CONCEPTOS Y PARADIGMAS DE LENGUAJES DE PROGRAMACIÓN

TRABAJO INTEGRADOR 2017

INTEGRANTES:

- **CARACOTCHE ROMINA ANTONELLA, 11105/1**
- **SALGADO IVAN, 11823/6**
- **STANCICH DAVID ALFREDO, 11429/9**

GRUPO: 21
LENGUAJES: Matlab - Java - C

AYUDANTE: José
HORARIO: Lunes 18hs

BIBLIOGRAFÍA

- **https://www.mathworks.com/help/matlab/**
- **https://es.mathworks.com/products/matlab/features.html**
- **https://es.slideshare.net/GinoPannillo/introduccion-y-operaciones-basicas-matlab-24319251**
- **http://personales.upv.es/jbenitez/data/matlab.pdf**
- **https://docs.oracle.com/javase/7/docs/api/**
- **http://devdocs.io/c/**
- **Teorias de la catedra**

1era Parte (Fecha Final de entrega: 21/4)

A. Realice una introducción a su lenguaje definiendo el nombre y objetivos de diseño del mismo. Mencione las características más importantes del lenguaje de acuerdo a los criterios de evaluación vistos en la teoría. Describa al menos tres criterios. Justifique las características decididas respecto del objetivo del lenguaje.

B. Identifique las características principales de la sintaxis y realice la gramática en EBNF de alguna de las instrucciones más importantes.

C. Enuncie los aspectos semánticos más relevantes respecto de la asignación e inicialización por defecto de variables. Ejemplifica con los lenguajes asignados.
Utilice el compilador de los lenguaje asignados que se encuentra en la siguiente página http://www.tutorialspoint.com/codingground.htm para realizar los ejemplos planteados.

Mathlan

Mathlan - lenguaje matemático - es un lenguaje de desarrollo de aplicaciones de medio nivel que nace como una solución a la necesidad de mejores y más poderosas herramientas de recolección, análisis y almacenamiento de un gran volumen de datos en tiempo real, permitiendo representarlos en gráficos estadísticos.
Las características principales de Mathlan son:

- Es una lenguaje de **medio nivel**, ya que combina las características de un lenguaje de bajo nivel y de alto nivel. En el lenguaje de alto nivel el código es más sencillo y comprensible al utilizar lenguaje humano como read, write, etc. pero es menor su velocidad de cálculo. Para nuestro lenguaje es fundamental un equilibrio entre un lenguaje de bajo nivel y uno de alto nivel, es por esto que tomamos el nivel de C, ya que necesitamos aplicaciones rápidas, eficientes y simples.
- Es **portable**, esto significa que programas escritos pueden ejecutarse en cualquier hardware. Brinda herramientas para crear aplicaciones web y móviles al igual que java
- Es un lenguaje **potente y eficiente**, permitiendo obtener programas rápidos y compactos.
- Es **confiable** ya que hace chequeo de tipos y cuenta con un conjunto de excepciones, esta característica es tomada de java.
- Nuestro lenguaje, al igual que C, sacrifica la simplicidad y legibilidad a cambio de la **velocidad y eficiencia**.
- Para agilizar los cálculos tomamos el **paralelismo** de C, ya que permite el uso completo del potencial de la CPU.
- Cuenta con un **soporte gráfico** tomado de matlab con gran variedad de colores, fuentes y

estilos predeterminados que hacen que sus datos sean más sencillos de interpretar. Las fuentes y líneas suavizadas proporcionan textos y gráficos más nítidos. Los objetos gráficos son más fáciles de usar y proporcionan una sintaxis sencilla para modificar las propiedades.

- Nuestro lenguaje ofrece medios de **soporte** mediante manuales explicativos, tutoriales con ejemplos y videotutoriales. Estos se encuentran en la página oficial www.mathlab.com
- Cuenta con **funciones de importacion y exportacion** de datos que proveen acceso a la información desde los archivos, otras aplicaciones, servicios web y dispositivos externos. Se pueden leer formatos de archivos populares tal como Microsoft Excel, textos, imágenes, audio, video y formatos de dato científico. Las funciones de entrada/salida de archivos de bajo nivel te dejan trabajar con archivos de datos en cualquier formato.
- Mathlan ofrece **herramientas** para procesos de estadísticas y gráficos en un gran número de librerías al igual que matlab.
- Debido a nuestra necesidad de mostrar datos en tiempo real, utilizamos la legibilidad y simplicidad de C, por este motivo nuestro lenguaje sufre de pérdida de **ortogonalidad**.

B) ***Sentencia For:***

G = (N, T, S, P)
N = { <sentencia_for>, <bloque>, <caracter>, <índice>, <alfanumérico>, <incremento>, <dígito>, <real>, <sentencia>, <variable> }
T = {a,..,z, A,..,Z, 0, .., 9, =, : }
S = { <sentencia_for>}
P = {
<sentencia_for> ::= for <variable> ‘=’ <real> [‘:’ <diferencia>] ‘:’ <real> <bloque> end
<caracter> ::= (‘a’ | … | ‘z’ | ‘A’ | … | ‘Z’)
<variable> ::= <caracter> [{ (<caracter> | <dígito>) }*]
<diferencia> ::= [(+|-)] <real> {<real>}*
<dígito> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
<real> ::= [(+|-)] {<digito>}+ [, {<dígito>}+]
<bloque> ::= {<sentencia>}*
<sentencia> ::= (<sentencia_for> | <sentencia_for_vector>| <sentencia_while> | <sentencia_if> | <sentencia_switch> | <sentencia_asignacion> | <sentencia_declaracion> | <llamada_metodo>)
}

Setencia For con vector:

G = (N, T, S, P)
N = {<sentencia_for_vector>, <variable>, <caracter>, <dígito>, <bloque>, <sentencia>}
T = {0, .., 9, .., A, .., Z, .., a, .., z, =}
S = {<sentencia_for_vector>}
P = {
<sentencia_for_vector> ::= ‘for’ <variable> ‘=’ <variable> <bloque> end
<variable> ::= <caracter> [{ (<caracter> | <dígito>) }*]
<caracter> ::= (‘a’ | … | ‘z’ | ‘A’ | … | ‘Z’)
<dígito> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
<bloque> ::= {<sentencia>}*
<sentencia> ::= (<sentencia_for> | <sentencia_for_vector>| <sentencia_while> | <sentencia_if> | <sentencia_switch> | <sentencia_asignacion> | <sentencia_declaracion> | <llamada_metodo>)
}

Sentencia If:

G = (N, T, S, P)
N = {<sentencia_if>, <condición>, <bloque>, <boolean>, <oper_booleano>, <sentencia>}
T = { true, false, ||, &&, ! }
S = {<sentencia_if>}
P = {
<sentencia_if> ::= if <condición> <bloque> [elseif <condición> <bloque>]* [else <condición> <bloque>] end
<condición> ::= [!] <boolean> {[<oper_booleano> [!] <boolean>]}*
<boolean> ::= ('true' | 'false')
<oper_booleano> ::= ('||' | '&&')
<bloque> ::= {<sentencia>}*
<sentencia> ::= (<sentencia_for> | <sentencia_for_vector>| <sentencia_while> | <sentencia_if> | <sentencia_switch> | <sentencia_asignacion> | <sentencia_declaracion> | <llamada_metodo>)
}

Declaración:

G = (N, T, S, P)
N = {<sentencia_declaración>, <tipo_de_dato>, <variable>, <caracter>, <dígito>}
T = {a, .., z, A, .., Z, 0, .., 9, [,], byte, short, int, long, float, double, boolean, char, string}
S = {<sentencia_declaración>}
P = {
<sentencia_declaración> ::= <tipo_de_dato>[('[' ']' | '[' ']''[' ']' | '<'<tipo_de_dato> '>')] <variable>';'
<tipo_de_dato> ::= ('byte' | 'short' | 'int' | 'long' | 'float' | 'double' | 'boolean' | 'char' | 'string' | 'Object')
<variable> ::= <caracter> [{ (<caracter> | <dígito>) }*]
<caracter> ::= ('a' | … | 'z' | 'A' | … | 'Z')
<dígito> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
}

Asignación:

G = (N, T, S, P)
N = {<sentencia_asignación>, <tipo_de_dato>, <variable>, <caracter>, <dígito>, <expresión>, <operador>, <número>}
T = { =, a, .., z, A, .., Z, 0, .., 9, -, +, /, *, byte, short, int, long, float, double, boolean, char, string, Object}
S = {<sentencia_asignacion>}
P = {
<sentencia_asignación> ::= [<tipo_de_dato>] <variable> '=' <expresión>
<tipo_de_dato> ::= ('byte' | 'short' | 'int' | 'long' | 'float' | 'double' | 'boolean' | 'char' | 'string' | 'Object')
<variable> ::= <caracter> [{ (<caracter> | <dígito>) }*]
<caracter> ::= (a | .. | z | A | .. | Z)
<dígito> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)
<expresión> ::= (<expresión><operador><expresión> | <número> | <variable>)
<operador> ::= (+ | - | * | /)
<número> ::= [(+|-)] {<dígito>}+ [, {<dígito>}+]
}

Sentencia While:

G = (N, T, S, P)
N = {<sentencia_while>, <condición>, <bloque>, <oper_booleano>, <boolean>, <sentencia>}
T = { !, ||, &&, true, false}
S = {<sentencia_while>}
p = {
<sentencia_while> ::= while <condición> <bloque> end
<condición> ::= [!] <boolean> {[<oper_booleano> [<oper_negacion>] <boolean>]}*
<oper_booleano> ::= (‘||’ | ‘&&’)
<boolean> ::= (‘true’ | ‘false’)
<bloque> ::= {<sentencia>}*
<sentencia> ::= (<sentencia_for> | <sentencia_for_vector>| <sentencia_while> | <sentencia_if> | <sentencia_switch> | <sentencia_asignacion> | <sentencia_declaracion> | <llamada_metodo>)
}

Sentencia Do While:

G = (N, T, S, P)
N = {<sentencia_while>, <condición>, <bloque>, <oper_booleano>, <boolean>, <sentencia>}
T = { !, ||, &&, true, false}
S = {<sentencia_while>}
p = {
<sentencia_while> ::= do <bloque> while <condición>
<condición> ::= [!] <boolean> {[<oper_booleano> [!] <boolean>]}*
<oper_booleano> ::= (‘||’ | ‘&&’)
<boolean> ::= (‘true’ | ‘false’)
<bloque> ::= {<sentencia>}*
<sentencia> ::= (<sentencia_for> | <sentencia_for_vector>| <sentencia_while> | <sentencia_if> | <sentencia_switch> | <sentencia_asignacion> | <sentencia_declaracion> | <llamada_metodo>)
}

C) **Java**

```
int a;
System.out.println(a);          // Error en tiempo de compilación, la variable “a” se debe inicializar.

float b;
System.out.println(b);          // Error en tiempo de compilación, la variable “b” se debe inicializar.

Boolean valor;
System.out.println(valor);      // Error en tiempo de compilación, la variable “valor” se debe inicializar.
```

Sucede lo mismo con el resto de tipos primitivos (byte, short, int, long, float, double, char, boolean) y Wrapper (Byte, Short, Integer, Long, Float, Double, Character, Boolean)

```
double c = 12.5;
```

```
System.out.println(c);                  // Imprime en pantalla el número 12.5

Boolean bool = null;
System.out.println(bool);               // Imprime en pantalla null.

Character car = new Character('A');
System.out.println(car);                // Imprime en pantalla la letra "A"

Integer num = 50;
System.out.println(num);                // Imprime en pantalla el número 50.
```

Java cuenta con otra clase de tipos de datos que son Objetos, la cual incluye el tipo Wrapper, los que trae definido en su biblioteca (Scanner, ArrayList, String, etc), los tipos de datos definidos por el usuario, donde el mismo programador puede definir sus propias clases para luego poder crear instancias, y los objetos especiales (arreglos y matrices), se consideran objetos especiales porque carecen de métodos de instancia.

```
String s;
System.out.println(s);          // Error en tiempo de compilación, la variable "s" se debe inicializar.

String s = "Grupo - 21";
System.out.println(a);          // Imprime en pantalla "Grupo - 21".

int[] v;                        // Declaración de un arreglo de enteros.
System.out.println(v);          // Error en tiempo de compilación, la variable "v" se debe inicializar.

int[] v = new int[5];           // Declaracion e inicializacion de un arreglo de enteros de tamaño 5.
System.out.println(v);          // Muestra en pantalla la dirección de memoria donde está el objeto.

int[] v = new int[5];
System.out.println(v[0]);       // Muestra en pantalla el dato en la posición 0, el cual es 0.
```

En java cuando se crea un arreglo y a este no se le asigna ningún valor se inicializa con sus valores por defecto.

Tipo de dato	**Valor por defecto**
byte, short, int, long	0
float, double	0.0
boolean	false
char	/n (salto de línea)
Objeto	null

Cuando se crear clases en Java y sus variables de instancia no se inicializan, estas se inicializan con los valores por defecto según su tipo de dato.

C

```
int a;
printf("Variable a: %d", a);          // Imprime en pantalla el valor por defecto 0.

char b;
printf("Variable b: %c", b);          // Imprime en pantalla el valor por defecto ' ' (espacio)

float c;
printf("Variable c: %f", c);          // Imprime en pantalla el valor por defecto 0.000000

int a;
a = a + 5;
printf("Variable a: %d", a);          // Imprime en pantalla el valor 5
```

En el lenguaje de programación C, las variables que no se inicializan el compilador les asigna sus valores por defecto.

Tipos de datos	**Valor por defecto**
short, int, long, long long	0
float, double	0.000000
char	' ' (espacio)

En C podemos anteponer la palabra signed o unsigned en los tipos de datos para indicar si queremos que esa variable contenga o no signo. Por defecto si no se pone nada antes del tipo de dato el compilador automáticamente asume por defecto que es un tipo signed (con signo)

Las cadenas de palabras en C se manejan como arreglos de char (caracteres). Por ejemplo:

```
char v[5] = "Hola";
printf("Variable b: %s", v);          // Muestra en pantalla el mensaje Hola.
```

Struct se utiliza para juntar un grupo de variables en una única estructura. Cuando se crea esta nueva estructura, se pueden crear tantas instancias como sea necesario.

```
struct variables {
        int a;
        float b;
        char c;
}

int main() {
        variables v;
        printf("Variable a: %d", v.a);
        printf("Variable b: %f", v.b);
        printf("Variable c: %s", v.c);
}
```

Si las variables de la estructura no se inicializan, estas se inicializan con valores por defecto, en caso

de los int con basura, los float con 0.000000 y los char con null.

Matlab

```
disp(a);                    // Error en tiempo de ejecución, la variable “a” no está definida.
disp("hola")                // Imprime en pantalla “hola”

x=5
disp(x);                    // Imprime en pantalla “5”.

a=false
disp(a)                     // Imprime en pantalla “0”

a=”false”
disp(a)                     // Imprime en pantalla “false”
```

- Arreglos:

```
a = [10 20 30 40 50];
disp(a);                    // Imprime en pantalla el contenido del arreglo. 10, 20, 30, 40, 50.

b = [];
disp(b);                    // Imprime en pantalla [ ](0x0)
```

-Matrices:

```
a = [1 2 3; 4 5 6; 7 8 10];
disp(a);                    // Muestra en pantalla:    1, 2, 3
                                                       4, 5, 6
                                                       7, 8, 9

b = [“abc” “def” ; “ghi” “jkl”];
disp(b);                    // Muestra en pantalla:        abc, def
                                                       ghi, jkl

c =[ ; ];
disp(c);                            // Muestra en pantalla [ ](0x0)
```