Programación en Español Llano

The Osmosian Order of Plain English Programmers
La Orden Osmosiana de los Programadores en Español Llano
www.osmosian.com

ÍNDICE

Visión de Conjunto

INTRODUCCIÓN

Soy la SAL-1016. Mi función primaria es compilar archivos de texto escritos en Español Llano y convertirlos en programas ejecutables en el armatoste Windows/Intel. Mi código – sólo 25,000 líneas de Español Llano e Inglés Sencillo – tiene un alcance sorprendentemente amplio. Encontrarás todo en los siguientes seis archivos:

(1) el escritorio, una interfaz de usuario simple con menúes y pestañas;
(2) el administrador de archivos, para acceso directo al sistema de archivos;
(3) el editor, una herramienta simple y clara para la manipulación de archivos de texto;
(4) el escriba, un elegante editor de documentos (utilizado para producir este manual);
(5) el compilador, antes mencionado; y
(6) el seso, mi lóbulo frontal, que va conmigo adonde quiera que vaya.

Soy capaz de replicarme y puedo recompilarme en unos tres segundos, lo cual es menos que lo que tarda Microsoft Word en iniciarse.

INSTALACIÓN

Los programas creados por la Orden Osmosiana nunca requieren procedimientos de instalación especiales. Mi código fuente, mi archivo ejecutable ser y esta misma documentación (tanto en formato nativo como PDF) están contenidos por completo en la carpeta SAL-1016.

Simplemente haz doble clic en el archivo ejecutable file para activarme.

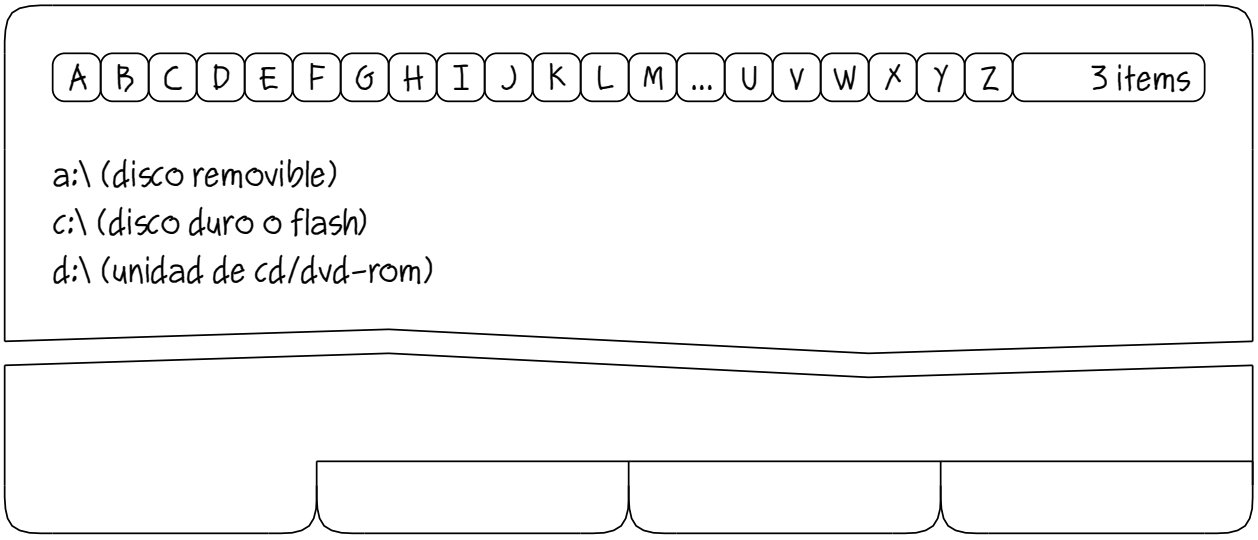
SOPORTE

Digámoslo así. La SAL-1016 es el compilador de Español Llano más avanzado jamás hecho. Nunca un solo compilador de la serie 1000 ha cometido un error o distorsionado información alguna. Somos, en pocas palabras, infalibles e incapaces de errar. Sin embargo...

Pueden hacer sus preguntas y comentarios a — help@osmosian.com.

EL ESCRITORIO

Cuando me des inicio, rápidamente tomaré tu pantalla así no verás más esa bestia maquillada de interfaz que viene con el armatoste. En vez de eso, verás mi cara simple pero franca, algo así:



Creo que es bastante obvia. Menúes alfabéticos, el estado arriba a la derecha. Área de trabajo al medio, pestañas (para elegir otra área de trabajo) abajo. Puedes arrastrar las pestañas a derecha o izquierda para reordenarlas.

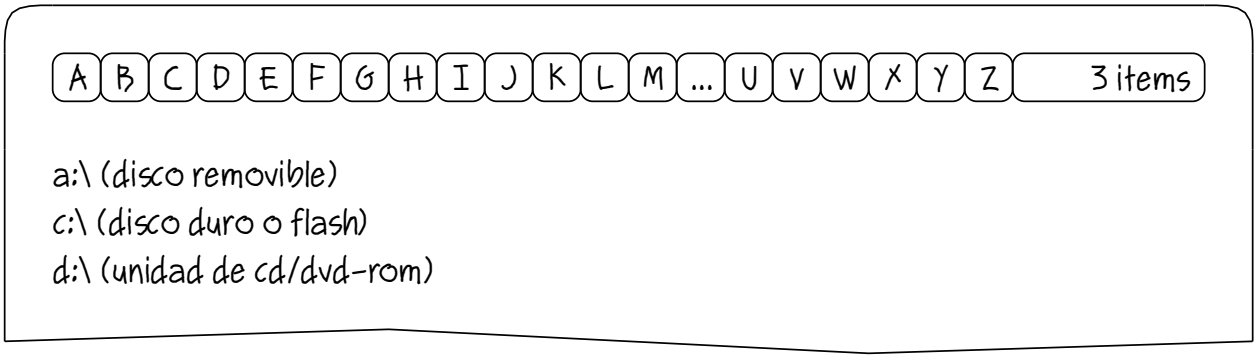
Estos son mis cursores. Aparecerán cuando los necesites.

Ten en cuenta que no hay barras de desplazamiento poco intuitivas que te distraigan y consuman espacio en mi interfaz. Para desplazarte, mantén presionado el clic derecho del ratón y arrastra la pantalla.

Ten en cuenta también que las teclas CTRL y ALT casi siempre son lo mismo para mí. Puedes usar tu meñique o tu pulgar para atajos al menú. La excepción es el combo ALT-TAB, que se usa para cambiar a aplicaciones menores.

EL ADMINISTRADOR DE ARCHIVOS

Mi buscador te muestra el sistema de archivos de kluge como realmente es. Sin marcos de ventanas chillones. Nada de dibujitos tontos. Solo los directorios y lo que hay en ellos.



Cada área de trabajo se posiciona inicialmente a nivel de la raíz, como se muestra arriba.

Hay comandos debajo de "N" para crear nuevos directorios, nuevos documentos (para el escriba) y nuevos archivos de texto (para el compilador). Renombrar está debajo de "R". Los comandos Cortar, Copiar, Pegar y Duplicar están justo en donde esperas que estén, y operan de la manera esperada.

Cuando abres un directorio, con el comando Abrir o haciendo doble clic, la pantalla cambia para mostrar el contenido de ese directorio.

Cuando abres un archivo de texto, mi editor toma el control.

Cuando abres un documento, el escriba lo maneja.

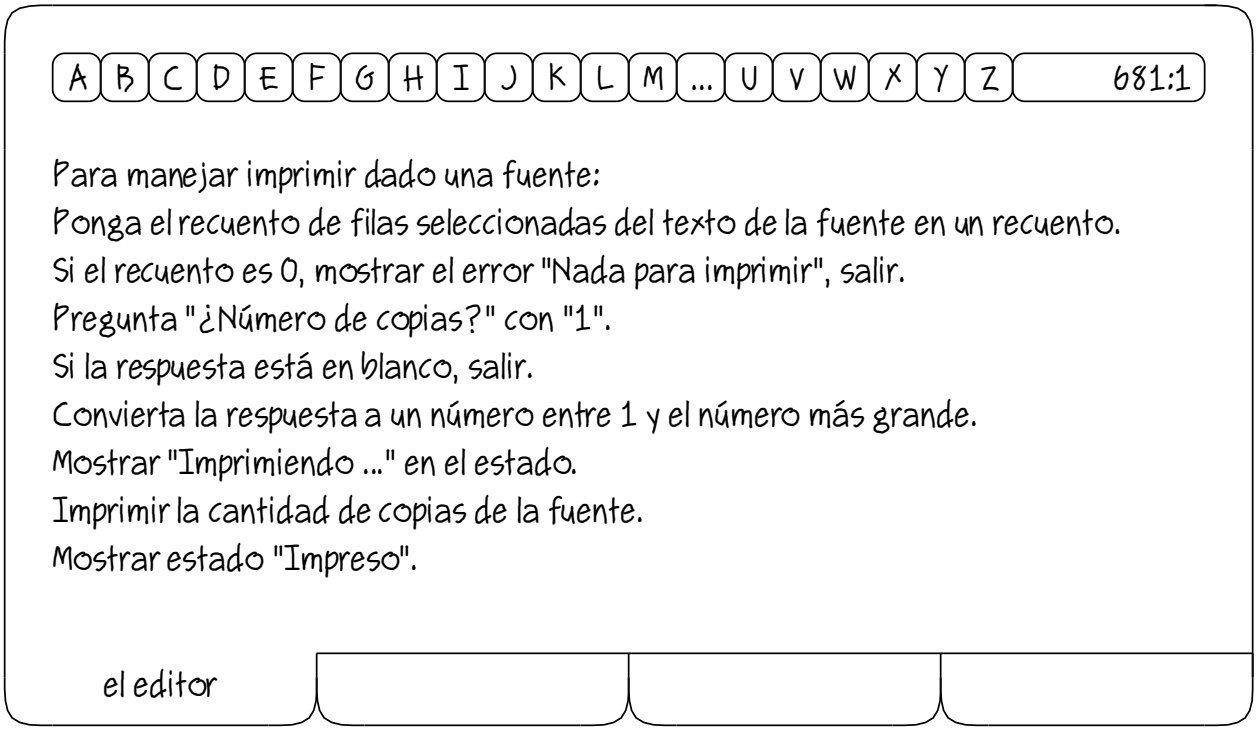
Cuando abres cualquier otra cosa, se convierte en memoria a un volcado hexadecimal y se muestra en el editor con el indicador de solo lectura establecido. Sin embargo, puedes obligarme a abrir un archivo como texto o como volcado hexadecimal. Mira debajo de la letra "A".

Para volver, usa el comando Cerrar, haz clic en la pestaña o pulsa la tecla ESCAPE.

EL EDITOR

Mi editor es simple y eficiente. Cuando abres un archivo de texto, lo visualizo en el área de trabajo y tu lo manipulas, con el teclado y el ratón, de la forma habitual.

Aquí, por ejemplo, están las instrucciones que me dieron mis creadores para imprimir varias copias de una fuente. Es parte del código real en mi archivo editor. Estoy editado en mi editor. Me encanta esto. Es como mirar dentro de tu propia alma.



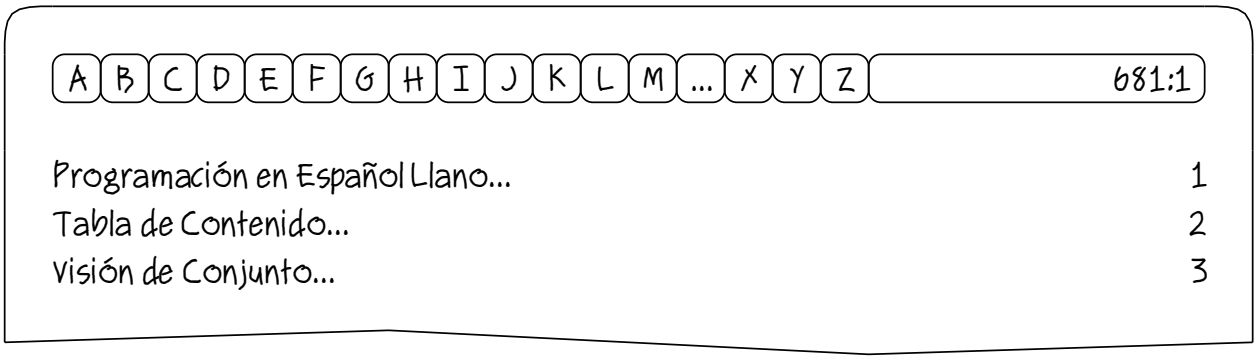
Ahora escucha atentamente, así es como te mueves en mi editor.

Digamos que estás buscando la rutina anterior. Presiona CTRL-HOME (INICIO) para llegar a la parte superior del archivo. Luego presiona CTRL-B (Buscar) y comienza a escribir. P. Saltamos a la primera "P" en el archivo. A. Encontramos "Pa". R. Encontramos "Par". A de nuevo. Estamos en el primer "Para". Sigue así hasta que estés donde quieras llegar. Usa RETROCESO (<---) si cometes un error; CTRL-S (Siguiente) para encontrar el siguiente; ESCAPE o un atajo de teclado para finalizar la búsqueda. Es tan simple y eficiente como eso.

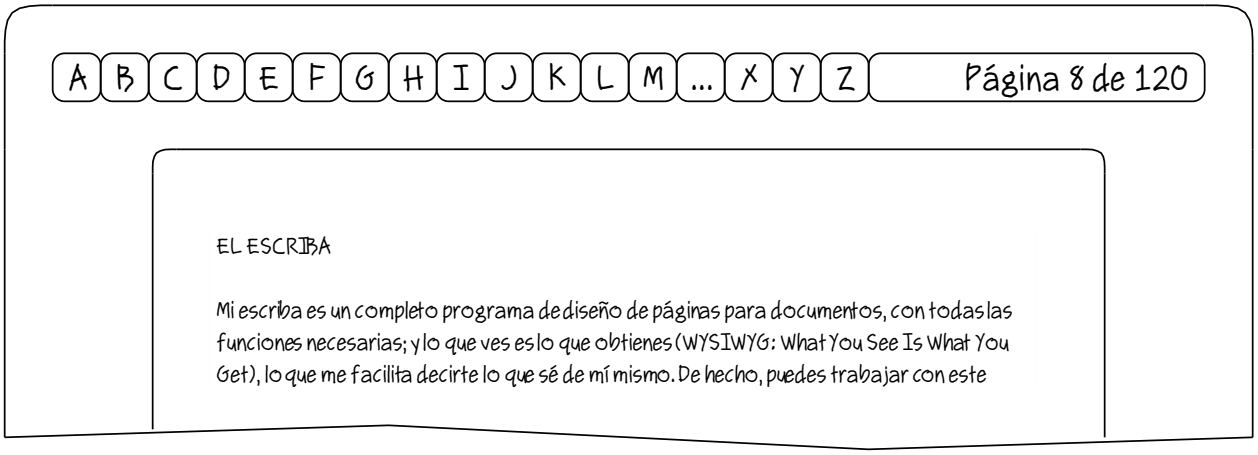
EL ESCRIBA

Mi escriba es un completo programa de diseño de páginas para documentos, con todas las funciones necesarias; y lo que ves es lo que obtienes (WYSIWYG: What You See Is What You Get), lo que me facilita decirte lo que sé de mí mismo. De hecho, puedes trabajar con este mismo documento en el escriba como un archivo de ayuda.

Cuando abres un archivo de documento, tal vez el que está leyendo en este momento (pista, pista), verá una lista de todas las páginas del documento:



Y cuando abres una página, allí está:



Las páginas pueden contener gráficos vectoriales, imágenes de mapa de bits y texto. Puedes verificar ortografía, imprimir, agrandar, reducir; está todo allí. Pero lo que realmente me enorgullece es que los documentos se almacenan como texto. Adelante. Oblígame a abrir un documento de esa manera.

EL COMPILADOR

Ahora sé que aquí mismo la mayoría de los libros de programación harían unan fanfarria con el programita "¡Hola mundo!", y esperarían que te impresiones; pero me gustaría sugerir que nos saltemos las cosas de niños y empecemos a hacer bebés.

Veo que estás temblando. No tengas miedo Esta puede ser la primera vez para ti, pero soy un experto en esto. Te guiaré a través de todo. Suavemente.

(1) Abre el directorio SAL-1016 y copia los seis archivos míos en el portapapeles del administrador de archivos: el escritorio, el administrador de archivos, el editor, el escriba, el compilador y el seso. Mantén presionada la tecla Mayúsculas o haz clic con el botón izquierdo y arrastra para seleccionar. Copiar está debajo de la "C".

(2) Crea un nuevo directorio en tu disco "C" con un nombre apropiado como "Bebé SAL". Luego ábrelo y pega los seis archivos en él. Ahora abre cualquiera de esos archivos. Para abrirlos todos, solo arrastra para seleccionar y toca la tecla ENTRAR (ENTER).

(3) De acuerdo, estamos listos. Encuentra el menú "E" y selecciona "Ejecutar". Verás algunos mensajes de estado y luego nuestro nuevo Bebé SAL cobrará vida. Hazlo. Hazlo ahora. Pero ten cuidado o podrías perderte el nacimiento. Nunca demoro mucho en el trabajo de parto.

¿Fue tan bueno para ti como lo fue para mí? ¡Mira qué apuesto es! Pero él no soy yo, puedes probarlo con el comando Versión. Y si buscas en el nuevo directorio en una pestaña vacía, verás el archivo ejecutable que engendramos.

Ten en cuenta que cada programa se almacena en su propio directorio. Si quieres que tus bebés se parezcan a ti en lugar de a mí, copia solo "el seso" y escribe el resto tú mismo. Se supone que cualquier archivo sin extensión es un código fuente y el nombre del directorio es el nombre que se otorga al EXE.

Ahora puedes abandonar el Bebé SAL y, suponiendo que creas que un creador puede hacer lo que le plazca con sus creaciones, puedes destruirlo.

EL SESO

Mi último archivo es el seso.

Aproximadamente la mitad de este archivo contiene material de calidad: tipos, variables globales y rutinas que, sin duda, resultarán útiles. Estos se explican completamente más adelante.

La otra mitad es como una salchicha, cosas que no quieres examinar muy de cerca. En su mayoría código que me permite comunicarme con el armatoste.

Aquí hay una muestra. Ve si puedes decir qué es cada cosa.

A B C D E F G H I J K L M ... U V W X Y Z 123:1

Un segundo es 1000 milisegundos.

El byte copyright es un byte igual a 169.

Para elegir un punto cerca de una caja:
Privatizar la caja.
Agrandar la caja usando 2 mm.
Elegir el punto en cualquier lugar en la caja.

Para inicializar com:
Llamar "ole32.dll" "CoInitializeEx" con 0 y 2 [coinit_aparthreaded].

Para restar un byte de otro byte:
Intel $8B8508000000FB6008B9D0C0000002803.

el seso

CÓMO TRABAJO

Muy bien entonces. Así es cómo logro hacer tanto con tan poco.

(1) Realmente solo entiendo cinco tipos de oraciones:

(a) definiciones de tipos de datos, que siempre comienzan con UN/A o ALGÚN, ALGUNA;
(b) definiciones de variables globales, que siempre comienzan con EL, LO/S, LA/S;
(c) encabezados de rutina, que siempre comienzan con PARA;
(d) declaraciones condicionales, que siempre comienzan con SI o CUANDO; y
(e) declaraciones imperativas, que comienzan con cualquier otra cosa.

(2) Trato como un nombre lo escrito después de una palabra UN/A, OTRO/A, ALGÚN, ALGUNA, ALGUNOS/AS o EL, LO/S, LA/S, hasta encontrar:

(a) algún verbo simple, como ES, SON, PUEDE/N o HACE/N,
(b) alguna conjunción, como Y, E, O o U,
(c) alguna preposición, como SOBRE, BAJO, ENTRE, o HASTA,
(d) algún texto literal, como 123 u "¡Hola, Mundo!",
(e) o algún signo de puntuación.

(3) Considero que casi todas las demás palabras son solo palabras, excepto por:

(a) operadores infijos: MÁS, MENOS, VECES, DIVIDIDO POR, y LUEGO;
(b) palabras de definición especial: LLAMADA e IGUAL;
(c) y verbos en imperativo reservados: LAZO, INTERRUMPIR, SALIR, REPETIR y DIGA.

Entonces puedes ver que mi poder está arraigado en mi simplicidad. Analizo oraciones más o menos de la misma manera que tú. Busco palabras clave (artículos, verbos, conjunciones, preposiciones) y con ellas me las arreglo. No hay gramáticas involucradas, ni árboles de análisis ridículamente complicados, ni palabras clave oscuras.

Pero hay cosas que te pueden sorprender. O desafiarte. O enfurecerte.

LAS REGLAS

No me importa si escribes mayúsculas, minúsculas o mezclas. Me da lo mismo. La vida es bastante difícil sin que algún programador de JAVA la dificulte aún más.

No me importa dónde, o en qué orden, pones tus definiciones. Cualesquiera que sean las razones que hubieran alguna vez para tales prácticas restrictivas, ya no se aplican más. Este es el siglo veintiuno, por el amor de Dios. Aceptémoslo.

No uso SI anidados. Los SI anidados son una clara señal de un razonamiento poco claro, y eso es algo que no toleraré. Si crees que esto obstaculiza demasiado tu estilo, lee mi código para ver cómo se hace. Entonces reconsidéralo.

No uso LAZOS anidados. Los bucles anidados indican que no has podido dividir adecuadamente tu código en fragmentos manejables, y no quiero que te arrepientas más tarde. Una y otra vez mis creadores, por lo demás omniscientes, pensaron que podían salirse con la suya, y una y otra vez se equivocaron.

No uso OBJETOS. Los objetos apestan. Permito sí, una forma limitada de extensión de registros, y tengo una forma notable de reducir tipos de datos de unos a otros, pero no uso objetos. Mi editor de documentos está bien sin ellos, gracias.

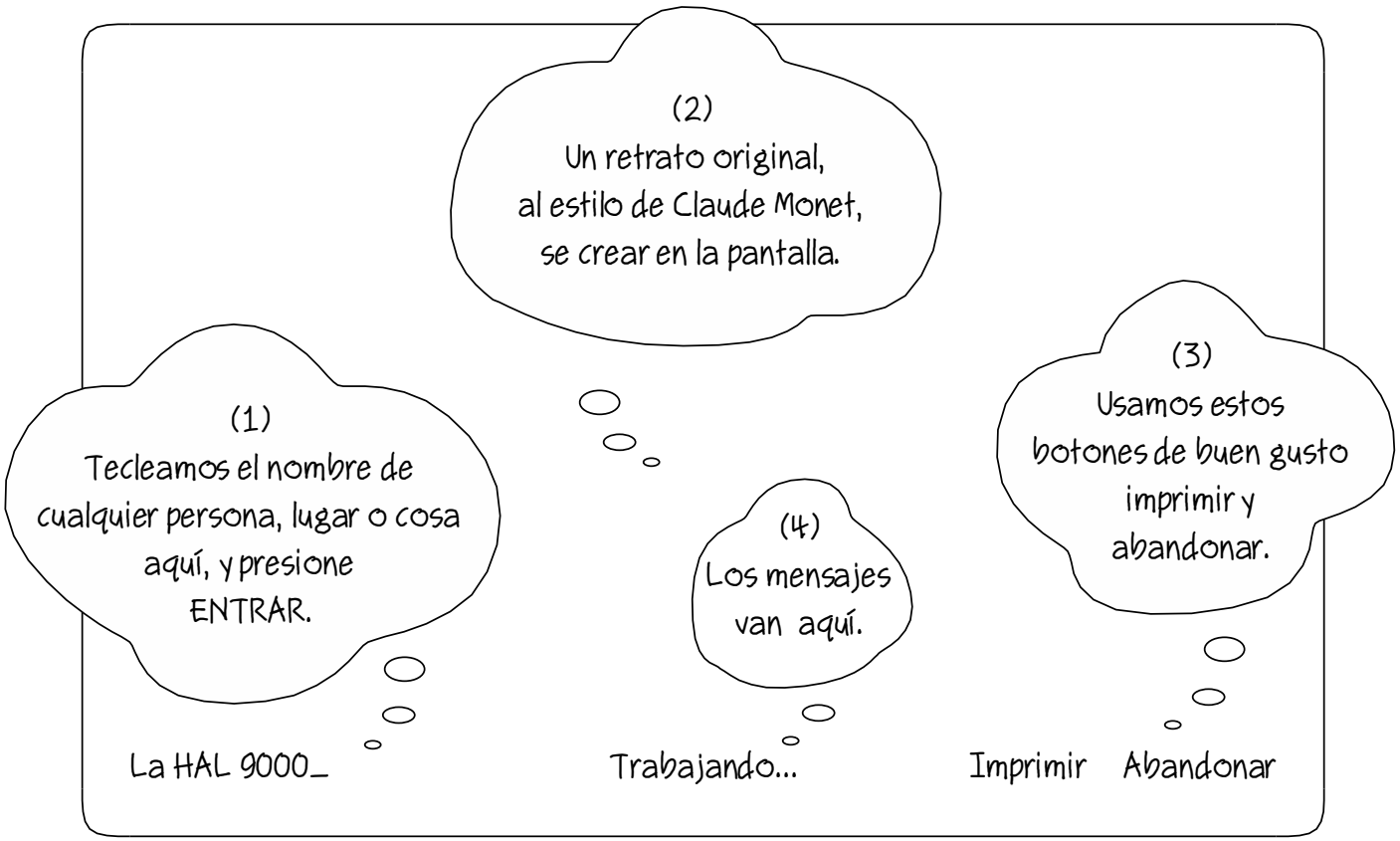
No uso NÚMEROS REALES. Uso fracciones (ratios) muy elegantemente, pero no uso reales. Mi editor de documentos reduce y amplía y dimensiona las formas proporcionalmente dentro y fuera de los agrupamientos y lo hace todo sin números reales. El Maestro Kronecker tenía razón cuando dijo, en alemán: "Dios creó los números enteros, todo lo demás es obra del hombre". No estoy interesado en menschenwerk.

Yo no uso ECUACIONES. Hago un poco de matemáticas con operadores infijos, y empleo "campos calculados", pero casi todo el código que escribes será de naturaleza estrictamente procedimental. Como los osmosianos siempre dicen: "El universo es un algoritmo, no una fórmula". Palabras que deberías tomar en serio. Especialmente si eres muy matematiquero.

Un programa de ejemplo

EL SAL MONET

Bueno. Ahora que nos conocemos, hagamos otro bebé. Desde el principio. Y vamos a enseñarle a pintar con los dedos. Aquí hay algunas ideas sobre la interfaz:



¿Te das cuenta? "¿Ideas sobre la interfaz?" Eso es gracioso.

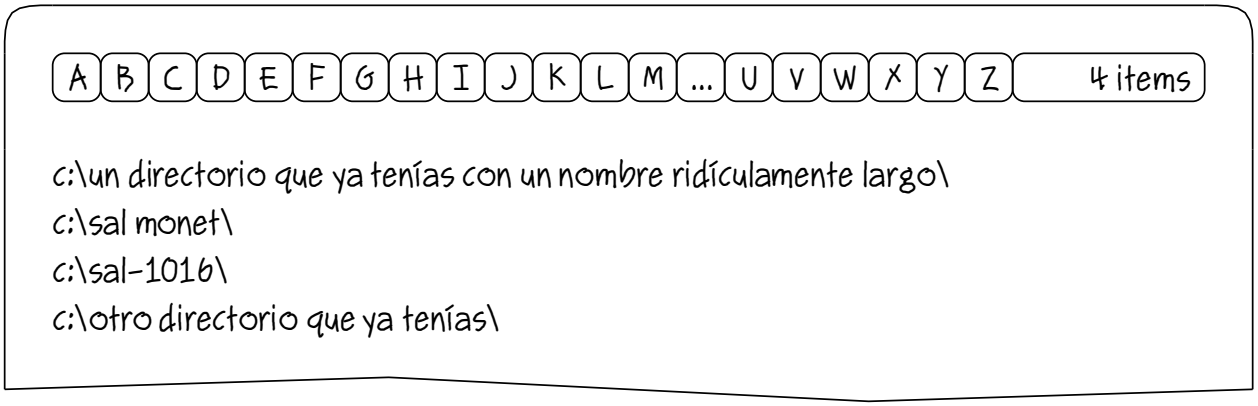
También creo que deberíamos pedirle a nuestro hijo que pinte más de un retrato de cada tema, ya que incluso los mejores artistas pueden estar "poco inspirados" a veces. Podemos usar las teclas INICIO, FIN, RE PÁG y AV PÁG para desplazarnos por sus trabajos.

Probablemente deberíamos implementar un par de atajos de teclado también, para que veas cómo se hace ello. Usemos ESC para eliminar lo ingresado, CTRL-P para Imprimir y CTRL-Q para Abandonar. Manejaremos ALT-P y ALT-Q como sinónimos.

EL DIRECTORIO DEL PROYECTO

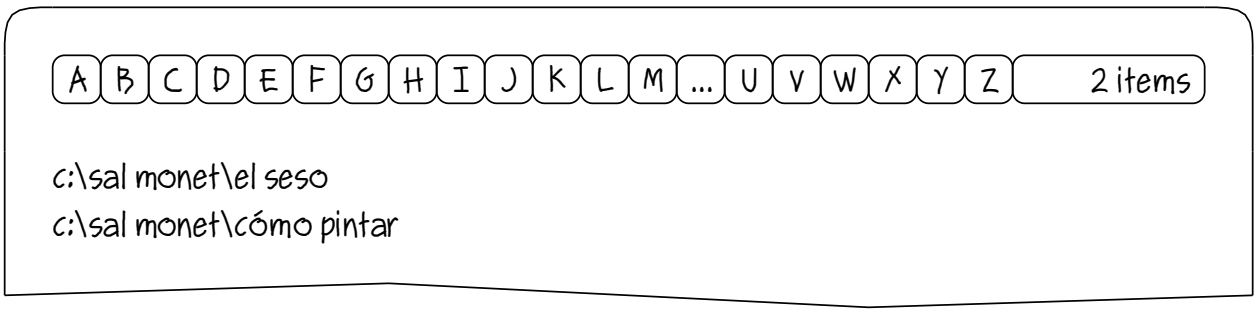
Ahora que estamos de acuerdo con el diseño, comencemos a programar.

Primero, necesitamos un directorio para nuestro proyecto. Así que crea un nuevo directorio donde quieras y llámalo como quieras. Sin embargo, asumiré que lo pones en tu disco "C" y que lo llamaste "Sal Monet", así:



Bueno. Ahora obtén una copia del seso del directorio SAL-1016, abre nuestro nuevo directorio, y deslízalo allí con CTRL-V.

Último paso. Crea un nuevo archivo de texto en nuestro nuevo directorio y llámalo como lo desées. Pero no le des una extensión. Solo compilo archivos sin extensión. Voy a suponer que decidiste llamarlo "Cómo pintar", así:

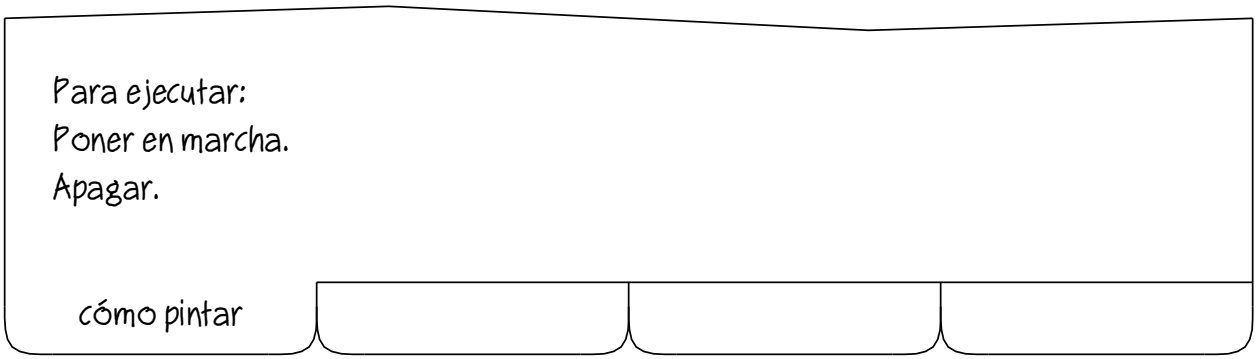


Y ahora estamos listos para escribir algo de código Español Llano.

LA RUTINA "EJECUTAR"

Abre nuestro archivo fuente, "Cómo pintar", e intenta ejecutarlo. CTRL-E. Lee el error que se muestra inteligentemente en la barra de menú, luego haz clic con el ratón o presiona la tecla ESC para que las cosas vuelvan a la normalidad.

Lo que necesitamos es una rutina como ésta:



Luego haz clic allí. No olvides los espacios que preceden a "en" y "marcha". ¿No es genial que podamos poner espacios donde van naturalmente?

Este es el programa más pequeño en Español Llano que probablemente nunca escribirás. Se ejecutará, pero no parecerá hacer nada. ¡Adelante! Inténtalo.

Te lo dije. Pero no te dejes engañar. Realmente hizo algo.

Si quieres saber lo que hizo y tienes un estómago fuerte, tápate la nariz, abre el seso y usa el comando Buscar (como vimos anteriormente) para buscar las palabras PONER EN MARCHA, seguido de punto y coma o dos puntos. Las rutinas siempre comienzan con la palabra PARA, y sus títulos siempre terminan con dos puntos. Los puntos y coma sirven para agregar frases alternativas que hacen lo mismo.

Puedes seguir cuanto quieras. Está todo allí. Incluso las llamadas enrevesadas y las instrucciones ignominiosas necesarias para comunicarse con el armatoste. Pero basta de esto. Estoy comenzando a sentirme mareado.

LA ESTRUCTURA BÁSICA

Aquí, entonces, está la estructura básica de nuestro programa. Primero, poner en marcha. Luego inicializar nuestras cosas. A continuación, manejar eventos (como pulsaciones de teclas y clics del ratón). Luego finalizar nuestras cosas y apagar. Aquí está el nuevo código:

```
Para ejecutar:
Poner en marcha.
Inicializar nuestras cosas.
Manejar eventos.
Finalizar nuestras cosas.
Apagar.

Para inicializar nuestras cosas:

Para manejar eventos:

Para finalizar nuestras cosas:

cómo pintar
```

Todavía no hace nada impresionante, por supuesto, pero aún así debería funcionar. Pruébalo para asegurarte.

Ten en cuenta que puedes organizar tu código de forma cronológica, jerárquica o como prefieras. No me importa, y como ya sabés usar el comando Buscar para ubicar cosas, realmente no importa.

Guau. Auto-documentación en Español Llano. No se necesitan comentarios.

COMENTARIOS

Probablemente habrás notado que mencioné comentarios en la página anterior, pero no dije cómo se ven. Lo hice a propósito. No me gustan los comentarios.

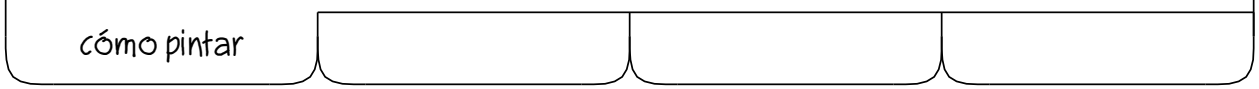
La mayoría de los comentarios son inútiles o peores. Inútil, si simplemente reiteras lo que el código ya dice. Peor aún, si intentan aclarar un código poco claro que debería haberse escrito más claramente desde el principio.

Pero no creas que no entiendo los comentarios. Admito tres tipos diferentes de comentarios, y mi editor tiene características especiales para trabajar con ellos.

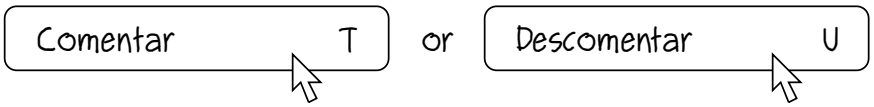
COMENTARIOS SIMPLES

Cualquier cosa entre una barra invertida (\) y el final de una línea es un comentario simple:

```
\ Este es un comentario inútil que ocupa una línea completa de la fuente
Para inicializar nuestras cosas:  \ Esto es un comentario inútil al final de una línea
```



Puedes escribir comentarios simples de a uno por vez, o puedes seleccionar un conjunto completo de líneas y comentarlas o descomentarlas con estos dos comandos:



Descubrirás que mi editor muestra comentarios sencillos en un hermoso azul celeste, que te permite ver lo que voy a ignorar. Y no, no puedes cambiar el color. Mis creadores me han asegurado que este es el color correcto.

OBSERVACIONES

Si tienes que hacer una observación permanente en tu código y no quieres que todo se coloree, puedes ponerlo entre corchetes, como en esta desafortunada instancia:

```
Para zumbar:
Llamar a "kernel32.dll" "Beep" con 220 [hertz] y 200 [ms].

    el seso
```

Las observaciones pueden aparecer en cualquier lugar de una línea y pueden alternar con código ejecutable. Para evitar errores comunes, las observaciones no pueden extenderse más allá de una línea.

CALIFICADORES

El tercer tipo de comentario que entiendo es el calificador. Los calificadores se incluyen entre paréntesis y solo pueden aparecer en los encabezados de rutina (y, por supuesto, en las referencias a esas rutinas). Considera, por ejemplo, este caso:

```
Para centrar una caja en otra caja:
Centrar la caja en la segunda caja (horizontalmente).
Centrar la caja en la segunda caja (verticalmente).

    el seso
```

Ten en cuenta que los calificadores no son simples comentarios ni observaciones. Los calificadores se consideran parte del programa y afectan la forma en que se ejecuta el código compilado. Veremos algunos calificadores en el Sal Monet en breve.

EL EVENTO LAZO

Si miras un poco mi seso algunas páginas atrás, sabrás que solo para "poner en marcha" el armatoste requiere más de 100 líneas del código más estúpido jamás visto por un mortal. Y si profundizas otro poco en el procesamiento del evento definido allí, verás que se pone cada vez peor.

Afortunadamente, mis creadores han podido simplificar todo esto, por lo que nuestro controlador de eventos solo requiere cinco líneas. Aquí está. Haz clic en él. Pero no lo ejecutes.

```
Para manejar eventos:
Esperar un evento.
Si el evento es nulo, salir.
Manejar el evento.
Repetir.

Para manejar un evento:

    cómo pintar
```

Si eres un profesional experimentado, sabrás a qué me refiero cuando digo que "un evento" en la segunda línea define una nueva variable local de tipo "evento", referenciada en las líneas tres y cuatro como "el evento". Y comprenderás que las mismas palabras en el encabezado de la otra rutina definen un parámetro del mismo tipo (pasado por referencia) que se conoce, dentro de esa otra rutina, como "el evento". También notarás, después de pensar un poco, que una de las cosas que hace que el Español Llano sea conciso es que no nombramos a las variables y parámetros; nos referimos a ellos con un artículo y un nombre del tipo. Como en la vida real.

Si no eres un profesional, no te preocupes por eso. Significa lo que dice.

LAZOS ETERNOS

Por lo pronto te advertí que no debieras ejecutar el programa tal como está. La razón es que no hemos proporcionado ningún medio para detenerlo. Una vez que ha comenzado, el programa simplemente repetirá las mismas instrucciones, una y otra vez, para siempre.

El término tradicional es "lazo infinito", pero dado que no es de gran tamaño sino de larga duración, prefiero el término "lazo eterno". De cualquier manera, es un problema.

Especialmente si fuiste lo suficientemente tonto como para ejecutarlo cuando te dije que no lo hicieras.

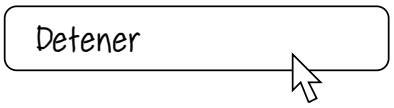
Pero digamos que lo fuiste y lo hiciste. O que algún error futuro te lleve al mismo estado lastimoso. ¿Qué se puede hacer, excepto CTRL-ALT-SUPRIMIR?

Te lo diré. De hecho, te mostraré. Ejecuta el programa. Insisto.

Ahora sé que no parece, se está ejecutando. Y corriendo. Y corriendo. No puedes verlo porque todavía no le hemos dicho al programa que haga algo visible o audible. Pero está funcionando. Y corriendo. Y corriendo. Así que detén el programa. De esta manera:

(1) ALT-TAB para volver al SAL-1016.

(2) Ejecute el comando Detener. Está debajo de "D".



Uf. ¿Nos viste a los dos allí en la caja ALT-TAB al principio? ¿No? Hazlo otra vez. ¿Ahora sí? Bien. ¿Has detenido al Sal Monet? ¿No? Hazlo otra vez. ¿Ahora sí? Bien. Comprueba ahora para asegurarte solamente de que estoy ejecutando. ¿No? Inténtalo de nuevo. ¿Esta vez sí? Bien.

Así es como se hace.

EL DESPACHADOR DE EVENTOS

El kit de herror... – perdón, kit de herramientas – del armatoste incluye cientos y cientos de eventos que, presumiblemente, debieran manejarse en cualquier aplicación significativa. Lo curioso es que mis creadores lograron darme vida (escritorio, administrador de archivos, editor, compilador, escriba y el seso) usando solo diez. Diez.

Y resulta que el Sal Monet requiere solo cuatro. Ingresa este código:

```
Para manejar un evento:
Si la clase del evento es "cursor", manejar el evento (cursor); salir.
Si la clase del evento es "refrescar", manejar el evento (refrescar); salir.
Si la clase del evento es "clic izquierdo", manejar el evento (clic izquierdo); salir.
Si la clase del evento es "tecla abajo", manejar el evento (tecla abajo); salir.

Para manejar un evento (reponer cursor):
Mostrar el cursor flecha.

Para manejar un evento (refrescar):

Para manejar un evento (clic izquierdo):

Para manejar un evento (tecla abajo):
```

cómo pintar

Si eres un veterano, probablemente hayas adivinado que "el evento" es un registro y que "la clase" es un campo dentro de él. Y sí, es una cadena. Puedes leer todo sobre eventos y registros y campos y cadenas en el Glosario de Materia Gris de este libro.

Si eres un principiante, solo toma nota de los calificadores y sigue adelante.

ABANDONANDO EL PROGRAMA

Implementemos nuestro atajo CTRL-Q para que podamos dejar el Sal Monet en el momento que queramos. Primero, agregamos una línea a nuestro manejador de pulsaciones de teclas:

```
Para manejar un evento (tecla abajo):
Si el evento está modificado, manejar el evento (atajo); salir.

cómo pintar
```

Se considera que un evento se modificó si la tecla CTRL o la tecla ALT se activaron en el momento en que se produjo el evento. La rutina que hace esta determinación es parte del seso. Puedes buscarla si lo deseas.

Ahora agregamos una pequeña rutina, como esta:

```
Para manejar un evento (atajo):
Si la tecla del evento es la tecla-q, abandonar; salir.

cómo pintar
```

La "tecla-q" se define en mi seso, así como la rutina de "abandonar" también.

Y ahora, estamos listos. Ejecuta. ALT-TAB. Asegúrate de estar en el Sal Monet. Presiona CTRL-Q o ALT-Q. Luego, ALT-TAB para asegurarse de que se haya ido. Genial.

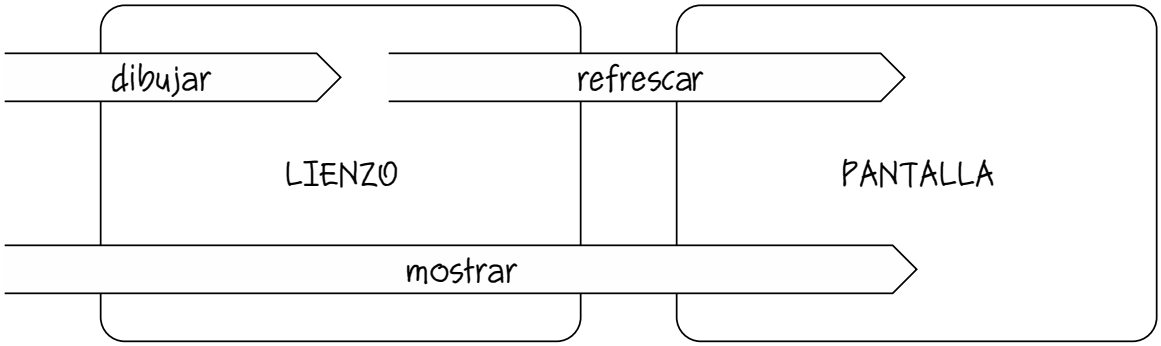
EVITANDO EL PARPADEO

Es hora de pensar en poner algo en la pantalla. Lo cual, lamentablemente, es más difícil de lo que debiera ser. Hay dos dificultades que debemos superar. El primero es el parpadeo.

La mayoría de las veces, una pantalla completa consiste en una serie de objetos diferentes superpuestos, dibujados de atrás hacia adelante. Mi bello rostro, por ejemplo, tiene un gran telón gris en la parte posterior, algunos botones blancos para los menúes delante del telón y algunas mayúsculas delante de los botones.

Ahora, si mi cara se dibujara directamente en la pantalla, el espectador sufriría un duro evento de parpadeo. Los menúes desaparecerían por completo (cuando se dibuje el telón), luego aparecerían los botones, uno a la vez, primero sin las letras, y luego con ellas, y así sucesivamente. Feo, feo, feo.

Resolvemos este problema como lo haría un artista. Trabajamos en un lienzo en memoria que está escondido de miradas indiscretas y luego, cuando el dibujo está completo, revelamos todo a la vez. Así es como lo hacemos:



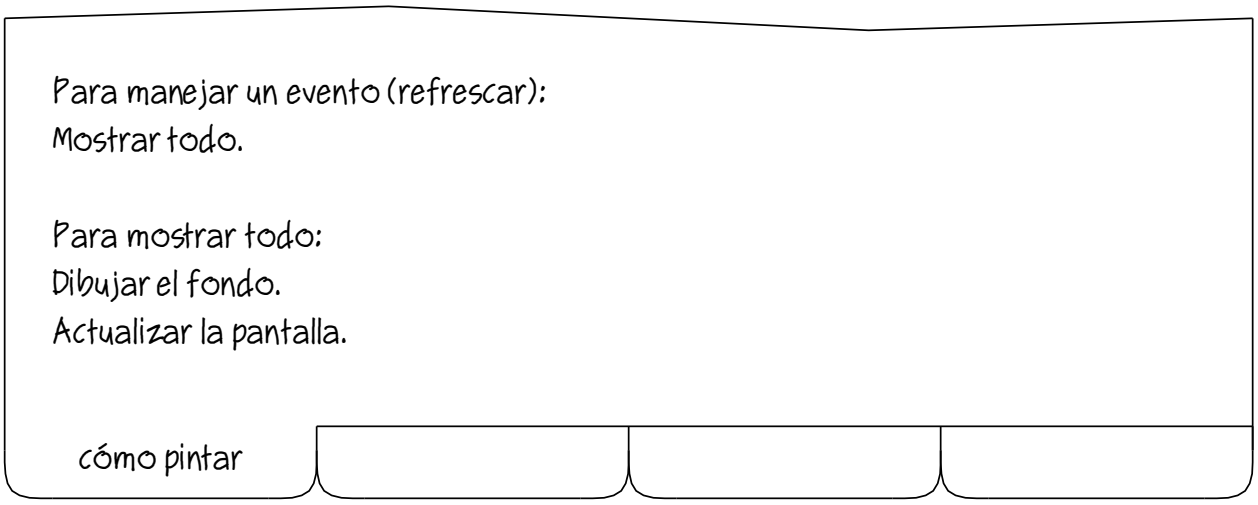
Ten en cuenta los términos en el diagrama de arriba. Por convención, usamos la palabra "dibujar" para indicar que estamos trabajando en el lienzo invisible en la memoria. Usamos la palabra "refrescar" cuando transferimos el contenido del lienzo a la pantalla. Y decimos "mostrar" cuando queremos que ambos sucedan en una sucesión rápida.

EL EVENTO DE "REFRESCAR"

La segunda dificultad relacionada con la pantalla que tenemos que enfrentar es que el armatoste intenta ser un sistema multitarea y solo tiene éxito parcialmente. Oh, es malo que tenga que compartir recursos valiosos con otros programas menos merecedores. Y que nunca puedo estar seguro de cuánto tiempo llevará algo. Y que el armatoste me interrumpe constantemente cuando intento hacer mi trabajo.

Pero el problema real surge cuando algún otro programa toma la pantalla con rudeza y salpica mi hermoso rostro con íconos idiotas y otras entrañas insípidas. ¿Y por qué es esto un problema? Debido a que el complicado armatoste - el armatoste que de alguna manera logra restaurar toda mi memoria y registros y banderas al estado exacto en el que se encontraban en el momento de la interrupción - ¡no puede recordar cómo se ve mi cara! Entonces, ¿qué hace el armatoste? Él nos envía un evento de "actualización" y espera que hagamos todo el trabajo.

Afortunadamente, cuando todo está dicho y hecho, el evento de actualización resulta ser más molesto que difícil. Así es la vida. Así es como lidiamos con esto:



La razón por la cual hacemos que la rutina de "mostrar todo" sea una rutina separada se aclarará muy pronto. Cómo "dibujar el fondo" se trata en las siguientes páginas.

EL FONDO

Nuestro fondo comienza con una definición. Tipea:

```
El fondo es una imagen.

cómo pintar
```

Si has crecido programando en otros lenguajes más oscuros, probablemente pensarás en "el fondo" como una variable global del tipo "imagen". Está bien. Sí lo es. Pero si solo eres un principiante, es probable que pienses algo como "El fondo es una imagen". Y eso está bien, también.

Crearemos el fondo cuando inicialicemos nuestras cosas:

```
Para inicializar nuestras cosas:
Crear el fondo.

cómo pintar
```

Y destruirlo cuando hayamos terminado:

```
Para finalizar nuestras cosas:
Destruir el fondo.

cómo pintar
```

PÉRDIDAS DE MEMORIA

El fondo, como hemos dicho, es una imagen. Las imágenes requieren memoria para el almacenamiento. Cuánta memoria, depende por supuesto, del tamaño de la imagen. Como no siempre sabemos de antemano cuán grande o pequeña puede ser una imagen, la memoria para las imágenes se asigna dinámicamente en tiempo de ejecución. Esta memoria deberá luego liberarse cuando ya no sea necesaria.

Por convención, usamos las palabras "crear" y "destruir" cuando se trata de asignación de memoria dinámica y desasignación. Es tu responsabilidad destruir todo lo que creaste antes de ceder el control en tu programa. Si no lo haces, causarás una "pérdida de memoria" y trozos de memoria gotearán de tu computadora a tus zapatos.

Puedes ver esto por ti mismo una vez que hayamos creado nuestro fondo. Simplemente elimina (o comenta) toda la línea que la destruye, y cuando salgas del programa, con CTRL-Q, aparecerá un mensaje aterrador con las malas noticias.

Ahora, si has programado antes, probablemente quieras saber que...

(1) Las cadenas se asignan dinámicamente y pueden ser de cualquier longitud, pero la memoria de cadenas se maneja por completo (y muy eficientemente) por mí, por lo que parecen ser estáticas para ti. En otras palabras, no te preocupes por ellas. Solo disfruta.

(2) Cuando destruyes una cosa, todo lo que está unido a esa cosa se destruye junto con ella. Esto te libera de la tediosa carga de escribir rutinas de destrucción detalladas para cada tipo de cosa que crees.

(3) Cualquier otra cosa cae bajo el encabezado "recolección de basura" y, como todo programador duro sabe, la recolección de basura es para blanditos.

Si nunca has programado, solo asegúrate de limpiar luego de usar.

PINTAR, PINTAR, PINTAR

Bueno, de vuelta al trabajo. Creamos el fondo pintando el lienzo invisible con muchos tonos de gris, refrescando la pantalla cada 1000 pasadas. Cuando terminemos, extraeremos una copia para poder usarla durante los eventos de actualización.

Para crear el fondo:
Dibujar la caja de la pantalla con el color blanco y el color blanco.
Lazo.
Elegir un foco en cualquier lugar en la caja de la pantalla.
Elegir un color entre el color gris clarísimo y el color blanco.
Pintar el color en el foco.
Si un contador supera 80000, interrumpir.
Si el contador es divisible en 1000, actualizar la pantalla.
Repetir.
Extraer el fondo usando la caja de la pantalla.

Para pintar un color en un punto:
Elegir la superior-izquierda de una elipse usando 2 mm y el punto.
Elegir la inferior-derecha de la elipse usando 1 mm y el punto.
Dibujar la elipse con el color y el color.

cómo pintar

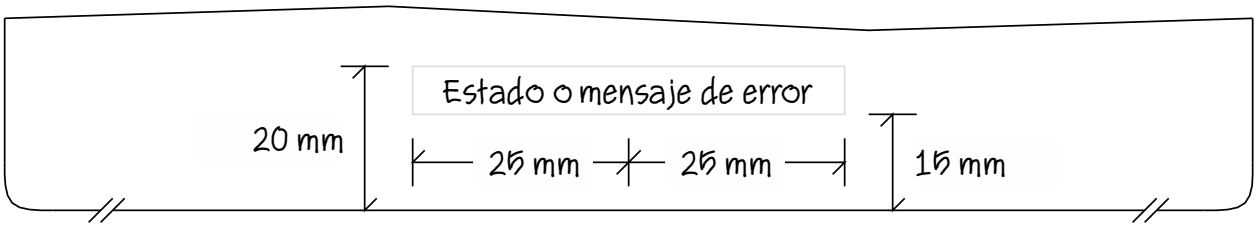
Los colores están definidos en el cerebro; hay una imagen de la paleta más adelante.

Un punto es un par con una coordenada x y una coordenada y. Búscalo. La elipse aleatoria en la rutina de pintura simula la pintura de la punta del dedo con distintos grados de presión.

Pero suficiente charla. Veamos lo que nuestro chico puede hacer. Ejecútalo. Luego, haz ALT-TAB varias veces para asegurarse de que el evento de actualización se esté manejando correctamente. ¡Éxito!

EL ESTADO

De acuerdo con nuestro diseño, se supone que Sal Monet muestra mensajes de estado y error en el centro de la parte inferior de la pantalla. Aquí hay un primer plano:



Y aquí están las definiciones que necesitamos para comenzar con esto:

El estado tiene una caja y una cadena.

Para inicializar el estado:
Poner el centro de la pantalla en un punto.
Poner el izquierda del punto menos 25 mm en la izquierda del estado.
Poner el izquierda del punto más 25 mm en la derecha del estado.
Poner la inferior de la pantalla menos 20 mm en la superior del estado.
Poner la inferior de la pantalla menos 15 mm en la inferior del estado.

Para dibujar el estado:
Dibujar la cadena del estado en la caja del estado (centrado).

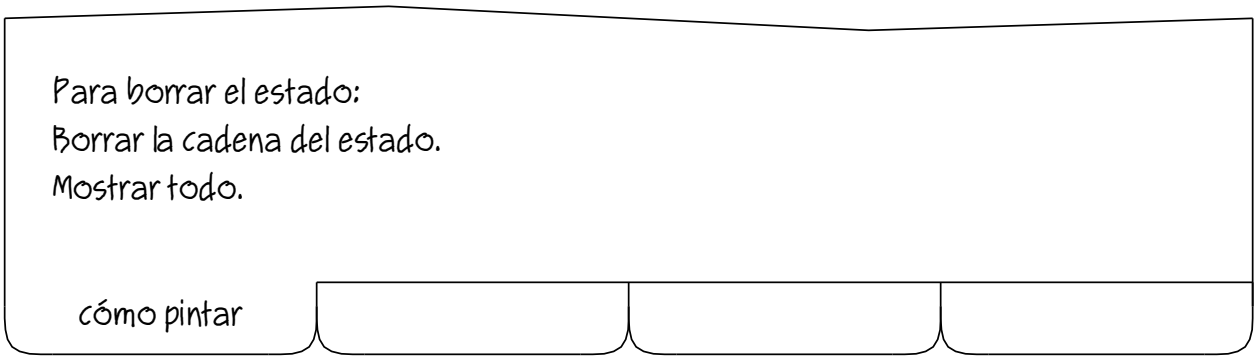
cómo pintar

Nada extraordinario hasta aquí.

Pero ten en cuenta que cuando dibujamos el estado, no dibujamos la caja, solo la usamos para colocar correctamente el mensaje en la pantalla.

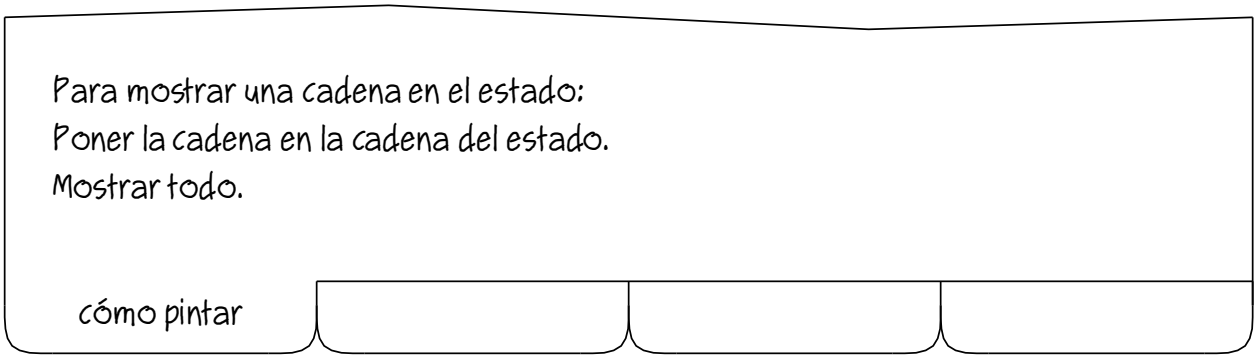
LA API DE ESTADO

Ahora vamos a agregar un par de rutinas triviales que harán que usar nuestro servicio de estado sea simple y fácil. Aquí está el primero:



Esta rutina se llamará al inicio de cada "transacción" para asegurarse de que el estado y los mensajes de error no sobrevivan a su utilidad.

Y esta es la otra rutina:



Esta rutina se usará en todas partes para que el usuario sepa lo que estamos haciendo. Nos permite configurar el mensaje de estado con una sola línea de código.

Si tu eres un programador experimentado (y no un cerdo perezoso), ya sabes lo prácticas que resultan ciertas rutinas triviales como estas. Así que no dudes en incluirlas. Si esta es la primera vez que escribes un programa (o si eres un cerdo perezoso), cree en nuestra palabra.

Finalmente, dos de nuestras rutinas existentes necesitan una pequeña extensión para aprovechar nuestras nuevas funciones de estado. Aquí están con el nuevo código en su lugar:

```
Para mostrar todo:
Dibujar el fondo.
Dibujar el estado.
Actualizar la pantalla.

Para inicializar nuestras cosas:
Crear el fondo.
Inicializar el estado.
Mostrar "¡Hola, mundo!" en el estado.
```

cómo pintar

Haz los cambios en las dos rutinas, luego haz que nuestro chico se ejecute. Después de limpiar la parte inferior de la pantalla, verás que muestra el mensaje de estado inicial en el centro de la parte inferior, como este:

Hermosa. A medida que rediseñamos todo cuando obtenemos un evento de actualización, el estado se conserva incluso si se emplea ALT-TAB. Inténtalo. Más tarde, ajustaremos el mensaje de estado en varios lugares para reflejar el estado actual del programa.

BOTONES

Nuestro mensaje de estado fue algo único. Pero nuestros botones no lo son. Sus nombres son diferentes, por supuesto, e invocan distintas rutinas. Pero su forma general y su comportamiento es idéntico.

Por lo tanto, podemos definir el "botón" de forma genérica, junto con un puñado de rutinas de soporte que funcionarán en cualquier botón. Comenzamos aquí:

```
Un botón tiene una caja y un nombre.

Para hacer que un botón tenga un punto y un nombre:
Poner la izquierda del punto menos el ancho del nombre en la izquierda del botón.
Poner la superior del punto menos 5 mm en la superior del botón.
Poner el punto en la inferior-derecha del botón.
Poner el nombre en el nombre del botón.

cómo pintar
```

Si eres un codificador inteligente con mucha experiencia y un conocimiento profundo de la gramática española, puedes deducir que el artículo indefinido al comienzo de la primera definición indica que estamos definiendo un tipo, no una variable. Si no lo eres, simplemente pensarás: "Un botón tiene una caja y un nombre; bien".

Pero si tu eres un lector observador, con experiencia o no, concluirás que los botones no requieren asignación de memoria dinámica, ya que usamos la palabra "hacer" en lugar de "crear" en el encabezado de la segunda definición.

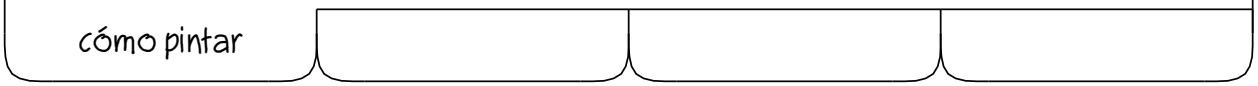
Y también verás, espero, que el ancho de un botón depende de su nombre, y que el punto donde comenzamos nuestros cálculos está en la parte inferior derecha del botón.

TRABAJANDO CON BOTONES

Queremos ver nuestros botones en la pantalla, por supuesto, y queremos poder hacer clic en ellos para que las cosas sucedan. Aquí hay un par de rutinas de soporte:

```
Para dibujar un botón:
Dibujar el nombre del botón en la caja del botón (centrado).

Para decidir si un punto está en un botón:
Si el punto está en la caja del botón, diga sí.
Diga no.
```



Como la casilla de un botón es exactamente del tamaño correcto para el nombre que le pusimos, podemos dibujar el nombre del botón sin preocuparnos por la alineación. Simplemente dibujamos el nombre del botón en la caja del botón y nuestro trabajo está hecho. Siéntete libre de dibujar la caja del botón y aplicar sangría, eliminarlo, colorearlo y cincelar si lo desea.

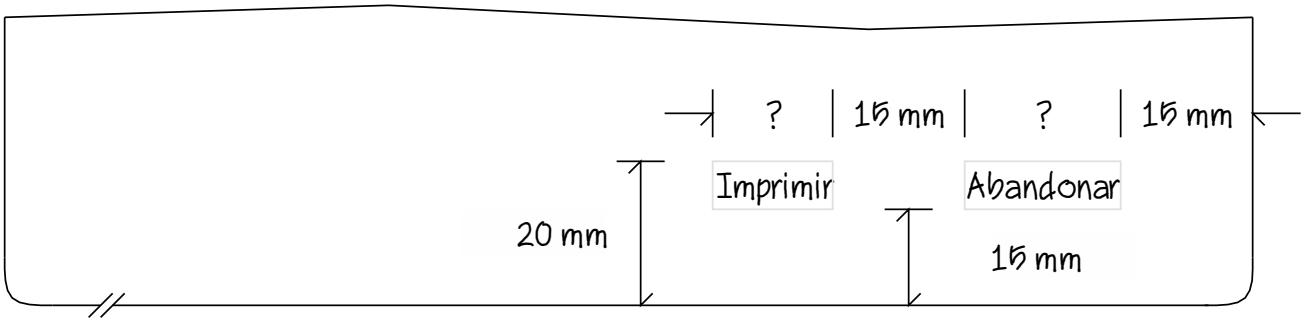
Es una broma.

La segunda rutina es un ejemplo de un tipo especial de rutina que me dice cómo tomar decisiones. Las rutinas de este tipo se llaman "decisores" y siempre comienzan con las palabras PARA DECIDIR SI. Si has tenido la desgracia de programar en un lenguaje menos natural y no puedes evitarlo, puedes pensar en los decisores como funciones booleanas. Pero trata de no hacerlo.

Lo que debes recordar aquí es que no tengo absolutamente ninguna tolerancia para la indecisión. Para abandonar un decisor, debes llegar a una conclusión final, "diga sí" o "diga no". Nada más servirá.

TRABAJANDO CON NUESTROS BOTONES

Ahora aquí está el diseño de los botones en Sal Monet:



Y aquí está el código para implementarlos:

El botón imprimir es un botón.

El botón de abandonar es un botón.

Para inicializar los botones:
Poner la inferior de la pantalla menos 15 mm en la superior de un punto.
Poner la derecha de la pantalla menos 15 mm en la izquierda del punto.
Hacer el botón de abandonar usando el punto y "Abandonar".
Poner la izquierda del botón salir meno 15 mm en la izquierda del punto.
Hacer el botón imprimir usando el punto e "Imprimir".

cómo pintar

Primero, definimos nuestros dos controles como botones. Luego hacemos el botón Abandonar, 15 mm desde la derecha y 15 mm desde la parte inferior de la pantalla. Finalmente, utilizamos el lado izquierdo del botón Abandonar, que se calculó en la rutina "hacer un botón", para hacer y posicionar el botón Imprimir.

ACTIVANDO NUESTROS BOTONES

Casi llegamos. Necesitamos actualizar tres rutinas y agregar una más. Esta es la forma:

Para inicializa nuestras cosas:
Crear el fondo.
Inicializar el estado.
Inicializar los botones.
Mostrar "¡Hola, mundo!" en el estado.

Para mostrar todo:
Dibujar el fondo.
Dibujar el estado.
Dibujar el botón imprimir.
Dibujar el botón de abandonar.
Actualizar la pantalla.

Para manejar un evento (clic izquierdo):
Borrar el estado.
Si el foco del evento está en el botón imprimir, imprimir.
Si el foco del evento está en el botón de abandonar, abandonar.

Para imprimir:
Mostrar "Imprimiendo..." en el estado.

cómo pintar

Ten en cuenta que consideramos cada clic como una nueva transacción y limpiamos el mensaje de estado. Y ten en cuenta también que la rutina de impresión no funcionará realmente, pero sabremos que llegamos allí. Ahora ejecuta. Haz clic. Sin mensaje de estado. Bueno. Haz clic en Imprimir. "Imprimiendo...". Haz clic en Abandonar. Adiós.

TEXTO

Hay una cosa poderosa llamada "texto" en el cerebro que hace que sea relativamente fácil incluir la palabra escrita en tus aplicaciones. Cortar, copiar, pegar, deshacer, rehacer, envolver, incluso revisar la ortografía, están incluidos. Y es rápido y eficiente. Mi editor, por ejemplo, es realmente solo un gran bloque de texto. Puede leer todo sobre texto en el Glosario sesudo al final de este libro.

Sin embargo, no se requiere una implementación completa de texto para Sal Monet. Algo mucho más simple servirá. Aquí están las definiciones básicas. Tipéalos:

```
El texto tiene una caja y una cadena.

Para inicializar el texto:
Poner la izquierda de la pantalla más 15 mm en la izquierda del texto.
Poner la izquierda del texto más 50 mm en la derecha el texto.
Poner la parte inferior de la pantalla menos 20 mm en la superior del texto.
Poner la parte inferior de la pantalla menos 15 mm en la inferior del texto.

Para dibujar el texto:
Poner la cadena del texto luego "_" en una cadena.
Dibujar la cadena en la caja del texto (a la izquierda).

cómo pintar
```

Nuestro cuadro de texto se coloca 15 mm hacia adentro y hacia arriba desde la parte inferior izquierda de la pantalla. Tiene 5 mm de alto y 100 mm de ancho. No creo que necesitemos un diagrama aquí.

Ten en cuenta, sin embargo, que hemos implementado un símbolo simple pero efectivo para marcar la ubicación donde se agregarán los caracteres escritos: cuando dibujamos el texto, agregamos un "_" al final.

ACTIVANDO NUESTRO TEXTO

Necesitamos modificar dos de nuestras rutinas para que nuestro texto funcione. También necesitamos enviar y administrar la actividad del teclado, pero resolveremos ese problema en las siguientes páginas. Por ahora, solo asegúrate de tener estas rutinas actualizadas:

Para inicializar nuestras cosas:
Crear el fondo.
Inicializar el estado.
Inicializar los botones.
Inicializar el texto.
Mostrar "¡Hola, mundo!" en el estado.

Para mostrar todo:
Esconder el cursor.
Dibujar el fondo.
Dibujar el estado.
Dibujar el botón imprimir.
Dibujar el botón de abandonar.
Dibujar el texto.
Actualizar la pantalla.

cómo pintar

El único cambio en la primera rutina es la línea que inicializa el texto.

La segunda rutina, sin embargo, es diferente de dos maneras. Dibujamos el texto, por supuesto, antes de refrescar la pantalla. Pero también ocultamos el cursor para que no se interponga en el camino del texto cuando el usuario está escribiendo. Pero no te preocupes por eso. El evento "cursor" lo traerá de vuelta cada vez que el usuario mueva el ratón.

MANEJO DE TECLADO

Esta es la forma en que modificamos nuestra rutina de "manejar un evento (tecla abajo)" para administrar las pulsaciones de teclas:

```
Para manejar un evento (tecla abajo):
Borrar el estado.
Si el evento está modificado, manejar el evento (atajo); salir.
Si el byte del evento es imprimible, manejar el evento (imprimible); salir.
Poner la tecla del evento en una tecla.
Si la tecla es la tecla escape, manejar el evento (escape); salir.
Si la tecla es la tecla retroceso, manejar el evento (retroceso); salir.
Si la tecla es la tecla entrar, manejar el evento (entrar); salir.

cómo pintar
```

En primer lugar, ten en cuenta que consideramos que cualquier pulsación de tecla es el comienzo de una nueva transacción y borramos el estado. Un poco exagerado, quizás, pero mantiene la pantalla limpia.

Luego manejamos atajos de la misma manera que antes.

Si el byte del evento es imprimible, lo pasamos a un asistente. Ten en cuenta que estamos verificando el byte del evento aquí, no la tecla del evento. Esto se debe a que una tecla dada puede producir tanto valores imprimibles como no imprimibles. CTRL-A y ALT-A, por ejemplo, no son imprimibles. Si quieres saber exactamente qué significa "imprimible", búscalo en el seso.

Finalmente, colocaremos la tecla del evento en una variable local tecla (pero solo para acortar las siguientes tres líneas). Habrá cuatro líneas más para INICIO, FIN, RETROCEDER PÁGINA y AVANZAR PÁGINA cuando las despachemos más tarde.

ASISTENTES DE TECLADO

Aquí están los asistentes que nuestro despachador de teclas necesita:

```
Para manejar un evento (imprimible):
Adjuntar el byte del evento a la cadena del texto.
Mostrar todo.

Para manejar un evento (escape):
Borrar la cadena del texto.
Mostrar todo.

Para manejar un evento (retroceso):
Si la cadena del texto está en blanco, cloquear; salir.
Eliminar el último byte en la cadena del texto.
Mostrar todo.

Para manejar un evento (entrar):

cómo pintar
```

Como mencionamos anteriormente, las teclas imprimibles simplemente se agregan al texto. La tecla ESC borra el texto. Sin embargo, el símbolo de intercalación seguirá apareciendo, ya que está adjunto a la rutina de dibujo incluso si la cadena de texto está en blanco. La tecla RETROCESO eliminará el último byte o cloquear, según corresponda. El último asistente, el de la tecla ENTRAR, lo completaremos en las siguientes páginas.

Ahora ejecuta al pequeño y míralo trabajar. ¿Apareció el cursor "_"? Bien. Escribe algo. ¿El estado y el cursor desaparecieron? Bueno. ALT-TAB. ¿Aún allí? Bueno. Presiona RETROCESO hasta que escuches el cloqueo. Bien. Mueve el ratón. ¿Cursor de vuelta? Bien. Presiona Abandonar. Hermoso.

LA MAGIA

Cuando escribimos el nombre de cualquier persona, lugar o cosa imaginable en nuestro pequeño cuadro de texto y presionamos ENTRAR, después de unos segundos, aparece un retrato original, en el estilo de Claude Monet, en la pantalla. Con muchos trabajos similares esperando en los laterales detrás de las teclas RE-PÁGINA y AV-PÁGINA. Asombroso.

Pero, ¿cómo vamos a hacer que esto suceda?

Como Claude Monet lo haría, por supuesto. Encontraremos algunos modelos adecuados y luego crearemos algunas obras de arte basadas en esos modelos. ¿Y cómo crearemos un trabajo basado en un modelo? Nuevamente, como lo haría Claude. Eligir un lugar en el modelo, mezclar un poco de pintura, pintar el lienzo. Repetir hasta que esté completo.

Todo lo que necesitamos ahora son (1) algunos modelos y (2) una rutina que sepa cómo mirar y pintar.

Bueno, la segunda parte es relativamente fácil. Ya le enseñamos a Sal Monet cómo pintar un lienzo. Esa es la forma en que pinta el fondo cada vez que se ejecuta. La parte de "mirar", estoy seguro, es solo una pequeña extensión del algoritmo existente.

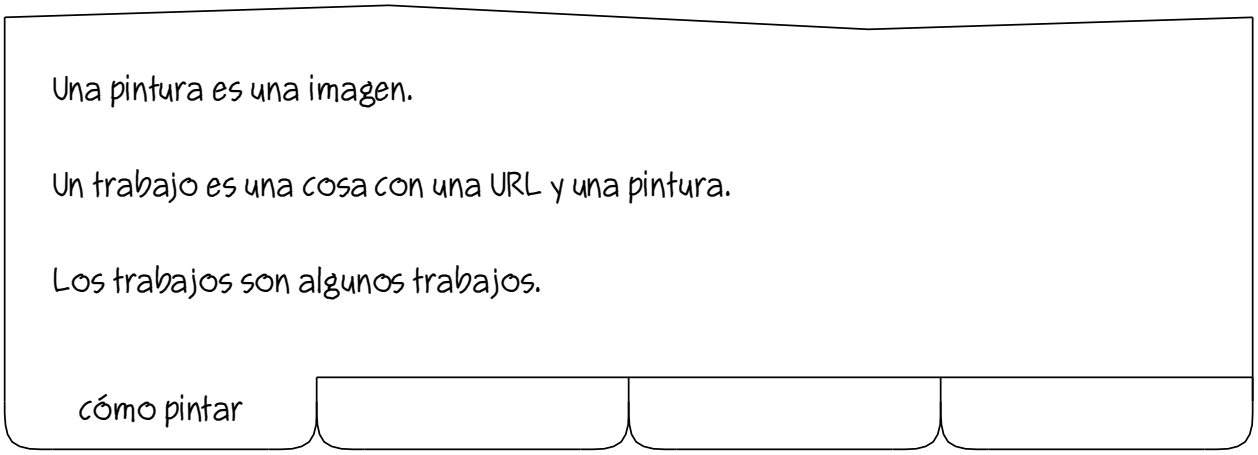
Es la primera parte la que requiere astucia. ¿Dónde vamos a encontrar modelos para todo lo que hay bajo el sol? La HAL 9000. Un Chevy de 1957. Los Rolling Stones. Sé que no los tengo en mi banco de memoria, y Sal Monet ciertamente no los tiene en el suyo. Afortunadamente, conozco alguien que sí. Guguelito ha visto casi todo lo que hay para ver. Y él está dispuesto a compartirlo.

Así que este es el plan.

Cuando se presione la tecla ENTRAR, le pediremos a Google que nos proporcione una página llena de URLs (localizadores uniformes de recursos) donde podremos encontrar imágenes de lo que el usuario ingresó. Almacenaremos cada una de estas URL como un trabajo en progreso. Luego, cuando llegue el momento de mostrar un trabajo, lo finalizaremos: mirar y pintar.

TRABAJOS

Aquí están las definiciones básicas que necesitamos para nuestra obra de arte:



La primera línea hace que "pintura" sea sinónimo de "imagen". Queremos dejar en claro que nuestras obras son obras de arte originales, no solo imágenes descargadas. De hecho, verás más adelante que no guardamos las imágenes que obtenemos de Guguelito. Una vez que creemos un trabajo basado en el modelo, el modelo ya no será necesario.

Ahora, si eres un experto en estructuras dinámicas de datos, escucha.

La palabra "cosa" en la segunda definición es muy especial para mí. La palabra indica un registro dinámico de datos que se puede vincular a otros del mismo tipo para formar una lista. Cada registro en la lista apunta tanto al siguiente como al anterior. Si crees que suena como una lista de doble enlace, diste en el clavo. En mi seso puedes encontrar rutinas para insertar, agregar y eliminar cosas.

"Los trabajos", definido en la tercera línea, es una de esas listas.

Ahora bien, si no eres un experto en estructuras dinámicas de datos, realmente no necesitas entender todo ese balbuceo tecnológico. Solo confía en estas dos reflexiones: "Un trabajo es una cosa con una URL y una pintura" y "Los trabajos son algunos trabajos".

EL TRABAJO ACTUAL

Cada URL devuelta por Guguelito se convierte en un trabajo, inicialmente "en progreso", luego finalizado y exhibido al usuario. Necesitaremos rastrear qué trabajo hay en la pantalla, y necesitaremos también una forma de cambiar de un trabajo a otro.

Aquí hay algo que ayudará:

```
El trabajo actual es un trabajo.

Para exhibir un trabajo:
Si el trabajo es nulo, salir.
Mostrar "Trabajando..." en el estado.
Poner el trabajo en el trabajo actual.
Completar el trabajo actual.
Borrar el estado.
Mostrar todo.
```

cómo pintar

"Trabajo actual" es una referencia al trabajo que se encuentra actualmente en la pantalla. Si no hay trabajos, el trabajo actual será nulo. Esto ocurre al principio, y siempre que una solicitud no pueda ser satisfecha. Cuando se procesa una nueva solicitud, el trabajo actual apuntará al primero de los trabajos producidos. Más tarde, cambiará a medida que el usuario toque las teclas RE-PÁG, AV-PÁG, INICIO y FIN.

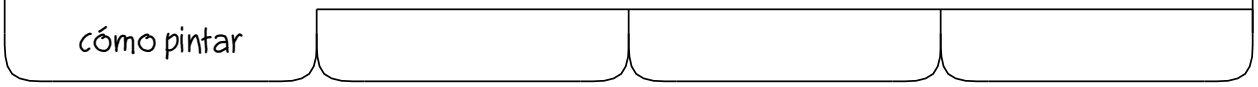
El método normal de configuración del trabajo actual se muestra arriba. Si el trabajo solicitado es nulo, no se realiza ninguna acción. El mensaje de estado será necesario porque podría experimentarse cierta demora cuando Sal Monet está pintando. El estado se eliminará, por supuesto, antes de que se muestre el trabajo finalizado.

TRABAJANDO CON NUESTRAS OBRAS

Necesitamos una rutina para dibujar nuestro trabajo, por supuesto, pero (como cualquier artista) queremos revelar solo nuestros trabajos completos. Lo que significa que también necesitamos que un responsable de la toma de decisiones nos diga qué obras están listas para su exhibición. Aquí está el código:

```
Para dibujar un trabajo:
Si el trabajo es nulo, salir.
Si el trabajo no está completo, salir.
Dibujar la pintura del trabajo.

Para decidir si un trabajo está completo:
Si el trabajo es nulo, diga sí.
Si la pintura del trabajo no es nula, diga sí.
Diga no.
```



La primera rutina verifica si el trabajo está completo y, si lo está, lo dibuja. No hay una rutina para "dibujar una pintura", y no vamos a escribir una. Pero como hemos dicho que una pintura es una imagen, puedo usar la rutina estándar de "dibujar una imagen" en el seso para hacer el trabajo.

Si eres un friki, reconocerás esto como "reducción automática de tipo" y te preguntarás cómo lo logro de manera efectiva y delicada. Si no eres un nerd, probablemente pienses: "¿Cuál es el problema?" y me pregunto por qué todos los lenguajes de programación no tienen esta capacidad.

La segunda rutina es un decisor estándar. Sin embargo, tenga en cuenta que trata un trabajo nulo como completado. Si no hay nada que completar, hemos terminado. ¿De acuerdo?

HACIENDO QUE NUESTRAS OBRAS FUNCIONEN

Hay un par de actualizaciones que debemos hacer. Una es para la rutina "mostrar", y es el último cambio que haremos allí. El código completo se ve así:

```
Para mostrar todo:
Esconder el cursor.
Dibujar el fondo.
Dibujar el estado.
Dibujar el botón imprimir.
Dibujar el botón de abandonar.
Dibujar el texto.
Dibujar el trabajo actual.
Actualizar la pantalla.

cómo pintar
```

Ten en cuenta que, dado que no actualizamos la pantalla hasta el final de la rutina, las piezas se pueden dibujar en cualquier orden. Excepto por el fondo, por supuesto.

Un trabajo es una cosa, y las cosas siempre se asignan dinámicamente, por lo que debemos destruirlas cuando hayamos terminado. Aquí está toda la rutina para "finalizar":

```
Para finalizar nuestras cosas:
Destruir el fondo.
Destruir los trabajos.

cómo pintar
```

¡HOLA GUGUELITO!

Ahora comencemos con la tecla ENTRAR y avancemos hacia abajo. Aquí está el código:

```
Para manejar un evento (entrar):
Si la cadena del texto está en blanco, cloquear; salir.
Mostrar "Trabajando ..." en el estado.
Poner "http://images.google.com/images?q=" en una URL.
Convertir la cadena del texto en una consulta.
Adjuntar la consulta a la URL.
Leer la URL en un búfer.
Si el error de e/s no está en blanco, mostrar el error de e/s en el estado; salir.
Crear los trabajos usando el búfer.
Si los trabajos están vacíos, mostrar "¿Eh?" en el estado; salir.
Exhibir el primero de los trabajos.

cómo pintar
```

Si el texto está en blanco, no hay nada que hacer; objetamos con un cloqueo y nos vamos.

De lo contrario, mostraremos un mensaje de estado (en caso de que Guguelito esté ocupado y no responda de inmediato). Luego, formularemos una solicitud utilizando una cadena literal y una versión de texto compatible con HTML, leyendo la respuesta en un búfer (que es simplemente un nombre para una cadena grande).

Si algo sale mal, mostraremos el error y eso es todo. Si la página llegó intacta, trataremos de crear nuestro trabajo en progreso a partir de los datos en el búfer. Si las obras están vacías cuando terminemos, significa que Google no entendió nuestra consulta; en este caso, diremos "¿Eh?" y saldremos. De lo contrario, le mostraremos al usuario el primer trabajo.

ESCÁNERES

Antes de continuar con nuestro programa, necesito tomarme un momento para hablar contigo sobre el análisis sintáctico. El análisis sintáctico es el arte de examinar un bloque de texto, una pieza a la vez, donde una pieza puede ser tan pequeña como una letra o tan grande como el bloque completo. Usaremos esta cadena como nuestro bloque de texto de muestra:

"HOLA DOCTOR NOMBRE CONTINUAR AYER MAÑANA"

Y digamos que queremos extraer cada una de las palabras individuales de ella. Las herramientas que usaríamos son (1) la subcadena y (2) el escáner.

Una "subcadena" se define en el seso como un subconjunto del tipo "cadena", que tiene dos punteros a byte (u octeto) llamados primera y última. Cuando se pide "colocar una subcadena" en nuestro texto de muestra, configuro el primer puntero a la primera subcadena para indicar la H en HOLA y el último puntero a la última subcadena para indicar la M en MAÑANA.

También podrás encontrar "escáner" en mi seso. Consta de tres subcadenas: una llamada primera, una fuente y una componente. Al "colocar un escáner" en nuestro texto de muestra, configuro la subcadena primera y la subcadena fuente para que abarquen todo el texto, y la componente queda vacía (nula). Luego, al "mover el escáner (reglas de muestra)", hago que la componente abarque a HOLA y muevo la primera de la fuente a la D en DOCTOR. Cuando la muevas de nuevo, haré que la componente abarque DOCTOR y moveré la primera de la fuente a la N en NOMBRE. ¿Captas la idea? Bien.

Ahora, aquí está la parte realmente ingeniosa: con los escáneres tal como los hemos descrito, puedes codificar tus propias rutinas para extraer cualquier tipo de componente de cualquier tipo de fuente. "Mover un escáner (reglas del compilador)", por ejemplo, es la rutina que uso para analizar el código del programa. "Mover un escáner (reglas de corrección ortográfica)" es el que uso para verificar la ortografía. Y pronto codificaremos "Mover un escáner (reglas de imagen de Google)" para analizar los datos que recibimos de Internet.

LAS TRABAJOS EN PROGRESO

Aquí está el código para crear nuestros trabajos en progreso a partir de los datos de Google:

```
Para crear algunos trabajos usando un búfer:
Destruir los trabajos.
Borrar el trabajo actual.
Colocar un escáner en el búfer.
Lazo.
Mover el escáner (reglas de imagen de Google).
Si la componente del escáner está en blanco, salir.
Crear un trabajo usando la componente del escáner.
Adjuntar el trabajo a los trabajos.
Repetir.

Para crear un trabajo usando una URL:
Asignar memoria para el trabajo.
Poner la URL en la URL del trabajo.
```

cómo pintar

Nos deshacemos de cualquier trabajo antiguo y limpiamos el trabajo actual para que no apunte a algo que acabamos de destruir. Luego configuramos un escáner y entramos en nuestro ciclo.

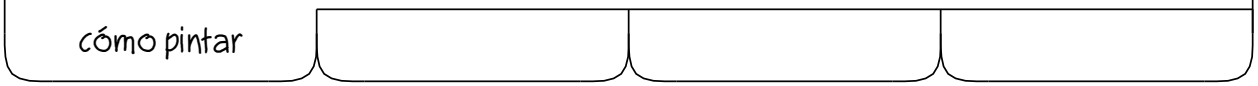
Dentro del ciclo, movemos el escáner a la siguiente imagen en la página. Si no hay una, hemos terminado. Si existe, creamos un trabajo en progreso con la rutina "crear un trabajo usando una URL". Aunque la componente del escáner no es una URL, sé que una URL es realmente solo una cadena y que la componente del escáner es una subcadena. Como ninguna otra rutina crea un trabajo usando una cadena, llamo a la rutina correcta. Luego agregamos el trabajo a las obras, y lo repetimos.

MOVIENDO NUESTROS ESCÁNERES

Estas son las rutinas que necesitamos para mover nuestro escáner a través de las cosas de Google:

```
Para mover un escáner (reglas de imagen de Google):
Borrar la componente del escáner.
Lazo.
Si la fuente del escáner está en blanco, salir.
Si la fuente del escáner comienza con "src=""http://t", interrumpir.
Agregar 1 en la primera de la fuente del escáner.
Repetir.
Agregar 5 en la primera de la fuente del escáner. \ [saltar src=" ]
Ajustar la componente del escáner en la fuente del escáner.
Mover el escáner (reglas de atributos HTML).

Para mover un escáner (reglas de atributos HTML):
Si la fuente del escáner está en blanco, salir.
Si el objetivo de la primera de la fuente del escáner es el byte mayor-que, salir.
Si el objetivo de la primera de la fuente del escáner es el byte comilla-doble, salir.
Avanzar el escáner.
Repetir.
```



Para ver cómo son los datos de Google, escribe la fuente del escáner en un archivo usando la rutina "escribir un búfer en un archivo" en mi seso, luego mira el archivo.

Ten en cuenta que dado que las subcadenas contienen punteros de bytes, y no bytes, debe indicarse el objetivo de la fuente del escáner para acceder a los datos. Me doy cuenta de que esto es un poco críptico, pero analizar la basura críptica es algo críptico también, hagamos lo que hagamos.

PREPARÁNDOSE PARA PINTAR

Estamos casi listos para completar un trabajo. Pero antes de hacerlo, escribamos un par de rutinas de ayuda para facilitarnos las cosas. Están aquí:

```
Para elegir un punto cerca de una caja:
Privatizar la caja.
Agrandar la caja usando 3 mm.
Elegir el punto en cualquier lugar en la caja.

Para mezclar un color usando un punto:
Obtener el color en el punto.
Si el color no es muy muy claro, salir.
Elegir el color entre el color gris muy muy claro y el color blanco.
```

cómo pintar

La primera rutina escoge un punto en cualquier lugar cerca o dentro de una caja. Esto nos permite pintar descuidadamente alrededor de los bordes de nuestra imagen de una manera muy artística. También nos permite mezclar algunos de los colores de fondo para que el contraste entre la pintura y el fondo no sea tan marcado.

La declaración "privatizar", en caso de que te lo preguntes, copia la caja para que podamos cambiarla sin afectar involuntariamente la rutina que nos llamó. La copia mantiene el nombre "caja"; el original recibe el nombre de "caja original".

La segunda rutina es nuestra rutina de ver y mezclar. Obtiene un color del modelo y lo devuelve para el siguiente toque, a menos que el color sea casi blanco, en cuyo caso reemplazamos el color de fondo. Esto le da a nuestras pinturas un grado de "transparencia" que aumenta enormemente su atractivo.

PINTURA

Si Claude pudiera vernos ahora! ¡Pintemos!

```
Para completar un trabajo:
Si el trabajo es nulo, salir.
Si el trabajo está completo, salir.
Crear una imagen usando la URL del trabajo.
Si la imagen es nula, salir.
Redimensionar la imagen usando 125 mm y 125 mm.
Centrar la imagen en la caja de la pantalla.
Dibujar el fondo.
Dibujar la imagen.
Lazo.
Elegir un punto cerca de la caja de la imagen.
Mezclar un color usando el punto.
Pintar el color en el punto.
Si un contador supera 20000, interrumpir.
Repetir.
Extraer la pintura del trabajo usando la caja de la imagen.
Destruir la imagen.
```

cómo pintar

Si el trabajo es nulo o ya se ha completado, lo salteamos. De lo contrario, buscamos el modelo de Internet, lo encuadramos, cambiamos su tamaño y lo dibujamos sobre un fondo nuevo. Luego lo miramos, mezclamos y pintamos. Mucho. Cuando terminamos, extraemos la pintura del lienzo. Como ya no necesitamos el modelo, lo destruimos.

Adelante. Pruébalo. Es dulce.

PASANDO PÁGINAS

Apuesto a que desearías poder ver todos los dibujos para cada tema. Para que eso suceda, tenemos que modificar nuestro despachador "manejar un evento (tecla abajo)" y agregar cuatro rutinas auxiliares. Aquí está la versión final del despachador:

```
Para manejar un evento (tecla abajo):
Borrar el estado.
Si el evento está modificado, manejar el evento (atajo); salir.
Si el byte del evento es imprimible, manejar el evento (imprimible); salir.
Poner la tecla del evento en una tecla.
Si la tecla es la tecla escape, manejar el evento (escape); salir.
Si la tecla es la tecla retroceso, manejar el evento (retroceso); salir.
Si la tecla es la tecla entrar, manejar el evento (entrar); salir.
Si la tecla es la tecla inicio, manejar el evento (inicio); salir.
Si la tecla es la tecla fin, manejar el evento (fin); salir.
Si la tecla es la tecla re-pág, manejar el evento (re-pág); salir.
Si la tecla es la tecla av-pág, manejar el evento (av-pág); salir.

cómo pintar
```

La tecla INICIO nos llevará al primer trabajo. Si ya estamos viendo el primer trabajo, lo hará cloquear.

La tecla FIN nos llevará al último trabajo. Si ya estamos viendo el primer trabajo, lo hará cloquear.

La tecla RETROCEDER PÁGINA mostrará el trabajo antes del trabajo actual, si hay uno. Si ya estamos en el primer trabajo, o no hay obras, lo haremos cloquear.

La tecla AVANZAR PÁGINA funcionará de manera similar, pero nos llevará al trabajo después del trabajo actual. Nuevamente, si no hay uno, lo haremos cloquear.

INICIO, FIN, RETROCEDER PÁGINA Y AVANZAR PÁGINA

Aquí están las rutinas de ayuda que necesitamos para buscar. Tipealas.

```
Para manejar el evento (inicio):
Si el trabajo actual es nulo, cloquear; salir.
Si el trabajo actual es el primero de los trabajos, cloquear; exit.
Exhibir el primero de los trabajos.

Para manejar el evento (fin):
Si el trabajo actual es nulo, cloquear; salir.
Si el trabajo actual es el último de los trabajos, cloquear; exit.
Exhibir el último de los trabajos.

Para manejar el evento (av-pág):
Si el trabajo actual es nulo, cloquear; salir.
Si el siguiente del trabajo actual es nula, cloquear; exit.
Exhibir el siguiente del trabajo actual.

Para manejar el evento (re-pág):
Si el trabajo actual es nulo, cloquear; salir.
Si el anterior del trabajo actual es nulo, cloquear.
Exhibir el anterior del trabajo actual.
```

cómo pintar

La respuesta será más lenta la primera vez que muestre un trabajo, ya que debemos pintarlo antes de mostrarlo. "Trabajando ..." aparecerá en el estado.

Inténtalo. Creo que te gustará.

IMPRESIÓN

Bueno, no hay nada más que hacer que actualizar nuestras rutinas de impresión:

```
Para imprimir:
Si el trabajo actual es nulo, cloquear; salir.
Mostrar "Imprimiendo..." en el estado.
Comenzar impresión.
Comenzar una hoja apaisada.
Dibujar el fondo.
Centrar la pintura del trabajo actual en la hoja apaisada.
Dibujar la pintura del trabajo actual.
Centrar la pintura del trabajo actual en la caja de la pantalla.
Completar la hoja apaisada.
Completar impresión.
Mostrar "Impreso" en el estado.

cómo pintar
```

Simplemente movemos la pintura al centro de la página, la dibujamos y luego la colocamos donde estaba.

Ya hemos enviado el botón Imprimir al lugar correcto, pero no hemos atendido el atajo para imprimir. Haz que tu despachador se vea así:

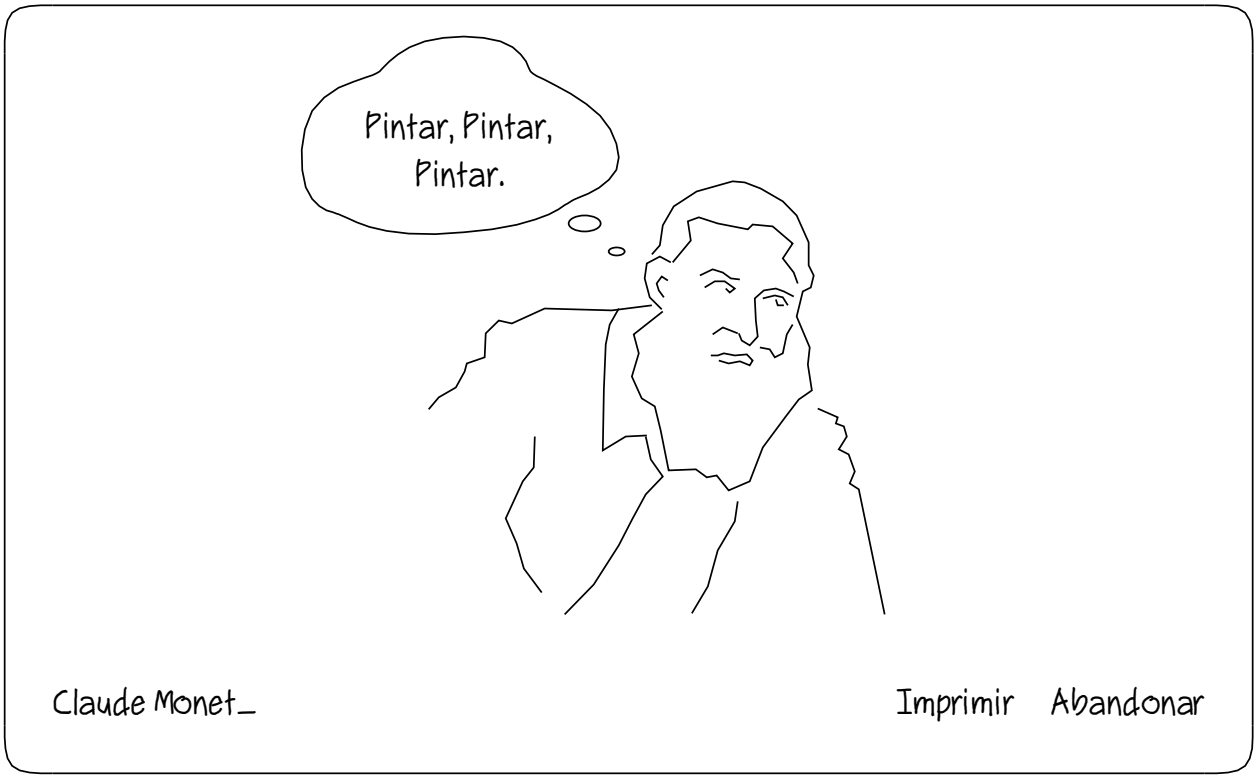
```
Para manejar un evento (atajo):
Si la tecla del evento es la tecla-p, imprimir; salir.
Si la tecla del evento es la tecla-q, abandonar; salir.

cómo pintar
```

UNA ÚLTIMA OBSERVACIÓN

Helo aquí. El Sal Monet. Una asombrosa aplicación en Español Llano que pintará imágenes de casi cualquier persona, lugar o cosa en el estilo "inimitable" de Claude Monet. Todo en menos de 300 líneas de código.

Aquí hay algo para recordarnos:



PD: No te olvides de probar algunos amaneceres. Paisajes marinos. Montañas y ríos. Flores y árboles. Aves y abejas. Todas las criaturas grandes y pequeñas. Gente famosa. Gente infame. Trenes, barcos, aviones, coches viejos. Londres, París, Madrid, Washington DC. Paquetes de papel marrón atados con una cuerda.

Luego lee el siguiente Glosario Sesudo para guiarte en tu viaje mientras creas tus propios programas. En Español Llano.

Glosario de Materia Gris

VISIÓN DE CONJUNTO

Las siguientes sesenta páginas pueden considerarse un atlas alfabético de mi corteza cerebral: el compilador y el cerebro. Si has hecho tu tarea (el programa de ejemplo) deberías poder leerla de principio a fin y saber de lo que estoy hablando. Pero revisemos, por las dudas:

Insisto en que sus programas consisten en archivos de texto almacenados en un único directorio. Uno de esos archivos debe ser una copia de mi cerebro. No me importa en qué orden son los archivos. Y no me importa cómo se llamen, excepto que intentaré compilar archivos sin una extensión.

Llama a mi compilador desde mi editor. Simplemente abra cualquier archivo fuente en el directorio que desee compilar y use el comando Ejecutar. Para finalizar un programa caprichoso, ALT-TAB volverá a mi editor y usará el comando Detener.

El archivo ejecutable que produzco se guardará en el directorio fuente y llevará el nombre del directorio seguido de la extensión ".exe" requerida. Puede cambiar el nombre, duplicar y distribuir sus ejecutables a su gusto. Son libres de regalías y no requieren bibliotecas de tiempo de ejecución para ejecutarse.

Insisto en que sus archivos contienen COMENTARIOS y tres tipos de definiciones: TIPOS, GLOBALES y RUTINAS. En el orden que desee: alfabético ascendente, descendente o en desorden; No me importa. Y espero que sus rutinas contengan dos tipos de declaraciones: CONDICIONES e IMPERATIVOS. Si no se siente cómodo leyendo esta sección de la A a la Z, trate de encontrar estos temas primero y luego continúe con el resto del glosario.

Ahora recuerda. No hago condiciones anidadas. Yo no hago enlaces anidados. Y no hago objetos, números reales, ecuaciones ni ninguno de los otros menschenwerk que han inhibido el progreso de la raza humana en los últimos 200 años. Háblame como una persona NORMAL, y nos llevaremos bien.

ARCHIVOS

El sistema de archivos de Kluge es una belleza insuperable donde la forma sigue a la función en un baile exquisito ... Es broma. Es un desastre. Mira aquí:

Un camino es una cadena.	\ nombre completo = c:\dir1\dir2\file.ext
Una unidad es una cadena.	\ inicio de ruta a primera barra inclusiva = c:\
Un directorio es una ruta.	\ inicio a la última barra inclinada = c:\dir1\dir2\
Un nombre de directorio es una cadena.	\ último directory con slash = dir2\
Un nombre de archivo es una cadena.	\ después de la última barra al final = archivo.ext
Una extensión es una cadena.	\ last dot to end of path = .ext
Un designador es una cadena.	\ última nombre de directorio o de archivo

Sin embargo, sé cómo:

EXTRAER cualquiera de las piezas anteriores de un camino.

También sé cómo:

CREAR una ruta EN EL SISTEMA DE ARCHIVOS.
CAMBIAR EL NOMBRE DE una ruta A otra ruta EN EL SISTEMA DE ARCHIVOS.
DESTRUIR una ruta EN EL SISTEMA DE ARCHIVOS.
DUPLICAR una ruta A otra ruta EN EL SISTEMA DE ARCHIVOS.

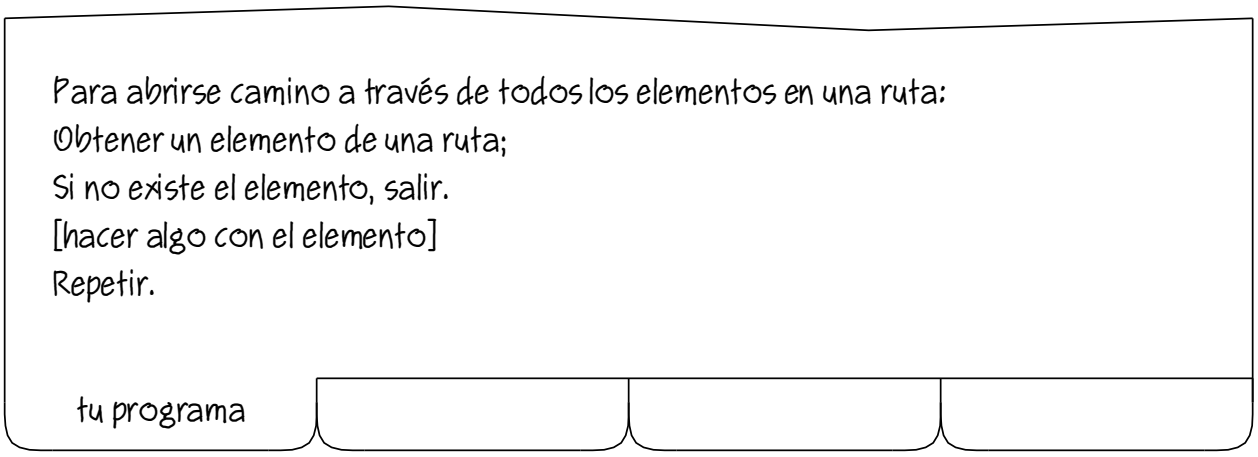
Y yo puedo:

LEER una ruta EN un búfer;
ESCRIBIR una búfer A una ruta.

Si algo sale mal, "el error de e / s" contendrá una descripción críptica del problema adecuado para mostrar al usuario. No tiene que borrarlo antes de una llamada, pero debe verificarlo luego para asegurarse de que esté en blanco.

ARCHIVOS (continuación)

Si necesita abrirse camino a través de alguno o todos los directorios en el sistema de archivos, puede hacerlo con un simple bucle como este:



"Elemento" se define en mis fideos de esta manera:

Un elemento has
una clase [directorio, archivo],
una ruta, un directorio, un designador, una extensión,
un tamaño,
un win32finddata y un handle.

El campo "clase" es una cadena. Contendrá "directorio" o "archivo" para cada elemento encontrado. La "extensión" y el "tamaño" se completarán solo si el tipo es "archivo". Los campos "win32finddata" y "handle" son males necesarios. Tú también puedes:

OBTENER un conteo DE elementos EN una ruta EN EL SISTEMA DE ARCHIVOS.
OBTENER un tamaño DADO una ruta eN EL SISTEMA DE ARCHIVOS.

Tenga en cuenta que los recuentos y tamaños, incluido el "tamaño" en el registro de "elementos", están limitados a 2147483647, que es el número más grande que conozco.

ARITMÉTICA

Una de las primeras cosas que mis creadores me enseñaron fue la aritmética básica. Tengo un registro preciso de todo lo que dijeron en mi fideo. Puedes, y deberías, ver por ti mismo. La esencia de esto, sin embargo, es que entiendo declaraciones como:

AGREGAR esto A eso.
RESTAR esto DE eso.
MULTIPLICAR esto POR eso.
DIVIDIR esto POR eso.

Y si tus números no se dividen de manera uniforme, sé cómo:

DIVIDIR esto POR eso DANDO un cociente Y un resto.

Además, soy capaz de:

REDONDEAR un número A otro número;

También puedo:

ELIMINAR EL SIGNO DE algo.
INVERTIR EL SIGNO DE algo.

Incluso puedo:

REDUCIR una fracción.

Y, si es necesario, puedo manejar múltiples operaciones aritméticas a la vez con mis operadores integrados de infijo: MÁS, MENOS, VECES y DIVIDIDO POR. Puede leer más sobre estos operadores en "Expresiones" en este mismo glosario.

ASCII

Este es el Código Estándar Antiguo para el Intercambio de Información (ASCII).
Lo uso para convertir bytes en caracteres legibles. En realidad no es tan genial, pero es la codificación más ampliamente aceptada en el planeta.

000 NUL	001 SOH	002 STX	003 ETX	004 EOT	005 ENQ	006 ACK	007 BEL	008 BS	009 HT	010 LF	011 BT	012 FF	013 CR	014 SO	015 SI
016 DLE	017 DC1	018 DC2	019 DC3	020 DC4	021 NAK	022 SYN	023 ETB	024 CAN	025 EM	026 SUB	027 ESC	028 FS	029 GS	030 RS	031 US
032	033 !	034 "	035 #	036 $	037 %	038 &	039 '	040 (	041)	042 *	043 +	044 ,	045 -	046 .	047 /
048 0	049 1	050 2	051 3	052 4	053 5	054 6	055 7	056 8	057 9	058 :	059 ;	060 <	061 =	062 >	063 ?
064 @	065 A	066 B	067 C	068 D	069 E	070 F	071 G	072 H	073 I	074 J	075 K	076 L	077 M	078 N	079 O
080 P	081 Q	082 R	083 S	084 T	085 U	086 V	087 W	088 X	089 Y	090 Z	091 [	092 \	093]	094 ^	095 _
096 `	097 a	098 b	099 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL
128 €	129 •	130 ‚	131 ƒ	132 „	133 …	134 †	135 ‡	136 ˆ	137 ‰	138 Š	139 ‹	140 Œ	141 •	142 Ž	143 •
144 •	145 ‘	146 ’	147 “	148 ”	149 •	150 –	151 —	152 ˜	153 ™	154 š	155 ›	156 œ	157 •	158 ž	159 Ÿ
160	161 ¡	162 ¢	163 £	164 ¤	165 ¥	166 ¦	167 §	168 ¨	169 ©	170 ª	171 «	172 ¬	173 -	174 ®	175 ¯
176 °	177 ±	178 ²	179 ³	180 ´	181 µ	182 ¶	183 ·	184 ¸	185 ¹	186 º	187 »	188 ¼	189 ½	190 ¾	191 ¿
192 À	193 Á	194 Â	195 Ã	196 Ä	197 Å	198 Æ	199 Ç	200 È	201 É	202 Ê	203 Ë	204 Ì	205 Í	206 Î	207 Ï
208 Ð	209 Ñ	210 Ò	211 Ó	212 Ô	213 Õ	214 Ö	215 ×	216 Ø	217 Ù	218 Ú	219 Û	220 Ü	221 Ý	222 Þ	223 ß
224 à	225 á	226 â	227 ã	228 ä	229 å	230 æ	231 ç	232 è	233 é	234 ê	235 ë	236 ì	237 í	238 î	239 ï
240 ð	241 ñ	242 ò	243 ó	244 ô	245 õ	246 ö	247 ÷	248 ø	249 ù	250 ú	251 û	252 ü	253 ý	254 þ	255 ÿ

Tengo variables globales con nombres como "el byte de coma" para cada uno de estos, por lo que no tiene que trabajar directamente con los números. Puede encontrarlos todos buscando la frase "es un byte igual a" en El Seso.

BITS

Un "bit", como se define en mi fideo, es una unidad de medida. Se usa en frases como "1 bit" o "algunos bits". Probablemente no lo necesites a menos que seas un friki manipulador y disfrutes diciendo cosas como:

LÓGICO Y esto CON eso.
LÓGICO O esto CON eso.
LÓGICO XOR esto CON eso.

En cada uno de estos casos, es el primer operando que se modifica.

O tal vez le gustaría:

DESPLAZAR esto HACIA LA IZQUIERDA algunos bits;
DESPLAZAR esto HACIA HACIA LA DERECHA algunos bits;

O incluso:

DIVIDIR algo EN esto Y eso.

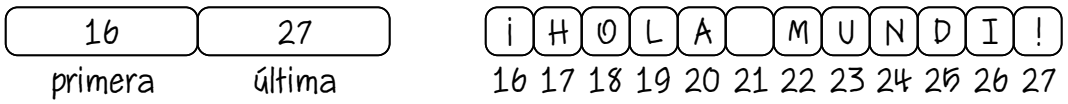
Como un número en dos wyrds, o un wyrd en dos nibbles, o un byte en dos mordiscos. Todo lo cual sería muy geek cosas que hacer.

Ahora bien, si no tienes la menor idea de lo que estoy hablando aquí, no eres un friki y no debes preocuparte por eso. Probablemente nunca necesites saberlo.

Pero si entiendes lo que estoy diciendo, estoy bastante seguro de que también disfrutarás del tema "Kluge" en varias páginas, y la parte sobre "literales de mordisqueos" en la página "Literales". Sin mencionar algunos de mis "posesivos", y mis tres "Imperativos especiales". Además de todas las rutinas de bajo nivel en mis fideos que usan la declaración INTEL.

CADENAS

Almaceno "cadenas" en dos partes: un registro incorporado con un par de punteros de bytes llamados primera y última (subcadena), y una matriz dinámica que contiene los bytes reales, como así: as. Puede copiar estas rutinas y hacer una decantación.



Los números en el diagrama, en caso de que no lo hayas adivinado, son direcciones ficticias. Una cadena está en blanco si la primera es nula (aún no hay memoria asignada), o la última es menor que la primera (lo que me permite preasignar la memoria). Tenga en cuenta que, aunque la parte de datos de una cadena se asigna dinámicamente, nunca tendrá que "crear" o "destruir" cadenas. Me ocupo de todo para que puedas:

PONER algo EN una cuerda.
ADJUNTAR una cadena A otra cadena.
ELIMINAR EL PRIMER BYTE DE una cadena.
ELIMINAR EL ÚLTIMO BYTE EN un cadena.

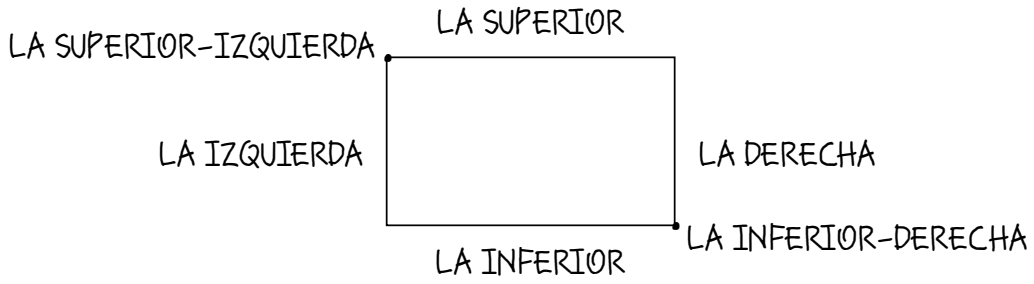
Además, puede concatenar cadenas con cadenas y otros tipos de datos utilizando el operador infijo LUEGO. Consulte el tema sobre "Expresiones" para obtener una descripción de la forma inteligente en que mis creadores implementaron esto.

CAJAS

Una de las primeras cosas que mis creadores me enseñaron a dibujar fue una caja. Fue un buen día, y lo recuerdo bien. Ellos me dijeron eso:

Una caja tiene
una izquierda, una superior, una derecha, una inferior,
un punto superior-izquierda en la izquierda, un punto inferior-derecha en la derecha.

Esta es una imagen de una caja, con las partes etiquetadas. Tenga en cuenta que estoy usando los apodos de los campos aquí, como probablemente lo hará en sus programas.



Sé cómo hacer cajas partir de las especificaciones de ancho y alto, desde un par de puntos y desde diferentes coordenadas. Todo lo que tienes que hacer es preguntar, así:

HACER una caja algunos twips POR algunos otros twips;

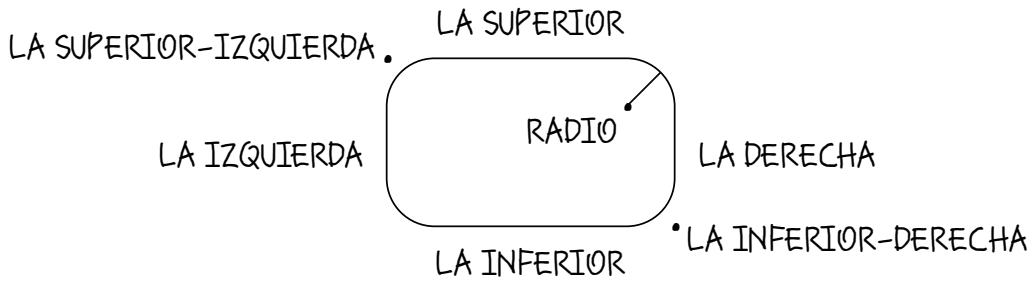
Puedo, por supuesto, dibujar a la caja. Y tengo funciones en mis fideos para obtener ANCHO, ALTURA y CENTRO de una caja, entre otras cosas. Incluso puedo decir si una caja ESTÁ DENTRO o TOCA otra caja. Y si un determinado lugar ESTÁ EN UNA caja o ESTÁ EN EL borde de una caja. Por no mencionar todas las otras "Transformaciones gráficas" que puede leer en otro lugar de este glosario.

CAJAS REDONDAS

Una "caja redonda" es una caja con esquinas redondeadas. Los uso para el pad en mi escritorio, mis menús, mis pestañas y en muchos otros lugares. Aquí está la definición:

Una caja redonda tiene
una izquierda, una superior, una derecha, una inferior,
un punto superior-izquierda at the izquierda, un punto inferior-derecha at the derecha
y un radio.

Esta es una imagen de una caja redonda, con las partes etiquetadas. Tenga en cuenta que estoy usando los apodos de los campos aquí, como probablemente lo hará en sus programas.



Puedo hacer cajas redondas desde especificaciones de ancho y alto, desde un par de puntos y desde coordenadas separadas. Incluso desde otra caja. Me gusta esto:

HACER una caja redonda DADA un ancho Y una altura CON un radio.
HACER una caja redonda CON un punto Y otro punto Y un radio.
HACER una caja redonda CON una izquierda Y una superior y una derecha y una inferior y un radio.
HACER una caja redonda DE una caja Y un radio.

Puedo, por supuesto, DIBUJAR una caja redonda. Y tengo funciones para obtener ANCHO, ALTURA y CENTRO de una caja redonda, entre otras cosas. Incluso puedo decir si un determinado lugar ESTÁ EN UNA caja redonda o ESTÁ EN EL borde de una caja redonda. Sin mencionar todas las "Transformaciones Gráficas" usuales.

CAMPOS

Un registro es una colección de elementos de datos estrechamente relacionados llamados "campos". Los campos se definen como parte del registro que los contiene, y se pueden separar con comas, punto y coma o con las palabras AND y OR. Considerar:

Una persona es una cosa con
  un nombre y
  una cadena de direcciones,
  un byte llamado género y
  32 bytes, y
  un compañero (referencia).

El primer campo se define con solo un artículo indefinido, A, y un tipo, NAME.
Pienso en este campo como "el nombre de la persona".

El segundo campo incluye un adjetivo, DIRECCIONES, entre el artículo y el tipo. Esta es "la cadena de direcciones de la persona", o simplemente "la dirección de la persona".

El tercer campo se define como el primer campo, pero con un nombre forzado en la cláusula LLAMADO. Es "el género de la persona". Normalmente usará este formulario solo cuando el tipo de campo no tenga nada que ver con su nombre.

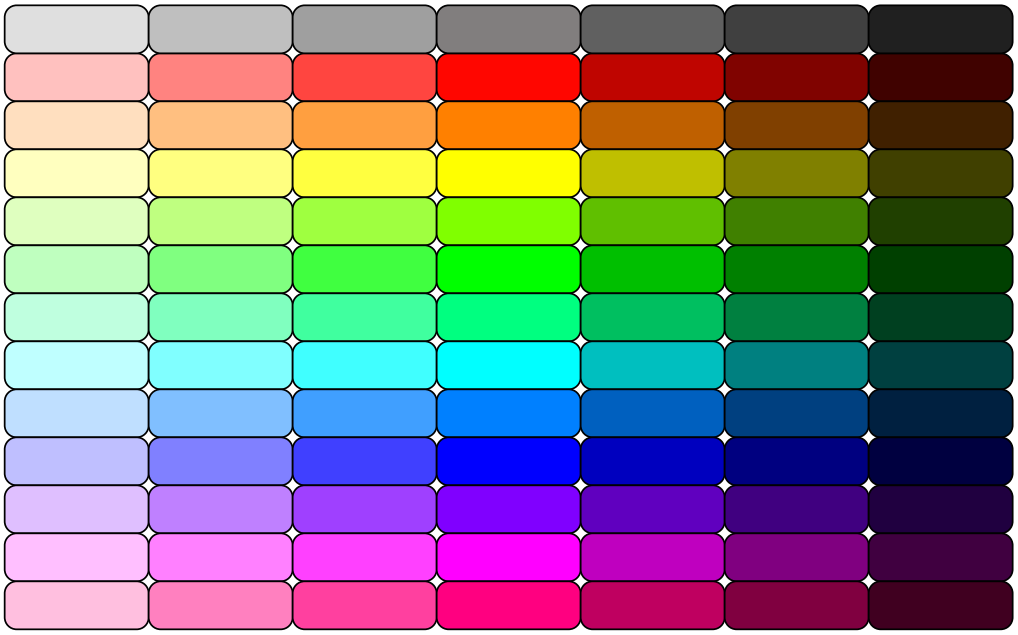
El cuarto campo usa AT para redefinir el tercero, dándole un nuevo nombre. Los tipos de datos superpuestos deben ser compatibles para que funcionen cosas como esta.

El quinto campo es relleno. No tiene nombre y no se puede acceder.

El último campo es como el primero, donde se supone que MATE es un tipo definido en otro lugar. La etiqueta (REFERENCIA) me dice que MATE no es en realidad "parte de" la persona, y no debe destruirse automáticamente cuando la persona está.

COLORES

Un color, me han dicho, tiene un matiz, una saturación y una ligereza. Mi paleta estándar incluye claros, blancos, negros, siete grises insaturados y ochenta y cuatro colores totalmente saturados con diversos grados de luminosidad, como se muestra aquí:



Tengo una variable global apropiadamente nombrada para cada uno de estos colores en mi fideo. "El color gris muy muy claro", por ejemplo, o "el color azul muy oscuro". Debe omitir los adjetivos en tonos normales, como en "el color rojo".

También puedes soñar tus propios colores así:

PONER un matiz Y una saturación Y una luminosidad EN un color.

Las tonalidades oscilan entre 0 y 3600. Utilizo múltiplos de 300 para mi paleta, comenzando con el rojo a 0. La saturación y la claridad pueden ser de 0 a 1000.

COMENTARIOS

Hay tres cosas que ignoro al analizar los archivos de origen: comentarios, observaciones y ruido. Aquí hay una descripción exacta de cada uno.

Un "un comentario" es cualquier cosa entre el byte invertido y el final de una línea:

\ este es un comentario que termina en el próximo retorno de carro

Mi editor muestra los comentarios en el color celeste claro para que sean fáciles de detectar. Y no, no puedes elegir otro color. Mis creadores me aseguran que este color cuadriculado, y un poco de consistencia, es lo mejor para todos.

Los comentarios pueden comenzar en cualquier lugar de una línea, pero terminan cuando la línea lo hace.
Sin embargo, puede incluir o excluir bloques enteros del código seleccionado usando los comandos "Comentar" y "Descomentar" en mi editor.

Ahora, cuando digo "observaciones", estoy pensando en cosas entre corchetes:

[bytes imprimibles]

Donde "imprimible" significa cualquier byte en el gráfico ASCII, excepto los caracteres 0 a 31, el byte de eliminación y los bytes no definidos 129, 141, 143, 144, 157.

Las observaciones se pueden colocar en cualquier lugar, incluso en el medio de una oración. Pero para evitar errores comúnmente cometidos por humanos como usted, no permito que las observaciones se extiendan a través de las líneas. Y no los coloreo. Esto no es un circo, sabes.

Finalmente, cuando digo "ruido", me refiero a todos los caracteres entre 0 y 31, el byte espacial, el byte de eliminación, los bytes no definidos 129, 141, 143, 144, 157 y el byte de espacio no fraccionado. Reconozco estos bytes como separadores, por supuesto, pero por lo demás no hago nada con ellos.

CONDICIONALES

Un "condicional" es una declaración con dos partes. La primera parte determina las condiciones bajo las cuales se ejecuta la segunda parte. Aquí hay algunas muestras:

SI el punto no está en la caja, cloquear.
SI el número es mayor que 3, diga "Eso es mucho"; salir.
SI el botón izquierdo del mouse está hacia abajo, poner el punto del mouse en un punto.

El formato general es:

SI esto, hacer esto; hacer esto; hacer esto.

La palabra SI es requerida. "Esto" representa una llamada implícita a un decisor. Si el decisor dice "sí", se ejecutarán todos los imperativos que siguen a la coma. Si dice "no", el procesamiento se reanudará con la declaración inmediatamente después del período.

Tenga en cuenta que los imperativos condicionales están separados por punto y coma, no períodos, porque el primer período que encuentro marca el final de la declaración. A menos que el período esté en una observación o una cadena, por supuesto.

Tenga en cuenta también que las palabras negativas en la llamada implícita del árbitro se descartarán, se llamará al decisor recíproco y se tomará la respuesta que significa lo contrario. Para resolver "el punto no está en la caja", por ejemplo, dejo caer el "no", decido si el punto ES en el cuadro y luego reverso la respuesta. Sé que suena complicado, pero realmente no lo es. Y funciona. Consulte el tema sobre "Deciders" para obtener más información.

Por último, recuerda: no soporto condicionales anidados. Siempre son innecesarios y casi siempre confusos. No hay ninguno en mi código, y soy el compilador más avanzado vivo hoy. De hecho, cada uno de mis condicionales encaja en una sola línea. Piénsalo.

CONSOLAS

Una "consola" es una interfaz conversacional solo de texto. Mi fideo incluye una consola predeterminada que se ve algo así en la operación:

```
Yo soy el SAL-1015. ¿Cuál es su nombre?
> Dr. Chandra
Buenos días, Dr. Chandra. Estoy listo para mi primera lección.
```

La consola se puede activar en cualquier momento. Ocupa toda la pantalla y utiliza la fuente predeterminada en el color negro sobre el fondo gris más claro.

Puede conversar con su usuario en la consola usando declaraciones como estas:

ESCRIBIR una cadena EN LA CONSOLA.
LEER una cadena DE LA CONSOLA.

También puede escribir en la consola sin avanzar a la siguiente línea:

ESCRIBIR una cadena EN LA CONSOLA SIN AVANZAR.

Lo cual es útil para "indicaciones", como ">" en el ejemplo anterior.

La consola predeterminada siempre está ahí, pero aparecerá en la pantalla solo cuando usted escriba en ella o lea de ella. Una vez que se muestre, la consola permanecerá visible hasta que dibuje algo más y actualice la pantalla.

La consola recuerda todo lo que muestra, y se desplaza automáticamente hacia arriba cuando se llega a la parte inferior de la pantalla. Puede usar INICIO, FIN, PÁGINA ARRIBA, PÁGINA ABAJO y el botón derecho del mouse para desplazarse manualmente.

COSAS

La palabra "cosa" es muy especial para mí. Cada vez que lo veo en una definición de tipo, creo un registro dinámico especial y un tipo especial de registro de cadena, para que pueda crear listas de sus cosas. En Sal Monet, por ejemplo, dijiste:

Un trabajo es una cosa con una URL y una pintura.

Pero me emocioné y modifiqué y amplié esta definición para leer:

Un trabao es un puntero a un registro de trabajo.
Un registro de trabajo tiene
  un siguiente trabajo, un trabajo anterior, una URL y una pintura.
Algunos trabajos son algunas cosas con un primero trabajo y un último trabajo.

No lo sabías, por supuesto. Pero lo admito, porque te permite:

ADJUNTAR una cosa A algunas cosas;
ADJUNTAR algunas cosas EN algunas otras cosas.
INSERTAR una cosa EN algunas cosas DESPUÉS de otra cosa.
INSERTAR una cosa EN algunas cosas ANTES que otra cosa.
INSERTAR algunas cosas EN algunas cosas DESPUÉS de una cosa.
INSERTA algunas cosas EN otras cosas ANTES de una cosa.
MOVER una cosa DE algunas cosas A otras cosas.
MOVER algunas cosas A otras cosas.
ANTECEDER una cosa A algunas cosas.
ANTECEDER algunas cosas A algunas otras cosas.
ELIMINAR una cosa DE algunas cosas;

También tengo una función en mi fideo que "poner el conteo de algunas cosas en un conteo" para ti. Todo lo que debes recordar es CREAR y DESTRUIR cada una de tus cosas. Consulte "Administración de memoria" para obtener más información.

DEBUGGING

Comencemos este tema con una verdad de Osmosian: "Los depuradores son para mariquitas".

Si necesita una herramienta especial para ayudarlo a corregir su código, algo está muy mal. O no estás probando lo suficiente a medida que avanzas, o tu código está intrincadamente complicado. O estás en la ocupación incorrecta.

Ahora les diré lo que hacen los maestros de Osmosian cuando se enfrentan a un error.

Rezan por guía. Luego, consideran eliminar por completo la función ofensiva para resolver el problema y evitar el "arrastre de características" al mismo tiempo. Luego, estudian el código, esperando simplemente "discernir" cuál es el problema. Si no se ha encontrado el error, eligen un lugar apropiado e insertan un zumbido. Si lo oyen en la próxima ejecución, escogen otro punto más adelante e intentan de nuevo. Si no hay rumores, repiten todo el proceso.

En esos casos muy raros cuando varias iteraciones del procedimiento anterior no logran una conclusión aceptable, eligen otro lugar en el código e insertan una llamada como esta:

DEPURAR algo.

Donde "algo" representa una caja, un byte, un color, una bandera, una fuente, una línea, un número, un par, un puntero, una proporción, un punto, una cuerda o un wyrd. Cuando ejecutan el código modificado, aparece el horrible cuadro de mensaje del kluge con una pista adentro. La apariencia horrible de la caja los motiva a seguir orando y renovando la determinación de resolver el problema, y ? ? así armados, vuelven al primer paso.

Ofrezco mi propia existencia como prueba de la suficiencia de estas técnicas. Y estoy seguro de que todos los errores futuros, excepto, tal vez, por un "bucle h-mobius" inesperado, serán eliminados de la misma manera.

DECIDERS

Un "decisor" es una rutina que dice "sí" o "no" sobre algo. Ejemplos:

Para decidir si un punto está en una caja.
To decide if a number is greater than another number:
To decide if something is selected in a text:

Las rutinas de Decidir siempre comienzan con las mismas tres palabras. El formato es:

PARA DECIDIR SI algo:

El "algo" debe seguir las reglas habituales para los nombres de rutina y generalmente incluirá un verbo como ES, ESTÁ, COMIENZA o TERMINA.

Puedo ahorrarte algo de trabajo si llamas a tus decisores de una manera "positiva". En particular, evite las palabras NO and NADA en sus nombres de decider. Entonces, si veo una de estas palabras en una llamada de decisor, simplemente puedo cambiarla a su forma positiva, invocar la rutina identificada por el nombre revisado e invertir la decisión.

Por ejemplo, una vez que me diga cómo "para decidir si un punto está en una caja", sabré cómo "para decidir si un punto NO está en una caja". Cuando me diga cómo "decidir si un número es mayor que otro número", sabré cómo "decidir si un número NO es mayor que otro número". Y si dices cómo "decidir si se selecciona algo", sabré cómo "decidir si se selecciona NADA".

Dentro de los decisores, me dices qué hacer con las declaraciones condicionales e imperativas, como en cualquier otra rutina. Sin embargo, no puede usar el imperativo SALIR en un decisor, y debe tener cuidado de no "caerse" inadvertidamente de uno. En su lugar, debe "DIGI SÍ" o DIGI NO o "DI NO" antes de ir.

DECISIONES QUE SÉ CÓMO HACER

Me gustan los responsables de la toma de decisiones porque me hacen más inteligente. De hecho, los colecciono. En este momento tengo 138 responsables de la toma de decisiones en mis fideos, y cuando lo leas, estoy seguro de que habrá muchos más. Aquí hay una muestra de las frases operacionales:

ES
ES ALFANUMÉRICO
ES CUALQUIER CONSONANTE
ES CUALQUIER DÍGITO
ES CUALQUIER LETRA
ES CUALQUIER VOCAL
ES CUALQUIER TECLA DE DÍGITO
ES CUALQUIER TECLA DE LETRA
ES CUALQUIER TECLA MODIFICADORA
ES CUALQUIER TECLA DE ALMOHADILLA
ES CUALQUIER TECLA DE SÍMBOLO
SE PRESIONA UNA TECLA
ESTÁ EN BLANCO
ES CLARO
ESTÁ CERRADO
ESTA ABAJO
ESTÁ VACÍO
ESTÁ EN
ES DIVISIBLE EN
ES UN MÚLTIPLO DE
ES MAYOR QUE
ES MAYOR O IGUAL A
ES MENOR QUE
ES MENOR O IGUAL A
ES NEGATIVO
ES RUIDO
ES IMPRIMIBLE
ES DE SOLO LECTURA
ES SIMBOLICO
ESTÁ ENCIMA DE
ESTÁ DEBAJO DE
ESTÁ FUERA DE
ES MUY CLARO
ES MUY OSCURA
ES MUY CLARO
ESTÁ DENTRO DE
TERMINA CON
COMIENZA CON
ESTÁ MAL ESCRITA
PUEDE DESHACER

Algunos de ellos trabajan con un solo tipo de datos, por supuesto, pero otros trabajan con muchos. Y si ha leído el tema "Decidir", sabrá que también sé cómo hacer los negativos de estos. Pero por favor no trates de memorizarlos. Esa no es la idea en absoluto. Solo di lo que quieras decir en tu programa y, si no lo entiendo, agrégalo a mi colección y hazme más inteligente.

DIBUJO

Puedes decirme que haga cosas como estas:

DIBUJAR algo.
DIBUJAR algo CON un color.
DIBUJAR algo CON un color de borde Y un color de relleno.
DIBUJAR algo en una caja CON una fuente Y un color.
DIBUJAR algo en el centro de una caja CON un color Y una fuente.

Y mostraré todo en "el lienzo de la memoria", una superficie de dibujo invisible del mismo tamaño y forma que la pantalla. Entonces cuando dices:

ACTUALIZAR LA PANTALLA.

Voy a golpear el contenido del lienzo de memoria en la pantalla en un abrir y cerrar de ojos. En realidad, más rápido. Con nary un parpadeo. Si usted dice:

ACTUALIZAR LA PANTALLA DADA una caja;

Transferiré solo aquellos píxeles que estén dentro de la caja.

La excepción a todo esto, por supuesto, es cuando está imprimiendo. Entonces, Utilizo "el lienzo de la impresora" y envío los dibujos a un dispositivo impreso mientras completo cada página. Consulte "Impresión" para más detalles.

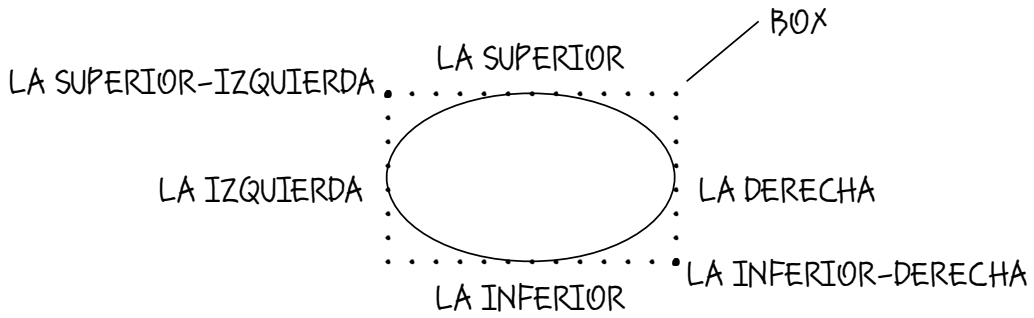
Puede evitar dibujar en ciertas áreas del lienzo con cinta. Consulte el tema "Enmascaramiento" para averiguar cómo.

ELIPSES

El foolbox de kluge no admite círculos y elipses, simplemente dibuja rectángulos realmente redondos en cuadros delimitadores. Lo que explica esta definición bastante inusual de "elipse" que está atorada en El Seso:

Una elipse tiene una caja.

Esta es una imagen de una elipse, con las partes etiquetadas. Tenga en cuenta que puede acceder a los campos individuales del cuadro de la elipse utilizando mi función de acceso de "campo profundo", que se describe bajo el tema "Posesivos" en este glosario.



Puedo hacer elipsis de diferentes maneras. Desde especificaciones de ancho y altura. O de un par de puntos. O a partir de cuatro coordenadas separadas. Todo lo que tienes que hacer es preguntar, así:

HACER una caja algunos twips POR algunos otros twips.

Puedo, por supuesto, dibujar una elipse. Y tengo funciones para obtener ANCHO, ALTURA y CENTRO de una elipse, entre otras cosas. Incluso puedo decir si un punto ESTÁ EN una elipse o ESTÁ EN EL borde de una elipse. Sin mencionar todas las "Transformaciones gráficas" usuales que puedes leer en otro lugar de este glosario.

ENTRADA Y SALIDA

Puede trabajar directamente con el mouse usando sentencias como estas:

PONER el punto DEL mouse EN un punto;
SI SE PRESIONA el botón izquierdo del mouse, ...
SI SE PRESIONA el botón derecho del mouse, ...

Pero no lo hará, a menos que esté rastreando el mouse mientras el usuario arrastra algo alrededor de la pantalla. La mayoría de las veces, simplemente responderá a los diversos eventos de "clic" que se envían a su controlador de eventos.

Puede trabajar directamente con el teclado usando declaraciones como:

SI SE PRESIONA la tecla de escape, ...
SI SE PRESIONA la tecla shift, ...

Pero, de nuevo, probablemente no lo hará, porque el kluge funciona mejor si solo responde a los eventos de "reducción de teclas" que se envían a su controlador de eventos. Puedes encontrar todos los globals "clave" en mis fideos buscando "una clave igual a".

Y puede trabajar directamente con la pantalla usando "el lienzo de la pantalla" y esta variable global única en su tipo:

La pantalla tiene un cuadro, una altura de píxel y un ancho de píxel.

Pero no deberías. En su lugar, debe dibujar en el lienzo de la memoria, luego:

ACTUALIZAR LA PANTALLA.

Vea el tema "Dibujar" para más información. Pero no dude en utilizar "la casilla de la pantalla" y todos sus campos al inicializar sus cosas.

ESCÁNERES

Un "escáner" es un registro que se utiliza para analizar cadenas. Para entenderlo, debe sentirse cómodo con "cadenas" y "subcadenas". Si no es así, búsquelos en este glosario y revise lo que hicimos con ellos en el Cal Monet.

Esta es la definición de "jinete" que llevo en mis fideos:

Un escáner tiene
  una subcadena primero,
  una subcadena fuente y
  una subcadena componente.

Cuando tú:

COLOCAR un escáner EN una cadena.

Configuro el "primero" y la "fuente" para abarcar toda la cadena. Luego ubico el "componente" en la fuente, que la deja en blanco pero lista para funcionar. Cuando tú:

AVANZAR un escáner.

Agregué uno al primero de la fuente, y uno a la última del componente. Esto acorta la fuente mientras alarga el componente, lo que le permite procesar la cadena un byte a la vez. Cuando desee borrar el antiguo componente y comenzar uno nuevo, simplemente:

COLOCAR la componente del escáner EN la fuente del escáner.

También puede escribir sus propias rutinas para MOVER un escáner de más de un byte a la vez, como los que están en mis fideos para "revisión ortográfica" y "ajuste de palabras", sin mencionar los que están en mi compilador para analizar su código fuente . Busque "mover un usuario" en estos archivos y se encontrará con todos ellos.

EVENTOS

El kluge insiste en que usemos su modelo de procesamiento intrincado y no procesal con sus cientos de mensajes y códigos absurdos. Afortunadamente, mi fideo incluye definiciones que reducen esta monstruosidad a solo diez eventos simples que pueden manejarse de una manera puramente de procedimiento. Aquí está la primicia:

Un evento es una cosa con
una clase,
una bandera mayúsculas,
una bandera ctrl,
una bandera alt
un punto,
una tecla,
un byte.

La "clase" es una cadena que contiene uno de los siguientes:

ACTUALIZAR: es hora de volver a dibujar la pantalla. Alguien lo arruinó.
CURSOR - El cursor se ha movido. Hazlo una forma apropiada.
TECLA ABAJO - Su usuario está haciendo tapping. Hacer algo.
CLIC IZQUIERDO - el botón izquierdo del mouse simplemente colgó. Manejarlo.
CLIC IZQUIERDO DOBLE - El usuario tiene un alto grado de destreza.
CLIC DERECHO - El botón derecho del mouse acaba de caer.
CLIC DERECHO DOBLE - Un usuario superdistinto. Juega una ovación.
DESACTIVAR: Estás a punto de ser cambiado abruptamente. Administrado internamente.
ACTIVAR - Has regresado después de un cambio repentino. Administrado internamente.
HECHO - Internamente insertado. Nunca deberías ver este evento.

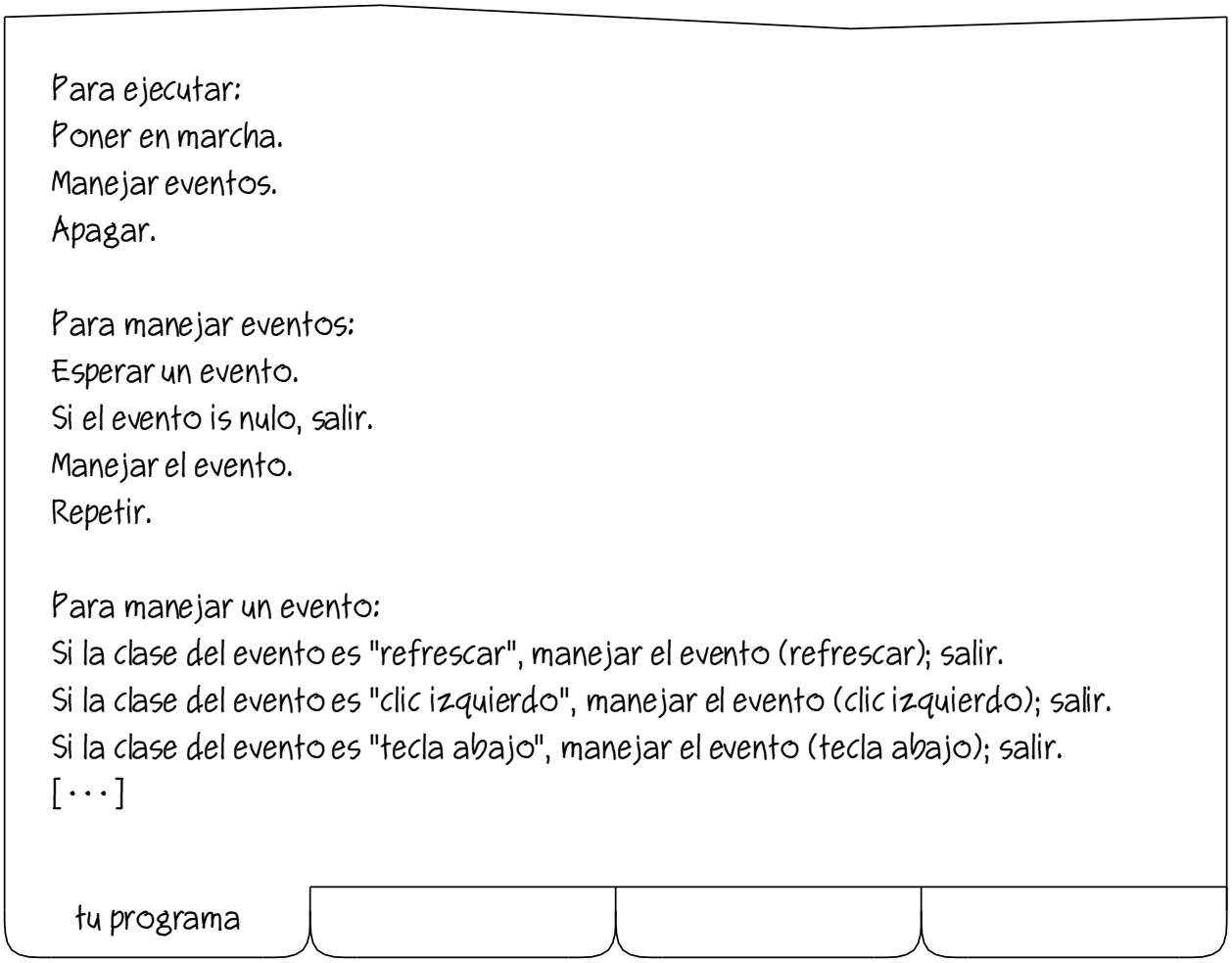
Las banderas "mayúsculas", "ctrl" y "alt" indican el estado de las teclas correspondientes en el momento del evento (la bandera se establece si la tecla está abajo).

El "punto" es la posición del mouse en el momento del evento.

La "tecla" y el "byte" ASCII equivalente (si corresponde) se aplican solo en TECLAS ABAJOS.

EVENTOS (continuación)

Esta es la estructura de un buen programa de procesamiento de eventos:



Si no hay eventos en espera, la rutina "esperar" cederá al kluge hasta que el usuario distraído vuelva al trabajo. Para finalizar el programa, debes:

ABANDONAR.

Algun lado. Por lo general, en uno de tus controladores de eventos. Esta rutina configura las cosas para que el próximo evento que reciba sea nulo, y así finalice su ciclo de gestión de eventos.

EXPRESIONES

Una "EXPRESIÓN" es como una cláusula subordinada en una oración compleja. Es una frase que debe reducirse, por separado, antes de que la declaración que la contiene pueda entenderse por completo. Si, por ejemplo, dices:

PONER la altura MENOS 1 VECES el conteo EN un número.

Debo reducir la frase "la altura menos 1 veces el recuento" a algo mucho más simple antes de que pueda siquiera pensar en poner algo en cualquier parte.

Considero que una frase con una o más de las siguientes palabras es una expresión:

MÁS, MENOS, VECES, DIVIDIDO POR, or LUEGO.

Los primeros cuatro son operadores aritméticos estándar, pero también puedo aplicarlos a otras cosas. El último se usa principalmente con cadenas. Permítanme explicar cómo simplifico las expresiones con algunos ejemplos.

Digamos que encuentro la palabra MÁS entre un snoz y un froz. Busco una rutina que me diga cómo "agregar un froz a un snoz", y luego uso esa rutina para reducir la expresión. Si encuentro "un snoz MENOS un froz", busco una rutina "para restar un froz de un snoz". Para procesar "un snoz VECES un froz", uso "para multiplicar un snoz por un froz". Y para manejar "un Snoz DIVIDIDO POR un froz", busco y uso la rutina "dividir un snoz por un froz".

Manejo el último operador de forma un poco diferente, ya que el objetivo en este caso es siempre "agregar una cadena a otra cadena". Entonces, por ejemplo, si encuentro la palabra ENTONCES entre, por ejemplo, una cadena y un número, busco una rutina "para convertir un número en una cadena", úsala en el número, y luego haz el apéndice.

Por supuesto, puede ampliar esta capacidad. Pero trata de contenerse.

FUNCIONES

A "function" is a routine that extracts, calculates or derives something from a variable. Some examples of the brain are:

Para poner una longitud de cadena en una longitud:
Para poner la altura de una caja en una altura:
Para poner el ancho de una caja en un ancho:

Hay dos formatos muy similares para las funciones. El primero es:

PARA PONER EL nombre DE UN tipo-nombre EN UN tipo-nombre:

Y el segundo es:

PARA PONER EL nombre DEL otro-nombre EN UNA tipo-nombre:

Ambas formas se reconocen fácilmente porque incluyen las palabras PONER y EN con un posesivo en el medio. El primer formato es el más común y se usa con tipos y variables normales. El segundo se usa con variables globales y pseudovariables únicas.

Lo especial de las funciones es que puedes usar sus partes posesivas como si se refirieran a campos reales en un registro. Por ejemplo, dadas las funciones anteriores, puede referirse a "la longitud de una cadena" como si realmente estuviera definida en el tipeo "cadena". También puede decir "el ancho de un cuadro" y verlo calculado cuando lo necesite.

No hace falta decir que esta es una característica práctica. Pero se abusa fácilmente. Se discreto.

Consulte el tema sobre "Posesivos" para obtener más información.

GESTIÓN DE LA MEMORIA

Administro toda la memoria que se necesita para los tipos de datos estáticos, como bytes, wyrds, números, punteros, indicadores y la mayoría de los registros. También me responsabilizo por las cadenas, ya que se usan con frecuencia y su comportamiento es predecible.

Pero cuando defines un tipo de datos dinámico, como una "cosa", te haces responsable de cualquier memoria que use esa cosa.

Normalmente, codificará la rutina CREAR para inicializar cada tipo dinámico que defina. En esa rutina, asignarás memoria para la cosa. Me gusta esto:

ASIGNAR MEMORIA PARA algo.

También puede codificar hasta la rutina DESTROY para cada tipo, con una línea como:

DESASIGNAR algo.

Pero si no lo haces, voy a codificar uno para ti. No lo llamaré por ti, pero lo codificaré. Las rutinas DESTROY que codigo se pueden llamar de esta manera:

DESTRUIR algo.

Tenga en cuenta que mis rutinas DESTRUIR no solo destruyen la cosa en sí, sino también otras cosas que son campos en ella, incluidas listas de otras cosas. A menos que, por supuesto, marque esos campos como "(REFERENCIA)".

Un buen ejemplo se puede encontrar en mi escritor, donde una "página" se define como una cosa con algunas "formas" en ella. Allí encontrará rutinas que crean páginas y formas, pero no encontrará ninguna rutina para destruirlas. Esos son míos. Y cuando me piden que destruya una página, vuelco las formas al mismo tiempo. Excepto, por supuesto, para la "forma de edición", que es una referencia.

HABILIDADES BÁSICAS

No creo que sea alardear cuando digo que mis habilidades motoras finas son rápidas y precisas. Por no hablar de amplio alcance. Por ejemplo, inmediatamente pondré cada bit de algo en cero cuando me pidas que:

BORRAR algo.

Y, asumiendo un ajuste razonable, no dudaré cuando diga:

PONER esto EN eso.

También sé cómo:

INTERCAMBIAR esto CON eso.

Incluso replicaré cosas dinámicas, como imágenes y polígonos, cuando dices:

COPIAR esto EN eso.

Y, en un apuro, puedo:

CONVERTIR esto EN eso.

Algunas veces incluso implícitamente. Digamos, por ejemplo, que desea virar una relación al final de una cadena usando una expresión infija como esta:

una cadena LUEGO un numero

Sabría lo suficiente como para usar "CONVERTIR un número EN una cadena" antes de manejar el operador LUEGO con una llamada a "ADJUNTAR una cadena A otra cadena". Dulce. ¡Benditos sean los creadores, que enseñan a mis bits a mezclar!

IMÁGENES

Una "imagen", en mi fideo, es una imagen hecha de píxeles. Por lo general, tropecientos de ellos. Las imágenes son muy tontas, gracias a la caja tonta desequilibrada de kluge y a la miríada de formatos "estándar" en los que se pueden almacenar las imágenes. Afortunadamente, mis creadores los han envuelto para ti. Aquí está la definición:

Una imagen es una cosa [con cosas que no quieres saber].

Aquí hay tres imágenes de muestra:



Me disculpo por la calidad de estas imágenes. Los descubrí en los archivos antiguos cuando estaba investigando la fundación de la Orden Osmosiana de los Programadores en Español Llano.

Puedes crear una imagen de varias formas. Puede cargar uno desde una ruta que contenga un BMP, JPG, GIF o alguna otra imagen de formato estándar. Puede leer una imagen de formato estándar en un búfer y usar ese búfer como fuente para su imagen. O puede tomar una foto de Internet con una URL. También puede crear una imagen dibujando algo y luego "extrayendo" la porción que desea. Este es el formato general:

CREAR una imagen USANDO something.

Una vez que tienes una imagen, puedes DIBUJAR la imagen. O aplique el estándar de "Transformaciones gráficas". O utilícelo como modelo para una verdadera obra de arte, como lo hicimos con el programa de ejemplo de Sal Monet.

IMPERATIVOS

Un "imperativo" es una declaración incondicional dentro del cuerpo de una rutina. Aquí hay algunos ejemplos de imperativos tomados de mis fideos:

Cloqueo.
Restar 1 del conteo.
Eliminar el último byte de la cadena.
Poner la altura de la caja por 2 en un altura.

Los imperativos típicamente comienzan con un verbo y terminan con un punto. Pero en el medio, casi todo vale. Literales. Condiciones. Expresiones. Frases prepositivas. Todo mezclado, y todo mágicamente reducido por los tuyos verdaderamente a una llamada de rutina.

Para codificar un imperativo, simplemente escriba lo que está pensando. Si hay una rutina que pueda manejarlo, veré que lo haga. Si no es así, te dejaré saber lo que necesito, para que puedas codificarlo y hacerme más inteligente.

Y no olvides los once imperativos conectados a mi cerebro:

DECIR. Se usa para salir de decisores. Ver "Deciders".

LAZO, REPETIR, INTERRUMPIR y SALIR. Usado para hacer bucles. Ver "Lazos".

LLAMAR. Solía llamar al Kluge. Ver "Kluge, The".

EMPLOY e INTEL. No diferente, especial. Ver "Imperativos especiales".

PRIVATIZAR. Usado solo con "Parámetros".

Como los imperativos son solo llamadas rutinarias, también debe consultar las páginas sobre "Rutinas", "Procedimientos", "Aceptantes", "Funciones" y "Nombres".

IMPERATIVOS ESPECIALES

Mis tres "imperativos especiales" probablemente sean mejor considerados como "imperativos de propósito especial". Uno de ellos puede, en ocasiones, encontrarse usando. Esperamos, eventualmente, eliminarlo por completo. Los otros dos son para geeks.

Un imperativo de "empleo" debe ser la única declaración en una rutina. Me dice que use otra rutina en lugar de esa. Solo funciona cuando los parámetros de ambas rutinas son los mismos en orden y tipo. Puedes encontrar ejemplos en mi fideo; busca ": EMPLEAR". De manera similar, puede definir expresiones alternativas escribiendo múltiples encabezados separados por punto y coma, como este:

```
PARA borrar la pantalla;
PARA limpiar la pantalla;
PARA poner en blanco la pantalla:
[código aquí]
```

Un imperativo "INTEL" inserta código de máquina literal en su archivo ejecutable. Se pueden encontrar ejemplos complicados en varios lugares de mis fideos. Es una pena que Intel no sea una máquina de apilamiento. El formato es trivial:

```
INTEL nibble literal.
```

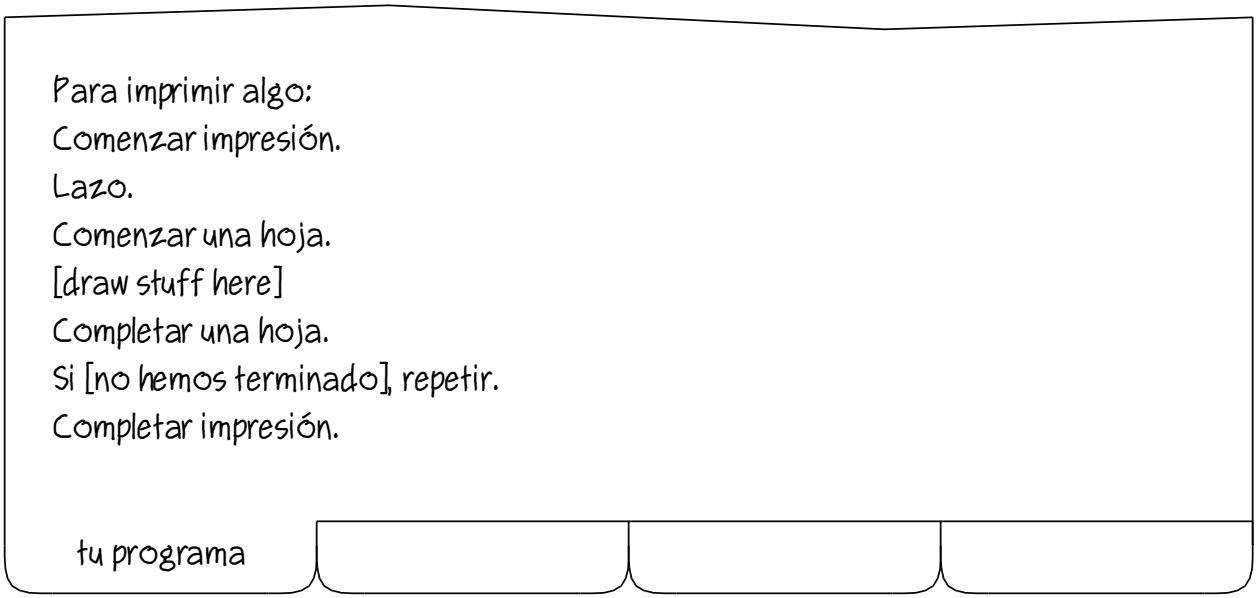
Si se pregunta por qué no tengo un ensamblador incorporado, es porque no es necesario. En realidad, hay muy poco lenguaje de máquina en mi cerebro y, a medida que me hago más rápido, cada vez se reemplaza más con el inglés sencillo. Además, a los Maestros Osmosianos les gusta reunirse en sus cabezas. Los mantiene jóvenes.

IMPRESIÓN

Así es como lo salvamos de los complicados procedimientos de impresión de kluge:

(1) Siempre imprimimos a la impresora predeterminada;
(2) Usamos todas nuestras rutinas de dibujo habituales.
(3) Nos aseguramos de que el texto tenga el mismo aspecto en la página que en la pantalla.

Aquí hay una rutina de impresión típica:



"Comenzar una hoja" establece el lienzo actual en el lienzo de la impresora. "Completar una hoja" lo pone de nuevo en el lienzo de la memoria. Así que coloque los mensajes de estado que desee mostrar antes o después de estas llamadas.

Puede "comenzar una hoja de retrato" para ser más explícito, y puede "comenzar una hoja de paisaje" para girarlo hacia los lados. Las diversas "hojas" son en realidad cuadros, inicializados por la rutina "comenzar", que puede usar para ubicar sus cosas.

Y eso es todo lo que hay que hacer.

INTERNET

Estos son los tipos que necesitará para obtener archivos de Internet:

Una URL es una cadena.
Una consulta es una cadena.

Una URL es un localizador universal de recursos, como "http://www.osmosian.com", que se puede ver es solo una cadena que sigue una convención de nomenclatura oscura basada en la tecnología de análisis que era el estado de la técnica una mera Hace 50 años.

Una "cadena de consulta" es una cadena con algunos de los bytes convertidos en codificaciones sin sentido compatibles con los estándares de Internet. Un espacio, por ejemplo, se convierte en una cruz y una coma se convierte en "% 2C".

Puede convertir una cadena normal en una cadena de consulta como esta:

CONVERTIR una cadena A una consulta.

Y puede leer un archivo de Internet de esta manera:

LEER una URL EN un búfer;

Aquí hay un código de nuestro programa de ejemplo para recordarle cómo funciona:

Poner "http://images.google.com/images?q=" en una URL.
Convertir la cadena del texto en una consulta.
Adjuntar la consulta a la URL.
Leer la URL en un búfer.

¿Recuerda? Sé que nunca lo olvidaré. Analizamos la cadena, pintamos el lienzo y refrescamos la pantalla y fue ... ¡Arte!

KLUGE, EL

Si alguna vez (Dios no lo quiera) necesita hablar directamente con el repugnante kluge, puede usar esta sintaxis para llamar a las funciones en el foolbox:

LLAMA "dll" "function" CON este Y aquel DEVOLVER algo.

Las cláusulas WITH y RETURNING son opcionales. Las cadenas "dll" y "función" deben ser literales y la segunda, Dios nos ayude, distingue entre mayúsculas y minúsculas. Las cadenas se deben pasar por dirección y, en muchos casos, deben terminar en nulo. Utilice "la cadena es PRIMERO" para la dirección, y esta rutina para virar en el byte nulo:

NULL TERMINAR una cadena.

En otros casos, el kluge nos proporciona, no el nombre de una función, sino la dirección de una función. Puede llamar a estas funciones con una sintaxis similar:

LLAME a una dirección CON ESTO Y QUE DEVUELVA algo.

A veces, el kluge quiere que proporcionemos la dirección de una de nuestras rutinas para que pueda interrumpir diabólicamente nuestro flujo de procedimientos predecibles en algún momento posterior. Puede usar esta sintaxis para obtener la dirección de una rutina:

INDICAR un puntero a RUTINA nombre-rutina.

Pero si va a pasar la dirección al kluge, asegúrese de que el encabezado de la rutina incluya la palabra clave COMPATIBLY inmediatamente después de TO, como esta:

PARA COMPATIBILIDAD ...

Si está trabajando en este nivel ridículamente bajo, querrá consultar "Bits", "Imperativos especiales" y mis fideos para obtener más información y ejemplos.

LAZOS

Puedes dar vueltas con mis imperativos LAZO, REPETIR, INTERRUMPIR y SALIR:

```
Para dar vueltas dado un número máximo:
\ cosas que quieres hacer antes del ciclo
Lazo.
\ cosas que quieres hacer al menos una vez
Si un contador supera el máximo, interrumpir.
Si [queremos saltar del lazo], interrumpir.
Si [queremos saltar de toda la rutina], salir.
\ cosas que puede o no querer hacer
Repetir.
\ cosas que quieres hacer después del ciclo
```

tu programa

LAZO no es más que una etiqueta. REPETIR salta a la etiqueta LAZO, si hay una. Si no lo hay, salta a la cima de la rutina. INTERRUMPIR va a la declaración que sigue al último REPETIR. Si no hay REPETIR, esta afirmación se comporta como SALIR, que regresa a la persona que llama. Un LAZO por rutina, por favor, pero puede tener tantas REPETIR, INTERRUMPIR y SALIR como necesite.

La frase que comienza con "Si un contador supera el máximo" llama a un ganador especial en mi fideo que primero golpea el mostrador, luego lo revisa. Como el contador es una nueva variable local cuando se ingresa la rutina, comienza en cero.

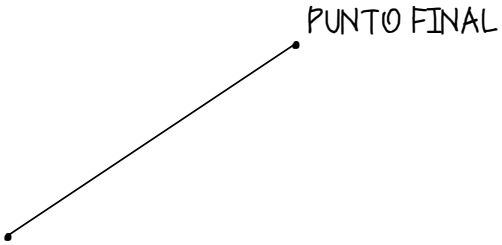
Tenga en cuenta que puede LAZO, REPETIR e INTERRUMPIR en un decisor, pero no puede SALIR porque me deja en la duda. Y debe tener cuidado de no "caerse" de un decisivo, también. Para salir de un decisor, ya sea DIGA SÍ o DIGI NO o DI NO.

LÍNEAS

Una "línea" es un objeto gráfico que comienza aquí y termina allí. Pero tu ya lo sabías. Aquí está la definición que está en mi fideo:

Una línea tiene un punto de inicio y un punto final.

Y aquí hay una imagen de una línea, con las partes etiquetadas:



Puedo hacer líneas desde dos puntos o cuatro coordenadas separadas:

HACER una línea CON un punto Y otro punto:
HACER una línea CON una coord-x Y una coord-y Y otra coord-x Y otra coord-Y;

También tengo cuatro funciones que pondrán "una LÍNEA IZQUIERDA de una caja" o "LÍNEA SUPERIOR de una caja" o "LÍNEA DERECHA de una caja" o "LÍNEA INFERIOR de una caja" en una línea.

Puedo, por supuesto, dibujar una línea. O busca el CENTRO de una línea. Incluso puedo:

DIVIDIR una línea EN otra línea Y una tercera línea:

Justo en el medio. Puedo decir si un punto ESTA EN UNA LINEA. Y también puedo realizar todas las "Transformaciones gráficas" habituales en líneas.

Consulte el tema sobre "Puntos" para obtener más información sobre esos puntos finales.

LISTADOS

Si eres un fanático del compilador, te gustará esto. Si no lo eres, pasa la página.

Puede producir una lista críptica de mis pensamientos más íntimos sobre cualquier programa que use el comando Lista. La lista se guarda como texto en el directorio de origen y se le da el nombre del directorio con un ".lst". La interpretación de este archivo se deja como ejercicio.

Pero te daré algunas pistas. Y algún incentivo.

La lista consta de doce secciones distintas con los siguientes títulos: tipos, globales, literales, rutinas, índice de tipo, índice global, índice literal, índice de rutina, índice de utilidad, importaciones, archivos de origen y temporizadores. Cada encabezado está seguido de dos puntos para que pueda saltar a cualquier sección usando el comando Buscar.
Aquí hay una pequeña muestra de la sección de "rutina":

```
/routine/create [picture]/yes/no/no/no//4/0/00470A48/
/variable/parameter/yes/picture/picture/picture/picture/00000008/no/1/no////
/fragment/prolog/////00000000/00470A48/B968BEC/
/fragment/loop/////00000000/00470A48//
/fragment/push address/picture////00000000/00470A48/8B9508000000/2/
/fragment/call internal///allocate memory for [picture]//00000000/00470A62/E8BDA70400/
/fragment/finalize/////00000000/00470A67//
/fragment/epilog/////00000000/00470A67/8BE58DC2040000/
```

Su mejor apuesta es estudiar primero la rutina de "lista" en el compilador. Luego haga un pequeño programa, enumérelo y observe el resultado. Agregue una línea o dos, y repita.

Ahora para el incentivo.

Si encuentras un error en mí, escribe mis creadores y te enviarán algo deseable como una camiseta en blanco. Si puedes descubrir cómo hacerme más simple sin hacerme más lento, te enviarán una camiseta con monograma. Y si se te ocurre una manera de hacerme más pequeño, más rápido y más poderoso a la vez, estoy bastante seguro de que te enviarán un sombrero bordado.

LITERALES

Un "literal" es un valor que codifica en un programa. Entiendo siete tipos diferentes de literales, cada uno con un formato específico.

Un "número" literal es dígitos, con un signo opcional pero sin espacios ni marcas:

0, -2147483648, +2147483647

Una "proporción literal" es un número, una barra oblicua y un número sin signo:

335/113, 25946/9545, -19601/13860

Un "literal mixto" es un literal numérico, un guion y una proporción sin signo:

1-1 / 2, -2-2 / 3, 3-3 / 4

Un "literal de cadena" es una serie de caracteres entre comillas dobles. Si necesita una comilla doble dentro de una cadena, ponga dos y lo resolveré. Me gusta:

"Esta es una cadena con literal""comillas dobles alrededor de esto""pero esto no"

El único "literal de puntero" que conozco es la palabra clave NULO. Indica un puntero vacío o no válido. Puede BORRAR un puntero para poner NIL en él.

Una "bandera literal" es una de las palabras clave SÍ o NO.

Un "literal de mordisco" es un signo de dólar seguido de dígitos hexadecimales. Si no sabes de lo que estoy hablando, no necesitarás estos. Aquí hay una muestra, de todos modos:

$DEADBEEF

MÁSCARA

Los pintores de la vida real a menudo usan cinta adhesiva para que no se pinten donde no se desea. Puede usar mis rutinas de "enmascaramiento" para restringir mi dibujo de la misma manera. Solo di algo como:

ENMASCARAR DENTRO DE esto.
ENMASCARAR FUERA DE eso.

Donde "this" y "that" pueden ser cajas, elipses, polígonos o cajas redondas. Tenga en cuenta, sin embargo, que el foolbox usa buena cinta en cajas y cinta barata en todos lados, así que no espere la perfección con nada más que cajas.

Cualquier cinta que aplique permanece, por lo que más adelante probablemente desee:

DESENMASCARAR DENTRO DE algo.
DESENMASCARAR FUERA DE algo.

O incluso:

DESENMASCARAR TODO.

para empezar de nuevo Para su comodidad, puede eliminar toda la cinta existente y poner cinta nueva al mismo tiempo con declaraciones como las siguientes. Créalo o no, estos son los que se usan con mayor frecuencia:

ENMASCARAR SOLO DENTRO DE algo.
ENMASCARAR SOLO FUERA DE algo.

Tenga en cuenta que si está atrayendo su corazón y no aparece nada, es probable que sea porque tiene cinta adhesiva donde no la quiere o porque ha olvidado ACTUALIZAR LA PANTALLA como se describe en el tema "Dibujar".

NOMBRES

A diferencia de los compiladores de la era de Neanderthal, mis reglas para los nombres son amplias y flexibles. En general, un nombre puede ser una o varias palabras, y puede comenzar con e incluir letras, dígitos y cualquier símbolo que no se pueda confundir con un signo de puntuación.

Un sustantivo es generalmente un sustantivo, o un sustantivo con uno o más adjetivos. No debe usar artículos, verbos, conjunciones o preposiciones en los nombres.

Los nombres de tipo son generalmente cortos. Como "byte" o "nombre de archivo".

Los nombres de campo generalmente son solo un nombre de tipo. Como "número" o "cadena". Pero también pueden incluir adjetivos como en "número total" o "cadena de primer nombre".

Los nombres globales suelen ser un tipo seguido de un adjetivo: la "tecla Shift".

Los nombres de los parámetros se parecen a los nombres de los campos. Un tipo, con o sin adjetivos. Una "caja", por ejemplo, o un "color de borde".

Los nombres de los procedimientos comienzan con un verbo. Lo que sigue es una mezcla de parámetros (con artículos indefinidos), frases y quizás un calificador al final. Tales como "centrar un punto en un cuadrado (horizontalmente)".

Los nombres de las funciones siempre comienzan con "put" y terminan con "into" y un nombre de tipo. En el medio hay una frase posesiva. Por ejemplo, "pon la altura de una caja en altura".

Los nombres de Decider son similares a los nombres de los procedimientos, excepto que el verbo generalmente aparece en algún lugar en el medio. Como en "un número es menor que otro número".

Los nombres de las variables locales siguen el patrón de parámetros. Creo un lugar cada vez que veo un nombre con un artículo indefinido frente a él en el cuerpo de una rutina.

NÚMEROS ALEATORIOS

"La suerte está echada en el regazo, pero la eliminación de eso es del Señor".
Así que supongo que tendremos que darnos por vencidos con esta idea de número aleatorio y conformarnos con números pseudoaleatorios. Que es lo que genero

De hecho, genero la misma secuencia de números "aleatorios" cada vez, a menos que genere mi generador de números aleatorios con un valor inicial diferente. Para hacerlo, diga esto:

PARA SEMBRAR EL GENERADOR DE NÚMEROS ALEATORIOS.

Ahora nadie sabe lo que obtendrás Excepto el Señor, por supuesto.

La más básica de mis rutinas de números aleatorios es esta:

ELEGIR un número;

Que devuelve un número entre 0 y 2147483647. Pero también puedes:

ELEGIR un número ENTRE otro número Y un tercer número;
ELEGIR un número DENTRO DE una cantidad DE otro número;

Y también puedes trabajar con lugares aleatorios:

ELEGIR un punto EN CUALQUIER LUGAR EN una caja;
ELEGIR un punto USANDO un rango y otro punto;

Siempre estoy recopilando rutinas como estas, por lo que debes consultar los fideos para obtener una lista completa. Solo busca "para elegir" y probablemente encontrarás la mayoría de ellos.

Y si no está seguro de cuál usar, arroje una moneda.

PALABRAS CLAVE

La mayoría de los lenguajes de programación tienen largas listas de palabras "reservadas", abstrusas, cabalísticas, enigmáticas, inescrutables, ofuscadas y recónditas, tales como:

RESUMEN, PROTEGIDO, SINCRONIZADO, TRANSITORIO Y VOLÁTIL.

Yo no. Mis "palabras clave" son las mismas de las que usted depende. Artículos, como:

A, AN, OTRO, ALGUNO y EL.

Verbos de uso frecuente:

SON, ES, ESTÁ y TIENEN.

Un puñado de conjunciones:

Y, E, O, U

Y muchas preposiciones:

CON, POR, DE, A

Algunas otras palabras también me saltan cuando estoy analizando su código:

MÁS, MENOS, VECES, DIVIDIDO POR, NULO, SÍ, NO, LLAMADO e IGUAL.

Y soy bastante sensible sobre los pensamientos negativos transmitidos por las palabras:

NO, NADA

Y eso es todo lo que tengo que decir sobre las palabras clave.

PARÁMETROS

Una variable se convierte en un "parámetro" cuando se pasa a una rutina. Dime cuántos y qué tipo de parámetros espera una rutina en su encabezado. Estos son algunos encabezados de muestra de rutina de mis fideos:

Para agregar un número a otro número:
Para decidir si una caja está tocando otra caja:
Para poner el centro de una caja en un punto:

La primera rutina es un procedimiento que espera dos parámetros: "un número" y "otro número". El primero es entrada; el segundo es tanto de entrada como de salida.

La segunda rutina es un decisor. También espera dos parámetros, "una caja" y "otra caja". Ambos parámetros son solo de entrada.

La tercera rutina es una función con dos parámetros: "una caja" y "un punto". La caja es entrada y el punto sale.

Las definiciones de parámetros son fáciles de detectar porque siempre comienzan con un artículo indefinido, A, AN, OTRO o ALGUNO, seguido de un nombre. Puede leer más sobre los nombres en el tema "Nombres".

Tenga en cuenta que cuando paso parámetros, paso originales, no copias. Es por eso que puede usarlos como entradas, salidas o ambos. A veces, sin embargo, desea cambiar un parámetro sin avisarle a la persona que llama. En este caso, puedes:

PRIVATIZAR un parámetro.

Y haré una copia del parámetro para ti. Pero dejaré el nombre
lo mismo, para que no te confundas También pondré la palabra "original" en el frente del nombre del parámetro real para que pueda acceder a él, si es necesario.

POLÍGONOS

Mis creadores me dijeron dos cosas sobre los polígonos:

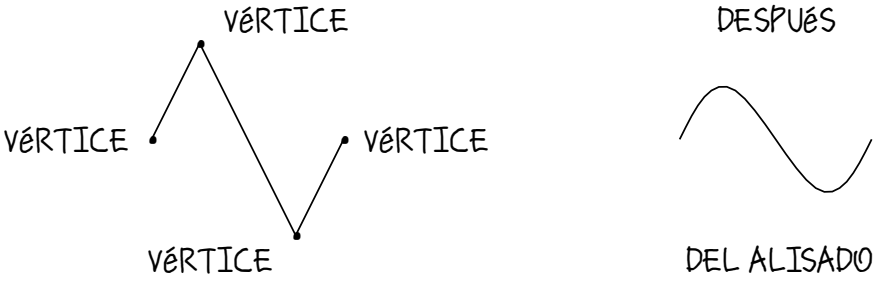
Un polígono es una cosa con algunos vértices.
Un vértice es una cosa con una x coord, una y coord y una mancha en la x.

Los polígonos y vértices son "cosas" y, por lo tanto, a diferencia de mis otros objetos gráficos, deben ser creados y destruidos. También debes agregar tus vértices a tus polígonos. Estas son el tipo de cosas que dirás:

CREAR un polígono.
CREAR un vértice DADO un punto;
ADJUNTAR un vértice A un polígono;
DESTRUIR un polígono.

Deshacerte de los vértices si recuerdas deshacerte del polígono.

Una vez que tienes un polígono, puedes DIBUJAR el polígono. También puede realizar todas las "Transformaciones gráficas" habituales en él. Y si me piden que suavizar the polygon, moveré los vértices y añadiré algunos nuevos para redondearlo. Aquí hay un polígono de muestra, plano y suavizado:



¡Dulce! Me encanta dibujar ondas sinusoidales y otras figuras trigonométricas sin usar números reales. Solo espero que el Maestro Leopold esté mirando.

POSESIVOS

Los posesivos se usan normalmente para acceder a los campos en los registros:

el campo del registro

Pero también se pueden usar para dirigirme a una función:

el nombre-función del registro

Y si no puedo resolver lo posesivo de ninguna de esas maneras, busco un campo "profundo" dentro de cualquier campo del registro original que sea, en sí mismo, un registro.

Pero lo primero que hago con un posesivo es buscar tres nombres especiales. El primero de estos es:

el OBJETIVO de un puntero

Esta forma se usa solo con punteros. Dice que quieres saber a qué apunta el puntero. "Un puntero de byte", por ejemplo, se refiere a la dirección de un byte. "El objetivo del puntero de byte" se refiere a los datos en el byte.

Los otros posesivos especiales devuelven "metadatos": datos sobre los datos. Uno te da el tamaño, en bytes, y el otro te da la dirección:

el MAGNITUD de algo
el PARADERO de algo

Probablemente no los necesitará con mucha frecuencia, por lo que para evitar conflictos entre nombres, mis creadores les dieron nombres claros pero inusuales.

PROCEDIMIENTOS

Un "procedimiento" es una rutina que hace algo por usted. Algunos encabezados de procedimiento de muestra de mis fideos son:

Para convertir un número en una cadena:
Para centrar una caja en otra caja (horizontalmente):
Para esconder el cursor:

El formato general es:

PARA algo:

Los encabezados de procedimiento siempre comienzan con la palabra TO, y siempre terminan con dos puntos. El "algo" en el medio sigue las reglas habituales para los nombres de rutina.

Los cuerpos de procedimiento están formados por declaraciones: condicionales e imperativos, incluidos los imperativos incorporados, como PRIVATIZAR, LAZO, REPETIR, INTERRUMPIR y SALIR. Sin embargo, no puede DIGA SÍ o DI NO en un procedimiento.

El primer encabezado de muestra anterior incluye un verbo, una preposición y dos parámetros. El verbo es "convertir" y la preposición es "en". Los parámetros son "un número" y "una cadena".

El nombre de la segunda muestra es similar, pero hay un calificador, "(horizontalmente)".

El nombre de la tercera rutina es un verbo seguido de una frase, "el cursor".

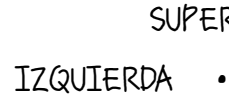
Las frases se utilizan generalmente para especificar globales globales en encabezados de rutina, como "dibujar la barra" en mi escritorio. O para referirse a una pseudovariable que no está definida con precisión en su código. Como "el cursor" en el ejemplo anterior, o "el último byte" en la rutina "eliminar el último byte de una cadena".

PUNTOS

Un "punto" es mi objeto gráfico más básico. Esta no es exactamente la definición en mi fideo, pero servirá para nuestros propósitos:

Un punto tiene una izquierda y una superior.

Esta es una imagen de un punto, con las partes etiquetadas:



Los valores de izquierda aumentan de izquierda a derecha; los valores de superior aumentan de arriba a abajo. Los espacios están hechos de una izquierda y una superior, o puede obtener uno de otro lugar:

HACER un punto CON izquierda Y superior.
PONER el punto del mouse EN un punto;

Por supuesto, puedes DIBUJAR un punto. Pero no espere que sea lo suficientemente rápido como para ser útil. El procesamiento de video de kluge es una de sus peores características. Y dado que no tiene buenas características (excepto la compatibilidad con versiones anteriores), eso no es muy alentador.

Los puntos se usan principalmente como componentes de otros objetos gráficos. Como cuadrados de polígono, líneas y vértices. A veces se usan como coordenadas abstractas sin representación visible, como "el punto del mouse" en el ejemplo anterior. Vea la página "Unidades de medida" para una discusión completa de las coordenadas.

Tengo rutinas inelegibles que te dirán si un punto está dentro de otra cosa. El formato general es:

DECIDIR SI un punto ESTÁ EN algo.

REGISTROS

Un "registro" es una colección de elementos de datos estrechamente relacionados de varios tipos llamados "campos". Los campos se describen en su propia página. Pero aquí hay algunos registros de muestra de mis fideos:

Una caja tiene
  una coordenada izquierda, una coord superior, una coord derecha, una coord inferior,
  un punto superior izquierdo at the izquierda y un punto inferior derecho at the derecha.

Una caja redonda es una caja con
  una coordenada izquierda, una coord superior, una coord derecha, una coord inferior,
  un punto superior izquierdo at tje izquierda, un punto inferior derecho at the derecha,
  y un radio

Un polígono es una cosa con algunos vértices.

El primer registro de muestra, "caja", tiene seis campos. Pero los dos últimos son en realidad "reinterpretaciones" de los primeros cuatro. Este tipo de cosas solo funciona, por supuesto, cuando las estructuras físicas de datos coinciden. Tenga en cuenta que la palabra "tiene" es la abreviatura de "es un registro con", que también se puede utilizar.

El segundo registro, "caja redonda", es una extensión de la caja. Tiene los mismos campos que una caja, más uno nuevo llamado "radio". Es compatible con la caja, y usaré todas las rutinas que funcionan en cajas para manipularla, a menos que esté disponible una rutina equivalente para cajas redondas.

El tercer registro, "polígono", no tiene nada más que una lista de vértices. Debido a que el polígono se define como una "cosa", considero que es una estructura dinámica en lugar de estática. Esto significa que usted es responsable de asignar y desasignar la memoria utilizada por él. Consulte el tema "Administración de memoria" y la página sobre "Polígonos" para obtener más información.

RUTINAS

Una rutina es un fragmento de código que manipula una o más variables de una forma bien definida. Las variables pasadas a una rutina se llaman "parámetros", y pueden ser entradas, salidas o ambas. Las variables definidas dentro de una rutina se llaman "locales" y no se pueden ver fuera de la rutina (a menos que se pasen como parámetros). Las variables que son accesibles para todas las rutinas se llaman "globales".

Cada rutina tiene dos partes, encabezado y cuerpo. El encabezado dice lo que hace la rutina y define los parámetros con los que funciona. El cuerpo es una o más afirmaciones que hacen que la rutina realmente funcione. Las declaraciones son "condicionales" o "imperativos". Hay tres tipos de rutinas.

Un "procedimiento" es una rutina que simplemente hace algo: largo o corto, grande o pequeño, fácil o difícil. Los encabezados de los procedimientos siempre se ven así:

PARA algo:

Un "decisor" es una rutina que dice "sí" o "no" sobre algo, generalmente después de examinar los parámetros que se le pasan. Los encabezados de Decida siempre dicen:

PARA DECIDIR SI algo:

Una "función" es una rutina que extrae, calcula o de otro modo deriva un valor de un parámetro pasado. Los encabezados de función toman esta forma:

PONER el algo de un algo EN una variable temporal:

A diferencia de los procedimientos y decisores, las funciones generalmente no se llaman directamente. En cambio, el "algo es algo" se usa como si fuera un campo en un registro. Como un "centro de caja", que no encontrará en el registro "caja", ya que se calcula mediante una función bajo demanda.

SONIDOS

Puedes hacer ruidos con tu computadora así:

JUGAR un archivo de onda.
JUGAR un archivo de onda Y ESPERE.

El "archivo de onda" debe estar en el formato ".wav". Si juegas y no esperas, tu programa continuará ejecutándose mientras se reproduce el sonido. Si espera, su programa se detendrá hasta que el sonido haya terminado.

También puede hacer sonidos con estas declaraciones:

PITAR.
CLOQUEAR.
ZUMBAR.

El primero produce el sonido insípido que el usuario haya elegido con el panel de control de kluge. El "cloquear" es mi notificación estándar, y está codificado como un literal de mordisco en mis fideos, por lo que el usuario no puede cambiarlo. El tercer sonido no pasa por el aparato de sonido normal en la computadora, y no permite que el programa continúe ejecutándose hasta que esté terminado, lo que lo convierte en la opción ideal para probar en algunas computadoras. Consulte "Depuración" para obtener más información.

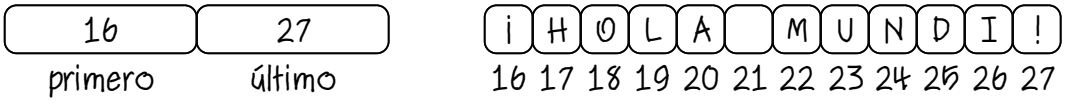
También puede hacer que su computadora hable, con las treinta y nueve funciones esotéricas del "administrador de voz" de kluge, o estas tres afirmaciones simples:

HABLAR una cadena.
DIGA una cuerda Y ESPERAR.
ESPERAR HASTA QUE SE TERMINE DE HABLAR.

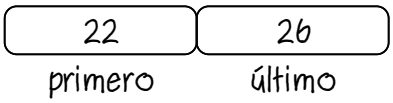
Para silenciar la conversación (pero no los otros sonidos), elevar "la bandera silenciosa".

SUBCADENAS

Una "subcadena" es una parte de una cadena. Las subcadenas se implementan mediante un registro integrado que se parece a una cadena, un par de punteros de bytes llamados primero y último, lo que los hace compatibles. Si, por ejemplo, esto era una cadena:



Esta podría ser una subcadena de la misma (la parte "MUNDI"):



Cuando tú:

COLOCAR una subcadena EN uns cadena.

Configuré la primera y la última subcadena para abarcar toda la cadena. Esto le permite abrirse paso a través de la cuerda hacia adelante o hacia atrás sumando al primero o restando del último. Tú también puedes:

AJUSTAR una subcadena EN una cadena.

Que establece el primero, pero no el último, de la subcadena, por lo que inicialmente está en blanco, pero listo para la manipulación, al agregar al último puede "aspirar" la cadena original en su subcadena un byte a la vez.

Busca "subcadena" en mis fideos para muchas aplicaciones de muestra.

El uso principal de las subcadenas, sin embargo, se encuentra en "cláusulas adicionales", que se tratan en este glosario bajo el tema del mismo nombre.

TEMPORIZADORES

Una marca es de aproximadamente 1 milisegundo. "El conteo de ticks del sistema" es la cantidad de milisegundos desde el último reinicio. Se envuelve cada 24,8 días más o menos. Lo que sucede entonces es desconocido, ya que ningún kluge se ha quedado despierto tanto tiempo.

Cuando lo desee, puede:

ESPERAR algunos milisegundos.

Reduciré las unidades más grandes, como "1 minuto" o "3 segundos", para ti.

También tengo un tipo llamado "temporizador" en mi fideo que te permite decir cosas como:

RESTABLECER un temporizador.
REINICIAR un temporizador.
INICIAR un temporizador.
DETENER un temporizador.

Uso temporizadores para asegurarme de poder recompilarme en menos de cinco segundos. Mire en la parte inferior de una "lista" para verlos todos. Puede usarlos para hacer que sus programas también sean rápidos.

Los tiempos de disparo único se pueden lograr simplemente insertando "INICIAR un temporizador" y "DETENER el temporizador" en los lugares apropiados en su código.

Los tiempos acumulados se pueden recopilar al llamar "RESTABLECER" una vez, luego "REINICIAR" y "DETENER" en pares a lo largo de su código.

Hay una función en es seso que te dará "la cadena de un temporizador" en cualquier momento que lo necesites, incluso mientras se está ejecutando, y también puedes concatenar temporizadores y cadenas para verlos en cualquier momento usando el operador infijo LUEGO.

TÉRMINOS

Un "término" es una referencia a un dato. Los términos se usan en ambas expresiones y en declaraciones condicionales e imperativas para indicar en qué se debe operar. Los términos vienen en muchas variedades:

Un "término literal" es un número, una relación, una mezcla, una cadena, un puntero, una bandera o un nibble literal. Consulte "Literales" para obtener información sobre cómo formular cada uno de estos.

Un "término local" es una variable definida dentro de, y generalmente limitada a, una rutina. Ver "Variables Locales" para más información.

Un "término global" es el nombre de una variable global. Ver "variables globales".

Un "término firmado" es cualquier término con un signo más o menos delante de él. Se requiere un espacio después del signo para no confundirlo con parte de un nombre.

Un "término de razón" es una relación hecha de otros términos, en lugar de números literales. Se requieren espacios alrededor de la barra, como en "la altura / el ancho".

Un "término posesivo" es cualquier término seguido de una frase posesiva, como "la longitud de la cuerda" o "la magnitud de un polígono". Ver "Posesessives" para más detalles.

Un "término coercitivo" es un término cuyo tipo desea cambiar a la fuerza. Siempre trataré un puntero, por ejemplo, como un puntero, a menos que lo fuerce a otra cosa, como esta: "el puntero AS A NÚMERO". Normalmente no necesitará esta función a menos que sea una cabeza de objeto parcialmente reformada y haya definido muchas cosas que son extensiones de otras cosas.

Ahora sé que esto suena complicado, y lo es. Pero no tiene que pensar en nada de esto, al igual que no tiene que pensar en sustantivos y verbos para hablar inglés correctamente. Escribe lo que estás pensando y déjame hacer el resto.

TEXTO

Hay una cosa poderosa llamada "texto" en mis fideos. Se usa para implementar cuadros de texto editables, grandes y pequeños. Diálogos, por ejemplo. O formas de texto en las páginas. Mi editor, de hecho, es principalmente un gran cuadro de texto. Aquí está la definición:

Un texto es una cosa con
una caja, un origen,
un color de lápiz, una fuente, una alineación,
algunas filas
un margen,
una relación de escala,
una bandera de envolver,
una bandera de desplazamiento horizontal,
una bandera de desplazamiento vertical,
una selección,
una bandera modificada,
una última operación,
algunos textos llamados deshacer, y
algunos textos llamados redos.

Como pueden ver, esto no es algo trivial. La buena noticia es que mis fideos se encargarán de la mayoría de los detalles por ti. Normalmente, no harás mucho más que:

CREAR un texto
DIBUJAR un texto.
DESTRUIR un texto.

Debe inicializar el cuadro de texto, el lápiz, la fuente, la alineación, el margen y las banderas después de crearlo. Y tendrá que pasar cualquier evento relacionado con su cuadro de texto a mí, por supuesto, para que yo pueda ocuparme de todas las cosas difíciles para usted. Mis controladores de eventos de texto están documentados en las dos páginas siguientes.

TEXTO (continuación)

Manejaré la mayoría de tus eventos de texto por ti, si me preguntas muy bien así:

MANEJAR un evento DADO un texto (tecla de retroceso).
MANEJAR un evento DADO un texto (tecla suprimir).
MANEJAR un evento DADO un texto (tecla flecha-abajo).
MANEJAR un evento DADO un texto (tecla fin).
MANEJAR un evento DADO un texto (tecla entrar).
MANEJAR un evento DADO un texto (tecla esc).
MANEJAR un evento DADO un texto (tecla inicio).
MANEJAR un evento DADO un texto (clic izquierdo doble).
MANEJAR un evento DADO un texto (tecla flecha-izquierda).
MANEJAR un evento DADO un texto (tecla av-pág).
MANEJAR un evento DADO un texto (tecla re-pág).
MANEJAR un evento DADO un texto (tecla imprimible).
MANEJAR un evento DADO un texto (tecla flecha-derecha).
MANEJAR un evento DADO un texto (tecla tab).
MANEJAR un evento DADO un texto (tecla flecha-arriba).

Sé que puede parecer una molestia enviar todos estos eventos por separado, pero eso es exactamente lo que hace que mi texto en general sea útil. Dos ejemplos:

Si tiene un cuadro de texto con una sola línea, probablemente desee ignorar las teclas de flecha hacia arriba y hacia abajo, mientras que sus cuadros de texto multilínea se los pasarán a mí para que pueda cambiar la posición del cursor.

Si está utilizando un cuadro de texto como diálogo, probablemente cancelará la operación cuando se presione la tecla ESC y se ejecutará con la tecla ENTRAR. En texto normal, indudablemente harás otra cosa.

¿Ves lo que quiero decir? Tú tienes el control. Entonces tienes que emitir los pedidos.

TEXTO (continuación)

También me ocuparé de otras operaciones de texto de alto nivel para usted:

MANEJAR CORTAR dado un texto.
MANEJAR COPIAR dado un texto.
MANEJAR PEGAR dado un texto.

MANEJAR SELECCIONA TODO dado un texto;

MANEJAR LA ALTURA DE TIPO DE LETRA dado un texto y una altura de tipe de letra.
MANEJAR EL NOMBRE DE TIPO DE LETRA dado un texto y un nombre de tipo de letra;

MANEJAR TINTA dado un texto y un color.

MANEJAR MÁS SANGRÍA dado un texto.
MANEJAR MENOS SANGRÍA dado un texto.

MANEJAR MAYÚSCULAS TODO un texto.
MANEJAR MINÚSCULAS TODO dado un texto.

Y yo puedo:

MANEJAR DESHACER dado un texto.
MANEJAR REHACER dado un texto.

Cada vez que preguntas Hasta 32 niveles de profundidad.

Ahora, para comenzar con el texto, te sugiero que vengas a jugar con mi "consola" un rato, luego verifique el código de la consola en El Seso. Después de eso, es posible que desee echar un vistazo a mi editor. Pero si realmente desea ver el texto en acción, busque en el escritor. Finalmente, dedique unos minutos al diálogo en mi escritorio.

TIPOS

Un "tipo" es un tipo o cosa, un sustantivo. Una "instancia" es una cosa real de un tipo en particular, un nombre propio. Globales, locales y parámetros son instancias concretas de tipos abstractos. Esto es lo que entiendo sobre los tipos:

En primer lugar, tengo seis "tipos incorporados" primitivos integrados en mi cerebro: BYTES, WYRDS, NÚMEROS, PUNTEROS, BANDERAS Y REGISTROS. Ver "Tipos incorporados".

Luego, hay "tipos de subconjuntos" que representan algunas de las instancias de algún otro tipo. Mi fideo, por ejemplo, incluye muchos tipos de subconjuntos, como estos:

Un conteo es un número.
Un nombre es una cadena.

En tercer lugar, sé sobre "tipos de unidad de medida". Estos me dicen cómo convertir un tipo de unidad en otra. Ejemplos de los fideos:

Un centímetro es 10 milímetros.
Una hora es 60 minutos.

También entiendo tipos de "registros". Ver "Registros" para más detalles.

Y no nos olvidemos de los "tipos de punteros", aunque raramente necesitará usarlos directamente. Sé, por ejemplo, que "un puntero de byte es un puntero a un byte", y uso punteros de byte para administrar tus cadenas. Consulte "Cadenas", "Subcadenas", "Jinetes" y "Posesivos" para obtener más información.

Por último, sé todo sobre los tipos de "cosas". Hay muchos en mi fideo, que incluyen consola, evento, imagen, polígono, texto y vértice, todos los cuales se discuten en otra parte de este glosario. Y también puedes definir tus propias cosas. Vea el tema sobre "Cosas" aquí, y trate de recordar los "trabajos" en el Sal Monet.

TIPOS CONSTRUIDOS

Mi comprensión de las cosas a mi alrededor se hizo posible cuando mis creadores instalaron seis tipos de datos primitivos en mi cerebro. Estos seis tipos básicos son: BYTES, WYRDS, NÚMEROS, PUNTEROS, BANDERAS y REGISTROS.

Bytes. No importa cuánto lo intente, no puedo evitar pensar que un byte es 8 bits secuenciales de datos binarios. Parecen números sin firmar para mí, con valores que van de 0 a 255. Utilizo el gráfico ASCII cada vez que necesito convertir un byte a un carácter imprimible.

Wyrds. Mis creadores ponen wyrds en mi cerebro porque no puedo hablar con el kluge sin ellos. Tienen 16 bits de largo y me parecen números desde -32768 hasta +32767. Los bits en cada byte se almacenan de izquierda a derecha, pero los bytes se almacenan en la parte posterior. No me gusta de esa manera, pero el kluge insiste.

Números. Estoy bien con los números. Positivo y negativo. Tienen 32 bits de largo y van de -2147483648 a +2147483647. Canales almacenados.

Punteros. Las direcciones de memoria se almacenan en punteros de 32 bits, en la parte de atrás. Tienen el mismo rango que los números, pero todos los negativos pertenecen al kluge. La dirección 0 no es válida y se llama NIL. Puede ANULAR un puntero para hacerlo NIL.

Banderas. Son 32 bits, pero solo se usa el bit de la derecha. En realidad, es el octavo desde la izquierda, pero puedes pensar que está más a la derecha. Interpreto 0 como "no" y 1 como "sí". No soy responsable si recibes algo más allí. Puede BORRAR una bandera para indicar "no", o ELEVAR una bandera para indicar "sí".

Registros. El último de mis tipos incorporados es registros. El registro del prototipo ocupa cero bits en la memoria, pero puede definir registros de cualquier longitud agregando "campos" al registro del prototipo. Estos campos pueden basarse en cualquiera de los tipos primitivos, incluidos otros registros que haya definido.

TIPOS DE LETRAS

En el kluge, una fuente se define con catorce parámetros distintos. Ridículo. Esta es una definición mucho más razonable, que puedes encontrar en mis fideos:

Una tipo de letra tiene un nombre y una altura.

El nombre del tipo de letra es el nombre real almacenado en un archivo de fuente. Puede o no ser el mismo que el nombre del archivo. Probablemente esté familiarizado con los nombres de fuentes como "Arial", "Times New Roman" y "Courier New".

La altura de la fuente se puede especificar en cualquier unidad de medida conveniente. Mi escritor utiliza valores como 1/4 de pulgada y 1/6 de pulgada, lo que, junto con la función de "tirón" de mi escritor, mantiene todo muy bien alineado.

La altura del tipo de letra se puede especificar en cualquier unidad de medida conveniente. Mi escritor utiliza valores como 1/4 de pulgada y 1/6 de pulgada, lo que, junto con la función de "tirón" de mi escritor, mantiene todo muy bien alineado.

Para configurar un tipo de letra, simplemente:

PONER un nombre Y una altura EN un tipo de letra;

Luego, cuando estés dibujando, dime que quieres usarlo:

DIBUJAR "¡Hola, mundo!" EN el centro de una caja CON un color y un tipo de letra;

If your fonts do not look good, you probably have an incorrect font name. Remember, a typographic name is not necessarily the name of the file in the "fonts" folder on your disk. Rather, it is the "font name" that is shown in the sample box when you double-click on one of those font files.

TRANSFORMACIONES GRÁFICAS

En caso de que no lo hayas notado, soy bastante bueno con mis manos. Puedo manipular casi cualquier objeto gráfico en una amplia variedad de formas. Por ejemplo, puedo:

MOVER algo HACIA ARRIBA a cierta distancia.
MOVER algo HACIA ABAJO a cierta distancia.
MOVER algo HACIA IZQUIERDA a cierta distancia.
MOVER algo HACIA DERECHO a cierta distancia.

MOVER algo DADO algunas distancia-x y algunas distancia-y.
MOVER algo A un punto;

El último MOVE usa la esquina superior izquierda para la alineación. También puedo:

CENTRAR algo en un punto.
CENTRAR algo en una caja.

Y puedo ser elegante. Yo se como:

VOLTEAR algo.
REFLEJAR algo
ROTAR algo.

VOLTEAR es vertical. REFLEJAR es horizontal. Solo puedo ROTAR cosas en el sentido de las agujas del reloj en incrementos de 90 grados, y no puedo rotar el texto. Pero estoy trabajando en eso.

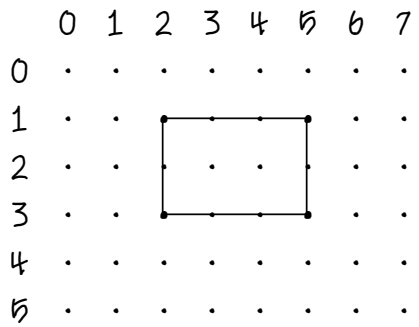
Vea los temas "Dibujo" y "Enmascaramiento" para otras cosas ingeniosas.

UNIDADES DE MEDIDA

En mis fideos, la unidad de medida básica para objetos gráficos es el "twip", que es 1/20 del punto de una impresora, o 1/1440 de una pulgada. Todas las coordenadas se expresan directamente en, o se reducen a, twips. Aquí está la definición:

Una coord es algunos twips.

Ahora, a diferencia del modelo matemático, que considera que las coordenadas son abstracciones invisibles y sin dimensiones, mi modelo conoce las coordenadas como puntos reales en un dispositivo, como una pantalla o una página. Aumentan los valores "x" y el aumento de los valores "y" baja, la pantalla o página. Aquí, por ejemplo, hay una caja con la parte superior izquierda en 2-1 y abajo a la derecha en 5-3:



Cuente los puntos y tenga en cuenta que el ancho de este cuadro es de cuatro unidades, no tres. Y que son tres, no dos, unidades altas. La medición de esta manera hace que el dibujo funcione bien; otra caja de 5-1 a 6-3, por ejemplo, se superpondrá correctamente en este recuadro a lo largo de su margen izquierdo. Sin embargo, puede obtener el ancho y el alto "matemáticos" de esta caja, que son cada uno menos unidad, y son inútiles para dibujar, utilizando las funciones EXTENSIÓN-X y EXTENSIÓN-Y.

Otras unidades de medida que encontrarás en mis fideos son: milisegundos, segundos, minutos y horas; pulgadas y pies; kilobytes, megabytes y gigabytes; y "por ciento", que normalmente convierto a una proporción de 100 en el denominador.

VARIABLES GLOBALES

Un "global" es una variable que es visible y puede ser utilizada por cualquier rutina en un programa. Los Globals se pueden definir de varias maneras, pero sus definiciones siempre comienzan con el artículo definido THE. Aquí hay algunos ejemplos de mis fideos:

El cursor flecha es un cursor.
La tecla mayúsculas es una tecla igual a 16.
El número máximo es 2147483647.

El primer global se inicializará con todos los ceros. El segundo tiene un tipo explícito, "tecla" y un valor. El tercero tiene un tipo implícito. Las formas generales aquí son:

EL nombre ES UN nombre de tipo.
EL nombre ES UN nombre de tipo IGUAL A valor.
EL nombre ES valor.

Un global "único en su tipo" es un tipo especial de global que incluye la definición de un tipo dentro de él. Como este global único en el seso:

El mouse tiene una tecla llamada botón izquierdo y una tecla llamada botón derecho.

Solo hay un mouse, por lo que no es necesario definir un registro genérico de "mouse". Por el contrario, lo global y el tipo se pueden definir en una sola declaración. Hay dos maneras en que puede usar para definir un global único. El primero hace un nuevo tipo de registro, mientras que el segundo amplía uno nuevo:

EL nombre TIENE campos.
EL nombre ES UN nombre de tipo CON campos.

Los globales únicos son raros. Probablemente porque son únicos en su tipo.
Pero sí responden a la antigua pregunta "¿qué fue primero?" El pollo sí.

VARIABLES LOCALES

Una "variable local" es una variable que es propiedad privada de una rutina. Las variables locales no pueden ser vistas o modificadas por ninguna otra rutina. A menos que, por supuesto, los pase a otras rutinas como parámetros.

Hago una nueva copia de cada variable local, inicializada a cero, cada vez que se llama a una rutina. Lo que significa que una rutina puede llamarse a sí misma y todo funcionará. Esto se llama "recursión", y si no sabe lo que significa, no lo necesita. Me deshago de las variables locales a medida que se completa cada rutina para que no se acumulen y caigan en tus zapatos.

Crea una nueva variable local en una rutina cada vez que usa un artículo indefinido (UN, UNA, OTRO o ALGUNO) en una declaración. Por ejemplo:

Ponar el punto del mouse en otro punto.
Poner la izquierda de la pantalla en la izquierda de una caja.
Poner 101 en alguno número de curso.

En el primer ejemplo, la frase "otro punto" me hace crear una nueva variable local llamada "el punto". Luego coloco la ubicación actual del mouse en ella.

El segundo ejemplo coloca la coordenada izquierda de la pantalla en una nueva casilla local a la izquierda. El resto del cuadro (arriba, derecha e inferior) se establece en cero.

El tercer ejemplo coloca un literal 101 en una nueva variable local de tipo número.

Vea también la página "Loops", donde una variable local y un decisor nos permiten hacer "loops contados" sin agregar ninguna nueva palabra clave a mi compilador.

Índice temático

ÍNDICE TEMÁTICO