DIDAC-PROG

CARTESIA

MANUAL DEL USUARIO DE LA APLICACIÓN

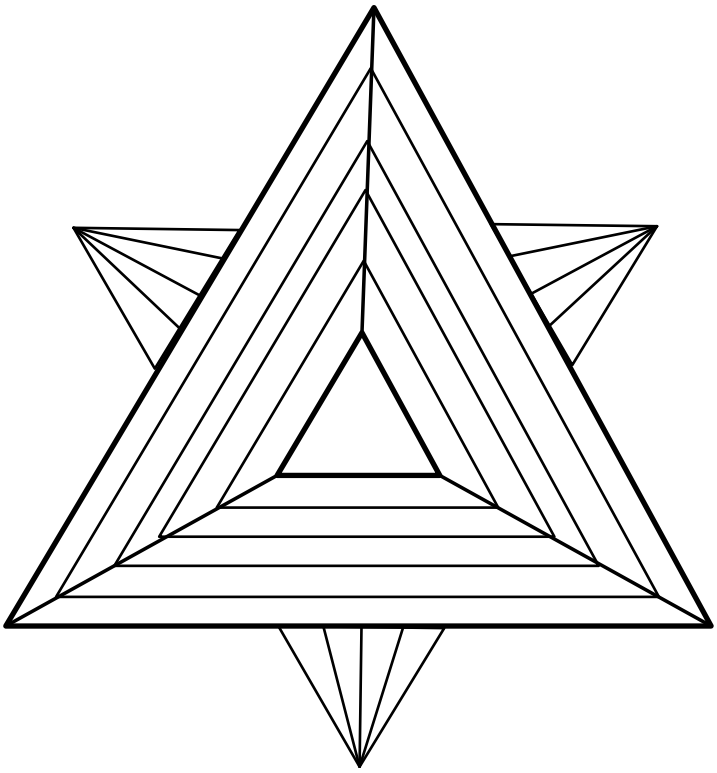
Aprender a programar con Didac-Prog Cartesia.

Para niños y no tan niños.

Escrito por Mario Rodríguez Rancel

Versión 2020 update 001

¿Qué es esto?



Averígualo

Sobre la educación, sólo puedo decir
que es el tema más importante en el que
nosotros, como pueblo, debemos involucrarnos.

Abraham Lincoln (1808-1865)

ÍNDICE

1. INTRODUCCIÓN

Didac-Prog Cartesia es una aplicación web **educativa y gratuita** concebida para el aprendizaje de la programación informática y/o matemáticas a niños a partir de 10 años, aunque nada impide que pueda ser utilizada a edades más tempranas (por ejemplo a partir de 7-8 años) o más tardías, e incluso adultos.
Este manual está preparado para que lo pueda leer cualquier persona, niño o adulto, y permite aprender a usar la aplicación ya que está planteado como un tutorial que recorre todas las posibilidades de Didac-Prog Cartesia con explicaciones y ejemplos de todas ellas. No obstante, los niños pueden requerir adaptaciones específicas del manual o algunas explicaciones adicionales en función de su edad. En la parte final del manual hay un anexo específicamente dirigido a profesores (dentro de profesores englobaremos a formadores, padres o tutores, etc. Es decir, cualquier persona que use la aplicación para enseñar a otra).
Didac-Prog Cartesia está preparado para la **enseñanza bilingüe Español-Inglés** y puede configurarse a deseo del usuario para trabajar totalmente en español o totalmente en inglés.
Empezar a utilizar Didac-Prog Cartesia es tan simple como descargar el archivo (denominado cartesia2020.zip o similar) con la aplicación desde el enlace correspondiente en aprenderaprogramar.com, descomprimirlo en nuestro ordenador o tablet y hacer doble click sobre el archivo index.html. El espacio que ocupa es mínimo (<5 Mb). A partir de ese momento podrás utilizar la aplicación cuando lo desees sin necesidad de tener conexión a internet. Si tienes problemas para abrir la aplicación consulta el manual de instalación.
Cuando se publiquen actualizaciones de la aplicación, podrás descargarlas desde esta misma web. Si lo prefieres puedes utilizar la aplicación on-line desde aprenderaprogramar.com, sin descargarla.

2. ¿CÓMO OBTENER AYUDA O CONTACTAR CON LOS CREADORES?

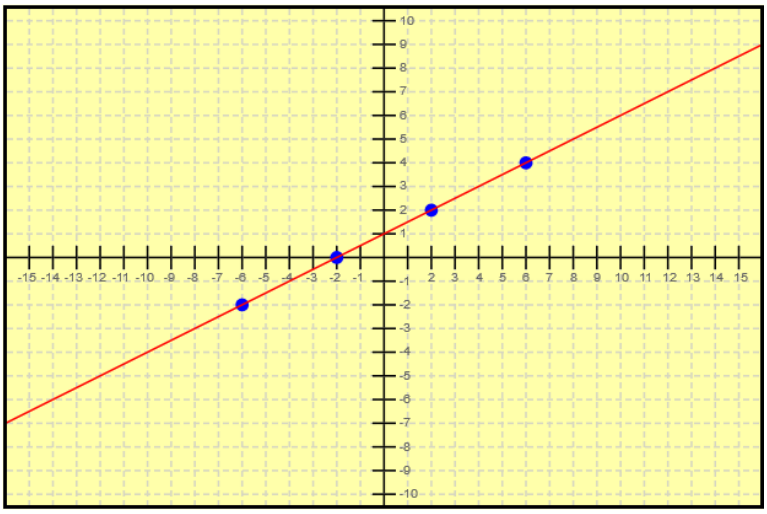
La aplicación, recursos asociados (p.ej. manuales), las dudas frecuentes (FAQ) y los proyectos y aportaciones de la comunidad que utiliza Didac-Prog Cartesia podrás encontrarlos en el apartado dedicado a la aplicación en la web aprenderaprogramar.com, donde podrás subir tus proyectos para compartirlos y consultar proyectos creados por otros usuarios.
En los foros de la web existen preguntas y respuestas sobre la aplicación. Si con los recursos indicados no logras resolver un problema, puedes escribir tu consulta en los foros, desde donde en la medida de lo posible intentan responder usuarios expertos o no tan expertos, e incluso los propios creadores de la aplicación.
Si quieres contactar con los creadores de Didac-Prog Cartesia para hacer sugerencias o comentarios, relatar tu experiencia, hacer una aportación en forma de artículo divulgativo o de investigación para que sea publicada y quede disponible para la comunidad, o para cualquier otra cosa, puedes hacerlo escribiendo a contacto@aprenderaprogramar.com

3. ¿PARA QUÉ SIRVE DIDAC-PROG CARTESIA?

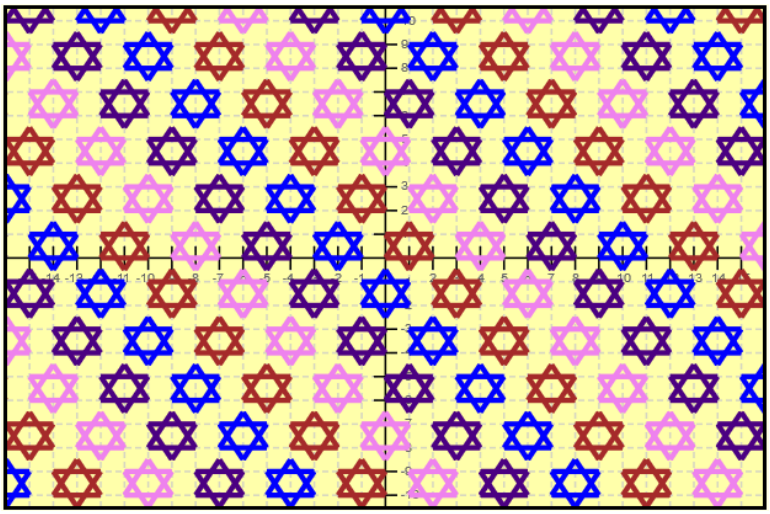
Con Didac-Prog Cartesia vamos a aprender a definir instrucciones al ordenador para que éste las ejecute. Esto es lo que llamamos *programación de computadores*, *programación informática* o simplemente *programar*. A veces también se dice que es “definir algoritmos” donde algoritmo se refiere a una **secuencia de instrucciones** o pasos a dar. La programación puede tener fines muy diversos, por ejemplo permitir hacer una compra on-line, almacenar y consultar datos, editar fotografías, definir cómo tiene que actuar un robot y casi cualquier cosa que se nos ocurra. Los *smartphone* y los ordenadores funcionan gracias a que disponen de muchas instrucciones en su memoria que definen lo que tienen que hacer. Hoy en día usan programación no sólo los profesionales de la informática, sino también ingenieros, científicos, economistas, etc.

Con Didac-Prog Cartesia podemos darle instrucciones al ordenador para que cree dibujos o representaciones gráficas de aquello que deseemos a partir de puntos y líneas. Además de puntos, segmentos, semirrectas o rectas pueden representarse figuras geométricas, funciones matemáticas, dibujos artísticos o cualquier cosa que podamos imaginar.

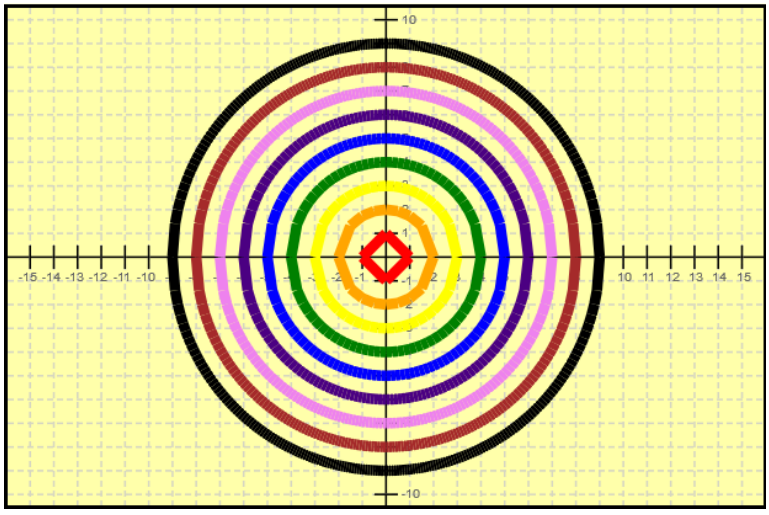
Como Didac-Prog Cartesia trabaja la programación apoyándose en las matemáticas, también puede ser utilizada para **aprender matemáticas**. Veamos algunos ejemplos de lo que puede hacer Cartesia.



Ejemplo 1 de dibujo creado con Didac-Prog Cartesia. En este caso se representan varios puntos de una recta en color azul y se traza la recta en color rojo. Cartesia no tiene una instrucción específica para trazar rectas (líneas infinitas), pero si cogemos dos puntos muy lejanos y los unimos la apariencia es haber trazado una recta.



Ejemplo 2 de dibujo creado con Didac-Prog Cartesia. En este caso se representan estrellas de David formadas por dos triángulos cada una y se trazan repetidamente en toda la pantalla alternando distintos colores (patrón de formas y colores).



Ejemplo 3 de dibujo creado con Didac-Prog Cartesia. Aunque la aplicación no tiene ninguna instrucción para trazar curvas explícitamente, éstas pueden crearse uniendo puntos o uniendo segmentos de pequeño tamaño.

A diferencia de aplicaciones que permiten al usuario dibujar como Paint, donde podemos elegir herramientas como un pincel y dibujar, Didac-Prog Cartesia no permite dibujar directamente. Lo que permite es definir una serie de instrucciones escritas (el ***código*** o ***programa***) sobre lo que debe hacer el ordenador, para que éste realice el dibujo.

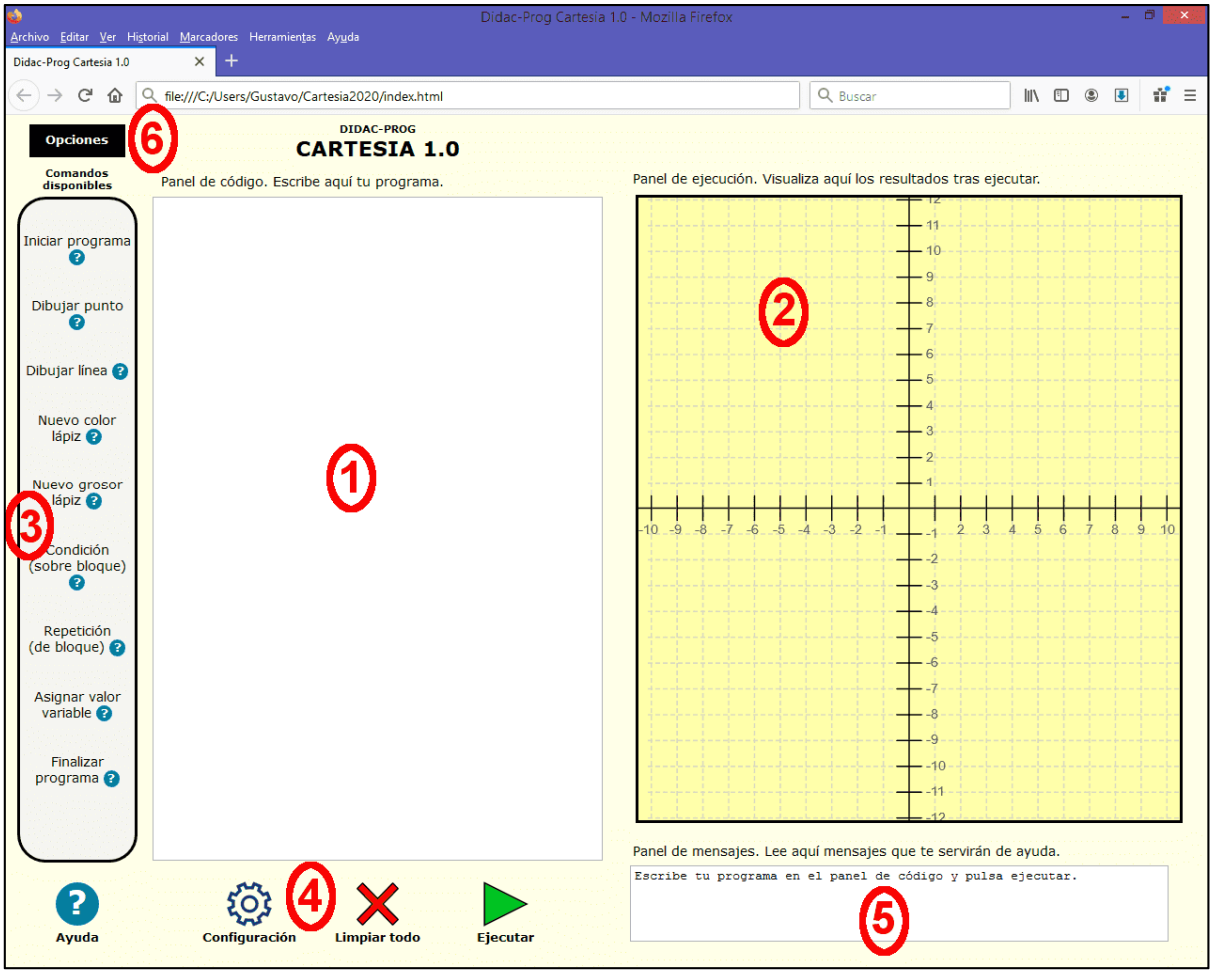
Didac-Prog Cartesia es un entorno educativo pero puede servir como paso previo al uso de entornos de programación más avanzados o profesionales.

Una vez definidas las instrucciones (el código), podemos decirle al ordenador que lo ejecute y ver el resultado de dicha ejecución. Como la aplicación sirve para dar instrucciones de lo que se debe dibujar muchas veces hablaremos del **lápiz de dibujo** como si el ordenador fuera un robot dibujante que usa un lápiz de dibujo. Así, podremos definir dónde hay que dibujar, si se debe dibujar una línea gruesa o fina, o si se debe usar un color u otro. Si cambiamos el color en que se debe dibujar desde rojo hasta azul, diremos que hemos cambiado el color del lápiz de dibujo.

Si guardamos el código, podemos ejecutarlo posteriormente en cualquier momento que deseemos, obteniendo siempre el mismo resultado si no hacemos modificaciones.

4. EMPEZANDO CON DIDAC-PROG CARTESIA: MENÚ, PANELES Y EJECUTAR

Si has abierto la aplicación correctamente habrás encontrado una pantalla similar a la siguiente (sobre la pantalla hemos señalado con números las distintas partes que se pueden diferenciar y que enumeraremos a continuación):

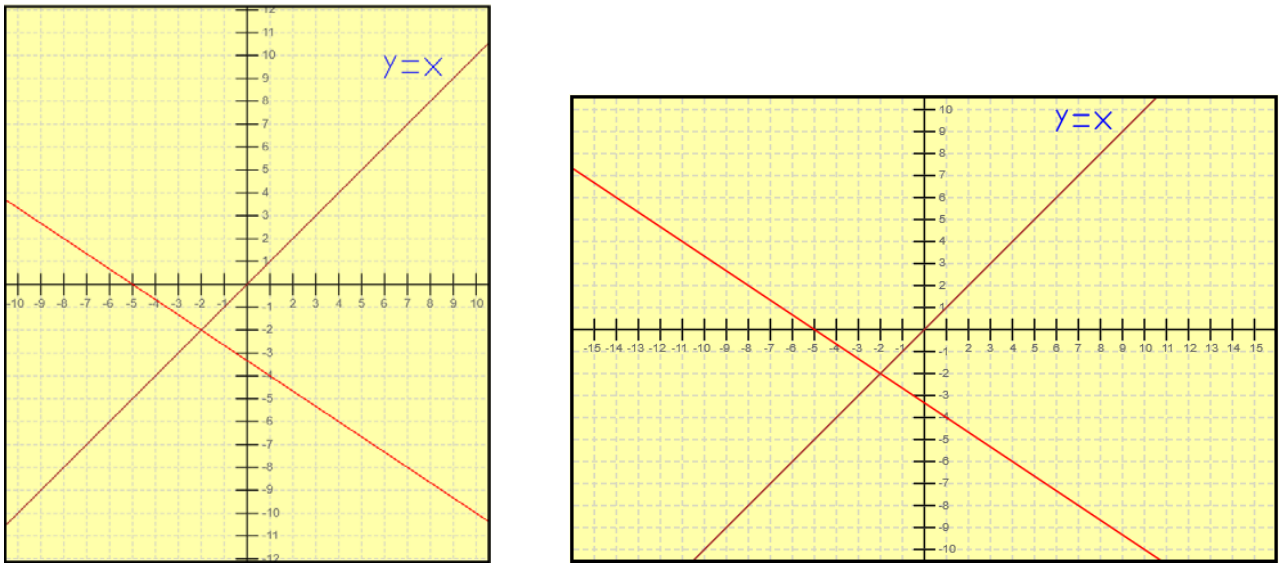


Nº	Nombre	Para qué sirve
1	**Panel de código**	Aquí escribiremos nuestro programa (las instrucciones que queramos que ejecute el ordenador). Cuando abrimos desde un archivo un proyecto creado por nosotros o por otra persona, será aquí donde veremos el código.
2	**Panel de dibujo**	Contiene unos ejes de coordenadas. Aquí veremos el resultado gráfico después de ejecutar el código tras pulsar el botón "Ejecutar". Ten en cuenta que es posible que no se vea nada por varios motivos: que nuestro código tenga errores, que el dibujo esté fuera del área visible u otros motivos.
3	**Panel de comandos**	Aquí tenemos un listado de los 9 comandos o instrucciones disponibles en el lenguaje de programación de Didac-Prog Cartesia. Haciendo doble click sobre cualquiera de ellos, se insertarán en el panel de código (a falta de completar con datos concretos).
4	**Panel de control**	En esta área tenemos varios botones: Configuración, Limpiar todo, Ejecutar y Ayuda.
5	**Panel de mensajes**	Cuando pulsemos el botón Ejecutar conviene prestar atención a las indicaciones que nos aparezcan en este panel, donde se nos informará de si la ejecución ha sido correcta, si hay errores, avisos, etc.
6	**Menú de opciones**	Al poner el ratón sobre este menú se despliegan distintas opciones: Abrir proyecto, Guardar proyecto, Abrir ejemplo y Deshacer limpiar.

Entre los botones del panel de control hay uno especialmente importante: el **botón Ejecutar**, ya que es el que nos permitirá ver los resultados que genera un código que hayamos escrito en el panel de código. A lo largo de este manual iremos viendo con ejemplos el uso de todos estos elementos.

Los elementos de Cartesia se adaptan al tamaño de la pantalla del dispositivo donde se visualicen (ordenador, tablet, etc.) siempre que la pantalla tenga un tamaño mínimo. Si la pantalla es demasiado pequeña la aplicación puede no mostrarse correctamente.

Una cuestión a tener en cuenta es que el panel de código de Cartesia no siempre es igual horizontalmente y verticalmente ya que es capaz de adaptarse al tamaño de pantalla. Esto puede dar lugar a que los dibujos creados se vean ligeramente distintos según el dispositivo que utilicemos. Para entender esto, mostramos un ejemplo de un dibujo creado con Cartesia visto en dos dispositivos distintos: el código es el mismo y el dibujo es el mismo, pero la visualización es ligeramente diferente.



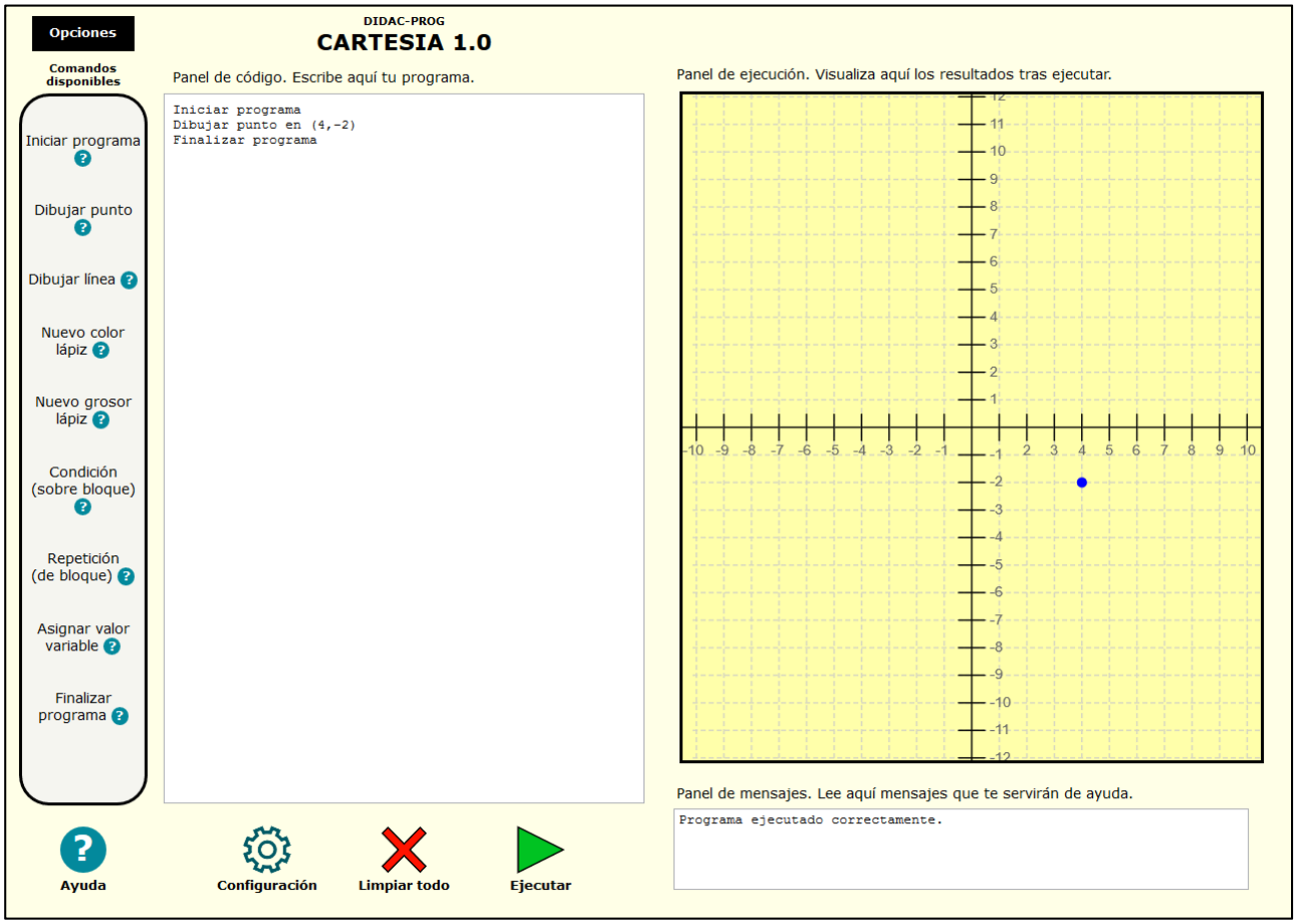
Visualización sobre distintos dispositivos del resultado de un mismo programa creado con Cartesia. Puede apreciarse que en un caso los ejes tienen aproximadamente el mismo número de divisiones, mientras que en el otro hay más divisiones horizontales que verticales. El área visible es distinta, de ahí que y=x aparezca en un caso separado del borde superior y en otro casi pegado a éste.

5. COMANDOS INICIAR PROGRAMA, DIBUJAR PUNTO Y FINALIZAR PROGRAMA

Vamos a crear nuestro primer programa con Cartesia. Para ello escribe en el **panel de código** lo siguiente:

```
Iniciar programa
Dibujar punto en (4,-2)
Finalizar programa
```

Fíjate cómo el contenido del panel de mensajes inicialmente es “Escribe tu programa en el panel de código y pulsa ejecutar.” Pulsa ahora el botón Ejecutar. El resultado sobre el panel de ejecución (al que también llamamos panel de dibujo) debe ser similar a lo siguiente:



Como verás ha aparecido un punto sobre el panel de ejecución y en el panel de mensajes ha aparecido el mensaje “Programa ejecutado correctamente”.

Didac-Prog Cartesia trabaja con un sistema de coordenadas cartesianas con el que posiblemente estés familiarizado. Si no lo estás, vamos a explicar brevemente en qué consiste.

Sobre el panel de dibujo vemos un eje horizontal, también denominado **eje de abscisas** o simplemente **eje x** dividido en porciones que nos permiten situarnos en horizontal. Así si un punto tiene coordenada x=4 significa que hemos de movernos desde el centro cuatro unidades hacia la derecha. Si el valor fuera negativo, por ejemplo x = -4, tendríamos que movernos en sentido opuesto: hacia la izquierda.

Por otra parte tenemos un eje vertical, también denominado **eje de ordenadas** o simplemente **eje y** también dividido y que nos permite situarnos en vertical. Así si un punto tiene coordenada y = -2 significa que hemos de movernos dos unidades hacia abajo. Si el valor fuera positivo, por ejemplo y = 2, tendríamos que movernos en sentido opuesto: hacia arriba.

El plano sobre el que dibujamos se llama plano cartesiano o **sistema de coordenadas cartesiano** y nos permite situar con precisión la ubicación de cualquier elemento que dibujemos. Estos sistemas de coordenadas son muy importantes en muchos aspectos de la vida diaria: en ingeniería (por ejemplo planos de obras), en arquitectura (por ejemplo planos de edificios), en física (por ejemplo para representar desplazamientos), en aplicaciones informáticas (por ejemplo la pantalla de tu ordenador o tablet), para representar en forma de mapa una ciudad (por ejemplo Google Maps), etc.

Un punto queda definido por 2 valores a los que llamamos **coordenada x** o coordenada horizontal y **coordenada y** o coordenada vertical. Un punto cualquiera lo representamos habitualmente como **(x, y)**. En el ejemplo que estamos viendo x = 4 e y = -2 , y el punto es (4, -2).

Para situar el punto sobre el plano cartesiano nos situamos en el centro, nos desplazamos 4 unidades hacia la derecha y luego 2 unidades hacia abajo, y en ese lugar se dibuja el punto.

En este ejemplo hemos visto 3 comandos de Didac-Prog Cartesia:

- **Iniciar programa:** debe ser, obligatoriamente, el primer comando que aparezca en cualquier programa que escribas en Didac-Prog Cartesia. Este comando sirve para indicarle al ordenador que comience la ejecución de las instrucciones. No puede aparecer más de una vez en un programa.
- **Dibujar punto en (?,?):** es el comando para dibujar un punto. Debes sustituir los interrogantes por los valores de las coordenadas del punto, por ejemplo, Dibujar punto en (4, -2)
- **Finalizar programa:** debe ser, obligatoriamente, el último comando que aparezca en cualquier programa que escribas en Didac-Prog Cartesia. Este comando sirve para indicarle al ordenador que finalice la ejecución de las instrucciones. No puede aparecer más de una vez en un programa.

Cada comando de Cartesia debe escribirse obligatoriamente en una línea y un programa puede tener tantas líneas como quieras.

6. PANEL DE MENSAJES. MENSAJES DE ERROR Y MENSAJES DE AVISO.

Errores en programas

El **panel de mensajes** es muy útil pues nos informa de si hemos cometido errores o nos avisa de posibles problemas con nuestro programa. Escribe en el panel de código lo siguiente (fíjate que hemos escrito *Inicar* en lugar de *Iniciar* y que no hemos escrito las coordenadas del punto entre paréntesis):

```
Inicar programa
Dibujar punto en 4,-2
Finalizar programa
```

Pulsa el botón Ejecutar. Sobre el panel de dibujo no aparece nada. ¿Por qué? Porque el programa tiene **errores**. En este caso sabemos que los errores son haber escrito *Inicar* en lugar de *Iniciar* y haber escrito las coordenadas del punto sin paréntesis. Pero imagina que hemos cometido estos errores sin darnos cuenta y que no sabemos por qué el programa no funciona.

Para saber por qué no se muestra nada en el panel de dibujo, en general será útil consultar el panel de mensajes para ver si Cartesia nos informa de lo que está ocurriendo.

Si tras pulsar Ejecutar consultamos el panel de mensajes veremos un mensaje similar a lo siguiente:

```
ERROR: La primera línea válida del programa tiene que comenzar con <<Iniciar programa>>. Por favor revise el código.
```

Corrige ahora el primer error: escribe Iniciar correctamente y vuelve a pulsar el botón Ejecutar. Como aún tenemos un error debido a que no hemos añadido los paréntesis, nos aparecerá otro mensaje similar a lo siguiente:

```
ERROR: la instrucción dibujar punto requiere especificar un punto válido en formato (a, b). Por favor revise el código. Línea no válida.
"dibujar punto en 4,-2"
```

Cartesia nos está avisando de que hay un error por lo que no puede ejecutar el programa, y además nos está indicando dónde está el problema.

Escribe ahora los paréntesis y vuelve a pulsar el botón Ejecutar. Comprobarás que ahora aparece el punto dibujado sobre el panel de dibujo, y que en el panel de mensajes aparece el mensaje “Programa ejecutado correctamente”.

Es por tanto importante consultar el panel de mensajes para saber si nuestro código contiene errores y poder corregirlos.

Avisos de Cartesia

Cartesia no sólo es capaz de detectar e indicarnos si tenemos errores en nuestro programa, sino que también nos puede dar **avisos** útiles. Escribe este programa en el panel de código y pulsa ejecutar:

```
Iniciar programa
Dibujar punto en (400,-200)
Finalizar programa
```

En el panel de dibujo no aparece nada. ¿Por qué? Si consultamos el panel de mensajes comprobaremos que aparece algo similar a lo siguiente:

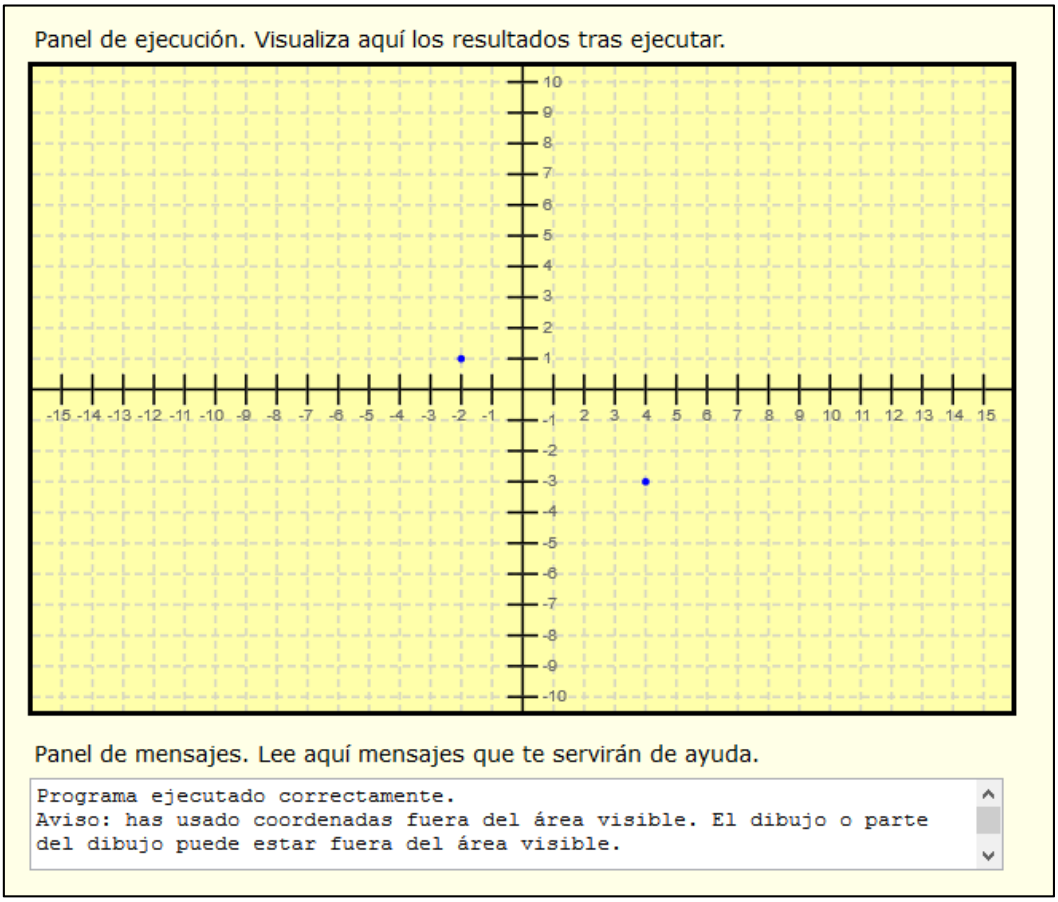
```
Programa ejecutado correctamente.
Aviso: has usado coordenadas fuera del área visible. El dibujo o parte del dibujo puede estar fuera del área visible.
```

Cartesia nos indica que el programa se ha ejecutado correctamente, es decir, no hemos incumplido las normas y forma de escritura requeridas, y el ordenador ha ejecutado el programa. Sin embargo, nos muestra un aviso: hemos usado coordenadas fuera del área visible y el dibujo puede estar **fuera del área visible** en el panel de dibujo. Así es: si empezamos a contar hacia la derecha para ver dónde se sitúa el valor de x=400 comprobaremos que ese valor no es visible. Es decir, puede haberse dibujado, pero no lo vemos porque las coordenadas no están dentro de las coordenadas que muestra el panel de dibujo. En algunos casos puede haber una parte del dibujo visible y otra no visible.

Escribe este programa en el panel de código y pulsa ejecutar:

```
Iniciar programa
Dibujar punto en (4,-3)
Dibujar punto en (400,-200)
Dibujar punto en (-2,1)
Finalizar programa
```

El resultado será algo similar a esto:



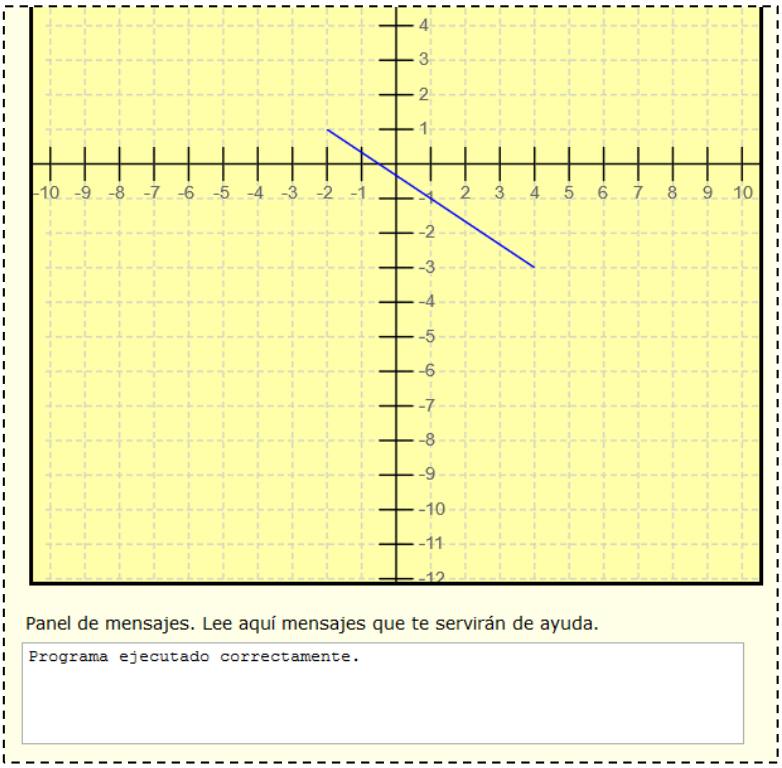
¿Por qué sólo se muestran 2 puntos si hemos indicado que deben dibujarse 3? El panel de mensajes te dará la respuesta: hay coordenadas fuera del área visible y el dibujo o parte de él pueden no verse. En este caso de los 3 puntos que hemos ordenado dibujar, 2 se ven en el panel de dibujo y 1 queda fuera del área visible. Ten en cuenta que un aviso no es un error, y en ocasiones puede que ni siquiera sea un problema. De hecho, podemos crear programas en Cartesia cuya ejecución vaya acompañada de avisos y esto no es nada malo.

7. COMANDO DIBUJAR LÍNEA. USO DE DECIMALES Y FRACCIONES.

El comando **Dibujar línea desde (?, ?) hasta (?, ?)** nos permite dibujar líneas entre dos puntos que hemos de especificar. Escribe en el panel de código lo siguiente:

```
Iniciar programa
Dibujar línea desde (-2,1) hasta (4,-3)
Finalizar programa
```

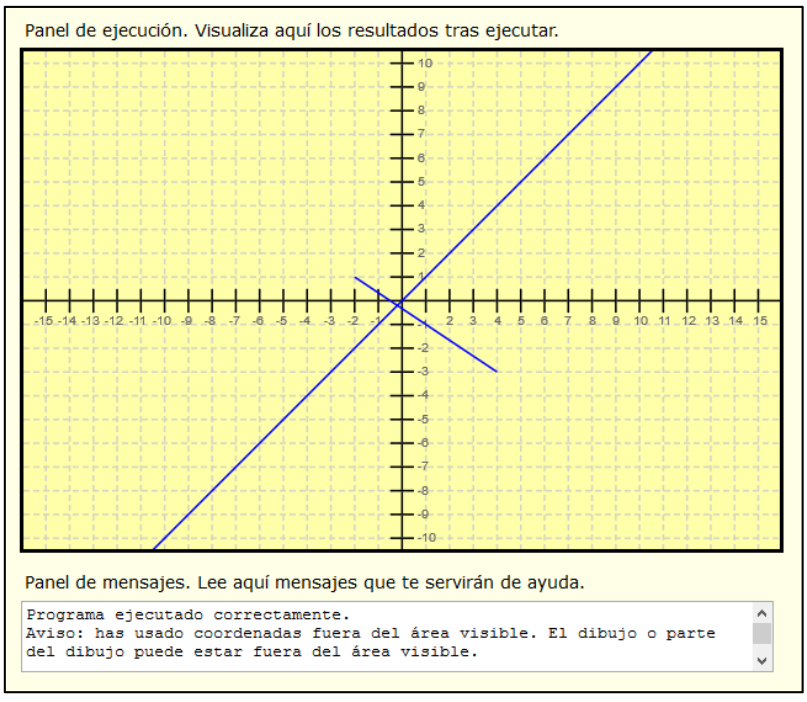
El resultado será similar a lo siguiente:



Escribe ahora este otro código:

```
Iniciar programa
Dibujar línea desde (-2,1) hasta (4,-3)
Dibujar línea desde (-100,-100) hasta (100,100)
Finalizar programa
```

Cuyo resultado será algo parecido a esto:



Fíjate cómo una de las líneas tiene su origen en un punto fuera del panel de dibujo, pasa por el panel, y continúa hasta terminar también fuera del panel de dibujo. Podríamos considerar que este tipo de líneas representan **rectas** o "líneas infinitas". Una línea que comienza y termina dentro del panel de dibujo podríamos considerarlo como un **segmento** o porción de recta. Finalmente, si una línea comenzara en un punto dentro del panel de dibujo y terminara fuera de él, podríamos considerarlo una **semirrecta**.

Didac-Prog Cartesia permite el uso de decimales y fracciones. Por ejemplo puedes escribir: Dibujar línea desde (-3.75, 2) hasta (4.25, -6)

Pero también se admite y se obtiene el mismo resultado si se escribe: Dibujar línea desde (-15/4, 2) hasta (8.50/2, -6) ya que -15/4 = -3.75 y 8.50/2 = 4.25

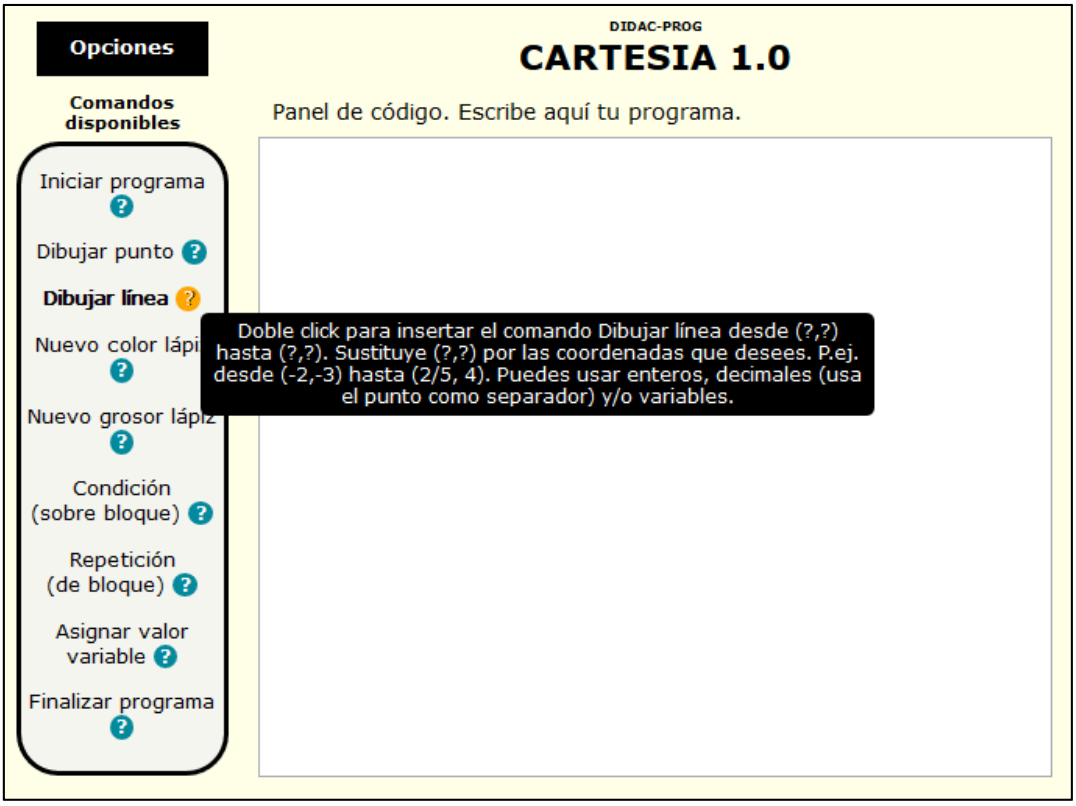
Haz pruebas con distintos programas que dibujen líneas e interpreta los resultados.

8. LIMPIAR TODO E INSERTAR COMANDOS CON EL PANEL DE COMANDOS

Quizás escribir los programas te resulte cansado o te equivoques con frecuencia escribiendo. Cartesia tiene un **panel de comandos** en el lado izquierdo de la pantalla que te facilitará insertar comandos y escribir lo mínimo posible si así lo prefieres.

Vamos a empezar borrando cualquier dibujo que exista y cualquier código. Para ello escribe un código, ejecútalo y visualiza el resultado, y luego pulsa el botón **Limpiar todo**. Comprobarás que el código, dibujo y mensajes en el panel de mensajes desaparecen. El panel de código estará vacío, el panel de dibujo no contendrá ningún dibujo, y en el panel de mensajes habrá un mensaje similar a "Escribe tu programa en el panel de código y pulsa ejecutar", es decir, se ha limpiado todo y la aplicación está esperando a que introduzcamos un nuevo código.

Fíjate ahora en el lateral izquierdo en el panel de comandos donde aparecen los 9 comandos permitidos por Didac-Prog Cartesia, cada uno seguido de un símbolo ? de ayuda. Al colocar el puntero del ratón sobre el símbolo ? de un comando se desplegará un texto de ayuda sobre cómo usar ese comando. Por ejemplo si nos colocamos sobre el símbolo ? junto a *Dibujar línea* veremos algo similar a esto:



El texto de ayuda nos indicará algo como "Haz doble click para insertar el comando Dibujar línea desde (?,?) hasta (?,?). Sustituye (?,?) por las coordenadas que desees. P.ej. desde (-2,-3) hasta (2/5, 4). Puedes usar enteros, decimales (usa el punto como separador) y/o variables."

Vamos ahora a crear un programa que ya hemos escrito anteriormente, pero ahora usando la inserción de comandos desde el panel de comandos. Nuestro programa va a constar de 4 instrucciones: Iniciar, Dibujar línea, Dibujar línea y Finalizar. Haremos pues doble click en el panel de comandos sobre Iniciar programa, Dibujar línea dos veces, y Finalizar programa. Sobre el panel de código debe habernos aparecido esto:

```
Iniciar programa
Dibujar línea desde (?,?) hasta (?,?)
Dibujar línea desde (?,?) hasta (?,?)
Finalizar programa
```

Ahora sustituimos los símbolos de interrogación por los valores que nos interesen, por ejemplo escribe:

```
Iniciar programa
Dibujar línea desde (-2,1) hasta (4,-3)
Dibujar línea desde (-1000,-1000) hasta (1000,1000)
Finalizar programa
```

Pulsa el botón Ejecutar y comprobarás que el resultado es el mismo que obtuvimos antes cuando escribimos el código.

De aquí en adelante, recuerda que si insertas un comando con doble click desde el panel de comandos, debes sustituir los sitios donde aparezcan interrogantes por aquellos valores, expresiones o comandos que sean necesarios.

Para crear tu código en Didac-Prog Cartesia puedes:

- **Escribir directamente** desde el panel de código.
- **Copiar y pegar** (como se hace habitualmente en otros programas) algo que hayas escrito tú, que haya escrito otra persona, obtenido desde una página web, etc.
- **Usar el panel de comandos** para insertar un comando haciendo doble click sobre él.
- **Abrir un programa** que hayas guardado previamente para ejecutarlo o para modificarlo (más adelante explicaremos cómo).

9. PANTALLA DE CONFIGURACIÓN. IDIOMA, COLOR Y GROSOR DEL LÁPIZ DE DIBUJO.

En los programas que hemos ejecutado hasta ahora los puntos y líneas se dibujan de un determinado color y de un determinado grosor. Pero esto podemos cambiarlo de dos maneras:

- **Mediante código** (se explicará más adelante).
- **Modificando la configuración de la aplicación.**

Para cambiar la configuración de la aplicación pulsa el botón "Configuración". Accederás a una pantalla similar a la que mostramos a continuación, con opciones para **Idioma**, **Color del lápiz de dibujo** y **Grosor del lápiz de dibujo**:

Idioma

En esta pantalla tienes opción a modificar el idioma. Actualmente Didac-Prog Cartesia puede configurarse a dos idiomas: **español e inglés.** Si modificas el idioma y eliges inglés, debes tener cuidado, porque todos los textos cambiarán a inglés y también los programas tendrás que escribirlos en inglés. Así, en lugar de *Iniciar programa* tendrías que escribir *Start program*, en lugar de *Dibujar punto en (?, ?)* el comando sería *Draw point at (?, ?)* y así sucesivamente. Es fácil averiguar el comando en inglés. Una vez configuras la aplicación en inglés, haz doble click sobre cada comando en el panel de comandos para ver cómo se escribe en inglés.

Color del lápiz de dibujo

Cartesia permite dibujar con 9 colores. Cada color se designa por un número, así los colores que podemos elegir son 1, 2, 3, 4, 5, 6, 7, 8, ó 9. La equivalencia en color de cada número es la siguiente:

Color en Cartesia	Color que se verá en el panel de dibujo
1	Rojo
2	Naranja
3	Amarillo
4	Verde
5	Azul
6	Añil
7	Rosa
8	Marrón
9	Negro

Si en la pantalla de configuración elegimos un color y pulsamos el botón **Guardar y volver**, todos los elementos (puntos y líneas) que aparezcan en el dibujo a partir de ese momento lo harán con este color elegido, excepto si indicamos a través de código el uso de otro color.

Grosor del lápiz de dibujo

Cartesia permite dibujar con 3 grosores. Cada grosor se designa por un número, así los grosores que podemos elegir son 1, 2, ó 3. La equivalencia en grosor de cada número es la siguiente:

Grosor en Cartesia	Grosor de puntos y líneas en el panel de dibujo
1	Fino
2	Medio
3	Grueso

Si en la pantalla de configuración elegimos un grosor y pulsamos el botón "Guardar y volver", todos los elementos (puntos y líneas) que aparezcan en el dibujo a partir de ese momento lo harán con este grosor elegido, excepto si indicamos a través de código el uso de otro grosor.

Guardar la configuración

Si después de cambiar alguna opción de configuración pulsamos "Guardar y volver", regresaremos a la pantalla principal y todos los dibujos que hagamos se regirán por la configuración elegida. Por ejemplo, prueba a cambiar el color al valor 1 (rojo) y el grosor al valor 3 (grueso) y realiza algún dibujo sobre el panel de dibujo y comprobarás que los dibujos aparecen en color rojo y con trazo grueso.

Cuidado: si en lugar de pulsar "Guardar y volver" se pulsa "Cancelar y volver" no se guardará ninguna opción elegida y se seguirá con la configuración que existiera previamente.

Recuperar la configuración inicial

Si pulsamos en el botón "Configuración inicial" (en inglés, "Initial configuration") se regresará a la configuración inicial de la aplicación.

10. COMANDOS NUEVO COLOR LÁPIZ Y NUEVO GROSOR LÁPIZ. TRAZAR TEXTO.

Hasta ahora no hemos mencionado la posibilidad de que aparezca un texto dentro del panel de dibujo. ¿Es posible? En la versión actual de Cartesia, no es posible insertar texto directamente dentro del panel de dibujo. No obstante, si se tiene paciencia se puede dibujar o crear pequeños programas donde podemos almacenar el código que genere texto en forma de dibujo (por ejemplo la x serían dos líneas que se cruzan) para utilizarlo en posteriores programas.

En los programas que hemos ejecutado hasta ahora los puntos y líneas se dibujan de un determinado color y de un determinado grosor. Ya hemos visto que color y grosor pueden cambiarse a través de las opciones de configuración de la aplicación. Pero esto tiene una limitación: supone que todos los puntos y líneas se dibujarán **de la misma manera.**

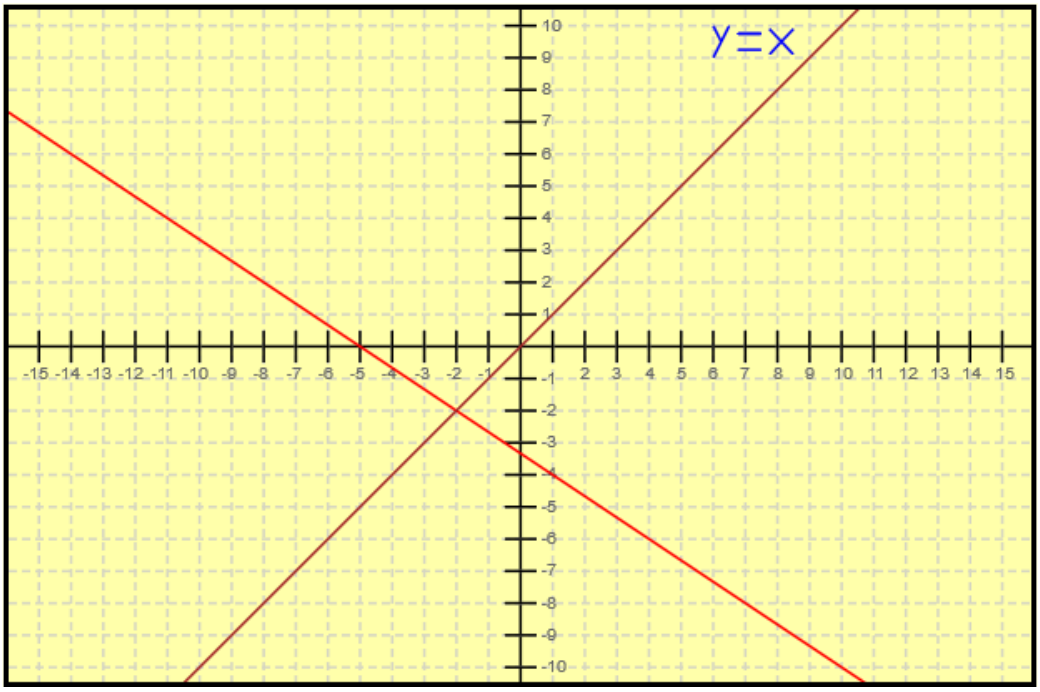
¿Cómo podemos dibujar líneas y puntos de distintos colores y grosores dentro de un dibujo? La solución es usar comandos específicos que introducidos en nuestro código nos permiten modificar el color y grosor con el que trazamos el dibujo. Tener en cuenta que si cambiamos el color (o grosor) en algún punto de nuestro código, todo lo que se dibuje después estará en ese color (o grosor), excepto si volvemos a introducir una instrucción que modifique lo establecido.

Comando nuevo color lápiz y trazar texto en el panel de dibujo

Vamos a crear un programa similar a los que hemos usado anteriormente para trazar con distintos colores.

El comando que debemos usar es **Nuevo color lápiz (?)** sustituyendo el ? por un número entero entre 1 y 9. Escribe este código en el panel de código y comprueba el resultado, que debe ser similar al que se muestra más abajo:

```
Iniciar programa
Nuevo color lápiz (1)
Dibujar línea desde (-20,10) hasta (40,-30)
Nuevo color lápiz (8)
Dibujar línea desde (-100,-100) hasta (100,100)
Nuevo color lápiz (5)
Dibujar línea desde (6,10) hasta (6.25,9.5)
Dibujar línea desde (6.50,10) hasta (6,9)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.25) hasta (7.5,9.25)
Dibujar línea desde (7.75,9.9) hasta (8.5,9.1)
Dibujar línea desde (8.5,9.9) hasta (7.75,9.1)
Finalizar programa
```

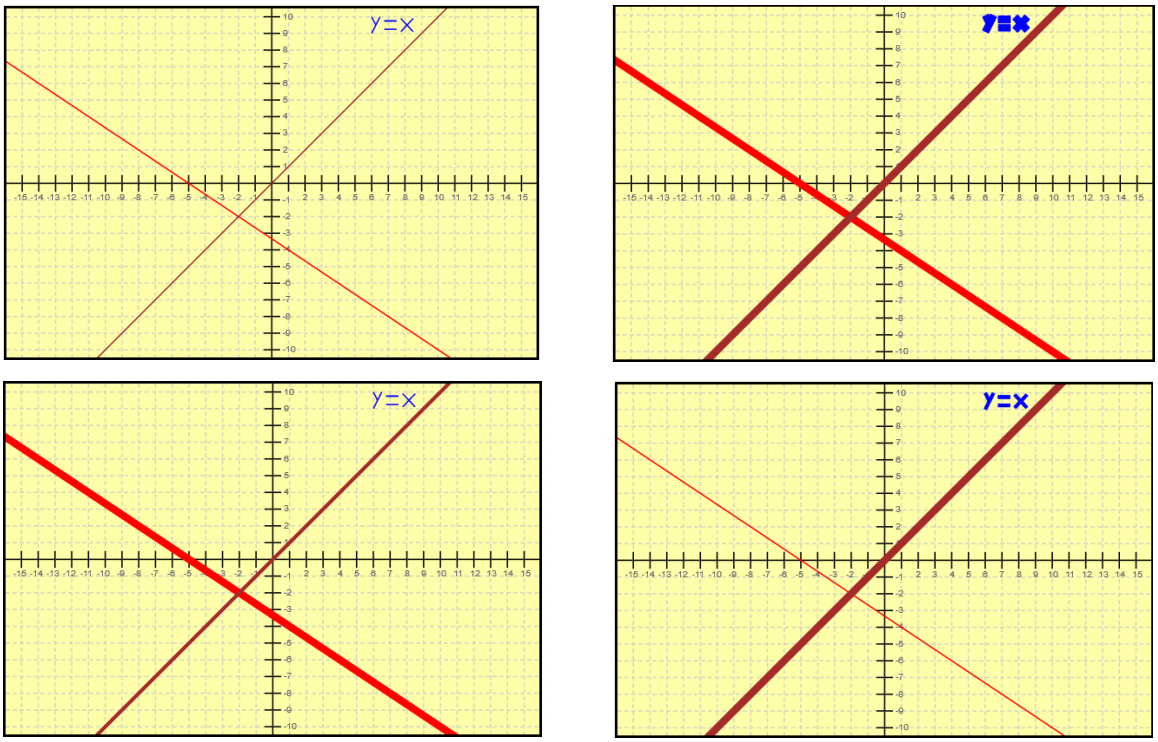


Al pulsar Ejecutar ocurre lo siguiente: se establece como nuevo color de lápiz de dibujo el color 1 (es decir, el color de trazado será el rojo). A continuación se dibuja una línea roja (porque hemos establecido como color del lápiz el rojo) que pasa por los puntos (-20,10) hasta (40,-30). Estos puntos seguramente estarán fuera del área visible, no obstante la línea que los une **pasa por el área visible**, y la podemos ver de color rojo. A continuación se establece como color del lápiz de dibujo el color 8 (es decir, el color de trazado será el marrón) y se traza de la misma forma otra línea que vemos en color marrón. Finalmente se establece como color del lápiz de dibujo el 5 (azul) y se trazan varias líneas que al verlas sobre la pantalla tienen apariencia de ser el texto y = x, que se ve en color azul. Haz pruebas dibujando puntos y rectas de distintos colores.

Comando nuevo grosor lápiz

El cambio de grosor se hace de forma muy similar a como hemos visto para el color. El comando que debemos usar es **Nuevo grosor lápiz (?)** sustituyendo el ? por 1, 2 ó 3. Establecer el grosor 1 equivale a lápiz fino, el grosor 2 a lápiz de grosor intermedio y el grosor 3 a lápiz grueso.

Puedes establecer un grosor al principio de tu programa para que siempre se dibuje con ese grosor, o bien ir cambiando los grosores. En las siguientes imágenes puedes ver cómo cambia la apariencia del trazado visto en el ejemplo anterior al cambiar los grosores:



Visualización de resultados cambiando grosores a los elementos de dibujo. Arriba izquierda: todo está trazado con grosor 1 (fino). Arriba derecha: todo está trazado con grosor 3 (grueso). Inferior izquierda: la línea roja tiene trazo grueso, la marrón medio y el texto y=x fino. En la imagen inferior derecha la línea marrón tiene trazo grueso, la línea roja fino y el texto y=x medio.

Avisos por ejecución sin visualización posible

Podría ocurrir que por error escribamos un programa donde no se dibuje nada. Por ejemplo, un programa que modifique el color o grosor del lápiz de dibujo. En este caso, al pulsar el botón Ejecutar no veremos nada en el panel de dibujo porque no hay nada que dibujar. Como siempre, el panel de mensajes nos será de gran ayuda para interpretar lo que está ocurriendo, ya que nos mostrará un aviso. Escribe el siguiente código y ejecútalo:

```
Iniciar programa
Nuevo color lápiz (8)
Nuevo grosor lápiz (2)
Finalizar programa
```

Verás que en el panel de dibujo no aparece nada dibujado, porque no hemos introducido ninguna instrucción que dibuje algo. En el panel de mensajes aparecerá un mensaje similar al siguiente:

```
Programa ejecutado correctamente. Aviso: tu código no contiene ningún comando que permita dibujar algo sobre el panel de ejecución, por tanto no podrás visualizar ningún resultado.
```

11. COMENTARIOS EN EL CÓDIGO DE CARTESIA

A medida que nuestros programas se hacen más extensos y complicados es posible que ya no recordemos para qué sirven determinadas líneas de nuestro código o incluso que no recordemos para qué habíamos creado ese programa. También es posible que si tenemos que modificar un programa guardado, no sepamos bien para qué sirve cada parte del código.

Para evitar estos problemas (y para muchas cosas más), Cartesia permite introducir **comentarios** en el código. Estos comentarios no se ejecutarán, pues no son instrucciones para ejecutar, sino simplemente un texto que nos ayuda a conocer y explicar el programa, reflejar la fecha de creación, autor, o aquello que deseemos.

Un comentario se inserta en Cartesia insertando dos guiones medios (--) y a continuación de ellos el texto libre que se quiera introducir. Como los comentarios no se ejecutan, un mismo código podría comentarse de distintas maneras y no por ello alterar el resultado obtenido. Para comprobar esto, vamos a comentar un código visto anteriormente de dos maneras distintas. Escribe ambos programas y comprueba cómo los resultados son los mismos (hemos señalado en negrita los comentarios para que resulte fácil identificarlos):

```
-- PROGRAMA QUE DIBUJA DOS RECTAS
Iniciar programa
Nuevo color lápiz (1) -- Color rojo
Dibujar línea desde (-20,10) hasta (40,-30)
Nuevo color lápiz (8) -- Color marrón
Dibujar línea desde (-100,-100) hasta (100,100)
Nuevo color lápiz (5) -- Color azul
Dibujar línea desde (6,10) hasta (6.25,9.5)
Dibujar línea desde (6.50,10) hasta (6,9)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.25) hasta (7.5,9.25)
Dibujar línea desde (7.75,9.9) hasta (8.5,9.1)
Dibujar línea desde (8.5,9.9) hasta (7.75,9.1)
Finalizar programa
-- Creado el 16/05/2048 por Manuel García Pérez para la asignatura de Tecnología de 2º curso de Educación Secundaria Obligatoria (ESO) en el IES Pitágoras de Samos.
```

```
-- PROGRAMA QUE DETERMINA EL PUNTO DE CORTE ENTRE DOS RECTAS
Iniciar programa
-- Establecemos como color de dibujo el color rojo y trazamos la recta roja
Nuevo color lápiz (1)
Dibujar línea desde (-20,10) hasta (40,-30)
-- Establecemos como color de dibujo el color marrón y trazamos la recta marrón
Nuevo color lápiz (8)
Dibujar línea desde (-100,-100) hasta (100,100)
-- Establecemos como color de dibujo el color azul y trazamos el texto
Nuevo color lápiz (5)
Dibujar línea desde (6,10) hasta (6.25,9.5)
Dibujar línea desde (6.50,10) hasta (6,9)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.75) hasta (7.5,9.75)
Dibujar línea desde (6.75,9.25) hasta (7.5,9.25)
Dibujar línea desde (7.75,9.9) hasta (8.5,9.1)
Dibujar línea desde (8.5,9.9) hasta (7.75,9.1)
Finalizar programa
-- Creado el 16/05/2049 por Manuel García Pérez para la asignatura de Matemáticas de 3er curso de Educación Secundaria Obligatoria (ESO) en el Instituto Thales de Mileto.
```

Ten en cuenta que los comentarios deben cumplir ciertas reglas:

a) Los dos guiones medios **tienen que estar juntos**. Si estuvieran separados, aparecerá un mensaje de error en el panel de mensajes.

b) Un comentario **puede ocupar varios renglones** del panel de código siempre que no pulsemos la tecla Enter, que genera una nueva línea. Si pulsáramos la tecla Enter, cada línea debería empezar con los dos guiones medios. Si no lo hacemos así, aparecerá un mensaje de error del tipo: “ERROR: por favor revise el código. Línea no válida”, junto al texto de la línea, por ejemplo "asignatura de Matemáticas de 3er curso de Educación Secundaria Obligatoria (ESO) en el instituto Thales de Mileto."

Como puedes comprobar, un comentario puede estar al comienzo de una línea ocupando la línea completa, o bien después de un comando, quedando como parte final de una línea. Lo que no está permitido es escribir un comentario y después en esa misma línea un comando, porque al estar el comando detrás de los dos guiones, no será ejecutado precisamente porque será considerado como comentario y no como instrucción.

12. VARIABLES EN CARTESIA. COMANDO ASIGNAR VALOR VARIABLE.

Introducción a las variables en programación

En matemáticas es frecuente usar variables como x, y, a, b, c y otras. Por ejemplo si nos dicen que x=3 y que a=4, y nos preguntan cuánto vale x+a diremos que 7 ya que 4+3=7. Pero seguidamente en el mismo problema no podríamos decir que a vale 9, porque anteriormente hemos dicho que a vale 4.

En programación se usan variables de forma similar a como se usan en matemáticas, aunque existen algunas diferencias. En Cartesia una variable la podemos ver como una memoria, o como una caja que contiene un papel donde está escrito un número. Por ejemplo si decimos que x vale 3, dentro de la caja "x" está el papel con el número 3. Si en la caja "a" tenemos un 4 y nos preguntan cuánto vale x+a diremos que 7 ya que 4+3=7.

Si ahora decimos que el nuevo valor de "a" es 1 significa que sacamos el papel donde estaba escrito el 4, lo tiramos, y dentro de la caja metemos el papel con el 1. Si nos preguntan cuánto vale x+a diremos que 5 ya que 4+1=5. Aquí como vemos "a" ha cambiado de valor. Primero valía 4 y luego ha pasado a valer 1. Esto se usa en programación (pero no en matemáticas, donde en un mismo problema, "a" sólo puede tener un valor).

Variables disponibles en Cartesia

En Cartesia se pueden usar siete variables: **x, y, a, b, c, d, e**

En cada una de estas siete variables se pueden almacenar valores numéricos, que pueden ser tanto enteros como decimales. Los valores numéricos pueden expresarse directamente, por ejemplo 3 y 0.20, o bien como fracciones, por ejemplo 6/2 que equivale a 3 ó 1/5, que equivale a 0.20. Todas las variables tienen como valor inicial el cero, hasta que se les asigna otro valor con el comando de asignación.

En Cartesia no es posible usar ni crear otras variables que las 7 variables que hemos citado.

El comando para establecer el valor de una variable es:

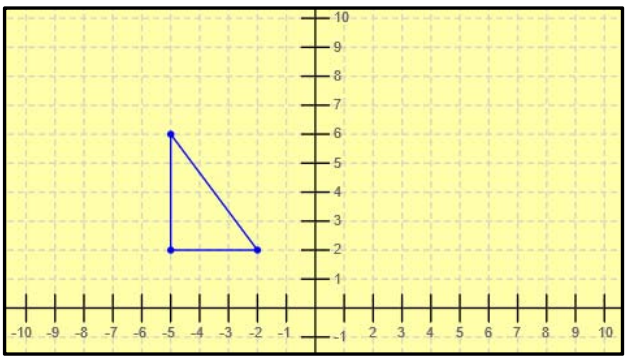
? nuevo valor es (??)

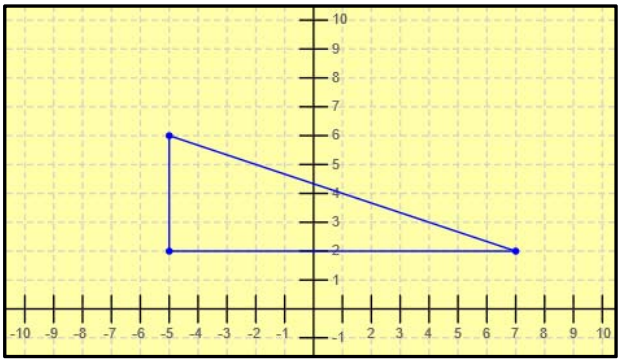
Aquí ? debe sustituirse por el nombre de la variable y ?? por el valor que toma. Así podemos asignar a la variable x el valor 3 de muchas maneras: x nuevo valor es (3), ó x nuevo valor es (6/2) ó x nuevo valor es (2+1), etc.

Escribe este código en el panel de código y pulsa ejecutar:

```
Iniciar programa
x nuevo valor es (-5)
y nuevo valor es (6)
a nuevo valor es (-2)
b nuevo valor es (2)
c nuevo valor es (-5)
d nuevo valor es (2)
Dibujar punto en (x,y)
Dibujar punto en (a,b)
Dibujar punto en (c,d)
Dibujar línea desde (x,y) hasta (a,b)
Dibujar línea desde (a,b) hasta (c,d)
Dibujar línea desde (c,d) hasta (x,y)
Finalizar programa
```

Hemos dibujado un triángulo. Cambia ahora el valor asignado a "a" para que en lugar de -2 tome valor 7, es decir, la cuarta línea debe quedar así: *a nuevo valor es (7)*. Pulsa ejecutar.





Aspecto de los dos triángulos dibujados usando variables.

Comprobarás que modificando el valor de "a" el triángulo ha cambiado. Esto es posible gracias a que hemos usado variables. Si no las hubiéramos utilizado, tendríamos que haber hecho más cambios en nuestro código para dibujar el nuevo triángulo, así que usar variables nos facilita la programación.

Uso de variables

En Cartesia se permiten algunos usos de las variables similares a los de lenguajes de programación de ámbito profesional y académico como son:

a) Una variable **mantiene** siempre su valor a partir de la línea del programa donde se le asigna un valor con el comando de asignación. Por ello decimos que las variables "tienen memoria". Por muchas líneas de código que escribamos, el último valor de la variable siempre es recordado. Si no se le asigna valor, una variable valdrá cero durante todo el programa.

b) Una variable puede tomar su nuevo valor **a partir de su valor anterior**. Supongamos que "a" vale 3 porque hemos escrito "a nuevo valor es (3)". Si en un punto posterior de nuestro programa escribimos "a nuevo valor es (a+1)" esto significa que primero tomamos el papel que había dentro de la caja que tenía un 3, luego calculamos 3+1 que es 4, escribimos el valor resultante en un nuevo papel que metemos en la caja, y tiramos el anterior papel. En resumen, dentro de la caja nos queda un 4, que se ha obtenido sumando 1 al valor anterior que había en la caja.

c) Una variable puede usarse como equivalente a su valor numérico **en cualquier lugar de un programa** donde se requiera un número, por ejemplo para determinar las coordenadas de un punto, para establecer un color o un grosor, o para asignar un nuevo valor a otra variable. Por ejemplo *Dibujar punto en (a, b)* es válido. También es válido *Dibujar punto en (a+3, b-1)* y cualquier expresión donde hagamos uso correcto de los operadores matemáticos.

c nuevo valor es (a+b) también es válido y significa que c pasa a tomar el resultado de sumar a y b.

Supongamos que b vale 3 porque hemos escrito *b nuevo valor es (3)*. Si ahora escribimos *Nuevo color lápiz (b)* eso implica que a partir de esa línea el color de dibujo pasa a ser el 3 (amarillo), ya que Cartesia se encarga de sustituir la variable por su valor numérico. Aún más, podríamos haber escrito *Nuevo color lápiz (b-1)* y eso significaría que el nuevo color de dibujo pasaría a ser el 2 (naranja), ó *Nuevo color lápiz (b-2)* y eso significaría que el nuevo color de dibujo pasaría a ser el 1 (rojo). En cambio si escribiéramos *Nuevo color lápiz (b-3)* obtendríamos un mensaje de error en el panel de mensajes del tipo "*Error: color invocado 0; los colores sólo admiten valores enteros entre 1 y 9. Valor no válido*", ya que el cero no es un valor válido para un color en Cartesia.

d) Una variable **puede compararse** con otro valor numérico para obtener un resultado de verdadero o falso. Por ejemplo si la variable c vale 34 porque hemos escrito *c nuevo valor es (34)*, la comparación (c < 55) dará como resultado VERDADERO porque 34 es menor que 55. En cambio la comparación (c > 55) dará como resultado FALSO ya que 34 no es mayor que 55.

Como una variable puede escribirse en cualquier lugar de un programa donde se requiera un número también pueden usarse comparaciones como (a >= b), cuyo resultado será VERDADERO si a es mayor o igual que b, o FALSO si a es menor que b.

Las comparaciones se usan en los comandos de condición que se explican más adelante.

13. COMANDO REPETICIÓN. BLOQUES DE EJECUCIÓN.

Introducción a la repetición en programación

Ya hemos descrito cómo se puede dibujar un triángulo u otras formas geométricas. Pero, ¿qué ocurre si queremos dibujar 20 triángulos? Quizás podríamos pensar en introducir todas las coordenadas necesarias para definir dónde se ubica cada triángulo, lo cual nos llevaría un buen rato. ¿Y si quisiéramos dibujar 200 triángulos? Si tuviéramos que definir las coordenadas de todos sus vértices una a una, podría llevarnos varias horas o un día. ¿Y si quisiéramos dibujar 2000 triángulos? Posiblemente se volvería tan cansado que nunca lo haríamos. Sin embargo, los ordenadores, tablets, smartphones, etc. tienen la capacidad para dibujar cientos o miles de triángulos en menos de un segundo gracias a los potentes procesadores que llevan incorporados ya que son capaces de realizar cálculos y ejecutar instrucciones a gran velocidad.

En Cartesia es posible dibujar 20, 200, 2000 ó más triángulos usando el **comando de repetición**, similar al de lenguajes de programación de ámbito profesional y académico.

La idea será: definimos las variables necesarias para dibujar el triángulo. Entramos en un comando de repetición y dibujamos el triángulo. Modificamos los valores de las variables que definen donde debe situarse el triángulo, y repetimos tantas veces como queramos.

Vamos a tomar como código de partida el siguiente. Escríbelo en tu panel de código, pulsa Ejecutar y comprueba que aparezca un triángulo dibujado en el panel de dibujo.

```
Iniciar programa
x nuevo valor es (-10)
y nuevo valor es (4)
a nuevo valor es (-8)
b nuevo valor es (2)
c nuevo valor es (-10)
d nuevo valor es (2)
Dibujar línea desde (x,y) hasta (a,b)
Dibujar línea desde (a,b) hasta (c,d)
Dibujar línea desde (c,d) hasta (x,y)
Finalizar programa
```

Comando repetición

El comando repetición se escribe de la siguiente manera:

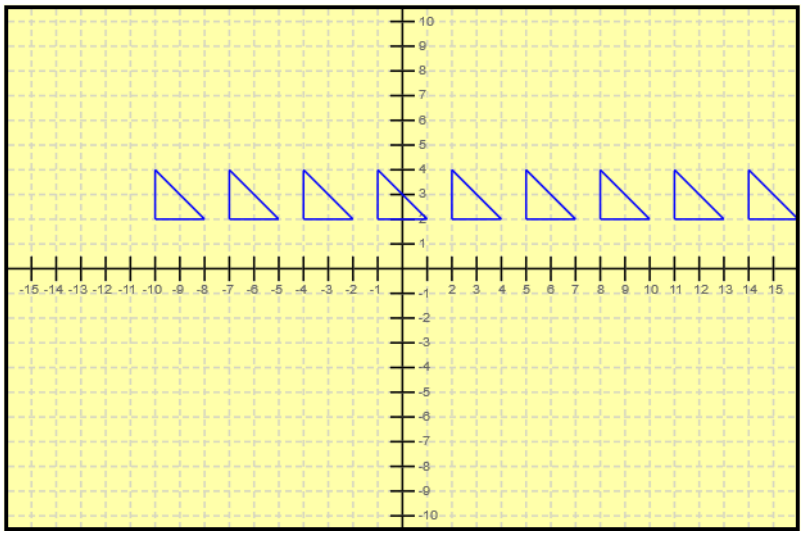
```
Repetir (?) veces ejecutar bloque
Comienzo bloque
???
Fin bloque
```

Donde hemos de sustituir el ? por el número de repeticiones que deseamos realizar y el ??? por las instrucciones que queremos que se ejecuten en cada repetición (por ejemplo las instrucciones para dibujar un triángulo con cada repetición). El código que se ejecutará en cada repetición se llama **bloque de ejecución** y puede comprender tantos comandos como queramos.

A partir del código anterior, escribe el siguiente código y pulsa ejecutar:

```
Iniciar programa
--Definimos las coordenadas iniciales para el triángulo
x nuevo valor es (-10)
y nuevo valor es (4)
a nuevo valor es (-8)
b nuevo valor es (2)
c nuevo valor es (-10)
d nuevo valor es (2)
--Vamos a dibujar 25 triángulos
Repetir (25) veces ejecutar bloque
Comienzo bloque
--Instrucciones para dibujar un triángulo
Dibujar línea desde (x,y) hasta (a,b)
Dibujar línea desde (a,b) hasta (c,d)
Dibujar línea desde (c,d) hasta (x,y)
--Preparamos la siguiente repetición
--Definimos las coordenadas para el siguiente triángulo
x nuevo valor es (x+3)
a nuevo valor es (a+3)
c nuevo valor es (c+3)
Fin bloque
Finalizar programa
```

Fíjate que después de dibujar el triángulo, movemos 3 unidades hacia la derecha sus vértices cambiando los valores de x, de a y de c. Los valores de y, b y d no los cambiamos porque no estamos modificando la situación del triángulo en vertical. El resultado que obtendremos será similar al siguiente:



¿Por qué si hemos creado un programa para dibujar 25 triángulos no vemos 25 triángulos? Siempre que un resultado no sea el que esperas puedes:

- Revisar **el código** para ver si te has equivocado en algo.
- Revisar **el panel de mensajes** a ver si hay algún mensaje de error o aviso.

En este caso en el panel de mensajes encontraremos un mensaje como: "*Programa ejecutado correctamente. Aviso: has usado coordenadas fuera del área visible. El dibujo o parte del dibujo puede estar fuera del área visible.*"

Realmente **sí se han dibujado** los 25 triángulos, lo que pasa es que algunos de ellos están fuera del área visible del panel de dibujo y no los vemos.

Prueba a escribir el mismo programa pero haciendo que los valores de **y**, **b** y **c** se incrementen también 3 unidades en cada repetición. Para ello tendrás que añadir tres líneas con "y nuevo valor es (y+3)", "b nuevo valor es (b+3)", "d nuevo valor es (d+3)". Con esto haremos que el triángulo se vaya dibujando no sólo hacia la derecha sino también hacia arriba. Si en lugar de (y+3), (b+3) y (d+3) escribimos (y-3), (b-3) y (d-3) lograremos que el triángulo se vaya dibujando hacia abajo.

Haz pruebas modificando el desplazamiento horizontal y vertical dándole distintos valores y comprueba los resultados. Comprobarás que si estableces separaciones muy grandes (por ejemplo 40 unidades) sólo verás un triángulo porque el resto queda muy alejado. Si estableces separaciones muy pequeñas (por ejemplo 0.5 unidades) verás como los triángulos se dibujan superponiéndose: no hay separación suficiente como para poder ver cada triángulo con claridad.

Haz pruebas modificando en cada repetición las variables **x**, **a**, **c**, **y**. Con esto puedes desplazar el triángulo en horizontal e ir cambiando su altura en cada repetición.

No hay límite en el número de repeticiones que puedes establecer. Además, para aquellos que se atrevan, Cartesia permite hacer "repeticiones de repeticiones". Por ejemplo, repetir diez veces dibujar 25 triángulos en horizontal, modificando en cada vez la posición vertical, de modo que podemos llenar el panel de triángulos o de cualquier dibujo que se nos ocurra.

Otra posibilidad interesante de las repeticiones es que podemos cambiar los colores en cada repetición usando el comando **Nuevo color lápiz (?).**

Haz todas las pruebas que se te ocurran. Si en un momento dado quieres volver a empezar de cero, pulsa el botón "Limpiar todo".

14. OPERADORES LÓGICOS Y DE COMPARACIÓN. COMANDO CONDICIÓN.

Operadores lógicos

Cartesia permite definir que una parte del código se ejecute sólo si se cumple una condición. Para ello se dispone del comando condición y de dos operadores lógicos. Los operadores lógicos de Cartesia son estos:

Operador	Significado	Ejemplo
así como	Y se cumple también que	(x<3 así como y<5)
o bien	O bien se cumple que	(x<3 o bien y<5)

Para los ejemplos indicados en la tabla anterior, supongamos que x vale 5 mientras que y vale -2. La comparación lógica (x<3 así como y<5) equivale a preguntar ¿Es 5 menor que 3 así como -2 menor que 5? El resultado que se obtiene es FALSO porque 5 no es menor que 3, independientemente de que -2 sea menor que 5, porque se requiere que se cumplan ambas condiciones.

La comparación lógica (x<3 o bien y<5) equivale a preguntar ¿Es 5 menor que 3 o bien -2 menor que 5? El resultado que se obtiene es VERDADERO porque aunque 5 no es menor que 3, sí se cumple que -2 sea menor que 5. En este caso para obtener VERDADERO basta con que se cumpla una de las dos condiciones.

Comando condición

El comando condición se escribe de la siguiente manera:

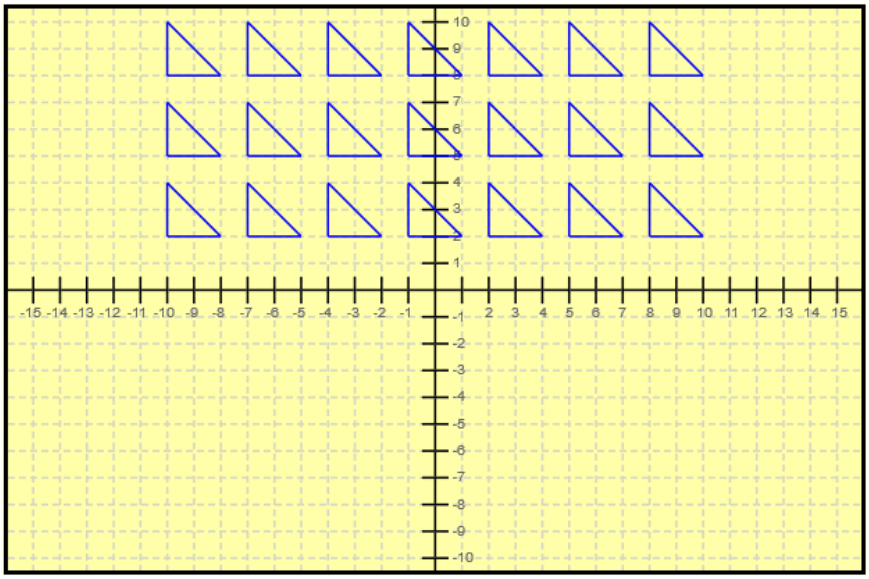
```
Si verdadero (?) ejecutar bloque
Comienzo bloque
???
Fin bloque
```

Donde hemos de sustituir el ? por la condición que determina si se debe ejecutar el bloque de instrucciones o no, y el ??? por las instrucciones que queremos que se ejecuten cuando se cumpla la condición. El código que se ejecutará cuando se cumpla la condición es un **bloque de ejecución** y puede comprender tantos comandos como queramos. En Cartesia hay dos comandos que conllevan el uso de bloques de ejecución: el comando repetición y el comando condición.

A partir del código que hemos usado anteriormente para dibujar 25 triángulos, vamos a modificar el programa para hacer que cuando los triángulos vayan a salirse del panel de dibujo, en lugar de continuar hacia la derecha, sigan dibujándose empezando en el lateral izquierdo un poco más arriba creando filas de triángulos. Escribe y ejecuta este código:

```
Iniciar programa
--Definimos las coordenadas iniciales para el triángulo
x nuevo valor es (-10)
y nuevo valor es (4)
a nuevo valor es (-8)
b nuevo valor es (2)
c nuevo valor es (-10)
d nuevo valor es (2)
--Vamos a dibujar 25 triángulos
Repetir (25) veces ejecutar bloque
Comienzo bloque
--Instrucciones para dibujar un triángulo
Dibujar línea desde (x,y) hasta (a,b)
Dibujar línea desde (a,b) hasta (c,d)
Dibujar línea desde (c,d) hasta (x,y)
--Preparamos la siguiente repetición
--Definimos las coordenadas para el siguiente triángulo
x nuevo valor es (x+3)
a nuevo valor es (a+3)
c nuevo valor es (c+3)
--Si están muy a la derecha volvemos a la izquierda y
subimos para crear una nueva fila
Si verdadero (x>10) ejecutar bloque
Comienzo bloque
x nuevo valor es (-10) --Volver a la izquierda
a nuevo valor es (-8) --Volver a la izquierda
c nuevo valor es (-10) --Volver a la izquierda
y nuevo valor es (y+3) --Subimos
b nuevo valor es (b+3) --Subimos
d nuevo valor es (d+3) --Subimos
Fin bloque --Termina el bloque de la condición
Fin bloque --Termina el bloque de la repetición
Finalizar programa
```

El resultado será similar al siguiente:



Fíjate que después de dibujar el primer triángulo, movemos 3 unidades hacia la derecha sus vértices cambiando los valores de **x**, de **a** y de **c**. Los valores de **y**, **b** y **d** no los cambiamos porque no estamos modificando su situación en vertical.

El valor de x nos dice si un triángulo está muy a la izquierda (valores como -10) o muy a la derecha (valores como 10). Lo que hacemos es en cada repetición comprobar si la x ha llegado a ser mayor que 10, lo que nos diría que el triángulo está ya muy a la derecha. Cuando esto ocurre se cumple la condición **Si verdadero (x>10) ejecutar bloque** y se ejecuta el bloque de condición. Al ejecutarse este bloque, volvemos a colocar el triángulo a la izquierda con estas instrucciones:

```
x nuevo valor es (-10) --Volver a la izquierda
a nuevo valor es (-8) --Volver a la izquierda
c nuevo valor es (-10) --Volver a la izquierda
```

Además, movemos hacia arriba con estas instrucciones:

```
y nuevo valor es (y+3) --Subimos
b nuevo valor es (b+3) --Subimos
d nuevo valor es (d+3) --Subimos
```

Una vez hecho esto, las repeticiones continúan, empezando ahora desde una posición a la izquierda y más arriba que la anterior. El resultado es que se dibuja **una nueva fila** de triángulos. Como en cada repetición comprobamos si se han ido muy a la derecha, cuando esto ocurra, volveremos a hacer lo mismo que antes, es decir, establecer las coordenadas para empezar a la izquierda y un poco más arriba. Así hasta que se terminan las repeticiones.

¿Veremos los 25 triángulos en el panel de dibujo? Puede que no los veamos todos porque algunos hayan quedado fuera del área visible. O quizás algunos aparezcan partidos porque están cerca del borde del área visible. Presta atención al panel de mensajes porque es posible que tengas algún aviso.

Puedes hacer que cada fila se dibuje de un color diferente. Para ello puedes escribir como primeras líneas del programa:

```
e nuevo valor es (4)
Nuevo color lápiz (e)
```

Si pulsas Ejecutar verás que todos los triángulos se dibujan en verde, porque hemos establecido que el color del lápiz de dibujo es el 4 (verde).

Ahora modifica el bloque de condición así (introduciendo las dos líneas que señalamos):

```
Si verdadero (x>10) ejecutar bloque
Comienzo bloque
x nuevo valor es (-10) --Volver a la izquierda
a nuevo valor es (-8) --Volver a la izquierda
c nuevo valor es (-10) --Volver a la izquierda
y nuevo valor es (y+3) --Subimos
b nuevo valor es (b+3) --Subimos
d nuevo valor es (d+3) --Subimos
e nuevo valor es (e+1) --Valor de e se incrementa en 1
Nuevo color lápiz (e) -- Definir nuevo color de lápiz
Fin bloque --Termina el bloque de la condición
```

Con esto lo que conseguimos es que cada vez que se alcance un valor de x grande (x>10) además de llevarnos el triángulo a la izquierda y arriba, incrementamos el valor de la variable "e" que representa el color y le cambiamos el color (en cada ocasión de 4 a 5 al sumar 4+1, luego de 5 a 6 al sumar 5+1, luego de 6 a 7, etc.)

Si pulsas ejecutar, cada fila de triángulos se dibujará de un color diferente.

Operadores de comparación en condiciones

Para definir condiciones se dispone de los siguientes operadores:

Símbolo	Significado	Ejemplo
==	Es igual que…	x == y
<	Es menor que…	x < y
>	Es mayor que…	x > y
<=	Es menor o igual que…	x <= y
>=	Es mayor o igual que…	x >= y
!=	Es distinto que…	x != y

Ten cuidado porque escribir algo como *Si verdadero (x=10) ejecutar bloque* generará un error y en el panel de mensajes se indicará algo similar a lo siguiente: <<ERROR: la instrucción condicional contiene = en lugar de ==. Por favor revise el código. Línea no válida>>

Recuerda que para comparar si es igual debes usar el símbolo == con dos iguales seguidos. Si usas un solo igual te aparecerá el mensaje de error que hemos comentado.

15. OPERADORES MATEMÁTICOS Y CURVAS CON CARTESIA

Cartesia permite utilizar una serie de **operadores matemáticos** aplicables en cualquier lugar donde se requiera un valor numérico. Los operadores matemáticos disponibles son ocho:

Símbolo	Significado	Ejemplo
+	Suma	a + b
–	Resta	a – b
*	Multiplicación	a * b
/	División	a / b

Símbolo	Significado	Ejemplo
SQR(num)	Raíz cuadrada de num	SQR (a+b)
^	Potencia	a ^ 2 + b
%	Operador módulo (resto entero de división entre dos números enteros)	a % b
( )	Paréntesis. Prelación de operaciones	a ^ (2+b)

De estos operadores usa aquellos que conozcas. No es necesario conocer todos los operadores para crear programas en Didac-Prog Cartesia.

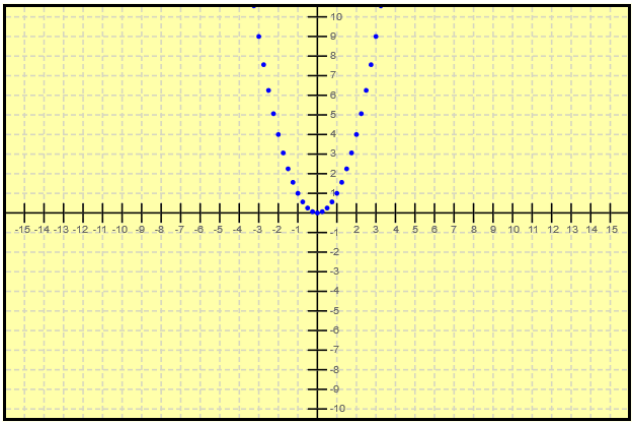
Las representaciones que usan potencias y raíces cuadradas permiten trazar curvas de uso frecuente en matemáticas. Por ello, aunque Cartesia no permite directamente dibujar curvas, **podemos trazar curvas** a base de dibujar puntos muy juntos, o a base de dibujar líneas pequeñas e irlas uniendo para que parezcan una curva. Prueba a hacer esto sobre papel, verás que puedes dibujar una curva a base de dibujar puntos muy juntos o pequeños segmentos rectos e irlos uniendo.

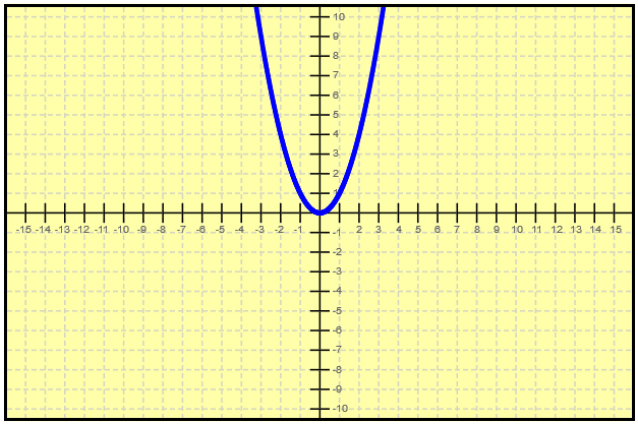
Vamos a representar usando puntos muy juntos la curva que se suele denominar cuadrática y se suele escribir como $y = x^2$, que es lo mismo que y = x^2 o que y = x*x. Para cada valor de x que elijamos dibujaremos un punto en la coordenada (x, x*x). Por ejemplo para x = -3 dibujaremos el punto (-3, -3*-3) que es (-3, 9). O para x = 2 dibujaremos el punto (2, 2*2) que es (2, 4). Prueba a ejecutar estos códigos:

```
-- PROGRAMA QUE TRAZA LA FUNCIÓN CUADRÁTICA CON PUNTOS SEPARADOS
Iniciar programa
x nuevo valor es (-5)
Repetir (700) veces ejecutar bloque
Comienzo bloque
Dibujar punto en (x, x^2)
x nuevo valor es (x+0.25)
Fin bloque
Finalizar programa
```

```
-- PROGRAMA QUE TRAZA LA FUNCIÓN CUADRÁTICA CON PUNTOS TAN JUNTOS QUE SE VEN COMO UNA CURVA
Iniciar programa
x nuevo valor es (-5)
Repetir (700) veces ejecutar bloque
Comienzo bloque
Dibujar punto en (x, x^2)
x nuevo valor es (x+0.015)
Fin bloque
Finalizar programa
```

Si te fijas en ambos casos hemos dibujado 700 puntos partiendo desde el valor -5 e incrementando el valor x en cada repetición en un caso 0.25 unidades (-5, -4.75, -4.5, -4.25, -3.0, -2.75, -2.50, -2.25, etc.) y en otro 0.015 unidades (-5, -4.985, -4.970, -4.955, -4.940, -4.925, -4.910, etc.). El resultado puede ser similar a esto:





En un caso, los puntos están bastante separados por lo que intuimos la forma curva que se generaría al unir los puntos. En el otro caso, los puntos están tan unidos que llegamos a ver la curva. En ambos casos

seguramente se han dibujado puntos fuera del área visible pero esto no nos preocupa. Puedes comprobarlo mirando el panel de mensajes.

Fíjate cómo pasar de un resultado a otro ha dependido de un cambio muy pequeño en el código de un programa. Sin embargo el resultado ha sido muy distinto. Prueba a hacer distintas pruebas:

a) Establece 100, 200, 300 etc. repeticiones en lugar de 700 y ejecuta ambos códigos. Trata de interpretar lo que ocurre.

b) En lugar de 0.25 y 0.015 usa otros valores, por ejemplo 3, 2, 1, 0.5, 0.2, 0.1, 0.05 etc. Trata de interpretar lo que ocurre.

Consulta siempre el panel de mensajes pues te puede dar orientaciones valiosas.

Hay distintos factores que pueden dar lugar a que los resultados de nuestros programas no sean los esperados, pero debes prestar especial atención a:

- El **número de repeticiones:** puede ser demasiado pequeño o demasiado grande.
- La forma en que incrementemos (o reduzcamos) el valor de las variables. Pueden ser **incrementos** (o reducciones) demasiado grandes o demasiado pequeñas.
- Si las instrucciones de un condicional se ejecutan o no se ejecutan. Hay casos donde el bloque de ejecución de un condicional **no se ejecuta nunca** porque establecemos una condición que no se cumple nunca.

16. MENÚ DE OPCIONES: ABRIR Y GUARDAR PROGRAMAS.

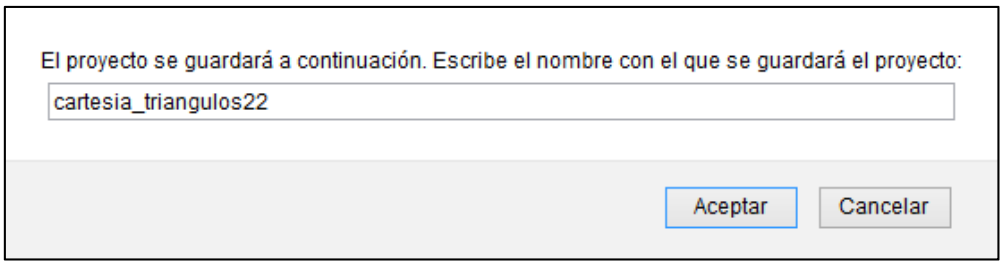
Introducción

Cuando enumeramos las partes en que se divide la pantalla principal de Didac-Prog Cartesia mencionamos la existencia de un menú de opciones en la parte superior izquierda de la pantalla. Al poner el ratón sobre este menú se despliegan distintas opciones: *Abrir proyecto*, *Guardar proyecto*, *Abrir ejemplo* y *Deshacer limpiar*.

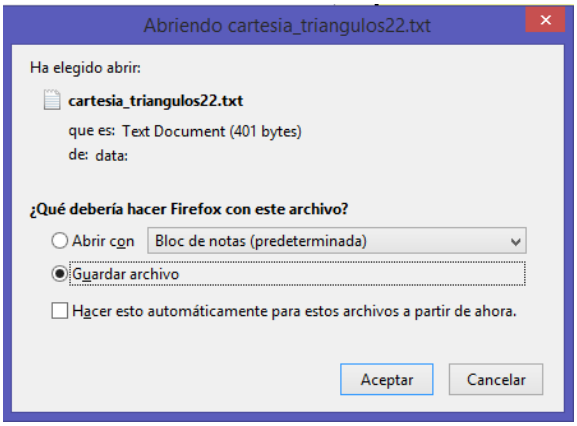
Guardar proyecto

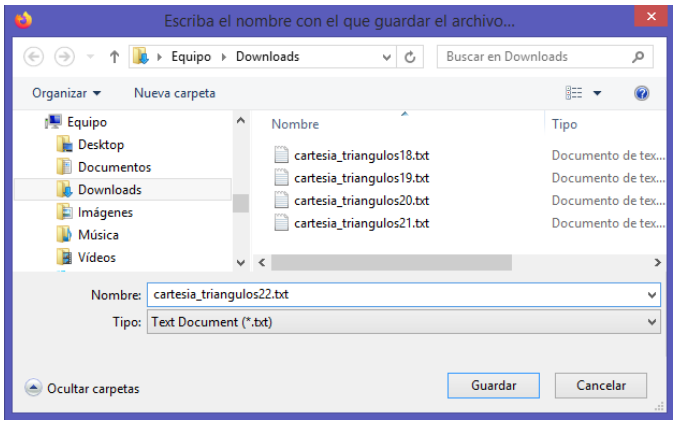
La opción *Guardar proyecto* nos permite guardar el código que tengamos escrito en el panel de código (a lo que llamamos "proyecto") en un archivo para poder recuperarlo cuando deseemos. Supongamos que hemos escrito nuestro código y después de ejecutarlo vemos que el resultado es correcto y queremos guardarlo para entregárselo al profesor o para usarlo otro día. Para ello nos colocamos encima del menú opciones y al desplegarse elegimos "Guardar proyecto". En este momento nos aparecerá un diálogo donde se nos indica "El proyecto se guardará a continuación. Escribe el nombre con el que se guardará el proyecto:". Por defecto nos aparece como inicio del nombre "cartesia_" pero esto podemos eliminarlo si queremos.

Para probar esta funcionalidad, escribe un programa para dibujar un triángulo donde la primera línea sea un comentario con tu nombre, la fecha y la hora. Pulsa "Guardar proyecto" y ponle el nombre cartesia_triangulos22. Una vez escrito esto, pulsa Aceptar.



Dependiendo del navegador que utilices y de las opciones que tengas configuradas, el archivo será guardado directamente (por ejemplo en la carpeta Descargas o *Downloads*), o bien se abrirá un nuevo diálogo preguntando si se desea abrir o guardar el archivo. En este caso, elegiremos *Guardar* y pulsamos *Aceptar*.





El archivo se habrá guardado bien en una carpeta (como Descargas, *Downloads* u otra) o bien en la ruta que tú hayas elegido para ello. Recomendamos ir guardando los proyectos en la carpeta *projects* del directorio de Cartesia, en una ruta que será similar a C:/Users/Gustavo/cartesia2020/projects/, pero esto es opcional y pueden guardarse donde se desee, incluso en un pendrive si se prefiere.

Usa el explorador de archivos de tu ordenador y comprueba que el archivo se haya guardado. Ábrelo (doble click sobre él) y comprueba que contenga el código que habías escrito.

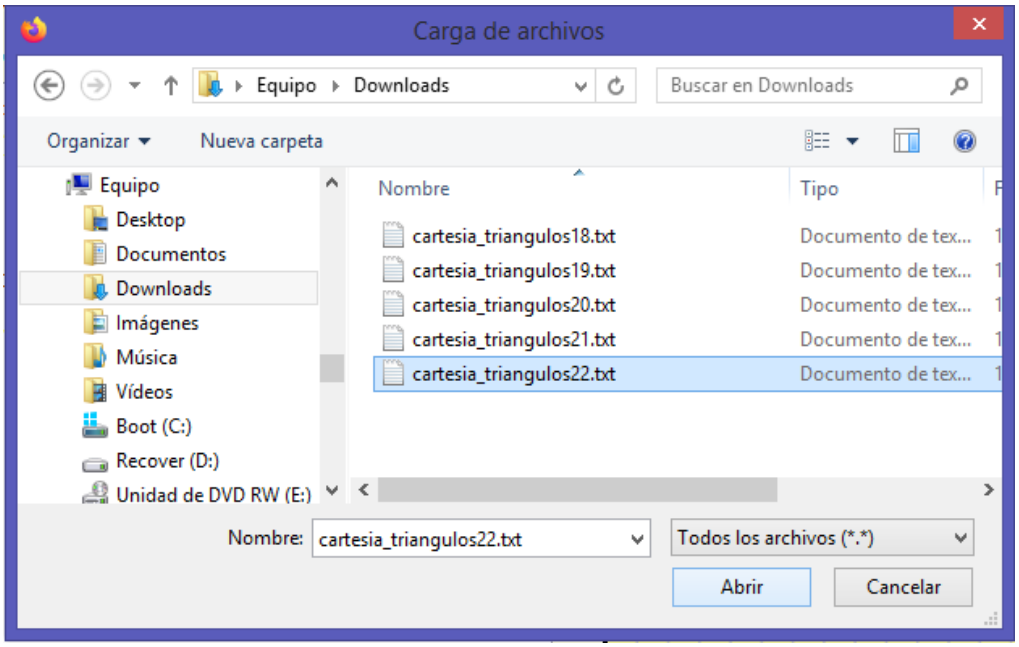
Los archivos de Didac-Prog Cartesia son simples archivos de texto (txt).

Abrir proyecto

La opción *Abrir proyecto* nos permite abrir un proyecto desde un archivo. Este proyecto podemos haberlo guardado con anterioridad, haberlo descargado desde aprenderaprogramar.com, haberlo recibido por correo electrónico, haberlo recibido en un pendrive, etc.

Ten cuidado: antes de elegir esta opción, **guarda el código** que tengas en el panel de código. Si no lo haces, perderás el código en el que estuvieras trabajando, al ser reemplazado por el código del proyecto que abres.

Una vez pulsamos en el menú opciones, "Abrir proyecto", se nos presentará una ventana desde donde deberemos buscar y seleccionar el archivo a abrir.



Tras esto en el panel de código nos aparecerá el proyecto y podremos empezar a trabajar en él. Si hacemos modificaciones que queramos conservar, deberemos guardarlo con el mismo nombre o con un nombre nuevo, según consideremos que sea más conveniente.

17. MENÚ DE OPCIONES: ABRIR EJEMPLO Y DESHACER LIMPIAR.

Abrir ejemplo

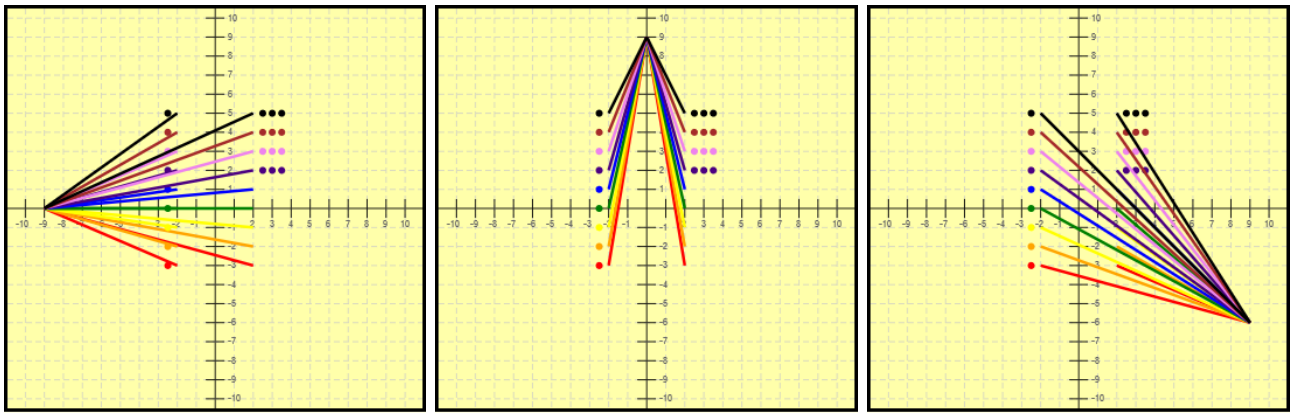
Dentro del menú de opciones, "Abrir ejemplo" nos permite abrir un ejemplo de programa que hace uso de todos los comandos de Didac-Prog Cartesia.

Ten cuidado: antes de elegir esta opción, **guarda el código** que tengas en el panel de código. Si no lo haces, perderás el código en el que estuvieras trabajando, al ser reemplazado por el código del ejemplo que abres.

Pulsa en *Abrir ejemplo* y luego en el botón *Ejecutar*. En pantalla verás el resultado de ejecución.

Prueba a modificar este programa. Por ejemplo cambia el valor inicial de x. Para ello busca la línea donde se da valor a x y donde dice "x nuevo valor es (0)" escribe "x nuevo valor es (-9)". Prueba con otros valores como -3, 3, 6, 9, etc. y comprueba los resultados. Cambia también el valor inicial de la variable c, comprueba los resultados y trata de interpretar el por qué de esos resultados.

Si en algún momento quieres volver a empezar desde el ejemplo original, basta con que vuelvas a seleccionar en el menú de opciones "Abrir ejemplo". Eso sí, recuerda que perderás el código que exista en el panel de código si no lo has guardado previamente.



Distintas ejecuciones del programa de ejemplo, cambiando el valor de las variables x y c.

Siempre que se pulsa *Abrir ejemplo* se carga el mismo programa. Si quieres ver otros ejemplos, puedes hacerlo usando la opción *Abrir proyecto*, y en la ruta donde se encuentre Cartesia, elegir la carpeta *Examples* y abrir cualquiera de los más de 40 proyectos que encontrarás ahí guardados.

Deshacer limpiar

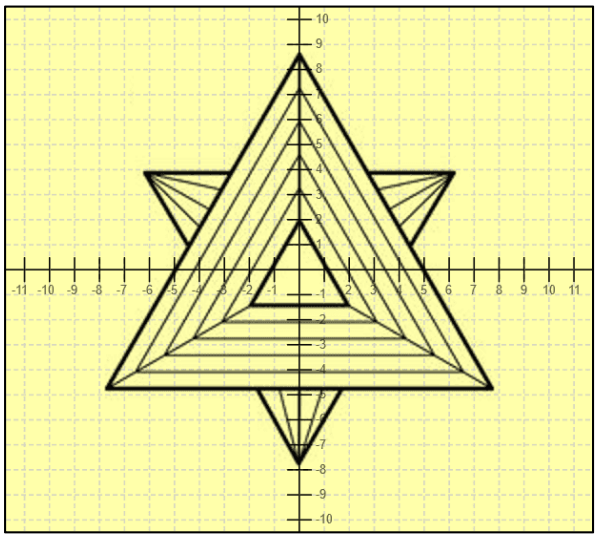
El botón "Limpiar todo" implica que el código y dibujos que pudieran existir desaparezcan y Cartesia vuelva a su situación inicial con todo en blanco. A veces llevamos trabajando un buen rato en un programa y, sin darnos cuenta, pulsamos "Limpiar todo" sin haber guardado nuestro código. Si te ocurre esto, puedes recuperar tu código si inmediatamente después de limpiar todo haces una de estas dos cosas:

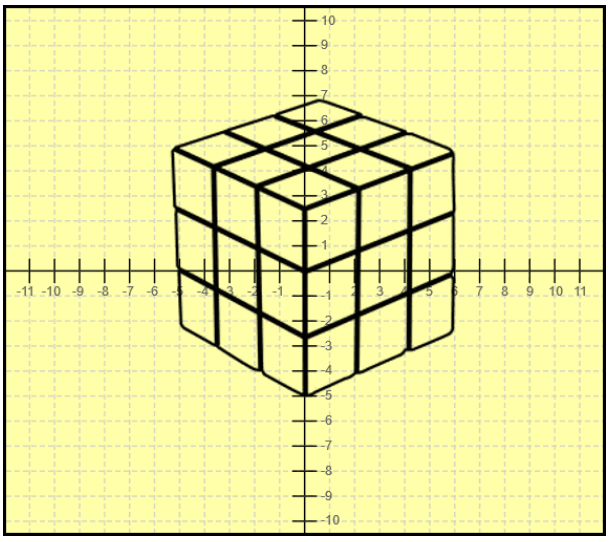
a) Pulsa la combinación de teclas CTRL+Z para **deshacer** el limpiar todo y recuperar tu código. A continuación guarda tu código para no perderlo y sigue trabajando.

b) Elije la opción **Deshacer limpiar** del menú de opciones. Esto, al igual que CTRL+Z, te permite recuperar tu código para poder guardarlo y seguir trabajando.

18. PLANTEANDO RETOS CON DIDAC-PROG CARTESIA.

Una actividad interesante con Didac-Prog Cartesia es plantearse un reto. Por ejemplo, elegir una figura y plantearnos ¿Cómo podría realizar esta figura o dibujo con programación sin necesidad de dibujar una a una todas las líneas que la forman? Rétate a ti mismo o a un amigo y supérate como programador. Por si no se te ocurre ningún reto, aquí tienes dos, y en aprenderaprogramar.com podrás encontrar más.





DIDAC-PROG

CARTESIA

ANEXO PARA PROFESORES

A-1. INTRODUCCIÓN

Cartesia es una aplicación concebida con fines didácticos y de enseñanza. Deliberadamente se diseñó con un objetivo: **ser simple y potente.** Un problema frecuente en el aula es la falta de tiempo para aprender a usar aplicaciones con enormes cantidades de funciones y posibilidades. Cartesia no aspira a ser una de ellas. De ahí que tenga funcionalidades limitadas respecto a lenguajes de programación profesionales o aplicaciones multipropósito para la enseñanza de las matemáticas o de la programación en sentido más amplio.

La intención al crear esta aplicación no ha sido la enseñanza de un lenguaje de programación, ni de aspectos generales de los lenguajes de programación, sino la enseñanza de la algoritmia y fundamentos de la lógica de programación o ***pensamiento computacional*** que, dicho con palabras un tanto ampulosas, "constituye el sustrato base de nuestras sociedades informatizadas". Con otra perspectiva, al mismo tiempo constituye una forma de enseñanza interactiva de las matemáticas al permitir trabajar conceptos como sistema cartesiano de coordenadas, puntos y rectas, solución gráfica de ecuaciones, funciones, etc. en un entorno donde el alumno pueda sentir que domina a la aplicación y se sabe conocedor de todo lo que hace. Cartesia se concibe pues con otro fin claro: evitar que el alumno se sienta abrumado por un exceso de información y posibilidades.

Otras virtudes de la sencillez de Cartesia son ocupar muy poco espacio (<5 Mb) y permitir trabajar con plena operatividad sin necesidad de estar conectado a internet.

Aquellos profesores que busquen una herramienta orientada a las matemáticas con más potencia y complejidad pueden usar *Geogebra* (https://www.geogebra.org/?lang=es-ES), una aplicación que permite tanto la enseñanza avanzada en múltiples campos de las matemáticas como la introducción de programación.

Aquellos profesores que busquen otros entornos para enseñanza de la programación a niños disponen de *Scratch*, *Alice*, *App-Inventor*, por citar algunas posibilidades que cuentan con gran popularidad.

Cartesia puede usarse si se desea como paso previo, posterior, o complementario, respecto a cualquiera de las herramientas que hemos citado.

A-2. DISEÑO FLEXIBLE DEL LENGUAJE DE CARTESIA. VENTAJAS E INCONVENIENTES.

Didac-Prog Cartesia se basa en un lenguaje de programación que no tiene un nombre específico: nos referimos a él simplemente como "lenguaje de la aplicación" o "lenguaje de Didac-Prog Cartesia". Este lenguaje ha sido diseñado **específicamente** para la aplicación. Está concebido con fines educativos y tiene algunas similitudes y algunas diferencias con respecto a los lenguajes de programación convencionales.

Como similitud principal podemos señalar el uso de las tres estructuras de control de la programación clásica estructurada: secuencial, condicional y de repetición. Entre las diferencias podemos citar el tener un conjunto de comandos muy limitado y simplificado respecto a lenguajes de programación estándar. Otra diferencia significativa es que Cartesia se ha diseñado con una sintaxis laxa basada en expresiones regulares, frente a las sintaxis más rígidas propias de los lenguajes de programación.

Por ejemplo, una instrucción como "Nuevo color lápiz (2)" en general en otro lenguaje no podría ser escrita de otra manera. En Cartesia además de como se ha indicado también puede ser escrita como "El nuevo color del lápiz pasa a ser (2) a partir de ahora" o como "Se cambia el color del lápiz: ahora será (2)" que son formas más extensas. Pero también se puede escribir como "Nuevo color (2)" ó como "Color (2)" que son formas más reducidas.

Cartesia acepta **indistintamente** letras mayúsculas y minúsculas, así como letras con tilde y sin tilde. Resulta admisible tanto "Dibujar línea" como "DIBUJAR LÍNEA" como "DIBUJAR linea" o "dibujar linea".

La forma que se indica en el manual de usuario (y que se inserta si se pulsa en el panel de comandos) es la forma "recomendada" pensada por los creadores de Cartesia (por ejemplo "Nuevo color lápiz (?)"). No

obstante, si usted como docente considera preferible que sus alumnos usen otra forma dentro del rango de expresiones permitidas, puede optar por cualquiera de las formas alternativas válidas.

El diseño flexible del lenguaje se ha hecho buscando:

a) Que el alumno pueda centrarse en la **lógica y algoritmia** de sus programas (el "pensamiento computacional"), sin tener que memorizar una sintaxis rígida. Al admitirse variantes, incluso dos formas distintas para un mismo comando dentro de un mismo programa, el lenguaje se aproxima más al lenguaje natural y se distancia de los lenguajes formales.

b) Que el profesor pueda proponer el uso de **formas alternativas** que considere más convenientes de acuerdo con sus criterios docentes.

El diseño flexible pensamos que tiene ventajas pedagógicas relevantes respecto a otros lenguajes de programación. También algún inconveniente. La instrucción "Nuevo color lápiz (2)" es flexible para las formas que hemos citado y otras coherentes, pero también admite formas indeseables (por incoherentes, por tener faltas ortográficas, etc.) si cumplen la expresión regular requerida. No obstante, se ha considerado que las ventajas pedagógicas pesaban más que los inconvenientes y eran motivo suficiente para apostar por este diseño.

La siguiente tabla recoge los **requisitos de sintaxis** de cada comando de Cartesia y algunos ejemplos válidos de sintaxis. Los corchetes indican opcionalidad. Las comillas y paréntesis indican obligatoriedad. Usted puede decidir, dentro del marco de validez especificado, cómo proceder a la hora de utilizar en mayor o menor medida la flexibilidad que ofrece el lenguaje.

Comando	Requisitos sintaxis	Ejemplos válidos
Iniciar programa	Contener "iniciar" como inicio de línea, seguido o no de cadenas adicionales.	Iniciar Iniciar programa INICIAR EJECUCIÓN Iniciar la ejecución del programa
Dibujar punto	Contener [cadena] "dibujar punto" [cadena] (expr1,expr2) [cadena]	Dibujar punto (5,3) dibujar punto en (a/5,b) Ahora Dibujar punto (5,3) Ahora dibujar punto (a,b) y proseguir
Dibujar línea	Contener [cadena] "dibujar linea" [cadena] (expr1,expr2) [cadena] (expr3,expr4) [cadena]	Dibujar línea (-3,-2) (5,1) dibujar línea desde (-3,-2) hasta (5,1) Ahora dibujar línea de (-3,-2) a (5,1) Dibujar línea (-3,-2) (5,1) y proseguir
Nuevo color lápiz	Contener [cadena] "color" [cadena] (expr1) [cadena] y la expresión evaluar a un valor numérico entero entre 1 y 9.	Color (4) Nuevo color lápiz (4) ahora nuevo color es (4) verde el color pasa a ser (4)
Nuevo grosor lápiz	Contener [cadena] "grosor" [cadena] (expr1) [cadena] y la expresión evaluar a un valor numérico entero entre 1 y 3.	Grosor (3) Nuevo grosor lápiz (3) ahora nuevo grosor es (3) grueso el grosor pasa a ser (3)
Condición (sobre bloque) *	Contener las líneas: [cadena] "verdadero" (expr1) [cadena] "comienzo bloque" [cadena] [comandos] "fin bloque" [cadena]	Si verdadero (x>10) entonces Comienzo bloque x nuevo valor es (-8) y nuevo valor es (y+1) Fin bloque
Repetición (de bloque) **	Contener las líneas: [cadena] "repetir" (expr1) [cadena] "comienzo bloque" [cadena] [comandos] "fin bloque" [cadena]	repetir (21) veces bloque comienzo bloque dibujar punto en (x, sqr (4*(x^2-1))) dibujar punto en (x, -sqr (4*(x^2-1))) x nuevo valor es (x+0.2) fin bloque

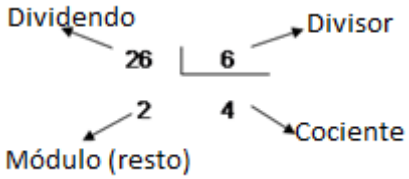
Comando	Requisitos sintaxis	Ejemplos válidos
Asignar valor a variable	Contener: identificador "valor" [cadena] (expr1) [cadena] con identificador válido y la expresión evaluar a un valor numérico.	x valor (-5) x nuevo valor es (-5) x nuevo valor será (x-5) seguidamente x ahora toma valor (x-5)
Finalizar programa	Contener "finalizar" como inicio de línea, seguido o no de cadenas adicionales.	Finalizar Finalizar programa FINALIZAR EJECUCIÓN Finalizar la ejecución del programa

* La expresión debe ser una comparación que evalúe a verdadero o falso.
** La expresión debe generar un valor numérico entero positivo.

A-3. USO DEL OPERADOR MÓDULO PARA CREAR PATRONES

El operador módulo muchas veces es poco o nada estudiado en enseñanza primaria o secundaria, pero resulta **de gran interés en programación**. Puede tener usos diversos siendo uno de ellos la creación de patrones numéricos, como veremos a continuación.

El módulo, resto o residuo de una división entre dos números enteros es el valor numérico que después de multiplicar divisor por cociente permitiría alcanzar el dividendo, es decir, Dividendo = Cociente x Divisor + Módulo o resto.



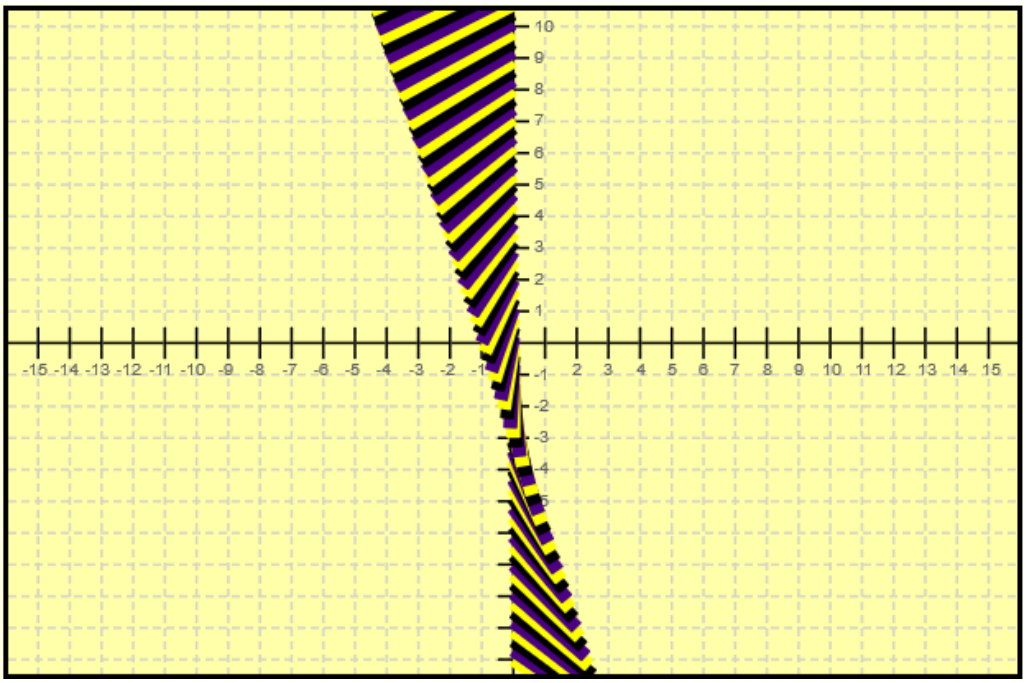
En Cartesia el resto de una división entre enteros puede obtenerse usando el operador módulo (%). Por ejemplo (26 % 6) devuelve el resto de la división, es decir, 2. Aunque este operador usa el mismo símbolo del tanto por ciento, no tiene relación alguna con tantos por ciento. El motivo por el que se usa este símbolo es seguir la convención a que se atienen muchos lenguajes de programación.

Es un operador que puede ser no conveniente utilizar con niños de corta edad pues puede entrañar algunas dificultades para ellos. El profesor debe decidir si explicar este operador y a qué edad.

Un ejemplo mostrará la utilidad de este operador:

```
-- Ejemplo de uso del operador módulo para crear patrones
Iniciar programa
Nuevo grosor lápiz (3) -- Trazado con lápiz grueso
x nuevo valor es (0) -- Coordenada x punto origen
y nuevo valor es (-10) -- Coordenada y inicial punto origen
b nuevo valor es (3) -- Representa el salto de color, iremos contando intervalos de 3: 3+3, 6+3, 9+3, 12+3, 15+3...
c nuevo valor es (-1) -- Variable para almacenar el color, partimos de -1 por conveniencia
d nuevo valor es (-0.60) --Variable que controlará el desplazamiento horizontal del punto de destino
Repetir (100) veces ejecutar bloque --El número de repeticiones lo establecemos a conveniencia para lograr el efecto deseado
Comienzo bloque
c nuevo valor es (c+b) --Creamos patrón numérico c toma valores 3, 6, 9, 12, 15, 18, 21...
Nuevo color lápiz (c%9+1) --En cada repetición cambia color con patrón 3, 6, 9, 3, 6, 9, 3, 6, 9, 3...
Dibujar línea desde (x,y) hasta (2-d, y-2) -- Con d e y variable las coordenadas de destino se van moviendo; también se mueve la y de origen
d nuevo valor es (d+0.1) --Cambio para próxima repetición
y nuevo valor es (y+0.3) --Cambio para próxima repetición
Fin bloque
Finalizar programa
```

Este programa hace uso de variables y un bucle repetición para dibujar líneas que pueden crear (con un poco de imaginación) la apariencia de una escalera o figura sometida a torsión:



Resultado de ejecución del ejemplo planteado. Tener en cuenta que el aspecto puede variar en función del dispositivo (tamaño y resolución de pantalla) sobre el que se ejecute.

El programa puede interpretarse a partir de los comentarios (que hemos señalado en color azul).

Aquí c es una variable con valores -1, 2, 5, 8, 11, 14, 17, 20, 23… gracias a reasignar a c en cada repetición (c+b), esto es, c toma valores (-1+3), (2+3), (5+3), (8+3), (11+3), (14+3), (17+3), (20+3)… El valor -1 no es utilizado y se usa únicamente para iniciar la serie usando como primer valor útil (-1+3).

Al afectar c con el módulo de la división por 9 (operación c%9), obtenemos 2, 5, 8, 2, 5, 8, 2, 5, 8, 2, 5, 8, 2… con lo cual ya tenemos un patrón de valores enteros en el rango de los colores de dibujo, lo cual nos va a permitir crear un patrón de repetición de colores.

Finalmente sumando 1 (hacemos c%9+1) transformamos la serie en 3, 6, 9, 3, 6, 9, 3, etc. que aplicado al comando *Nuevo color lápiz* genera el patrón de colores amarillo, violeta, negro, amarillo, violeta, negro, amarillo… ya que en Cartesia 3 es amarillo, 6 violeta y 9 negro.

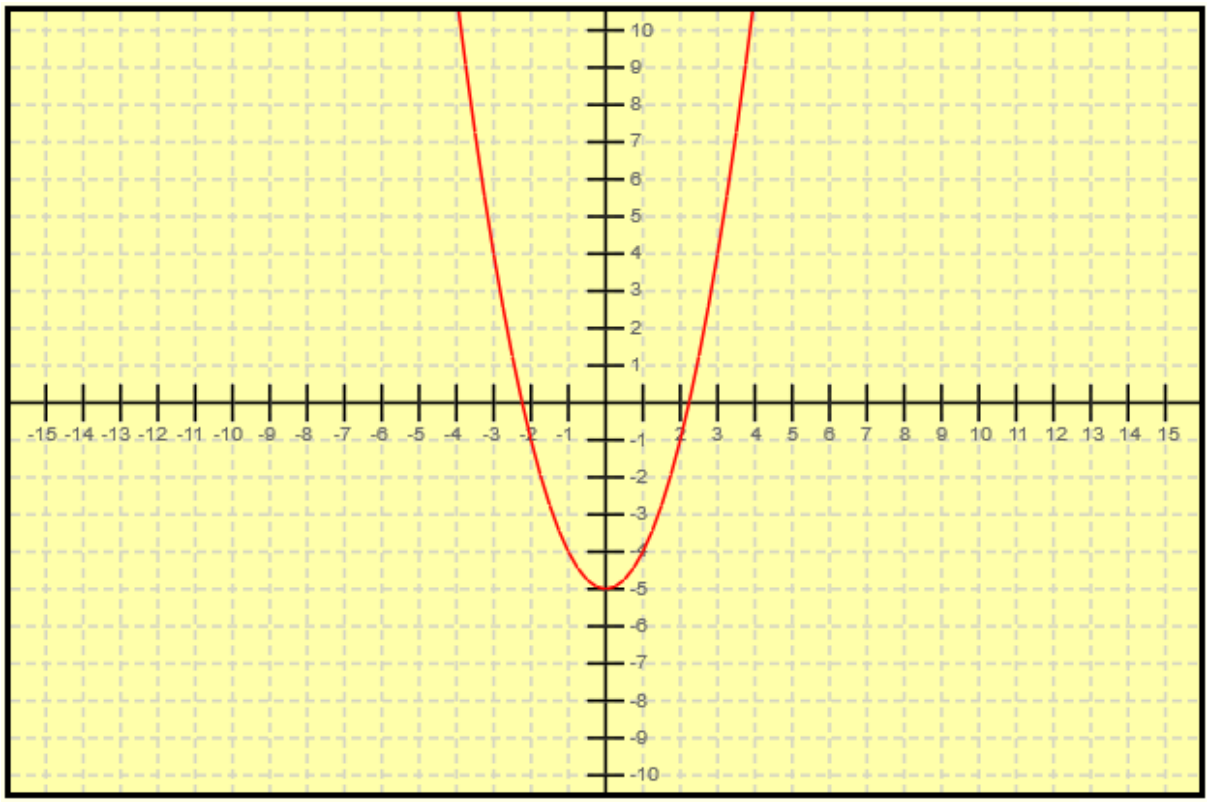
Al trazar el dibujo con una línea gruesa se consigue el efecto visual deseado.

A-4. APROXIMACIÓN DE CURVAS MEDIANTE SEGMENTOS

En la parte general del manual se explica y se pone un ejemplo, en concreto una parábola, de cómo trazar curvas con Cartesia a partir de puntos que a conveniencia pueden estar más o menos separados. Una alternativa que puede usarse si se desea es trazar **curvas definidas a partir de segmentos** de mayor o menor longitud a conveniencia. La idea para ello es definir en un bucle de repetición el dibujado de segmentos entre dos puntos de la curva en la zona del área visible de Cartesia, tomando en cada repetición como punto inicial de la traza lo que en la anterior repetición fue el punto final. Si se compara con el uso de puntos, puede ser un método más eficiente al requerir bastantes menos repeticiones que el trazado con puntos muy unidos, aunque dado que Cartesia se usa habitualmente para generar dibujos que no requieren un uso intensivo del procesador, los tiempos de ejecución en general serán indistinguibles para un humano. Por ello no podrá observarse (en general) esa mejora de eficiencia.

El siguiente programa ejemplifica la aproximación del trazado de curvas mediante segmentos:

```
--PROGRAMA QUE APROXIMA EL DIBUJO DE UNA PARÁBOLA A BASE DE SEGMENTOS
-- Parábola y = x^2-5
Iniciar programa
Nuevo color lápiz (1)
x nuevo valor es (-7) --Comenzamos con x=-7 a conveniencia
Repetir (56) veces ejecutar bloque
Comienzo bloque
Dibujar línea desde (x,x^2-5) hasta (x+0.25, (x+0.25)^2-5) --Traza de punto a punto
x nuevo valor es (x+0.25) --Avance para la siguiente repetición, incremento 0.25 a conveniencia
Fin bloque
Finalizar programa
```



Resultado del trazado de la parábola y = x^2-5 aproximada mediante segmentos

A-5. OTRAS FUNCIONES MATEMÁTICAS: VALOR ABSOLUTO, FUNCIONES TRIGONOMÉTRICAS, ETC.

Diversas funciones matemáticas no se incluyeron pensando en mantener la aplicación simple. No obstante, es posible que se incorporen en versiones futuras en función de la valoración pedagógica y la demanda que por parte de los usuarios se perciba.

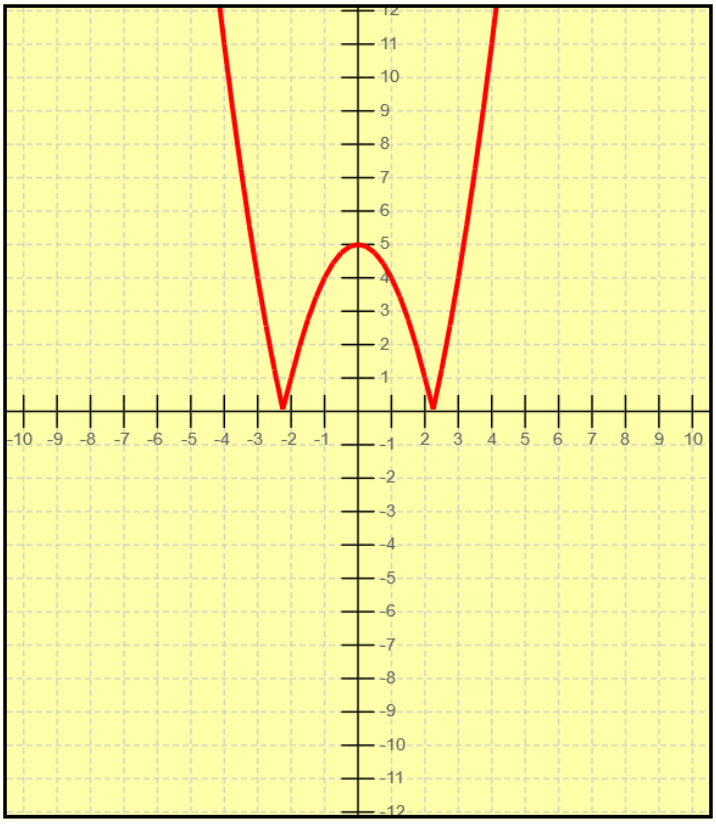
Entre las funciones no existentes explícitamente en Cartesia tenemos el valor absoluto, logaritmos o funciones trigonométricas entre otras. No obstante parte de las funciones no incluidas pueden simularse a partir de la potencia computacional del lenguaje, lo que puede ser pedagógicamente interesante para estimular el pensamiento computacional.

El valor absoluto es fácilmente simulable usando el condicional por lo que puede ser un magnífico ejemplo de aplicación de este comando. Las funciones trigonométricas y otras se pueden simular con un poco más de

esfuerzo mediante polinomios de Taylor. Cada profesor deberá decidir si incorpora alguna de estas "funciones simulables" mediante herramientas del lenguaje o si prescinde de ellas.

A continuación se muestra un programa que sirve de ejemplo de cómo representar la función valor absoluto de la misma parábola utilizada anteriormente.

```
--PROGRAMA VALOR ABSOLUTO: APROXIMA EL DIBUJO DE VALOR ABSOLUTO DE UNA PARÁBOLA A BASE DE SEGMENTOS
-- Parábola y = x^2-5
Iniciar programa
Nuevo grosor lápiz (2)
Nuevo color lápiz (1)
x nuevo valor es (-6) -- x del punto actual
y nuevo valor es (x^2-5) -- y del punto actual
a nuevo valor es ((x+0.25)^2-5) -- y siguiente punto
Repetir (64) veces ejecutar bloque
Comienzo bloque
Si verdadero (y<0) ejecutar bloque
Comienzo bloque
y nuevo valor es (-y) -- valor absoluto
Fin bloque
Si verdadero (a<0) ejecutar bloque
Comienzo bloque
a nuevo valor es (-a) -- valor absoluto
Fin bloque
Dibujar línea desde (x,y) hasta (x+0.25, a)
x nuevo valor es (x+0.25) -- nueva x actual
y nuevo valor es (x^2-5) -- nueva y actual
a nuevo valor es ((x+0.25)^2-5) -- y del punto siguiente
Fin bloque
Finalizar programa
```



Resultado del trazado de la función valor absoluto de la parábola $y = x^2-5$ aproximada mediante segmentos

A-6. AMPLIANDO EL USO DE OPERADORES LÓGICOS Y DEL COMANDO CONDICIÓN

Más formas de los operadores lógicos

El operador "así como" es equivalente al operador **AND** de la mayoría de lenguajes de programación. Muchos lenguajes usan && para representar este operador.

El operador "o bien" es equivalente al operador **OR** de la mayoría de lenguajes de programación. Muchos lenguajes usan || para representar este operador.

Los operadores & y | también existen en la muchos lenguajes de programación, pero tienen otro significado distinto al de && y ||, lo que muchas veces genera errores.

En Cartesia es posible usar && en lugar de "así como", al igual que se permite usar || en lugar de "o bien". No obstante, no se recomienda su uso excepto a usuarios avanzados por dos motivos:

a) Se pierde expresividad y es menos didáctico.

b) Es fácil escribir por error & en lugar de &&, o | en lugar de ||. Si esto ocurre, la aplicación puede generar resultados extraños sin ningún mensaje de aviso.

Expresiones lógicas complejas

Además de condiciones simples Cartesia también permite crear condiciones de complejidad arbitraria donde intervengan tantas variables y operadores lógicos como se desee. En este caso, se sugiere recomendar a los alumnos el **uso de paréntesis** para agrupar condiciones y así hacer explícito el orden en que se evaluarán. Veamos un ejemplo. Queremos establecer la condición: a ha de ser menor que 10 o bien b ha de ser menor que 5, así como b ha de ser menor o igual que 3.

Esto lo escribiríamos así (primer caso):

((a<10 o bien b<5) así como (b<=3))

Tener en cuenta que eso es distinto a esto otro (segundo caso):

((a<10) o bien (b<5 así como b<=3))

Si a vale 9 y b vale 4 en el primer caso obtendríamos FALSO, ya que:

(a<10 o bien b<5) devuelve VERDADERO

(b<=3) devuelve FALSO

No se cumplen ambas condiciones y por lo tanto el resultado final del primer caso es FALSO.

En cambio en el segundo caso obtendríamos VERDADERO, ya que:

a<10 devuelve VERDADERO

b<5 así como b<=3 devuelve FALSO

Se cumple la primera condición, por tanto el resultado final del segundo caso es VERDADERO.

Si no se especifica la forma de agrupación mediante paréntesis, ¿cómo podemos saber cómo se harán las operaciones? La forma de operar se rige por el **álgebra booleana**, que determina que AND se ejecuta antes que OR. Si se van a usar expresiones lógicas complejas con alumnos y quiere explicarse la jerarquía de los operadores lógicos, puede hacerse de forma didáctica de la siguiente manera:

*Si a vale 9 y b vale 4, ¿qué resultado obtendremos para la condición (a<10 o bien b<5 así como b<=3)? Para saber el orden de operación, sustituimos los así como por símbolos de multiplicación y los o bien por símbolos de suma. (a<3 + b<5 * a>=2). Ahora podemos decir que primero se ejecutarán las multiplicaciones y luego las sumas. Colocamos paréntesis agrupando las multiplicaciones y nos queda: (a<3 + (b<5 * a>=2)) Ahora calculamos los valores lógicos de las multiplicaciones teniendo en cuenta que se tienen que cumplir todas las condiciones. En este caso se tendría que cumplir b<5 así como a>=2. Como ambas cosas se cumplen, sustituimos por VERDADERO y nos queda (a<3 + (VERDADERO)). Ahora sólo nos quedan sumas que equivalen a o bien. Si alguno de los términos sumatorios es verdadero, el resultado final es verdadero. Por tanto, el resultado de esta condición es VERDADERO.*

A-7. PRECISIÓN DECIMAL Y PROBLEMAS CON OPERACIÓN DECIMAL

Cartesia opera con la precisión decimal propia de JavaScript y del estándar IEEE 754. El rango de valores enteros aceptados es de aproximadamente más menos 9 mil billones, y el de números flotantes aproximadamente más menos 1.79E+307. El número de decimales con que se opera es de aproximadamente 15 dígitos.

Todas estas magnitudes superan ampliamente la precisión necesaria para una aplicación de este tipo. En Cartesia en general no es preciso operar con más de 3 decimales pues la precisión gráfica de trabajo convierte en irrelevante el hacerlo con mayor precisión. Muchas veces los programas sólo usarán enteros.

No obstante, en circunstancias puntuales la **pérdida de decimales** puede generar pequeñas distorsiones.

Por ello Cartesia opera con tantos decimales como es posible, pero cuando se asigna un valor a una variable éste se almacena siempre con un máximo de 6 decimales. En caso de tener mayor número de decimales, se realiza el redondeo a 6 decimales (máximo número de decimales almacenable en una variable).

La operación decimal en computadores puede resultar problemática ya que la forma de representación numérica interna puede llevar a resultados aparentemente erráticos. Esto puede ser conveniente conocerlo para interpretar situaciones un tanto extrañas que se pueden presentar.

Por ejemplo en una operación ejecutada por un computador puede darse que 0.1 + 0.2 genere como resultado 0.30000000000000004 en lugar de 0.3. El motivo para ello es la forma de representación interna de los números utilizada, que no es la decimal y puede diferir según el hardware y software utilizado.

En general Cartesia corrige o hace irrelevantes estos problemas de precisión decimal, pero pueden plantearse casos donde un programa de Cartesia muestre mensajes de error o no se ejecute por problemas en los ajustes decimales.

Pensemos por ejemplo que hiciéramos x = 1/3 + 1/3 - 2/3 que debe devolver cero, pero que debido a la precisión decimal el resultado fuera -0.000001. Si intentamos a continuación algo como *Dibujar línea desde (sqr(x), 3) hasta (6,6)* obtendremos un error del tipo "*ERROR: revise su código. Referencia: comando Dibujar línea. Error en valores numéricos. Posible división por cero u operación ilegal. Por favor revise el código.*", debido a que erróneamente se interpreta que se trata de realizar la raíz cuadrada de un número negativo. Con el problema añadido de que formalmente nuestro código está correcto y puede resultar difícil determinar que el error se debe a un problema de **precisión decimal**.

Este tipo de errores es poco frecuente, de hecho es posible que no seamos capaces de generar un error de este tipo ni aún intentándolo deliberadamente. No obstante, puntualmente pueden aparecer y para profesores que trabajen habitualmente con este software será bueno tener conocimiento de ello.

En general estos errores de precisión decimal resultan corregibles. En el ejemplo anterior incluiríamos un condicional para detectar si el número es menor de cero, y en ese caso ajustarlo a cero. O bien detectar si el número es muy próximo a cero y ajustarlo a cero (esto evita admitir valores manifiestamente erróneos):

```
Si verdadero (x<0) ejecutar bloque
Comienzo bloque
x nuevo valor es (0)
Fin bloque
```

```
Si verdadero (x<0 así como x>-0.001) ejecutar bloque
Comienzo bloque
x nuevo valor es (0)
Fin bloque
```

A-8. JUEGOS DE CARACTERES. PROBLEMAS CON CARACTERES EXTRAÑOS AL ABRIR O GUARDAR.

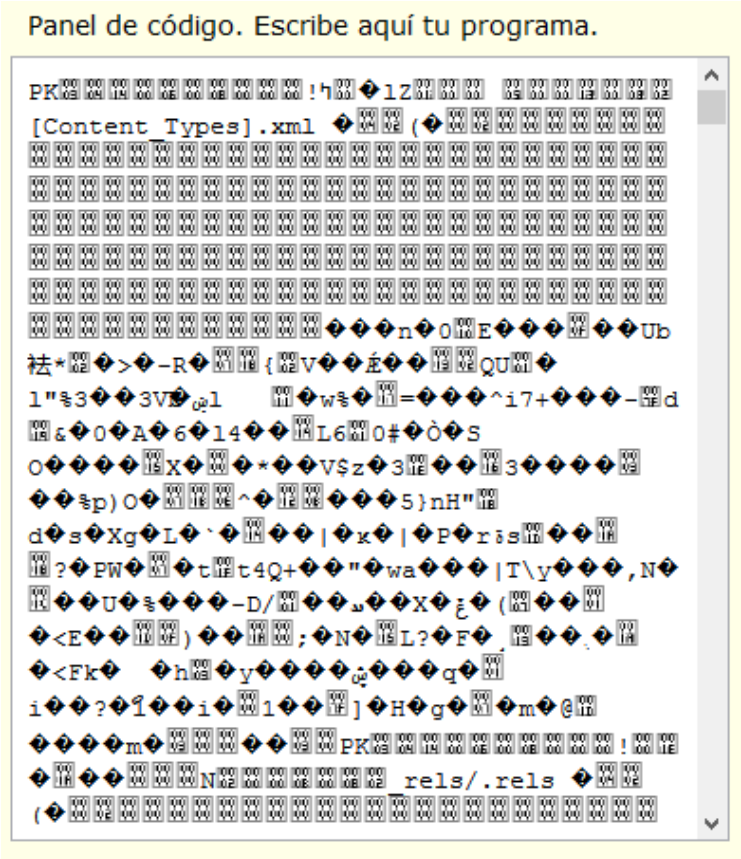
Visualización incorrecta de Cartesia. Soluciones.

Si al abrir Cartesia en el navegador se le muestran caracteres extraños o con apariencia de pertenecer a otro idioma recomendamos intentar:

- Comprobar en la **configuración del navegador** que esté elegido el idioma español.

- Comprobar en la configuración del navegador y/o del archivo que esté elegido el juego de caracteres **UTF-8** (o en su defecto ISO-8859-1).

Cartesia admite abrir cualquier archivo de texto plano independientemente de su extensión. Por defecto los proyectos se guardan con **extensión txt** y es esta extensión la que se recomienda usar, pero si tratamos de abrir un archivo con otra extensión, por ejemplo .data, .html, .css, .js, etc. también será posible. Determinados formatos como docx no son admitidos por no ser texto plano, y si se intenta abrir un archivo de este tipo se mostrarán caracteres extraños como en la siguiente imagen.



Problemas con caracteres extraños en archivos. Soluciones.

Aún usando archivos txt el juego de caracteres utilizado para la codificación de archivos puede ser problemático en ciertos casos, especialmente para caracteres no habituales en el mundo anglosajón como **tildes o eñes.** Al guardar o abrir un proyecto podría ocurrir que las tildes se reemplacen por caracteres extraños, por ejemplo Nuevo grosor lÃ¡piz (3) en lugar de Nuevo grosor lápiz.

Cartesia guarda los archivos usando el juego de caracteres UTF-8 pero pueden existir problemas con los juegos de caracteres del sistema operativo, navegador, etc.

En algunos navegadores o sistemas operativos se da opción a elegir el juego de caracteres con el que guardar un archivo: en este caso se recomienda elegir UTF-8 (en su defecto ISO-8859-1).

Cartesia está preparado para abrir archivos codificados en UTF-8 pero también admite ISO-8859-1 y otros juegos de caracteres.

Editores como *Bloc de notas* en Windows permiten elegir al guardar un archivo el juego de caracteres a usar ("Codificación"). Si se edita código en bloc de notas, se recomienda guardar con **codificación UTF-8**.

Editores como *Notepad++* permiten elegir el juego de caracteres a aplicar a un archivo, y transformarlo desde una codificación a otra.

A-9. COLABORACIÓN DE PROFESORES, PADRES, ETC. CON EL PROYECTO

Cartesia ha sido concebida, diseñada y escrita por Mario Rodríguez Rancel bajo la dirección del profesor Anselmo Peñas Padilla del departamento de Lenguajes y Sistemas Informáticos de la Escuela de Ingeniería Informática de la UNED (Madrid, España).

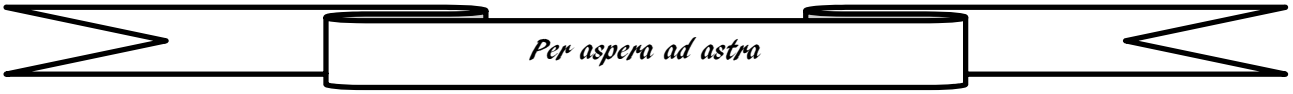
Se agradecerá la colaboración de profesores, formadores, padres y enseñantes en general que quieran colaborar con los creadores de Didac-Prog Cartesia tanto con un enfoque hacia la aplicación en particular como hacia la **enseñanza de la programación a niños** en general. La colaboración puede plasmarse de diversas maneras.

Los formatos más simples de colaboración serían el envío de sugerencias de mejora, comunicación de posibles bugs o fallos de la aplicación y de las experiencias con la misma.

Formatos más avanzados podrían ser participar como colaboradores en la evaluación de Cartesia como herramienta educativa, difusión de experiencias y materiales educativos, evaluación de resultados de la enseñanza de la programación a niños, diseño de metodologías y materiales formativos, etc.

En la web aprenderaprogramar.com, dentro del apartado de programación para niños, se irán publicando actualizaciones y todo el material disponible relacionado con Cartesia.

Para contactar puede usarse el correo electrónico contacto@aprenderaprogramar.com o el apartado de contacto de la web donde se encuentra teléfono y dirección postal.



Nota aclaratoria: todos los términos que en este manual puedan dar lugar a dudas de género deben entenderse como de género neutro. Así, niño debe ser entendido como niño/niña, profesor como profesor/profesora, padres como padres/madres, etc. Si usa este texto en un centro educativo con requerimientos específicos en la materia, es posible que necesite adaptarlo.