

Introducción

Muy bien, apartir de ahora comenzaremos un largo camino hacia convertirnos en programadores de juegos, primero debes saber que la programación de juegos es una ciencia muy compleja, donde se relacionan gran cantidad de factores, por ejemplo programación basica, programación grafica, matematica, fisica, y un largo etc., por lo que no esperes que de la noche a la mañana hagas el proximo super exito de ventas. Ten en cuenta que a lo largo de este manual nos remontaremos a los años 80' hasta llegar mas o menos a mediados de la decada del 90' (jejje pensarás que estoy loco) lo que quiero decir es que empezaremos ha hacer juegos super sencillos, ya que de esta forma iremos atravesando los procesos por los que pasaron los programadores de aquella epoca.

El presente manual tiene como objetivo enseñar la programación en 2d, quizás en otra oportunidad creare un manual en 3d, pero por ahora con las 2d tenemos bastante ;) .

haaa me olvidaba nunca programe en blitz basic asi que este será un aprendizaje también para mi, asi que encargo a todos aquellos que lleguen a leer este manual que por favor, me hagan llegar todas las sugerencias, correcciones, o hasta cosas que le parezcan agregar a ajeansalle@gmail.com .

Primero lo primero: Como Descargar Blitz3d

Para descargar el programa Blitz 3d debemos dirigirnos a la página principal del programa en:

www.blitzbasic.com

y luego dirigimos al apartado **Products** y seleccionar **TRY NOW** del producto Blitz 3d

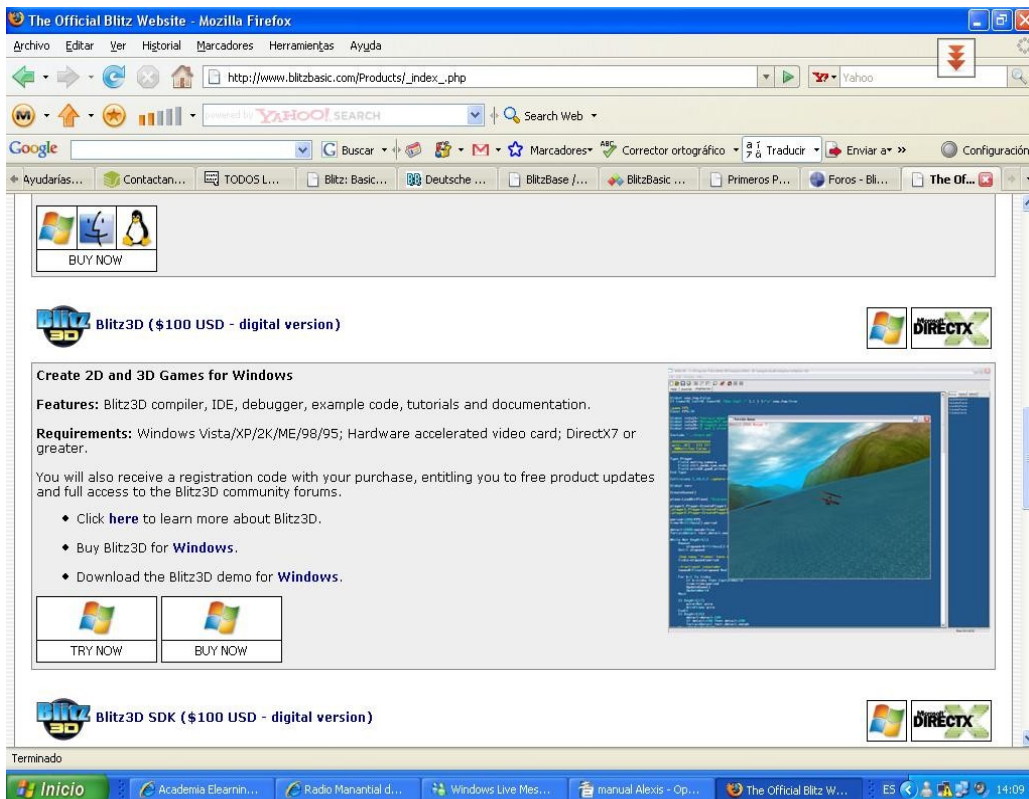


Fig. 1 Pantalla principal para descargar el programa Blitz 3d

Instalación del Programa

Una vez descargado el programa debemos ejecutar el archivo, por ejemplo **Blitz3DDemo183**, y luego seguir los pasos de instalación los cuales son los mismos que cualquier programa de Windows, en caso de que alguien tenga problemas con esto, les pido que me escriban así lo específico.

Restricciones de la versión de prueba:

La principal restricción del programa versión demo, es que no se pueden crear ejecutables (es decir que sí o si se debe tener el entorno para ejecutar los juegos) y tiene un límite de tamaño del archivo, lo cual para proyectos complejos no nos serviría, igualmente la versión Full, cuesta 100 USD (lo cual no es mucho) y se puede comprar por internet.

Para realizar este curso utilizaremos la versión de prueba del programa.

Pantalla Principal

Ni bien ejecutemos el programa veremos la pantalla principal del mismo, el cual esta compuesta por un menú, la barar de herramientas, con botones los cuales utilizaremos para hacer determinadas acciones, como por ejemplo ejecutar el juego ypor último encontramos el área central, que es donde escribiremos el código.

En el área central, cada vez que abramos el programa aparecerá la ayuda del mismo, podremos ver desde aca algunos ejemplos, acceder a los comandos y referencia del lenguaje .

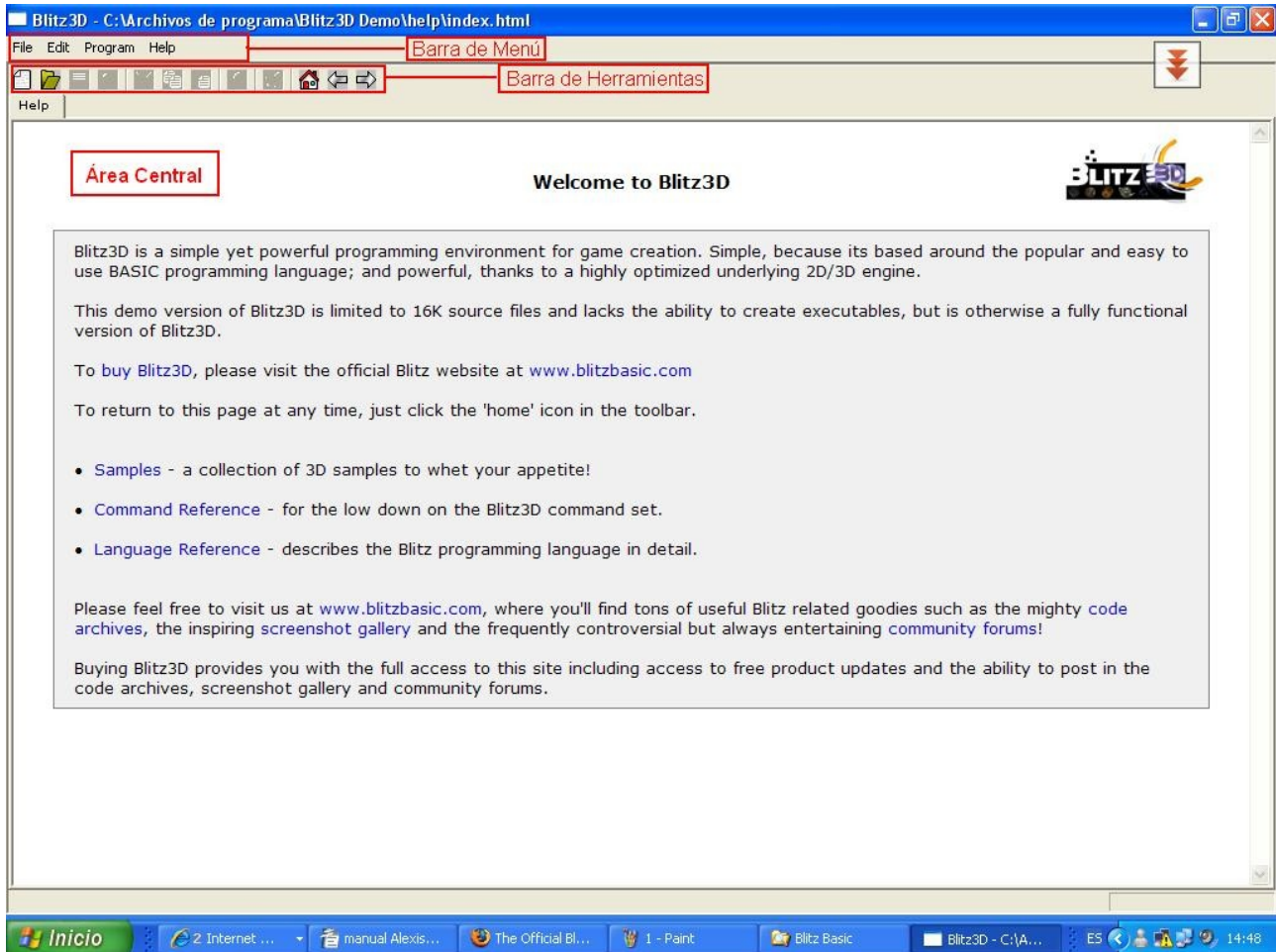


Fig 2 Pantalla Principal del programa Blitz 3d

EL entorno de Desarrollo “IDE”


Barra de Herramientas

Permite realizar operaciones basicas con el codigo Fuente, a continuación detallamos la función de cada botón



Fig. 3 Barra de Herramientas.

 Nuevo: Abre una nueva pestaña para escribir código, la pestaña actual no se cierra.

 Abir: Abre un archivo con código fuente que tengamos guardado.

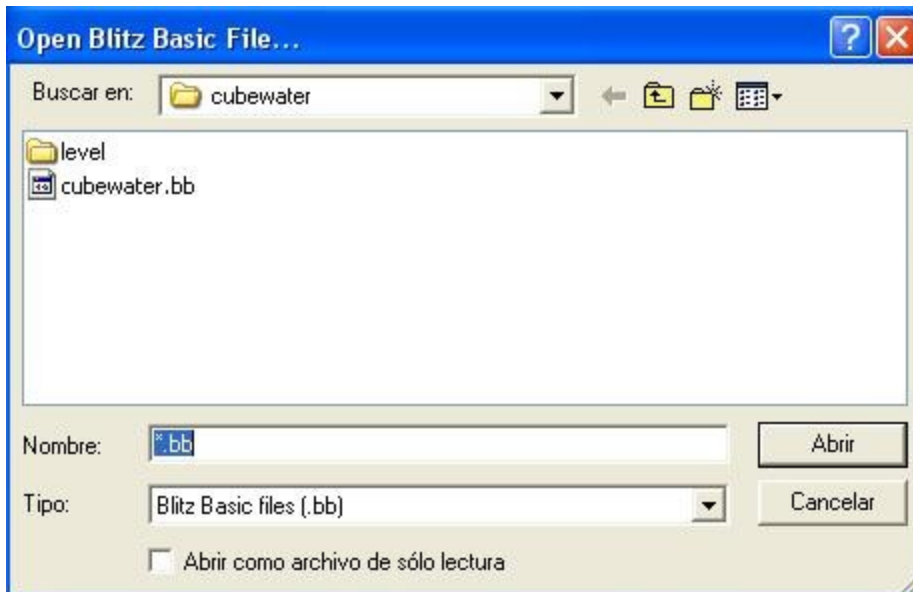






Fig 4. Ventana Abrir


 Guardar: Nos permite guardar el archivo en nuestra PC.

 Cerrar: Cierra la Pestaña Actual

 Cortar: cumple la misma función que en todos los programas windows, corta el texto seleccionado para que luego pueda ser colocado en otra posición o pestaña.

 Copiar: Cumple una función parecida a la anterior, pero no borra el texto que se copia.

 Pegar: Coloca en la posición donde se encuentra el cursor, el texto que fue copiado o cortado.

 Buscar : Permite buscar una palabra en el código fuente.

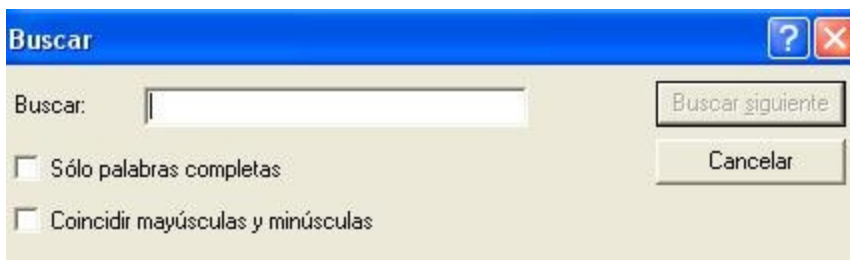




Fig. 5. Ventana Buscar

 Ejecutar: Ejecuta el código del programa, si todo sale bien tendríamos que ver nuestro juego =>

 Home: Nos envía a la pantalla de ayuda.

Barra de Menú

Archivo/File: Nos encontramos con los items comunes al manejo de archivos, por ejemplo nos permite abrir, guardar , cerrar, etc.



Fig. 6: Menú Archivo/File

Edición/Edit: Permite cortar, copiar, pegar, seleccionar o reemplazar el código fuente. También podemos desde este menú mostrar o esconder la barra de herramientas.

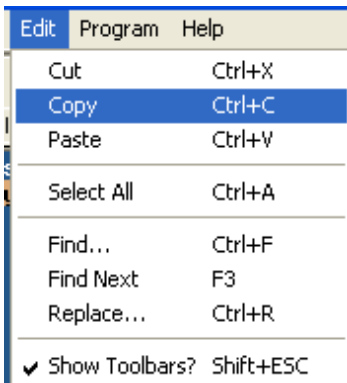


Fig. 7: Menú Edición/Edit

Programa/Program: Permite correr el programa, ejecutar el debugger para encontrar errores, crear el ejecutable,

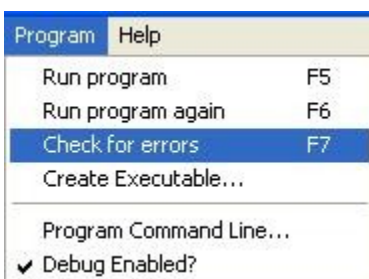


Fig. 8: Menú Programa/Program

Ayuda/Help: Muestra la ayuda del programa, muy util para poder aprender, y cuando nos olvidamos algún comando.

Conceptos Basicos de Blitz3d

¡Vamos a Programar!, pero... ¿Qué es un programa?

Un programa es un conjunto de instrucciones las cuales son ejecutadas por el compilador (este se encarga de traducir, las sentencias de un lenguaje al código máquina, es decir a 0 y 1) de forma secuencial.

Hay gran diversidad de lenguajes de Programación (estos los veremos en detalle en el modulo introducción a la programación de computadores) como por ejemplo Assembler, el cual es un lenguaje de bajo nivel o el C, C++, C#, Basic, Etc.

```
DATOS SEGMENT M1 DB "HOLA
MUNDO",10,13,"$" DATOS ENDS
CODIGO SEGMENT ASSUME
CS:CODIGO, DS:DATOS MOV
AX,DATOS MOV DS,AX LEA DX,M1
MOV AH,9 INT 21H MOV AX,4C00H INT
21H CODIGO ENDS END
```

Codigo1: Tipico programa que imprime en pantalla “Hola Mundo” realizado en assembler.

```
Print “Hola M undo”
```

Codigo 2: Hola mundo en el lenguaje Basic

En los cuadros con el código 1 y 2 vemos la gran diferencia entre un lenguaje de Alto nivel, como el basic, y uno de bajo nivel como el Assembler, podemos notar como para hacer una misma tarea, se simplifica utilizando los lenguajes de alto Nivel.

En el modulo correspondiente a introducción a la programación, veremos como se debe codificar correctamente, las herramientas con que cuentan los programadores, como son los diagramas de flujo y Pseudocódigo y muchas más cosas.

Codificación en Blitz 3d. ¡Comenzemos a Programar!

Muy bien antes de comenzar a crear nuestros programas debemos saber algunas cosas respecto al compilador de Blitz3d.

Comentarios

Los comentarios permiten ingresar líneas de texto, las cuales no serán tomadas por el compilador como código a ejecutarse, sino que servirán para el programador, para saber que hace una determinada línea o bloque de código. Los comentarios cumplen la función de documentar el código, para los primeros programas no será necesario, ya que serán super sencillos, pero es una buena práctica y se vuelve imprescindible en proyectos de mediana o gran envergadura. El símbolo que se usa para avisar al compilador que se trata de un comentario es el punto y coma (;).

```
;Imprime en la pantalla el mensaje Hola Mundo  
Print "Hola M undo"
```

Código 3: En este caso se explica que hace la sentencia "Print".

Palabras Reservadas Estas son palabras que utiliza el lenguaje internamente y no pueden ser utilizadas por identificadores (como por ejemplo: Variables, Nombres de funciones, Etiquetas etc)

```
After, And, Before, Case, Const, Data, Default, Delete, Dim, Each, Else,  
E lse, End, Endl f, Exit, False, Field, First, Float, For, Forever, Function,  
Global, G osub, Goto, If, Insert, Int, Last, Local, M od, New, Next, Not, Null,  
Or, Pi, Read, Repeat, Restore, Return, Sar, Select, Shl, Shr, Step, Str, Then,  
To, True, Type, Until, Wend, While, X or, I nclude
```

Cuadro 1: Palabras reservadas utilizadas por el lenguaje Blitz3d.

Identificadores

Los identificadores se utilizan para poder acceder a una porción de memoria con un determinado valor, es decir en lugar de acceder a la dirección de memoria 0001, accedemos a la variable "Vida". Resumiendo un poco, los identificadores serán los nombres de las variables, constantes, estructuras, Funciones, etc.

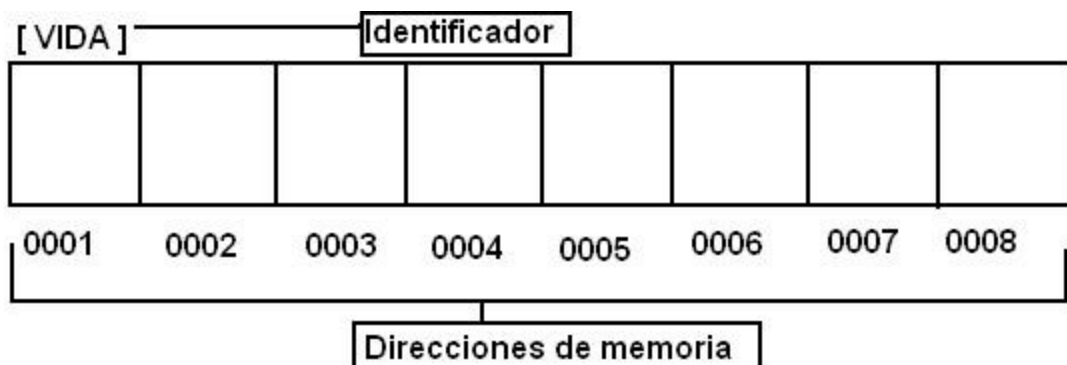


Fig 9: Dibujo de bancos de memoria de una pc, notese que en este caso la variable "Vida", se encuentra en la posición 0001

Los identificadores tienen una serie de reglas para definirlos:

- Debe empezar siempre por un carácter alfabético, aunque después puede estar precedido por Símbolos o Números.

Los siguientes son ejemplos de identificadores validos:

_Puntaje
Puntaje_1
Vida
vida

Los siguientes son ejemplos de identificadores no validos:

123Vida
@Vida

Debemos tener en cuenta que para B3d, es indistinto si escribimos en mayúsculas o minúsculas, por ejemplo VIDA y vida, se tratan de la misma variable.

También Blitz 3d reconoce si utilizas un mismo nombre para diferentes tipos de datos, por ejemplo si defines una variable Puntaje, y luego creas una función Puntaje, Blitz sabe en que momento utilizar cada una.

Tipos de Datos

Blitz trabaja con 3 tipos de datos:

Enteros o integer: Estos son los números que no tienen parte decimal, como por ejemplo -30, 158 y abarcan del rango -2147483648 al +2147483647

Punto flotante o Float: Estos son los números que tienen la parte decimal, por ejemplo 3.5 , 1.0. se debe tener en cuenta que a la hora de realizar calculos, estos se calculan mas lentos que los números enteros.

Cadena o string: son los valores alfanumericos, un dato de tipo string puede contener cualquier tipo de carácter, letras, simbolos o números, pero se debe tener en cuenta que con estos últimos no se podrán realizar calculos.

VARIABLES

Las variables son los tipos de datos mas simples. Son nombres o identificadores que el programador asigna a una porción de memoria donde almacenará valores numericos o caracteres.

Automaticamente, B3D, identifica el tipo de dato número entero. En caso de los valores de tipo números flotantes y alfanúmericos debe ser avisado. Para ello se necesita añadir un simbolo a continuación del identificador "\$" para texto y "#" para coma flotante.

Por Ejemplo en la declaración **Nivel=1**, B3D identifica que el valor de la variable "**Nivel**" es un entero, aunque se podria colocar Nivel%=1 (el simbolo % indica que se trata de un valor Entero).

Si quisieramos crear una variable de tipo string con el texto, "Jugador 1 ha muerto" deberiamos hacer:

Mensaje\$="Jugador 1 a Muerto"

Un ejemplo de declaración numérica con coma sería:
Velocidad# = 180.3

Para asignar una valor a una variable se debe utilizar el simbolo Igual (=).

Ejemplo de Asignación

Vida = 3

En las variables se pueden incluir operadores matematicos para poder realizar operaciones con estas, por ejemplo si nuestro personaje perdio una vida podriamos hacer.

Vida = Vida - 1

En este caso la variable vida pasa a tener 2

Alcance de las Variables

Las variables pueden ser:

Globales: Son aquellas variables que se declaran una sola vez, y en las diferentes partes del programa se van modificando, sin importar si se modifica en el programa principal o en las funciones.

Local: Las variables locales, tienen su campo de acción en las funciones que la declaran.

Para declarar una variable Global se debe hacer:

```
; Declaración de una variable global  
Global Vida = 3  
;Declaración de una variable local  
Local Vida=3
```

Código 4: Declaración del alcance de las variables

En caso de que la variable no se declare, la tomara por defecto como **Local**.

CONSTANTES: Las constantes son en realidad variables pero con valores fijos que no serán cambiados durante la ejecución del programa. Adquieren utilidad en aspectos del programa que nunca cambian como el valor del seno de 90° o una resolución de pantalla.

Para declarar una constante se utiliza la palabra Reservada “Const”

```
; Declaración de una constante  
Const Max_Velocidad = 200
```

Código 5: Declaración de una constante

Instrucciones del Lenguaje

Sentencias

Las sentencias son las ordenes que le damos al programa para que realice una determinada acción. Por ejemplo si quiero que el programa imprima en pantalla el Hola Mundo, debo indicarle al ordenador colocando la sentencia Print “Hola Mundo”

Las sentencias se pueden agrupar en diferentes categorías:

1. Sentencias de flujo de programa o condicionales
2. Sentencias para bucles
3. Sentencias para calculos o de asignación
4. Sentencia de Control de programa

Flujo de programa:

Estas sentencias permitirán cambiar el rumbo del programa después de comprobar y evaluar si se cumple algún tipo de condicion .

Estas condiciones estan formadas por expresiones que contiene los operadores de comparación, por ejemplo una expresión de este tipo podria ser Vida < 1.

| Operadores de Comparación |
|----------------------------------|
| Mayor que: > |
| Mayor o Igual que: >= |
| Menor que: < |
| Menor o Igual que: <= |
| Igual que: = |
| Distinto que: <> |

Cuadro 2: Operadores de comparación

| Operadores de Lógicos |
|------------------------------|
| NOT |
| AND |
| OR |
| XOR |

Cuadro3: operadores lógicos

En los operadores de comparación se necesitan dos valores para comparar:

Por ejemplo:

Vida>0

También disponemos de operadores lógicos como: **NOT**(compara que no se cumpla una condición), **AND**(compara que dos expresiones son ciertas), **OR** (compara que al menos una de las dos expresiones es cierta) **XOR** (compara que solo una de las dos expresiones es cierta).

Siempre que realicemos estas comprobaciones, obtendremos como resultado los calores 0 (false) o 1 (True). Además se pueden encadenar expresiones condicionales separadas por operadores lógicos y englobar expresiones dentro de otras utilizando parentesis, por ejemplo:

| |
|--|
| (Vida<1 or energia<100) and puntos = 0 |
|--|

Cuadro 4: Expresiones encadenas

En este caso se evalúa primero lo que está entre paréntesis y el resultado se comparará con la siguiente expresión; así, se cumplirá la condición si puntos vale 0 y vida es menor que 1 o si puntos vale 0 y energía es menor a 100.

Hay que tener en cuenta que los operadores lógicos convierten sus operandos a valores enteros y el resultado que producen también.

Estructura IF ...Then

Su construcción Básica es la siguiente:

```
If expresión then sentencias1
Else sentencias2
```

o también

```
If expresión
    sentencias1
Else
    sentencias2
Endif
```

Cualquiera de las dos formas es válida, lo que hace esta estructura es evaluar si la expresión es verdadera, en ese caso ejecuta la sentencias1; si llegara a ser falsa ejecuta las sentencias2.

```
If vida<1 Then jugador_muere
Else vida=vida-1
```

Código 6: Ejemplo del uso de la estructura If

Lo que hace el código 6 es Si (If) vida es menor a 1 (vida<1) entonces (Then) “jugador_muere”; Si no (Else) a vida le resta 1 (vida=vida-1)

Ae pueden encadenar varias estructuras if.

```
If expresión1
    sentencias1
Elseif expresión2
    sentencias2
Else
    sentencias3
Endif
```

En esta estructura encadenada, las “sentencias3” se ejecutarán solamente si la “expresion1” del if o la “expresion2” del ElseIf Fueran Verdaderas (True)

```

If nombre$="Juan" Then
  Print "Saludos, Juan Gonzalez"
Else If nombre$="Pedro"
  Print "Saludos, Pedro Ramirez"
Else
  Print "No me acuerdo tu apellido"
Endif

```

Código 7: Ejemplo del uso de la estructura If encadenada.

En el código 7 el programa se fija si el nombre ingresado por el usuario es “**Juan**”, en caso de que sea verdadero mostrara “**Saludos juan gonzalez**”, caso contrario entrará en el **Else If** y preguntará si es “**Pedro**” si llegara a ser verdadero mostrará “**Saludos, Pedro Ramirez**” por último si no es ninguno de los 2 mostrará en pantalla “**No me acuerdo de tu apellido**”.

¡A PROGRAMAR!

Muy bien vamos a ver un ejemplo del uso del If

```

nombre$=Input("Como es tu nombre: ")
If nombre$= "Juan"
Print "Hola Juan"
Else
Print "No te Conozco"
EndIf

```

Ejemplo1: El codigo de nuestro primer programa

En este caso se utiliza una función nueva la cual es Input, esta función lo que hace es pedirle al usuario que ingrese un texto, cuando este lo ingrese se lo asignamos automáticamente a la variable nombre\$.

Por último hacemos la comparación si la variable **nombre\$** tiene el texto **juan**, Mostrará “**Hola Juan**”, caso contrario mostrará “**No te conozco**”

Ejecutemos el programa presionando la tecla **F5** o en el botón “**Run**” para ver que es lo que hace nuestro programa.

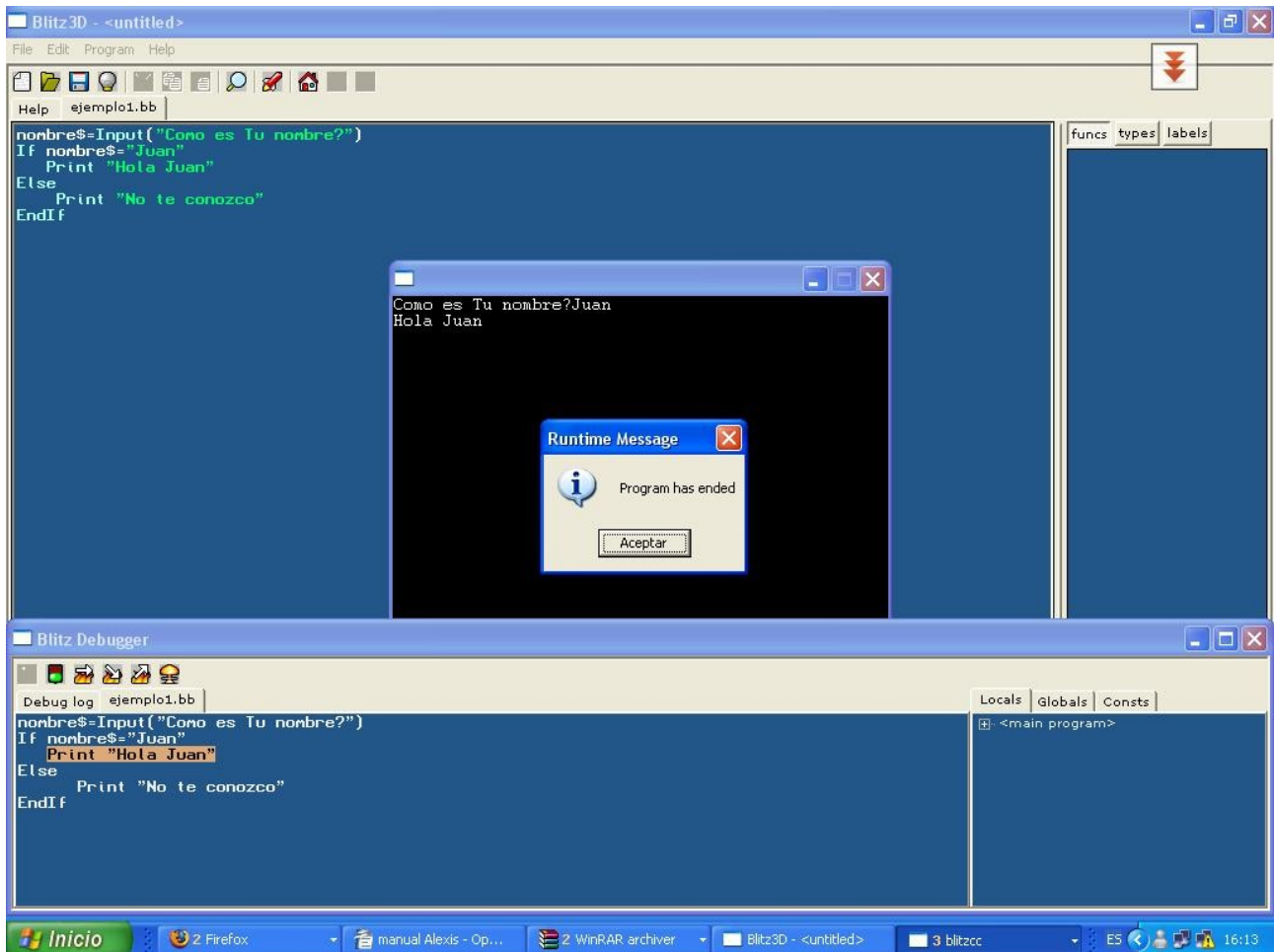


Fig 10: Nuestro primer programa Ejecutandose

Estructura Select ... Case

Select expresión

Case expresiones1

case expresiones2

default expresiones3

End Select

Este tipo de estructuras nos ayudan a evitar un gran numero de estructuras If... Then en nuestro programa. Imaginense que quieren dibujar en pantalla diferentes explosiones dependiendo del tipo de disparo que se haya efectuado. Esto se podría realizar mediante muchos if, por ejemplo,

```

If disparo=1 then
dibujar_explosion1
If disparo2 Then
dibujar_explosion2
If disparo3 then
dibujar_explosion3

```

Realmente este sistema es algo engorroso y poco elegante. La estructura "Select... Case" es una alternativa más apropiada.

```

Select disparo; Según el valor de disparo
Case 1 dibujar_explosion1; En caso de que disparo=1 dibujamos la explosión1
Case 2 dibujar_explosion2; En caso de que disparo=2 dibujamos la explosión2
Default dibujar_explosion3; En el caso restante es decir disparo=3 dibujamos la explosión3
End Select

```

Código 9: Uso de la estructura Case

¡A PROGRAMAR!

Ahora veremos el uso de la estructura case, y aprovecharemos a aplicar lo ya visto hasta aquí.

```

; Pedimos al usuario que ingrese el número del mes

mes=Input("Ingrese el número de Mes ")

Select mes
  Case 1 nombre_mes$="Enero"
  Case 2 nombre_mes$="Febrero"
  Case 3 nombre_mes$="Marzo"
  Case 4 nombre_mes$="Abril"
  Case 5 nombre_mes$="Mayo"
  Case 6 nombre_mes$="Junio"
  Case 7 nombre_mes$="Julio"
  Case 8 nombre_mes$="Agosto"
  Case 9 nombre_mes$="Septiembre"
  Case 10 nombre_mes$="Octubre"
  Case 11 nombre_mes$="Noviembre"
  Case 12 nombre_mes$="Diciembre"
  Default nombre_mes$="El mes ingresado no existe"
End Select

If (mes >= 1) And (mes <=12)
  Print "El mes " + mes + " corresponde a " + nombre_mes$
Else
  Print " " + nombre_mes$
End If

```

Ejemplo 2: Uso de la estructura case

En este pequeño ejemplo creamos un programita para decirle al usuario el nombre del mes, a partir de un número ingresado por el.

Lo primero que vemos es que se le pide al usuario que ingrese un número.

```
mes=Input("Ingrese el número de Mes ")
```

Luego entramos en el Select

```
Select mes
```

```

  Case 1 nombre_mes$="Enero"
  Case 2 nombre_mes$="Febrero"
  Case 3 nombre_mes$="Marzo"
  Case 4 nombre_mes$="Abril"

```

```

Case 5 nombre_mes$="Mayo"
Case 6 nombre_mes$="Junio"
Case 7 nombre_mes$="Julio"
Case 8 nombre_mes$="Agosto"
Case 9 nombre_mes$="Septiembre"
Case 10 nombre_mes$="Octubre"
Case 11 nombre_mes$="Noviembre"
Case 12 nombre_mes$="Diciembre"
Default nombre_mes$="El mes ingresado no existe"

```

```
End Select
```

Lo que hace el programa es preguntar el contenido de la variable mes (variable entera) en caso de que su valor sea, por ejemplo, 1 asignamos a la variable **nombre_mes\$**, el valor “**Enero**”. Entonces en la variable **nombre_mes\$** tendremos el nombre del mes dependiendo del número que haya ingresado la persona, ahora que pasa si la persona ingresa un número erróneo? Por ejemplo el 13? en este caso entrará en la línea:

```
Default nombre_mes$="El mes ingresado no existe"
```

De esta forma la variable tendrá el texto que avisará de que se ha ingresado mal el número del mes.

Hasta acá el programa está haciendo lo que queremos, pero aún no nos muestra nada, para que nos muestre deberemos escribir el siguiente código:

```

If (mes >= 1) And (mes <=12)
    Print "El mes "+ mes + " corresponde a " + nombre_mes$
Else
    Print " " + nombre_mes$
End If

```

Ups, y acá usamos el if (vean como de apoco vamos aplicando lo aprendido! =))

Bien lo que hacemos en este bloque de código, es ver si el número que ingreso es correcto (Es muy probable que el usuario, sufra equivocaciones, así que nosotros deberemos tratar por todos los medios, que el programa que diseñemos “Avisar de estas equivocaciones”).

El If, va a preguntar si el valor ingresado en la variable mes, se encuentra en el rango del 1 al 12, si es así imprime en pantalla un mensaje con el nombre del mes. Ahora...ese print está bastante raro...no? Expliquemolo.

```
Print "El mes "+ mes + " corresponde a " + nombre_mes$
```

En este caso se está combinando un texto con variables, el compilador tomará como texto lo que está entre comillas dobles, el resto lo que está entre el símbolo más (+) es la variable de esta forma el mensaje que se mostrará será como el siguiente:

El Mes 1 corresponde a Enero

GoTo Label: Esta Sentencia desvia el flujo del programa hasta la etiqueta definida luego de la palabra reservada.

```
Print "el programa comienza"
Goto etiqueta1
Print "esto no se imprimira nunca en pantalla"

.etiqueta1
Print "¡Saltamos hasta aqui!"

;espera que pulse ESC para acabar
While Not KeyHit(1)
Wend
```

Ejemplo5: Código donde se puede ver el uso de la sentencia Goto

Fijemonos que es lo que hace el programa, primero imprime el mensaje “**El programa comienza**”, luego usamos la sentencia **goto etiqueta1**, es decir que automáticamente ira a la línea 5 del código, luego imprimira “**¡Saltamos hasta aqui!**” el while final se utiliza para que el programa no termine la ejecución hasta que no le digamos.

While Not KeyHit(1)

Wend

lo que quiere decir es “Mientras no este presionada la tecla ESC.” se mantiene el código en ejecución.

LA función **KeyHit(numero de tecla)** se utiliza para saber si una tecla es presionada, el parametro número tecla es un valor numérico y cada número representa una tecla, en este caso la tecla 1 es ESC.

Sentencias para Bucles

En este tipo de estructura se realiza un bucle de las sentencias hasta cumplirse determinada condición

Estructura While... Wend

construcción básica

While expresión

Sentencias

Wend

Mientras (While) se cumpla de la expresión, se ejecutan las sentencias.

Esta estructura es utilizada cuando queremos que se realicen operaciones sólo cuando se cumpla cierta condición.

Antes de entrar en la estructura para realizar el grupo de sentencias, se evalúa la expresión y

mientras sea verdadera, se permanecerá en el interior de la estructura .

En el ejemplo:

```

vida_jugador=3
While Vida_Jugador>0
    ;Dibuja y mueve jugador
    ;dibuja y mueve los enemigos
    ;Atacan los enemigos al jugador
    If disparo_enemigo colisiona con jugador then
        Vida_jugador=Vida_Jugador-1
Wend
muere_Jugador

```

Código 10: ejemplo del uso del While

Asignamos 3 vidas al jugador. Mientras(While) las “**Vidas del Juador**” no sean 0, “Dibujaremos y moveremos al jugador y a los enemigos”, “Atacarán los enemigos”. Si un “Disparo del enemigo” alcanza “Al jugador”, éste pierde una vida. Se vuelve a Preguntar si la “vida del jugador” no es 0; como ahora vale 2 se vuelve a ejecutar las acciones anteriores y así hasta que las vidas del jugador valgan 0. Se saldrá de la estructura y se ejecutará “Muere Jugador”.

¡A PROGRAMAR!

Escribamos el siguiente código en Blitz

```

cont=0
While cont<=100
    Print cont
    cont=cont+1
Wend

```

ejemplo 3: uso del while

Este ejemplo es bastante sencillo de entender lo que hace el programa es inicializar la variable cont, en 0 luego entra en el bucle, el cual se hará mientras la variable cont sea <=100. una vez que entro en el bucle veremos que imprimirá en pantalla el primer número, luego a la variable cont le sumará 1 y volverá a preguntar si cont es <=100, volviéndose a ejecutar la sentencia si es verdadero. En nuestro ejemplo se imprimen los números del 1 al 100, consecutivamente.

Estructura For ... Next

For variable = valor Inicial **To** valor final **Step** salto en el incremento del valor

Sentencias

Next

Este tipo de estructuras para bucles funciona preguntando por la evolución del valor de una variable. Antes de entrar en el bucle, se asigna el “valor inicial” a la “variable” y se entra en el bucle.

Una vez ejecutadas las “sentencias” con la función “next” se regresa al for. Se incrementa una unidad el valor de “variable” (opcionalmente se puede incrementar en las unidades que se indique en “step” (por ejemplo -1, para decrementar el bucle) y se vuelve a entrar en el bucle. Esta operación se repetirá hasta que el valor contenido de la “variable” alcance el “valor final”. Por

ejemplo, queremos contar del 1 al 10 de dos en dos e imprimirlo en pantalla:

```
For contador = 1 to 10 step 2
    Print contador
Next
```

Codigo 11: Uso del for con Step

Para realizar la cuenta pero al reves, se deben cambiar el “valor Final” por el “Valor Inicial” y colocar “Step” en negativo. Por ejemplo, si queremos hacer una cuenta atrás del 10 al 1:

```
For contador = 10 to 1 step -1
    Print contador
Next
```

Codigo 11: decrementando el contador en un bucle For

También puede haber bucles dentro de otros bucles. El bucle interior se repetirá tantas veces como indique el bucle exterior.

```
For y = 1 to 600
    For x = 1 to 800
        dibujar_puntp x,y
    Next
Next
```

Codigo 12: Ejemplo de Bucles anidados

```
Print "-----PRograma Tabla Multiplicar-----"
For y = 1 To 10
    For x = 1 To 10
        Total_Tabla=x*y
        Print y + "*" + x + "Es igual a" + Total_Tabla
        Continuar$=Input("Presione Enter para continuar")
    Next
Next
```

Ejemplo 3: Uso del bucle For, para generar una tabla de multiplicar.

Muy bien, expliquemos el codigo del ejemplo 3 en este caso la primera linea no deberia ser necesaria explicar a esta altura del manual no? La segunda linea comienza a usarse el bucle, lo que queremos hacer es una tabla de multiplicar del 1 al 10 entonces si pensamos como podriamos hacerla con lapiz y papel, seria:

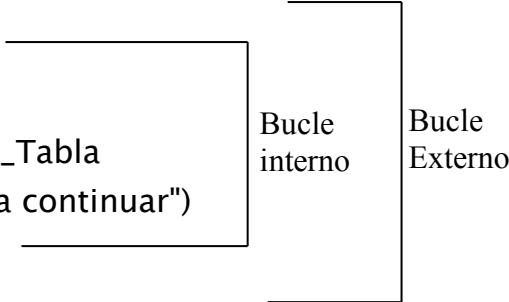
```
1 x1
1x 2
1 x 2
....
1x8
1x9
1x10
2 x 1
2x2
```

Si nos fijamos el operando de la izquierda se mantiene en un número fijo, hasta que el de la derecha va variando del 1 al 10. para representar esto en un programa de Computación, podremos hacerlo usando dos bucles For, anidados.

```

For Y = 1 To 10
  For x = 1 To 10
    Total_Tabla=x*y
    Print y + "*" + x + "Es igual a" + Total_Tabla
    Continuar$=Input("Presione Enter para continuar")
  Next
Next

```



En este bucle fijese que cuando $Y = 1$, entra al otro bucle donde la variable X valdra 1 tambien, pero cuando termine de ejecutarse el bucle interno, incrementara el valor de X a 2, pero y seguira valiendo 1, hasta que se cumpla que x valga 10 entonces recien ahi Y se incrementa en uno, valiendo de esta forma $Y=2$, luego vuelve a entrar en el bucle de la X y vuelve a hacer lo mismo que lo anterior, asi sucesivamente hasta que $Y=10$ con lo que sale del bucle.

Fijense que para podere realizar la multiplicación se hace $X*Y$ y eso se guarda en la variable `Total_Tabla`.

En la linea siguiente se muestra el resultado de la multiplicación. Por ultimo agregue una sentencia `Input`, para que el usuario tenga que presionar una tecla para poder continuar y pueda ir viendo los resultados de a poco.

En B3D existe un tipo de bucle For que nos permite recorrer cada uno de los objetos que forman parte de una estructura de datos. Se trata de la estructura `For...Each`

```

For variable = Each nombre de la estructura de datos o tipo de datos
  Sentencias
Next

```

Es muy utilizado para actualizar grandes grupos de objetos. Pero explicaremos el uso de esta sentencia con detalle cuando hablemos de estructuras de datos.

Estructura Repeat... Until

```

Repeat
  Sentencia
Until Expresión

```

Este bucle de repetición, permite realizar una seria de sentencias hasta que se cumpla una condición dada. La diferencia con el bucle "While" es que en este bucle se ingresa aunque sea una vez.

```

X=10
Repeat
  Print x
  x=x+1
Until X=20

```

Codigo 12: Uso del bucle repeat

```

X=10
Repeat
  Print x
  x=x+1
Until x=20

```

10
11
12
13
14
15
16
17
18
19

Ejemplo 4: aquí vemos el código del ejemplo anterior y su salida en la consola.

Muy bien hasta acá tenemos bastante para ir probando, quizás esta sea una de las unidades más aburridas, porque ni siquiera hemos visto cómo imprimir un punto de color en pantalla, pero es importante que puedas ver los fundamentos básicos y cómo tratar los problemas. Ya que si sabemos cómo solucionar un problema, luego trasladarlo a la PC, es más fácil.

ARRAY

Una matriz es en realidad un identificador que se relaciona con muchas posiciones de memoria. Estos arrays pueden ser de una (Vector) o dos dimensiones (Matriz).

Para definir un array se utiliza la palabra reservada Dim seguida del identificador y entre paréntesis la cantidad de celdas de memoria que se le asignará a ese identificador. Por ejemplo, vamos a definir una matriz unidimensional llamada “enemigos” que contenga 5 celdas de memoria:

Dim Tabla(4)

se coloca entre paréntesis un 4 en lugar de un 5 porque la primera casilla empieza en 0 y no en el 1, así pues tenemos 5 valores desde 0 hasta 4.

Podemos asignar valores a cada posición de memoria de la tabla en el momento de la declaración:

Dim Tabla(4) = 12, 30, 40, 50, 68

Esto significa que en la posición 0 de tabla hay un 12, en la 1 de Tabla hay un 30 y así sucesivamente.

Para pasar un valor cualquiera de la tabla a otra variable, por ejemplo el valor 50 a “fuerza”, escribiríamos:

Fuerza=Tabla(3)

Porque la posición 3 (en realidad es la cuarta) contiene el valor 50.

¡A PROGRAMA!

Bueno veamos un ejemplo un poco más interesante que los anteriores, escribo el código y ahora les explico qué es lo que vamos a hacer.

```

;ponemos el modo grafico en 800 x 600, para lograr tener la pantalla mas grande
Graphics 800,600

;*****
;                               Modulo correspondiente al menú
;*****
.menu
; borramos todo el texto que hay en pantalla
Cls
;ubicamos el cursor en la posición x=30 e y=20 , asi escribimos los textos en ese lugar
Locate 30,20

;imprimimos el menú
Print "----- Sistema de Alumnos -----"

Print " "
Print "1-Cargar alumnos"
Print " "
Print "2 -Consultar datos de alumnos"
Print " "

;preguntamos al usuario a donde quiere ir, si presiona 1 va a la seccion carga y si presiona 2
;va a la sección lectura
resp$=Input("ingrese número de opción ")
If resp$="1"
    Goto carga
Else
    Goto leer
EndIf

;*****
;                               Modulo correspondiente a la carga de alumnos
;*****
.carga
Cls
Locate 30,20
Print "----- Cargar Alumnos -----"
Dim alumno$(4)
Print " "
;llenamos el vector desde la posición 0 a la 4
For i=0 To 4
    alumno$(i)=Input("Ingrese nombre de alumno Número: " + i + " ")
Next
;borramos todo lo que hay en pantalla
Cls
Locate 30,20
Print "----- Listado de Alumnos Cargados ----- "

; Leemos el vector antes cargado y lo mostramos en pantalla
For i=0 To 4
    Print " "
    Print alumno$(i)
Next

Print " "
Print "ha cargado todos los alumnos"

Print " "

;Pedimos al usuario que ingrese enter para ir al menú principal
Input("presione la tecla ENTER para volver al menú")
If KeyHit(28)
    Goto menu
EndIf

;*****
;                               Modulo correspondiente a la lectura de alumnos
;*****
.leer
Cls
Locate 30,20
Print "----- Leer Alumnos -----"
;pedimos al usuario que ingrese la posición que quiere leer
pos=Input ("ingrese posición que desea leer")

;mostramos el contenido de la posición ingresada por el usuario
Print "ha leído la posición " + pos + " el alumno que se encuentra en esa posición se llama "
; muestra el texto hasta que la persona presione ESC
While Not KeyHit(1)
Wend

```

Muy bien, ahora pasemos a explicar lo que hace este pequeño sistema la primera instrucción nueva que aparece es :

Graphics 800,600

Lo que hace esta sentencia es colocar el modo grafico en una resolución de 800 x 600, lo primero que notaremos al cambiar la resolución, es que la ventana se agranda.

Otra resolución por ejemplo es 640 x 480.

estos dos parametros son los minimos que necesita esta función, pero tambien tiene otros parametros opcionales que pueden ser incluidos o no.

La función se compone de:

Graphics width, height, color depth,[mode]

Width y height: estos valores representan al tamaño de la pantalla expresado en pixeles.

Color Depth: representa la calidad del color expreado en bit (0,16,24 o 32 bit)

Mode: Corresponde al modo de video

0 : auto – Modo ventana en debug, Pantalla completa cuando no se usa el debug (este es el modo por defecto)

1 : Modo pantalla completa

2 : Mode ventana

3 : Modo ventana personalizada

Es decir que si ponemos:

Graphics 800,600 ,16,1

Estamos indicando que la resolución es de 800x600 con una profundidad de color de 16 bit y en modo fullscreen.

Luego el comando **CLS**, es utilizado para borrar todo el texto que hay en pantalla.

Luego de borrar la pantalla colocamos el cursor en la posición X=30 y=20 de la pantalla

Locate 30,20

Esto es para que el texto que se vaya escribiendo empiece siempre en esa posición, caso contrario veremos como el texto a medida que aparece se va bajando.

Una vez ubicado el cursor de texto entraremos en el proceso para imprimir el menú, no tiene mucha complicación, fíjense que las líneas:

```

resp$=Input("ingrese número de opción ")
  If resp$="1"
    Goto carga
  Else
    Goto leer
  EndIf

```

lo único que hace es tomar del teclado la opción elegida por el usuario y ver, si es 1 salta a la carga de datos y si es 2 (o otro carácter) iría a la sección de lectura de un dato.

Sección Carga de Alumnos

Bien esta sección es la que se encarga de cargar los alumnos al vector.

Lo primero que se hace es definir el array alumno, en este caso esta definido como string con 5 posiciones (recordemos de la 0 a la 5)

```
Dim alumno$(4)
```

luego procedemos al llenado del array

```

For i=0 To 4
  alumno$(i)=Input("Ingrese nombre de alumno Número: " + i + " ")
Next

```

utilizamos un para (FOR) que repetira 5 veces el bucle. La sentencia que se encuentra dentro del bucle lo unico que hace es leer el dato ingresado y colocarlo en la posición correspondiente del vector, notese que dentro de los parentesis de **“alumno\$”** Se encuentra la variable **i**, esto sirve para que cada vez que ingrese en el bucle se llene una posición diferente del vector.

Por ejemplo la primera vez se llenara la posición 0, la segunda vez la 1 y así sucesivamente.

Con ese código ya pudimos cargar el vector, ahora listaremos los nombres cargados, con el siguiente código.

```

For i=0 To 4
  Print " "
  Print alumno$(i)
Next

```

fijemonos, que este código no nos debería de ser difícil de entender, lo que hace es nuevamente con un bucle For, recorrer el vector y lo va mostrando (**Print alumno\$(i)**)

Modulo Lectura de alumnos

Bueno veremos que para leer el dato de una posición del vector no es muy complicado, lo primero que debemos hacer es pedir al usuario que ingrese la posición que desea leer.

```
pos=Input ("ingrese posición que desea leer")
```

En pos se guardará el número elegido por el usuario.

Luego deberemos mostrar el nombre que se encuentra en una determinada posición.

```
Print "ha leído la posición " + pos + " el alumno que se encuentra en esa posición se llama " + alumno$(pos)
```

Bien para implementar esto, no tenemos más que hacer que colocar un print, con las variable correspondiente (pos) la cual contiene el valor ingresado por el usuario, por ejemplo si el usuario eligió el número 3 imprimiría en pantalla;

Ha leído la posición **3** El Alumno que se encuentra en esa posición se llama **Juan**

Tarea para el hogar!

Bien el programa está funcional, pero sería bueno que se implemente lo siguiente:

- Cuando se elige una opción (la 1 o la 2) que permita elegir esas 2 si se presiona alguna otra tecla que me muestre un mensaje de error y me vuelva al menú para ingresar el número correcto.
- Que el menú tenga una tercera opción que lo que haga sea salir del programa.
- Cuando quiera leer la posición de un alumno, si esta no existiera, me de un mensaje de error.

Y si tienen alguna cosita más para mejorar el programita posteenlo en el foro así todos aprendemos!

Array Bidimensional

Un array bidimensional (tabla o matriz) es un array con dos índices.

Para localizar o almacenar un valor en el array se deben especificar dos posiciones (dos subíndices), uno para la fila y otro para la columna. Los elementos se referencian con el formato:

Tabla(2,4) elemento de la fila 3 y columna 4.

| | 1 | 2 | 3 | 4 |
|---|--------|--------|---|--------|
| 1 | T(1,1) | T(1,2) | | T(1,4) |
| 2 | | | | |
| 3 | | | | |
| 4 | T(4,1) | | | T(4,4) |

Fig. 11: Array de dos dimensiones T con 4 filas y 4 columnas.

Como definir un array

para definir un array de dos dimensiones debemos hacerlo de la siguiente manera:

Dim notas(4,3)

en este caso estamos definiendo que nuestro array tendra 5 filas (de la 0 a la 4) y 4 columnas (de la 0 a la 3) con lo cual nos da un total de 20 posiciones para almacenar valores (5x4=20)

----- FIN???? -----

proximamente seguiremos con el tema de los array, lo que pasa es que prometi en el foro que subiria el manual este fin de semana!! y me gusta cumplir!! saludos! Foreros!!

espero que todo esto este bastante bien explicado cualquier consulta por favor no duden avisarme en el foro

algunas partes fueron sacadas del manual de diseño y programación de juegos! ,me gusto mucho como estaban explicadas ais que las puse..

Recuerden, que estoy aprendiendo con ustedes asi que cualquier cosa que quieran agregar, corregir o explicar mejor, por favor posteenlo en el foro!!

saludos!!

Alexis Jeansalle (Crashon182)