiones

```
<def exp="XMASY">
<expr tipo="SUMA">
<expr tipo="VAR">X</expr>
<expr tipo="VAR">Y</expr>
</expr>
</def>
```

En el ejemplo anterior si bien existen saltos de línea para una mejor visualización pero no se exige que sea soportado.

7. Advertencia sobre el manejo de la memoria

Cuando un programa contiene errores en el manejo de la memoria, su comportamiento puede ser inestable. Esto implica que algunas veces funciona correctamente y otras no. En ciertos casos esto puede inducir a creer (erróneamente) que ciertos programas, que en realidad son incorrectos, funcionan correctamente. Este aspecto es influenciado, entre otras cosas, por el sistema operativo en el que se ejecuten los programas. Recomendamos tener sumo cuidado con este punto y testear los módulos el sistemas operativos Windows XP. CON LA VERSIÓN DE MINGW INSTALADA EN LAS SALAS DE INFORMÁTICA DE LA FACULTAD.

8. Sobre la individualidad del trabajo

El laboratorio es INDIVIDUAL. Los estudiantes pueden estudiar en grupo y consultarse mutuamente, pero NO pueden realizar en grupo las tareas de codificación, escritura, compilación y depuración del programa.

Los trabajos de laboratorio que a juicio de los docentes no sean individuales serán eliminados, con la consiguiente pérdida del curso, para todos los involucrados. Además todos los casos serán enviados a los órganos competentes de la Facultad, lo cual puede acarrear sanciones de otro carácter y gravedad para los estudiantes involucrados.

No existirán instancias en el curso para reclamos frente a la detección por parte de los docentes de trabajos de laboratorio no individuales, independientemente de las causas que pudiesen originar la no individualidad. A modo de ejemplo y sin ser exhaustivos: utilizar código realizado en cursos anteriores u otros cursos, perder el código, olvidarse del código en lugares accesibles a otros estudiantes, prestar el código o dejar que el mismo sea copiado por otros estudiantes, dejar la terminal con el usuario abierto al retirarse, enviarse código por mail, etc. Asimismo, se prohíbe el envío de código al grupo de noticias del curso, dado que el mismo será considerado como una forma de compartir código y será sancionada de la manera más severa posible.

Es decir que se considera a cada estudiante RESPONSABLE DE SU TRABAJO DE LABORATORIO Y DE QUE EL MISMO SEA INDIVIDUAL. NO CONFIAR en el borrado automático del directorio pub de las salas Windows. Es decir antes de cerrar sesión borrar todos los archivos del directorio.

9. Fecha de entrega

El trabajo debe entregarse el día **lunes 30 de agosto de 2010 antes de las 22:00 horas**. La entrega se realizará mediante un formulario que se habilitará oportunamente en la página web del curso.