¡Vamos a programar!

Phrogram KPL v 2

Pablo Flórez Valbuena

Lidia Getino Llamas

¡Vamos a programar!

Phrogram KPL v 2

Materiales educativos multimedia

Año 2007

Consejería de Educación

Junta de Castilla y León

Índice

Presentación

Estamos en el siglo XXI y el desarrollo de las tecnologías informáticas es una realidad que forma parte de nuestro presente y seguirá formando parte de nuestro futuro.

Pero,... ¿en qué consiste la enseñanza de la informática?. En un principio tenía que ver con la enseñanza de la programación, principalmente en BASIC, el más sencillo y utilizado de los lenguajes informáticos de alto nivel.

Con el paso del tiempo han surgido dudas a este respecto: ¿por qué enseñar a programar cuando hay en el mercado tanto software programado?. Parece que no hay necesidad de aprender a programar, salvo como parte de una preparación específica en una carrera de programador, que requiere una formación muy superior a la que pueden proporcionar las escuelas e institutos. En los últimos años, la enseñanza de la informática se ha centrado en una formación a nivel de usuario, se enseña al alumnado a ser consumidor del software que otros han diseñado, pero esto puede no ser suficiente.

Algunos pensamos que la enseñanza de la informática es algo más amplio, requiere el desarrollo de una capacidad lógica y de síntesis que debe ir construyéndose poco a poco, desde la etapa escolar. Es importante que los alumnos sean "usuarios", pero no es menos importante que sean capaces de pensar, deducir, relacionar y aplicar conceptos con el fin de diseñar, es decir de construir conocimiento. Sino es así,... ¿quién diseñará el software en el futuro, si sólo creamos usuarios de aplicaciones?.

Hoy en día, los lenguajes predominantes son Java, C# y VBasic.NET, pero no fueron diseñados para que un principiante se inicie en el mundo de la programación. Así pues ha nacido un nuevo lenguaje, el KPL (Kid´s Programming Language -Lenguaje de Programación para Niños-), pensado y diseñado para la iniciación de los adolescentes en la programación, creado por *Morrison Schwartz, Inc.* Es un lenguaje de programación que corre con Windows y que, por encima de todo es,

¡SENCILLO, FORMATIVO Y... DIVERTIDO!

El trabajo que aquí presentamos es innovador en cuanto a su temática y planteamiento, es fácil de manejar y requiere una necesaria reflexión en el aula.

Hemos utilizado la versión 2 de KPL, es decir, Phrogram (aparecida en verano de 2006 y disponible sin ningún cargo al igual que KPL v 1.1).

Esperamos que os animéis a sumergiros en este apasionante mundo de la programación. La guía didáctica os indicará la forma de hacerlo, os dará una visión de lo que hemos pretendido conseguir con este trabajo y de cómo podéis comenzar programar. Hemos diseñando una serie de actividades, ejemplificadas con cálculos y operaciones matemáticas, que de manera progresiva os acercarán a algunas de las estructuras y herramientas básicas del KPL. Así pues **ÁNIMO Y...**

¡VAMOS A PROGRAMAR!

Guía didáctica

Objetivos

Este material pretende introducir de manera progresiva algunas de las estructuras y herramientas básicas que tiene disponibles el lenguaje de programación KPL *Kid´s Programming Language* (Lenguaje de Programación para niños) para que cualquier alumno de enseñanza no universitaria pueda animarse a conocer el apasionante mundo de la programación.

Con todo ello, no se pretende enseñar un lenguaje de programación, sino que se trata de que los alumnos adquieran una correcta y precisa metodología de trabajo, expresándose de manera lógica y formal, independientemente del lenguaje de programación utilizado.

Así pues los objetivos fundamentales que se pretenden conseguir, con el aprendizaje del KPL son:

1. Favorecer el desarrollo de la inteligencia, con especial atención la creatividad y la capacidad lógico-deductiva.
2. Profundizar en la didáctica específica del área de matemáticas.
3. Potenciar la investigación y la cultura emprendedora en relación con las TIC.
4. Conocer y utilizar vocabulario y/o términos en lengua inglesa.
5. Promover la lectura compresiva.
6. Fomentar el trabajo en equipo, pilar fundamental en la programación.

Contenidos

Los contenidos del CD se desarrollan a lo largo de seis grandes bloques:

1. Introducción.
2. Guía didáctica:
    - Objetivos.
    - Contenidos.
    - Metodología.
    - Actividades.
    - Guía de utilización.
    - Recomendaciones.
    - Créditos.
3. Actividades.
    - Actividad 1: Instalación y registro.
    - Actividad 2: Escribir texto y constantes matemáticas.
        - Escribir texto.
        - Escribir constantes matemáticas.

- Actividad 3: Movimiento de objetos.
  - Movimiento lineal de una imagen.
  - Giro de una imagen.
  - Tambaleo de una imagen.
- Actividad 4: Tabla de multiplicar.
  - Tabla de multiplicar del número 8.
  - Tablado multiplicar de cualquier número.
- Actividad 5: Tipos de triángulos.
- Actividad 6: Dibujar polígonos.
  - Dibujar un cuadrado.
  - Otra manera de dibujar un cuadrado.
  - Dibujar un rectángulo.
  - Dibujar una estrella de 6 puntas.
- Actividad 7: Sumar números.
  - Suma de los 100 primeros números naturales.
  - Suma de los primeros *n* números naturales.
  - Suma de los cuadrados de los primeros *n* números naturales.
- Actividad 8: Ecuación de primer grado.
- Actividad 9: Ecuación de segundo grado.
- Actividad 10: Números primos.
  - Números primos entre 1 y 100.
  - Números primos entre dos valores introducidos por el usuario.
- Actividad 11: Máximo Común Divisor.
- Actividad 12: Simplificar fracciones.
  - Reducir a fracción simple la fracción 1024/3584.
  - Reducir a fracción simple una fracción cualquiera.
- Actividad 13: Letra del NIF.
- Actividad 14: Contador de caracteres.
- Actividad 15: Simulador de Lotería Primitiva.

4. Recursos.

5. Vídeos.

6. Ayuda:

   - Glosario de términos
   - Guía en pdf.

Metodología

El trabajo que se presenta, es sencillo en su uso. Está realizado con una tecnología que permite su visualización en un navegador web estándar. Introduce de forma progresiva y sistemática una metodología para la programación de computadores.

A lo largo de las actividades, se estudian las estructuras y las herramientas necesarias para la programación que están disponibles en el lenguaje KPL v2, y se realizan los ejemplos prácticos propuestos.

Se incluyen ejercicios concretos que pueden repercutir directamente en el aula. No se trata de una propuesta cerrada, sino que es el profesor el que debe contextualizarlos adaptándolos a los diferentes cursos y niveles de enseñanza, en función del nivel curricular de sus alumnos.

Los ejercicios requieren, antes de ir a la computadora, conocimiento de algunos aspectos curriculares del área de matemáticas. La reflexión previa en el aula, favorecerá la asimilación de contenidos matemáticos que pueden presentar una especial dificultad al alumno.

Se trata pues, de una metodología flexible que constituye un trabajo innovador en cuanto a su temática, contenidos y desarrollo.

Actividades

Los contenidos se desarrollan a través de quince actividades que se plantean, ejecutan y resuelven.

El planteamiento cuenta con el resumen explicativo de la actividad, los objetivos que se pretenden conseguir y los contenidos que se desarrollan en la misma.

La ejecución viene dada por el guión de la actividad, que refleja los pasos seguidos en la elaboración del programa. Según se va avanzando en las diferentes actividades, se otorga mayor autonomía al alumno aunque el grado de exigencia va aumentando.

La resolución supone la solución de la actividad. Contamos con dos herramientas, por un lado los vídeos didácticos y por el otro, el código fuente utilizado en cada programa, que una vez copiado en Phrogram, ejecutará el programa diseñado.

Las actividades se centran en contenidos del currículo de matemáticas de diferentes niveles educativos. Será el profesor el que determine que actividades realizará con sus alumnos y diseñará otras adecuadas al nivel cognitivo de los discentes, con el fin de llevar a cabo con éxito el proceso de enseñanza-aprendizaje.

Algunos de los algoritmos utilizados en la resolución de las actividades podrían cambiarse por otro u otros más eficientes, con el fin de optimizar el programa correcto.

Guía de utilización

El objetivo fundamental de este trabajo es iniciar, tanto a profesores como alumnos de niveles de enseñanza previos a la enseñanza universitaria, en el mundo de la programación, con el fin de que sean capaces de entender la estructura básica de un programa y las herramientas elementales de la programación en KPL v2.

Para empezar a navegar por la página web, debes introducir el CD-ROM en el lector de CD o DVD del PC. El trabajo dispone de un fichero autoarrancable que iniciará la aplicación tras unos segundos. Si el contenido del CD se ha pasado a una carpeta del disco duro del ordenador, se debe iniciar la aplicación ejecutando el fichero index.html.

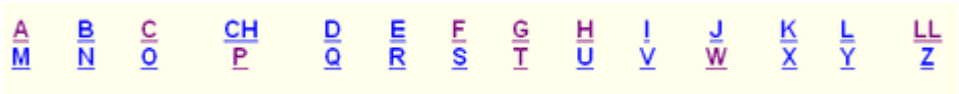
El formato en el que se presenta el proyecto corresponde a una sencilla página web, en la que se puede navegar fácilmente:

- En la parte superior de la página, aparece un menú con seis grandes bloques de contenidos, desde cualquier página se puede acceder a ellos con el fin de facilitar la navegación. Cuando nos encontramos en alguno de estos bloques nos aparece un recuadro que enmarca las páginas y que trata de representar la solapa de una carpeta clasificadora por el punto en el que se

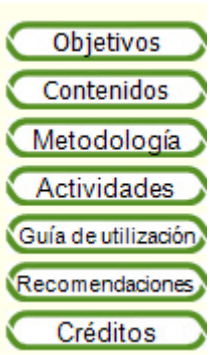
encuentra abierta y/o un título de color gris. Esto sirve para indicarnos en todo momento en que página nos encontramos:

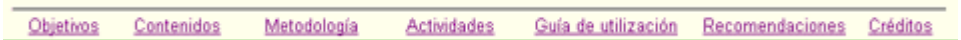
- Cuando accedemos a un apartado de los diferentes bloques, el título que aparece debajo del menú principal está enmarcado en color verde, indicándonos perfectamente en que parte de la guía didáctica, en que actividad o en que letra del glosario nos encontramos:

- El menú de ayuda te permite acceder al glosario de términos y la guía del trabajo en formato pdf. Una vez en el glosario, desde cualquier letra, se visualizan todas las demás del alfabeto:



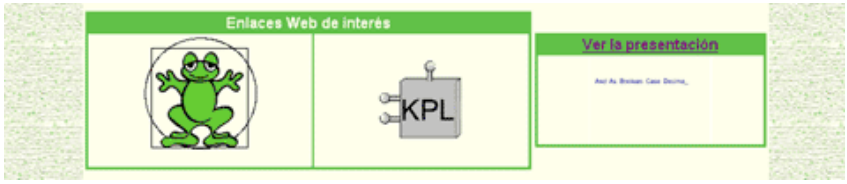
- La guía didáctica y las actividades disponen de un menú fijo en la parte izquierda y otro en la parte inferior que permite acceder de unos contenidos a otros fácilmente:





- Desde todas las páginas se accede a la página oficial de Phrogram, de KPL y a la página de inicio de este trabajo que te permitirá visualizar la animación

del principio, a través unos enlaces que se encuentran en la parte inferior de la página:



Los vídeos didácticos son una herramienta muy útil, pues nos facilitan la compresión de las actividades. Disponen de una barra de herramientas (en la parte inferior de la pantalla) que nos permite parar el vídeo, avance y retroceso rápido, …

Cada persona marcará sus propios ritmos de trabajo en el análisis de las actividades propuestas y en la realización de los ejercicios que se enuncian al final de cada actividad.

Recomendaciones

Para la óptima visualización del CD-ROM se requiere en el equipo:

- Sistema operativo: Microsoft Windows 98 SE o superior.
- Navegador web: Internet Explorer 5.5 o superior.

Página optimizada a 1024 x 768. Se visualiza perfectamente a 800 x 600.

La visualización de documentos requiere de la instalación Adobe Reader 5 o superior (en el menú Recursos puede obtener el Adobe Reader 7.0.8 en español).

La instalación de Phrogram requiere:

- Microsoft Windows 2000, XP o Vista.

Créditos

El material ha sido creado por:

- Pablo Flórez Valbuena.
  Profesor de Matemáticas del IES “El Señor de Bembibre” de León.

- Lidia Getino Llamas.
  Profesora de Tecnología del CEPA “Faustina Álvarez García” de León.

Esta página ha sido diseñada con Microsoft Office FrontPage 2003, Macromedia Flash MX 2004, Swish v2.0 y Adobe Photoshop 7.0.

La música de la presentación se ha obtenido gratuita y abiertamente del Banco de Imágenes y Sonidos del cnice - Ministerio de Educación y Ciencia.

Actividades

Actividad 1

Instalación y registro

Planteamiento

Resumen explicativo

Con esta actividad, aprenderemos a instalar la versión 2 del Lenguaje de Programación para niños, KPL, y a poner el "parche" en español. Además registraremos el producto para no tener que trabajar con la versión TRIAL.

Comenzaremos a familiarizarnos con el área de trabajo.

Objetivos a conseguir

1. Instalar el Software de KPL.
2. Registrar producto.
3. Conocer el entorno de trabajo.

Contenidos

1. Instalar desde el ejecutable Phrogramsetup.exe.
2. Localizar la carpeta donde está instalado KPL v2.
3. Poner el parche en español con PhrogramSpanish.zip.
4. Registrar el producto o trabajar en el modo trial.
5. Utilización de los iconos de la barra de herramientas, del menú,..., con el fin de familiarizarnos con el área de trabajo.

Ejecución

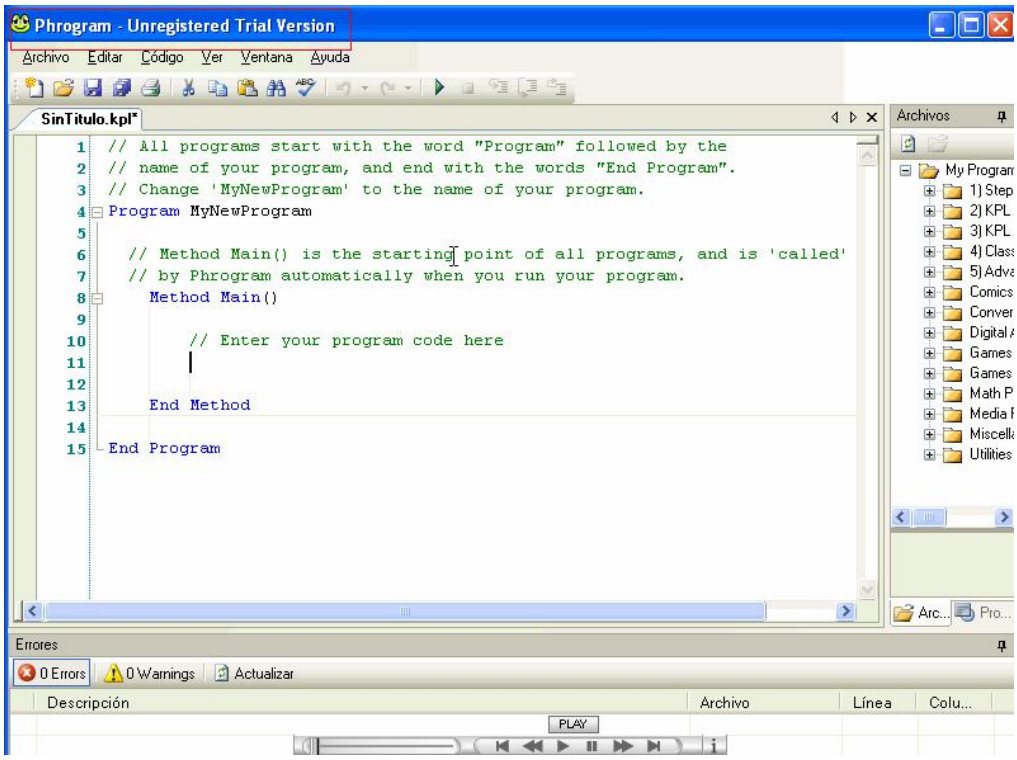
Guión de la actividad

Desde RECURSOS ejecutamos PhrogramSetup.exe (o bien descargamos PhrogramSetup.exe desde la página oficial de Phrogram).

Al instalar PhrogramSetup.exe, InstallShieldWizard prepara la instalación de Microsoft(R).NET 2.0 Framework en la computadora, en el caso de no tenerlo. Framework NET 2.0 sólo puede instalarse en computadoras que corren con Windows 2000, Windows XP o Windows Vista.

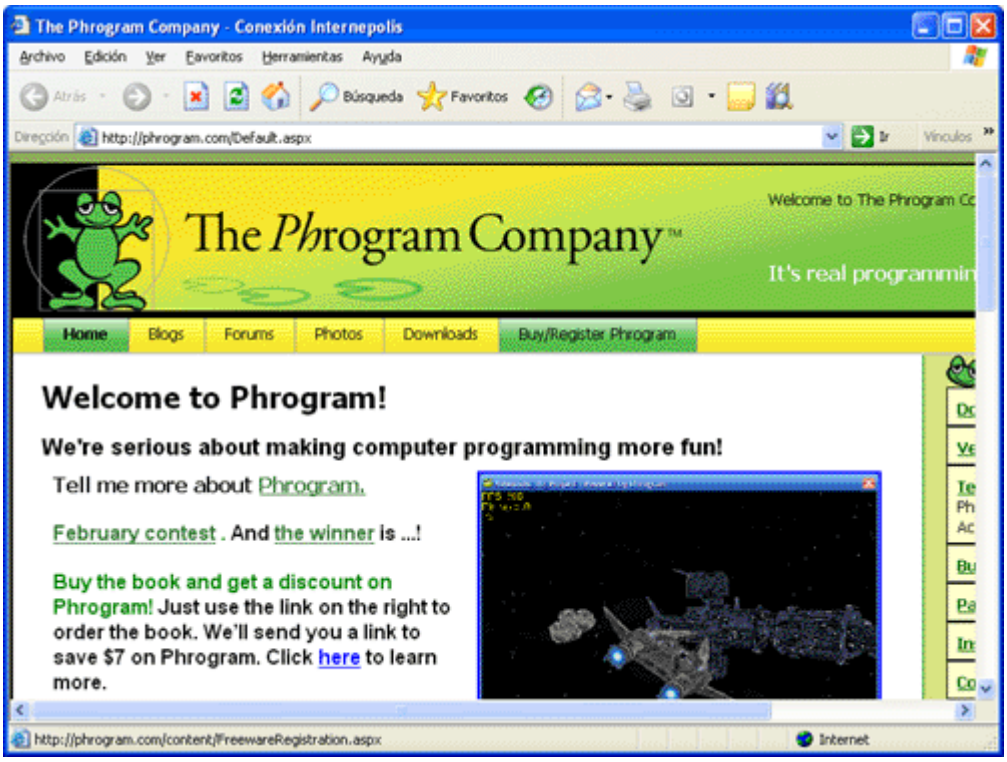
Esta instalación básica incluye todo el contenido programático de KPL v2. Está en Inglés, pero existen programas adicionales adaptados a diferentes idiomas, entre ellos el español. Para poner el programa KPL v2 en español bastará con descomprimir el archivo PhrogramSpanish.zip, que tienes en RECURSOS o desde la página oficial de Phrogram, en la carpeta en la que hayas instalado el programa (por defecto se instala en C:\Archivos de programa\The Phrogram Company\ Phrogram).

Cuando arrancamos el programa esteremos trabajando en el modo Trial, es decir sin registro del producto, como se puede ver en la parte superior de la ventana de Phrogram:

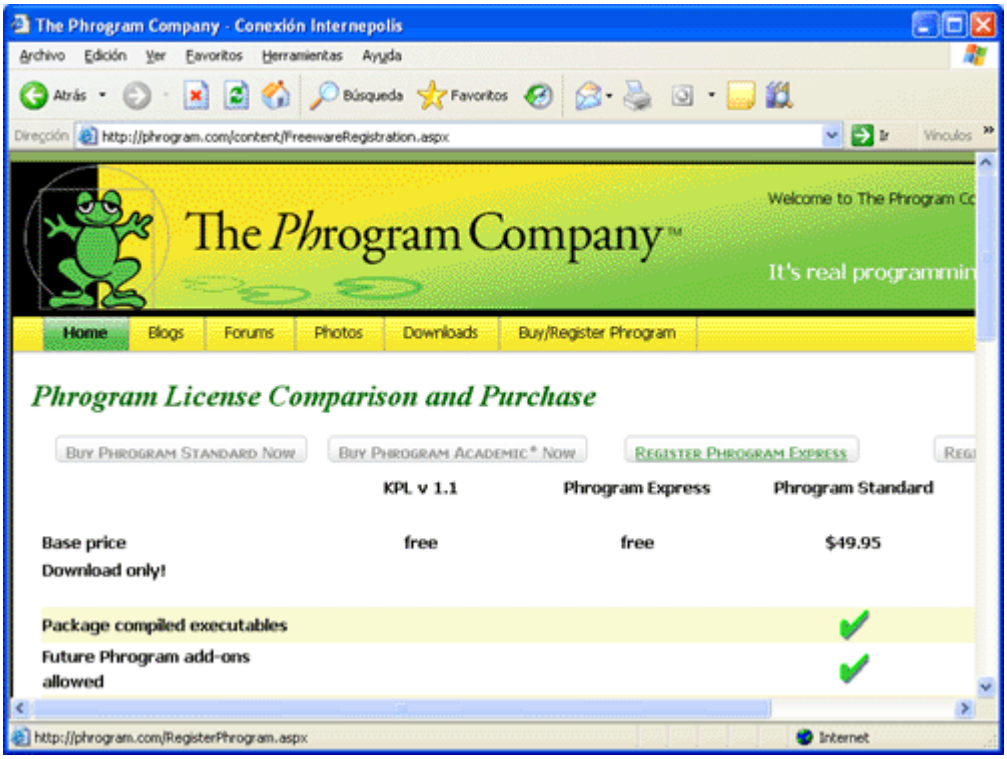


Si decidimos registrarnos, y sólo si tenemos salida a Internet desde nuestro PC, iremos a la página oficial de Phrogram pulsando sobre ayuda / Get registration key, dentro del propio Phrogram:

o desde el CD-Rom picando sobre  se cargará la página siguiente

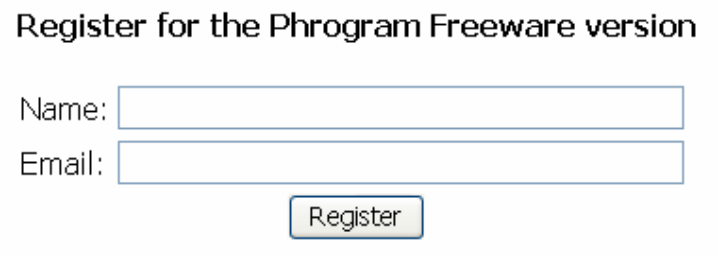


de forma que picando sobre la pestaña Buy/Register Phrogram llegaremos a la página:



seleccionando posteriormente Register Phrogram Express.

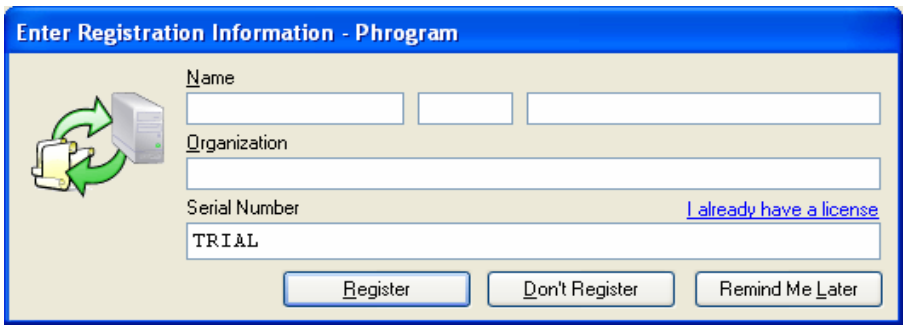
En ambos casos se abre una ventana similar a esta:



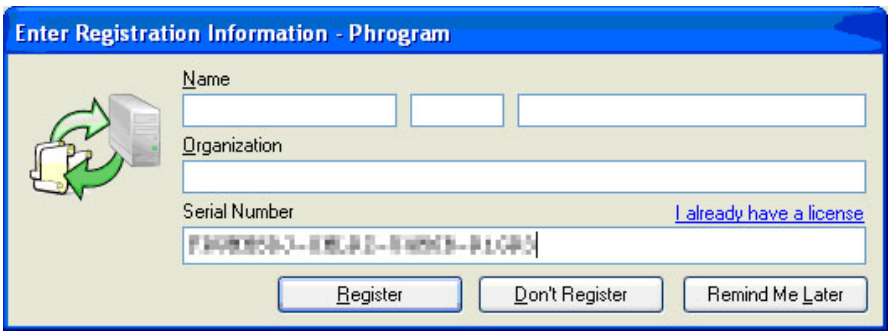
Introduciremos un *Name* y un *correo electrónico* al que nos enviará el administrador una clave. Después, al picar en *register*, nos sale un mensaje agradeciéndonos registrar la versión del programa de libre distribución de KPL V2 y que en 15-30 minutos nos llegará la clave.

Una vez que tengamos la clave, vamos a ayuda / Enter registration key, dentro de Phrogram:

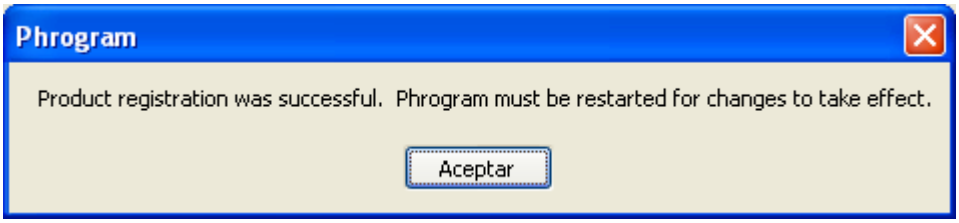
Y nos aparece:



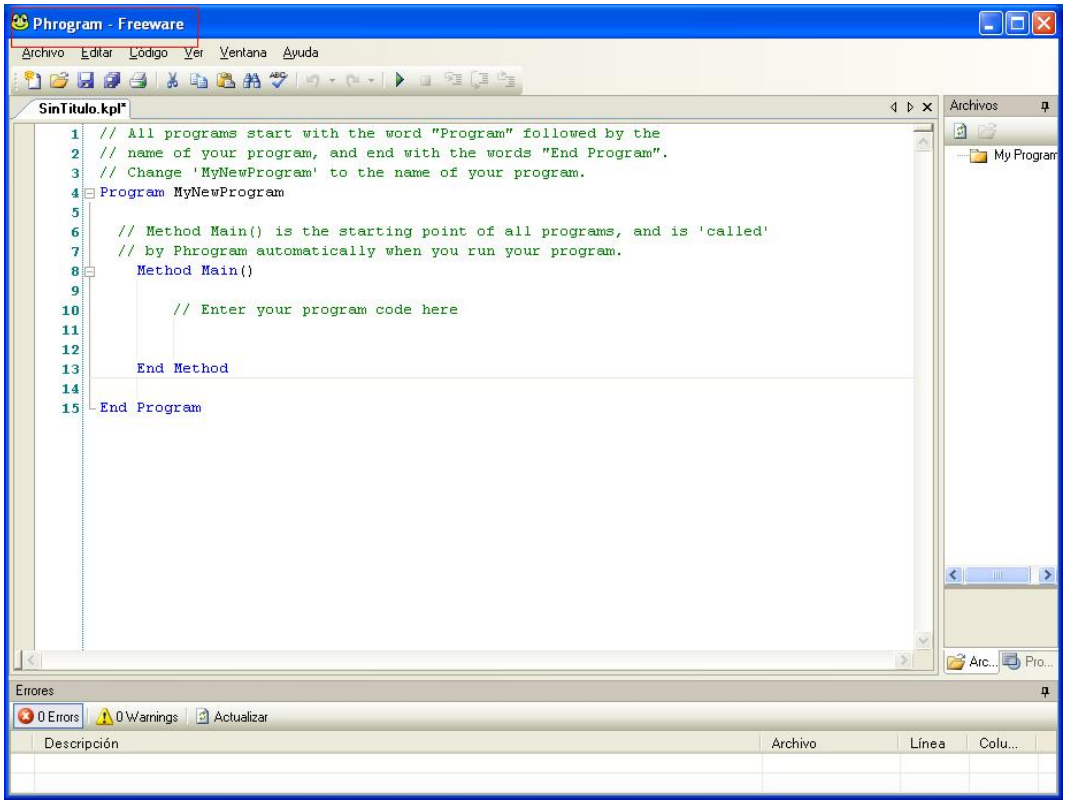
Y sobre Serial Number escribiremos nuestra clave:



Pulsaremos sobre register y nos saldrá una ventana como esta:



Daremos a aceptar y se cerrará el programa. Cuando volvamos a arrancar Phrogram de nuevo nuestro producto estará registrado, como se puede ver en la parte superior de la ventana:



El entorno de trabajo que presenta KPL v2 es sencillo y similar a cualquier software de los que trabajan bajo Windows: barra de menús, barra de herramientas, barra de estado,…

Actividad 2

Escribir texto y constantes matemáticas

Planteamiento

Resumen explicativo

Trataremos de hacer dos programas sencillos con los que escribamos un texto y unas constantes matemáticas.

Objetivos a conseguir

1. Elaborar un programa con el que escribamos un texto y otro con constantes matemáticas para diferenciarlas de los caracteres.

2. Analizar la estructura final de los programas y hacer pequeñas variaciones sobre ellos.

Contenidos

1. Abrir KPL.

2. Poner comentarios a un programa.

3. Identificar la estructura de un programa: Program y Method.

4. Instrucciones PrintLine y Print.

5. El número π, el número e y la función coseno en KPL.

6. Verificar errores en el programa.

7. Ejecutar un programa.

8. Guardar.

9. Abrir y cerrar un programa.

Ejecución

Guión de la actividad 2.1: Escribir texto

Arrancar el KPL, clicando dos veces sobre el icono del escritorio 

.

Eliminar las 3 primeras líneas de programación que son comentarios sobre el programa y que van precedidas de // y en color verde.

Escribir nuestro primer comentario en las líneas borradas anteriormente, poniéndote tú como autor:

```
// PROGRAMA: Escribir
// AUTORES: Lidia y Pablo
// DESCRIPCIÓN: Este programa introduce las instrucciones
// necesarias para imprimir por pantalla caracteres
```

Poner título al programa: Borrar MyNewProgram en poner Escribir en su lugar.

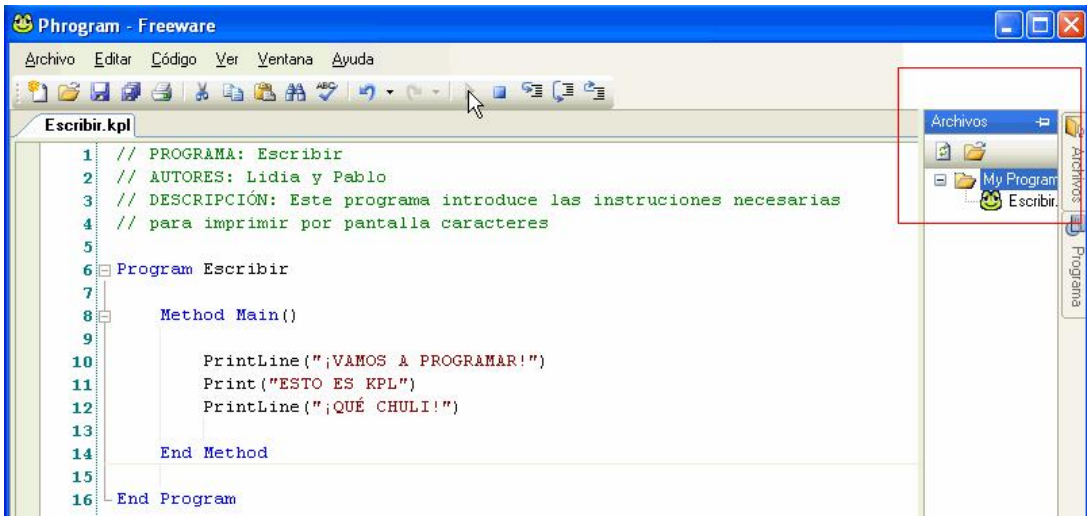
Vamos al Método Principal, Method Main(). Eliminar la línea de programación que es un comentario sobre el programa y que van precedidas de // y en color verde.

Teclear:

```
PrintLine("¡VAMOS A PROGRAMAR!")
Print("ESTO ES KPL")
PrintLine("¡QUÉ CHULI!")
```

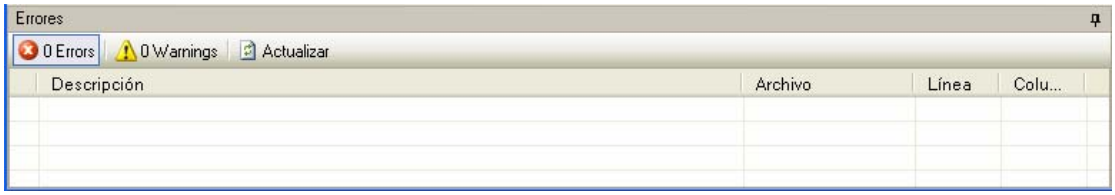
Guardar el archivo: Picando sobre el icono  o Archivo/Guardar, aunque es bueno habituarse a utilizar Guardar como nuevo programa (el *guardar como* de siempre), es decir, guardaremos el archivo kpl y escribiremos el nombre *Escribir*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

Además si lo guardamos en My Programs Files, lo podremos ver desde la pestaña Archivos que está en la parte derecha del entorno de trabajo de Phrogram:



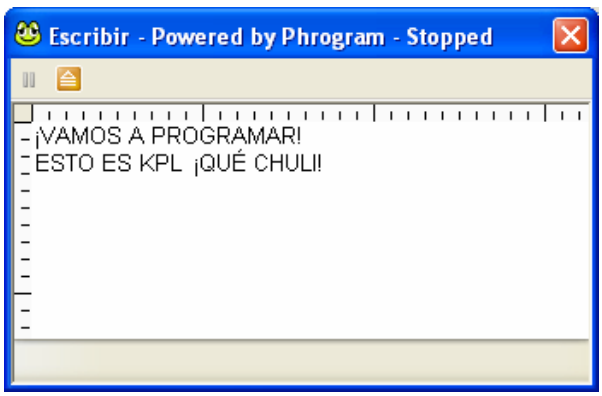
Se guardará un archivo de extensión kpl. Puedes cerrar el Phrogram y picar con el botón derecho del ratón sobre el archivo *Escribir* que acabamos de crear y seleccionar *open*, y así comprobarás que se abre el programa anterior.

En el caso de existir algún error en lo que hemos escrito, que es nada más y nada menos que el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR sobre la parte inferior de la pantalla de Phrogram:



y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Ejecutar el programa: F5 ó  . Sale una ventana emergente en la que se ejecuta nuestro programa:



¿Qué diferencia encuentras entre las instrucciones Print y PrintLine?

Guión de la actividad 2.2: Escribir constantes matemáticas

Arrancar el KPL, picando dos veces sobre el icono del escritorio .

Eliminar las 3 primeras líneas de programación que son comentarios sobre el programa y que van precedidas de // y en color verde.

Escribir comentario en las líneas borradas anteriormente:

```
// Programa: Escribir Constantes Matemáticas
// Autores: Lidia y Pablo
// Descripción: Esta actividad quiere introducir las instrucciones
// necesarias para imprimir por pantalla constantes matemáticas
// y diferenciarlas de los caracteres
```

Poner título al programa: Borrar MyNewProgram y poner *Escribirconstantesmatematicas* en su lugar.

En el Method Main(). Eliminar la línea de programación que es un comentario y teclear:

```
        //imprimir el valor del número pi
        PrintLine(pi)

        //imprimir el valor del número e
        printLine(e)

        //imprimir el valor de una expresión matemática
        Print("cos(2*pi) + e = ")
        Print("1 + e = ")

        printLine(cos(2*pi)+e)
```

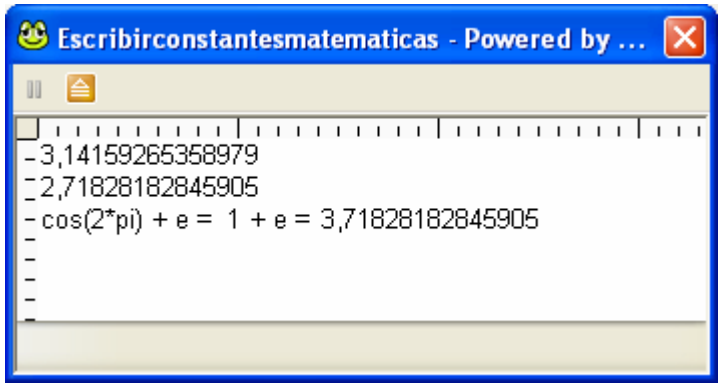
Guardar el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre:

Escribirconstantesmatematicas

seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el código fuente, KPL nos informa con un aviso de ERROR, sobre la parte inferior de la ventana de Phrogram, y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Ejecutar el programa: F5 ó  . Sale una ventana emergente en la que se ejecuta nuestro programa:



Resolución

Código fuente 2.1: Escribir texto

```
// PROGRAMA: Escribir
// AUTORES: Lidia y Pablo
// DESCRIPCIÓN: Este programa introduce las instrucciones
// necesarias para imprimir por pantalla caracteres

Program Escribir

      Method Main()

            PrintLine("¡VAMOS A PROGRAMAR!")
            Print("ESTO ES KPL")
            PrintLine("¡QUÉ CHULI!")

      End Method

End Program
```

Código fuente 2.2: Escribir constantes matemáticas

```
// Programa: Escribir Constantes Matemáticas
// Autores: Lidia y Pablo
// Descripción: Esta actividad quiere introducir las instrucciones
// necesarias para imprimir por pantalla constantes matemáticas
// y diferenciarlas de los caracteres

Program Escribirconstantesmatematicas

      Method Main()

            //imprimir el valor del número pi
            PrintLine(pi)

            //imprimir el valor del número e
            printLine(e)

            //imprimir el valor de una expresión matemática
            Print("cos(2*pi) + e = ")
            Print("1 + e = ")
            printLine(cos(2*pi)+e)

      End Method

End Program
```

Actividad 3

Movimiento de objetos

Planteamiento

Resumen explicativo

Se diseñarán tres programas sencillos para ver algunos movimientos de una imagen.

Objetivos a conseguir

1. Elaborar un programa para mover un objeto linealmente, otro programa realizando un movimiento con giro y otro con un movimiento de tambaleo.

2. Analizar la estructura final de los programas y hacer pequeñas variaciones sobre ellos.

Contenidos

1. Crear un Method dentro de nuestro programa distinto del principal.

2. Instrucciones LoadSprite, MoveSpriteToPoint, ShowSprite, MoveSpriteByAmount, RotateSpriteBy.

3. Sentencia Loop.

4. Instrucción Delay y maximize.

5. Definición de variable.

6. Utilización de programas anteriores (o código fuente de otros programas).

Ejecución

Guión de la actividad 3.1: Movimiento lineal de una imagen

Para esta actividad y las dos siguientes necesitaremos una imagen que está en los RECURSOS del CD.

Vamos a los RECURSOS, picaremos sobre la imagen de Alicia. Se abre una ventana emergente y clicando con el botón derecho sobre la imagen, elegiremos *guardar imagen como,* buscaremos la ruta: C:\Documents and Settings\kpl\Mis documentos\My Phrogram Files\Media Files\Images, y guardaremos.

Arrancar el KPL, clicando dos veces sobre el icono del escritorio .

Escribir nuestro comentario

```
// Programa: Alicia
// Autores: Lidia y Pablo
// descripción: Este programa hace que se mueva un objeto
```

Poner título al programa: Borrar MyNewProgram y poner *Alicia* en su lugar.

Después de la línea donde tenemos `Program Alicia`, escribimos nuestro Método o procedimiento que se llamará cargaralicia():

```
Method cargaralicia()
      LoadSprite("alicia","alicia.gif")
      MoveSpriteToPoint("alicia",0,0)
      ShowSprite("alicia")
End Method
```

LoadSprite("alicia","alicia.gif"), carga la imagen *alicia.gif* como el sprite (sprectro/objeto) *Alicia*.

MoveSpriteToPoint("alicia",0,0) moverá el sprite a las coordenadas (0,0) punto que, por defecto, coincide con la parte superior izquierda de la ventana donde se ejecuta el programa.

ShowSprite ("alicia") muestra el sprite definido.

Un método, como el que hemos definido, no es más que otro programa, al que invocaremos por su nombre.

En el Método Principal, Method Main(), lo primero que escribimos es:

```
cargaralicia()
```

es decir, llamamos al método que lleva ese nombre para que se ejecute.

Después escribimos el código:

```
Loop 100
      MoveSpriteByAmount("alicia",4,1)
      Delay(20)
End Loop
```

La sentencia Loop 100 hace que se repita 100 veces cada una de las líneas (instrucciones, sentencias, …) que hay entre Loop 100 y End Loop.

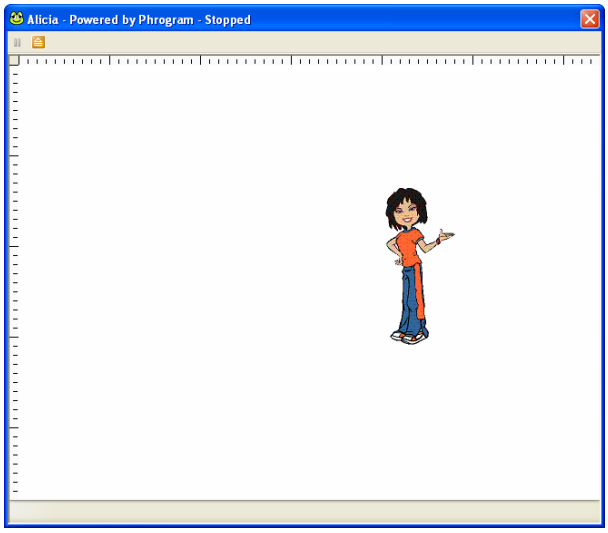
MoveSpriteByAmount mueve el sprite "alicia" la cantidad especificada de pixeles, desde su posición actual.

Delay provoca una pausa para el tiempo especificado (con Delay(1000) la computadora hace una pausa durante 1 segundo).

Guardaremos el archivo y escribiremos el nombre *Alicia*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Ejecutar el programa: F5 ó  . Sale una ventana emergente en la que se ejecuta nuestro programa:

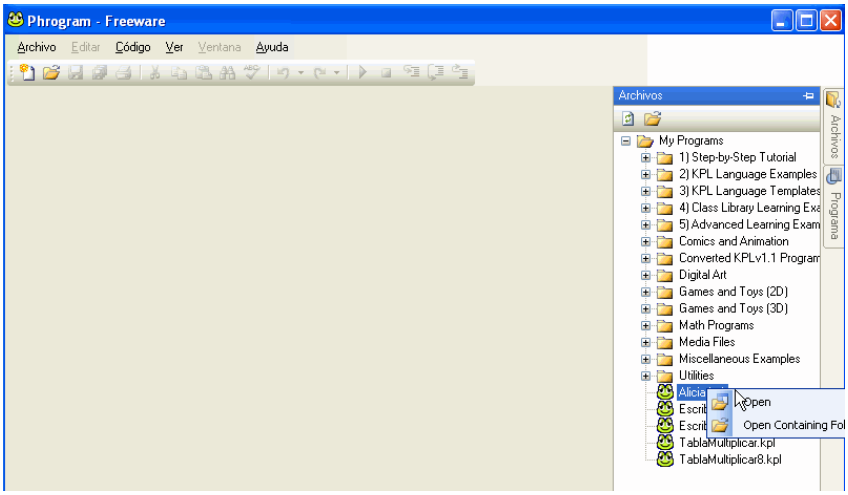


Puedes probar a hacer cambios en el código fuente y observar los resultados; por ejemplo: Poner Delay(2), cambiar la posición inicial de Alicia, cambiar a Loop 200, …

Guión de la actividad 3.2: Giro de una imagen

Arrancar el KPL, haciendo doble click sobre el icono del escritorio.

Vamos a utilizar el código fuente del programa anterior, para ello abrimos el archivo Alicia.kpl de la forma siguiente:



Sobre las 3 primeras líneas de comentarios, apuntamos 2 cambios:

```
// Programa: Alicia2
// Autores: Lidia y Pablo
// descripción: Este programa hace que se mueva girando un objeto
```

Y en título poner *Alicia2.*

Después en el método cargaraliacia():

```
    MoveSpriteToPoint("alicia",50,50)
```

con lo que el sprite arrancará en el punto de coordenadas (50,50)

En el Método Principal, introducimos

```
    maximize()
```

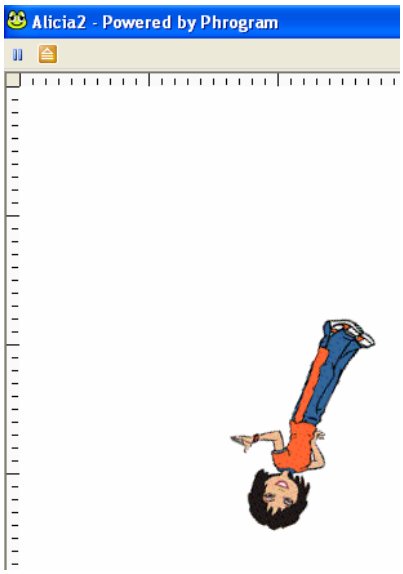
que maximiza la pantalla de la consola.

Después cambiamos a Loop 360, MoveSpriteByAmount("alicia",1,1) e introducimos ***RotateSpriteBy("alicia", 5)*** que gira el sprite la cantidad de 5 grados su orientación actual, y delay(10), quedando el código:

```
Loop 360
      MoveSpriteByAmount("alicia",1,1)
      RotateSpriteBy("alicia",5)
      elay(10)
End Loop
```

¡MUY IMPORTANTE!, Guardaremos el archivo, como nuevo programa y escribiremos el nombre A*licia2*, seleccionando la carpeta donde quiera ir almacenando todas las actividades.

Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.



En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Ejecutar el programa: F5 ó  . Se abre una ventana emergente en la que se ejecuta nuestro programa.

Prueba con otros valores en RotateSpriteBy, por ejemplo con:

a. RotateSpriteBy("alicia",1),
b. RotateSpriteBy("alicia",0.1),
c. RotateSpriteBy("alicia",20).

Guión de la actividad 3.3: Tambaleo de una imagen

Arrancar el KPL.

Vamos a utilizar el código fuente del programa anterior, es decir abrimos el archivo Alicia2.kpl

Sobre las 3 primeras líneas de comentarios, apuntamos dos cambios, quedando:

```
// Programa: Alicia3
// Autores: Lidia y Pablo
// descripción: Este programa hace que se tambalee un objeto
```

Y en título poner *Alicia3.*

En el Método Principal, añadimos después de la línea donde está

```
maximize()
```

una nueva linea con el código:

```
Define angulo As Integer=5
```

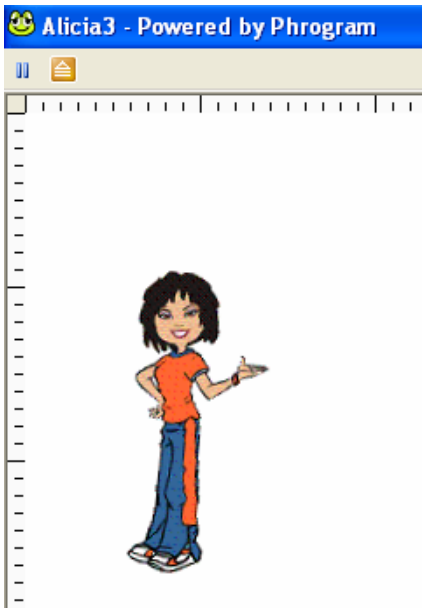
Con ello definimos una variable, llamada *angulo*, cuyos elementos o valores (tipo de dato) son enteros y que toma el valor 5.

Después cambiamos a Loop 300, *RotateSpriteBy("alicia", angulo)*, añadimos la línea `angulo=angulo*-1` que provoca que la variable *angulo* se transforme en `-`(*angulo)* cada vez que se repite el Loop, y delay(50), quedando el código:

```
Loop 300
      MoveSpriteByAmount("alicia",1,1)
      RotateSpriteBy("alicia",angulo)
      angulo=angulo*-1
      Delay(50)
End Loop
```

Guardaremos el archivo, MUY IMPORTANTE, como nuevo programa y escribiremos el nombre *Alicia3*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.



Ejecutar el programa: F5 ó  . Sale una ventana emergente en la que se ejecuta nuestro programa:

Prueba dentro de la sentencia Loop lo que sucede cuando cambias `angulo=angulo*-1` por:

a. angulo=angulo*-2
b. angulo=angulo*3

Resolución

Código fuente 3.1: Movimiento lineal de una imagen

```
// Programa: Alicia
// Autores: Lidia y Pablo
// descripción: Este programa hace que se mueva un objeto

Program Alicia

      Method cargaralicia()
            LoadSprite("alicia","alicia.gif")
            MoveSpriteToPoint("alicia",0,0)
            ShowSprite("alicia")
      End Method

      Method Main()

            cargaralicia()
            Loop 100
                  MoveSpriteByAmount("alicia",4,1)
                  Delay(20)
            End Loop


      End Method

End Program
```

Código fuente 3.2: Giro de una imagen

```
// Programa: Alicia2
// Autores: Lidia y Pablo
// descripción: Este programa hace que se mueva girando un objeto

Program Alicia2

      Method cargaralicia()
            LoadSprite("alicia","alicia.gif")
            MoveSpriteToPoint("alicia",50,50)
            ShowSprite("alicia")
      End Method

      Method Main()
            maximize()
            cargaralicia()
            Loop 360
                  MoveSpriteByAmount("alicia",1,1)
                  RotateSpriteBy("alicia",5)
                  Delay(10)
            End Loop


      End Method

End Program
```

Código fuente 3.3: Tambaleo de una imagen

```
// Programa: Alicia3
// Autores: Lidia y Pablo
// descripción: Este programa hace que se tambalee un objeto

Program Alicia3

      Method cargaralicia()
            LoadSprite("alicia","alicia.gif")
            MoveSpriteToPoint("alicia",50,50)
            ShowSprite("alicia")
      End Method

      Method Main()
            maximize()
            Define angulo As Integer=5
            cargaralicia()
            Loop 300
                  MoveSpriteByAmount("alicia",1,1)
                  RotateSpriteBy("alicia",angulo)
                  angulo=angulo*-1
                  Delay(50)
            End Loop


      End Method

End Program
```

Actividad 4

Tabla de multiplicar

Planteamiento

Resumen explicativo

Se diseñarán dos programas sencillos que imprimen tablas de multiplicar.

Objetivos a conseguir

1. Elaborar un programa para listar la tabla de multiplicar del número 8 y otro programa que listará la tabla de multiplicar de un número cualquiera.

2. Analizar la estructura final de los programas y hacer pequeñas variaciones sobre ellos.

Contenidos

1. Sentencia Loop.

2. Definición de variable.

3. Imprimir variables.

4. Instrucciones SetConsoleTextAlignment, ConsoleWriteLine, ConsoleReadInt y ConsoleWrite.

Ejecución

Guión de la actividad 4.1: Tabla de multiplicar del número 8

Arrancar el KPL, clicando dos veces sobre el icono del escritorio 

.

Escribir nuestro comentario:

```
/*Programa: Tabla de multiplicar del 8
      Autores: Lidia y Pablo
      Descripción: Este programa imprime por pantalla la tabla de
      multiplicar del 8
      ¿Y si queremos la del 9?*/
```

Poner título al programa: Borrar MyNewProgram y poner *TablaMultiplicar8* en su lugar.

En el Método Principal, Method Main(), definimos dos variables N y k con tipo de datos enteros, y escribimos dos comentarios:

```
      Define N As Integer           //número que se trabaja
      Define k As Integer           //números del 1 al 10
```

Ponemos el comentario:

```
      //imprimir la tabla
```

Asignamos el valor 8 a la variable N:

```
      N=8
```

Escribimos estas dos líneas de texto:

```
      PrintLine("Tabla de multiplicar por "+N)
      PrintLine("=================")
```

e inicializamos la variable k:

```
      k=0
```

Tecleamos la sentencia Loop siguiente:

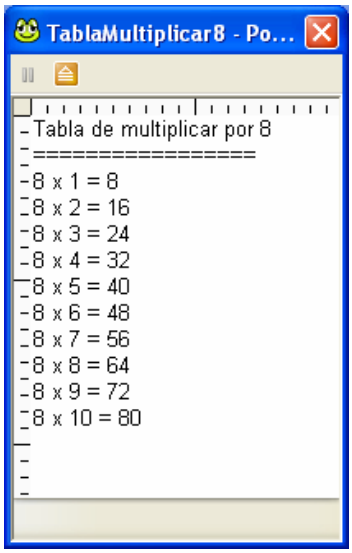
```
      Loop 10
            k=k+1
            PrintLine(N+" x "+k+" = "+N*k)
      End Loop
```

Observa que antes de empezar Loop, k vale 0, después de la primera vez que se ejecuta Loop, k vale 1, después de la segunda vez que se ejecuta Loop, k vale 2, y así hasta 10. También cada vez que se ejecuta Loop, se imprimirá `N+" x "+k+" = "+N*k`, distinguiéndose claramente las variables (van sin comillas) y los caracteres (van entrecomillados).

Guardaremos el archivo con el nombre *TablaMultiplicar8*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Ejecutar el programa:



Puedes probar a cambiar N=8 por N=9 y así obtendremos la tabla de multiplicar del 9.

Guión de la actividad 4.2 Tabla de multiplicar de cualquier número

Arrancar el KPL.

Vamos a utilizar el código fuente del programa anterior, para ello abrimos el archivo *TablaMultiplicar8*.kpl que está por defecto la carpeta My Programs Files dentro de Mis Documentos.

Editamos los comentarios primeros para que nos quede:

```
/*Programa: Tabla de multiplicar
      Autores: Lidia y Pablo
      Descripción: Este programa imprime por pantalla la tabla de
      multiplicar de un número introducido por el usuario*/
```

Y en título ponemos *TablaMultiplicar.*

Guardaremos el archivo, MUY IMPORTANTE, como nuevo programa y escribiremos el nombre *TablaMultiplicar*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el Método Principal, añadimos en una línea nueva, después de la definición de las variables:

```
      SetConsoleTextAlignment("center")
```

que configura el modo de alineamiento del texto de la ventana de consola, es decir, en la ventana emergente que obtenemos al ejecutar el programa.

Después:

```
ConsoleWriteLine("Vamos a escribir la tabla de multiplicar de un número")
```

ConsoleWriteLine escribe texto en la consola y después regresa al inicio de la línea siguiente.

Posteriormente hacemos que la consola pida un dato al usuario introduciendo:

```
N=ConsoleReadInt("Introduce el valor de N = ",True)
```

ConsoleReadInt lee el número entero introducido en el área de entrada de la consola. Por tener dentro True, el texto "Introduce el valor de N = " y la respuesta del usuario serán desplegadas en la consola.

Dejamos dos líneas en blanco tecleando:

```
ConsoleWriteLine("")
ConsoleWriteLine("")
```

Eliminamos la línea donde está:

```
N=8
```

Eliminamos las líneas:

```
PrintLine("Tabla de multiplicar por "+N)
PrintLine("==================")
```

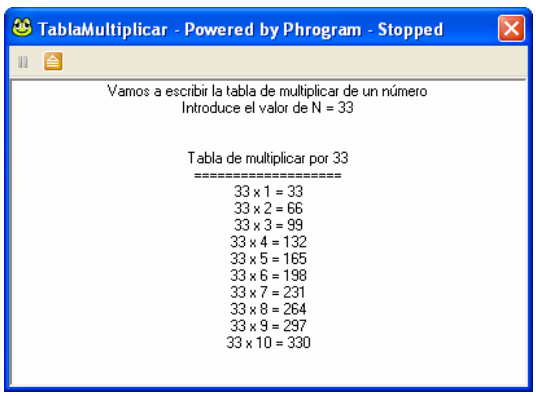
Y añadimos:

```
ConsoleWrite("Tabla de multiplicar por ")
ConsoleWriteLine(N)
ConsoleWriteLine("====================")
```

Dentro de Loop cambiamos *PrintLine* por *ConsoleWriteLine.*

Damos sobre el icono  para guardar.

Ejecutar el programa: Sale la consola en la que se ejecuta nuestro programa. Nos pide un número, damos el 33 por ejemplo y pulsamos INTRO, obteniendo:



Prueba a introducir otros valores para N.

Resolución

Código fuente 4.1: Tabla de multiplicar del número 8

```
/*Programa: Tabla de multiplicar del 8
      Autores: Lidia y Pablo
      Descripción: Este programa imprime por pantalla la tabla de
      multiplicar del 8
      ¿Y si queremos la del 9?*/

Program TablaMultiplicar8

      Method Main()

            Define N As Integer           //número que se trabaja
            Define k As Integer           //números del 1 al 10

            //imprimir la tabla
            N=8
            PrintLine("Tabla de multiplicar por "+N)
            PrintLine("=================")
            k=0
            Loop 10
                  k=k+1
                  PrintLine(N+" x "+k+" = "+N*k)
            End Loop

      End Method

End Program
```

Código fuente 4.2: Tabla de multiplicar de cualquier número

```
/*Programa: Tabla de multiplicar
      Autores: Lidia y Pablo
      Descripción: Este programa imprime por pantalla la tabla de
      multiplicar de un número introducido por el usuario*/

Program TablaMultiplicar

      Method Main()

            Define N As Integer           //número que se trabaja
            Define k As Integer           //números del 1 al 10
            SetConsoleTextAlignment("center")
            ConsoleWriteLine("Vamos a escribir la tabla de
multiplicar de un número")
            N=ConsoleReadInt("Introduce el valor de N = ",True)
            ConsoleWriteLine("")
            ConsoleWriteLine("")
            //imprimir la tabla
            ConsoleWrite("Tabla de multiplicar por ")
```

```
            ConsoleWriteLine(N)
            ConsoleWriteLine("==================")
            k=0
            Loop 10
                  k=k+1
                  ConsoleWriteLine(N+" x "+k+" = "+N*k)
            End Loop

      End Method

End Program
```

Actividad 5

Tipos de triángulos

Planteamiento

Resumen explicativo

Se elaborará un programa que clasifica triángulos según sus lados.

Objetivos a conseguir

1. Crear un programa que lea la longitud de los 3 lados de un triángulo y estudie si existe o no el triángulo, y en caso de existir que analice si el triángulo es equilátero, isósceles, escaleno y/o rectángulo.

2. Analizar la estructura final del programa.

Contenidos

1. Sentencia IF, junto con ELSE y ELSE IF.

2. Operadores lógicos AND y OR.

3. Instrucción ConsoleReadDecimal.

Ejecución

Guión de la actividad

Arrancar el KPL, haciendo click dos veces sobre el icono del escritorio.

Escribir nuestro comentario inicial:

```
/*Programa: Triángulo
      Autores: lidia y pablo
      Descripción: este programa lee la longitud de los 3 lados
      de un triángulo y analiza que tipo de triángulo es
      (no existe triángulo, equilátero, isósceles, escaleno y
      rectángulo)*/
```

Poner título al programa: Borrar MyNewProgram y poner *Triangulo* en su lugar.

En el Método Principal, Method Main(), definimos tres variables con tipo de datos decimal utilizando *ConsoleReadDecimal*, que lee el número decimal introducido en el área de entrada de la consola:

```
      //definir variables y leer vértices
      Var ladoAB As Decimal=ConsoleReadDecimal("Longitud del lado
AB? ",True)
      Var ladoAC As Decimal=ConsoleReadDecimal("Longitud del lado
AC? ",True)
      Var ladoBC As Decimal=ConsoleReadDecimal("Longitud del lado
BC? ",True)
```

Añadimos una línea en blanco:

```
      ConsoleWriteLine("")
```

Se introduce el concepto de la sentencia de selección IF ("si"):

```
      If <condición> Then
          Acción1
      Else
          Acción2
      End If
```

En nuestro caso, para el análisis del triángulo e impresión de resultados, tendremos en cuenta que si la suma de 2 lados cualesquiera del triángulo no es mayor que el tercer lado, entonces no existe triángulo. En caso contrario, estudiaremos cuando es equilátero, isósceles o escaleno:

```
      If <suma de 2 lados<=tercer lado> Then
            No existe triángulo
      Else
            Existe triángulo, y puede ser equilátero, isósceles o escaleno
      End If
```

Nos ayudaremos de las funciones lógicas OR (es decir "o") y AND (es decir "y") para definir con precisión las condiciones, y de ELSE IF para simplificar el código.

Teclearemos:

```
      If (ladoAB+ladoAC<=ladoBC) Or (ladoAB+ladoBC<=ladoAC) Or
(ladoBC+ladoAC<=ladoAB) Then
            ConsoleWriteLine("No existe triángulo")
      Else
            //si los 3 son iguales tenemos un equilátero
            If (ladoAB=ladoAC) And (ladoAC=ladoBC) Then
                  ConsoleWriteLine("El triángulo es equilátero")
                  ConsoleWriteLine("")
            //si 2 son iguuales tenemos un triángulo isósceles
            Else If (ladoAC=ladoAB) Or (ladoBC=ladoAB) Or
(ladoBC=ladoAC) Then
                  ConsoleWriteLine("El triángulo es isósceles")
            //en caso contrario escaleno
            Else
```

```
                    ConsoleWriteLine("El triángulo es escaleno")
            End If
      End If
```

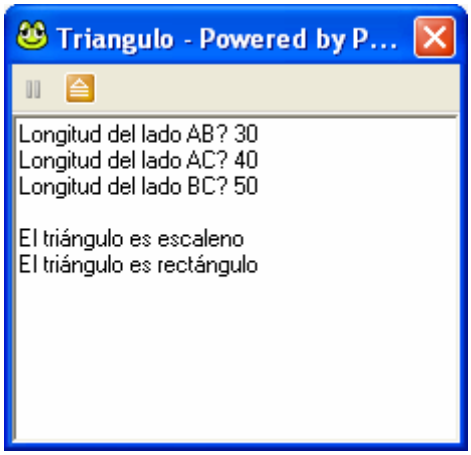
Finalmente utilizamos otro IF para imprimir “El triángulo es rectángulo” cuando se verifica el teorema de Pitágoras:

```
      If (ladoAB*ladoAB+ladoAC*ladoAC=ladoBC*ladoBC) Or
(ladoAB*ladoAB+ladoBC*ladoBC=ladoAC*ladoAC) Or
(ladoBC*ladoBC+ladoAC*ladoAC=ladoAB*ladoAB) Then
            ConsoleWriteLine("El triángulo es rectángulo")
            ConsoleWriteLine("")
      End If
```

Guardaremos el archivo con el nombre *Triángulo*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), se abre la consola pidiéndonos el valor de cada lado. Si damos los datos ladoAB=30, ladoAC=40 y ladoBC=50, obtendremos:



Puedes probar y analizar unos casos, ejecutando el programa con los siguientes datos:

a) ladoAB=4, ladoAC=4 y ladoBC=9
b) ladoAB=5, ladoAC=5 y ladoBC=5
c) ladoAB=20, ladoAC=20y ladoBC=10

Resolución

Código fuente

```
/*Programa: Triángulo
      Autores: lidia y pablo
      Descripción: este programa lee la longitud de los 3 lados de
      un triángulo y analiza que tipo de triángulo es (no existe
      triángulo, equilátero, isósceles escaleno y rectángulo)*/

Program Triangulo

      Method Main()
```

```
            //definir variables y leer vértices
            Var ladoAB As Decimal=ConsoleReadDecimal("Longitud del
lado AB? ",True)
            Var ladoAC As Decimal=ConsoleReadDecimal("Longitud del
lado AC? ",True)
            Var ladoBC As Decimal=ConsoleReadDecimal("Longitud del
lado BC? ",True)
            ConsoleWriteLine("")

            //análisis del triángulo e impresión de resultados
            //si la suma de 2 lados no es mayor que el tercero,
            //entonces no existe triángulo
            If (ladoAB+ladoAC<=ladoBC) Or (ladoAB+ladoBC<=ladoAC)
Or (ladoBC+ladoAC<=ladoAB) Then
                  ConsoleWriteLine("No existe triángulo")
            Else
                  //si los 3 son iguales tenemos un equilátero
                  If (ladoAB=ladoAC) And (ladoAC=ladoBC) Then
                        ConsoleWriteLine("El triángulo es
equilátero")
                        ConsoleWriteLine("")
                  //si 2 son iguuales tenemos un triángulo
                  //isósceles
                  Else If (ladoAC=ladoAB) Or (ladoBC=ladoAB) Or
(ladoBC=ladoAC) Then
                        ConsoleWriteLine("El triángulo es
isósceles")
                  //en caso contrario escaleno
                  Else
                        ConsoleWriteLine("El triángulo es
escaleno")
                  End If
            End If

            //además si verifica el Teorema de Pitágoras, el
            //triángulo será rectángulo
            If (ladoAB*ladoAB+ladoAC*ladoAC=ladoBC*ladoBC) Or
(ladoAB*ladoAB+ladoBC*ladoBC=ladoAC*ladoAC) Or
(ladoBC*ladoBC+ladoAC*ladoAC=ladoAB*ladoAB) Then
                  ConsoleWriteLine("El triángulo es rectángulo")
                  ConsoleWriteLine("")
            End If

      End Method

End Program
```

Actividad 6

Dibujar polígonos

Planteamiento

Resumen explicativo

Se elaborarán cuatro programas que dibujan cuadrados, rectángulos y una estrella de seis puntas.

Objetivos a conseguir

1. Crear dos programas, con técnicas diferentes, que sean capaces de imprimir por pantalla un cuadrado, con los lados de un color determinado y el grosor de la línea dado.

2. Crear un programa que simule el trazado de un rectángulo.

3. Diseñar un programa que dibuje una estrella de seis puntas.

4. Analizar la estructura final de los programas y hacer variaciones sobre ellos.

Contenidos

1. Instrucciones: Color, PenWidht, DrawLine, Pen, MoveTo y Rectangle.

2. Sentencia While.

Ejecución

Guión de la actividad 6.1: Dibujar un cuadrado

Arrancar el KPL, clicando dos veces sobre el icono del escritorio .

Escribir nuestro comentario inicial:

```
/*Programa: Cuadrado
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un cuadrado*/
```

Poner título al programa: Borrar MyNewProgram y poner *Cuadrado* en su lugar.

En el Método Principal, Method Main(), escribimos:

```
Color(green)
```

y así definimos el color con el que vamos a dibujar el cuadrado, después tecleamos:

```
PenWidth(5)
```

con lo que el ancho de la pluma será de 5 pixeles.

Por último, escribimos el código:

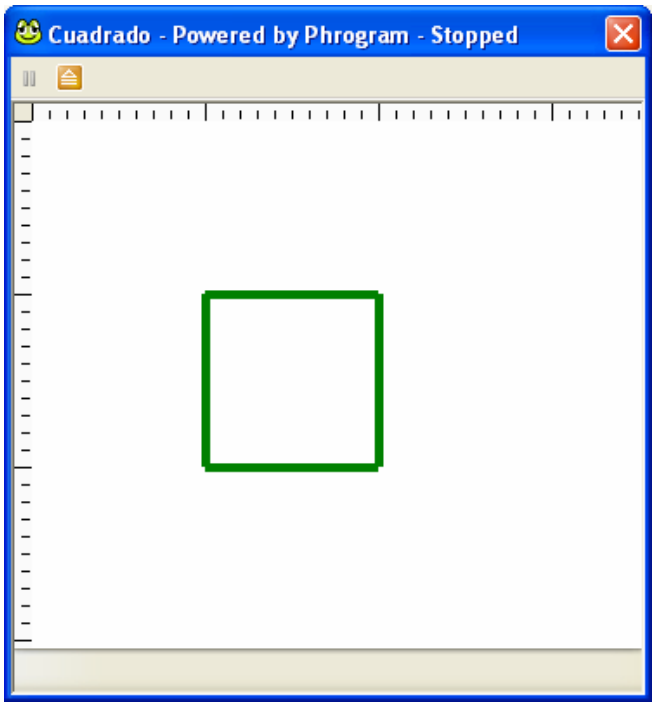
```
DrawLine(100,100,100,200)
Drawline(100,200,200,200)
DrawLine(200,200,200,100)
DrawLine(200,100,100,100)
```

DrawLine(x_1 As Decimal, y_1 As Decimal, x_2 As Decimal, y_2 As Decimal), siendo (x_1, y_1) las coordenadas del punto inicial del segmento y (x_2, y_2) las coordenadas del punto final, dibuja una línea en la pantalla. Así por ejemplo, DrawLine(100,100,100,200) dibuja una línea entre los puntos (100,100) y (100,200)

Guardaremos el archivo con el nombre *Cuadrado*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtendremos:

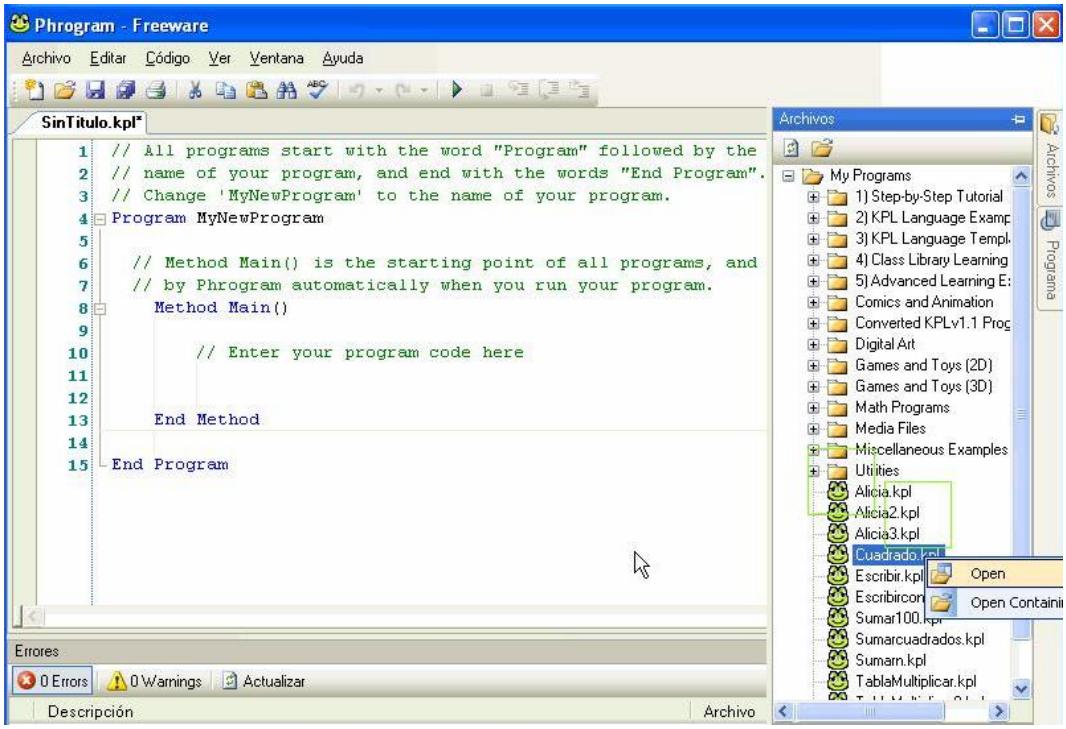


Puedes probar y cambiar las coordenadas de los extremos de los segmentos.

Guión de la actividad 6.2: Otra manera de dibujar un cuadrado

Arrancar el KPL.

Abrimos el archivo *Cuadrado.kpl* creado anteriormente:



Editamos el código de *Cuadrado.kpl* dejando el primer comentario así:

```
/*Programa: Cuadrado2
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un cuadrado*/
```

Poner título al programa: Añadimos un 2 y nos quedará *Cuadrado2*.

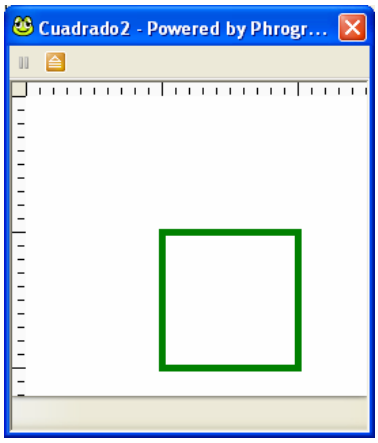
En el Método Principal, Method Main(), borramos todo el contenido y tecleamos:

```
            Pen(False)
            MoveTo(100,100)
            Color(green)
            PenWidth(5)
            Pen(True)
            Rectangle(100,100,False)
```

Con lo que lo primero es decirle a la computadora `Pen(False)`, es decir que desaparezca la pluma, después `MoveTo(100,100)`, moviendo el puntero al punto (100,100), lo siguiente es decirle que al color de la pluma sea verde y el ancho en pixeles 5, y más tarde `Pen(True)` con lo que se activa la pluma para dibujar. Finalmente el método `Rectangle(r,s,False)` dibuja un rectángulo en el color de la pluma actual (verde), empezando a la posición de la pluma actual (el punto (100,100)) y longitudes en

píxeles (r=100 ancho, s=100 alto). Como hemos puesto FALSE el rectángulo no se rellenará.

Guardar el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Cuadrado2*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.



En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtendremos:

Puedes probar y cambiar el ancho y largo en *Rectangle* y obtener un rectángulo. Y si queremos que el rectángulo esté relleno, ¿qué tendremos que hacer?

Guión de la actividad 6.3: Dibujar un rectángulo

Arrancar el KPL.

Escribir el comentario:

```
/*Programa: Rectángulo
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un rectángulo*/
```

Poner título al programa: *Rectángulo*.

En el Método Principal, Method Main(), borramos todo el contenido y tecleamos:

```
            Define x As Integer=100
            Define y As Integer=100
            Define contador As Integer

            Pen(False)
            MoveTo(x,y)
            Pen (True)
            Color(green)
            PenWidth(5)
```

Con lo que después de definir tres variables con tipo de datos Integer, y asignar a x=100 e y=100, nos situamos en el punto (100,100), y dejamos la pluma preparada para dibujar con un grosor 5 y de color verde.

Después escribimos el código:

```
For contador=1 To 100
      x=x
      y=y+1
      MoveTo(x,y)
      Delay(3)
Next
```

Con este FOR lo que hacemos es dejar la coordenada "x" invariante y la "y" aumentarla en 1 pixel cada pasada del *contador* (de 1 a 100), de manera que si la "y" valía inicialmente 100, después de que contador sea 100, la "y" será igual a 200. La instrucción delay ralentiza el movimiento. Con todo ello, al ejecutar el programa comprobaremos la sensación de movimiento que se produce al dibujar el lado.

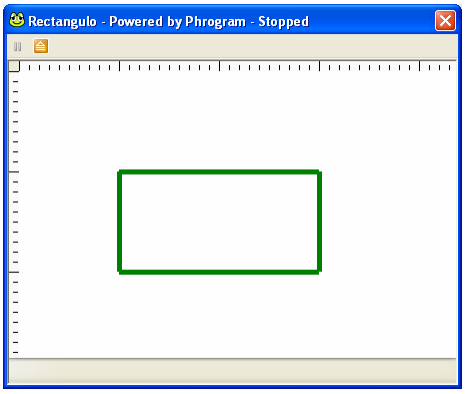
Para que la pluma nos dibuje los otros tres lados, escribimos el código:

```
For contador=1 To 200
      x=x+1
      y=y
      MoveTo(x,y)
      Delay(3)
Next

For contador=1 To 100
      x=x
      y=y-1
      MoveTo(x,y)
      Delay(3)
Next

For contador=1 To 200
      x=x-1
      y=y
      MoveTo(x,y)
      Delay(3)
Next
```

Guardar el archivo con el nombre *Rectangulo*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtendremos:



La gran diferencia con los dos cuadrados de las actividades 6.1 y 6.2 es el movimiento generado para dibujar el rectángulo en esta actividad 6.3.

Guión de la actividad 6.4: Dibujar una estrella de 6 puntas

Arrancar el KPL.

Escribir el comentario:

```
/*Programa: Estrella6puntas
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja una estrella de 6 puntas*/
```

Poner título al programa: *Estrella6Puntas*.

En el Método Principal, Method Main(), borramos todo el contenido y tecleamos:

```
Define x As Integer=40
Define y As Integer=10
Define contador As Integer

Pen(False)
Moveto(x,y)
Pen(True)
Color(blue)
PenWidth(3)
```

Con lo que después de definir tres variables con tipo de datos Integer, y asignar a x=40 e y=10, nos situamos en el punto (40,10), y dejamos la pluma preparada para dibujar con un grosor 3 y de color azul.

Después escribimos el código:

```
While x<=70
      y=y+1
      x=x+1
      MoveTo(x,y)
      Delay(10)
End While
```

La sentencia WHILE (“mientras”), que tiene la estructura:

WHILE <Expresión Boolean>

……

END WHILE

nos dice, en nuestro caso, que mientras x sea menor o igual que 70, se ejecuten todas las instrucciones que tenemos dentro de While. Inicialmente x es 40, pero cada vez que se ejecute x=x+1, el valor de x aumentará en uno hasta llegar a 71, momento en el que saldremos del While a la siguiente línea de programación. Este While nos dibujará un lado de la estrella con moviendo.

Para que la pluma nos dibuje el resto de la estrella, escribimos el código:

```
While x>=10
      y=y
      x=x-1
      MoveTo(x,y)
      Delay(10)
End While

While x<40
      y=y-1
      x=x+1
      MoveTo(x,y)
      Delay(10)
End While

Pen(False)
While y<=50
      y=y+1
      x=x
      MoveTo(x,y)
      Delay(10)
End While
Pen(True)

While x>=10
      y=y-1
      x=x-1
      MoveTo(x,y)
      Delay(10)
End While

While x<=70
      y=y
      x=x+1
      MoveTo(x,y)
      Delay(10)
End While

While x>40
      y=y+1
      x=x-1
      MoveTo(x,y)
      Delay(10)
End While
```

Guardar el archivo con el nombre *Estrella6Puntas*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Programs Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtendremos:

Resolución

Código fuente 6.1: Dibujar un cuadrado

```
/*Programa: Cuadrado
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un cuadrado*/

Program Cuadrado

      Method Main()

            Color(green)
            PenWidth(5)
            DrawLine(100,100,100,200)
            Drawline(100,200,200,200)
            DrawLine(200,200,200,100)
            DrawLine(200,100,100,100)

      End Method

End Program
```

Código fuente 6.2: Otra manera de dibujar un cuadrado

```
/*Programa: Cuadrado2
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un cuadrado*/

Program Cuadrado2

      Method Main()

            Pen(False)
            MoveTo(100,100)
            Color(green)
            PenWidth(5)
            Pen(True)
            Rectangle(100,100,False)

      End Method

End Program
```

Código fuente 6.3: Dibujar un rectángulo

```
/*Programa: Rectángulo
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja un rectángulo*/

Program Rectangulo

      Method Main()

            Define x As Integer=100
            Define y As Integer=100
            Define contador As Integer

            Pen(False)
            MoveTo(x,y)
            Pen (True)
            Color(green)
            PenWidth(5)

            For contador=1 To 100
                  x=x
                  y=y+1
                  MoveTo(x,y)
                  Delay(3)
            Next

            For contador=1 To 200
                  x=x+1
                  y=y
                  MoveTo(x,y)
                  Delay(3)
            Next

            For contador=1 To 100
                  x=x
                  y=y-1
                  MoveTo(x,y)
                  Delay(3)
            Next

            For contador=1 To 200
                  x=x-1
                  y=y
                  MoveTo(x,y)
                  Delay(3)
            Next

      End Method

End Program
```

Código fuente 6.4: Dibujar una estrella de 6 puntas

```
/*Programa: Estrella6puntas
      Autores: Lidia y Pablo
      Descripción: Este programa dibuja una estrella de 6 puntas*/

Program Estrella6Puntas

      Method Main()

            Define x As Integer=40
            Define y As Integer=10
            Define contador As Integer

            Pen(False)
            Moveto(x,y)
            Pen(True)
            Color(blue)
            PenWidth(3)

            While x<=70
                  y=y+1
                  x=x+1
                  MoveTo(x,y)
                  Delay(10)
            End While

            While x>=10
                  y=y
                  x=x-1
                  MoveTo(x,y)
                  Delay(10)
            End While

            While x<40
                  y=y-1
                  x=x+1
                  MoveTo(x,y)
                  Delay(10)
            End While

            Pen(False)
            While y<=50
                  y=y+1
                  x=x
                  MoveTo(x,y)
                  Delay(10)
            End While
            Pen(True)

            While x>=10
                  y=y-1
                  x=x-1
                  MoveTo(x,y)
                  Delay(10)
            End While

            While x<=70
```

```
                  y=y
                  x=x+1
                  MoveTo(x,y)
                  Delay(10)
            End While

            While x>40
                  y=y+1
                  x=x-1
                  MoveTo(x,y)
                  Delay(10)
            End While

      End Method

End Program
```

Actividad 7

Sumar números

Planteamiento

Resumen explicativo

Trataremos de hacer tres programas sencillos con los que calcularemos la suma de series de números enteros.

Objetivos a conseguir

1. Elaborar un programa con el que se calcule la suma de los 100 primeros números naturales.
2. Diseñar un programa con el que se calcule la suma de los n primeros números naturales.
3. Crear un programa con el que se calcule la suma de los cuadrados de los n primeros números naturales.
4. Analizar la estructura final de los programas y hacer pequeñas variaciones sobre ellos.

Contenidos

Afianzar los contenidos:

- Definir variables.
- Sentencia FOR.
- Instrucción ConsoleReadInt

Ejecución

Guión de la actividad 7.1: Suma de los 100 primeros números naturales

Arrancar el KPL, haciendo clic dos veces sobre el icono del escritorio. 

Escribir nuestro comentario inicial:

```
// Programa: Sumar100
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los 100 primeros
// números naturales
```

Poner título al programa: Borrar MyNewProgram y poner *Sumar100* en su lugar.

En el Método Principal, Method Main(), escribimos:

```
            Var m As Integer
            Var suma As Integer
```

para definir las variables.

Después tecleamos:

```
            printLine("Vamos a calcular la suma de los 100
primeros números naturales")
            PrintLine("1+2+3+4+ ... + 99+100")
```

para que el usuario entienda lo que hacemos y ahora escribimos:

```
            For m=1 To 100
                  suma=suma+m
            Next
```

Esta sentencia que va desde m igual a 1 hasta 100, va asignando a la variable suma los siguientes valores:

Para m=1, *suma*=0+1=1
Para m=2, *suma*=1+2=3
Para m=3, *suma*=3+3=6
Para m=4, *suma*=6+4=10
……
Para m=99, *suma*=4851+99=4950
Para m=100, *suma*=4950+100=5050

y de esta manera, cuando salimos del bucle el último valor de *suma* será el valor deseado.

Por último, escribimos el código:

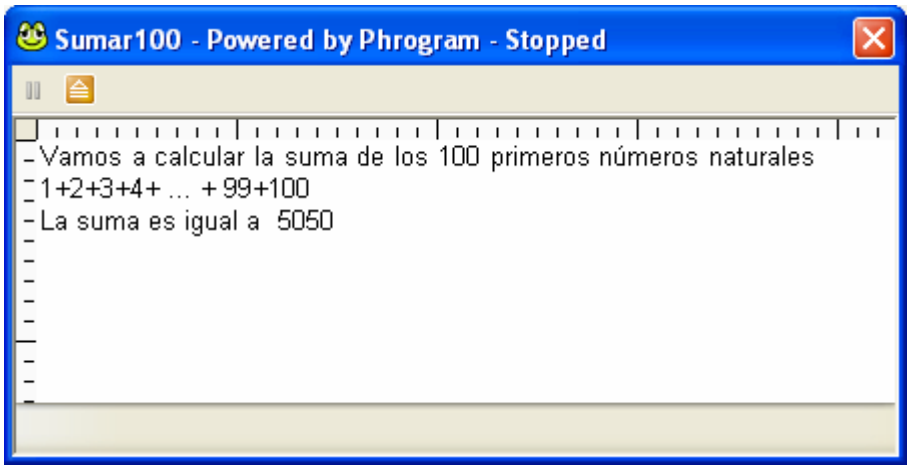
```
            Print("La suma es igual a ")
            Print(suma)
```

que nos imprimirá por pantalla el resultado de la *suma*.

Guardaremos el archivo con el nombre *Sumar100*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtendremos:

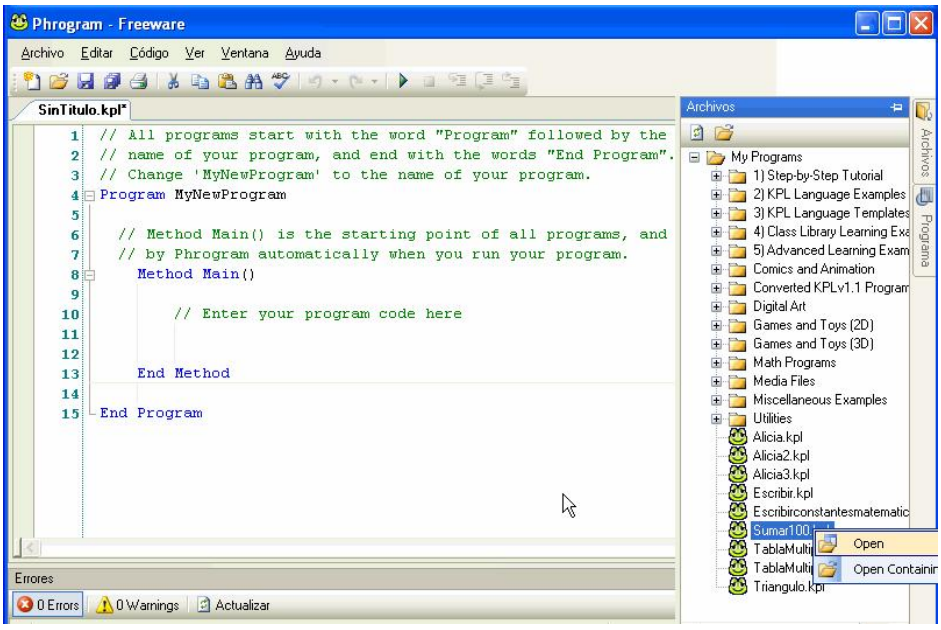


¿Qué tendrás que cambiar en el código del programa para calcular la suma de los 200 primeros números naturales? (Nota: la *suma* te tiene que salir 20100)

Guión de la actividad 7.2: Suma de los primeros *n* números naturales

Arrancar el KPL.

Abrimos el archivo *Sumar100* creado anteriormente bien desde la carpeta My Programs Files que está en Mis Documentos o bien desde:



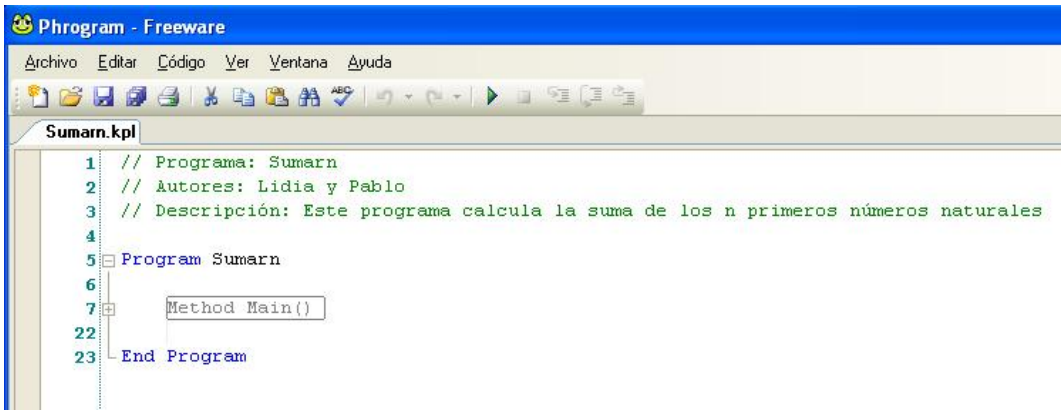
Editamos el código de *Sumar100.kpl* dejándo el primer comentario así:

```
// Programa: Sumarn
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los n primeros
// números naturales
```

Poner título al programa: *Sumarn*.

Guardar el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Sumarn*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

Desplegamos el Método Principal, Method Main()



y empezamos a editar el código. Añadimos la definición de una nueva variable n, que será el extremo de la serie cuya suma vamos a calcular:

```
Var n As Integer
```

Cambiamos en las dos líneas siguientes PrintLine por ConsoleWriteLine y 100 por n, ya que vamos a trabajar con la consola, quedándonos así:

```
ConsoleWriteLine("Vamos a calcular la suma de los n primeros números naturales")
ConsoleWriteLine("1+2+3+4+ ... +(n-1)+n")
```

Pedimos al usuario el valor de la variable n, es decir:

```
n=ConsoleReadInt("Introduce el valor de n = ",True)
```

En la sentencia FOR, su interior lo dejamos como está, salvo 100 que se cambia por n:

```
For m=1 To n
      suma=suma+m
Next
```

Y por último, quitamos las dos líneas con Print y escribimos el código:

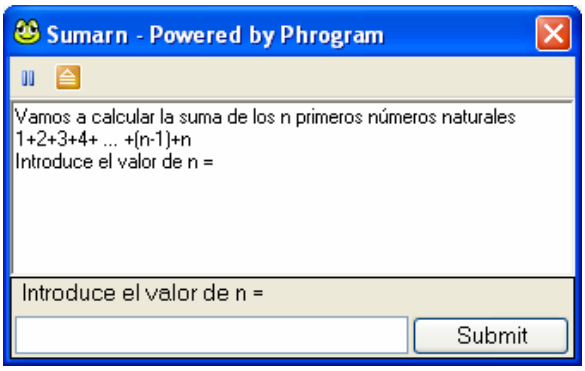
```
ConsoleWrite("La suma es igual a "+suma)
```

que nos imprimirá por pantalla el resultado de la *suma*.

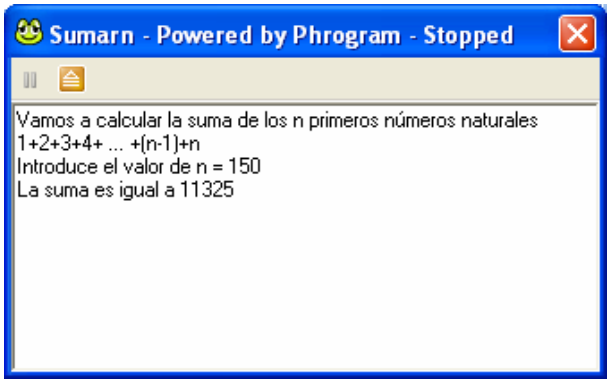
Guardamos el archivo picando sobre el icono  o Archivo/Guardar.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), se activa la consola y nos pide el valor de n:



Si introducimos, por ejemplo, 150 y damos a intro o Submit obtenemos:

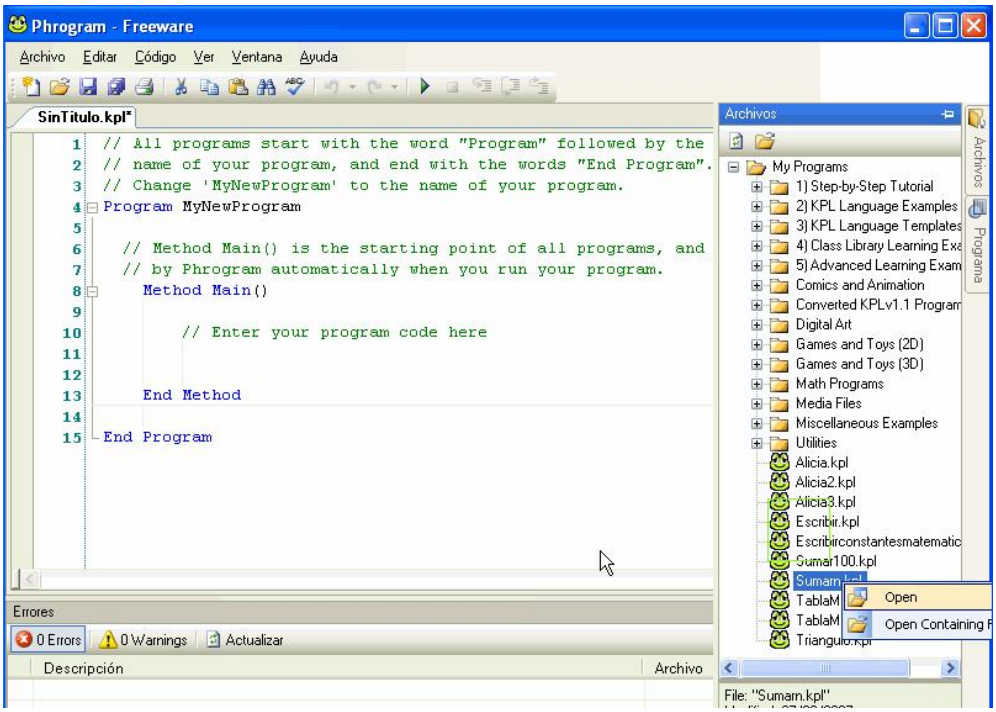


Puedes probar con distintos valores para n, como por ejemplo 100, 200, 3333. ¡Seguro que obtienes 5050, 20100 y 5556111 respectivamente!

Guión de la actividad 7.3: Suma de los cuadrados de los primeros *n* números naturales

Arrancar el KPL.

Abrimos el archivo *Sumarn* creado anteriormente bien desde la carpeta My Programs Files que está en Mis Documentos o bien desde:



Editamos el código de *Sumarn.kpl* dejándo el primer comentario así:

```
// Programa: Sumarcuadrados
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los cuadrados de
// los n primeros números naturales
```

Poner título al programa: *Sumarcuadrados*.

Guardar el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Sumarcuadrados*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

Desplegamos el Método Principal, Method Main() y empezamos a editar el código. Cambiamos en las dos líneas siguientes a la definición de las variables los caracteres en ConsoleWriteLine para que nos quede así:

```
      ConsoleWriteLine("Vamos a calcular la suma de los cuadrados
de los n primeros números naturales")
      ConsoleWriteLine("1x1 + 2x2 + 3x3 + 4x4 + ... +(n-1)x(n-1) +
nxn")
```

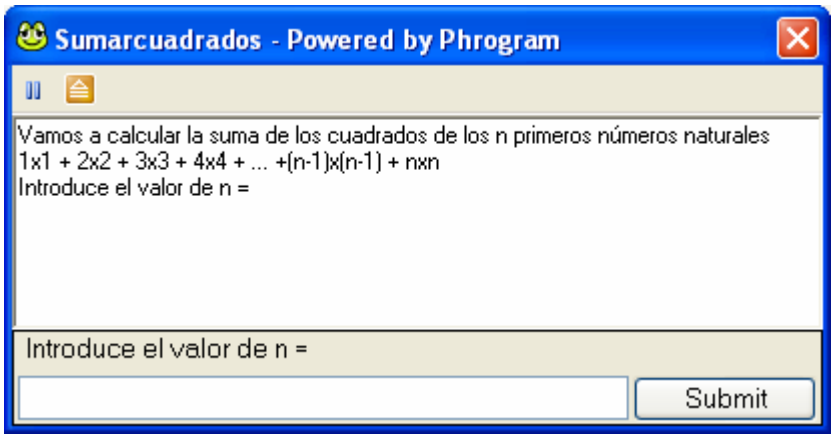
En el interior de la sentencia FOR añadimos a la variable suma, m al cuadrado, en lugar de m:

```
      suma=suma+m*m
```

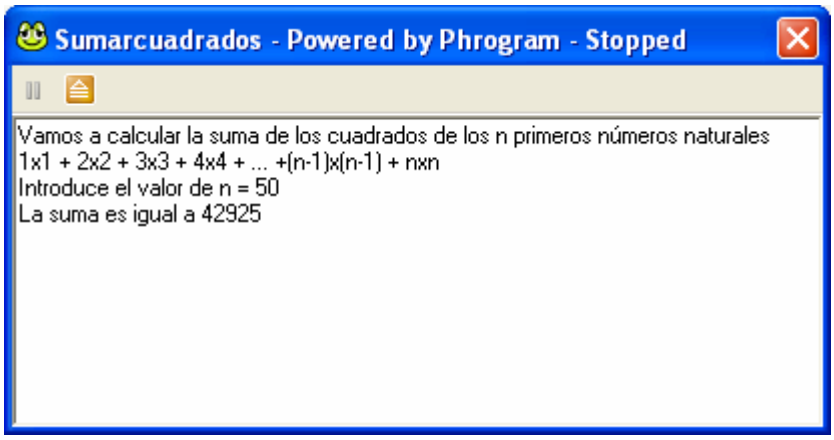
Guardamos el archivo.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), se activa la consola y nos pide el valor de n:



si introducimos, por ejemplo, 50 y damos a intro o Submit obtenemos:



Puedes probar con distintos valores para n.

Resolución

Código fuente 7.1: Suma de los 100 primeros números naturales

```
// Programa: Sumar100
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los 100 primeros
// números naturales

Program Sumar100

      Method Main()

            Var m As Integer
```

```
            Var suma As Integer
            printLine("Vamos a calcular la suma de los 100
primeros números naturales")
            PrintLine("1+2+3+4+ ... + 99+100")
            For m=1 To 100
                  suma=suma+m
            Next
            Print("La suma es igual a ")
            Print(suma)

      End Method

End Program
```

Código fuente 7.2: Suma de los primeros *n* números naturales

```
// Programa: Sumarn
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los n primeros
// números naturales

Program Sumarn

      Method Main()

            Var m As Integer
            Var n As Integer
            Var suma As Integer
            ConsoleWriteLine("Vamos a calcular la suma de los n
primeros números naturales")
            ConsoleWriteLine("1+2+3+4+ ... +(n-1)+n")
            n=ConsoleReadInt("Introduce el valor de n = ",True)
            For m=1 To n
                  suma=suma+m
            Next
            ConsoleWrite("La suma es igual a "+suma)


      End Method

End Program
```

Código fuente 7.3: Suma de cuadrados de los primeros *n* números naturales

```
// Programa: Sumarcuadrados
// Autores: Lidia y Pablo
// Descripción: Este programa calcula la suma de los cuadrados de
// los n primeros números naturales

Program Sumarcuadrados

      Method Main()
```

```
            Var m As Integer
            Var n As Integer
            Var suma As Integer
            ConsoleWriteLine("Vamos a calcular la suma de los
cuadrados de los n primeros números naturales")
            ConsoleWriteLine("1x1 + 2x2 + 3x3 + 4x4 + ... +(n-
1)x(n-1) + nxn")
            n=ConsoleReadInt("Introduce el valor de n = ",True)
            For m=1 To n
                  suma=suma+m*m
            Next
            ConsoleWrite("La suma es igual a "+suma)


      End Method

End Program
```

Actividad 8

Ecuación de primer grado

Planteamiento

Resumen explicativo

Elaboraremos un programa que resuelva la ecuación de primer grado.

Objetivos a conseguir

Elaborar un programa con el que imprima por pantalla la solución de una ecuación de primer grado de la forma $a \cdot x + b = 0$, siempre que $a \neq 0$.

Analizar la estructura final del programa.

Contenidos

Afianzar los contenidos:

- Variables con tipo de dato decimal.
- Sentencia WHILE.
- Instrucción ConsoleReadDecimal.

Ejecución

Guión de la actividad

Arrancar el KPL, clicando dos veces sobre el icono del escritorio .

Escribir nuestro comentario inicial:

```
// Programa: Ecuación 1
// Autores: lidia y pablo
// Descripción: este programa nos muestra la solución de
// la ecuación de primer grado
```

Poner título al programa: Borrar MyNewProgram y poner *Ecuacion1* en su lugar.

En el Método Principal, Method Main(), escribimos:

```
      Var valora As Decimal
      Var valorb As Decimal
      Var valorx As Decimal
```

para definir las variables sobre Decimal.

Después tecleamos:

```
      ConsoleWriteLine("Vamos a resolver una ecuación de primer
grado de la forma:")
      ConsoleWriteLine("   a x + b = 0")
      ConsoleWriteLine("")
```

para que el usuario entienda desde la consola lo que hacemos.

Ahora escribimos:

```
      valora=ConsoleReadDecimal("Introduzca el valor de a =
",True)
```

asignando el valor que introduzca el usuario cuando le pidamos “Introduzca el valor de a = ” a la variable *valora*. Inmediatamente después ponemos el while:

```
      While valora=0
            ConsoleWriteLine("SI a ES CERO NO TENDREMOS UNA
ECUACIÓN DE PRIMER GRADO")
            ConsoleWriteLine("POR FAVOR INTRODUZCA UN VALOR
DISTINTO DE CERO")
            valora=ConsoleReadDecimal("Introduzca el valor de a =
",True)
      End While
```

con ello, creamos un bucle del que no se sale nunca si el usuario introduce el valor cero (0) para la variable “a”. Incluso le ponemos un comentario sobre el asunto en cuestión.

Tecleamos el código:

```
      valorb=ConsoleReadDecimal("Introduzca el valor de b =
",True)
```

y después:

```
      valorx=(valorb*-1)/(valora)
```

que asignará a la variable *valorx* la solución de la ecuación de primer grado.

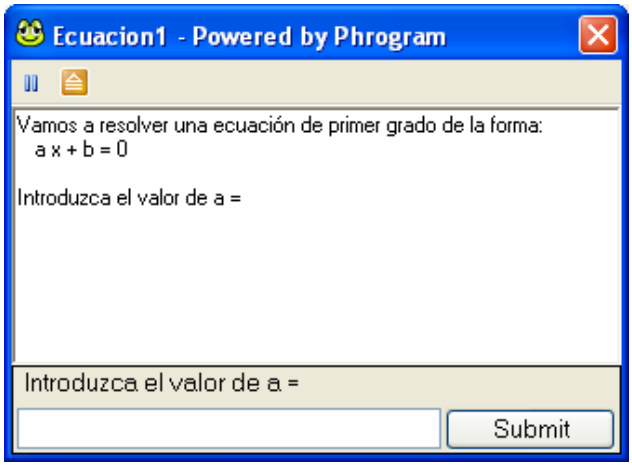
Por último, hacemos que se imprima la solución con un salto de línea antes:

```
            ConsoleWriteLine("")
            ConsoleWrite("Entonces x = "+valorx)
```

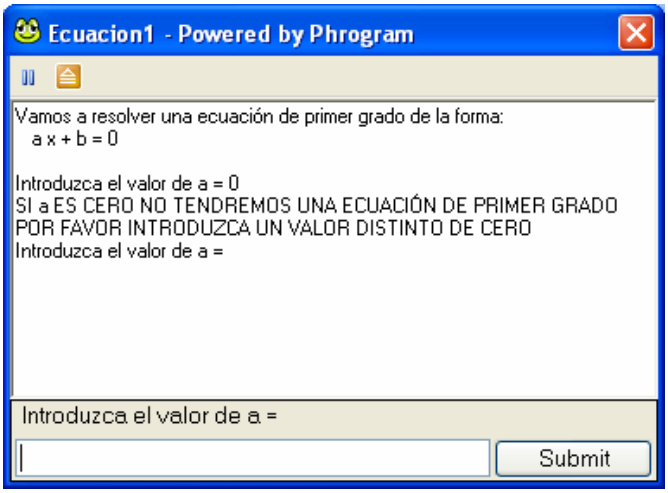
Guardamos el archivo picando sobre el icono  o Archivo/Guardar, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

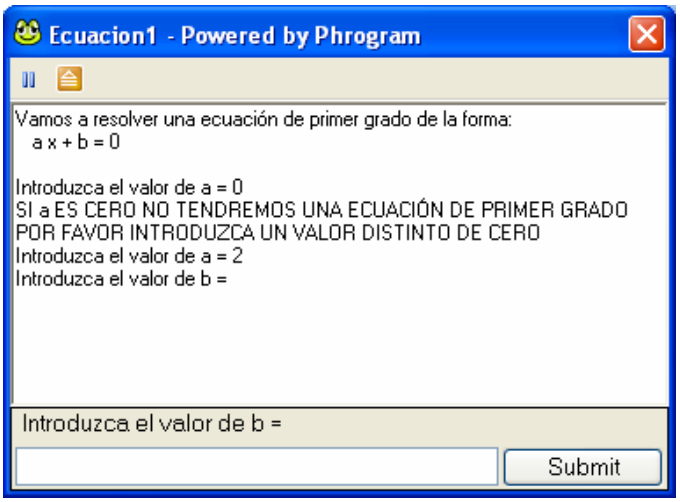
Al ejecutar el programa (F5), se activa la consola y nos pide el valor de a:



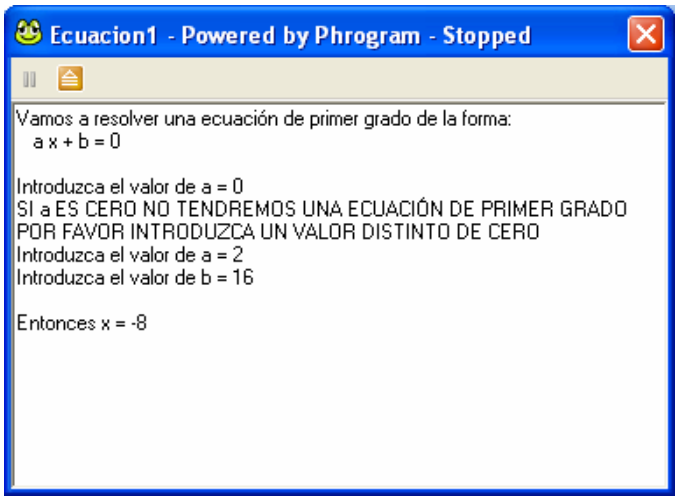
si introducimos 0, obtenemos:



y si ahora introducimos, por ejemplo, 2 y damos a intro, nos pedirá el valor de b:



Introducimos 16 por ejemplo y la solución es:



Puedes probar para los valores de a y b siguientes:

1. a = 3 , b = - 2
2. a = 3 , b = - 1
3. a = 4 , b = 1024

Resolución

Código fuente

```
// Programa: Ecuación 1
// Autores: lidia y pablo
// Descripción: este programa nos muestra la solución de la
// ecuación de primer grado*/

Program Ecuacion1

      Method Main()

            Var valora As Decimal
            Var valorb As Decimal
            Var valorx As Decimal
            ConsoleWriteLine("Vamos a resolver una ecuación de
primer grado de la forma:")
            ConsoleWriteLine("   a x + b = 0")
            ConsoleWriteLine("")
            valora=ConsoleReadDecimal("Introduzca el valor de a =
",True)
            While valora=0
                  ConsoleWriteLine("SI a ES CERO NO TENDREMOS UNA
ECUACIÓN DE PRIMER GRADO")
                  ConsoleWriteLine("POR FAVOR INTRODUZCA UN VALOR
DISTINTO DE CERO")
                  valora=ConsoleReadDecimal("Introduzca el valor
de a = ",True)
            End While
            valorb=ConsoleReadDecimal("Introduzca el valor de b =
",True)
            valorx=(valorb*-1)/(valora)
            ConsoleWriteLine("")
            ConsoleWrite("Entonces x = "+valorx)

      End Method

End Program
```

Actividad 9

Ecuación de segundo grado

Planteamiento

Resumen explicativo

Elaboraremos un programa que resuelva la ecuación de segundo grado.

Objetivos a conseguir

1. Elaborar un programa con el que imprima por pantalla las soluciones de una ecuación de segundo grado de la forma $a \cdot x^2 + b \cdot x + c = 0$, siempre que $a \neq 0$.

2. Analizar la estructura final del programa.

Contenidos

1. Instrucciones SetConsoleTextAlignment, sqrt.

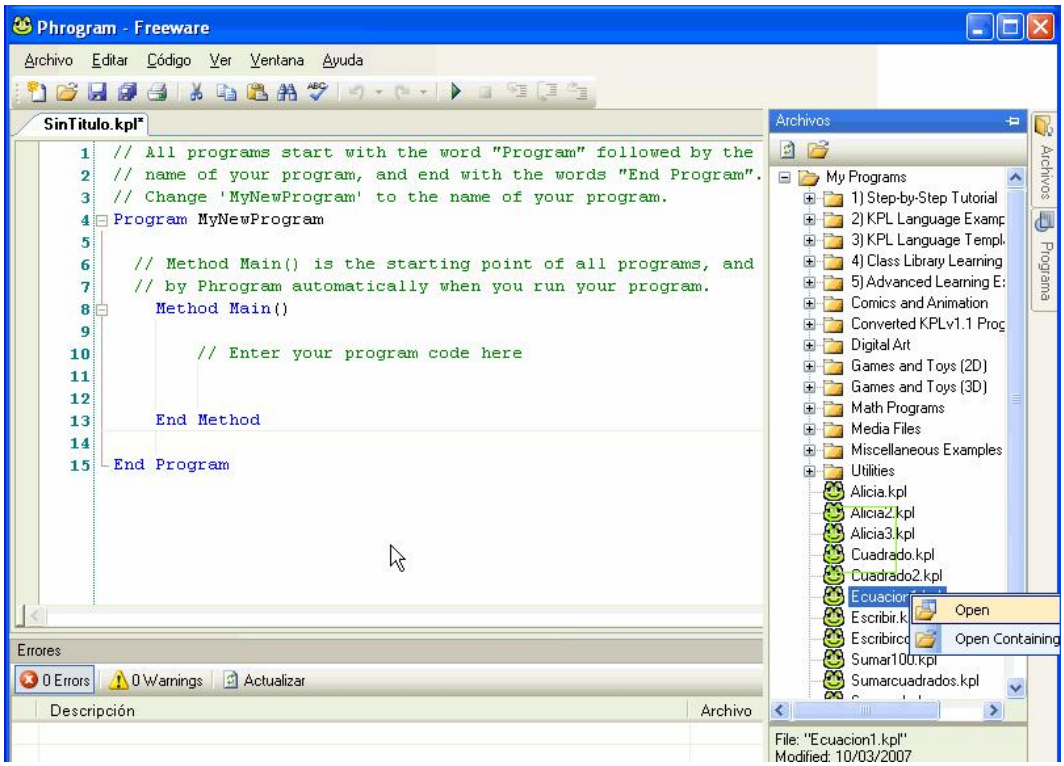
2. Afianzar los contenidos: Sentencia IF

Ejecución

Guión de la actividad

Arrancar el KPL.

Abrimos el archivo *Ecuacion1* creado en la actividad 8 bien desde la carpeta My Programs Files que está en Mis Documentos o bien desde:



Editar el comentario inicial para que quede así:

```
// Programa: Ecuación 2
// Autores: lidia y pablo
// Descripción: este programa nos muestra las soluciones de la
// ecuación de segundo grado, si es que existen
```

Poner título al programa: *Ecuacion2*.

Guardar el archivo picando en Archivo/Guardar como nuevo programa:

y escribiremos el nombre *Ecuacion2*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el Método Principal, Method Main(), añadimos la variable con tipo de dato decimal *valorc*:

```
Var valorc As Decimal
```

Quitamos *valorx* y añadimos las líneas:

```
Var valorx1 As Decimal
Var valorx2 As Decimal
Var discriminante As Decimal
```

para definir las dos variables donde van a estar las soluciones y la variable discriminante, que será el valor del discrimínate de la ecuación de segundo grado.

Escribimos:

```
SetConsoleTextAlignment("center")
```

que configura el modo de alineamiento centrado del texto de la ventana de consola.

Editamos el código de las dos líneas siguientes para que quede:

```
ConsoleWriteLine("Vamos a resolver una ecuación de segundo
grado de la forma:")
ConsoleWriteLine("   a x^2 + b x + c = 0")
```

Dentro del While que tenemos para asegurarnos que el usuario introduce un valor para "a" es distinto de cero, cambiamos el texto PRIMER GRADO por SEGUNDO GRADO.

Después de:

```
valorb=ConsoleReadDecimal("Introduzca el valor de b =
",True)
```

tecleamos el código:

```
valorc=ConsoleReadDecimal("Introduzca el valor de c =
",True)
```

para asignar a *valorc* el valor del término independiente de la ecuación de segundo grado que introduzca el usuario a través de la consola.

Después eliminamos la línea

```
valorx=(valorb*-1)/(valora)
```

y añadimos:

```
discriminante=(valorb*valorb)+(4*(valora)*(valorc*-1))
```

con lo que asignamos a la variable *discriminante* el valor $b^2 - 4 \cdot a \cdot c$.

Por último, eliminamos las dos líneas siguientes y escribimos el código:

```
If discriminante>=0 Then
      valorx1=((valorb*-1)+sqrt(discriminante))/(2*(valora))
      valorx2=((valorb*-1)+(sqrt(discriminante))*-1)/(2*(valora))
      ConsoleWriteLine("")
      ConsoleWriteLine("*****DOS SOLUCIONES REALES*****")
      ConsoleWrite("x = "+valorx1)
      ConsoleWriteLine("")
      ConsoleWrite("x = "+valorx2)
      If valorx1=valorx2 Then
            ConsoleWriteLine("")
            ConsoleWriteLine("***SOLUCIÓN DOBLE***")
      End If
Else
      ConsoleWriteLine("")
      ConsoleWriteLine("*****NO EXISTEN SOLUCIONES REALES*****")
End If
```

¿Qué hacemos con esto? La clave está en el signo del discriminante. Si el discriminante es mayor o igual que cero, sabemos que la ecuación tiene dos soluciones reales que son:

$$x_1 = \frac{-b+\sqrt{b^2-4\cdot a\cdot c}}{2\cdot a} = \frac{-b+sqrt(discri\min ante)}{2\cdot a}$$

$$x_2 = \frac{-b-\sqrt{b^2-4\cdot a\cdot c}}{2\cdot a} = \frac{-b-sqrt(discri\min ante)}{2\cdot a}$$

Obsérvese que sqrt es la función matemática disponible en KPL que devuelve la raíz cuadrada. Si el discriminante es menor que cero, sabemos que la ecuación de segundo grado no tiene solución real. Así pues, la idea es la siguiente:

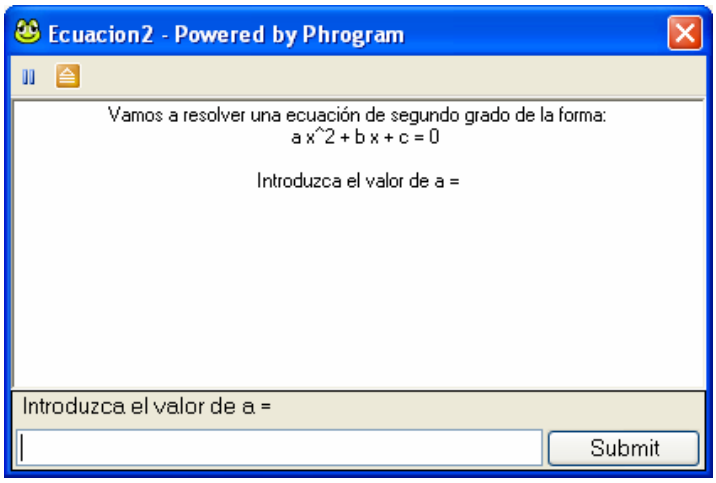
```
      If discriminante>=0 Then
            //existen 2 soluciones reales
      Else
            //no existen soluciones reales
      End If
```

Además en el caso de que el discriminante sea mayor o igual que cero añadimos un IF para que en el caso de que las 2 raíces sean iguales, es decir, el discriminante cero, el programa imprima por pantalla que la solución es doble.

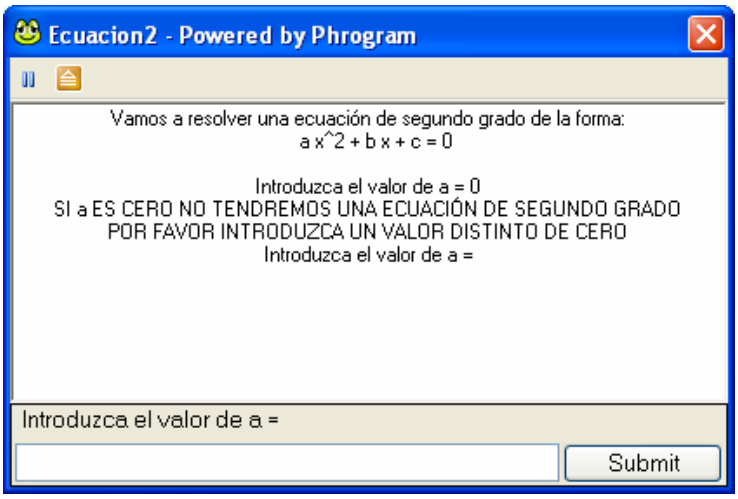
Guardamos el archivo picando sobre el icono  o Archivo/Guardar.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

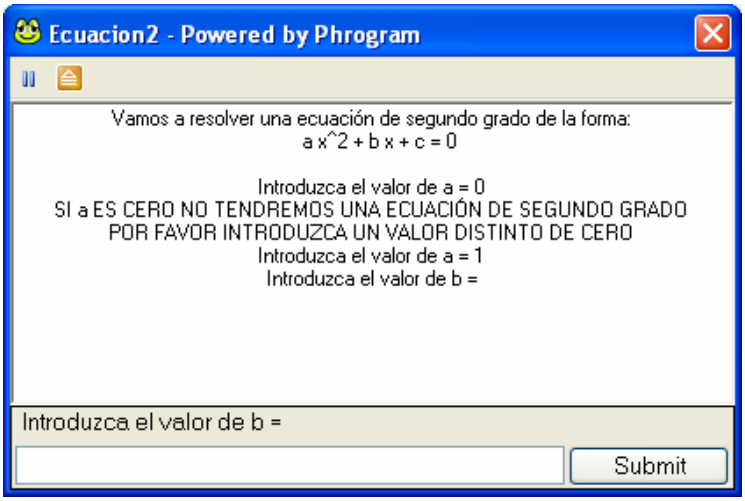
Al ejecutar el programa (F5), se activa la consola y nos pide el valor de a:



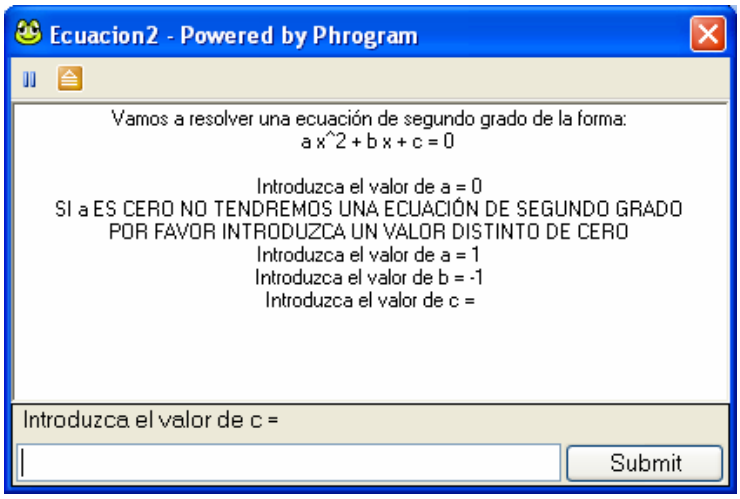
si introducimos 0, obtenemos:



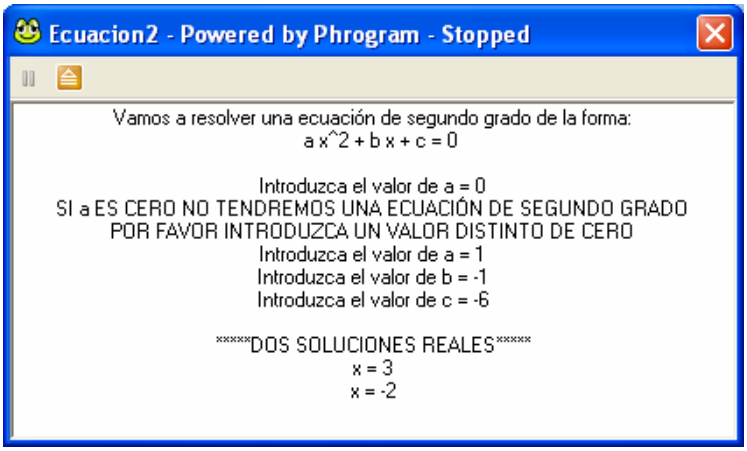
y si ahora introducimos, por ejemplo, 1 y damos a intro, nos pedirá el valor de b:



introducir -1, damos a intro, y después nos pedirá el valor de c:



Introducimos -6 por ejemplo y la solución es:



Puedes probar a ejecutar el programa con distintos valores para a, b y c, como por ejemplo:

1) a = 1, b = 4, c = 4
2) a = 2, b = -1 c = 5
3) a = 1, b = 5, c = 6

Resolución

Código fuente

```
// Programa: Ecuación 2
// Autores: lidia y pablo
// Descripción: este programa nos muestra las soluciones de la
// ecuación de segundo grado, si es que existen

Program Ecuacion2

      Method Main()
```

```
            Var valora As Decimal
            Var valorb As Decimal
            Var valorc As Decimal
            Var valorx1 As Decimal
            Var valorx2 As Decimal
            Var discriminante As Decimal

            SetConsoleTextAlignment("center")
            ConsoleWriteLine("Vamos a resolver una ecuación de
segundo grado de la forma:")
            ConsoleWriteLine("   a x^2 + b x + c = 0")
            ConsoleWriteLine("")

            valora=ConsoleReadDecimal("Introduzca el valor de a =
",True)
            While valora=0
                  ConsoleWriteLine("SI a ES CERO NO TENDREMOS UNA
ECUACIÓN DE SEGUNDO GRADO")
                  ConsoleWriteLine("POR FAVOR INTRODUZCA UN VALOR
DISTINTO DE CERO")
                  valora=ConsoleReadDecimal("Introduzca el valor
de a = ",True)
            End While
            valorb=ConsoleReadDecimal("Introduzca el valor de b =
",True)
            valorc=ConsoleReadDecimal("Introduzca el valor de c =
",True)
            discriminante=(valorb*valorb)+(4*(valora)*(valorc*-1))
            If discriminante>=0 Then
                  valorx1=((valorb*-
1)+sqrt(discriminante))/(2*(valora))
                  valorx2=((valorb*-1)+(sqrt(discriminante))*-
1)/(2*(valora))
                  ConsoleWriteLine("")
                  ConsoleWriteLine("*****DOS SOLUCIONES
REALES*****")
                  ConsoleWrite("x = "+valorx1)
                  ConsoleWriteLine("")
                  ConsoleWrite("x = "+valorx2)
                  If valorx1=valorx2 Then
                        ConsoleWriteLine("")
                        ConsoleWriteLine("***SOLUCIÓN DOBLE***")
                  End If
            Else
                  ConsoleWriteLine("")
                  ConsoleWriteLine("*****NO EXISTEN SOLUCIONES
REALES*****")
            End If

      End Method

End Program
```

Actividad 10

Números primos

Planteamiento

Resumen explicativo

Elaboraremos dos programas que impriman la lista de números primos entre dos números naturales.

Objetivos a conseguir

1. Elaborar un programa con el que se imprima la lista de números primos entre 1 y 100.

2. Diseñar otro programa que imprima los números primos entre dos números naturales introducidos por el usuario. Se puede decir que el segundo es una mejora del primero.

3. Analizar la estructura final del programa y hacer pequeñas variaciones sobre ellos.

Contenidos

- Concepto de FUNCTION, RETURN.
- Resto en una división de números naturales: k MOD d.
- Min y Max.

Ejecución

Guión de la actividad 10.1: Números primos entre 1 y 100

Arrancar el KPL y crear un nuevo programa.

Escribimos en las primeras líneas de código el comentario:

```
/*Programa: Primos1
      Autores: Lidia y Pablo
      Descripción: Este programa imprime la lista de los números
      primos entre 1 y 100*/
```

Poner título al programa: Borrar MyNewProgram y poner *Primos1* en su lugar.

Antes del Método Principal, Method Main(), definimos la variable k, que vamos a utilizar tanto en el Method Main() como en la Function con la que trabajaremos:

```
      Var k As Integer  //número a ensayar
```

Escribimos el comentario:

```
//función que indica si "k" es primo
```

A continuación definimos la Función EsPrimo. Una función es como un subprograma que realiza una determinada acción y que devuelve un valor. FUNCTION debe contener siempre la sentencia RETURN, que devuelve como valor de la función el resultado de los cálculos realizados. Además FUNCTION acaba siempre con End Function. Tecleamos el código:

```
      Function EsPrimo(k As Integer) As Boolean
            Var d As Integer  //posible divisor
            For d=2 To (-1+k)
                  If k Mod d=0 Then
                        Return False
                  End If
            Next
            Return True
      End Function
```

Esta función devuelve True (verdadero) si k es primo y False (falso) si k no es primo. Lo primero que hacemos es definir la variable d con tipo de datos enteros, que actuará de posible divisor de k. A continuación abrimos un FOR desde d=2 hasta (k-1) y dentro un IF que provoca la salida de la función con RETURN False en el caso de que k sea divisible por d ("k Mod d" es la forma de escribir el resto que sale al dividir k entre d), es decir el resto de dividir k entre d es cero.

En el caso de k Mod d sea distinto de cero para todos los valores de d entre 2 y (k-1) entonces el programa salta a la línea siguiente y se ejecuta RETURN True, es decir, k es primo.

Ahora ya dentro del Método Principal, Method Main(), definimos las variables n y m y les asignamos los valores 1 y 100 respectivamente, pues queremos calcular los primos entre esos dos valores:

```
      Var n As Integer
      Var m As Integer
      n=1
      m=100
```

Escribimos:

```
      //impresión del resultado
      PrintLine("Los números primos comprendidos entre los valores
"+n+" y "+m+" son:")
      PrintLine("")
```

y después un FOR en el que k recorre todos los valores entre n (=1) y m (=100) y que imprime k cuando k es primo:

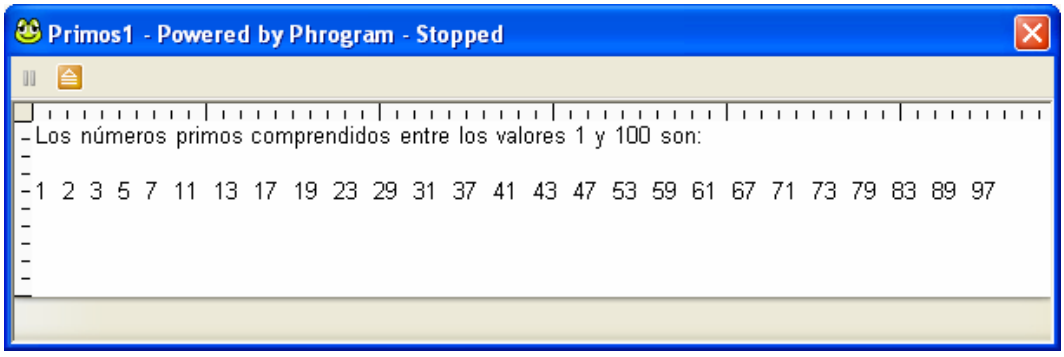
```
      For k=n To m
            If EsPrimo(k) Then
                  Print(k)
                  Print("     ")
            End If
      Next
```

Obsérvese que EsPrimo(k) es la forma de invocar la función que hemos definido anteriormente, entonces, cuando EsPrimo(k) sea True se imprimirá k y unos espacios en blanco.

Guardamos el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Primos1*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

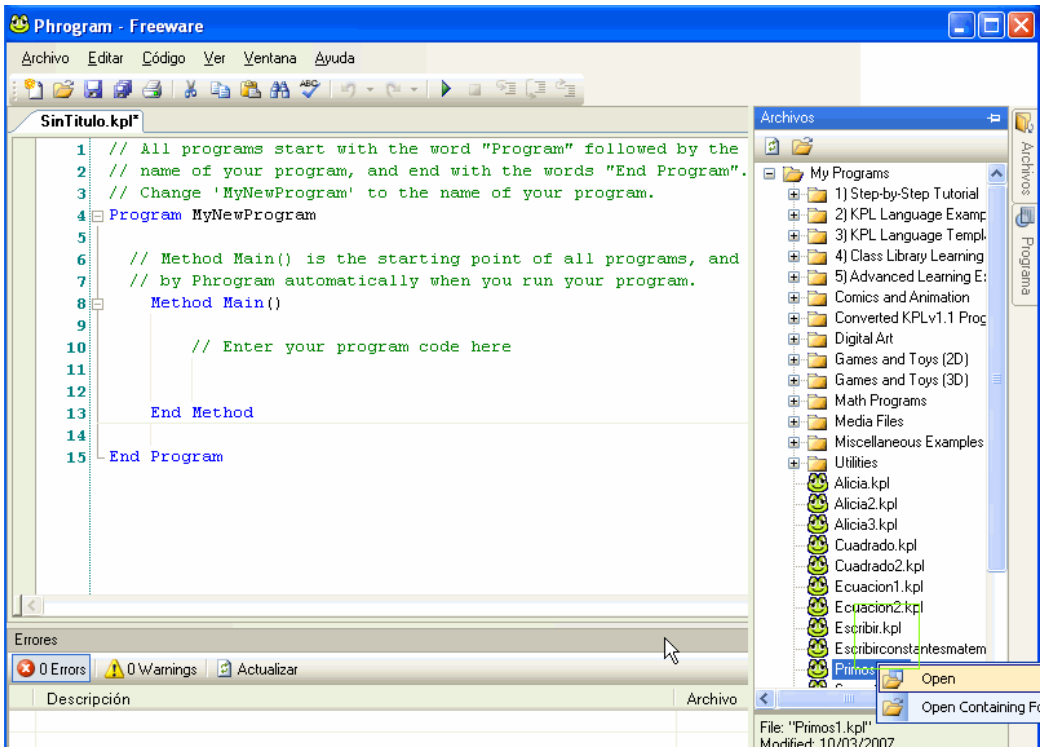
Al ejecutar el programa (F5), obtenemos:



¿Qué tendremos que cambiar en el código fuente de nuestro programa para obtener los primos entre 25 y 121?

Guión de la actividad 10.2: Números primos entre dos valores introducidos por el usuario

Arrancar el KPL y abrimos el archivo *Primos1* creado anteriormente bien desde la carpeta My Programs Files que está en Mis Documentos o bien desde:



Editamos las primeras líneas de código para que el comentario nos quede así:

```
/*Programa: Primos2
      Autores: Lidia y Pablo
      Descripción: Este programa imprime la lista de los números
      primos entre dos valores naturales introducidos por el
      usuario, dando el resultado en 15 columnas*/
```

Poner título al programa: *Primos2* en lugar de *Primos1*.

Guardamos el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Primos2*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

La Function EsPrimo() no la tocamos.

En el Método Principal, Method Main(), definimos la variable columna:

```
      Var columna As Integer        //contador de columnas
```

A continuación añadimos el código:

```
      ConsoleWriteLine("Sabías que el número de números naturales
primos es infinito?")
      ConsoleWriteLine("Vamos a hallar todos los primos entre dos
números naturales dados")
      ConsoleWriteLine("")
```

Hacemos que la variable n tome el valor que introduzca el usuario a través de la consola:

```
Var n As Integer=ConsoleReadInt("Introduzca un número natural:
",True)
```

e introducimos un WHILE para que ese valor de n sea un número entero mayor que cero:

```
While n<=0
      ConsoleWriteLine("Por favor, tiene que ser mayor que cero")
      n=ConsoleReadInt("Introduzca un número natural: ",True)
End While
```

Idem para la variable m:

```
While m<=0
      ConsoleWriteLine("Por favor, otra vez igual,tiene que ser
mayor que cero")
      m=ConsoleReadInt("Introduzca otro número natural: ",True)
End While
```

Escribimos a continuación:

```
//impresión del resultado
ConsoleWriteLine("")
ConsoleWriteLine("Los números primos comprendidos entre los
valores "+Min(n,m)+" y "+Max(n,m)+" son:")
ConsoleWriteLine("")
```

donde Min(n,m) es una función matemática predefinida disponible en KPL que devuelve el menor valor de n y m, y Max(n,m) es también una función matemática predefinida disponible en KPL que devuelve el mayor valor de n y m.

Después editamos el FOR en el que k recorre todos los valores entre Min(n,m) y Max(n,m) y que imprime k cuando k es primo:

```
      For k=Min(n,m) To Max(n,m)
            If EsPrimo(k) Then
                  ConsoleWrite(k)
                  ConsoleWrite("     ")
                  columna=columna+1
                  If columna=15 Then
                        columna=0
```

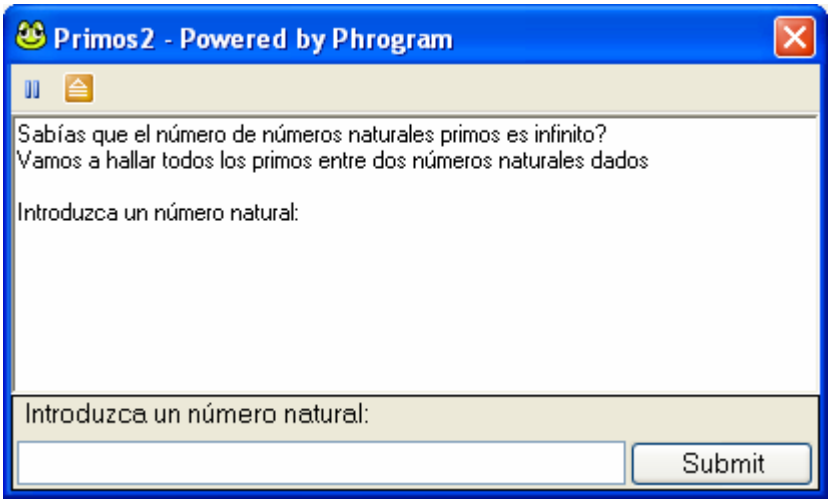
```
                    ConsoleWriteLine("")
                End If
            End If
        Next
```

Cuando EsPrimo(k) sea True se imprimirá k, unos espacios en blanco y además el contador de columnas aumentará en 1 (ver columna=columna+1). A continuación hemos introducimos un IF para que cuando el contador de columnas sea 15, entonces asigna a la variable `columna` el valor cero y además hace un salto de línea de impresión.

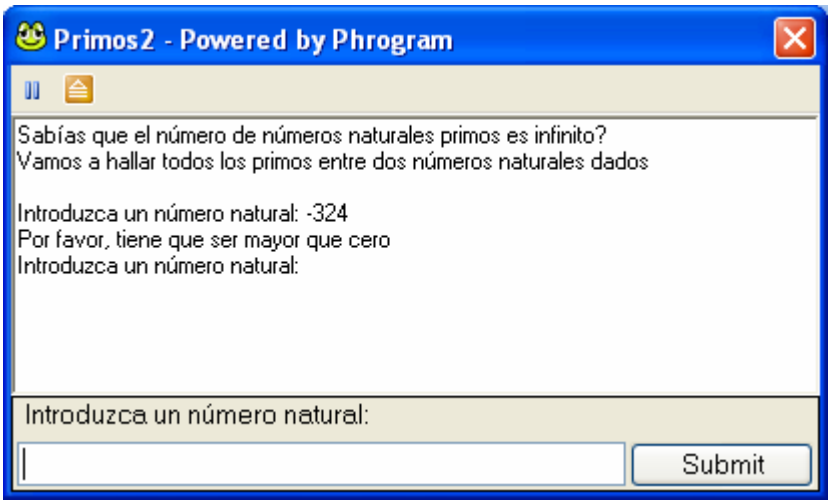
Guardamos el archivo picando sobre el icono  o Archivo/Guardar.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

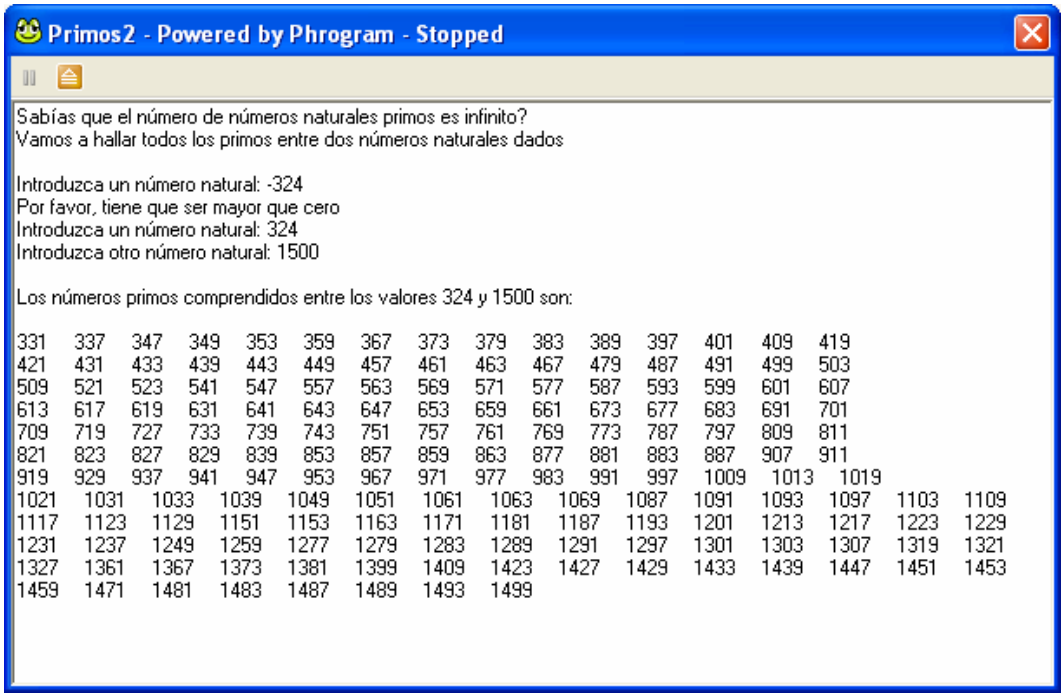
Al ejecutar el programa (F5), nos pedirá el primer el número natural:



Si por ejemplo introducimos -324 y damos a intro, obtendremos:



Ahora si introducimos 324, intro y después nos pedirá otro número natural; introducimos por ejemplo 1500, intro y obtenemos:



Puedes comprobar que si se introduce 1500 primero y 324 después, el programa imprime el mismo resultado. Esto se lo tenemos que agradecer a las funciones matemáticas Min y Max.

¿Cuáles son los primos entre 5000 y 20000?

Resolución

Código fuente 10.1: Números primos entre 1 y 100

```
/*Programa: Primos1
      Autores: Lidia y Pablo
      Descripción: Este programa imprime la lista de los números
      primos entre 1 y 100*/

Program Primos1

      Var k As Integer  //número a ensayar

      //función que indica si "k" es primo
      Function EsPrimo(k As Integer) As Boolean
            Var d As Integer  //posible divisor
            For d=2 To (-1+k)
                  If k Mod d=0 Then
                        Return False
                  End If
            Next
            Return True
      End Function

      Method Main()
```

```
            Var n As Integer
            Var m As Integer
            n=1
            m=100

            //impresión del resultado
            PrintLine("Los números primos comprendidos entre los
valores "+n+" y "+m+" son:")
            PrintLine("")
            For k=n To m
                  If EsPrimo(k) Then
                        Print(k)
                        Print("     ")
                  End If
            Next

      End Method

End Program
```

Código fuente 10.2: Números primos entre dos valores introducidos por el usuario

```
/*Programa: Primos2
      Autores: Lidia y Pablo
      Descripción: Este programa imprime la lista de los números
      primos entre dos valores naturales introducidos por el
      usuario, dando el resultado en 15 columnas*/

Program Primos2

      Var k As Integer  //número a ensayar

      //función que indica si "k" es primo
      Function EsPrimo(k As Integer) As Boolean
            Var d As Integer  //posible divisor
            For d=2 To (-1+k)
                  If k Mod d=0 Then
                        Return False
                  End If
            Next
            Return True
      End Function

      Method Main()

            Var columna As Integer        //contador de columnas

            ConsoleWriteLine("Sabías que el número de números
naturales primos es infinito?")
            ConsoleWriteLine("Vamos a hallar todos los primos
entre dos números naturales dados")
            ConsoleWriteLine("")

            Var n As Integer=ConsoleReadInt("Introduzca un número
natural: ",True)
```

```
            While n<=0
                  ConsoleWriteLine("Por favor, tiene que ser mayor
que cero")
                  n=ConsoleReadInt("Introduzca un número natural:
",True)
            End While
            Var m As Integer=ConsoleReadInt("Introduzca otro
número natural: ",True)
            While m<=0
                  ConsoleWriteLine("Por favor, otra vez
igual,tiene que ser mayor que cero")
                  m=ConsoleReadInt("Introduzca otro número
natural: ",True)
            End While

            //impresión del resultado
            ConsoleWriteLine("")
            ConsoleWriteLine("Los números primos comprendidos
entre los valores "+Min(n,m)+" y "+Max(n,m)+" son:")
            ConsoleWriteLine("")
            For k=Min(n,m) To Max(n,m)
                  If EsPrimo(k) Then
                        ConsoleWrite(k)
                        ConsoleWrite("     ")
                        columna=columna+1
                        If columna=15 Then
                              columna=0
                              ConsoleWriteLine("")
                        End If
                  End If
            Next

      End Method

End Program
```

Actividad 11

Máximo Común Divisor

Planteamiento

Resumen explicativo

Crear un programa que calcule el Máximo Común Divisor de dos números.

Objetivos a conseguir

1. Elaborar un programa con el que imprima por pantalla el Máximo Común Divisor de 148 y 212.
2. Analizar la estructura final del programa.
3. Diseñar un programa, editando el anterior, que imprima por pantalla el Máximo Común Divisor de dos números naturales cualesquiera.

Contenidos

Afianzar los contenidos:

1. Sentencias IF y FOR.
2. Resto de dividir dos números naturales: N Mod k.

Ejecución

Guión de la actividad

Arrancar el KPL, clicando dos veces sobre el icono del escritorio .

Escribir nuestro comentario inicial:

```
/*Programa: Máximo Común Divisor
      Autores: Lidia y Pablo
      Descripción: Este programa imprime el MCD de los números 148
      y 212, teniendo en cuenta que el MCD es el máximo
      número entero que divide a ambos*/
```

Poner título al programa: Borrar MyNewProgram y poner *Mcd* en su lugar.

En el Método Principal, Method Main(), definimos las variables con las que vamos a trabajar y les asignamos los valores N=148, M=212 y X=1:

```
Var X As Integer        //X será el MCD
Var k As Integer
Var N As Integer
Var M As Integer
N=148
M=212
X=1
```

Si queremos calcular el MCD de 900 y 1536 sólo tendremos que editar el código fuente y escribir en estas líneas `N=900` y `M=1536`

Tecleamos:

```
For k=1 To N
      If (N Mod k=M Mod k) And (N Mod k=0) Then
            X=k
      End If
Next
```

con ello, k recorre todos los valores entre 1 y N=148 y si el resto de dividir N entre k es igual al resto de dividir M entre k y ambas cosas son iguales a cero, entonces X será k, es decir, k será el MCD. Para construir esta estructura se ha tenido muy en cuenta que el MCD de dos números es el máximo número natural que divide a ambos.

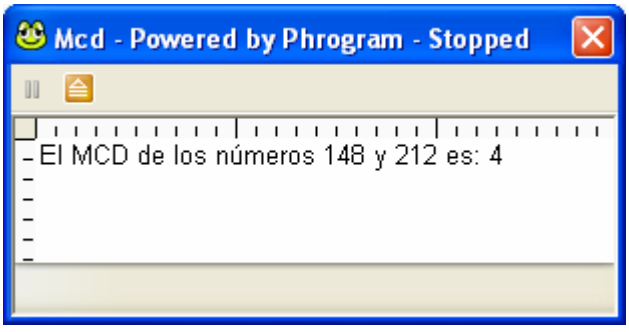
Y por último escribimos el código siguiente que hace que visualicemos por pantalla el resultado:

```
Print("El MCD de los números "+N+" y "+M+" es: "+X)
```

Guardamos el archivo picando sobre el icono  o Archivo/Guardar, escribiremos el nombre Mcd, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), obtenemos:



¿Cuál es MCD de 1536 y 900?

Puedes elaborar un programa editando el anterior en el que se pida al usuario que introduzca dos números naturales e imprima por pantalla su MCD. Ten en cuenta que para ello tendrás que utilizar ConsoleReadInt y ConsoleWrite.

Resolución

Código fuente 11

```
/*Programa: Máximo Común Divisor
      Autores: Lidia y Pablo
      Descripción: Este programa imprime el MCD de los números 148
      y 212, teniendo en cuenta que el MCD es el máximo número
      entero que divide a ambos*/

Program Mcd

      Method Main()

            Var X As Integer        //X será el MCD
            Var k As Integer
            Var N As Integer
            Var M As Integer
            N=148
            M=212
            X=1

            For k=1 To N
                  If (N Mod k=M Mod k) And (N Mod k=0) Then
                        X=k
                  End If
            Next

            Print("El MCD de los números "+N+" y "+M+" es: "+X)

      End Method

End Program
```

Actividad 12

Simplificar fracciones

Planteamiento

Resumen explicativo

Elaboraremos dos programas para simplificar fracciones, es decir buscar una fracción equivalente a la dada en la que MCD (numerador, denominador) = 1.

Objetivos a conseguir

1. Elaborar un programa que reduce a fracción simple la fracción 1024/3584, y otro programa que reduce a fracción simple cualquier fracción introducida por el usuario. Se puede decir que el segundo es una mejora del primero.
2. Analizar la estructura final del programa y hacer pequeñas variaciones sobre ellos.

Contenidos

1. Diseñar un METHOD.

2. Instrucción ABS.

Ejecución

Guión de la actividad 12.1: Reducir a fracción simple la fracción 1024/3584

Arrancar el KPL y crear un nuevo programa.

Escribimos en las primeras líneas de código el comentario:

```
/*Programa: SimplificarFracción1
      Autores: Lidia y Pablo
      Descripción: Este Programa reduce a fracción simple la
      fracción 1024/3584*/
```

Poner título al programa: Borrar MyNewProgram y poner *Fraccion1* en su lugar.

Antes del Método Principal, Method Main(), definimos las variables n (numerador) y d (denominador), que vamos a utilizar tanto en el Method Main() como en la Method ReducirFraccion () con el que trabajaremos:

```
Var n As Integer
Var d As Integer
```

A continuación definimos el Método ReducirFraccion. Un Method es como un subprograma que realiza una determinada acción. METHOD acaba siempre con End Method. Tecleamos el código:

```
Method ReducirFraccion()
      //simplificamos la fracción n/d
      Var divisor As Integer
      divisor=2
      While (divisor<=n) And (divisor<=d)
            While (n Mod divisor=0) And (d Mod divisor=0)
                  n=n/divisor
                  d=d/divisor
            End While
            divisor=divisor+1
      End While
End Method
```

Lo primero que hacemos es definir la variable divisor con tipo de datos enteros y asignarle el valor 2. A continuación abrimos un WHILE en el que las acciones que lleva dentro sólo se ejecutarán cuando la variable divisor sea, a la vez, menor o igual que n y d. La primera acción que tenemos es un WHILE con las condiciones `n Mod divisor=0` y `d Mod divisor=0`, es decir, que el resto de dividir n entre divisor y d entre divisor sean ambos cero. En este caso se ejecutan las asignaciones `n=n/divisor` y `d=d/divisor`, con lo que "n" se transforma en el cociente n/divisor, y a "d" se le asigna el valor de d/divisor. La siguiente línea de programación hace que la variable divisor aumente en 1 (`divisor=divisor+1`).

Este Method que hemos definido lo que hace es dividir numerador y denominador por todos los divisores de ambos, con lo que al final la fracción será simple.

Ahora ya dentro del Método Principal, Method Main(), asignamos los valores 1024 y 3584 a las variables n y d respectivamente, pues queremos reducir a fracción simple 1024/3584:

```
n=1024
d=3584
```

Escribimos el código siguiente:

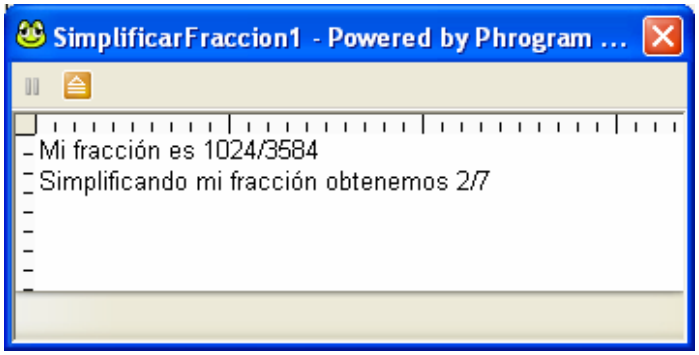
```
PrintLine("Mi fracción es "+n+"/"+d)
ReducirFraccion()
PrintLine("Simplificando mi fracción obtenemos "+n+"/"+d)
```

Obsérvese que ReducirFraccion() es la forma de invocar el Method que hemos definido anteriormente, entonces, cuando ReducirFraccion() se ejecuta, la fracción n/d será simple.

Guardamos el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *SimplificarFraccion1*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

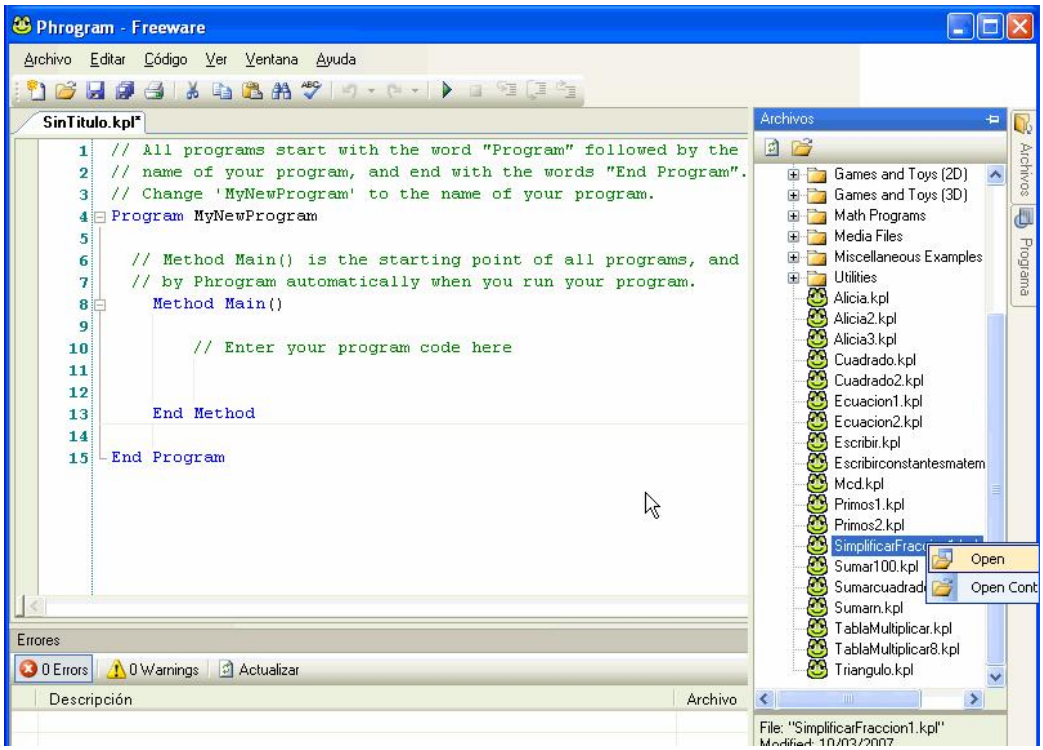
Al ejecutar el programa (F5), obtenemos:



¿Qué tendremos que cambiar en el código fuente de nuestro programa para reducir a fracción simple 5610/9000? ¿Obtienes 187/300?

Guión de la actividad 12.2: Transformar a fracción irreducible una fracción cualquiera

Arrancar el KPL y abrimos el archivo *SimplificarFraccion1* creado anteriormente bien desde la carpeta My Programs Files que está en Mis Documentos o bien desde:



Editamos las primeras líneas de código para que el comentario nos quede así:

```
/*Programa: SimplificarFracción2
      Autores: Lidia y Pablo
      Descripción: Este Programa reduce a fracción simple
      cualquier fracción introducida por el usuario*/
```

Poner título al programa: *SimplificarFraccion2* en lugar de *SimplificarFraccion1.*

En el Method ReducirFraccion(), en las condiciones del primer while, cambiamos n por Abs(n) y d por Abs(d) para poder trabajar con enteros negativos, quedándonos la línea así: `While (divisor<=Abs(n)) And (divisor<=Abs(d))`. ABS es una función matemática predefinida disponible en KPL que devuelve el valor absoluto del número.

También dentro de nuestro Method ReducirFraccion(), añadimos las líneas:

```
      If n<0 And d<0 Then
            n=Abs(n)
            d=Abs(d)
      End If
      If n>0 And d<0 Then
            n=-n
            d=-d
      End If
```

Con el primer IF, lo que hacemos es decirle al programa que si n y d son negativos, lo trasforme a positivos pues en la fracción n/d, "*menos dividido por menos es igual a más*". El segundo IF, hace que cuando el signo negativo esté en el denominador, por *convenio y elegancia* lo pondremos en el numerador.

En el Método Principal, Method Main(), definimos las variables:

```
	n=ConsoleReadInt("Introduzca numerador: ",True)
	d=ConsoleReadInt("Introduzca denominador: ",True)
```

A continuación programamos a la defensiva, estudiando el caso en el que el usuario nos pueda introducir cero para el denominador:

```
	//ojo con el denominador igual a cero
	While d=0
		d=ConsoleReadInt("Por favor, el denominador no puede
ser cero; introduzca denominador: ",True)
	End While
```

Cambiamos `PrintLine("Mi fracción es "+n+"/"+d)` por:

```
	ConsoleWriteLine("Mi fracción es "+n+"/"+d)
```

y finalmente donde está `PrintLine("Simplificando mi fracción obtenemos "+n+"/"+d)` introducimos el código:

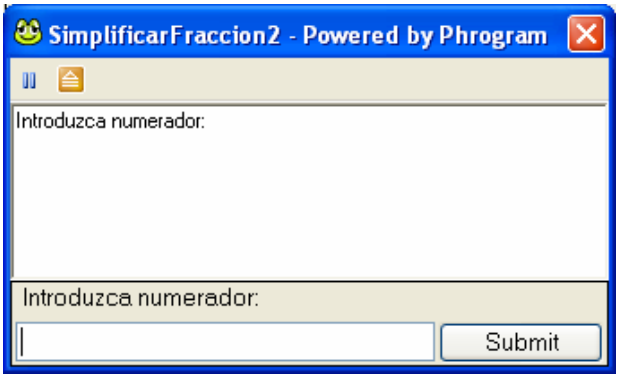
```
	If d<>1 Then
		ConsoleWriteLine("Simplificando mi fracción obtenemos
"+n+"/"+d)
	Else
		ConsoleWriteLine("Simplificando mi fracción obtenemos
un entero "+n)
	End If
```

Con ello hacemos que si el denominador es distinto de uno, por pantalla se escribirá la fracción simplificada, pero si el denominador es 1, para dejarlo “más bonito”, el programa dirá que la fracción es un número entero e imprimirá sólo el numerador.

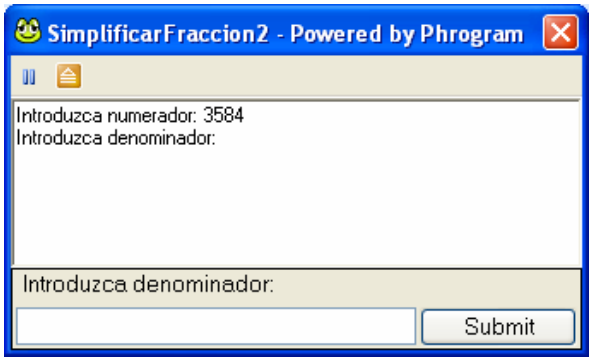
Guardamos el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *SimplificarFraccion2* , seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

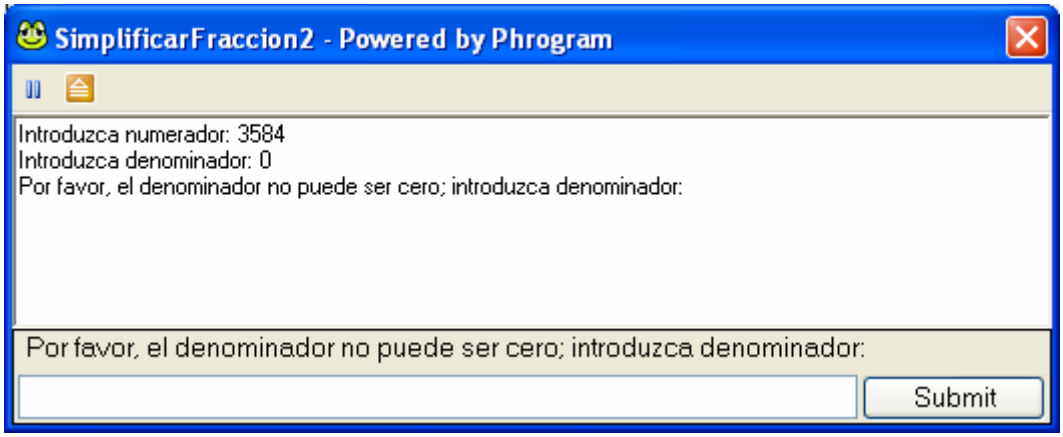
Al ejecutar el programa  , nos pedirá el numerador:



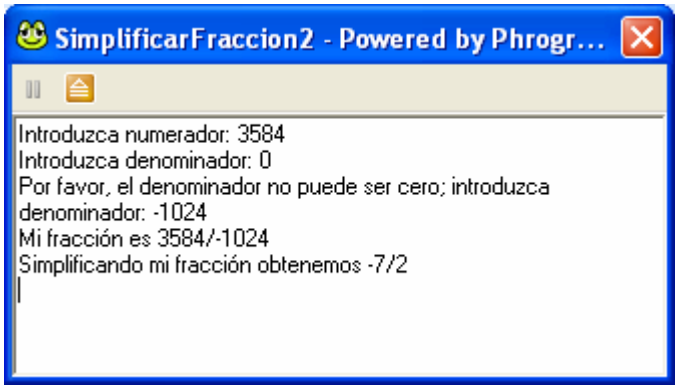
Por ejemplo introducimos 3584 y damos a intro, obtendremos:



Ahora para el denominador introducimos 0, intro y obtenemos:



No podemos poner cero en el denominador, introducimos -1024, intro y sale:



Utilizando el programa que hemos diseñado reduce a fracción simple:

a. 5000 / 2510
b. 5610 / 9000
c. 1024/512
d. 123456 / 78910

Resolución

Código fuente 12.1: Reducir a fracción simple la fracción 1024/3584

```
/*Programa: SimplificarFracción1
      Autores: Lidia y Pablo
      Descripción: Este Programa reduce a fracción simple la
      fracción 1024/3584*/

Program SimplificarFraccion1

      Var n As Integer
      Var d As Integer

      Method ReducirFraccion()
            //simplificamos la fracción n/d
            Var divisor As Integer
            divisor=2
            While (divisor<=n) And (divisor<=d)
                  While (n Mod divisor=0) And (d Mod divisor=0)
                        n=n/divisor
                        d=d/divisor
                  End While
                  divisor=divisor+1
            End While
      End Method

      Method Main()

            n=1024
            d=3584
            PrintLine("Mi fracción es "+n+"/"+d)
            ReducirFraccion()
            PrintLine("Simplificando mi fracción obtenemos
"+n+"/"+d)

      End Method

End Program
```

Código fuente 12.2: Transformar a fracción irreducible una fracción cualquiera

```
/*Programa: SimplificarFracción2
      Autores: Lidia y Pablo
      Descripción: Este Programa reduce a fracción simple
      cualquier fracción introducida por el usuario*/

Program SimplificarFraccion2

      Var n As Integer
```

```
      Var d As Integer

      Method ReducirFraccion()
            //simplificamos la fracción n/d
            Var divisor As Integer
            divisor=2
            While (divisor<=Abs(n)) And (divisor<=Abs(d))
                  While (n Mod divisor=0) And (d Mod divisor=0)
                        n=n/divisor
                        d=d/divisor
                  End While
                  divisor=divisor+1
            End While
            If n<0 And d<0 Then
                  n=Abs(n)
                  d=Abs(d)
            End If
            If n>0 And d<0 Then
                  n=-n
                  d=-d
            End If
      End Method

      Method Main()

            n=ConsoleReadInt("Introduzca numerador: ",True)
            d=ConsoleReadInt("Introduzca denominador: ",True)
            //ojo con el denominador igual a cero
            While d=0
                  d=ConsoleReadInt("Por favor, el denominador no
puede ser cero; introduzca denominador: ",True)
            End While
            ConsoleWriteLine("Mi fracción es "+n+"/"+d)
            ReducirFraccion()
            //si la división es exacta podemos decir que el
            //resultado es un entero
            If d<>1 Then
                  ConsoleWriteLine("Simplificando mi fracción
obtenemos "+n+"/"+d)
            Else
                  ConsoleWriteLine("Simplificando mi fracción
obtenemos un entero "+n)
            End If

      End Method

End Program
```

Actividad 13

Letra del NIF

Planteamiento

Resumen explicativo

Crear un programa que calcule la letra del NIF a partir de los dígitos del DNI.

Objetivos a conseguir

1. Diseñar un programa que imprima por pantalla el NIF a partir del DNI.

2. Además el programa pedirá el nombre del usuario, pondremos color de fondo a la consola, tipo y color de letra e incluso haremos que se ejecute un archivo de sonido.

3. Analizar la estructura final del programa y hacer pequeñas variaciones sobre ellos.

Contenidos

1. Concepto de ARRAY.

2. Afianzar el concepto de RETURN.

3. Instrucciones ShowConsole, SetConsoleFont, SetConsoleBackgroundColor, SetConsoleFontColor, ConsoleReadLine y PlaySound.

Ejecución

Guión de la actividad

En primer lugar explicaremos que pasos debemos seguir para obtener la letra del **NIF** partiendo del **DNI**. El proceso es muy fácil, simplemente deberemos dividir el **DNI** entre **23** y quedarnos con el resto. Seguidamente deberemos mirar en la siguiente tabla para obtener la letra que forma parte del **NIF**:

Resto	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
Letra	***T***	***R***	***W***	***A***	***G***	***M***	***Y***	***F***	***P***	***D***	***X***	***B***	***N***	***J***	***Z***	***S***	***Q***	***V***	***H***	***L***	***C***	***K***	***E***

Ahora empezamos arrancando el KPL y creando un nuevo programa.

Escribimos en las primeras líneas de código el comentario:

```
/*Programa: Nif
      Autores: Lidia y Pablo
      Descripción: Este programa devuelve a partir del DNI, sin
      letra, el NIF, es decir, DNI+letra*/
```

Poner título al programa: Borrar MyNewProgram y poner *Nif* en su lugar.

En el Método Principal, Method Main(), definimos la variable llamada *vector* que es lo que en programación se conoce como un ARRAY, es decir, un tipo de datos complejo que consiste en una colección de variables del mismo tipo, y que es semejante al concepto matemático de vector, de la siguiente forma:

```
      // Definimos un vector de índice 23 y con elementos
      // caracteres
            Var vector As String[23]
            vector[1]="T"
            vector[2]="R"
            vector[3]="W"
            vector[4]="A"
            vector[5]="G"
            vector[6]="M"
            vector[7]="Y"
            vector[8]="F"
            vector[9]="P"
            vector[10]="D"
            vector[11]="X"
            vector[12]="B"
            vector[13]="N"
            vector[14]="J"
            vector[15]="Z"
            vector[16]="S"
            vector[17]="Q"
            vector[18]="V"
            vector[19]="H"
            vector[20]="L"
            vector[21]="C"
            vector[22]="K"
            vector[23]="E"
```

asignando valores a sus 23 elementos teniendo en cuanta la forma en la que se calcula la letra del NIF.

Escribimos el código siguiente:

```
      //mostramos la consola, ponemos fuente verdana tamaño 24,
      //color naranja de fondo y la pluma azul
      ShowConsole()
      SetConsoleFont("Verdana",24)
      SetConsoleBackgroundColor(orange)
      SetConsoleFontColor(blue)
```

Con `ShowConsole()` se muestra la consola, con `SetConsoleFont("Verdana",24)` se fija el tipo de letra en verdana y el tamaño de la letra a 24, con `SetConsoleBackgroundColor(orange)` se fija el color naranja de

fondo para la consola (el color puede especificarse a través del número de color o de su nombre) y con `SetConsoleFontColor(blue)` se fija el color azul de la letra.

A continuación tecleamos las siguientes líneas de código:

```
      //pedimos al usuario su nombre (no es necesario pero queda
      //bonito :-)
      Var Usuario As String=ConsoleReadLine("Por favor, introduce
tu nombre: ",True)
      If Usuario<>"" Then
            ConsoleWriteLine("Hola "+Usuario)
      Else
            ConsoleWriteLine("No has introducido tu nombre,
chiao!")
            Return      //salimos inmediatamente del programa
      End If
```

Con `ConsoleReadLine` se lee el texto introducido en el área de entrada de la consola y se guarda en la variable *Usuario*. El IF que hemos diseñado hace que si el usuario no introduce un nombre, el programa finaliza inmediatamente con RETURN; en caso contrario, el programa saluda al usuario con Hola más el nombre introducido.

Escribimos el código siguiente, donde definimos la variable que guardará el valor del DNI, y nos aseguramos, con el WHILE, de que ese valor introducido está entre 0 y 99999999, es decir tiene máximo 10 dígitos:

```
Var x As Integer=ConsoleReadInt("Introduce tu DNI sin letra: ",
True)
//detectamos posibles errores en la introducción del DNI
While x<=0 Or x>99999999
      ConsoleWriteLine("Por favor, tiene que ser mayor que cero y
menor que 99.999.999")
      x=ConsoleReadInt("Introduce tu DNI sin letra: ", True)
End While
```

Definimos la variable resto teniendo en cuenta que lo que queremos hacer es dividir el **DNI** entre **23** y quedarnos con el resto (recuerda que x Mod 23 es el resto de dividir x entre 23):

```
      //se calcula el resto de dividir x entre 23
      Var resto As Integer=x Mod 23
```

Tecleamos las líneas siguientes para que el programa imprima por pantalla la solución:

```
      ConsoleWriteLine("")
      ConsoleWriteLine(Usuario+", tu NIF es "+x+"-
"+vector[resto+1])
```

Finalmente añadimos un efecto de sonido:

```
      //efecto de sonido durante 2 segundos, ¿queda bien?
      PlaySound("Utopia Asterisk.wav")
      Delay(2000)
```

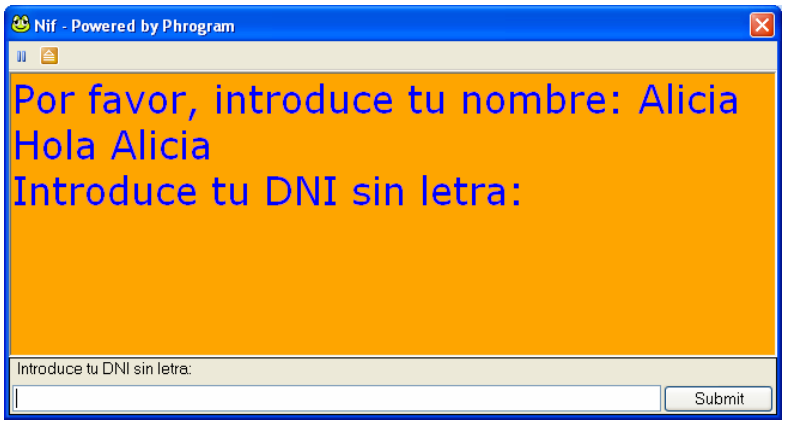
La acción `PlaySound("Utopia Asterisk.wav")` ejecuta el archivo de sonido llamado *Utopia Asterisk.wav* que está en C:\Documents and Settings\kpl\Mis

documentos\My Phrogram Files\Media Files\Sounds. Con `Delay(2000)` retardamos el efecto 2 segundos.

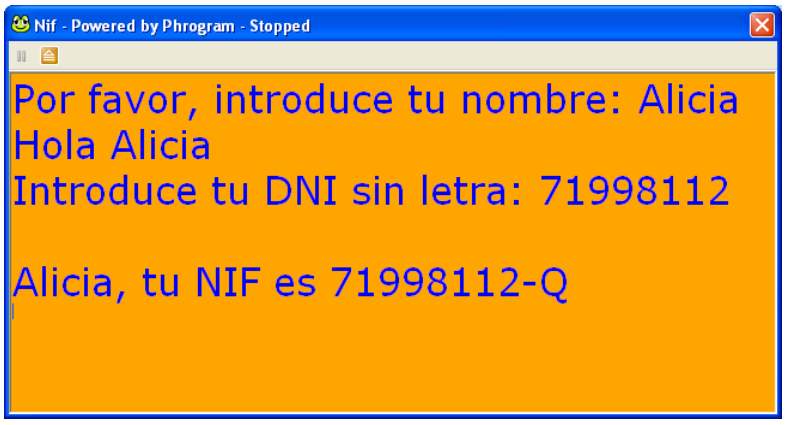
Guardamos el archivo picando en Archivo/Guardar como nuevo programa y escribiremos el nombre *Nif*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

Al ejecutar el programa (F5), nos pedirá un nombre, introducimos Alicia, damos intro y obtenemos:



Introducimos por ejemplo el DNI 71998112, pulsamos intro y nos sale la siguiente ventana acompaña de un efecto de sonido:



Ejecuta el programa y prueba con tu DNI.

Haz pruebas y cambia en el código fuente el color de fondo, el tipo y color de letra, el efecto de sonido, etc.

Resolución

Código fuente 13

```
/*Programa: Nif
      Autores: Lidia y Pablo
      Descripción: Este programa devuelve a partir del DNI, sin
      letra, el NIF, es decir, DNI+letra*/

Program Nif

      Method Main()

            // Definimos un vector de índice 23 y con elementos
            // caracteres
            Var vector As String[23]
            vector[1]="T"
            vector[2]="R"
            vector[3]="W"
            vector[4]="A"
            vector[5]="G"
            vector[6]="M"
            vector[7]="Y"
            vector[8]="F"
            vector[9]="P"
            vector[10]="D"
            vector[11]="X"
            vector[12]="B"
            vector[13]="N"
            vector[14]="J"
            vector[15]="Z"
            vector[16]="S"
            vector[17]="Q"
            vector[18]="V"
            vector[19]="H"
            vector[20]="L"
            vector[21]="C"
            vector[22]="K"
            vector[23]="E"

            // mostramos la consola, ponemos fuente verdana tamaño
            // 24, color naranja de fondo y la pluma azul
            ShowConsole()
            SetConsoleFont("Verdana",24)
            SetConsoleBackgroundColor(orange)
            SetConsoleFontColor(blue)

            // pedimos al usuario su nombre (no es necesario pero
            // queda bonito :-)
            Var Usuario As String=ConsoleReadLine("Por favor,
introduce tu nombre: ",True)
            If Usuario<>"" Then
                  ConsoleWriteLine("Hola "+Usuario)
            Else
                  ConsoleWriteLine("No has introducido tu nombre,
chiao!")
                  Return    //salimos indemiatamente del programa
```

```
            End If

            Var x As Integer=ConsoleReadInt("Introduce tu DNI sin
letra: ", True)
            //detectamos posibles errores al introducir el DNI
            While x<=0 Or x>99999999
                  ConsoleWriteLine("Por favor, tiene que ser mayor
que cero y menor que 99.999.999")
                  x=ConsoleReadInt("Introduce tu DNI sin letra: ",
True)
            End While

            //se calcula el resto de dividir x entre 23
            Var resto As Integer=x Mod 23

            ConsoleWriteLine("")
            ConsoleWriteLine(Usuario+", tu NIF es "+x+"-
"+vector[resto+1])
            //efecto de sonido durante 2 segundos, ¿queda bien?
            PlaySound("Utopia Asterisk.wav")
            Delay(2000)

      End Method

End Program
```

Actividad 14

Contador de caracteres

Planteamiento

Resumen explicativo

Crear un programa que cuenta en un texto introducido por el usuario el número de caracteres totales, el número de espacios en blanco, el número de dígitos numéricos y el número de vocales.

Objetivos a conseguir

1. Diseñar un programa que cuenta en un texto introducido por el usuario el número de caracteres totales, el número de espacios en blanco, el número de dígitos numéricos y el número de vocales.

2. Analizar la estructura final del programa y hacer pequeñas variaciones sobre él.

Contenidos

Instrucción ConsoleReadKey.

Ejecución

Guión de la actividad

Arrancar el KPL y crear un nuevo programa.

Escribimos en las primeras líneas de código un comentario inicial:

```
/*Programa: ContarCaracteres
      Autores: Lidia y Pablo
      Descripción: Este programa cuenta en un texto introducido
      por el usuario el número de caracteres totales, el número de
      espacios en blanco, el número de dígitos numéricos y el
      número de vocales*/
```

Poner título al programa: Borrar MyNewProgram y poner *ContarCaracteres* en su lugar.

En el Método Principal, Method Main(), definimos las variables que necesitamos de la siguiente forma:

```
      Var caracter As String
      Var caracteres As Integer     //contador de sílabas
      Var blancos As Integer        //contador de blancos
      Var digitos As Integer        //contador de digitos
      Var vocales As Integer        //contador de vocales
```

Escribimos el código siguiente:

```
      While caracter<>"."
            caracter=ConsoleReadKey()
            ConsoleWrite(caracter)
            caracteres=caracteres+1

            If caracter=" " Then
                  blancos=blancos+1
            End If
            If caracter="0" Or caracter="1" Or caracter="2" Or
caracter="3" Or caracter="4" Or caracter="5" Or caracter="6" Or
caracter="7" Or caracter="8" Or caracter="9" Then
                  digitos=digitos+1
            End If

            If caracter="a" Or caracter="e" Or caracter="i" Or
caracter="o" Or caracter="u" Then
                  vocales=vocales+1
            End If

      End While
```

La variable `caracter=ConsoleReadKey()` lee una tecla presionada por el usuario en el área de entrada de la consola y devuelve el carácter presionado. Para cada carácter, el programa actualizará los contadores de caracteres, espacios en blanco, dígitos numéricos y vocales. Todo esto se ejecutará mientras la variable *caracter* sea distinto de “.” (avisaremos al usuario que el proceso finaliza cuando se teclea un punto “.”)

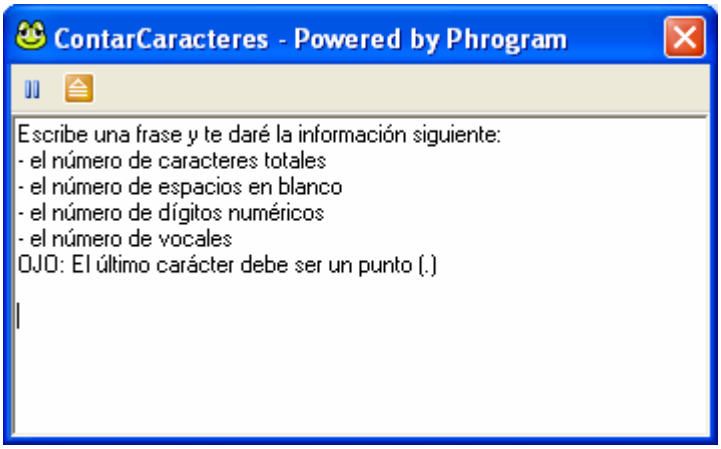
Finalmente añadimos el código siguiente para imprimir por pantalla los resultados:

```
      //imprimir el resultado
      ConsoleWriteLine("")
      ConsoleWriteLine("")
      ConsoleWriteLine("RESULTADOS")
      ConsoleWriteLine("número de caracteres totales =
"+caracteres)
      ConsoleWriteLine("número de espacios en blancos = "+blancos)
      ConsoleWriteLine("número de dígitos numéricos = "+digitos)
      ConsoleWriteLine("número de vocales = "+vocales)
```

Guardamos el archivo picando sobre el icono  o Archivo/Guardar, y escribiremos el nombre *ContarCaracteres*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

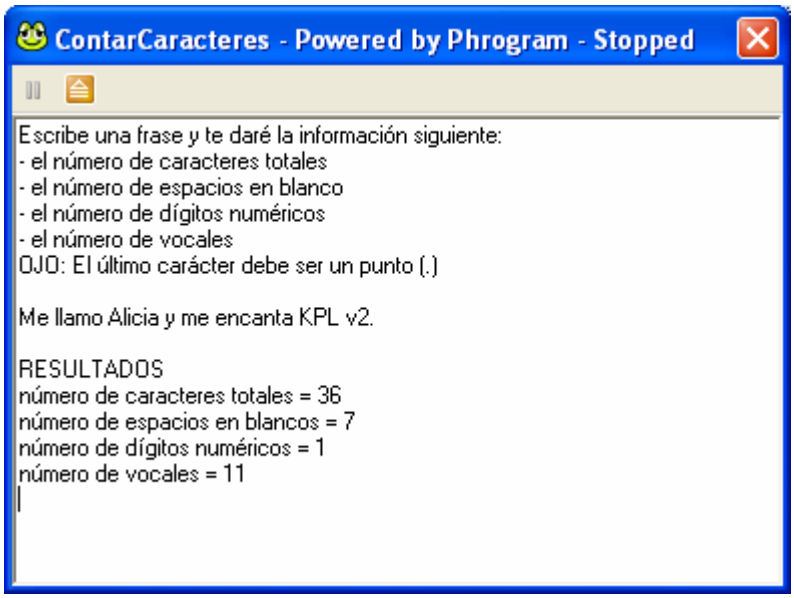
Al ejecutar el programa (F5), nos sale la siguiente ventana con una explicación inicial y en la que el cursor espera que el usuario introduzca un texto:



Introducimos por ejemplo:

Me llamo Alicia y me encanta KPL v2.

En el mismo momento de teclear el punto el programa nos muestra la ventana:



Ejecuta el programa y prueba a introducir otro texto.

¿Qué tendríamos que hacer en el código fuente para que el programa nos cuente el número de veces que aparece la sílaba “pa”?.

Resolución

Código fuente 14

```
/*Programa: ContarCaracteres
      Autores: Lidia y Pablo
      Descripción: Este programa cuenta en un texto introducido
      por el usuario el número de caracteres totales, el número
      de espacios en blanco, el número de dígitos numéricos y el
      número de vocales*/

Program ContarCaracteres

      Method Main()

            Var caracter As String
            Var caracteres As Integer     //contador de sílabas
            Var blancos As Integer        //contador de blancos
            Var digitos As Integer        //contador de digitos
            Var vocales As Integer        //contador de vocales

            ConsoleWriteLine("Escribe una frase y te daré la
información siguiente:")
            ConsoleWriteLine("- el número de caracteres totales")
            ConsoleWriteLine("- el número de espacios en blanco")
            ConsoleWriteLine("- el número de dígitos numéricos")
            ConsoleWriteLine("- el número de vocales")
            ConsoleWriteLine("OJO: El último carácter debe ser un
punto (.)")
            ConsoleWriteLine("")

            //si el carácter introducido es distinto de "." se
            //ejecuta While
            While caracter<>"."
                  caracter=ConsoleReadKey()
                  ConsoleWrite(caracter)
                  caracteres=caracteres+1
                  If caracter=" " Then
                        blancos=blancos+1
                  End If
                  If caracter="0" Or caracter="1" Or caracter="2"
Or caracter="3" Or caracter="4" Or caracter="5" Or caracter="6" Or
caracter="7" Or caracter="8" Or caracter="9" Then
                        digitos=digitos+1
                  End If
                  If caracter="a" Or caracter="e" Or caracter="i"
Or caracter="o" Or caracter="u" Then
                        vocales=vocales+1
                  End If
            End While

            //imprimir el resultado
            ConsoleWriteLine("")
            ConsoleWriteLine("")
            ConsoleWriteLine("RESULTADOS")
            ConsoleWriteLine("número de caracteres totales =
"+caracteres)
```

```
            ConsoleWriteLine("número de espacios en blancos =
"+blancos)
            ConsoleWriteLine("número de dígitos numéricos =
"+digitos)
            ConsoleWriteLine("número de vocales = "+vocales)

      End Method

End Program
```

Actividad 15

Simulador de Lotería Primitiva

Planteamiento

Resumen explicativo

Crear un programa que simule el juego de la lotería primitiva.

Objetivos a conseguir

1. Diseñar un programa que simule el juego de la lotería primitiva hasta 924 apuestas. El programa preguntará al usuario si desea jugar más apuestas una vez marcados 6 números. El sistema será capaz de generar una combinación de 6 números más complementario de forma aleatoria. Además el programa listará la combinación ganadora ordenada y el complementario y nos dirá cuantos aciertos, incluido el complementario, ha tenido el usuario.

2. Analizar la estructura final del programa.

Contenidos

1. Afianzar el concepto de METHOD, ARRAY, RETURN.

2. Instrucción Random.

3. Ordenar elementos en un ARRAY.

Ejecución

Guión de la actividad

Arrancar el KPL y crear un nuevo programa.

Escribimos en las primeras líneas de código un comentario inicial:

```
/*Programa: Primitiva
      Autores: Lidia y Pablo
      Descripción: Programa que simula el juego de la lotería
      primitiva*/
```

Poner título al programa: Borrar MyNewProgram y poner *Primitiva* en su lugar.

Primero definimos dos variables tipo ARRAY (premiados tendrá 7 elementos y boleto 12, pues marcar 12 números supone jugar 924 apuestas) que vamos a necesitar en distintos métodos:

```
Var premiados As Integer[7]
Var boleto As Integer[12]
```

Vamos a definir 3 métodos antes de ir al Método Principal:

- Method CubrirBoleto ()
- Method GenerarNumeros ()
- Method Informar ()

En el primero escribimos el código:

```
Method CubrirBoleto()
      Var i As Integer

      boleto[1]=ConsoleReadInt("Introduzca número 1: ",True)
      While boleto[1]<1 Or boleto[1]>49
            boleto[1]=ConsoleReadInt("Por favor, introduzca bien
número 1: ",True)
      End While

      boleto[2]=ConsoleReadInt("Introduzca número 2: ",True)
      While boleto[2]<1 Or boleto[2]>49 Or boleto[2]=boleto[1]
            boleto[2]=ConsoleReadInt("Por favor, introduzca bien
número 2: ",True)
      End While

      boleto[3]=ConsoleReadInt("Introduzca número 3: ",True)
      While boleto[3]<1 Or boleto[3]>49 Or boleto[3]=boleto[1] Or
boleto[3]=boleto[2]
            boleto[3]=ConsoleReadInt("Por favor, introduzca bien
número 3: ",True)
      End While

      boleto[4]=ConsoleReadInt("Introduzca número 4: ",True)
      While boleto[4]<1 Or boleto[4]>49 Or boleto[4]=boleto[1] Or
boleto[4]=boleto[2] Or boleto[4]=boleto[3]
            boleto[4]=ConsoleReadInt("Por favor, introduzca bien
número 4: ",True)
      End While

      boleto[5]=ConsoleReadInt("Introduzca número 5: ",True)
      While boleto[5]<1 Or boleto[5]>49 Or boleto[5]=boleto[1] Or
boleto[5]=boleto[2] Or boleto[5]=boleto[3] Or boleto[5]=boleto[4]
            boleto[5]=ConsoleReadInt("Por favor, introduzca bien
número 5: ",True)
      End While

      boleto[6]=ConsoleReadInt("Introduzca número 6: ",True)
      While boleto[6]<1 Or boleto[6]>49 Or boleto[6]=boleto[1] Or
boleto[6]=boleto[2] Or boleto[6]=boleto[3] Or boleto[6]=boleto[4]
Or boleto[6]=boleto[5]
            boleto[6]=ConsoleReadInt("Por favor, introduzca bien
número 6: ",True)
```

```
      End While

      boleto[7]=ConsoleReadInt("Si desea jugar 7 apuestas,
introduzca número 7; en caso contrario pulse cero (0): ",True)
      While boleto[7]<0 Or boleto[7]>49 Or boleto[7]=boleto[1] Or
boleto[7]=boleto[2] Or boleto[7]=boleto[3] Or boleto[7]=boleto[4]
Or boleto[7]=boleto[5] Or boleto[7]=boleto[6]
            boleto[7]=ConsoleReadInt("Por favor, introduzca bien
número 7 o teclee cero (0) si desea salir: ",True)
      End While
      If (boleto[7]=0) Then
            For i=8 To 12
                  boleto[i]=0
            Next
            Return
      End If

      boleto[8]=ConsoleReadInt("Si desea jugar 28 apuestas,
introduzca número 8; en caso contrario pulse cero (0): ",True)
      While boleto[8]<0 Or boleto[8]>49 Or boleto[8]=boleto[1] Or
boleto[8]=boleto[2] Or boleto[8]=boleto[3] Or boleto[8]=boleto[4]
Or boleto[8]=boleto[5] Or boleto[8]=boleto[6] Or
boleto[8]=boleto[7]
            boleto[8]=ConsoleReadInt("Por favor, introduzca bien
número 8 o teclee cero (0) si desea salir: ",True)
      End While
      If (boleto[8]=0) Then
            For i=9 To 12
                  boleto[i]=0
            Next
            Return
      End If

      boleto[9]=ConsoleReadInt("Si desea jugar 84 apuestas,
introduzca número 9; en caso contrario pulse cero (0): ",True)
      While boleto[9]<0 Or boleto[9]>49 Or boleto[9]=boleto[1] Or
boleto[9]=boleto[2] Or boleto[9]=boleto[3] Or boleto[9]=boleto[4]
Or boleto[9]=boleto[5] Or boleto[9]=boleto[6] Or
boleto[9]=boleto[7] Or boleto[9]=boleto[8]
            boleto[9]=ConsoleReadInt("Por favor, introduzca bien
número 9 o teclee cero (0) si desea salir: ",True)
      End While
      If (boleto[9]=0) Then
            For i=10 To 12
                  boleto[i]=0
            Next
            Return
      End If

      boleto[10]=ConsoleReadInt("Si desea jugar 210 apuestas,
introduzca número 10; en caso contrario pulse cero (0): ",True)
      While boleto[10]<0 Or boleto[10]>49 Or boleto[10]=boleto[1]
Or boleto[10]=boleto[2] Or boleto[10]=boleto[3] Or
boleto[10]=boleto[4] Or boleto[10]=boleto[5] Or
boleto[10]=boleto[6] Or boleto[10]=boleto[7] Or
boleto[10]=boleto[8] Or boleto[10]=boleto[9]
            boleto[10]=ConsoleReadInt("Por favor, introduzca bien
número 10 o teclee cero (0) si desea salir: ",True)
      End While
      If (boleto[10]=0) Then
            For i=11 To 12
```

```
                  boleto[i]=0
            Next
            Return
      End If

      boleto[11]=ConsoleReadInt("Si desea jugar 462 apuestas,
introduzca número 11; en caso contrario pulse cero (0): ",True)
      While boleto[11]<0 Or boleto[11]>49 Or boleto[11]=boleto[1]
Or boleto[11]=boleto[2] Or boleto[11]=boleto[3] Or
boleto[11]=boleto[4] Or boleto[11]=boleto[5] Or
boleto[11]=boleto[6] Or boleto[11]=boleto[7] Or
boleto[11]=boleto[8] Or boleto[11]=boleto[9] Or
boleto[11]=boleto[10]
            boleto[11]=ConsoleReadInt("Por favor, introduzca bien
número 11 o teclee cero (0) si desea salir: ",True)
      End While
      If (boleto[11]=0) Then
            boleto[12]=0
            Return
      End If

      boleto[12]=ConsoleReadInt("Si desea jugar 924 apuestas,
introduzca número 12; en caso contrario pulse cero (0): ",True)
      While boleto[12]<0 Or boleto[12]>49 Or boleto[12]=boleto[1]
Or boleto[12]=boleto[2] Or boleto[12]=boleto[3] Or
boleto[12]=boleto[4] Or boleto[12]=boleto[5] Or
boleto[12]=boleto[6] Or boleto[12]=boleto[7] Or
boleto[12]=boleto[8] Or boleto[12]=boleto[9] Or
boleto[12]=boleto[10] Or boleto[12]=boleto[11]
            boleto[12]=ConsoleReadInt("Por favor, introduzca bien
número 12 o teclee cero (0) si desea salir: ",True)
      End While

End Method
```

En este método vamos pidiendo al usuario que introduzca los números que desea jugar, asegurándonos de que no los repite y que el número está entre 1 y 49 (entre 0 y 49, si estamos con `boleto[i]`, para i ≥ 7). Además, cuando llegamos al séptimo número y sucesivos, el sistema le indicará al usuario si quiere seguir jugando o dejarlo. En caso de querer dejarlo, la variable boleto tomará el valor cero a partir de ese elemento y saldremos del método con RETURN.

En el segundo método escribimos:

```
Method GenerarNumeros()

//con este método generamos la combinación ganadora y la ordenamos
      Var auxiliar As Integer
      Var g As Integer
      Var h As Integer

      premiados[1]=Random(1,49)

      premiados[2]=Random(1,49)
      While premiados[2]=premiados[1]
            premiados[2]=Random(1,49)
      End While

      premiados[3]=Random(1,49)
      While premiados[3]=premiados[1] Or premiados[3]=premiados[2]
```

```
            premiados[3]=Random(1,49)
      End While

      premiados[4]=Random(1,49)
      While premiados[4]=premiados[1] Or premiados[4]=premiados[2]
Or premiados[4]=premiados[3]
            premiados[4]=Random(1,49)
      End While

      premiados[5]=Random(1,49)
      While premiados[5]=premiados[1] Or premiados[5]=premiados[2]
Or premiados[5]=premiados[3] Or premiados[5]=premiados[4]
            premiados[5]=Random(1,49)
      End While

      premiados[6]=Random(1,49)
      While premiados[6]=premiados[1] Or premiados[6]=premiados[2]
Or premiados[6]=premiados[3] Or premiados[6]=premiados[4] Or
premiados[6]=premiados[5]
            premiados[6]=Random(1,49)
      End While

      premiados[7]=Random(1,49)
      While premiados[7]=premiados[1] Or premiados[7]=premiados[2]
Or premiados[7]=premiados[3] Or premiados[7]=premiados[4] Or
premiados[7]=premiados[5] Or premiados[7]=premiados[6]
            premiados[7]=Random(1,49)
      End While

      //ordenamos la combinación ganadora de 6 números
      For g=2 To 6
            auxiliar=premiados[g]
            h=g-1
            While h>=1 And auxiliar<premiados[h]
                  premiados[h+1]=premiados[h]
                  h=h-1
            End While
            premiados[h+1]=auxiliar
      Next
End Method
```

La función RANDOM devuelve un número entero al azar entre 1 y 49. Cada número que genera el programa comprueba que no ha sido extraído anteriormente y si es así genera otro hasta que sea diferente de todos los anteriores. El premiados[7] será el complementario. Al final del método se ordena la combinación de 6 números con un FOR.

A continuación tecleamos el tercer método que simplemente lo tenemos para escribir información por pantalla cuando lo invoquemos:

```
Method Informar()
      //listamos información
      ConsoleWriteLine("Si usted marca 6 números, jugará 1
apuesta")
      ConsoleWriteLine("Si usted marca 7 números, jugará 7
apuestas")
      ConsoleWriteLine("Si usted marca 8 números, jugará 28
apuestas")
      ConsoleWriteLine("Si usted marca 9 números, jugará 84
apuestas")
```

```
      ConsoleWriteLine("Si usted marca 10 números, jugará 210
apuestas")
      ConsoleWriteLine("Si usted marca 11 números, jugará 462
apuestas")
      ConsoleWriteLine("Si usted marca 12 números, jugará 924
apuestas")
End Method
```

En el Método Principal, Method Main(), definimos las variables que necesitamos de la siguiente forma:

```
      Var acertados As Integer
      Var acertadocomplementario As Boolean
      Var i As Integer
      Var n As Integer
      Var m As Integer
      Var f As Integer
```

Escribimos el código siguiente:

```
      ConsoleWriteLine("=====JUEGO DE LA PRIMITIVA=====")
```

Tecleamos:

```
      Informar()        //invocamos nuestro método Informar

      CubrirBoleto()    //invocamos nuestro método CubrirBoleto
```

con lo que invocamos primero el método Informar y después el CubrirBoleto.

A continuación dejamos dos lineas en blanco con:

```
      ConsoleWriteLine("")
      ConsoleWriteLine("")
```

Después introducimos las líneas de programación siguientes invocando el método GenerarNumeros y añadiendo algún efecto simpático:

```
      //esto es para retardar y dar emoción al asunto
      ConsoleWriteLine("Generando combinación ganadora")
      Delay(2000)
      ConsoleWriteLine("¡Qué nervios!")
      Delay(3000)
      GenerarNumeros()  //invocamos nuestro método generarnumeros
      ConsoleWriteLine("Combinación ganadora: "+premiados[1]+" -
"+premiados[2]+" - "+premiados[3]+" - "+premiados[4]+" -
"+premiados[5]+" - "+premiados[6]+" y el complementario
"+premiados[7])
```

Escribimos a continuación:

```
      ConsoleWriteLine("")
      ConsoleWriteLine("")
```

Para contar el número de aciertos y si hemos acertado el complementario diseñamos un FOR en el que comparamos los elementos de premiados con los

elementos de boleto. En caso de que coincida alguno, la variable aciertos aumentará en una unidad, y si el acierto se produce en premiados[7], es decir, en el complementario, la variable acertadocomplementario tomará el valor True:

```
For n=1 To 7
      For m=1 To 12
            If (premiados[n]=boleto[m]) Then
                  If n<>7 Then
                        acertados=acertados+1
                  Else
                        acertadocomplementario=True
                  End If
            End If
      Next
Next
```

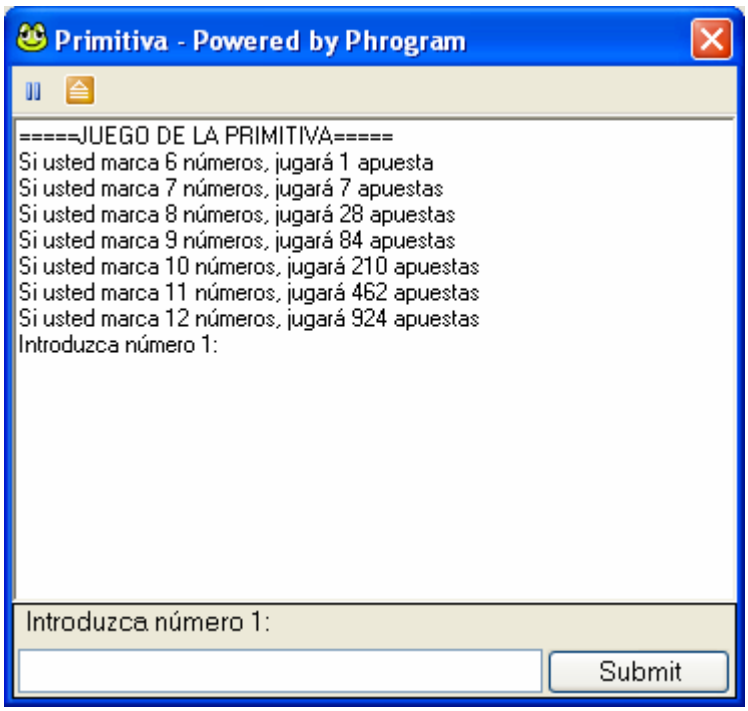
Por último imprimimos por pantalla el resumen de aciertos:

```
ConsoleWrite("Usted ha tenido "+acertados+" aciertos")
If acertadocomplementario=True Then
      ConsoleWriteLine(" y el complementario")
End If
```

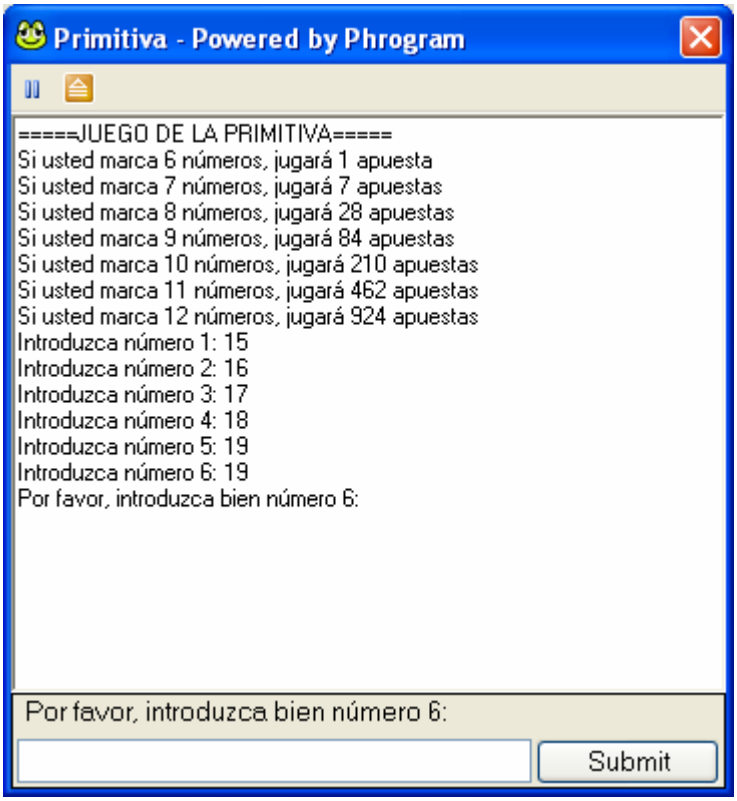
Guardamos el archivo picando sobre el icono  o Archivo/Guardar, y escribiremos el nombre *Primitiva*, seleccionando la carpeta donde quiera ir almacenando todas las actividades. Por defecto la carpeta elegida es My Phrograms Files que está en Mis Documentos.

En el caso de existir algún error en el CÓDIGO FUENTE de nuestro programa, KPL nos informa con un aviso de ERROR y el cursor se sitúa sobre la línea y secuencia errónea. Lo rectificaremos.

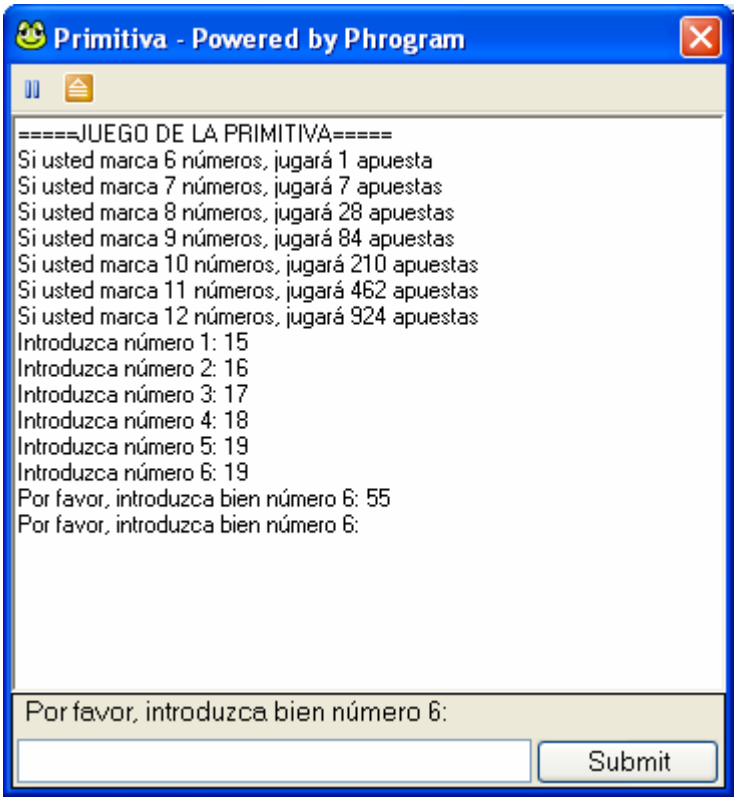
Al ejecutar el programa (F5), nos sale la siguiente ventana con una explicación inicial y esperando que el usuario introduzca el primer número en la consola:



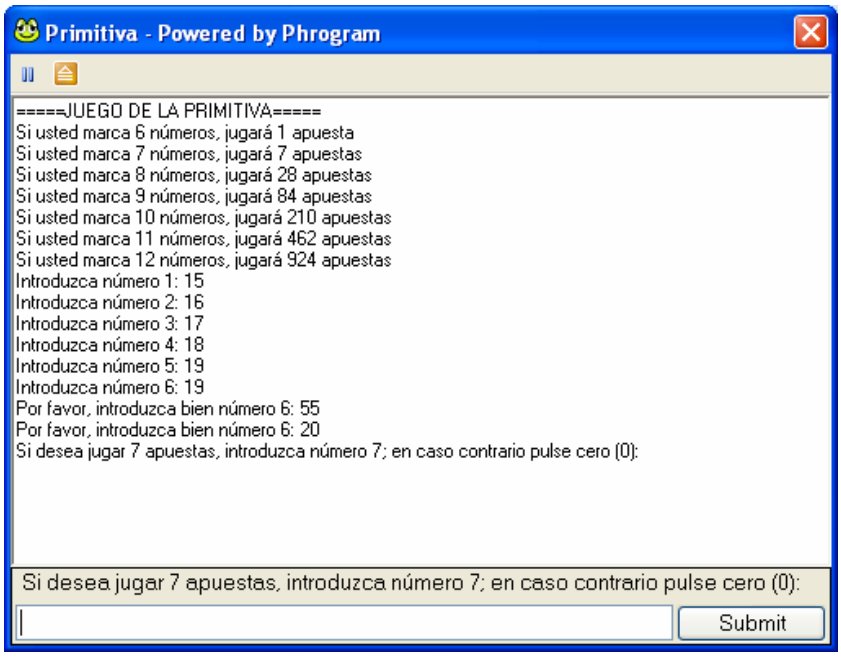
Introducimos por ejemplo 15,16,17,18,19 y 19 y llegamos a:



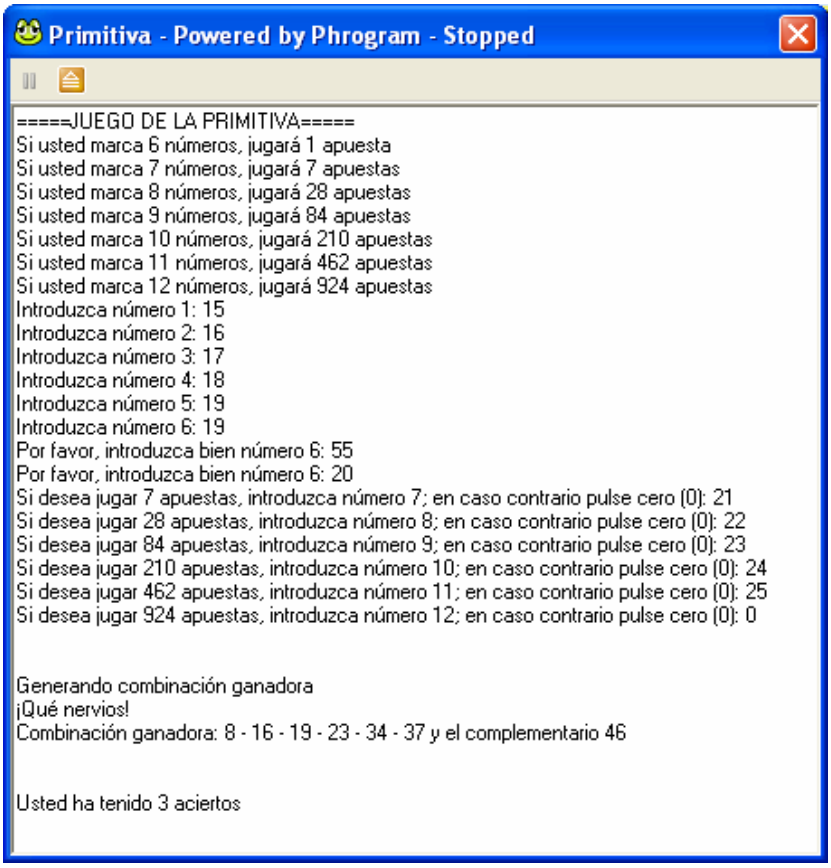
El sistema nos dice que hemos introducido mal el sexto número y que lo volvamos a introducir (¡está repetido!). Introducimos 55 y ¿qué obtenemos?



Qué también está mal pues no son válidos los números menores de 1 y mayores de 49. Introducimos 20 y obtenemos:



Vamos a jugar hasta 462 apuestas, así que introduciremos 11 números (por ejemplo añadimos el 21, 22, 23, 24 y 25) y cuando nos diga el programa si deseamos jugar 924 apuestas teclearemos un cero (0), obteniendo por tanto en este ejemplo:



¡Sólo he acertado 3! Espero que tú tengas más suerte. Ejecuta el programa y prueba con otros números.

Resolución

Código fuente 15

```
/*Programa: Primitiva
      Autores: Lidia y Pablo
      descripción: Programa que simula el juego de la lotería
      primitiva*/

Program Primitiva

      Var premiados As Integer[7]
      Var boleto As Integer[12]

      Method CubrirBoleto()
            Var i As Integer

            boleto[1]=ConsoleReadInt("Introduzca número 1: ",True)
            While boleto[1]<1 Or boleto[1]>49
                  boleto[1]=ConsoleReadInt("Por favor, introduzca
bien número 1: ",True)
            End While

            boleto[2]=ConsoleReadInt("Introduzca número 2: ",True)
            While boleto[2]<1 Or boleto[2]>49 Or
boleto[2]=boleto[1]
                  boleto[2]=ConsoleReadInt("Por favor, introduzca
bien número 2: ",True)
            End While

            boleto[3]=ConsoleReadInt("Introduzca número 3: ",True)
            While boleto[3]<1 Or boleto[3]>49 Or
boleto[3]=boleto[1] Or boleto[3]=boleto[2]
                  boleto[3]=ConsoleReadInt("Por favor, introduzca
bien número 3: ",True)
            End While

            boleto[4]=ConsoleReadInt("Introduzca número 4: ",True)
            While boleto[4]<1 Or boleto[4]>49 Or
boleto[4]=boleto[1] Or boleto[4]=boleto[2] Or boleto[4]=boleto[3]
                  boleto[4]=ConsoleReadInt("Por favor, introduzca
bien número 4: ",True)
            End While

            boleto[5]=ConsoleReadInt("Introduzca número 5: ",True)
            While boleto[5]<1 Or boleto[5]>49 Or
boleto[5]=boleto[1] Or boleto[5]=boleto[2] Or boleto[5]=boleto[3]
Or boleto[5]=boleto[4]
                  boleto[5]=ConsoleReadInt("Por favor, introduzca
bien número 5: ",True)
            End While

            boleto[6]=ConsoleReadInt("Introduzca número 6: ",True)
            While boleto[6]<1 Or boleto[6]>49 Or
boleto[6]=boleto[1] Or boleto[6]=boleto[2] Or boleto[6]=boleto[3]
Or boleto[6]=boleto[4] Or boleto[6]=boleto[5]
                  boleto[6]=ConsoleReadInt("Por favor, introduzca
bien número 6: ",True)
            End While
```

```
            boleto[7]=ConsoleReadInt("Si desea jugar 7 apuestas,
introduzca número 7; en caso contrario pulse cero (0): ",True)
            While boleto[7]<0 Or boleto[7]>49 Or
boleto[7]=boleto[1] Or boleto[7]=boleto[2] Or boleto[7]=boleto[3]
Or boleto[7]=boleto[4] Or boleto[7]=boleto[5] Or
boleto[7]=boleto[6]
                  boleto[7]=ConsoleReadInt("Por favor, introduzca
bien número 7 o teclee cero (0) si desea salir: ",True)
            End While
            If (boleto[7]=0) Then
                  For i=8 To 12
                        boleto[i]=0
                  Next
                  Return
            End If

            boleto[8]=ConsoleReadInt("Si desea jugar 28 apuestas,
introduzca número 8; en caso contrario pulse cero (0): ",True)
            While boleto[8]<0 Or boleto[8]>49 Or
boleto[8]=boleto[1] Or boleto[8]=boleto[2] Or boleto[8]=boleto[3]
Or boleto[8]=boleto[4] Or boleto[8]=boleto[5] Or
boleto[8]=boleto[6] Or boleto[8]=boleto[7]
                  boleto[8]=ConsoleReadInt("Por favor, introduzca
bien número 8 o teclee cero (0) si desea salir: ",True)
            End While
            If (boleto[8]=0) Then
                  For i=9 To 12
                        boleto[i]=0
                  Next
                  Return
            End If

            boleto[9]=ConsoleReadInt("Si desea jugar 84 apuestas,
introduzca número 9; en caso contrario pulse cero (0): ",True)
            While boleto[9]<0 Or boleto[9]>49 Or
boleto[9]=boleto[1] Or boleto[9]=boleto[2] Or boleto[9]=boleto[3]
Or boleto[9]=boleto[4] Or boleto[9]=boleto[5] Or
boleto[9]=boleto[6] Or boleto[9]=boleto[7] Or boleto[9]=boleto[8]
                  boleto[9]=ConsoleReadInt("Por favor, introduzca
bien número 9 o teclee cero (0) si desea salir: ",True)
            End While
            If (boleto[9]=0) Then
                  For i=10 To 12
                        boleto[i]=0
                  Next
                  Return
            End If

            boleto[10]=ConsoleReadInt("Si desea jugar 210
apuestas, introduzca número 10; en caso contrario pulse cero (0):
",True)


            While boleto[10]<0 Or boleto[10]>49 Or
boleto[10]=boleto[1] Or boleto[10]=boleto[2] Or
boleto[10]=boleto[3] Or boleto[10]=boleto[4] Or
boleto[10]=boleto[5] Or boleto[10]=boleto[6] Or
boleto[10]=boleto[7] Or boleto[10]=boleto[8] Or
boleto[10]=boleto[9]
                  boleto[10]=ConsoleReadInt("Por favor, introduzca
bien número 10 o teclee cero (0) si desea salir: ",True)
```

```
            End While
            If (boleto[10]=0) Then
                  For i=11 To 12
                        boleto[i]=0
                  Next
                  Return
            End If

            boleto[11]=ConsoleReadInt("Si desea jugar 462
apuestas, introduzca número 11; en caso contrario pulse cero (0):
",True)
            While boleto[11]<0 Or boleto[11]>49 Or
boleto[11]=boleto[1] Or boleto[11]=boleto[2] Or
boleto[11]=boleto[3] Or boleto[11]=boleto[4] Or
boleto[11]=boleto[5] Or boleto[11]=boleto[6] Or
boleto[11]=boleto[7] Or boleto[11]=boleto[8] Or
boleto[11]=boleto[9] Or boleto[11]=boleto[10]
                  boleto[11]=ConsoleReadInt("Por favor, introduzca
bien número 11 o teclee cero (0) si desea salir: ",True)
            End While
            If (boleto[11]=0) Then
                  boleto[12]=0
                  Return
            End If

            boleto[12]=ConsoleReadInt("Si desea jugar 924
apuestas, introduzca número 12; en caso contrario pulse cero (0):
",True)
            While boleto[12]<0 Or boleto[12]>49 Or
boleto[12]=boleto[1] Or boleto[12]=boleto[2] Or
boleto[12]=boleto[3] Or boleto[12]=boleto[4] Or
boleto[12]=boleto[5] Or boleto[12]=boleto[6] Or
boleto[12]=boleto[7] Or boleto[12]=boleto[8] Or
boleto[12]=boleto[9] Or boleto[12]=boleto[10] Or
boleto[12]=boleto[11]
                  boleto[12]=ConsoleReadInt("Por favor, introduzca
bien número 12 o teclee cero (0) si desea salir: ",True)
            End While

      End Method

      Method GenerarNumeros()

            //con este método generamos la combinación ganadora y
            //la ordenamos
            Var auxiliar As Integer
            Var g As Integer
            Var h As Integer

            premiados[1]=Random(1,49)

            premiados[2]=Random(1,49)

            While premiados[2]=premiados[1]
                  premiados[2]=Random(1,49)
            End While

            premiados[3]=Random(1,49)
            While premiados[3]=premiados[1] Or
premiados[3]=premiados[2]
                  premiados[3]=Random(1,49)
```

```
            End While

            premiados[4]=Random(1,49)
            While premiados[4]=premiados[1] Or
premiados[4]=premiados[2] Or premiados[4]=premiados[3]
                  premiados[4]=Random(1,49)
            End While

            premiados[5]=Random(1,49)
            While premiados[5]=premiados[1] Or
premiados[5]=premiados[2] Or premiados[5]=premiados[3] Or
premiados[5]=premiados[4]
                  premiados[5]=Random(1,49)
            End While

            premiados[6]=Random(1,49)
            While premiados[6]=premiados[1] Or
premiados[6]=premiados[2] Or premiados[6]=premiados[3] Or
premiados[6]=premiados[4] Or premiados[6]=premiados[5]
                  premiados[6]=Random(1,49)
            End While

            premiados[7]=Random(1,49)
            While premiados[7]=premiados[1] Or
premiados[7]=premiados[2] Or premiados[7]=premiados[3] Or
premiados[7]=premiados[4] Or premiados[7]=premiados[5] Or
premiados[7]=premiados[6]
                  premiados[7]=Random(1,49)
            End While

            //ordenamos la combinación ganadora de 6 números
            For g=2 To 6
                  auxiliar=premiados[g]
                  h=g-1
                  While h>=1 And auxiliar<premiados[h]
                        premiados[h+1]=premiados[h]
                        h=h-1
                  End While
                  premiados[h+1]=auxiliar
            Next

      End Method

      Method Informar()
            //listamos información
            ConsoleWriteLine("Si usted marca 6 números, jugará 1
apuesta")
            ConsoleWriteLine("Si usted marca 7 números, jugará 7
apuestas")
            ConsoleWriteLine("Si usted marca 8 números, jugará 28
apuestas")
            ConsoleWriteLine("Si usted marca 9 números, jugará 84
apuestas")
            ConsoleWriteLine("Si usted marca 10 números, jugará
210 apuestas")
            ConsoleWriteLine("Si usted marca 11 números, jugará
462 apuestas")
            ConsoleWriteLine("Si usted marca 12 números, jugará
924 apuestas")
      End Method
```

```
      Method Main()

            Var acertados As Integer
            Var acertadocomplementario As Boolean
            Var i As Integer
            Var n As Integer
            Var m As Integer
            Var f As Integer

            ConsoleWriteLine("=====JUEGO DE LA PRIMITIVA=====")

            Informar()        //invocamos nuestro método Informar

            CubrirBoleto() //invocamos nuestro método CubrirBoleto

            ConsoleWriteLine("")
            ConsoleWriteLine("")

            //esto es para retardar y dar emoción al asunto
            ConsoleWriteLine("Generando combinación ganadora")
            Delay(2000)
            ConsoleWriteLine("¡Qué nervios!")
            Delay(3000)
            GenerarNumeros()  //invocamos el método generarnumeros
            ConsoleWriteLine("Combinación ganadora:
"+premiados[1]+" - "+premiados[2]+" - "+premiados[3]+" -
"+premiados[4]+" - "+premiados[5]+" - "+premiados[6]+" y el
complementario "+premiados[7])

            ConsoleWriteLine("")
            ConsoleWriteLine("")

            For n=1 To 7
                  For m=1 To 12
                        If (premiados[n]=boleto[m]) Then
                              If n<>7 Then
                                    acertados=acertados+1
                              Else
                                    acertadocomplementario=True
                              End If
                        End If
                  Next
            Next

            ConsoleWrite("Usted ha tenido "+acertados+" aciertos")
            If acertadocomplementario=True Then
                  ConsoleWriteLine(" y el complementario")
            End If
      End Method

End Program
```

Recursos

Desde el CD-ROM puedes acceder a los siguientes recursos que te ayudarán a trabajar con KPL:

- Archivo ejecutable PhrogramSetup.exe.
- PhrogramSpanish.zip.
- Imagen de Alicia para realizar la Actividad 3.
- Adobe Reader 7.0.8 en español.

Glosario

En este glosario te presentamos los términos que son necesarios para realizar las actividades. Te puede ayudar a comprender y asimilar mejor los conceptos y si quieres profundizar todavía más, puedes consultar la Guía del Usuario Phrogram (desde la ayuda del propio programa).

A

ABS:

Función matemática predefinida disponible en KPL. Abs(x) devuelve el valor absoluto (positivo) del número x, es decir $|x|$.

ACOS:

Función matemática predefinida disponible en KPL. Acos(x) devuelve el ángulo (en radianes) cuyo coseno es x.

ARRAY:

Tipo de datos complejo que consiste en una colección de variables del mismo tipo, y se define de la forma siguiente: **DEFINE <Identificador> AS <Tipo> [<Expresión>]**. Es semejante al concepto matemático de vector.

ARCTAN:

Función matemática predefinida disponible en KPL. ArcTan(x) devuelve el ángulo (en radianes) cuya tangente es x.

ARCTAN2:

Función matemática predefinida disponible en KPL. ArcTan2(x,y) devuelve el ángulo (en radianes) cuya tangente es $\frac{y}{x}$.

ASIN:

Función matemática predefinida disponible en KPL. Asin(x) devuelve el ángulo (en radianes) cuyo seno es x.

B

BOOL:

Tipo de dato simple que designa un valor de Boolean que puede contener el valor VERDADERO (TRUE) o FALSO (FALSE).

C

CEILING:

Función matemática predefinida disponible en KPL. Ceiling(x) devuelve el entero más pequeño mayor que x, es decir el mínimo del conjunto $\{y \in Z / y \geq x\}$.

CLEARCONSOLE:

ClearConsole() limpia la consola.

COLOR:

Método que fija el color actual del sistema. No es utilizado en el objeto pluma. Se invoca con ***Color(X As integer)***, siendo ***X*** el color a utilizar.

COLORES DISPONIBLES:

El color se invocar por el nombre o a través de un número entero entre 1 y 140, según la tabla de equivalencia:

NOMBRE	ENTERO	NOMBRE	ENTERO	NOMBRE	ENTERO
AliceBlue	1	AntiqueWhite	2	Aqua	3
Aquamarine	4	Azure	5	Beige	6
Bisque	7	Black	8	BlanchedAlmond	9
Blue	10	BlueViolet	11	Brown	12
BurlyWood	13	CadetBlue	14	Chartreuse	15
Chocolate	16	Coral	17	CornflowerBlue	18
Cornsilk	19	Crimson	20	Cyan	21
DarkBlue	22	DarkCyan	23	DarkGoldenrod	24
DarkGray	25	DarkGreen	26	DarkKhaki	27
DarkMagenta	28	DarkOliveGreen	29	DarkOrange	30
DarkOrchid	31	DarkRed	32	DarkSalmon	33
DarkSeaGreen	34	DarkSlateBlue	35	DarkSlateGray	36
DarkTurquoise	37	DarkViolet	38	DeepPink	39
DeepSkyBlue	40	DimGray	41	DodgerBlue	42
FireBrick	43	FloralWhite	44	ForestGreen	45
Fuchsia	46	Gainsboro	47	GhostWhite	48
Gold	49	Goldenrod	50	Gray	51
Green	52	GreenYellow	53	Honeydew	54
HotPink	55	IndianRed	56	Indigo	57
Ivory	58	Khaki	59	Lavender	60
LavenderBlush	61	LawnGreen	62	LemonChiffon	63
LightBlue	64	LightCoral	65	LightCyan	66
LightGoldenrodYellow	67	LightGray	68	LightGreen	69
LightPink	70	LightSalmon	71	LightSeaGreen	72
LightSkyBlue	73	LightSlateGray	74	LightSteelBlue	75
LightYellow	76	Lime	77	LimeGreen	78
Linen	79	Magenta	80	Maroon	81
MediumAquamarine	82	MediumBlue	83	MediumOrchid	84

NOMBRE	ENTERO	NOMBRE	ENTERO	NOMBRE	ENTERO
MediumPurple	85	MediumSeaGreen	86	MediumSlateBlue	87
MediumSpringGreen	88	MediumTurquoise	89	MediumVioletRed	90
MidnightBlue	91	MintCream	92	MistyRose	93
Moccasin	94	NavajoWhite	95	Navy	96
OldLace	97	Olive	98	OliveDrab	99
Orange	100	OrangeRed	101	Orchid	102
PaleGoldenrod	103	PaleGreen	104	PaleTurquoise	105
PaleVioletRed	106	PapayaWhip	107	PeachPuff	108
Peru	109	Pink	110	Plum	111
PowderBlue	112	Purple	113	Red	114
RosyBrown	115	RoyalBlue	116	SaddleBrown	117
Salmon	118	SandyBrown	119	SeaGreen	120
Seashell	121	Sienna	122	Silver	123
SkyBlue	124	SlateBlue	125	SlateGray	126
Snow	127	SpringGreen	128	SteelBlue	129
Tan	130	Teal	131	Thistle	132
Tomato	133	Turquoise	134	Violet	135
Wheat	136	White	137	WhiteSmoke	138
Yellow	139	YellowGreen	140		

COMENTARIOS:

Se realizan para aclarar determinados pasos en el código fuente de un programa y no son más que simples anotaciones, que aparecen en color verde. Cuando se ejecuta el programa, no se tienen en cuenta. Pueden ser de línea o de bloque. Los de línea van precedidos de // y los de bloque empiezan por /* y acaban con */.

CONSOLEREADDECIMAL:

Método al que llamaremos con ***ConsoleReadDecimal(Prompt As String, EchoToConsole As Boolean) As Decimal***, y que lee el número decimal introducido en el área de entrada de la consola. En *Prompt* escribiremos entre comillas el texto que solicita al usuario teclear un número. Si en *EchoToConsole As Bolean* ponemos *True*, la solicitud y la respuesta del usuario serán desplegadas en la consola.

CONSOLEREADINT:

El método ***ConsoleReadInt (Prompt As String, EchoToConsole As Boolean) As Integer***, y que lee el número entero introducido en el área de entrada de la consola. En *Prompt* escribiremos entre comillas el texto que solicita al usuario teclear un número. Si en *EchoToConsole As Bolean* ponemos *True*, la solicitud y la respuesta del usuario serán desplegadas en la consola.

CONSOLEREADKEY:

ConsoleReadKey() As String lee una tecla presionada por el usuario en el área de entrada de la consola y devuelve el carácter presionado.

CONSOLEREADLINE:

ConsoleReadInt (Prompt As String, EchoToConsole As Boolean) As Integer, lee el texto introducido en el área de entrada de la consola. En *Prompt*

escribiremos entre comillas el texto que solicita al usuario teclear una cadena de caracteres. Si en *EchoToConsole As Bolean* ponemos *True*, la solicitud y la respuesta del usuario serán desplegadas en la consola.

CONSOLEWRITE:

ConsoleWrite(Message as String), con *Message* entre comillas, escribe texto en la consola, sin regresar al inicio de la línea siguiente.

CONSOLEWRITELINE:

ConsoleWriteLine(Message as String), con *Message* entre comillas, escribe texto en la consola y después regresa al inicio de la línea siguiente.

COS:

Función matemática predefinida disponible en KPL. Cos(x) devuelve el valor coseno de x (x en radianes).

COSH:

Función matemática predefinida disponible en KPL. CosH(x) devuelve el valor del coseno hiperbólico en la abscisa x, es decir $\frac{e^x + e^{-x}}{2}$.

D

DECIMAL:

Tipo de dato simple que designa un número real con 28 dígitos como máximo.

DEFINE:

Término utilizado para definir variables en la forma **DEFINE <Identificador> AS <Data Type> [= <Expression>]**

DEGREESTORADIANS:

Función matemática invocada como DegreesToRadians ***(x As Decimal) As Decimal***, siendo ***x*** un valor en grados y devuelve el ángulo ***x*** en radianes.

DELAY:

Instrucción que provoca una pausa para el tiempo especificado, es decir, si escribimos Delay(1000) la computadora hace una pausa durante 1 segundo, si fuese Delay(500), causaría en la computadora un retardo de medio segundo, y si fuera Delay(3000) causaría en la computadora un retardo de 3 segundos.

DRAWLINE:

Método que dibuja una línea en la pantalla. Se invoca con DrawLine(x_1 As Decimal, y_1 As Decimal, x_2 As Decimal, y_2 As Decimal), siendo (x_1, y_1) las coordenadas del punto inicial del segmento y (x_2, y_2) las coordenadas del punto final.

E

EXP:

Función matemática predefinida disponible en KPL. Exp(x) devuelve el valor de la exponencial de base e en la abscisa x, es decir e^x .

F

FLOOR:

Función matemática predefinida disponible en KPL. Floor(x) devuelve el entero más grande menor que x, es decir el máximo del conjunto $\{y \in Z / y \leq x\}$.

FOR:

Sentencia utilizada para construir estructuras iterativas o bucles, mediante una variable, que sirve de contador para controlar el número de iteraciones a realizar. En cada iteración la variable se incrementa en una unidad o en *valor incremento* si utilizamos **Step**. Su estructura es la siguiente:

FOR <Variable> = <Valor inicial> TO <Valor Final> [<Step> <Valor incremento>]

……

NEXT

Obsérvese la diferencia entre los dos ejemplos siguientes; al ejecutar el primero se imprime por pantalla 20 veces Hola en 20 líneas, y en el segundo se imprime por pantalla 4 veces Hola en 4 líneas.

```
Define X As Int
For X = 1 To 20
    PrintLine( "Hola")
Next
```

```
Define X As Int
For X = 1 To 20 Step 5
    PrintLine( "Hola")
Next
```

FUENTES DISPONIBLES:

Las fuentes se invocan por el nombre o como String, según la tabla de equivalencia siguiente:

NOMBRE	STRING
Arial	"Arial"
ArialBlack	"Arial Black"
ArialNarrow	"Arial Narrow"
ArialUnicodeMS	"Arial Unicode MS"
BookAntiqua	"Book Antiqua"
BookmanOldStyle	"Bookman Old Style"
Century	"Century"
CenturyGothic	"Century Gothic"
ComicSansMS	"Comic Sans MS"
Garamond	"Garamond"
Georgia	"Georgia"
Impact	"Impact"
LucidaConsole	"Lucida Console"
LucidaSans	"Lucida Sans"
LucidaSansUnicode	"Lucida Sans Unicode"
MicrosoftSansSerif	"Microsoft Sans Serif"
Symbol	"Symbol"
Tahoma	"Tahoma"
TimesNewRoman	"Times New Roman"
TraditionalArabic	"Traditional Arabic"
TrebuchetMS	"TrebuchetMS"
Verdana	"Verdana"
Webdings	"Webdings"
Wingdings	"Wingdings"

FUNCTION:

Palabra clave que seguida por un identificador válido y éste siempre seguido de dos paréntesis que pueden contener una lista de parámetros o argumentos, se transforma en un subprograma que realiza una determinada acción y que devuelve un valor. Estructura semejante a METHOD, salvo que éste no devuelve un valor. FUNCTION debe contener siempre la sentencia RETURN, que devuelve como valor de la función el resultado de los cálculos realizados. FUNCTION acaba siempre con End Function.

G

GETCOLOR:

El método ***GetColor(Red As Integer, Green As Integer, Blue As Integer)*** permite definir un nuevo color al especificar sus componentes RGB.

I

IDENTIFICADOR:

O nombre utilizado para designar variables, structures y arrays. El identificador representa el valor almacenado en dicha variable, structure o array.

IF:

Sentencia de selección que tiene la forma siguiente:

```
If <condición> Then
    Acción1
Else
    Acción2
End If
```

La ejecución de la sentencia IF consiste en evaluar la expresión de “condición”, y a continuación ejecutar o bien la “Acción 1” (si se cumple la condición), o bien la “Acción 2” (si la condición no se cumple).

INT:

Tipo de dato simple que designa un número entero.

L

LOADSPRITE:

Función que carga una imagen como sprite. Se invoca ***LoadSprite(SpriteName As String, FileName As String) As Sprite***, siendo *SpriteName* el nombre del sprite y *FileName* la ruta del archivo de la imagen a cargar.

LOG:

Función matemática predefinida disponible en KPL. Log(x) devuelve el valor del logaritmo neperiano en la abscisa x, es decir $Ln\,x$.

LOG10:

Función matemática predefinida disponible en KPL. Log(x) devuelve el valor del logaritmo decimal en la abscisa x, es decir $Log_{10}(x)$.

LOGBASE:

Función matemática predefinida disponible en KPL. LogBase(x,a) devuelve el valor del logaritmo en base a en la abscisa x, es decir $Log_{a}(x)$.

LOOP:

Sentencia utilizada para construir estructuras iterativas o bucles. LOOP puede controlarse con un valor numérico o una variable de tipo entero, por tanto su estructura se corresponderá con una de estas dos (“Acción” se repetirá n veces):

Loop n Acción End Loop	Define X As Int = n Loop X Acción End Loop

M

MAX:

Función matemática predefinida disponible en KPL. Max(x,y) devuelve el mayor valor entre x e y.

MAXIMIZE:

Maximize() maximiza la pantalla de la consola.

METHOD:

Palabra clave que seguida por un identificador válido y éste siempre seguido de dos paréntesis que pueden contener una lista de parámetros o argumentos, se transforma en un subprograma que realiza una determinada acción. Su estructura es:

Method EscribirTexto() PRINT("Hello World") End Method	Method EscribirTexto(Text As String) PRINT(Text) End Method

El Method Main( ) es el Programa Principal dentro de la estructura del programa. El Metthod se asemeja a los procedimientos utilizados en otros lenguajes de programación. La utilización de RETURN en alguna línea dentro de Method provoca que termine la ejecución del subprograma en ese momento y se volverá al punto siguiente a donde se invocó. METHOD acaba siempre con End Method.

MIN:

Función matemática predefinida disponible en KPL. Min(x,y) devuelve el menor valor entre x e y.

MOVESPRITEBYAMOUNT:

Método que mueve un sprite la cantidad especificada desde su posición actual. Se invoca ***MoveSpriteByAmount(SpriteName As String, AmountX As Decimal, AmountY As Decimal)***, siendo *SpriteName* el nombre del sprite y *AmountX* y *AmountY* la cantidad de pixeles que se moverá el sprite en la coordenada X e Y respectivamente.

MOVESPRITETOPOINT:

Método que mueve un sprite hasta el punto especificado. Se invoca ***MoveSpriteToPoint(SpriteName As String, X As Decimal, Y As Decimal)***, siendo *SpriteName* el nombre del sprite y *X* e *Y* las coordenadas a las que se moverá el sprite.

MOVETO:

Método que mueve la posición de la pluma a una nueva ubicación. Se invoca con ***MoveTo(X As Decimal, Y As Decimal)***, siendo X e Y las coordenadas de la nueva posición.

P

PALABRAS CLAVE:

Las palabras clave son parte del propio lenguaje de KPL y no pueden usarse como un identificador. Son las siguientes:

AND	AS	BOOLEAN	BREAK	CASE
CLASS	CONSTANT	CONSTRUCTOR	CONTINUE	DECIMAL
DEFINE	DO	DOWNTO	EACH	ELSE
ELSEIF	END	ENUM	EXIT	FALSE
FOR	FUNCTION	GET	GLOBAL	GOTO
HANDLES	IF	IMPLEMENTS	INHERITS	INTEGER
INTERFACE	LOOP	METHOD	MOD	MODULE
NEXT	NOT	OPERATOR	OR	PRIVATE
PROGRAM	PROPERTY	PUBLIC	RETURN	SELECT
SET	STEP	STRING	STRUCTURE	THEN
TO	TRUE	VAR	WHILE	

PEN:

Método que designa el objeto del sistema utilizado para dibujar. Se invoca con ***Pen (X As Boolean)***, siendo ***X*** True (verdadero para que la pluma dibuje al moverse) o False (falso para que la pluma no dibuje al moverse).

PENWIDTH:

Método que designa el ancho con el que pintará la pluma. Se invoca con ***PenWidtg(X As Decimal)***, siendo ***X*** el ancho en pixeles de la pluma.

PLAYSOUND:

Al invocar este método, en la forma **PlaySound(“*nombrearchivo*”)** se ejecuta el archivo de sonido llamado *nombrearchivo* que tenemos en C:\Documents and Settings\kpl\Mis documentos\My Phrogram Files\Media Files\Sounds.

POWER:

Función matemática predefinida disponible en KPL. Power(x,y) devuelve el valor x^y.

PRINT:

Método ***Print (Text As String)***, siendo Text el texto a escribir, imprime el texto dado a partir de la posición actual de la pluma. El texto es seguido por un espacio.

PRINTLINE:

Método ***PrintLine(Text As String)***, siendo Text el texto a escribir, imprime el texto dado en la posición actual de la pluma. La posición de la pluma es colocada después al inicio de la línea que sigue debajo de la línea recién impresa.

PROGRAM:

Palabra clave con la que empieza un programa. Después de ella va el nombre del programa y al final del programa se tiene que acabar con **End Program**.

PUTPIXEL:

Método invocado en la forma **PutPixel (X como DECIMAL, Y como DECIMAL)**. Pone un píxel en pantalla a las coordenadas (X, Y), en el color de la pluma actual.

R

RADIANSTODEGREES:

Función matemática invocada como ***RadiansToDegrees(x As Decimal) As Decimal***, siendo ***x*** un valor en radianes y devuelve el ángulo ***x*** en grados

RANDOM:

Función que devuelve un número entero al azar entre el MIN y MAX que se dan al invocar la función: Random (MIN valor entero, MAX valor entero)

RECTANGLE:

Método invocado en la forma **Rectangle (Anchura como ENTERO, Altura como INTERO, relleno como BOOLEAN)**. Dibuja un rectángulo en el color de la pluma actual, empezando a la posición de la pluma actual y longitudes en píxeles. Si en relleno ponemos TRUE el rectángulo estará lleno y si ponemos FALSE el rectángulo no se llenará.

RETURN:

Palabra clave utilizada siempre con FUNCTION y que devuelve como valor de la función el resultado de los cálculos realizados, provocando la finalización inmediata de la función. RETURN se puede insertar en cualquier punto de la parte ejecutable de FUNCTION y es posible utilizar más de una sentencia RETURN. También se puede utilizar con METHOD y sirve para terminar la ejecución del subprograma en ese momento y volver al punto siguiente a donde se invocó.

ROTATESPRITEBY:

Método que gira un sprite una cantidad especificada. Se invoca ***RotateSpriteBy(SpriteName As String, AmountOfChange As Decimal)***, siendo *SpriteName* el nombre del sprite y *AmountOfChange* la cantidad de grados que girará el sprite desde su orientación actual.

ROUND:

Función matemática predefinida disponible en KPL. Se invoca de la siguiente forma: **Function Round (d AS Decimal) AS Decimal**, y devuelve el entero que resulta del redondeo de d.

S

SETCONSOLEBACKGROUNDCOLOR:

SetConsoleBackgroundColor(Color As Integer) fija el color de fondo para la consola. El color puede especificarse a través del número de color o de su nombre.

SETCONSOLEBULLETMODE:

SetConsoleBulletMode (IsBulletModeOn As Boolean) fija o cancela el modo de viñetas de la consola. Si IsBulletModeOn es true entonces el modo de viñetas se enciende.

SETCONSOLEFONT:

El método ***SetConsoleFont(Fontname As String, FontSize As Decimal)*** fija el tipo de letra, siendo *Fontname* el nombre del tipo de letra y *FontSize* el tamaño de la letra.

SETCONSOLEFONTCOLOR:

El método ***SetConsoleFontColor(Color As Integer)*** fija el color de la letra.

SETCONSOLEFONTSIZE:

El método ***SetConsoleFontSize(FontSize As Decimal)*** fija el tamaño de la letra, siendo *FontSize* el tamaño que se utilizará para la letra.

SETCONSOLEFONTSTYLE:

El método ***SetConsoleFontStyle(Bold As Boolean, Italic As Boolean, Underline As Boolean)*** fija el estilo de letra de la consola, de manera que si *Bold* es true, entonces el texto siguiente se imprimirá en negrita, si *Italic* es true, entonces el texto siguiente se imprimirá en itálica y si *Underline* es true, entonces el texto siguiente se imprimirá subrayado.

SETCONSOLEINDENT:

El método ***SetConsoleIndent(Indent As Integer)*** activa o desactiva el modo de sangría, siendo *Indent* la distancia de sangría.

SETCONSOLETEXTALIGNMENT:

Método que configura el modo de alineamiento del texto de la ventana de consola. Se invoca de la forma siguiente: SetConsoleTextAlignment ("X"), siendo center, left y right los valores válidos para X.

SHOWCONSOLE:

Método que muestra la consola. Se invoca con ***ShowConsole ()***.

SHOWSPRITE:

Método que muestra el sprite definido. Se invoca ***ShowSprite (SpriteName As String)***, siendo *SpriteName* el nombre del sprite.

SIGN:

Función matemática invocada como ***Sign(Valor As decimal) As Integer***, siendo ***Valor*** un valor decimal y que devuelve el signo de ***Valor***, es decir, si Valor es menor que cero devuelve -1, si Valor es mayor que cero devuelve 1 y si Valor es cero, entonces devuelve cero.

SIN:

Función matemática predefinida disponible en KPL. Sin(x) devuelve el valor seno de x (x en radianes).

SPRITE:

Espectro, objeto.

SQRT:

Función matemática predefinida disponible en KPL. Sqrt(x) devuelve el valor de la raíz cuadrada de x, es decir $\sqrt{x}$.

STOP:

Método invocado por Stop(), que provoca la parada del programa y el cierre de la ventana donde se está ejecutando.

STRING:

Tipo de dato simple que designa una unidad de texto representada por una sucesión de caracteres, siempre entre comillas.

STRUCTURE:

Tipo de dato complejo que se define en la forma **STRUCTURE <Identificador>**. Es semejante a los tipos registros en otros lenguajes de programación. Se accede a los campos en una variable de STRUCTURE usando el nombre de la variable, seguido por un punto (".") y después el nombre del campo.

T

TAN:

Función matemática predefinida disponible en KPL. Tan(x) devuelve el valor tangente de x (x en radianes).

TANH:

Función matemática predefinida disponible en KPL. TanH(x) devuelve el valor de la tangente hiperbólica en la abscisa x, es decir $\frac{e^x - e^{-x}}{e^x + e^{-x}}$.

TICKCOUNT:

Función matemática predefinida disponible en KPL. Se invoca con **TickCount ( ) AS DECIMAL** y devuelve el the number of seconds since KPL started running – useful as a timer.

V

VAR:

Se utiliza para definir variables de la forma siguiente: Var <Identificador> As <Tipo dato>.

VARIABLE:

En programación una variable representa un valor almacenado que se puede modificar en cualquier momento. No se puede confundir con el concepto matemático de variable algebraica. Se define mediante **DEFINE <Identificador> AS <Tipo de Dato> [= <Expression>]** , siendo ***Identificador*** el nombre de la variable.

W

WHILE:

Sentencia utilizada para construir estructuras recursivas. Su estructura viene dada por:

```
WHILE <Expresión Boolean>

Acción

END WHILE
```

Mientras la “Expresión Boolean” sea cierta, se ejecuta “Acción” de forma repetitiva. Cuando el resultado es falso finaliza la ejecución de la sentencia.