

PROGRAMACION

TECNOLOGÍAS DE LA INFORMACIÓN Y DE LA
COMUNICACIÓN
1º BACHILLERATO
IES EDUARDO VALENCIA

ÍNDICE

- LENGUAJES DE PROGRAMACION
- BASIC
- PROGRAMACION EN GAMBAS
- VISUAL BASIC



1. PROGRAMAS INFORMÁTICOS

UN **PROGRAMA INFORMÁTICO** ES UN CONJUNTO DE INSTRUCCIONES GRABADAS EN LA MEMORIA DEL ORDENADOR QUE LE INDICA LAS TAREAS QUE ESTE DEBE REALIZAR.

DICHAS INSTRUCCIONES SE GRABAN EN LA MEMORIA DEL ORDENADOR, EN FORMATO BINARIO (**CÓDIGO MÁQUINA**).

2. LENGUAJES DE PROGRAMACIÓN

LENGUAJE DE PROGRAMACIÓN ES UN IDIOMA CREADO PARA FACILITAR LA COMUNICACIÓN ENTRE EL PROGRAMADOR Y EL ORDENADOR.

UN LENGUAJE CONSTA DE UN CONJUNTO DE PALABRAS (INSTRUCCIONES) QUE TIENEN SU PROPIA TRADUCCIÓN AL CÓDIGO MÁQUINA O BINARIO, QUE ES EL ÚNICO QUE ENTIENDE EL ORDENADOR.

2. LENGUAJES DE PROGRAMACIÓN

LENGUAJES INTÉRPRETES: CADA INSTRUCCIÓN SE VA TRADUCIENDO A BINARIO EN EL ORDEN EN QUE SE VAN ENCONTRANDO.

2. LENGUAJES DE PROGRAMACIÓN

LENGUAJES COMPILADOS: LA SECUENCIA DE INSTRUCCIONES QUE COMPONEN EL PROGRAMA SE ESCRIBEN COMO UN TEXTO (**CÓDIGO FUENTE**), Y DESPUÉS DICHO CÓDIGO FUENTE SE COMPILA (SE TRADUCE EL CONJUNTO A CÓDIGO MÁQUINA).

3. FUNDAMENTOS DE PROGRAMACIÓN

CONJUNTO DE INSTRUCCIONES: LISTA DE COMANDOS QUE COMPRENDE UN LENGUAJE DE PROGRAMACIÓN. CADA INSTRUCCIÓN REALIZA UNA FUNCIÓN EN PARTICULAR. PUEDE NECESITAR COMPLEMENTARSE CON UN VALOR (ARGUMENTO)

EJ: ECHO "HOLA MUNDO", PRINT "HOLA MUNDO"

3. FUNDAMENTOS DE PROGRAMACIÓN

CONSTANTE: VALOR FIJO QUE SE UTILIZARÁ EN EL PROGRAMA Y QUE NO VARIARÁ. EJ: $\pi=3.14$

VARIABLE: VALOR QUE SE UTILIZARÁ A LO LARGO DEL PROGRAMA, Y QUE PUEDE VARIAR SU CONTENIDO CON EL TIEMPO. EJ: NOMBRE="Pedro", NOMBRE="Teresa"

3. FUNDAMENTOS DE PROGRAMACIÓN

MUCHOS LENGUAJES NECESITAN UNA DECLARACIÓN PREVIA DE VARIABLES (AVISAR DE QUE SE VA A UTILIZAR UN N° DETERMINADO DE VARIABLES, PARA QUE SE PUEDA RESERVAR UN ESPACIO DETERMINADO DE MEMORIA)

3. FUNDAMENTOS DE PROGRAMACIÓN

TIPOS COMUNES DE VARIABLE:

- ENTERA (INTEGER)
- FLOTANTE (FLOAT)
- LÓGICA (VERDADERO/FALSO)
- STRING (TEXTO)
- BYTE (0-255)
- OBJECT

(ESTOS TIPOS PUEDEN VARIAR DE UN LENGUAJE A OTRO)

3. FUNDAMENTOS DE PROGRAMACIÓN

SUBROUTINA: CÓDIGO QUE EJECUTA UNA TAREA, PERO NO DEVUELVE NINGÚN VALOR (EJEMPLO: DIBUJAR EN PANTALLA, ACTIVAR UN SONIDO...)

FUNCIÓN: CÓDIGO QUE SIEMPRE DEVUELVE UNO O MÁS DATOS. PUEDE (O NO) NECESITAR UNO O VARIOS DATOS DE ENTRADA.

3. FUNDAMENTOS DE PROGRAMACIÓN

OPERACIONES MATEMÁTICAS

- SUMA: $a+b$
- RESTA: $a-b$
- MULTIPLICACIÓN: $a*b$
- DIVISIÓN: a/b
- RESTO O MÓDULO: $a\%b$ (devuelve el resto de una división como entero)
- POTENCIA: a^b

3. FUNDAMENTOS DE PROGRAMACIÓN

OPERACIONES LÓGICAS

- AND: $a \& b$ (true si $a = \text{true}$ y $b = \text{true}$)
- OR: $a \parallel b$ (true si $a = \text{true}$ o $b = \text{true}$)
- NOT: $!a$ (true si $a = \text{false}$ y al revés)

3. FUNDAMENTOS DE PROGRAMACIÓN

SENTENCIAS DE CONTROL DE FLUJO

IF (expresión a comprobar) THEN(conjunto de instrucciones a completar) ELSE (conjunto de instrucciones complementarias)

3. FUNDAMENTOS DE PROGRAMACIÓN

SENTENCIAS DE CONTROL DE FLUJO

WHILE (prueba lógica)

(-

-

-

Conjunto de comandos a ejecutar

- BREAK puede romper este bucle

-

-)

3. FUNDAMENTOS DE PROGRAMACIÓN

SENTENCIAS DE CONTROL DE FLUJO

SELECT valor a comprobar

CASE primer valor

Comandos para cuando suceda el primer caso

BREAK

CASE segundo valor

Comandos para cuando suceda el segundo caso

BREAK

Y así sucesivamente...

END SELECT

3. FUNDAMENTOS DE PROGRAMACIÓN

SENTENCIAS DE CONTROL DE FLUJO

FOR variable=valor1 TO valor2 [STEP paso]

- Conjunto de comandos a cumplir

NEXT

3. FUNDAMENTOS DE PROGRAMACIÓN

INCREMENTAR: VALOR++

DECREMENTAR: VALOR--

3. FUNDAMENTOS DE PROGRAMACIÓN

PROGRAMACIÓN SECUENCIAL: EL PROGRAMA CONSTA DE UN CONJUNTO DE INSTRUCCIONES QUE SE CUMPLEN SIGUIENDO UN ORDEN DETERMINADO

3. FUNDAMENTOS DE PROGRAMACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS:
EL PROGRAMA NO SIGUE UN ORDEN DETERMINADO, SINO QUE REACCIONA A LA VARIACIÓN EN LAS PROPIEDADES DE LOS OBJETOS DEFINIDOS EN EL PROGRAMA

3. FUNDAMENTOS DE PROGRAMACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS:
EL PROGRAMA NO SIGUE UN ORDEN DETERMINADO, SINO QUE REACCIONA A LA VARIACIÓN EN LAS PROPIEDADES DE LOS OBJETOS DEFINIDOS EN EL PROGRAMA

3. GAMBAS

GAMBAS ES UNA SUITE (CONJUNTO DE PROGRAMAS INFORMÁTICOS) QUE PERMITE CREAR PROGRAMAS INFORMÁTICOS BASADOS EN EL LENGUAJE INFORMÁTICO **BASIC**, PARA SU UTILIZACIÓN EN S.O. LINUX.

TANTO **GAMBAS** COMO **VISUAL BASIC** UTILIZAN UNA SERIE DE ELEMENTOS COMUNES (FORMULARIOS, CONTROLES, ETC...), QUE LOS DEFINEN COMO HERRAMIENTAS DE DESARROLLO VISUAL.

3. GAMBAS

PROYECTO: ES EL CONJUNTO DE ARCHIVOS QUE VAN A CONFORMAR LA APLICACIÓN QUE PROGRAMAMOS. DICHOS ARCHIVOS ESTÁN EN CÓDIGO FUENTE QUE AL FINAL HABRÁ QUE COMPILAR. TODOS LOS ARCHIVOS DEBEN GUARDARSE EN EL MISMO DIRECTORIO.

LOS ARCHIVOS DE UN PROYECTO PUEDEN SER **MÓDULOS, CLASES Y FORMULARIOS**

3. GAMBAS

MÓDULO: CÓDIGO FUENTE QUE CONTIENE
EL CÓDIGO DEL PROGRAMA A EJECUTAR

3. GAMBAS

CLASE: TIPO ESPECIAL DE CÓDIGO QUE PERMITE DESARROLLAR **OBJETOS** DE DICHA CLASE QUE RESPONDEN A ESE TIPO DE CÓDIGO.

POR EJEMPLO, PODRÍAMOS DESARROLLAR EL CÓDIGO DE UNA CLASE LLAMADA **COCHE**, Y DESPUÉS DESARROLLAR VARIOS OBJETOS COMO **TOYOTA**, **SEISCIENTOS**, **PORSCHE** Y **PEGASO**, QUE TENDRÍAN UN COMPORTAMIENTO BASE PARECIDO.

3. GAMBAS

FORMULARIO: ES UN TIPO ESPECIAL DE CÓDIGO QUE DESARROLLA LAS ÁREAS GRÁFICAS (VENTANAS), DE INTERACCIÓN CON EL USUARIO.

3. GAMBAS

FORMULARIO: ES UN TIPO ESPECIAL DE CÓDIGO QUE DESARROLLA LAS ÁREAS GRÁFICAS (VENTANAS), DE INTERACCIÓN CON EL USUARIO.

3. GAMBAS

EN CUALQUIER TIPO DE CÓDIGO FUENTE, NECESITAREMOS **DECLARAR LAS VARIABLES Y CONSTANTES** Y A CONTINUACIÓN DESARROLLAR LAS **SUBROUTINAS Y LAS FUNCIONES**.

3. GAMBAS

TIPOS DE DATOS (VARIABLES o CONSTANTES):

- **SHORT, INTEGER o LONG:** ENTEROS
- **FLOAT:** DECIMALES
- **BOOLEAN:** (TRUE o FALSE)
- **BYTE:** NÚMERO ENTRE 0 Y 255
- **STRING:** CADENA DE CARACTERES
- **DATE:** FECHA/HORA
- **OBJECT:** OBJETO

3. GAMBAS

DECLARACIÓN DE VARIABLES (O CONSTANTES): EN TODOS LOS CASOS, SE TRATA DE EXPLICARLE AL PROGRAMA QUE UTILIZAREMOS UNA VARIABLE CON UN NOMBRE DETERMINADO, INDICÁNDOLE QUÉ TIPO DE VARIABLE SERÁ (ENTERA, LÓGICA, ETC...).

Ej: INT Velocidad
(significaría que utilizaremos una variable llamada Velocidad, que trabajará con números enteros).

3. GAMBAS

DECLARAR VARIABLES EN SUBROUTINAS O FUNCIONES:

`DIM variable AS tipo_de_variable`

Estas variables sólo serán válidas dentro de la subrutina o función en que se han declarado.

3. GAMBAS

DECLARAR VARIABLES AL PRINCIPIO DEL MÓDULO O CLASE:

[STATIC] PUBLIC/PRIVATE variable AS
tipo_de_variable

- STATIC: sólo se utiliza en las clases. Define una variable como común para todos los objetos de esa clase.
- PUBLIC: la variable será válida no sólo para este módulo, sino para todos los del proyecto
- PRIVATE: la variable sólo valdrá para este módulo.

3. GAMBAS

EJEMPLOS: PÁGINAS 44-45 DE LIBRO-
GAMBAS-2 (VER MENTOR)

3. GAMBAS

SUBROUTINA: CÓDIGO QUE EJECUTA UNA TAREA, PERO NO DEVUELVE NINGÚN VALOR (EJEMPLO: DIBUJAR EN PANTALLA, ACTIVAR UN SONIDO...)

FUNCIÓN: CÓDIGO QUE SIEMPRE DEVUELVE UNO O MÁS DATOS. PUEDE (O NO) NECESITAR UNO O VARIOS DATOS DE ENTRADA.

3. GAMBAS

DECLARACIÓN DE SUBROUTINAS:

```
PUBLIC/PRIVATE SUB nombre_de_subrutina()  
;código  
;que integre  
;la subrutina  
END
```

Si los paréntesis están vacíos, no necesita datos de entrada. Si los necesitara, deberíamos indicar el nombre y el tipo de datos entre ellos: (valor1 AS Integer, valor2 AS Float...)

3. GAMBAS

NOMBRES RESERVADOS PARA SUBROUTINAS:

Main: es el nombre de la subrutina principal, por la que empieza la ejecución de código

_New (nombre_objeto): crea un objeto de una clase determinada

_free (nombre_objeto): destruye un objeto de una clase determinada

Objeto_evento: subrutina que se lleva a cabo cuando se produce un evento en un objeto. Ej: Button1_click() iniciaría la subrutina cuando hiciéramos click en el botón Button1.

3. GAMBAS

EJERCICIOS: VER EJEMPLOS DE PÁGINAS
47 Y 48 DE LIBRO-GAMBAS-2 (VER MENTOR)

3. GAMBAS

DECLARACIÓN DE MATRICES:

`DIM Nombre_Matriz[x,y,z....] AS tipo_de_variable`

Crearemos una matriz con el nombre `Nombre_Matriz` con las dimensiones `x,y,z...` para alojar variables del tipo que hayamos declarado.

EJ: `DIM Alumnos[10,2] AS String`

Hemos creado una matriz llamada `Alumnos` de dos dimensiones (10 Filas y 2 Columnas) donde podremos guardar datos de tipo `String` (por ejemplo, para guardar Nombre y Apellido1 de 10 alumnos)

3. GAMBAS

OPERACIONES MATEMÁTICAS:

- SUMA: $A+B$
- RESTA: $A-B$
- MULTIPLICACIÓN: $A*B$
- DIVISIÓN: A/B
- RESTO DE UNA DIVISIÓN: $A \text{ MOD } B$
- POTENCIA: A^B

MÁS OPERACIONES: PÁGINA 55 DEL LIBRO-
GAMBAS-2

3. GAMBAS

OPERACIONES LÓGICAS:

- A AND B (Y)
- A OR B (O)
- NOT A (NO)

3. GAMBAS

FUNCIÓN PRINT:

PRINT "HOLA QUE TAL" (Escribe en pantalla lo que pongamos entre comillas).

PRINT Variable (Escribe en pantalla el contenido de Variable)

3. GAMBAS

FUNCIÓN IF...THEN

IF Expresión THEN

;código

;a

;ejecutar

ELSE

;código alternativo

;a

;ejecutar

ENDIF

3. GAMBAS

**EJEMPLO: VER PÁGINA 61 DE LIBRO-
GAMBAS-2**

3. GAMBAS

FUNCIÓN SELECT

SELECT CASE Expresión

CASE Caso1

;código

;a

;ejecutar

CASE Caso2

;código alternativo

;a

;ejecutar

END SELECT

3. GAMBAS

**EJEMPLO: VER PÁGINA 62 DE LIBRO-
GAMBAS-2**

3. GAMBAS

FUNCIÓN FOR...NEXT

FOR variable=numeroa TO numerob

;código

;a

;ejecutar

NEXT variable

(Repite el código a ejecutar un número de veces determinado).

3. GAMBAS

**EJEMPLO: VER PÁGINA 63 DE LIBRO-
GAMBAS-2**

3. GAMBAS

FUNCIÓN WHILE o REPEAT

WHILE Condición (o REPEAT Condición)

;código

;a

;ejecutar

;mientras el código

;sea cierto

WEND

(Con REPEAT, el código se ejecutará al menos una vez).

3. GAMBAS

**EJEMPLO: VER PÁGINA 65 DE LIBRO-
GAMBAS-2**

3. GAMBAS

APLICACIONES GRÁFICAS:

SE COMPONEN DE UN **FORMULARIO** (VENTANA PRINCIPAL) QUE ALOJAN UNA SERIE DE CONTROLES, DEFINIDOS POR VARIAS PROPIEDADES.

LAS PROPIEDADES DE UN CONTROL SE NOMBRAN ASÍ: Nombredecontrol.Propiedad

EJ: Button1.Width=300

La propiedad Width (Ancho) del botón llamado Button1 es de 300 píxeles

3. GAMBAS

PROPIEDADES COMUNES:

X e Y: coordenadas horizontal y vertical

WIDTH y HEIGHT: Ancho y Alto del objeto

VISIBLE (true o false): Visibilidad del objeto

FOREGROUND (0-255): Color de primer plano

BACKGROUND (0-255): Color de fondo

VALUE: valor introducido en un control (por ejemplo, una etiqueta o caja de texto)

PICTURE: imagen que aparece en algunos controles

3. GAMBAS

CONTROLES BÁSICOS:

LABEL: Etiqueta

TEXTLABEL: Etiqueta que puede trabajar con HTML

TEXTBOX: Caja de texto

TEXTAREA: Caja de texto de varias líneas

BOTONES

CHECKBOX: Caja de verificación

RADIO BUTTON: Opción a seleccionar (sólo una en el grupo)

PICTURE BOX: Muestra una imagen

3. CAMBIOS CONTROLES BÁSICOS

PROGRESS BAR: Barra de progreso

SLIDER: Barra deslizante que el usuario puede modificar

MOVIE BOX: Puede mostrar animaciones GIF

SCROLL BAR: Barra para desplazarse a lo largo de una ventana

TIMER: Control invisible que controla el tiempo con una frecuencia determinada en la propiedad correspondiente, de modo que puede realizar una llamada a un evento cada ciclo.

3. GAMBAS

CLASE MESSAGE: PUEDE MOSTRAR UNA VENTANA CON UN TEXTO EN EL DESARROLLO DEL PROGRAMA

VER PÁGINA 101-106 DE LIBRO-GAMBAS-3
EN MENTOR

3. GAMBAS

CREACIÓN DE MENÚS: SE REALIZAN ORGANIZÁNDOLOS EN JERARQUÍA DE ÁRBOL. PARA ELLO, SÓLO HAY QUE PINCHAR CON EL BOTÓN DERECHO EN EL FORMULARIO Y ESCOGER LA OPCIÓN Edición de Menús.

VER PÁGINA 111-114 DE LIBRO-GAMBAS-3 EN MENTOR

3. GAMBAS

INTRODUCCIÓN AL DIBUJO CON PRIMITIVAS

VER PÁGINA 120-123 DE LIBRO-GAMBAS-3
EN MENTOR

3. GAMBAS

EVENTOS COMUNES A LA MAYORÍA DE LOS
CONTROLES:

- **CLICK**
- **DBLCLICK**
- **DRAG (ARRASTRAR)**
- **DRAGMOVE**
- **DROP**
- **MOVE**
- **GOTFOCUS**
- **TIMER**

3. GAMBAS

EVENTOS COMUNES A LA MAYORÍA DE LOS
CONTROLES:

- **LOSTFOCUS**
- **KEYPRESS**
- **KEYRELEASE**
- **LEAVE**
- **MOUSEDOWN**
- **MOUSEDRAG**
- **MOUSEMOVE**
- **MOUSEUP**
- **MOUSEWHEEL**

3. GAMBAS

TRABAJO (ENTREGAR A TRAVÉS DE MENTOR): EXPLICAR EN QUÉ CONSISTE AL MENOS CADA UNO DE LOS EVENTOS INDICADOS EN LAS ANTERIORES DIAPOSITIVAS.