MANUAL DE PRÁCTICAS PARA LA ASIGNATURA DE PROGRAMACION DE LA UPIBI

ELABORADO POR:

M EN C. SANDRA VÁZQUEZ ROMERO

FEBRERO 2011.**PRÓLOGO**

La programación desde el punto de vista de la ingeniería es una herramienta básica ya que permite optimizar procesos iterativos complejos. Debe de ser una herramienta elemental para los estudiantes que cursan carreras de ingeniería. La impartición de la asignatura “Taller de Programación” en el primer semestre de la carrera, permite que los estudiantes desarrollen la habilidad de diseñar algoritmos numéricos para la solución de problemas planteados en las diferentes especialidades del el área de Ingeniería Biotecnológica.

El objetivo principal de estas prácticas es que sirvan al alumno de la UPIBI, como guía en el desarrollo del taller Teórico- Práctico de Programación, sin pensar que estas prácticas harán que el alumno trabaje de manera independiente del profesor, ya que la modalidad de este tipo de asignatura, es que el profesor imparta la teoría de manera paralela con la practica en el laboratorio de computo.

La estructura de las prácticas está pensando desde las definiciones básicas de los algoritmos numéricos hasta la ejecución de los mismos en el lenguaje de programación de Matlab. Buscando que el alumno pueda crear y desarrollar las habilidades y competencias necesarias para la resolución de problemas de ingeniería en biotecnología.

Este manual de prácticas está compuesto por nueve practicas y éstas podrán llevarse a cabo una o dos practicas por sesión ya que la duración de cada sesión es de tres horas, esto dependerá de la respuesta de aprendizaje cognitivo de los alumnos en cada grupo y de las estrategias de aprendizaje que implemente el profesor para su impartición.

Esperando que el contenido de este manual sea en beneficio de la comunidad de los estudiantes de la UPIBI y en especifico para la academia de Matemáticas Aplicadas. Agradeceré cualquier retroalimentación que me hagan hayan llegar del mismo para enriquecer este material didáctico.

Índice de contenidos

|  |  |
| --- | --- |
| Prologo | 2 |
|  |  |
| Índice | 3 |
|  |  |
| Practica No.1 “Algoritmos numéricos” | 4 |
|  |  |
| Practica No.2 “Algoritmos secuénciales “ | 10 |
|  |  |
| Practica No.3 “Entorno del lenguaje de programación” | 14 |
|  |  |
| Practica No.4 “Operaciones con Escalares y Conceptos Básicos de Programación”. | 17 |
|  |  |
| Practica No. 5 “Tipos de datos y operadores en el lenguaje de programación” | 24 |
|  |  |
| Practica No.6 “Estructuras secuénciales” | 28 |
|  |  |
| Practica No.7 “Estructuras de control” | 36 |
| Practica No.8 “Estructuras de control anidadas” | 41 |
| Practica No.9 “Funciones” | 45 |

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO. I

NOMBRE DE LA PRÁCTICA

Algoritmos Numéricos

1.1 OBJETIVO GENERAL

El alumno desarrollará la habilidad de diseñar algoritmos numéricos para la resolución de problemas de ingeniería en biotecnología.

1.1.1 OBJETIVOS PARTICULARES.

El alumno identificará los pasos a seguir para desarrollar algoritmos numéricos.

El alumno diseñará algoritmos numéricos en sus 3 primeros pasos del método de solución de problemas, en función de la carrera que curse en la UPIBI.

1.2 INTRODUCCION

Algoritmo es un conjunto de acciones que determinan la secuencia de los pasos a seguir para resolver un problema específico, los algoritmos numéricos son aquellos que nos permitan resolver problemas matemáticos

Sus pasos deben estar definidos con precisión de forma que no existan ambigüedades que den origen a elegir una opción equivocada. Los algoritmos son finitos; es decir, su ejecución termina en un número determinado de pasos. La mayoría de los algoritmos de utilidad al programador poseen 3 partes principales:



Los algoritmos pueden representarse a través de un conjunto de palabras por medio de las cuales se puede representar la lógica de un programa. Este conjunto de palabras constituyen lo que se conoce como ***pseudocódigo*.**

**Otra forma de representar los algoritmos pueden ser de manera gráfica, a esta representación se le conoce como *diagramas de flujo*.**

Una vez analizando el problema, posiblemente tengamos varias formas de resolverlo. Lo más importante es determinar cuál es la mejor alternativa para resolver el problema planteado. Esto es la solución que produce los resultados esperados en el menor tiempo y al menor costo.

Los pasos a seguir para dar solución a un problema desde un enfoque computacional son los siguientes:

1. Definición del problema
2. Análisis del problema
3. Selección de la mejor alternativa
4. Crear Diagrama de Flujo
5. Prueba de Escritorio
6. Codificación
7. Compilación.

1.3 MATERIAL

Cuaderno o 5 hojas blancas de papel bond tamaño carta.

Lápiz, ó lapicero

Goma, sacapuntas

Pluma o bolígrafo.

Plantilla de acrílico o cartón para diagramas de flujo.

1.4 DESARROLLO

El alumno seguirá atentamente la explicación que proporcionará el profesor con el apoyo de las notas que se anexan a continuación para que pueda el alumno adquiera la habilidad de resolver problemas de ingeniería con un enfoque computacional.

1.4.1 Representación de algoritmos numéricos

Para llevar a cabo esta práctica es necesario conocer más a fondo los conceptos de Pseudocódigo y Diagramas de Flujo que ya que son las representaciones más usadas para el desarrollo de algoritmos numéricos.

1.4.1.1 Diagramas de flujo

Un diagrama de flujo es una representación gráfica de un algoritmo o de una parte del mismo. La ventaja de utilizar un diagrama de flujo es que se le puede construir ***independientemente del lenguaje de programación****,* pues al momento de llevarlo a código se puede hacer en cualquier lenguaje. Dichos diagramas se construyen utilizando ciertos símbolos de uso especial como son rectángulos, óvalos, pequeños círculos, etc.; como se describen a continuación.

**Símbolos y su Significado**

inicio **Terminal**. Representa el inicio y fin de un programa.

proceso **Proceso**. Son acciones que el programa tiene que realizar

decision **Decisión**. Indica operaciones lógicas o de comparación.

ES **Entrada**. Nos permite ingresar datos.

Mostrar **Salida**. Es usado para indicar salida de resultados

conector **Conector**. Enlaza dos partes cualesquiera de un programa

direccion **Línea de flujo**. Indica dirección de flujo del diagrama. Las flechas de flujo no deben cruzarse. Los diagramas se leen de arriba hacia abajo y de izquierda a derecha.

 **Conector fuera de página**. Representa conexión entre partes del algoritmo representadas en páginas diferentes.

**Recomendaciones para el diseño de Diagramas de Flujo**

Se deben se usar solamente líneas rectas de flujo horizontales y/o verticales.

Se debe evitar el cruce de líneas utilizando los conectores.

Se deben usar conectores solo cuando sea necesario.

No deben quedar líneas de flujo son conectar.

Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.

Todo texto escrito dentro de un símbolo deberá ser escrito claramente, evitando el uso de muchas palabras

**1.4.1.2 Pseudocódigo**

Es combinación del lenguaje de programación y español (ingles o cualquier otro idioma) que se emplea, dentro de la programación estructurada, para realizar el diseño de un programa. El pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos.

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocódigo utiliza palabras que indican el proceso a realizar **y que a continuación se enlista y su función:**

|  |  |
| --- | --- |
| **Palabra** | **Función** |
| **INTRODUCE** | Ingresar valores a través del teclado |
| **ENTONCES** | **Complemento de la selección SI – ENTONCES** |
| **ESCRIBE** | **Visualiza un dato en pantalla** |
| **FIN** | **Finaliza un bloque de instrucciones** |
| **HASTA** | **Cierra la iteración HAZ – HASTA** |
| **HAZ** | **Inicia la iteración HAZ – HASTA** |
| **INICIO** | **Inicia un bloque de instrucciones** |
| **LEER** | **Leer un dato del teclado** |
| **MIENTRAS** | **Inicia la iteración mientras** |
| **NO** | **Niega la condición que le sigue** |
| **O** | **Disyunción lógica** |
| **O – BIEN** | **Complemento opcional de la selección SI – ENTONCES** |
| **PARA** | **Inicia un número fijo de iteraciones** |
| **SI** | **Inicia la selección SI-ENTONCES** |
| **Y** | **Conjunción lógica** |
| **{** | **Inicio de comentario** |
| **}** | **Fin de comentario** |
| **←** | **Asignación** |

***Ventajas de utilizar un Pseudocódigo a un Diagrama de Flujo***

Ocupa menos espacio en una hoja de papel

Permite representar en forma fácil operaciones repetitivas complejas

Es muy fácil pasar de pseudocódigo a un programa en algún lenguaje de programación.

Si se siguen las reglas se puede observar claramente los niveles que tiene cada operación

Sin embargo como se comento anteriormente tanto el diagrama de flujo como el pseudocódigo son solo formar de representar el algoritmo, dependerá del alumno la manera en que se le facilite su representación.

Ejemplo de un ejercicio utilizando la representación grafica (diagrama de flujo) y la representación textual (pseudocódigo).

Un alumno desea saber cuál será su calificación final en la materia de Programación. Dicha calificación se compone de los siguientes porcentajes:

55% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

15% de la calificación de un trabajo final.

Pseudocódigo

Inicio

Leer c1, c2, c3, ef, tf

prom = (c1 + c2 + c3)/3

ppar = prom \* 0.55

pef = ef \* 0.30

ptf = tf \* 0.15

cf = ppar + pef + ptf

Imprimir cf

Fin

Para algunos alumnos desarrollarán mejor actitud hacia el pseudocódigo y otros hacia el diagrama de flujo.

1.4.2 Diseño de algoritmos numéricos

1.4.2.1 Ejercicio propuesto para el diseño del algoritmo numérico.

A continuación el profesor explicará el siguiente ejemplo de algoritmo, el alumno deberá seguir paso a paso al profesor en el desarrollo del mismo.

Un paciente que está sometido a un tratamiento de control de diabetes requiere llevar una bitácora de los niveles de glucosa medidos una vez por día durante una semana. Realice un algoritmo numérico para dar solución a este ejercicio.

Paso 1.- Verificar que el problema este bien definido, es decir que no haga ambigüedades; que cuando el alumno lea el enunciado del problema no existan diferencias en la percepción que se tenga del ejercicio.

Paso 2.- Analizar el problema.

|  |  |
| --- | --- |
| Datos de entrada: | 7 Mediciones de glucosa M1,M2,M3,M4,M5,M6,M7 |
| Datos de salida : | Bitácora de mediciones |
| Proceso: | Como se van almacenar las 7 mediciones y se visualizaran para que el paciente obtenga la bitácora. |

Paso 3.- Elección de la mejor solución.

Propuesta de solución I .-Hacer una base de datos para creación de la bitácora

Propuesta de solución II.- Se realiza un programa iterativo que ingrese las mediciones del paciente y entregue como resultado la bitácora

Propuesta de solución III. Hacer la bitácora de manera manual

Ventajas y desventajas de las propuestas de solución.

Propuesta de solución I.- Ventajas, el paciente tendrá la información de sus mediciones de manera electrónica y disponible en un archivo. Desventajas, la creación de bases de datos requiere más recursos informáticos y por ende más costo.

Propuesta de solución II. Ventajas, el paciente tendrá la información de sus mediciones de manera electrónica y podrá ejecutar el programa las veces que sea necesario según su control, además que requiere menos recursos informáticos que una base de datos. Desventajas, los datos se imprimen y se almacenan en papel.

Propuesta de solución III.- Ventajas, el paciente realizará la bitácora en papel de manera sencilla. Desventaja, cada vez que requiera hacer la bitácora tendrá que estar contando de manera manual.

Desde el punto de vista de Ingeniera en Biotecnología la opción de solución más óptima es la II ya que es más rápida y de menor costo, ya que se podrá aplicar a diferentes pacientes que requieran estar bajo dicho control médico.

El profesor realizará con el alumno por lo menos 2 ejercicios adicionales, que deberán ser entregados como reporte de la práctica.

1.5 Problemas propuestos a desarrollar por los alumnos:

I.- Una muestra de 20 ratas son sometidas a una dieta rica en carbohidratos y se requiere medir su peso, y obtener un promedio del peso de las ratas de dicha muestra elaborar un algoritmo para dar solución a ese ejercicio.

II.- Un grupo de 10 niños tiene problemas de discapacidad, por lo que es necesario diseñar prótesis para el pie izquierdo, diseñe un algoritmo para poder diseñar dichas prótesis.

Los ejercicios deberán ser resueltos con los 3 primeros pasos de la solución de problemas y anexarlos en el reporte de la práctica.

1.6 CONCLUSIONES.

Desarrolle sus concusiones con respecto a la solución de problemas de ingeniería aplicando algoritmos numéricos y el diseño de los mismos con un enfoque computacional.

1.7 EVALUACIÓN.- La práctica tiene un porcentaje de evaluación del 30%, mismo que se promediará con el número de prácticas realizadas durante el parcial y que deberá corresponder al 30% de la calificación por parcial.

1.8 BIBLIOGRAFIA.

Fundamentos de Programación, Algoritmos, Estructuras de datos y Objetos; Luis Joyanes Aguilar Mc Graw Hill

Algoritmos Fundamentales[Knuth, D.E.](http://www.lsf.com.ar/resulta.aspx?key=KNUTH,%20D.E.&criterio=ae)Prentice Hall.

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.2

NOMBRE DE LA PRÁCTICA

Algoritmos Secuénciales

2.1 OBJETIVO GENERAL.

El alumno desarrollará la habilidad para diseñar algoritmos secuenciales.

2.1.1 OBJETIVOS PARTICULARES.

El alumno desarrollará la habilidad de describir la secuencia lógica de un algoritmo con un enfoque computacional.

El alumno conocerá las reglas para cambiar formulas matemáticas a expresiones validas para la computadora, además de diferenciar constantes y variables desde el punto de vista computacional.

El alumno a través de la elección que utilice para la representación del algoritmo, identificará y empleará las palabras o símbolos requeridos para el diseño del algoritmo.

2.2 INTRODUCCION

Actualmente la computadora no solamente es una maquina que puede realizar procesos para darnos resultados sin que tengamos la noción exacta de las operaciones que realiza para llegar a esos resultados. Con la computadora además de lo anterior también podemos diseñar soluciones a la medida de problemas específicos que se nos presenten. Más aun, si estos involucran operaciones matemáticas complejas y/o repetitivas, o requieren del manejo de un volumen muy grande de datos.

El diseño de soluciones a la medida de nuestros problemas, requiere como en otras disciplinas una metodología que nos enseñe de manera gradual, la forma de llegar a estas soluciones que son precisamente los algoritmos.

A las soluciones creadas por computadora se les conoce como ***programas*** y no son más que una serie de operaciones que realiza la computadora para llegar a un resultado, con un grupo de datos específicos. Lo anterior nos lleva al razonamiento de que un programanos sirve para solucionar un problema específico.

Para poder realizar programas, además de conocer la metodología mencionada, también debemos de conocer, de manera específica las funciones que pueden realizar la computadora y las formas en que se pueden manejar los elementos que hay en la misma, es decir la forma en que debemos de traducir nuestros algoritmos en instrucciones reconocidas por la computadora.

2.3 MATERIAL:

Plantilla acrílico o cartón de símbolos de diagramas de flujo

Cuaderno o 5 hojas blancas de papel bond

Lápiz, ó lapicero

Goma, sacapuntas

Pluma o bolígrafo.

2.4 DESARROLLO

Para llevar a cabo un algoritmo numérico secuencial es necesario definir que es una estructura secuencial.

El alumno seguirá paso a paso la explicación del profesor y podrá hacer uso de las siguientes notas para ampliar sus conocimientos teóricos sobre los algoritmos secuenciales.

2.4.1 Estructuras Secuenciales

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

Una estructura secuencial se representa de la siguiente forma como pseudocódigo:

Inicio

Accion1

Accion2

.

.

AccionN

Fin

Para llevar a cabo un algoritmo secuencial es necesario tener bien definidos los conceptos de: constante, variable y asignación, ya que estos conceptos pueden variar en desde la perspectiva computacional a la matemática y debe de considerarse a la hora de diseñar el algoritmo.

* ***Constante:*** Una constante es un dato numérico o alfanumérico que no cambia durante la ejecución del programa.

Ejemplo:

pi = 3.1416

* ***Variable:*** Es un espacio en la memoria de la computadora que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambiar durante la ejecución del programa. Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con el cual podamos identificarla dentro de un algoritmo

Ejemplo:

X=1

X=X+1

* ***Identificadores.*** Los *identificadores* representan los datos de un programa (constantes, variables, tipos de datos). Un identificador es una secuencia de caracteres que sirve para identificar una posición en la memoria de la computadora, que nos permite accesar a su contenido.

Ejemplo: suma

Promedio\_1

cal1,cal2,etc

A continuación se describen las reglas que se han de emplear para crear identificadores.

***Reglas para formar un Identificador en la mayoría de los lenguajes de programación.***

* Debe comenzar con una letra (A a Z, mayúsculas o minúsculas) y no deben contener espacios en blanco.
* Letras, dígitos y caracteres como el guión bajo ( \_ ) están permitidos después del primer carácter.
* La longitud de identificadores puede ser de hasta 8 caracteres

En la siguiente práctica se profundizará más en estos conceptos. .

2.4.2 Ejercicio propuesto de algoritmo secuencial que contenga hasta el 5 paso de la solución de un problema.

El profesor explicará el ejercicio propuesto, y los alumnos lo retroalimentarán con sus participaciones buscando obtener una solución optima para el desarrollo del algoritmo secuencial.

|  |  |
| --- | --- |
| Paso 1.- Planteamiento de Problema  Realiza la suma de dos temperaturas tomadas a un paciente en un día. | Paso 2.- Análisis del Problema  DE.- t1, t2  DS.- suma  Proceso  suma ←t1 +t2 |
| Paso 3.- Pseudocódigo  Inicio  Introduce valores de t1 y t2  Suma←t1+t2  Escribe suma  Fin. | Paso 4.-Diagramas de Flujo |
| Paso 5.-Prueba de Escritorio  t1 = 37  t2= 39  Suma =37+39  76 |  |

El profesor desarrollará de manera conjunta con los alumnos un algoritmo secuencial propuesto por él según sea la especialidad del grupo asignado.

2.5 Ejercicios propuestos a resolver por el alumno que deberán ser entregados en el reporte de la práctica.

I.- Diseñe un algoritmo secuencial que calcule la edad biológica de una persona.

II.- La presión, el volumen y la temperatura de una masa de aire se relacionan por la formula diseñe un algoritmo secuencial para la solución de esta ecuación:

masa = (presión \* volumen)/ (0.37 \* (temperatura + 460))

2.6 CONCLUSIONES.

Describa la importancia del uso de algoritmos secuenciales para la solución de problemas de ingeniería en su especialidad

2.7 BIBLIOGRAFIA.

Introducción a la Computación para Ingenieros, Steven C. Chapa, Raymond P. Canale, Editorial Mc Graw Hill

Técnicas De Programación GarcíaEditorial: [Mc Graw Hill](http://www.buscalibros.cl/libros-editorial-mcgraw-hill-id_97897010.htm)

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.3

NOMBRE DE LA PRÁCTICA

Entorno del lenguaje de programación

3.1 OBJETIVO GENERAL

El alumno conocerá y se relacionará con el entorno del lenguaje de programación matlab para la implementación de algoritmos numéricos aplicados a la ingeniería biotecnológica.

3.1.1 OBJETIVOS PARTICULARES.

El alumno conocerá y se relacionará con el uso de las diversas ventanas en que opera matlab.

El alumno ejecutará comandos básicos dentro de las ventanas de matlab para guardar y configurar los dispositivos y reconocer su uso.

3.2 INTRODUCCION

Matlab significa MATrix LABoratory, es un lenguaje de programación (inicialmente escrito en C) para realizar cálculos numéricos con **vectores y matrices**. Sin embargo dado su ambiente de ventanas de fácil manejo permite que usuarios con poco conocimiento en lenguajes de programación puedan realizar programas que trabajen con números escalares, tanto reales como complejos, facilitando su uso para aplicaciones de ingeniería. Cuenta con paquetes de funciones especializadas que permite desarrollar programas complejos sin profundizarse con conceptos de lenguajes de programación de bajo nivel o intermedios.

3.3. MATERIAL

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

3.4 DESARROLLO

3.4.1 Pulse en el botón de inicio de su PC y seleccione el programa de matlab aparecerá la siguiente pantalla

En esta pantalla se describen los elementos básicos del escritorio de Matlab los cuales son:

**Command Windows**: Es la ventana donde seejecutan todas las instrucciones y programas. Se escribe la instrucción o el nombre del programa y se da Enter.

**Command History**: En esta ventana se muestra los últimos comandos ejecutados en Command Windows. Se puede recuperar el comando haciendo doble click

**Current directory**: Situarse en el directorio donde se va a trabajar

**Help** (también se puede usar desde comand windows)

**Workspace**: Para ver las variables que se están usando y sus dimensiones (si son matrices)

**Editor del Matlab**: Todos los ficheros de comandos Matlab deben de llevar la extensión .m

3.4.2 Posiciónese en la ventana de Commands, deberá aparecer el prompt “ >>” en esta ventana podrá realizar la asignación de valores a diferentes variables y en la ventana de Workspace se visualizará el tipo de variable y su contenido, por ejemplo realice las siguientes operaciones:

T=20

t= 5

x=6 \*5+(10-7)

Observe que para matlab T y t son variables con datos diferentes los que significa que es sensible a las mayúsculas y que “x” contendrá el resultado de la operación aritmética. Es así como en matlab se introduce valores a variables.

Realice las siguientes operaciones:

a)H=(T\*x)/t

b)p=H\*H

c)b=T/p +H

Describe que sucedió en cada operación.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.4.3 A continuación posiciónate en la ventana de comandos para que puedas ejecutar los siguientes comandos y aprendas su uso. Todos los comandos deberán escribirse con minúsculas, para que puedan ser reconocidos por matlab como comandos.

* **El comando who**  da información sobre variables que se están usando y sus dimensiones (si son matrices) y se visualiza en la ventana Workspace.

Teclea el comando who

>who ←

Observa si solo están las variables que ejecutamos anteriormente, en caso de que haya más de 6 variables significa que otro usuario dejo en memoria de matlab el uso de más variables por lo que puede ocasionar que alguna operación aritmética no entregue un resultado correcto.

* Comando clear permite borrar todas la variable en uso y su contenido.

>clear ←

Observa que ya no existen variables en uso, todas fueron borradas.

* Comando clc permite limpiar los comandos en uso y llevar el prompt al inicio de la pantalla, pero no borra variables.

>clc

* Comando cd permite cambiar de directorio o unidad de trabajo.

>cd .. ←

>cd a:←

* Comando diary on crea un archivo donde se hace una bitácora de comando u operaciones realizadas en la ventana de comandos.

>diary on a:\sesion1 guarda la bitácora en el archivo sesión1 en la unidad a

>diary off cierra el archivo de la sesion1

Ejemplo de constantes características en matlab.

pi=π, NaN(not a number, 0/0), Inf=**∞**.

>pi←

>NaN←

>Inf←

>i←

>j←

3.5 Ejercicios propuestos para los alumnos:

I.- Crea un archivo denominado trabajo1 en la unidad de trabajo externa (USB) que contenga la información de la bitácora de las siguientes operaciones:

pi

h=4\*pi

y=67\*5+(h^2)+pi

Verifique con el comando who cuantas variables hay en la ventana de trabajo, ejecute el comando clc y describa que ocurre\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, cambie la unidad de trabajo (de C a la USB), ejecute el comando clear y describa lo que sucede\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ y por último cierre el archivo trabajo1.

3.6 CONCLUSIONES.

Explica ¿Que es matlab?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

¿En tu especialidad para que te puede servir aprender matlab?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Proporciona 5 aplicaciones en tu especialidad que requieran realizar algoritmos numéricos en matlab.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.7 BIBLIOGRAFIA.

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab,Delores M Etter,Pearson.

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.4

NOMBRE DE LA PRÁCTICA

Operaciones con Escalares y Conceptos Básicos de Programación

4.1 OBJETIVO GENERAL

El alumno conocerá el funcionamiento y las reglas para cambiar formulas, constantes e identificadores y tipos de datos desde el lenguaje convencional matemático hasta el lenguaje computacional.

4.1.1 OBJETIVOS PARTICULARES.

El alumno conocerá y aprenderá a traducir conceptos básicos del lenguaje matemático a un lenguaje computacional.

El alumno conocerá e interpretará el uso de operadores matemáticos y lógicos en expresiones numéricas.

4.2 INTRODUCCION

Cualquier cálculo matemático por lo general se especifica como una combinación de constantes numéricas y variables que pueden ser de distintos tipos de datos como son escalares, vectores o matrices, utilizando operadores de tipo aritméticos básicos como son los siguientes:

+ Suma

- Resta

\* Multiplicación

/ División

Con la combinación de estos y el uso de constantes, y asignación de variables se pueden llevar a cabo operaciones aritméticas que permitan resolver problemas de ingeniería. A continuación se describen los tipos de datos así como su uso dentro del lenguaje de programación.

4.3 MATERIAL

Cuaderno o 5 hojas blancas de papel bond tamaño carta.

Lápiz, ó lapicero

Goma, sacapuntas

Pluma o bolígrafo

4.4 DESARROLLO

Para diseñar un algoritmo que permita resolver problemas que vayan de lo simple a lo complejo es indispensable conocer los tipos de datos y su forma de operar, así como la forma en que lo interpretará la computadora para que realice de manera adecuada y optima y entregue los resultados de la ejecución del algoritmo planteado.

El profesor dará a conocer a los estudiantes los tipos de datos que han de traducirse de un lenguaje matemático a un lenguaje computacional, permitiendo con ello que el alumno se familiarice con el uso de conceptos básicos en el lenguaje de programación.

4.4.1 Tipos De Datos

El alumno atenderá la explicación proporcionada por el profesor y realizará las notas correspondientes según lo crea necesario.

A la mayoría de los datos se les asigna un “tipo” que dependerá de la información que contendrá. Un dato puede ser un simple carácter, tal como ‘z’, un valor real como 94.5. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable, a continuación se muestran los tipos de datos desde el punto de vista computacional.

Numéricos

Simples Lógicos

Alfanuméricos (string)

Tipos de

datos Arreglos (Vectores, Matrices)

Estructurados Registros

(Def. por el Archivos

usuario) Apuntadores

Para la presente práctica solo consideraremos los datos simples

Datos Numéricos***:*** Permiten representar valores ***escalares***de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.

Datos Lógicos***:*** Son aquellos que solo pueden tener dos valores (cierto o falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).

Datos Alfanuméricos (String o de cadena): Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es permite representar números como cadenas alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

Ejemplo:

“nombre del alumno”

“edad”

Una vez que conocemos estos tipos de datos, pueden ser empleados dentro de expresiones. Las expresiones son combinaciones de constantes, variables, símbolos de operaciones, paréntesis y nombres de funciones especiales. Por ejemplo:

Z= a+(b + 3)/c

Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

Una expresión consta de operadores, operandos y asignaciones. Según sea el tipo de datos que manipulan, se clasifican las expresiones en: Aritméticas, Relaciónales y Lógicas. Los operadores son elementos que relacionan de forma diferente, los valores de una o más variables y/o constantes. Es decir, los operadores nos permiten manipular valores.

Aritméticos

Tipos de Operadores Relaciónales

Lógicos

***Los operadores aritméticos:*** permiten la realización de operaciones matemáticas con los valores (variables y constantes). Los operadores aritméticos pueden ser utilizados con tipos de datos enteros o reales.

***Operadores Aritméticos***

+ Suma

- Resta

\* Multiplicación

/ División

mod Modulo (residuo de la división entera)

Ejemplos:

Expresión Resultado

7 / 2 3.5

12 mod 7 5

4 + 2 \* 5 14

Los operadores aritméticos tienen prioridades entre sí, así como las expresiones matemáticas que cuentan con paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera, el paréntesis más interno se evalúa primero.

Prioridad de los operadores dentro de una misma expresión:

1.- ^ Exponenciación

2.- \*, /, mod Multiplicación, división, modulo.

3.- +, - Suma y resta.

Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

El alumno realizara las siguientes operaciones sin el uso de calculadora, con el fin de comprobar la prioridad de los operadores.

Ejemplos:

3 + 5 \* (10 - (2 + 4)) = 23 3 + 5 \* (10 - 6) = 3 + 5 \* 4 = 3 + 20 = 23

3.5 + 5.09 - 14.0 / 40 = 5.09 3.5 + 5.09 - 3.5 = 8.59 - 3.5 = 5.09

* 1. \* (1.5 + 3.0 \* 4.1) = 28.98 2.1 \* (1.5 + 12.3) = 2.1 \* 13.8 = 28.98

***Los Operadores Relaciónales:*** Se utilizan para establecer una relación entre dos valores. Compara estos valores entre sí y esta comparación produce un resultado de certeza o falsedad (verdadero o falso). , comparan valores del mismo tipo (numéricos o cadenas), los operadores relaciónales tienen menor prioridad que los aritméticos.

***Operadores Relaciónales***

> Mayor que

< Menor que

> = Mayor o igual que

< = Menor o igual que

< > Diferente

= Igual

El alumno realizará las siguientes operaciones con los operadores relacionales y comprobará el resultado obtenido a continuación.

Ejemplos:

Si a = 5 b = 10 c = 30

a + b > c Falso

a - b < c Verdadero

a - b = c Falso

a \* b < > c Verdadero

***Los Operadores Lógicos:*** Estos operadores se utilizan para establecer relaciones entre valores lógicos, pueden ser resultado de una expresión relacional.

***Operadores Lógicos***

And Y

Or O

Not Negación

***Tabla de verdad del*** ***Operador And***

*Operando1 Operador Operando2 Resultado*

V AND *V* V

*F* V F

F V F

F F F

***Tabla de verdad del Operador Or***

*Operando1 Operador Operando2 Resultado*

V OR V V

V F V

F V V

F F F

***Tabla de verdad del Operador Not***

*Operando Resultado*

V F

F V

***Prioridad de los Operadores Lógicos***

Not

And

Or

***Prioridad de los Operadores en General***

1.- ( )

2.- ^

3.- \*, /, Mod, Not

4.- +, -, And

5.- >, <, > =, < =, < >, =, Or

El alumno comprobará el uso de los operadores lógico con los siguientes ejercicios.

***Ejemplos:***

a = 10 b = 12 c = 13 d =10

**1)** ((a > b)or(a < c)) and ((a = c) or (a > = b))

F V F F

V F

F

**2)** ((a > = b) or (a < d)) and (( a > = d) and (c > d))

F F V V

F V

F

4.4.2 Existen otros conceptos básicos que debe conocer el alumno y este es el concepto de las ***“Variable***” este se pueden clasificar en función de su contenido o del uso que se les dé, a continuación se describen los tipos de variables.

Numéricas

Por su Contenido Lógicas

Alfanuméricas (String)

Variables

De Trabajo

Por su Uso Contadores

Acumuladores

**Por su Contenido**

***Variable Numéricas:*** Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal.

Ejemplo:

iva=0.15 costo=2500

***Variables Lógicas:*** Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.

***Variables Alfanuméricas:*** Esta formada por caracteres alfanuméricos (letras, números y caracteres especiales).

Ejemplo:

letra=’a’ apellido=’lopez’ direccion=’Av. Libertad #190’

***Por su Uso***

***Variables de Trabajo:*** Variables que reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un programa.

Ejemplo:

suma=a+b/c

***Contadores:*** Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con incrementos generalmente de uno en uno.

Ejemplo:

C= C+ 1

***Acumuladores:*** Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

Ejemplo:

C= C+ SUMA

***Asignación:*** La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

* ***Simples:*** Consiste en pasar un valor constate a una variable (a=15)
* ***Contador:*** Consiste en usarla como un verificador del número de veces que se realiza un proceso (a=a+1)
* ***Acumulador:*** Consiste en usarla como un sumador en un proceso (a=a+b)
* ***De trabajo:*** Donde puede recibir el resultado de una operación matemática que involucre muchas variables (a=c+b\*2/4).

***- Lectura:*** La lectura consiste en recibir desde un dispositivo de entrada (p.ej. el teclado) un valor. Esta operación se representa en un pseudocódigo como sigue:

Leer a, b

Donde “a” y “b” son las variables que recibirán los valores

***Escritura:*** Consiste en mandar por un dispositivo de salida (p.ej. monitor o impresora) un resultado o mensaje. Este proceso se representa en un pseudocódigo como sigue:

Escribe “El resultado es:”, R

Donde “El resultado es:” es un mensaje que se desea aparezca y R es una variable que contiene un valor.

El profesor proporcionará al alumno una serie de ejercicios donde diseñe algoritmos secuenciales que contengan asignación de variables, constantes uso de operadores aritméticos, de relación y lógicos.

4.5. Ejercicios propuestos para solución de los alumnos.

I Realice las siguientes operaciones con los operadores relacionales, lógicos y aritméticos:

X=19 Y=5 Z=3

X>Y<Z X and Y or Z

Z<Y<X Z and Yand Z

X<Z<Y X or Z or Y

X>Z realice su negación.

II.- a=35 b=4 c=2

S=(a+c(b/a)(c\*c\*a)/b)\*b

F=(a\*c)+((c+a/b)\*a)/c

4.6 CONCLUSIONES.

El alumno describirá la importancia, su diferencia y la aplicaciones del uso de escalares, variables, identificadores y la prioridad de los operadores.

4.7 BIBLIOGRAFIA.

Programación lógica Pascual Julián Iranzo / María Alpuente Frasnedo Editorial Pearson

Técnicas De Programación GarcíaEditorial: [Mc Graw Hill](http://www.buscalibros.cl/libros-editorial-mcgraw-hill-id_97897010.htm)

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.5

NOMBRE DE LA PRÁCTICA

Tipos de datos y operadores en el lenguaje de programación

5.1 OBJETIVO GENERAL

El alumno conocerá e interpretará el tipo de datos, sus operadores y las instrucciones en el lenguaje de programación de matlab.

5.1.1 OBJETIVOS PARTICULARES.

El alumno conocerá y ejecutará en la ventana de comandos de matlab los diferentes tipos de datos que utiliza matlab.

El alumno conocerá y ejecutará en la ventana de comando los operadores matemáticos y lógicos que utiliza matlab.

5.2 INTRODUCCION

Matlab permite que el usuario de manera intuitiva aprenda a programar ya que utiliza instrucciones básicas en el idioma ingles que permite que a través de la ventana de comando o bien dentro del editor el usuario cree sus propios programas de manera fácil, a continuación se describirá el uso de la ventana de comando y una lista de instrucciones básica.

5.3 MATERIAL

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Plantilla de acrílico o cartón para diagramas de flujo.

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

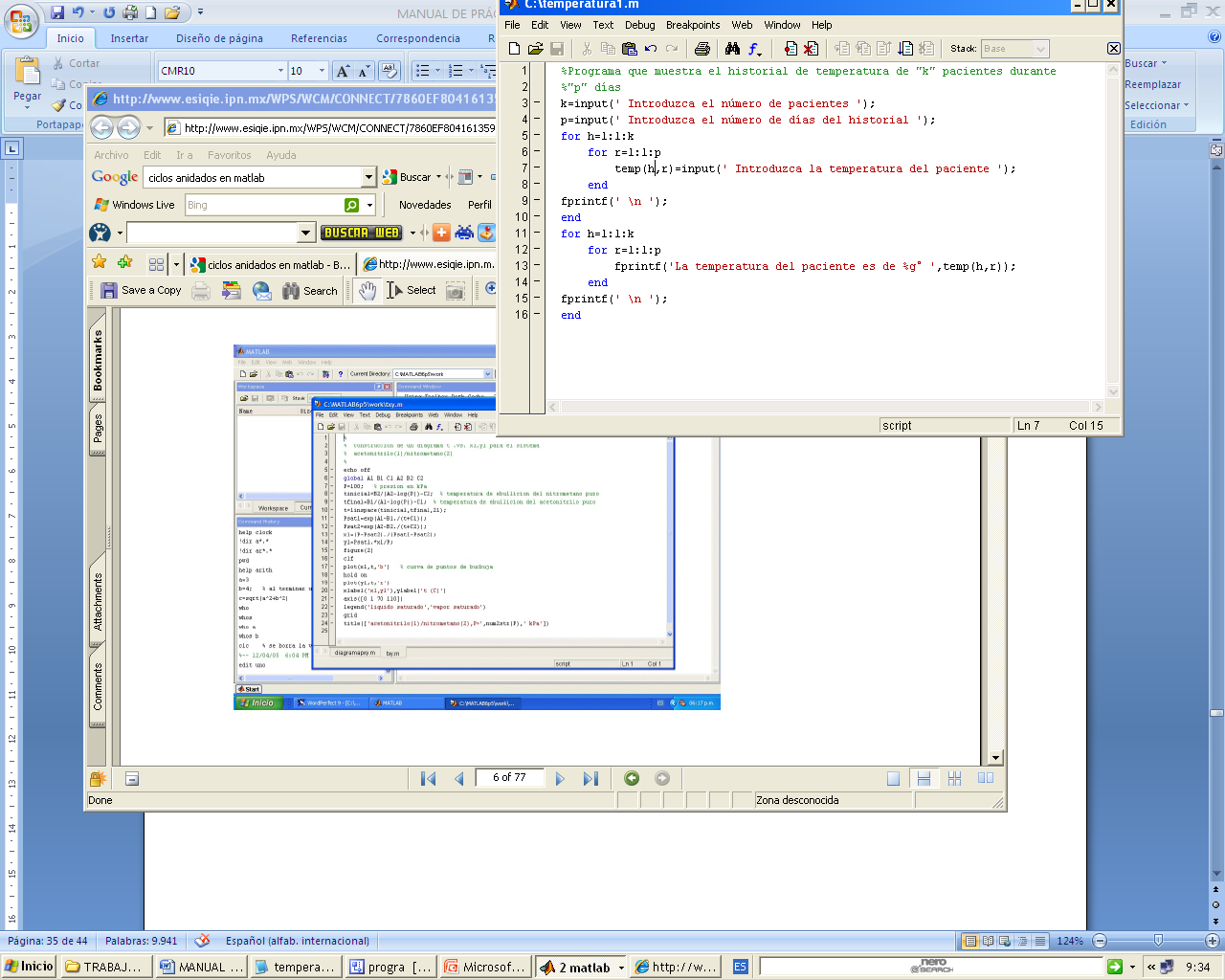
Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

5.4 DESARROLLO

El alumno escribirá y comprobará en el editor de matlab los comando que a continuación explicará el profesor y guardará en su medio de almacenamiento externo (USB) los programas realizado.

El editor permite tanto crear y modificar archivos con extensión \*.m, así como ejecutarlos paso a paso para ver si contienen errores (proceso de Debug o depuración, eliminar errores al programa). También Matlab permite que utilicemos cualquier editor (edit de DOS, Word, Notepad, etc.), ya que los archivos son sólo de texto. El siguiente gráfico muestra la ventana principal del Editor/Debugger.



El editor muestra con diferentes colores los diferentes tipos o elementos constitutivos de los comandos. (en verde los comentarios, en rojo las cadenas de caracteres, etc.). El editor además indica que las comillas o paréntesis que se abren se cierren correctamente.

A continuación se describen las instrucciones y sentencias básicas de matlab.

***Instrucciones de entrada y salida***:

* i**nput**: Esta función permite imprimir un mensaje en la línea de comandos de Matlab y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario.

Ejemplo: n = input('Ingrese un número: ');

* **disp**: Esta función permite imprimir en pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre. En realidad, disp siempre imprime vectores y/o matrices: las cadenas de caracteres son un caso particular de vectores.

Ejemplo: disp('El programa ha terminado');

***Sentencia If: condición Simple***

En su forma más simple, la sentencia if se describe como sigue:

if condicion

sentencias

end

Observe que la condición no va encerrada entre paréntesis.

Existe también la "bifurcación múltiple" o Condición Múltiple, en la que pueden concatenarse tantas condiciones como se desee, y que tiene la forma:

* if condicion1

bloque1

elseif condicion2

bloque 2

elseif condicion3

bloque 3

else % opción por defecto para cuando no se cumplan las otras condiciones.

bloque 4

end

NOTA: El uso de else es opcional. Para crear condiciones debemos conocer tanto los operadores relacionales como los operadores lógicos.

Operadores relacionales Operadores lógicos

Igualdad == and & (ampersand)

Desigualdad ~ = or | (pipe)

menor < not ~ (tilde)

mayor > xor o exclusivo

menor o igual <= any verdadero si algún elemento del vector es verdadero

mayor o igual >= todos son verdaderos

Ejemplo de uso de if:

nro = input('Ingrese un número positivo: ');

if mod(nro,2) == 0

disp('es par');

else

disp('es impar');

end

Ejemplo de uso de if/elseif:

nro = input('Ingrese un número positivo: ');

if nro==1

disp('Ud. ingreso 1');

elseif nro==2

disp('Ud. ingreso 2');

elseif nro==3 d

isp('Ud. ingreso 3');

else

disp('El número es mayor que 3');

end

***Sentencia for: Ciclo Fijo***

La sentencia for es un ciclo fijo que permite repite un conjunto de sentencias un número predeterminado de veces con incrementos por default de 1.

La siguiente construcción ejecuta sentencias con valores de i de 1 a n, variando en uno:

for i=1:n

sentencias

end

***For anidados: Ciclos Anidados Fijos***

for i=1:n

for j=1:m

sentencias

end

end

Ejemplos de uso de for:

nro = input('Ingrese un número positivo: ');

for i=1:nro

disp(i)

end

***Sentencia While: Ciclo Condicionado***

La sintaxis de la estructura while es la siguiente.

while condicion

sentencias

end

donde la condición puede ser una expresión vectorial o matricial. Las sentencias se siguen ejecutando mientras haya elementos distintos de cero en la condición, es decir, mientras haya algún o algunos elementos verdaderos. El bucle termina cuando todos los elementos de condición son falsos (es decir, cero).

Ejemplo de uso de while:

nro = input('Ingrese un número positivo: ');

i=0;

while i <= nro

disp(i);

i=i+1;

end;

5.5 Ejercicios propuestos para el alumno.

Deberá codificar los algoritmos realizados en prácticas anteriores y guardarlos en archivos con extensión .m para ser los ejecutados en la ventana de comandos de Matlab, así mismo deberán ser entregados con el reporte de la práctica.

5.6 CONCLUSIONES.

El alumno describirá el número de pasos que ha desarrollado en la solución de problemas, e identificará cada uno de ellos.

5.7 BIBLIOGRAFIA.

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab, Delores M Etter, Pearson.

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.6

NOMBRE DE LA PRÁCTICA.

Estructuras secuénciales

6.1 OBJETIVO GENERAL

El alumno desarrollará la habilidad para diseñar algoritmos secuenciales, condicionados simples, condicionados múltiples

6.1.1OBJETIVOS PARTICULARES.

El alumno desarrollará e implementará los algoritmos secuenciales en el entorno de programación en matlab empleado estructuras condicionadas simples y múltiples.

6.2 INTRODUCCION

El teorema del programa estructurado, demostrado por Böhm-Jacopini, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

Secuencia

Instrucción condicional.

Iteración (bucle de instrucciones) con condición al principio.

Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, éstas pueden ser construidas mediante las tres básicas.

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

La estructura selectiva permite la realización de una instrucción u otra según un criterio, solo una de estas instrucciones se ejecutara. Esta instrucción selectiva puede presentar dos mensajes, uno a es mayor que b, y el otro a no es mayor que b, solo uno de ellos será presentado, según el resultado de la comparación de a y b, si el resultado de a > b es cierto, se presenta el primer mensaje, si es falso el segundo, las palabras IF, THEN, ELSE, END IF; son propias de la instrucción (palabra reservadas) que tienen un significado en el lenguaje, sirven de separadores, y el usuario no debe utilizarlas salvo para este fin.

**IF** señala el comienzo de la instrucción condicional, y se espera que después esté la condición de control de la instrucción.

**THEN** señala el fin de la condición, y después estará la instrucción a realizar si la condición es cierta.

**ELSE** separa la instrucción que se ejecutará **si la condición es cierta** de la que se ejecutará **si es falsa**.

**END IF** indica que la instrucción condicional finaliza y el programa seguirá su curso.

Las estructuras iterativas se analizarán en prácticas posteriores, para la presente practica solo estudiaremos la estructura secuencial condicionada simple, anidada y múltiple.

### 6.3 MATERIAL

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Plantilla de acrílico o cartón para diagramas de flujo.

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

6.4 DESARROLLO

El alumno realizará notas sobre el uso las sentencias secuenciales condicionadas explicadas por el profesor.

Seguirá paso a paso las instrucciones del profesor y ejecutara las sentencias en el editor de texto de matlab así como su ejecución en la ventana de comandos

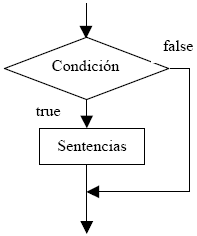
.

6.4.1 Forma Condicional Simple

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen dos tipos básicos, las simples y las múltiples.

* ***Simples:*** Las estructuras condicionales simples se les conoce como “Tomas de decisión”. Estas tomas de decisión tienen la siguiente forma:

***Pseudocódigo***

 Si <condición> entonces

Acción(es)

Fin-si

6.4.2Forma condicional doble***:***  Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición.

Se representa de la siguiente forma:

***Condicionada doble***

***Pseudocódigo***

Si <condición> entonces

Acción(es)

si no

Acción(es)

Fin-si

Donde:

Si ………………… Indica el comando de comparación

Condición………… Indica la condición a evaluar

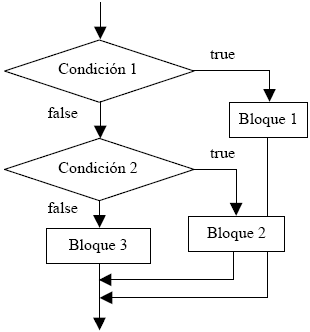
entonces……..…… Precede a las acciones a realizar cuando se cumple la condición

acción(es)………… Son las acciones a realizar cuando se cumple o no la condición

si no……………… Precede a las acciones a realizar cuando no se cumple la condición

fin

Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.

***6.4.3 Múltiples:*** Las estructuras de comparación múltiples, son tomas de decisión especializadas que permiten comparar una variable contra distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma común es la siguiente:

***Condición Múltiple***

Si <condición> entonces

Acción(es)

si no

Si <condición> entonces

Acción(es)

si no

.

. Varias condiciones

Fin de si.

El alumno seguirá paso a paso el desarrollo del diseño de los siguientes algoritmos y deberá entregarlos en el reporte de la práctica.

Ejercicio No. 1

|  |  |  |
| --- | --- | --- |
| Planteamiento de Problema  Indicar si valor del voltaje de una señal del electrocardiograma de un paciente un número es positivo o negativo. | Diagramas de Flujo | Codificación en Matlab  L= input('Escribe el valor del voltaje ');  if L >= 0  fprintf ('"positivo"');  else  fprintf ('"negativo"');  end |
| Análisis del Problema  DE.- L{voltaje de la señal}  DS.- “POSITIVO” , “NEGATIVO”  Proceso  Si L ≥ 0 entonces  L “positivo”  Sino  L “negativo”  Fin | Prueba de Escritorio  L = 5  5 ≥ 0 Si  “positivo”  L = -3  -3 ≥ 0 No  “negativo” | Ejecución  >> E7  Escribe el valor del voltaje 5  "positivo"  >> E7  Escribe el valor del voltaje -3  "negativo"  >> |

Ejercicio No.2

|  |  |  |
| --- | --- | --- |
| Planteamiento de Problema  Se necesita determinar si el peso de una rata es múltiplo de 2 y 5 para determinar el tipo de dieta especial que hay que proporcionarle. | Diagramas de Flujo  no si | Codificación  n=input('Escribe el peso de la rata');  fprintf ('\t\n');  m = rem(n,2);  if m == 0;  r = rem(n,5);  if r == 0  fprintf('"Es múltiplo de 2 y 5')  end  end |
| Análisis del Problema  DE.- N{peso de rata}  DS.- “múltiplo de 2 y 5”  Proceso  M ←Res (n/2)  Si M = 0 entonces  Si ← Res (N/5)  “múltiplo de 2 y 5”  Fin  Fin | Prueba de Escritorio  N = 10  M =Res (10/2)  0 = 0 Si  R = Res (10/5)  0 = 0  “múltiplo de 2 y 5”  N = 6  M = 0 Res (6/2)  0 = 0 Si  R = Res (6/5)  0 ≠ 1.2 No | Ejecución  >> E10  Escribe el peso de la rata 10    "Es múltiplo de 2 y 5>> |

Ejercicio No.3

|  |  |  |
| --- | --- | --- |
| Planteamiento de Problema  Dado 3 lecturas numéricas realizadas a un conejo en el laboratorio verificar que sean enteros y positivos,  mostrar cual es el mayor de las lecturas. | Diagramas de Flujo    Prueba de escritorio  l=5  m=2  n=6  a=res(5/1)=0  a=0 y 5>=0? Si entonces  b=res(2/1)=0  b=0 y 2>=0 ? si entonces  c=res(6/1)=0  c=0 y 6 >=0 ?si entonces  5>=2? Si entonces  5>=6 no  6 es mayor  Fin  l=4  m=5  n=6  a=res(4/1)=0  a=0 y 4>=0? Si entonces  b=res(5/1)=0  b=0 y 5>=0 ? si entonces  c=res(6/1)=0  c=0 y 6 >=0 ?si entonces  4>=5? no  5>=6 no  6 es mayor  fin | Codificación  l = input('Primer número a verificar');  fprintf ('\t\n');  m = input('Segundo número a verificar');  fprintf ('\t\n');  n = input('Tercer número a verificar');  fprintf ('\t\n');  a = rem(l,1);  if a == 0;  l >= 0;  b = rem (m,1);  if b == 0;  m >= 0;  c = rem (n,1);  if c == 0;  m >= 0;  if l > m;  if l > n;  fprintf ('"Primer número mayor"');  else  fprintf('"Tercer número mayor"');  end  else  if m > n;  fprintf('"Segundo número mayor"');  else  fprintf('"Tercer número mayor"');  end  end  end  end  end |
| Análisis del Problema  DE.- L, M, N.  DS.- “n, mayor”  “m, mayor”  “l, mayor”  Proceso  a ←Res (l/1)  Si a = 0 y l ≥ 0 entonces  b ←Res (m/1)  Si b = 0 y  m ≥ 0 entonces  c ←Res (n/1)  Si l > m entonces  Si l > n entonces  l ”mayor”  sino  n “mayor”  Fin  Sino  Si m> h  entonces  m” mayor”  Sino  h “mayor”  Fin  Fin  Fin  Fin | Ejecución  >> E13  Primer número a verificar 4    Segundo número a verificar 5    Tercer número a verificar 6    "Tercer número mayor">> |

* 1. Ejercicios propuestos para realización del alumno.

I.- Calcular la diferencia positiva de dos mediciones realizadas a un paciente con glucosa.

II.- Obtener el promedio de 3 calificaciones parciales del 0 al 10, si el promedio es mayor o igual que 6 mostrar "aprobado" sino "reprobado.

III.- Dada la medición de temperatura de un paciente indicar:

Si es mayor que 37 "fiebre"

Si es mayor que 41 "Convulsión".

Si es menor que 35 "hipotermia”.

Si es igual a 37 "temperatura Adecuada".

6.6 CONCLUSIONES.

El alumno explicará en sus propias palabras la importancia de emplear estructuras secuenciales condicionadas en el diseño de algoritmos numéricos en la Biotecnología

6.7 BIBLIOGRAFIA.

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab, Delores M Etter, Pearson

Programación lógica Pascual Julián Iranzo / María Alpuente Frasnedo Editorial Pearson

Técnicas De Programacion GarcíaEditorial: [Mc Graw Hill](http://www.buscalibros.cl/libros-editorial-mcgraw-hill-id_97897010.htm)

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICO NO.7

NOMBRE DE LA PRÁCTICA

Estructuras de control

7-1 OBJETIVO GENERAL

El alumno adquirirá y desarrollará la habilidad para diseñar algoritmos con estructuras de control (condicionado y fijo) como herramientas para la solución de problemas de ingeniería en biotecnología.

7.1.1 OBJETIVOS PARTICULARES.

El alumno adquirirá y desarrollará la habilidad de diseñar algoritmos iterativos condicionados, e implementarlos a través del lenguaje de programación de matlab para la solución de problemas de ingeniería en biotecnología.

El alumno adquirirá y desarrollará la habilidad de diseñar algoritmos iterativos fijos, e implementarlos a través del lenguaje de programación de matlab para la solución de problemas de ingeniería en biotecnología empleando los conceptos de vectores o arreglos unidimensionales.

7.2 INTRODUCCION

En muchos programas será necesario ejecutar acciones en forma repetitiva. Una parte de un programa que repite una sentencia o un grupo de sentencias se denomina **ciclo**. En matlab se tiene dos formas de representar ciclos, uno es fijo y el otro condicionado.

Una de estas formas es a través de la sentencia ***while*** (o ciclo ***while***).

Observe que luego de la palabra *while* se encuentra una comparación (expresión booleana) entre paréntesis, después se encuentran una serie de sentencia que deberán de realizarse si se cumple la condición y terminará con la palabra end, a dicho grupo de sentencias se le conoce como el ***cuerpo del ciclo condicionado***. Las sentencias entre la palabra while – end se repiten mientras la expresión booleana tenga el valor de verdadero. A cada repetición del ciclo se le conoce como **iteración**.

7.3 MATERIAL

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Plantilla de acrílico o cartón para diagramas de flujo.

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

* 1. DESARROLLO
     1. Ciclos con un Numero Indeterminado de Iteraciones ( Hacer-Mientras )

Son aquellos en que el número de iteraciones no se conoce con exactitud, ya que esta dado en función de un dato dentro del programa.

* ***Hacer-Mientras:*** Esta es una estructura que repetirá un proceso durante “N” veces, donde “N” puede ser fijo o variable. Para esto, la instrucción se vale de una condición que es la que debe cumplirse para que se siga ejecutando. Cuando la condición ya no se cumple, entonces ya no se ejecuta el proceso.

La forma de esta estructura es la siguiente:

CUERPO DEL

CICLO

CONDICION

Hacer mientras <condición>

Accion1 NO

Accion2

.

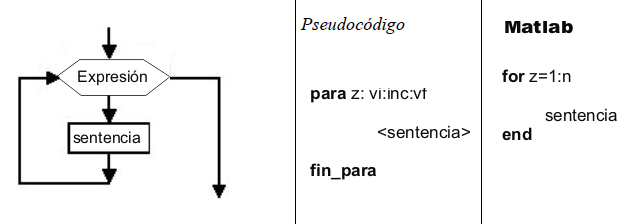
. SI

AccionN

Fin-mientras

7.4 .2 Repetir-Hasta

Esta es una estructura similar en algunas características, a la anterior. Repite un proceso una cantidad de veces, pero a diferencia del Hacer-Mientras, el Repetir-Hasta lo hace hasta que la condición se cumple y no mientras, como en el Hacer-Mientras. Por otra parte, esta estructura permite realizar el proceso cuando menos una vez, ya que la condición se evalúa al final del proceso, mientras que en el Hacer-Mientras puede ser que nunca llegue a entrar si la condición no se cumple desde un principio.



Ejemplos de ejercicios con ciclos o iteraciones condicionadas y fijas.

Ejercicio No.1

|  |  |  |
| --- | --- | --- |
| Planteamiento de Problema  Encuentre el promedio de la presión de un paciente que es valorado durante n días. | Diagramas de Flujo | Prueba de Escritorio  n = 5  a = 0  w = 1  Presión 80  a = 0 +80 = 80  w = 1 + 1 = 2  Presión 70  a = 80 + 70 =150  w = 2 + 1 = 3  Presión 60  a = 150 + 60 = 210  w = 3 + 1 = 4  Presión 90  a = 210 + 90 = 300  w = 4 + 1 = 5  Presión 80  a = 300 +80 = 380  w = 5 + 1 = 6  promedio 380/ 5 = 76 |
| Análisis del Problema  DE.- n, presión  DS.- promedio  Proceso  a = 0  repetición de n veces  a = a + presión  fin repetición  Promedio= a/n  Fin del problema | Codificación  % Programa que encuentre el promedio de la presión de un paciente que  % es valorado durante n días.  n = input ('Introduce la cantidad de días en que se le tomará la presión ');  a = 0 ;  for w = 1:1:n  presion = input('Introduce la nueva presion obtenida ');  a = a + presion;  end  promedio = a / n ;  fprintf ('El promedio de la presión del paciente es de %g',promedio) | Ejecución  >> E24  Introduce la cantidad de días en que se le tomará la presión 5  Introduce la nueva presion obtenida 80  Introduce la nueva presion obtenida 70  Introduce la nueva presion obtenida 60  Introduce la nueva presion obtenida 90  Introduce la nueva presion obtenida 80  El promedio de la presión del paciente es de 76>> |

Ejercicio No.2

|  |  |  |
| --- | --- | --- |
| PLANTEAMIENTO DEL PROBLEMA  'Obtener la intensidad de un semiconductor que tiene un determinado voltaje y resistencia siempre y cuando sea mayor que 0 | DIAGRAMA DE FLUJO | CODIFICACIÒN  clc, clear  disp('Obtener la intensidad de un semiconductor que tiene un determinado voltaje y resistencia siempre y cuando sea mayor que 0')  V=input('Introducir el valor de V: ');  R=input('Introducir el valor de R: ');  while R<0  V=input('Introducir nuevamente el valor de V: ');  R=input('Introducir nuevamente el valor de R; ');  end  I=V/R  fprintf('%g es la intensidad',I) |
| ANALISIS DEL PROBLEMA  DATOS DE ENTRADA: V,R  DATOS DE SALIDA: I  PROCESO:  Mientras R<0  introducir nuevamente R  fin del mientras  I=V/R  I  Fin | PRUEBA DE ESCRITORIO  de V: 20  R: 1  I=V/R  I = 20 | EJECUCIÒN  Obtener la intensidad de un semiconductor que tiene un determinado voltaje y resistencia siempre y cuando sea mayor que 0  Introducir el valor de V: 20  Introducir el valor de R: -1  Introducir nuevamente el valor de V: 20  Introducir nuevamente el valor de R; 1  I =  20  2 es la intensidad>> |

7.5 Ejercicios propuestos para realizar por el alumno.

I.- Diseñar el algoritmo para calcular el factorial de un número proporcionado por el usuario, e implementarlo en matlab.

7.2 Diseñar un algoritmo que calcule el promedio de pesos de una muestra de “n” conejos que serán sometidos a un tratamiento farmacéutico.

7.6 CONCLUSIONES.

El alumno escribirá con sus propias palabras la importancia de diseñar algoritmos empleando estructuras de control iterativas.

7.7BIBLIOGRAFIA.

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab, Delores M Etter, Pearson

Programación lógica Pascual Julián Iranzo / María Alpuente Frasnedo Editorial Pearson

Técnicas De Programación GarcíaEditorial: [Mc Graw Hill](http://www.buscalibros.cl/libros-editorial-mcgraw-hill-id_97897010.htm)

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO. 8

NOMBRE DE LA PRÁCTICA

Estructuras de control anidadas

8.1 OBJETIVO GENERAL

El alumno desarrollará la habilidad de crear algoritmos con estructuras de control anidadas para resolver problemas de ingeniería en biotecnología.

8.1.1 OBJETIVOS PARTICULARES.

El alumno aprenderá el concepto de arreglos bidimensionales o matrices empleado estructuras de control repetitivas anidadas.

8.2 INTRODUCCION

Ciclos Anidados Es posible construir un programa donde se aniden los ciclos; esto es, ejecutar un ciclo dentro de otro, siempre que el ciclo interno esté totalmente contenido dentro del ciclo externo, si no, el algoritmo no es válido.

El caso típico de un ciclo anidado es la asignación de valores a una matriz o arreglo bidimensional.

8.3 MATERIALES

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Plantilla de acrílico o cartón para diagramas de flujo.

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

8.4 DESARROLLO

El alumno relacionará y conjuntará los conceptos aprendidos en prácticas anteriores, deberá seguir la explicación del profesor para poder diseñar e implementar algoritmos numéricos con estructuras anidadas creando arreglos bidimensionales o matrices.

Ejercicio No. 1

|  |  |  |
| --- | --- | --- |
| Planteamiento del Problema  Realiza un programa que muestre el historial de temperatura de k pacientes durante p días | Diagrama de Flujo | Codificación  %Programa que muestra el historial de temperatura de "k" pacientes durante  %"p" días  k=input(' Introduzca el número de pacientes ');  p=input(' Introduzca el número de días del historial ');  for h=1:1:k  for r=1:1:p  temp(k,r)=input(' Introduzca la temperatura del paciente ');  fprintf('La temperatura del paciente es de %g° ',temp(h,r));  end  fprintf(' \n ');  end  for h=1:1:k  for r=1:1:p  fprintf('La temperatura del paciente es de %g° ',temp(h,r));  end  fprintf(' \n ');  end |
| Análisis del problema**:**  Datos de entrada: k,p,temp  Datos de salida: temp (historial)  Proceso:  Repetir p veces durante k veces  temp | Prueba de escritorio  k=3 p=2  k=1  temp=36  temp=35  k=2  temp=36.5  temp=37  k=3  temp=36  temp=35.9   1. 35   36.5 37  36 35.9 | Ejecución del programa  Introduzca el número de pacientes 2  Introduzca el número de días del historial 3  Introduzca la temperatura del paciente 36  Introduzca la temperatura del paciente 37  Introduzca la temperatura del paciente 35.5    Introduzca la temperatura del paciente 36  Introduzca la temperatura del paciente 38.5  Introduzca la temperatura del paciente 37.5    La temperatura del paciente es de 36° La temperatura del paciente es de 37° La temperatura del paciente es de 35.5°  La temperatura del paciente es de 36° La temperatura del paciente es de 38.5° La temperatura del paciente es de 37.5° |

Ejercicio No. 2

|  |  |  |
| --- | --- | --- |
| Planteamiento del problema  Programa que calcule el promedio de longitud de una prótesis de una pierna izquierda para niños de 14 años, el rango del la longitud de la pierna debe ser entre 90-110 cm | Diagrama de Flujo | Codificación  % Programa que calcula el promedio de una prótesis de pierna izquierda para  % niños de 14 años cuya longitud de prótesis sea entre 90 y 110 cm  z=input(' Introduzca el número de niños de 14 años ');  m=0;  for y=1:1:z  w=input(' Introduzca la longitud de prótesis de la pierna izquierda ');  while w<90|w>110  w=input(' Introduzca un nuevo valor de longitud ');  end  m=m+w;  end  q=m/z;  fprintf(' El promedio de longitudes de las prótesis es %g ',q); |
| Análisis del problema:  Datos de entrada: z,w  Datos de salida: q  Proceso:  m=0  Repetir z veces  Mientras sea w<90 ó w>110  Solicita nuevo valor de w  finaliza  m=m+w  Regresa a z hasta finalizar  q=m/z | Prueba de Escritorio  z=3  m=0  y=1  w=120  120<90 ó 120>110 si  w=100  100<90 ó 100>110 no  m=0+100=100  y=2  w=98  98<90 ó 98>110 no  m=100+98=198  y=3  w=97  97<90 ó 97>110 no  m=198+97=245  q=295/3=98.33  98.33 | Ejecución  >> prompro  Introduzca el número de niños de 14 años 3  Introduzca la longitud de prótesis de la pierna izquierda 120  Introduzca un nuevo valor de longitud 100  Introduzca la longitud de prótesis de la pierna izquierda 98  Introduzca la longitud de prótesis de la pierna izquierda 97  El promedio de longitudes de las prótesis es 98.3333 >> |

8.5 Ejercicios propuestos para realización por parte de los alumnos.

I.- Diseñe un algoritmo que calcule el promedio de peso por día de una muestra de ratas de laboratorio que son sometidas a una dieta durante “n” días.

II. Diseñe un algoritmo que calcule el promedio de glucosa por paciente de una muestra de pacientes que son sometidos a un tratamiento para disminuir el nivel de glucosa, así como el promedio de glucosas de toda la muestra.

8.6 CONCLUSIONES.

El alumno explicará con sus propias palabras la importancia del uso de arreglos bidimensionales en su área de especialización dentro de la biotecnología.

8.7 BIBLIOGRAFIA.

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab, Delores M Etter, Pearson

Programación lógica Pascual Julián Iranzo / María Alpuente Frasnedo Editorial Pearson

Técnicas De Programación GarcíaEditorial: [Mc Graw Hill](http://www.buscalibros.cl/libros-editorial-mcgraw-hill-id_97897010.htm)

MANUAL DE PRÁCTICAS DEL TALLER DE PROGRAMACIÒN

PRACTICA NO.9

NOMBRE DE LA PRÁCTICA

Funciones

9.1 OBJETIVO GENERAL

El alumno aprenderá las funciones predefinidas que contiene Matlab como herramientas para el diseño de algoritmos que den solución a problemas de Ingeniería en Biotecnología.

9.1.1 OBJETIVOS PARTICULARES.

El alumno aprenderá las funciones que contiene Matlab para el manejo de Polinomios.

El alumno aprenderá las funciones que contiene Matlab para el manejo de Matrices.

El alumno aprenderá las funciones que contiene Matlab para el manejo de Graficación

9.2 INTRODUCCION

Matlab como todos los lenguajes de programación contiene librerías que contiene programas definidos para diferentes aplicaciones según corresponda a la necesidad o especialidad del usuario.

Existen funciones predefinidas por Matlab y otras que puedes ser creadas por el usuario a través del editor de texto, para fines de aprendizaje del curso solo se describirá el uso de las funciones predefinidas.

Matlab contienen funciones matemáticas y funciones que permiten graficar de manera fácil para el usuario a durante el desarrollo de la practica se describirán alguna de las funciones matemáticas y de graficación de Matlab.

9.3 MATERIALES

Computadora Personal con Sistema Operativo Windows XP, Vista o similar

Software Matlab versión 7.0 o posterior instalado en la PC.

Mouse Pad

Plantilla de acrílico o cartón para diagramas de flujo.

Lápiz o lapicero

Goma, sacapuntas

Pluma o bolígrafo

Cuaderno o hojas blancas tamaño carta.

Dispositivo de almacenamiento externo (USB)

9.4 DESARROLLO

El alumno de manera paralela ejecutará y creará las notas correspondientes de las funciones de matemáticas que el profesor explicará a continuación, Matlab y que serán ejecutadas en la ventana de comandos de Matlab.

9.4.1 Funciones Matemáticas en Matlab

El alumno asignará valor a la variable” x” y ejecutará asignando una variable a cada expresión matemática siguiente,

|  |  |
| --- | --- |
| Nombre de la función logarítmica en Matlab | Operación algorítmica que realiza la función |
| exp(x), | Eleva el exponente x |
| fix(x) | Elimina la parte decimal de x |
| floor(x) | Mayor entero menor o igual que un real dado de x |
| imag (x) | Parte imaginaria de un numero complejo de x |
| log(x) | Obtiene el logaritmo de la variable x |
| log2(x) (en base 2) | Obtiene el logaritmo en base 2 de la variable x |
| log10(x) (en base 10), | Obtiene el logaritmo en base 10 de la variable x |
| mod(x) | Obtiene el modulo de x |
| rem(x) | Obtiene el residuo de una división de x |

|  |  |
| --- | --- |
| Nombre de función trigonométrica en Matlab. | Operación que realiza la función trigonométrica. |
| sin(x) | Seno de la variable x |
| cos(x) | Coseno de la variable x |
| tan(x) | Tangente de la variable x |
| asin(x) | Arco seno de la variable x |
| acos(x) | Arco coseno de la variable x |
| atan(x) | Arco tangente de la variable x |

Otras funciones básicas de Matlab para el área de ingeniería.

|  |  |
| --- | --- |
| abs(x) | Valor absoluto de la variable x |
| int(x) | Parte entera de la variable x |
| round(x) | Redondea al entero más próximo de la variable x |
| sign(x) | función signo |
| sqrt(x) | Obtiene la raíz cuadrada de la variable x |

El alumno realizará las siguientes operaciones en la ventana de comandos de Matlab y guarda sus resultados para que los reporte al entregar la práctica.

1.- d= 5 exp(2) y=abs(-300)

2.- k= log(10) n=int(3.13467)

3.- s=log2(8) h=round(7.4789)

4.- p=log10(100) g=sqrt(144)

5.- j=sin(pi)

6.- t= cos(90)

7.- z=tan(180)

8.- f=asin(270)

9.- u=acos(180)

10.- w=atan(pi)

9.4.2 Vectores y Matrices en Matlab.

Otro tema muy importante es el uso de los vectores y las matrices, en MATLAB la asignación de ambos se trabaja igual. Pero las operaciones aritméticas son diferentes, ya que deberán seguir una seria de reglas para cada uno de ellos, estas reglas se enuncian más adelante.

La asignación de variables en MATLAB es sencilla, y los vectores y matrices no son la excepción. Cuando se desea dar el valor a toda una matriz se puede realizar directamente de la siguiente forma:

A=[1 2 3 4; 5 6 7 8;9 0 1 2];   
A=[1, 2, 3, 4;5, 6, 7, 8;9, 0, 1, 2];

Donde la matriz escrita arriba es:

1 2 3 4

5 6 7 8

9 0 1 2

Las filas se separan por punto y coma y las columnas por espacios o comas. De lo anterior se ve fácilmente que un vector fila se asigna así:

v=[1 2 3]   
v=[1, 2, 3];

y su resultado al ejecutarlo se mostraría de la siguiente manera:

1 2 3

Y un vector columna se asigna así

v=[1; 2; 3];

El resultado obtenido al ejecutarlo sería así:

1

2

3

Si se desea asignar una matriz es necesario tener en una variable determinada un “Z” numero de vectores columna con “P” filas.

Ejemplo

Designe una matriz llamada R de 5 columnas y 3 filas.

R=[3 2 5 6 7;9 16 1 67 23;12 45 32 4 6]

Al ejecutarlo en Matlab tendríamos el siguiente resultado.

3 2 5 6 7

9 16 1 67 23

12 45 32 4 6

Algunas de las operaciones de vectores con un escalares son las siguientes: Siendo v: vector, k: escalar:

|  |  |
| --- | --- |
| Asignación en Matlab | Operación Aritmética |
| P=v+k | adición o suma |
| P= v-k | sustracción o resta |
| P =v\*k | multiplicación |
| P =v/k | divide cada elemento de v por k |
| P= k./v | divide k por cada elemento de v |
| P= v.^k | potenciación de cada componente de v a k |
| P= k.^v | potenciación k elevado a cada componente de v |

El alumno realizará las siguientes operaciones con vectores y escalares en la ventana de comandos de Matlab y guarda sus resultados para que los reporte al entregar la práctica.

Siendo G un vector columna y T=34

G=[23;45;6;7;9;98;6]

|  |
| --- |
| 1.- T=v+k |
| 2.- Y= v-k |
| 3.- A=v\*k |
| 4.- PP =v/k |
| 5.- H= k./v |
| 6.- S= v.^k |
| 7.- B= k.^v |

**Operaciones entre vectores y matrices:**

|  |  |
| --- | --- |
| Operador Aritmético | Operación Aritmética |
| + | adición o suma |
| – | sustracción o resta |
| \* | multiplicación matricial |
| .\* | producto elemento a elemento |
| ^ | potenciación |
| .^ o | elevar a una potencia elemento a elemento |
| \ | división-izquierda |
| / | división-derecha |
| ./ y .\ | división elemento a elemento |
| B=A’ | matriz traspuesta: (en complejos calcula la traspuesta conjugada, sólo si la traspuesta es B=A.’) |

El alumno asignará las siguientes vectores y matices en Matlab y realizará las siguientes operaciones aritméticas con ellos.

A=[23 45 67 32] B=[56 89 21 15] C=[34 56 78 20;67 64 23 29; 1 43 62 76;81 48 37 39] D=[37 48 20 31;59 38 92 56;78 64 28 40;38 46 29 35]

|  |  |
| --- | --- |
| VECTORES | MATRICES |
| T=A+B | J=C+D |
| S=A–B | SS=C–D |
| AA=A\*B | R=C\*D |
| FF=A.\*B | H=C.\*D |
| DD=A^B | MM=C^D |
| I=A.^ oB | YY=C.^ Od |
| K=A\B | H=C\D |
| G=A/B | PP=C/D |
| N=A./B ; M=A .\ B | U=C./D XX=C .\D |
|  | BB=C’  CC=B’ |

9.4.3 Funciones Matemáticas para el uso de Polinomios en Matlab.

Una de las funciones matemáticas mas utilizadas por estudiantes de ingeniería es el manejo de Polinomios.Los polinomios se representan en Matlab por un vector fila de dimensión n+1 siendo “n” el grado del polinomio.

Ejemplo: x3+2x-7 se representa por

>> pol1=[1 0 2 -7] donde el polinomio es asignado a la variable pol1, se ha de observar que el polinomio es de orden 3, pero no tiene termino de orden 2, por lo que para Matlab es necesario indicar la posición vacía con “0” como se mostró en la asignación anterior.

Funciones preestablecidas más usadas para polinomio en Matlab.

|  |  |
| --- | --- |
| Función en Matlab | Función que realiza |
| roots(x) | Encuentra las raíces de un polinomio x |
| poly(x) | Devuelve el polinomio característico de la matriz X |
| p=poly(r) | Da los coeficientes del polinomio p cuyas raíces son el vector r |
| p=polyfit(x,y,n) | Polinomio de orden n que ajusta los puntos(x,y) |
| y=polyval(x,p) | Evalúa al polinomio x en un determinado punto p. |

El cálculo de las raíces de un polinomio es a través de la función roots (da un vector columna, aunque pol1 es un vector fila)

Ejemplo

pol1=[2 4 0 2]

>>raices=roots(pol1)

Entregará como resultado tres raíces ya que el polinomio es de orden 3.

Para evaluar en un punto determinado el polinomio se emplea la función polyval

Ejemplo

>>y=polyval(pol1,x)

El alumno realizará los siguientes ejercicios de polinomios en la ventana de Matlab, analizará sus resultados con su profesor y compañeros, mismos que deberá incluir en el reporte de la práctica.

Ejercicio.

El siguiente polinomio es la representación de un proceso farmacéutico donde el principio activo es el paracetamol y se requiere optimizar su uso, por lo que se requiere obtener sus raíces y encontrar su máximo aprovechamiento.

P=

a)Utilizar la función roots para determinar sus raíces.

b) Proponer un rango en función de las raíces encontradas y evaluar el polinomio P en dicho intervalo.

9.4.4 Funciones empleada por Matlab para la graficación.

Las funciones de graficación en 2D son las siguientes.

|  |  |
| --- | --- |
| Función | Descripción |
| **plot(V1,V2, opcion)** | Crea un gráfico a partir de vectores con escalas lineales sobre ambos ejes, *opción*: permite elegir color y trazo de la curva |
| **loglog(X,Y)** | Realiza gráficas similares a plot(V1,V2) pero con escala logarítmica en los dos ejes. |
| **semilogx(X,Y):** | Permite tener una grafica con escala lineal en el eje de las Y y logarítmica en el eje de las X. |
| **semilogy(X,Y):** | Permite tener una grafica con escala lineal en el eje de las X y logarítmica en el eje de las Y. |

Funciones para agregar títulos a los gráficos.

|  |  |
| --- | --- |
| title(´título´) | Agrega un título al dibujo |
| xlabel(´...´) | Agrega un título al eje ´X´. xlabel off lo elimina |
| ylabel(´...´) | Agrega un título al eje ´Y´. ylabel off lo elimina |
| legend() | Define rótulos para las líneas |
| hold | Permite mantener el grafico existente con todas sus propiedades, de modo que el siguiente grafico que se realice se sitúe sobre los mismos ejes y se superponga al existente. |
| grid | Activa la cuadrícula en el dibujo |
| subplot(m,n,p) | Divide la ventana gráfica en mxn subventanas y coloca el gráfico corriente en la ventana  p-ésima , empezando a contar por la parte superior izquierda y de izquierda a derecha hasta acabar la línea. |
| Text(x,y,’texto’ | Sitúa el texto en el punto en el punto (x,y) dentro del grafico 2-D |

Tabla de opciones para agregar a la función plot



El alumno graficará el polinomio asignado en la sección anterior y procederá a graficarlo, mismas que deberán incluirse en el reporte de la práctica.

Ejercicio

Grafique la siguiente función ) representada en escala normal, en escala logarítmica y en escala semilogaritmica.

Ingrese los siguientes comandos en la ventana de matlab

>> x=0:0.01:3;

y=abs(exp(-0.5\*x).\*sin(5\*x));

subplot(2,2,1)

plot(x,y)

title(‘normal’)

hold on

subplot on

subplot(2,2,2)

loglog(x,y)

title(‘logaritmica’)

subplot(2,2,3)

semilogx(x,y)

title(‘semilogaritmica en el eje X’)

subplot(2,2,4)

semilogy(x,y)

title(‘semilogaritmica en el eje Y’)

Una vez realizados los ejercicios propuestos el profesor responsable del curso, asignará ejercicios complementarios para anexarlos a la práctica.

9.5 CONCLUSIONES.

El alumno proporcionará sus conclusiones en relación de lo aprendido en los conceptos de funciones matemáticas, vectores, matrices y graficación.

9.6 BIBLIOGRAFIA.

Graficación con Matlab Cairo, Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall)

Matlab y sus Aplicaciones en la Ciencias y la Ing., [Pérez López](http://www.lsf.com.ar/resulta.aspx?key=PEREZ%20LOPEZ&criterio=ae), Editorial: [Prentice Hall](http://www.lsf.com.ar/resulta.aspx?key=Prentice%20Hall&editorial=Prentice%20Hall).

Solución De Problemas De Ingeniería Con Matlab, Delores M Etter, Pearson