Práctica 3

Construir una herramienta que permita compilar y ejecutar ficheros con secuencias de expresiones de cálculo matemático.

Orientación

Se pide una herramienta que pueda interpretar y ejecutar ficheros con expresiones de la forma:

x = 3;
y = 2 * pi * x;
z = x * sin (y + pi/6);

Como resultado de la ejecución se presentarán los valores que toman las distintas variables que aparecen en el fichero. Se valorará que la herramienta incorpore las operaciones aritméticas y las funciones propias de una calculadora científica (sin(),cos(),log(),exp(),…), aplicadas a números tanto enteros como reales en notación científica.

La herramienta puede realizarse con una etapa de análisis léxico y una etapa de análisis sintáctico, haciendo uso de herramientas como flex y bison. Al análisis sintáctico es necesario añadirle aquellas acciones semánticas que permitan realizar las operaciones descritas. Como no vamos a sintetizar código ejecutable, podemos utilizar una tabla de símbolos para almacenar las variables que aparecen en el fichero de entrada, y los valores que toman a lo largo del cálculo.

Detalles de clase

- Bison es un programa que genera analizadores sintácticos

- La consola de comandos debe de tener un prompt que permita al usuario escribir expresiones matemáticas.

- No vamos a generar código objeto, pues la consola interpreta y ejecuta.

. El usuario puede escribir "2+2", y al apretar el Intro, después la consola interpretará y ejecutara.

- Se puede utilizar el ";" para indicar si se quiere hacer un Echo o no.

- La consola debería realizar todas las operaciones matemáticas, incluso en notación científica. Se debe trabajar en notación científica por defecto.

- A la hora de redactar la sintaxis de ese interprete, se debe tener en cuenta la precedencia de operadores + - * / ^ ()

- Incorporar funciones matemáticas básicas: trigonométricas

- Incorporar algunas constantes: PI, E

- La consola de comandos no solo ejecuta código, sería bueno que manejara la noción de variable y permitir cosas como:

    x= 2* pi;

    y = sin(x);

  Para ello es bueno utilizar una tabla de símbolos (la de las otras prácticas).

- A nivel léxico hay que tratar los identificadores. Habrá tres tipos: constantes, variables y funciones. Cuando el analizador léxico encuentra el identificador, se le pregunta a la tabla de símbolos qué es. Puede que sea algo nuevo, o que ya esté insertado.

- Para implementar las funciones (sin, cos, tan, ...) implementar la noción de librería. Para que el usuario invoque a la función seno, es necesario que aparezca ese identificador en la tabla de símbolos y que no se pueda realizar para hacer una asignación (sin = 2). Una forma es usar la librería de math.h de C. Así pues, cuando se incluye, sabemos que si

ejecutamos cualquier código en C, entonces se utilizará la implementación sin() que esta en math.h. Para ello hay que analizar lexicamente (easy) , sintacticamente (expresión que se corresponde a una expresión bien formada) y semanticamente (cuando nos encontramos con sin, invocamos la función seno de math.h y se pasa el valor x).

Hay varias formas de hacerlo:

- A nivel léxico:
    1. Introducir cada cadena al analizador léxico. -> NO RECOMENDABLE, difícil ampliación.
    2. Gestionar a nivel léxico cada función a través de la tabla de símbolos.
- A nivel semántico: Trabajar con las funciones como punteros a funciones. Ejemplo en el manual de bison.

- Tiene que poder responder a comandos introducidos en la propia consola.

- Debe poder interpretar ficheros de comandos. Debe haber un comando para poder cargar el fichero (cargar, leer y ejecutarlo). También se deberá poder arrancar la consola con el fichero desde el momento inicial.

- Es importante que tenga una ayuda que explique como utilizar la consola, así como un mensaje inicial que indique como acceder a dicha ayuda.

- Comando 'workspace', mostrar las variables y su contenido. Incluir algún comando para limpiar el workspace (eliminar variables)

-Como no se generará código objeto, necesitamos almacenar los valores de las variables. Esto se puede hacer en la tabla de símbolos. Para las variables se puede añadir un nuevo campo en la tabla, que se llame valor que se vaya actualizando.

- Hay que tener cuidado con la gramática que se va a diseñar. Ejemplos en el manual de bison. Es importante que se compruebe el resultado de la compilación con bison: evitar errores reducción - reducción, reducción – desplazamiento.

Gestión de errores

- Utilizar la gestión de errores de bison:

    - Saber si una expresión tiene paréntesis mal emparejados
    - Si se está realizando alguna operación incorrecta (dividir por 0, por ejemplo)
    - Advertir al usuario de la no industrialización de variables

- La consola no se puede quedar bloqueada por un error.

Posibles extensiones

- Plantear el tema de matrices / vectores

- Funciones de más de un argumento: avg(x,y,z,...), min(x,y,z,...), etc

Gestión de tipos

- Se puede hacer gestión de tipos, pero puede ser muy complicado.

- Se puede hacer como python, mathlab, en la que la verificación de tipos es dinámica. Se mete un valor a una variable (z=5.2), y elegir el tipo más adecuado para su interpretación; se pueden realizar cast implícitos, etc...