CIMAT

Informática II para Bachillerato

INPUT OUTPUT(Variables)

Estructuras de control if,switch

José Luis Alonzo Velázquez

CIMAT

Sesión 2

IDE

¿Qué es un IDE?

Un **entorno de desarrollo integrado** o **IDE** (acrónimo en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

IDE

¿Qué es un IDE?

Un **entorno de desarrollo integrado** o **IDE** (acrónimo en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Componentes Clásicas

- Un editor de texto.

Componentes Clásicas

- Un editor de texto.
- Un compilador.

Componentes Clásicas

- Un editor de texto.
- Un compilador.
- Un intérprete.

Componentes Clásicas

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.

Componentes Clásicas

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.

Componentes Clásicas

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.

Componentes Clásicas

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuario.

Un editor de texto

Un **editor de texto** es un programa que permite crear y modificar archivos digitales compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto o texto plano. El programa lee el archivo e interpreta los bytes leídos según el código de caracteres que usa el editor. Hoy en día es comúnmente de 7- ó 8-bits en ASCII o UTF-8, rara vez EBCDIC.

Un editor de texto

Un **editor de texto** es un programa que permite crear y modificar archivos digitales compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto o texto plano. El programa lee el archivo e interpreta los bytes leídos según el código de caracteres que usa el editor. Hoy en día es comúnmente de 7- ó 8-bits en ASCII o UTF-8, rara vez EBCDIC.

Ejemplo:

Emacs es un editor de texto con una gran cantidad de funciones, muy popular entre programadores y usuarios técnicos.

Emacs

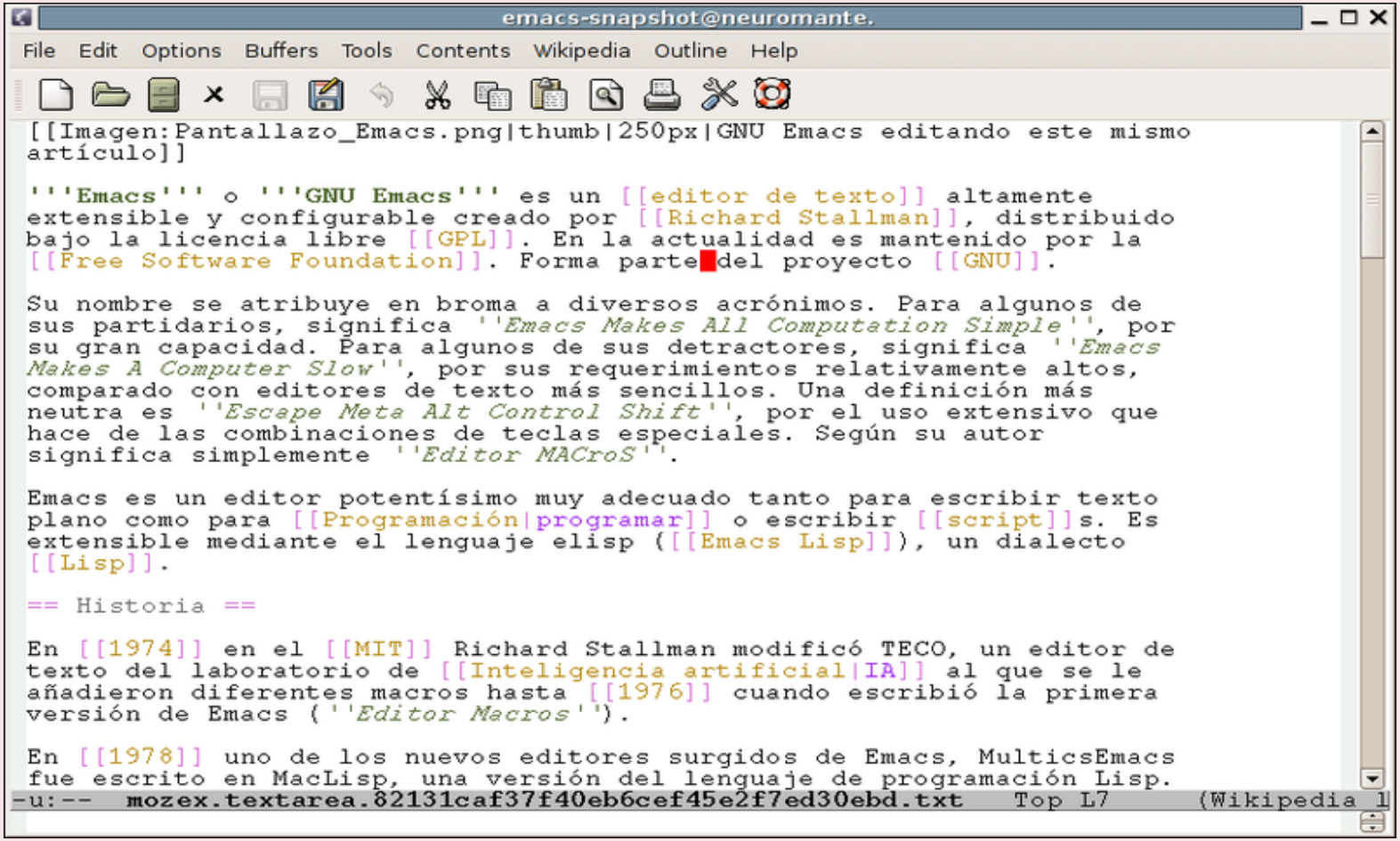

Figura : Captura de pantalla de una ventana Emacs.

Un compilador

Un **compilador** es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto. Este proceso de traducción se conoce como compilación. Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje de máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a como piensa un ser humano, para luego compilarlo a un programa más manejable por una computadora.

Intérprete

Un **Intérprete** o interpretador es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel. Los intérpretes se diferencian de los compiladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los primeros (los intérpretes) sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

Depurador

Un **depurador** (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático.

Depurador

Un **depurador** (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático.

Uso

Al iniciarse la depuración, el depurador lanza el programa a depurar. Éste se ejecuta normalmente hasta que el depurador detiene su ejecución, permitiendo al usuario examinar la situación. El depurador permite detener el programa en:

- Un punto determinado mediante un punto de ruptura.

Depurador

Un **depurador** (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático.

Uso

Al iniciarse la depuración, el depurador lanza el programa a depurar. Éste se ejecuta normalmente hasta que el depurador detiene su ejecución, permitiendo al usuario examinar la situación. El depurador permite detener el programa en:

- Un punto determinado mediante un punto de ruptura.
- Un punto determinado bajo ciertas condiciones mediante un punto de ruptura condicional.

Depurador

Un **depurador** (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático.

Uso

Al iniciarse la depuración, el depurador lanza el programa a depurar. Éste se ejecuta normalmente hasta que el depurador detiene su ejecución, permitiendo al usuario examinar la situación. El depurador permite detener el programa en:

- Un punto determinado mediante un punto de ruptura.
- Un punto determinado bajo ciertas condiciones mediante un punto de ruptura condicional.
- Un momento determinado cuando se cumplan ciertas condiciones.

Depurador

Un **depurador** (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático.

Uso

Al iniciarse la depuración, el depurador lanza el programa a depurar. Éste se ejecuta normalmente hasta que el depurador detiene su ejecución, permitiendo al usuario examinar la situación. El depurador permite detener el programa en:

- Un punto determinado mediante un punto de ruptura.
- Un punto determinado bajo ciertas condiciones mediante un punto de ruptura condicional.
- Un momento determinado cuando se cumplan ciertas condiciones.
- Un momento determinado a petición del usuario.

Control de versiones

Se llama **control de versiones** a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Control de versiones

Se llama **control de versiones** a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

GUI

La **interfaz gráfica de usuario**, conocida también como GUI (del inglés graphical user interface) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Algunos IDE's soportan múltiples lenguajes, tales como Eclipse o NetBeans, ambas basadas en Java o MonoDevelop, basado en C#. El soporte para lenguajes alternativos es a menudo proporcionada por plugins, que les permite ser instalado en el mismo IDE al mismo tiempo. Por ejemplo, Eclipse y NetBeans tiene plugins para C/C++ , Ada, Perl, Python, Ruby y PHP , entre otros lenguajes.

help disp

```
DISP Display array.
    DISP(X) displays the array, without printing
    the array name.In all other ways it's the
    same as leaving the semicolon off an
    expression except that empty arrays
    don't display.

    If X is a string, the text is displayed.
```

Importancia

Esta función aunque pareciera ser inútil es la manera correcta de imprimir mensajes en la consola. A lo largo del curso veremos la gran utilidad de poder silenciar las funciones y/o asignaciones. Y utilizar la función disp como una buena practica de programación en MATLAB.

OUTPUT

El hola mundo que hicimos en la clase pasada es tan solo un programa(script) que escribe un mensaje en pantalla através de la instrucción **disp** . Es decir, produce una salida(output). Sin embargo este programa no hace nada más. Además no recibe ninguna entrada(input) del usuario.
Los típicos programas realmente muestran salidas que dependen de entradas dadas por el usuario del programa, es decir, un programa es usualmente interactivo. Para esto necesitamos lo que se conoce como variables.

Variable

Para poder leer algo, necesitamos un lugar donde poner lo leído, i.e. necesitamos un lugar en la memoria de la maquina donde podamos guardar esta información. A este "lugar" lo llamaremos **objeto**.
Un objeto es una región de memoria que tendrá un **tipo** que especifica que clase de información esta siendo colocada en el. Este objeto es llamado **variable**. Será en estas variables donde guardaremos información en nuestros programas.

Los nombres de las variables pueden usar cualquier carácter común, mas no se debe utilizar acentos, ni espacios en los nombres de las variables.

Variable

Una buena practica de programación es utilizar nombres de variables que les recuerden la utilidad de la variable, siempre en minúsculas. Si se utilizan dos o más palabras para la variable se acostumbran usar dos formatos, el primero es poner un guión bajo entre cada palabra, y el segundo, es comenzar con minúscula y cada palabra siguiente iniciara con mayúscula.

Ejemplos:

```
1 variable_uno
2 variableUno
3 variable_ejemplo_de_buena_practica
4 variableEjemploDeBuenaPractica
```

Ejemplos:

```
1 disp('Hola Mundo');
2 n=3;
3 disp(n);
4 nombre='Roberto';
5 disp(nombre);
6 arreglo=[1:10];
7 disp(arreglo);
```

Tipos de Variables típicos de C

bool x	x es a Booleano (valor true and false).
char x	x is a character (usually 8 bits).
int x	x is the default integer type.
float x	x is a floating-point number.
double x	x is a double-precision floating-point number.

La función de salida sprintf

`[s, errmsg] = sprintf(format, A, ...) formats the data in matrix A (and in any additional matrix arguments) undercontrol of the specified format string and returns it in the MATLAB string variable s. The sprintf function returns an error message string errmsg if an error occurred. errmsg is an empty matrix if no error occurred.`

Ejemplo:

```
1 n=3;
2 sprintf('Hola Mundo %d',n);
3 mensaje=sprintf('Hola Mundo %d',n);
4 disp(mensaje);
```

Especificador	Salida	Ejemplo
c	Carácter	'a'
d	Notación Decimal	392.5
e	Notación científica usando e	3.9265e+2
E	Notación científica usando E	3.9265E+2
f	Decimal de punto flotante	392.65
g	El más corto entre %e or %f	392.65
G	El más corto entre %E or %f	392.65
o	Octal sin signo	610
s	Cadena de caracteres	ejemplo
u	Notación decimal sin signo	7235
x	Entero Hexadecimal	7fa
X	Entero Hexadecimal	7FA

Escape de caracteres especiales.

Character	Description
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\\	Backslash
\“ (two single quotes)	Single quotation mark
%%	Percent character

¿Como hago que el usuario entre datos?

```
1 variable = input( const char *);
```

Ejemplo

```
1 clear
2 clc
3 nombre = input('Dame el nombre del alumno: ');
4 edad = input('Dame la edad del alumno: ');
5 mensaje=sprintf('El alumno %s tiene %d años',nombre,
      edad);
6 disp(mensaje);
```

¿Y donde utilizo los especificadores?

Ejemplo

```
1 clear
2 clc
3 nombre = input('Dame el nombre del alumno: ','s');
4 edad = input('Dame la edad del alumno: ');
5 mensaje=sprintf('El alumno %s tiene %d años',nombre,
      edad);
6 disp(mensaje);
```

Estructuras de selección

MATLAB tiene dos(al menos) estructuras de control para la selección, **if** (selección simple y binaria) y **switch** (selección múltiple).

Sintaxis de la estructura de control **if**

if ⟨Condición⟩
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
else
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
end

Ejemplo

```
numero=input('Escribe un numero: ');
if numero >= 4
    mensaje=sprintf('El numero %d >= 4',numero);
else
    mensaje=sprintf('El numero %d < 4',numero);
end
disp(mensaje);
```

Como hacer comentarios

```
1 % esto comenta una linea.
2 % es decir, este texto no sera procesado.
```

Estructuras de selección(decisión)

MATLAB tiene dos(al menos) estructuras de control para la selección, **if** (selección simple y binaria) y **switch** (selección múltiple).

Sintaxis de la estructura de control **if**

if ⟨Condición⟩
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
else[a]
⟨Instrucción⟩
⟨Instrucción⟩
⋮
⟨Instrucción⟩
end

[a] La instrucción "else" es opcional.

Checar paridad de un número

```
clear
clc
numero=input('Escribe un numero: ');
if mod(numero,2)==0
    mensaje=sprintf('El numero  %d es par',numero);
else
    mensaje=sprintf('El numero  %d es impar',numero);
end
disp(mensaje);
```

Una de las cualidades de los operadores de control es la posibilidad de anidarlos.

```
if ⟨Condición⟩
        ⟨Instrucción⟩
        if ⟨Condición2⟩
                ⟨Instrucción⟩
        end
        ⟨Instrucción⟩
else
        ⟨Instrucción⟩
end
```

Checar paridad de un número

```
numero=input('Escribe un numero: ');
if mod(numero,2)==0
    mensaje=sprintf('El numero %d es par',numero);
    if mod(numero,7)==0
        mensaje=sprintf('El numero %d no solo es par,
     si no también múltiplo de 7',numero);
    end
else
    mensaje=sprintf('El numero %d es impar',numero);
end
disp(mensaje);
```

Los siguientes dos códigos son equivalentes:

```
numero=input('Escribe un numero: ');
if mod(numero,2)==0
    mensaje=sprintf('El numero %d es par.\n',numero);
else
    if mod(numero,3)==0
        mensaje=sprintf('El numero %d es multiplo de 3.\n',numero);
    else
        mensaje=sprintf('');
    end
end
disp(mensaje);
```

```
numero=input('Escribe un numero: ');
if mod(numero,2)==0
    mensaje=sprintf('El numero  %d es par.\n',numero);
elseif mod(numero,3)==0
    mensaje=sprintf('El numero  %d es multiplo de 3.\n
    ',numero);
else
    mensaje=sprintf('');
end
disp(mensaje);
```

Problema para clase

Hacer un menu que despliegue lo siguiente en pantalla:
Escoja una opción:
a)opcion 1
b)opcion 2
c)opcion 3
s)salir
si se escoge la opcion 1 imprima "Se eligio la opcion 1",
analogamente 2 y 3.

Estructura de Control if

```
method = input('Qué metodo deseas utilizar: ','s');
if strcmp(method,'derivar')
    disp('El metodo es derivar');
else
    if strcmp(method,'integrar')
        disp('El metodo es integrar');
    else
        if strcmp(method,'limpiar')
            disp('El metodo es Limpiar')
        else
            disp('El metodo es desconocido');
        end
    end
end
```

Estructura de Control if

```
1  method = input('Qué metodo deseas utilizar: ','s');
2  if strcmp(method,'diferencias finitas') || strcmp(
       method,'derivar')
3      disp('El metodo es derivar');
4  elseif strcmp(method,'integrar')
5      disp('El metodo es integrar');
6  elseif strcmp(method,'limpiar')
7      disp('El metodo es Limpiar');
8  else
9      disp('El metodo es desconocido');
10 end
```

Estructuras de selección

Aunque la sentencia if de es muy potente, en ocasiones su escritura puede resultar tediosa, sobre todo en casos en los que el programa presenta varias elecciones después de chequear una expresión: selección múltiple o multialternativa. En situaciones donde el valor de una expresión determina qué sentencias serán ejecutadas es mejor utilizar una sentencia switch en lugar de una if.

Sintaxis de la estructura de control **switch**

```
switch (selector)
        case ⟨opcion 1⟩:
                ⟨bloque de instrucciones⟩
                break;
        case ⟨opcion 2⟩:
                ⟨bloque de instrucciones⟩
                break;
                        ⋮
        case ⟨opcion n⟩:
                ⟨bloque de instrucciones⟩
                break;
        default:
                ⟨bloque de instrucciones⟩
end
```

Estructura de Control Switch

```
1  method = input('Qué metodo deseas utilizar: ','s');
2  switch lower(method)
3      case {'diferencias finitas','derivar'}
4          disp('El metodo es derivar');
5      case 'integrar'
6          disp('El metodo es integrar');
7      case 'limpiar'
8          disp('El metodo es Limpiar')
9      otherwise
10         disp('El metodo es desconocido');
11 end
```

La palabra reservada **break** permite que el flujo de programa se detenga justo después de la ejecución de la sentencia anterior a ese **break**, impidiendo que se ejecuten las sentencias correspondientes a las siguientes alternativas del **switch**. Por tanto, debemos obligatoriamente acabar cada bloque de sentencias correspondiente a cada alternativa con una sentencia **break**.

Por otro lado, la alternativa **otherwise** es opcional y engloba un conjunto de sentencias (que puede ser vacío, contener una sola sentencia o varias) que se ejecutan en caso de que ninguna de las alternativas del switch tenga un valor coincidente con el resultado de evaluar la expresión del selector.

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

- "**Errores**": son errores que **impiden que el programa pueda ejecutarse**, los programas con "errores" no pueden pasar de la fase ejecución.

Tipos de Errores

Los compiladores clasifican los errores en dos tipos, dependiendo de lo serios que sean:

- "**Errores**": son errores que **impiden que el programa pueda ejecutarse**, los programas con "errores" no pueden pasar de la fase ejecución.
- "**Warnings**": son errores de poca entidad, (según el compilador o debugger que por supuesto, no tiene ni idea de lo que intentamos hacer). **Estos errores no impiden** pasar a la fase de ejecución.