GRUPO 08

Integrantes:

- Negoci, Facundo – 9502/0
- Figueroa, Mauro – 9310/4
- Galliani, Lucrecia - 12280/1

Lenguajes asignados:

- C
- Matlab
- Java

Bibliografía

- Matlab Una introducción con ejemplos prácticos de Amos Gilat.
- Manual Básico de Matlab Servicios Informáticos U.C.M.
- https://es.mathworks.com
- https://www.oracle.com
- http://www.tutorialspoint.com

Introducción y objetivos:

A continuación, describiremos un nuevo lenguaje llamado **"C-MJ" (C- M**atlab **J**ava), que presenta características basadas en los lenguajes de programación C*, Java y MATLAB.*

El objetivo principal de "C-MJ" es el procesamiento de datos y análisis estadísticos. Este lenguaje está dirigido a programadores, con el objetivo de que estos logren desarrollar de manera simple y eficiente programas para el procesamiento y análisis de datos provenientes de bases de datos, archivos de texto o archivo de intercambio de datos.

Este lenguaje está basado en C debido a su eficiencia, flexibilidad y expresividad, incorpora el paradigma de objetos tomado de JAVA debido al gran soporte de este además su manejo de errores y excepciones, e introduce la simplicidad y legibilidad de MATLAB para el manejo de matrices y desarrollo de cálculos matemáticos.

❖ Legibilidad y simplicidad:

C-MJ cuenta con el manejo de matrices y cálculos matemáticos de MATLAB ya que este está basado en matrices y además el álgebra lineal en MATLAB parece el álgebra lineal de un libro de texto. Eso permite capturar las matemáticas que subyacen a sus ideas de manera directa, lo que se traduce en que su código es más fácil de escribir, leer, comprender y mantener.

❖ Eficiencia:

C-MJ está basado principalmente en C el cual es un lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones optimas además de proporcionar facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

❖ Soporte:

C-MJ al introducir conceptos y características de JAVA logra un gran soporte ya que JAVA es uno de los lenguajes de programación más usados lo cual resulta en la existencia de una gran cantidad de tutoriales, cursos, documentación, libros, etc. Además, este lenguaje puede ser implementado en diferentes plataformas y todos los requerimientos necesarios pueden ser descargados de manera libre y gratuita desde la web.

Sintaxis del Lenguaje

La sintaxis de C-MJ va a ser muy similar a la sintaxis de C, para una mayor claridad a la hora de leer el código se redefinió los delimitadores de comienzo y fin de las sentencias if, switch, for y while
Quedan así:

```
if(<expresión>): <sentencia> endif;
while(<expresión>): <sentencia> endwhile;
for(<expresión>): <sentencia> endfor;
switch(<expresión>): <sentencia> endswitch;
```

Todas las sentencias deben terminar con el carácter;
Los identificadores pueden utilizar letras y números y no hay una longitud mínima ni máxima. No se permite el uso de palabras reservadas en variables y funciones, las palabras reservadas son las del lenguaje C y Java mas endif, endfor, endswitch, endwhile
Los operadores aritméticos son: +, -, *, /, %, ++, --
Los comentarios deben comenzar con /* y terminar con */
El comienzo de una función está dado por el carácter { y el fin con }

El conjunto de caracteres que usa es UTF-8
A continuación, se describe la definición en EBNF la gramática de la estructura FOR y la forma de cargar en una matriz los datos provistos por un archivo en C-JM.
FOR
G =(N,T,S,P)
N={<sentencia-for><expresión><sentencia><función><operación> <asignación> <operador-asignación> <expresión-unaria> <operador-unario> <valor> <variable> <letra> <lógico> <numero><operador>
}
T={(a | ... | z | A | ... | Z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | "+" | "-" | "*" | "/" | "%" | "//" | "^" | "=" | "||" | "!=" | "!" | "!=" | "==" | "&&" | "#" | "||" | ":=" | "." | ";" | *= | /= | %= | += | "-=" | "<<=" | ">>=" | "&=" | "^=" | "|=" | "_" | ":") }
S={<sentencia-for>}
P= {<sentencia-for> ::= for ({<expresión>}+ ; {<expresión>}+ ; {<expresión>}+) : <sentencia> endfor;

<expresión> ::= <asignación> | <condicional> | <expresión-unaria>
<condición>::= (<boolean> | <variable> | <función> | <comparación>)
<sentencia>::= ({<operación>}+ | [<función>]*)
<operación>::=(<asignación> | <sentencia>)
<asignación> ::= <variable> <operador-asignación> (<variable> | <valor> | <función> | <operación>)
<operador-asignación> ::= (= | *= | /= | %= | += | -= | <<= | >>= | &= | ^= | |=)
<expresión-unaria> = <variable>{<operador-unario>}+
<operador-unario> ::= (& | * | + | -| ~ | !)
<valor>::=(<numero>| <lógico>| <string>)
<string> ::= {(<letra> | <numero> | <otro> | <letra><numero><otro>)}+
<variable>::=({<letra>}+ | <string>)
<letra>::=(a | ... | z | A | ... | Z |)
<numero>::=(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
<operador>::=(<lógico>|<operador-unario>)
<matemático>::=("+" | "-" | "*" | "/" | "%" | "//" | "^" | "=")
<lógico>::=("&&" | "||" | "!=")
<función> ::= <nombre>({<variable>}*);
<nombre> ::= {<letra>}+
<otro> ::= ("-" | ":" | "/" | "." | "_")
}
ASIGNAR UN ARCHIVO A UNA MATRIZ
G =(N,T,S,P)
N={<asignación-file-matriz> <filename> <variable_list> <variable> <string> <numero> <letra> <otro> }
T={(a | ... | z | A | ... | Z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | "-" | ":" | "/" | "." | "_") }
S={<asignación-file-matriz>}
P= { <asignacion-file-matriz> ::= load.(<filename>).into(<variable_list>);
<filename> ::= (<string> | <variable>)
<variable_list> ::= <variable>{","<variable>}*
<variable>::=({<letra>}+ | <string>)
<string> ::= {(<letra> | <numero> | <otro> | <letra><numero><otro>)}+
<letra>::=(a | ... | z | A | ... | Z |)
<numero>::=(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
<otro> ::= ("-" | ":" | "/" | "." | "_")
}
}

Aspectos semánticos de asignación e inicialización de variables

❖ C:

Variables en C: El nombre de la variable debe estar compuesto por letras, dígitos y el carácter guion bajo, debe comenzar con una letra o un guion bajo. C es case sensitive por lo que distingue entre mayúsculas y minúsculas. Las variables pueden tomar los siguientes tipos: char, int, float, double, void, Enumeration, Pointer, Array, Structure, Union, etc.
En C las variables se deben definir especificando su tipo. Se pueden definir de la siguiente forma:
type variable_list;

Variable_list puede consistir en uno o más identificadores separados por coma, por ej.:

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

Las variables pueden ser inicializadas en su declaración.
Si no se le asigna un valor por defecto a las variables estáticas, estas son inicializadas implícitamente con el valor NULL (todos los bytes tienen el valor 0), el resto de las variables tienen valor indefinido.
En C las variables pueden ser locales, globales o dinámicas.

Locales:
Son las definidas en el interior de una función y son visibles solo en esa función especifica. Las variables locales de las funciones no existen en memoria hasta que se ejecuta la función, es decir, se crean automáticamente en la entrada a la función y se liberan también automáticamente cuando se termina la ejecución.

Globales:
Son las que se declaran fuera de la función y son visibles desde cualquier función, incluyendo main. La memoria asignada a una variable global permanece asignada a través de la ejecución del programa, tomando espacio valido según se utilice.

Dinámicas:
Se crea y libera durante la ejecución del programa, se crean a petición y se liberan cuando ya no es necesaria. Una variable dinámica puede ser global o dinámica.

```
#include <stdio.h>
int a, b, c;        /* declaración de variables globales*/

int main(){
        int valor;          /*declaración de variable local*/
        printf("Tres valores: ");
        scanf("%d %d %d ", &a, &b, &c);
        valor = a+b+c;
}
```

❖ Matlab:

Matlab es un lenguaje de tipado dinámico, por lo que necesita que se especifique el tipo de la variable. Al momento de definir una variable se le debe asignar un valor para que Matlab pueda alocar el espacio de memoria apropiado, si la variable ya existe se reemplaza el contenido original y aloca el nuevo espacio de memoria donde sea necesario.

En Matlab, cada variable es un arreglo o una matriz, por ejemplo, x = 3, crea una matriz de 1x1 y

almacena el valor 3 en ese elemento. En cuanto al tipo de variables a utilizar pueden ser: entero, real, complejo, carácter, escalares, matriciales, etc.

Reglas sobre los nombres de las variables:

- Pueden tener una longitud de hasta 63 caracteres en Matlab 7 y 31 en Matlab 6.0
- Pueden contener letras, dígitos y el carácter de subrayado.
- deben empezar por una letra y no debe contener espacios en blanco.
- Es case sensitive.

Las variables creadas por una función pertenecen al espacio de trabajo de dicha función, que es independiente del espacio de trabajo base. Es decir, las variables de las funciones son locales y temporales, MATLAB reserva una zona de memoria cuando comienza a ejecutar una función, almacena en esa zona las variables creadas por esa función, y "borra" dicha zona cuando termina la ejecución de la función.
Para hacer que una variable de una función pertenezca al espacio de trabajo base, hay que declararla global, la palabra clave global en una función hace que las variables sean globales.

Ejemplo de código en Matlab

```
r = [7 8 9 10 11];
t = [2, 3, 4, 5, 6];
res = r + t;
disp(res);
res = 'Hola';
disp(res)
```

❖ **Java:**

Es un lenguaje fuertemente tipado por lo que cada variable debe tener un tipo. Hay ocho tipos primitivos en java y cada uno de ellos es una palabra reservada: int, short, long, byte, float, double, char.
El nombre de una variable no debe comenzar con un digito y todos los caracteres deben ser letras, dígitos o _. El símbolo $ está permitido, pero está reservado para fines específicos y no debe utilizarse el símbolo $ en identificadores.
Sintaxis para definir una variable:
Tipo variable1, variable2, …
Las variables pueden ser inicializadas explícitamente mediante una sentencia de asignación o se puede combinar la declaración de una variable con una sentencia de asignación.
Tipo variable_1 = valor_1, variable_2 = valor_2

Constantes en Java
Las constantes se identifican con la palabra clave final, y nunca se puede modificar su valor.
Cuando se necesitan valores constantes en diferentes métodos de una clase se debe declarar la constante junto con las variables de instancia de una clase y etiquetarse con las palabras clave static y final, la palabra clave static significa que la constante pertenece a la clase.
Los valores por defecto de los diferentes tipos de datos son: byte: 0, int: 0, long: 0, float: 0.0f, double: 0.0d, char: '\u0000', String: null, boolean: false.

Código de ejemplo de Java:

```
public class HelloWorld{
        static String legajo = "1111";

    public static void main(String []args){
      String legajo = "2222";
      System.out.println(legajo);
```

```
      imprimir();
    }
    public static void imprimir(){
       System.out.println(legajo);
    }

}
```

Alcance:
En Java tenemos tres tipos de ámbito que pueden aplicar a una variable:

- Local
- Global
- Estático

Variables de ámbito local

Las variables de *ámbito local*, o de bloque, son aquellas que sólo pueden ser accedidas desde el bloque de código en el que han sido declaradas.

Variables de ámbito global

Las variables de *ámbito global*, o de instancia, son aquellas que pertenecen a cada instancia concreta de la clase donde han sido declaradas, y dependiendo del modificador de visibilidad usado podrían ser sólo accedidas desde la propia instancia a la que pertenecen

Variables estáticas

Las *variables estáticas*, o de clase, son aquellas que pertenecen a la propia clase donde han sido declaradas, y dependiendo del modificador de visibilidad usado podrían ser sólo accedidas desde la propia clase en la que han sido declaradas

❖ **C-MJ:**

C-MJ es un lenguaje en el cual siempre se debe declarar el tipo de cada variable como en C y en Java y además estas pueden cambiar de tipo dinámicamente como en Matlab, se debe especificar su nuevo tipo antes de asignarle el nuevo valor.

Los valores por defectos son los mismo que en C, las variables estáticas son inicializadas implícitamente con el valor NULL (todos los bytes tienen el valor 0), el resto de las variables tienen valor indefinido.

Como el lenguaje soporta tanto programación orientada a objetos como procedural, el alcance de las variables va a depender de la forma en que se programe.

Si es procedural las variables pueden ser locales, globales o dinámicas, al igual que en C.

Si es orientada a objetos las variables pueden ser locales, globales o estáticas, al igual que en Java.