DATA BUS

Número 4 Segunda Epoca Marzo 1.993

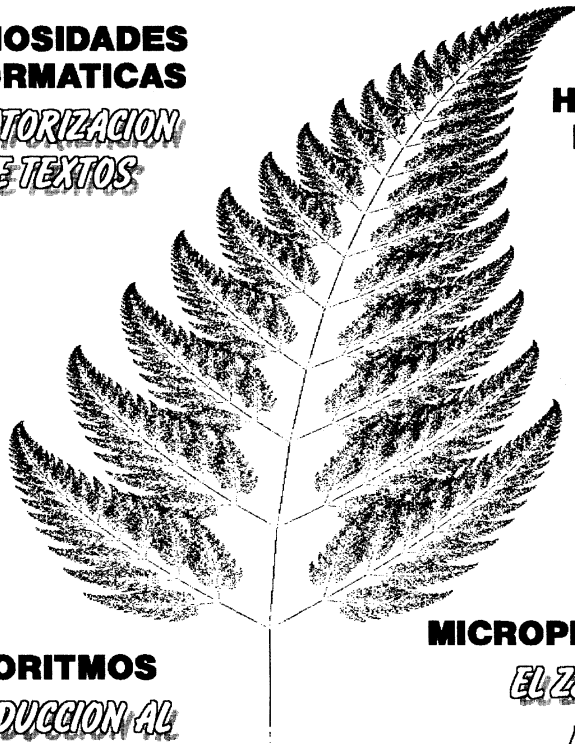
EDITA
ASOCIACION JUVENIL PARA LA INFORMATICA VIGUESA
AJV

CURIOSIDADES INFORMATICAS

ALEATORIZACION DE TEXTOS

HISTORIA DE LA INFORMATICA

PARTE VII



Y LAS ULTIMAS NOTICIAS EN EL MUNDO DE LA INFORMATICA

MICROPROCESADORES

EL Z80 DE ZILOG

PARTE IV

ALGORITMOS

INTRODUCCION AL PROCESADO DIGITAL DE IMAGEN

DATA BUS

Número 4

Segunda época

Marzo de 1.993

DIRECTOR

Jesús Cea Avión

REDACTORES

Jesús Cea Avión

Nacho Agulló Sousa

Jose Manuel Suárez Pousa

COLABORADORES

David Martínez Oliveira

Francisco Bellas Aláez

DISEÑO Y MAQUETACION

Jose Manuel Suárez Pousa

Jesús Cea Avión

ASOCIACION JUVENIL PARA LA INFORMATICA VIGUESA

AJIV

C/Caldas de Reis, 12-6 Izq.
36209 VIGO
Telf: (986) 23 18 35

Depósito legal
VG-87-90

¡¡¡ ATENCION !!!

Si deseas colaborar con nosotros en *DATA BUS* aportando ideas, críticas, artículos, etc., no tienes más que hacernos llegar tu trabajo. Para ello puedes enviárnoslo a la dirección de la asociación, o bien ponerte en contacto con la persona que te suministró este ejemplar.

¡¡¡ CONTAMOS CON VOSOTROS !!!

INDICE

EDITORIAL

Siguiendo una tradición no escrita, ***DATA BUS*** vuelve a salir con retraso. En el número anterior se dijo que aparecería a finales de Febrero o principios de Marzo, pero la verdad es que ha terminado de maquetarse hoy, Viernes Santo (una demora de un mes). En nuestro descargo constan los retrasos en la entrega de algunos artículos y las numerosas tareas, tanto escolares como referentes a la convocatoria de subvenciones, que hemos tenido que atender.

Este es un número muy especial por varias razones. La primera es que, por primera vez, podemos contar con un artículo de calidad escrito por personas ajenas a A.J.M.. Damos la bienvenida a estos nuevos colaboradores y esperamos que pronto nos hagan llegar un nuevo artículo. Parece que el Editorial del último número tuvo algunos efectos, aunque necesitamos mucha más gente interesada en que ***DATA BUS*** mejore. ¡¡Animo!!.

La otra razón son los listados incluidos, tanto en Basic como en C. El que se incluyan o no en futuros números depende completamente de la acogida que les dispenséis. Hacednos llegar vuestras opiniones.

Quizás notéis unas molestas lineas verticales en la revista. Son debidas al mal funcionamiento de un injector de la impresora. Siguiendo las leyes de Murphy, los cartuchos de las impresoras de chorro de tinta siempre se acaban en mitad de una página de suma importancia, pero cuando son defectuosos parecen durar eternamente. Es de esperar que para el próximo número ya habremos acabado con él (crucemos los dedos...). Os pedimos disculpas por este problema ajeno a nuestra voluntad.

Por último, aunque no por ello menos importante, ¿Cuándo verá la luz el siguiente número de ***DATA BUS***? Sabemos que los plazos se han hecho para incumplirse, pero no podemos dejar de comprometernos en ese sentido. Si todo va bien (es decir, mejor que hasta ahora), el siguiente número de ***DATA BUS*** saldrá a la calle a finales de Julio. ¿Alguien se atreve a apostar algo?

La Redacción

Una somera introducción al

Procesado de Imagen

Francsco Bellas Aláez

David Martínez Oliveira

Introducción

Es algo que suena a tecnología punta relacionado con disciplinas tan poco populares como la meteorología, astronáutica, diagnosis de enfermedades o la identificación de ballenas. En realidad, hasta no hace mucho así era, puesto que los equipos personales no disponían ni de la potencia de cálculo ni de la capacidad gráfica necesaria.

De un tiempo a esta parte se hizo muy fácil conseguir un equipo muy potente por un módico precio, con una capacidad gráfica de gran calidad. Debido a esta popularización y a lo apasionante del tema, pretendemos describir en este artículo, de una forma somera e informal, los primeros pasos que podríamos seguir para introducirnos en el apasionante mundo del procesado digital de señales.

Las personas relacionadas con el tema, o con la informática en general, podrían encontrar algunas partes del artículo demasiado superfluas, casi inocentes, pero como indica el título se trata de una breve introducción.

¿Qué es la imagen digital?

En realidad la pregunta lógica sería ¿qué es la imagen analógica?, y al referirnos a analógica me refiero a continua. Nadie percibe las imágenes como un conjunto de puntos de diversos colores sino que recibe una

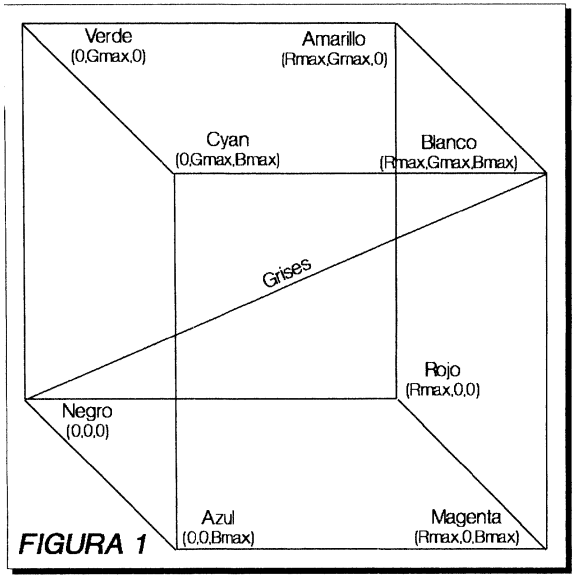


FIGURA 1

imagen continua de su entorno. Sin embargo el ojo humano no es más que un digitalizador. En la retina disponemos de una serie de células fotosensibles (unas a la intensidad y otras al color) que transmiten la información al cerebro a través del nervio óptico. Es decir, de la imagen continua que se proyecta en la retina solo estamos "viendo" un conjunto finito de puntos, lo que sería una imagen digital. Así el concepto de imagen analógica en el sentido de continuidad, en principio no existe en el hombre.

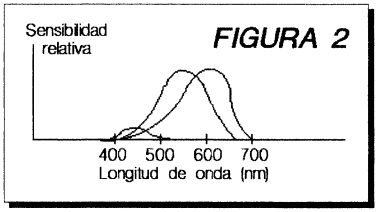


FIGURA 2

Las imágenes digitales con las que trabajaremos provienen de un digitalizador o escaner y no son más que una matriz numérica en la que cada número representa la intensidad del punto correspondiente de la imagen digitalizada. La matriz es bidimensional para de esa forma acceder a los puntos a través de sus coordenadas, en el sentido general de éstas. Estamos superponiendo a la imagen una rejilla y cada cuadrícula de esa rejilla es una entrada de la matriz a la que se le asigna un código numérico que representa la intensidad (o intensidad media) presente en la cuadrícula.

Así pues cuantas más entradas tenga nuestra matriz más resolución tendrá, los puntos serán más pequeños y, por lo tanto, será más probable que en esa cuadrícula solo tengamos una intensidad y no un conjunto que tengamos que promediar. Cuanto mayor sea el número (de valores) que podamos almacenar en una entrada de la matriz, mayor será el número de colores que estemos representando en nuestra imagen.

```
Paleta de Grises
0 1 2 3 4 5 6 7  Intensidades
2 7 4 6 0 3 1 5  de Gris

Tabla de Indexado
0 1 2 3 4 5 6 7  Indices de acceso
4 6 0 5 2 7 3 1  ordenado a la paleta
```

FIGURA 3

Antes de continuar, simplemente señalar una sutil diferencia entre el sistema digital compuesto por el ojo humano y un digitalizador. Las muestras que se toman con el digitalizador se toman en principio en puntos fijos de la imagen, con lo cual, necesitamos un número mínimo de muestras para poder reproducir la señal correctamente, es decir, tenemos una resolución umbral por encima de la cual la imagen se representa tal cual. Estamos en el caso de que en cada cuadrícula nos encontramos con una sola intensidad (este límite es el impuesto por el teorema de Nyquist de la teoría de muestreo), por debajo de ese límite la imagen pierde calidad debido a lo que se conoce como *"aliasing"*, que no vamos a describir aquí pues por si mismo, este tema supondría un todo un artículo.

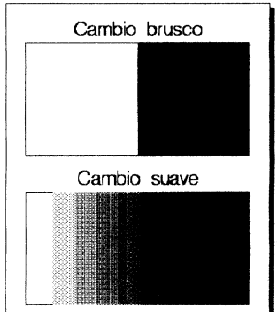


FIGURA 4

Sin embargo las células fotosensibles en el ojo humano no están distribuidas uniformemente sino que siguen una distribución aleatoria de disco de Poisson, lo que va a producir un muestreo estocástico y cuya finalidad última es reducir ese aliasing del que hablábamos antes.

El preprocesado

Ahora que tenemos claro lo que es una imagen digital, supongamos que nos hacemos con una. Probablemente no sea más que un fichero en alguno de los formatos gráficos habituales: *PCX, GIF, TIFF,* etc... Por lo tanto para poder pasar la información en el fichero a una matriz con la que trabajar comodamente tendremos que conocer la forma en que ésta se codifica dentro de estos ficheros. En el fichero encontramos además información de la resolución de la imagen (y por lo tanto de las dimensiones de nuestra matriz), el número de colores y su definición, etc...

En este artículo vamos a trabajar con imágenes en tonos de grises, la razón es muy sencilla. En primer lugar es muchísimo más fácil trabajar con este tipo de imágenes y en segundo lugar si se dispone de un número muy reducido de colores (pongamos 16) los resultados son mucho mejores con 16 tonos de grises que con 3 rojos 6 azules y 7 verdes, por ejemplo. Como hemos indicado al principio se trata de un artículo de introducción y no queremos complicar las cosas. De todas formas en la bibliografía indicada se indica como trabajar con color. En general, simplemente suele consistir en aplicar el algoritmo descrito para tonos de gris en cada una de las componentes por separado R (rojo), G (verde) y B (azul).

Por lo tanto el primer paso que debemos seguir es el de pasar la imagen a grises, y para ello un poco de teoría del color. Todo color puede representarse por tres componentes: una roja que denotaremos por la letra R de Red, otra verde (G de Green) y otra azul (B de Stokes, no es coña, de Blue), por lo tanto podremos tener una representación tridimensional del color con valores normalizados, es decir, cada componente se divide por el valor máximo que puede alcanzar ésta, lo que convierte el valor máximo normalizado en 1 (ver Figura 1).

El origen (R=0, G=0, B=0) se correspondería con el color negro, y de forma análoga el vector (1,1,1) representa el blanco. La recta que une ambos se corresponde con todos los tonos de grises representables. De la misma forma, moviéndonos en cada uno de los ejes obtendremos las tonalidades de cada componente (léase rojo, verde o azul). Por lo tanto para representar un gris es preciso que las tres componentes sean iguales R=G=B.

```
for (i=0;i<ancho_imagen;i++) {
    x = i % 4;
    for (j=0;j<alto_imagen;j++) {
        in = imagen[i][j];
        b0 =in >> 4;
        if ((b0+1)>0xf) b1=0xf; else b1 = b0+1;
        vi = in & 0xf;
        y = j % 4;
        if (vi>matriz_dither[x][y]) pinta (i,j,b1);
            else pinta (i,j,b0);
    }
}
```

LISTADO 1

Como ya indicamos, en la retina existen una serie de células fotosensibles, que se dividen en dos grupos: conos y bastones. Los conos a su vez se dividen en tres grupos, cada uno de los responde a una de las componentes anteriormente mencionadas (R, G, B). Como podemos apreciar en la figura 2 la respuesta de cada uno de ellos a la componente correspondiente no es la misma, para cada grupo de conos, por lo tanto a la hora de determinar el tono de gris que correspondería a un determinado color debemos de tenerlo (y meterlo en la

cárcel) en cuenta. La ecuación que nos da la intensidad en función de las componentes R, G, B es: Y = 0.299R + 0.587G + 0.114B, que no es ni más ni menos que la señal de luminancia que se envía en radiodifusión televisiva *(broadcasting TV)*.

Ahora estamos en condiciones de pasar la imagen a tonos de grises. Mediante la ecuación anterior transformamos la paleta RGB en una secuencia de tonos de grises. Para ello leemos las tres componentes de cada entrada de la paleta, le aplicamos la ecuación y sustituimos cada una de las componentes por Y.

El siguiente paso consiste en ordenar los tonos de gris, que ahora podemos hacer puesto que hemos pasado de un espacio de tres dimensiones a uno de una sola dimensión. A continuación tenemos que hacer corresponder las entradas de la matriz donde tenemos almacenada la imagen con la nueva definición de colores. Mediante una matriz auxiliar de indexado almacenamos el índice correspondiente a la entrada de la matriz de la paleta ordenada, es decir, si el gris que debería ocupar la posición tercera por su orden de intensidad, se encuentra en la entrada número diez de la paleta, en la matriz auxiliar almacenaríamos un diez en la entrada tres (ver figura 3). De esta forma podemos acceder a los puntos de la imagen por sus valores de intensidad ordenados, utilizando la matriz auxiliar.

Puesto que estamos preprocesando la imagen para que su manipulación posterior sea más sencilla, sería interesante almacenar la nueva imagen con sus valores de paleta ordenados. Es decir, ordenamos la paleta y cambiamos las entradas de la matriz en la que se encuentra la imagen según la matriz auxiliar anteriormente definida. Llegado a este punto estamos en condiciones de procesar comodamente nuestra imagen original.

Visualización de las imágenes en la pantalla.

Supongamos que tenemos un ordenador que solo puede representar 16 colores en pantalla, y nuestra imagen almacenada en memoria tiene 256 colores, ¿cómo podemos tener una visualización lo más fidedigna posible de la imagen almacenada en memoria? Este problema se soluciona con las técnicas de *dithering* que procedemos a explicar a continuación.

Inicialmente estas técnicas eran utilizadas para convertir imágenes en tonos de grises a imágenes monocromas, para poder visualizarlas en dispositivos de este tipo (pantallas de plasma, impresoras,...). La forma más sencilla de obtener una imagen monocromática es fijar un nivel de gris que constituye el umbral que nos indicará si el punto se pinta. Por ejemplo, si pintamos puntos blancos sobre fondo negro, una vez fijado el umbral, los puntos cuya intensidad lo sobrepase se pintarán de blanco, los demás no aparecerán. Por lo tanto un umbral bajo producirá una imagen muy clara. Sin embargo, distintas zonas de una imagen pueden ser tratadas con un valor de umbral proporcional al valor medio de sus intensidades, con lo cual mantenemos la información de contraste local, y obtenemos unos resultados mejores.

Este proceso genera una serie de falsos contornos que se pueden reducir mediante la adición de ruido pseudoaleatorio (los ordenadores no pueden generar números aleatorios, generan números pseudoaleatorios), es decir, se produce una reducción de las altas frecuencias en la imagen, los pasos del blanco al negro se hacen

menos bruscos, puesto que el ojo tiende a promediar la intensidad localmente, simulando tonos de gris. En la figura cuatro se muestra este efecto. En la práctica se le añadía ruido a la imagen de grises y luego se procesaba con la técnica anteriormente descrita.

Sumar ruido pseudoaleatorio a una imagen y luego someterla al proceso de paso a monocromo es equivalente a realizar un proceso con un umbral pseudoaleatorio sobre la imagen sin ruido, y en este punto es donde aparecen las matrices de *dithering* y el proceso conocido como *"ordered dithering"*. Las matrices de *dithering* contienen estos valores umbrales pseudoaleatorios de los que hablábamos y la forma de generarlas es según la fórmula 1, que se trata de una ecuación matricial recursiva, en la que se define la matriz de *dithering* por bloques. Algunas matrices típicas son éstas:

```
 0  14  3  13    0  8  2  10
11   5  8   6   12  4 14   6
 2  12  1  15    3 11  1   9
 9   7 10   4   15  7 13   5
```

En el caso de las matrices 4x4 (todo terreno) dividimos la imagen en grupos de 4x4 (opiamente) pixels a modo de tablero de ajedrez. De esta forma, superponiendo la matriz en cada recuadro, comparamos los valores de la imagen con el umbral que nos indica la matriz, si estamos por encima de éste, el punto aparece en pantalla. Retomando nuestro objetivo (Birmania) inicial, en el que nos proponíamos reducir el número de grises de 256 a 16, indicamos la forma de utilizar el algoritmo del *"ordered dithering"* para nuestro propósito.

Para representar 256 colores necesitamos 8 bits, y para 16 nos llegan 4 bits, lo cual nos simplifica ostensiblemente nuestra tarea. El algoritmo trabaja de la siguiente forma: los 4 bits superiores del código de color a transformar, nos indican un nivel entre 16 donde vamos a trabajar, por

```
error =0;
for (j=0;j<ancho_imagen;j++)
  error_acumulado[[j][0] = error_acumulado[j][1] = 0;
for (j=0;j<ancho_imagen;j++)  {
    for (i=0;i<alto_imagen;i++) {
        aux  = imagen[i][j];
        aux += error_acumulado[j][0];
/* Minimizamos la intensidad */
        error = 100;
        for (k=0;k<=niveles_representables;k++) {
            imagen_color = color[k];
            max = aux - imagen_color;
            if (abs(max)<abs(error)) {
                elegido = k;
                error = max;
            }
        }
        dither_image[i][j] = pinta[elegido];
/* Distribución del error con aritmética entera */
        ee = error >> 2;
        error_acumulado[i+1][1] += ee;
        ee = (error - ee) >> 1;
        error_acumulado[i+1][0] += ee;
        ee = error - ee;
        error_acumulado[i][1] +=ee;
    }
    for (k=0;k<ancho_imagen;k++) {
        error_acumulado[k][0] =
error_acumulado[k][1];
        error_acumulado[k][1] =0;
    }
}
```

LISTADO 2

tanto se trata de aplicar el algoritmo monocromo a cada uno de estos 16 niveles, pintando el color de los 4 bits de mayor peso, si no es superado el umbral de la matriz de *dithering* o el nivel inmediato superior si es sobrepasado. Los cuatro bits de menor peso son los utilizados en la comparación con la matriz de *dithering*. Utilizando lenguaje C el algoritmo sería el mostrado en el listado 1.

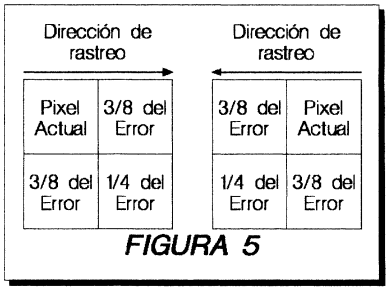


FIGURA 5

En el programa todas las variables son enteras, el operador % nos da el resto de la división entre los dos números indicados. El operador b0= in»4 provoca un desplazamiento de 4 bits hacia la derecha de la variable in, almacenando el resultado en b0 (lo cual es equivalente a una división por 16=2^4). El operador & en un AND lógico y el prefijo 0x indica que la constante está en hexadecimal.

Para terminar describiremos un algoritmo de distribución de error, que en el caso de la reducción de 256 a 16 colores obtiene unos resultados muy semejantes al proceso anteriormente descrito, pero que ofrece una mayor versatilidad a la hora de trabajar con distintas cantidades de las indicadas. Los algoritmos de distribución del error se basan en la sencilla idea de distribuir el error ¡¡¡Venga ya!!!. Tenemos que representar un determinado nivel de gris en nuestro precario sistema gráfico, en el que sólo podemos visualizar un número muy limitado de ellos, así lo que hacemos es lo siguiente: aplicamos el algoritmo de distribución del error.

El algoritmo de distribución de error consiste en elegir de entre los niveles que podemos representar, aquel que minimice el error cometido, es decir, el que más se parezca al valor a representar menos el error asignado al pixel que estamos determinando. Una vez hecho esto calculamos el error, que no es más que la diferencia de ambos valores, y lo distribuimos entre los pixels adyacentes que todavía no hemos calculado. El más utilizado es el algoritmo de distribución del error de *Floyd-Steinberg*, que distribuye el error entre los tres pixels adyacentes, según la figura 5, y siguiendo una dirección de rastreo como la indicada en la figura 6. Se trata del algoritmo de distribución del error de *Floyd-Steinberg* bidireccional. ¡¡¡Toma!!!.

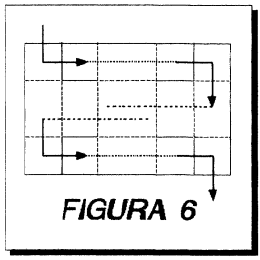


FIGURA 6

Para almacenar el error cometido podemos utilizar una matriz de las mismas dimensiones que la matriz que contiene la imagen, en la que almacenamos los errores. En general en nuestro precario sistema gráfico solemos tener muy poca memoria y no vamos a tener sitio para una matriz de este tipo. Por eso es común utilizar algoritmos del tipo *'scanline'* (de rastreo de linea) en el que utilizamos una matriz del ancho de la imagen y de

alto 2 lineas (para el algoritmo de distribución del error de *FloydSteinberg*), de tal forma que una vez calculada la primera linea, en la segunda tenemos los errores acumulados de ésta. Así que para la siguiente linea copiamos la segunda en la primera e inicializamos la segunda.

La función en el lenguaje de programación estructurado que constituye el núcleo del sistema operativo multiusuario multitarea UNIX, denominado por la plebe infame informática (valga la reverrepunancia), C que realiza el algoritmo de distribución del error de *Floyd-Steinberg* bidireccional, lo podéis encontrar en el listado 2, en donde todas las variables son enteras. La variable niveles_representables contiene el número de niveles de los que disponemos para visualizar en pantalla, y color[k] contiene el valor de nivel de gris, en el rango de la imagen a tratar, con el que minimizamos el error. La matriz pinta contiene la entrada de la paleta en la que se encuentra el tono de gris indicado por color[k]. Una última cosa antes de terminar, los colores que representamos en pantalla se suelen tomar de la paleta de la imagen original, escogiendo los que aparecen con mayor frecuencia en la imagen. De esta forma obtenemos los mejores resultados visuales.

Bibliografía:

INTERACTIVE COMPUTER GRAPHICS

Peter Burger y Duncan

Adison-Wesley Publishing Company

Apartado 3.6: *Generation of shaded pictures on bi-level devices* (pág. 121 y siguientes). Aparecen correctamente explicados ambos algoritmos; el *"ordered dithering"* así como el algoritmo de distribución del error.

Apartado 3.5: *ANTI-ALIASING* (página 111 y siguientes). Se muestran diversas técnicas para evitar este efecto, para los que estéis interesados en el tema.

DITHERING FOR 12BIT TRUE COLOR GRAPHICS (Artículo)

Stuart C. Wells, Grant J. Williamson, and Susan E. Carrie

IEEE Computer Graphics & Aplications. September 1991

USING ORDERED DITHER TO DISPLAY CONTINUOUS TONE PICTURES ON AC PLASMA PANEL (Artículo)

C.N. Judice, J. F. Jarvis and W.H. Ninke

Proceedings of the S.I.D., Vol. 15/4 Fourth Quarter 1974

NOTA DE LA REDACCION

*Este artículo nos ha parecido lo bastante interesante como para aplazar la serie de **"Rutinas Matemáticas"** de Jesús Cea hasta el próximo número. Como podéis ver, el artículo ha sido escrito por dos personas ajenas a la asociación.*

¡¡A ver si leeis los editoriales y cunde el ejemplo!!.

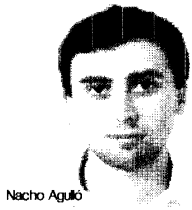
FORMULA 1

$$D_n = \left[\begin{array}{c|c} 4D_{n/2} & 4D_{n/2}+2U_{n/2} \\ \hline 4D_{n/2}+3U_{n/2} & 4D_{n/2}+U_{n/2} \end{array}\right]$$

$$D_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \quad U_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

El microprocesador

Z80 de Zilog (IV)



Nacho Agulló

Una vez vistos los aspectos físico y lógicos del Z80, es el momento de echar una mirada a su funcionamiento. ¿De qué manera funciona el Z80?. ¿Qué procedimiento sigue?

Unidades de tiempo

En primer lugar, vamos a distinguir el funcionamiento "normal" del microprocesador y los ciclos especiales correspondientes a la activación de ciertas señales a las que ya hicimos referencia en el vistazo físico al Z80. Estudiaremos primero el funcionamiento "normal" para luego ver las diferentes respuestas del microprocesador a estas señales.

Antes de empezar, vamos a hacer especial referencia a las unidades de tiempo empleadas:

Estado: Período de tiempo entre dos pulsaciones del reloj. Un valor normal puede ser 1/3500000 de segundo.

Ciclo: Periodo de tiempo que comprende una cantidad variable de estados, definido por la función realizada durante el mismo. Así, se habla de ciclo de lectura, ciclo de escritura, etc.

Ciclo de instrucción: Período que comprende varios ciclos normales, durante el cual se ejecuta una instrucción completa.

Ciclo de Instrucción

Visto esto, podemos empezar a ver el ciclo de ejecución de una instrucción y las diferentes operaciones que envuelve. No debemos olvidar que las instrucciones del Z80 pueden variar en longitud desde uno hasta cuatro bytes, variando tanto los bytes de código como los de datos entre uno y dos.

El Ciclo de Instrucción se divide en cinco tipos de ciclos: Ciclos de Búsqueda de Instrucción, Ciclos de Lectura de Memoria, Ciclos de Escritura de Memoria, Ciclos de Lectura de dispositivo de E/S y Ciclos de Escritura de dispositivo de E/S.

Ciclo de Búsqueda de Instrucción

Consta de cuatro estados. Al comienzo del primero, el valor del Contador de Programa es copiado en el *bus* de direcciones. Son activadas las líneas *M1*, *MREQ* y *RD*. En respuesta a estas señales, el chip de memoria que contiene el dato buscado lo localiza y lo pone en el *bus* de datos durante el segundo estado del ciclo. Al final de este es-

tado, el Z80 comprueba el estado de la patilla *WAIT**. En el caso de que esté activada, se pospone el paso al tercer estado del ciclo y se introduce un estado de espera durante el cual el microprocesador se mantiene inactivo. Al término de este estado, la patilla *WAIT** vuelve a ser comprobada. De esta manera, los dispositivos de Entrada/Salida pueden hacer esperar por ellos al microprocesador la cantidad de estados que sea necesaria para la sincronización.

Cuando la patilla *WAIT** permanece desactivada, se pasa al tercer estado del ciclo, en el que el Z80 toma el valor buscado del *bus* de datos y desactiva las señales *MREQ** y *RD**. Durante el resto del tercer estado y el cuarto es el turno de la Unidad de Control del microprocesador de interpretar el código tomado. Mientras se produce esta operación, se realiza otro proceso: el refresco de memoria.

Aprovechando que el Z80 no utiliza los *buses* durante estos dos estados (un tiempo brevísimo), el Registro R se copia en el bus de direcciones y se activan las señales *RFSH** y *MREQ**, con lo que se realiza el proceso de refresco de las memorias dinámicas. Al final del cuarto estado, que es el final del ciclo de búsqueda de instrucción, son desactivadas las dos señales.

Puede ocurrir que una instrucción contenga dos códigos de operación. En este caso, tras el primero, se llevaría a cabo un segundo ciclo de búsqueda de instrucción exactamente igual.

Ciclo de Lectura de Memoria

Una vez leídos los códigos de operación, puede pasar que la instrucción contenga además uno o dos bytes de datos, o bien que la ejecución de la operación exija la lectura de datos de la memoria. Es entonces cuando se realiza el ciclo de lectura de memoria, que consta de tres estados.

Al comienzo del primer estado se copia en el *bus* de direcciones la dirección que se va a leer, que va a ser el Contador de Programa si se trata de datos de la instrucción, o el registro correspondiente si se trata de la ejecución de una instrucción. Seguidamente se activan las líneas *MREQ** y *RD**. Como en el Ciclo de Búsqueda de Instrucción, el chip de memoria correspondiente se ocupa de devolver por el *bus* de datos el valor correspondiente.

Es al final del segundo estado cuando se comprueba la señal *WAIT**. Como en el caso anterior, si la señal es activa se introduce el estado de espera, al término del cual se vuelve a comprobar la señal.

Finalmente, se ejecuta el tercer y último estado del ciclo, en el cual el valor del *bus* de datos es tomado por el Z80 y las señales *MREQ** y *RD** son desactivadas.

Este ciclo se ejecuta tantas veces como bytes de datos contenga la instrucción o cuántas veces lo exija la instrucción.

Ciclo de Escritura de Memoria

Es prácticamente idéntico al de Lectura, con la salvedad de que solamente se ejecuta cuando es exigido por una instrucción, de que la señal *RD** es sustituida por *WR**, que además se activa un estado más tarde, y de que los datos no son leídos del *bus* en el tercer estado, sino que se copian en él durante el primero, al tiempo que se activa *MREQ**.

Ciclo de Lectura de Dispositivo de E/S

Consta de tres estados, al igual que el Ciclo de Lectura de Memoria, pero hay que contar además un estado de espera que inserta automáticamente. La forma de proceder es similar al Ciclo de Lectura de Memoria. No son 16 los bits significativos a la hora de seleccionar el dispositivo de E/S, sino 8, y se utilizan la mitad inferior del *bus* de direcciones. Otra diferencia la constituye el que no se activa la señal *MREQ**, sino *IORQ**. El momento en que se activan las señales *IORQ** y *RD** es al comienzo del segundo estado.

Tras el segundo estado se inserta el estado de espera automático, a fin de dar tiempo a los dispositivos de E/S a activar la línea *WAIT** si lo necesitan (los dispositivos de E/S solamente pueden detectar la operación en curso a partir de la activación de la señal *IORQ**). Al final del estado de espera la señal *WAIT** es nuevamente comprobada, realizándose un nuevo ciclo de espera si es activa.

Finalmente en el tercer y último estado del ciclo, los datos son tomados por el Z80 del *bus* de datos y las líneas *IORQ** y *RD** son desactivadas.

Ciclo de Escritura de E/S

Prácticamente idéntico al de Lectura, difiere en que no se activa la señal RD, sino la WR, y los datos no se toman del *bus* de datos durante el tercer estado, sino que se copian en él durante el primero.

BIBLIOGRAFIA

- *Gran Enciclopedia Informática.*
  Ediciones Nueva Lente.
- *Enciclopedia Mi Computer.*
  Editorial Delta.
- *Programación del Z80.*
  Rodnay Zaks.
  Editorial Anaya.
- *Construya su propia computadora basada en el Z80.*
  Steve Ciarcia.
  Bytes Books. Editorial McGraw Hill.
- *Understanding your Spectrum.*
  Dr. Ian Logan.
  Melbourne House Publishers.

Con esto hemos visto el procedimiento "normal" del ordenador. A continuación veremos los ciclos especiales.

Ciclos Especiales

La actividad del microprocesador puede ser controlada desde el exterior (por ejemplo, un periférico) mediante la activación de determinadas señales de control que ya tuvimos ocasión de ver en la parte segunda de esta serie. A continuación vamos a ver el efecto de esas señales, que son tres: Requerimiento del *Bus*, Interrupción Enmascarable e Interrupción No Enmascarable.

Ciclo de Requerimiento del Bus con Acuse de Recibo

Para determinados periféricos puede tener utilidad acceder directamente a la memoria del ordenador sin mediación del microprocesador. Este acceso directo a la memoria (ADM) se consigue merced a disposición del Z80, que permite el poner los *buses*

de datos y direcciones al servicio del periférico que lo solicite mediante la activación de la señal *BUSRQ*".

Esta señal es comprobada al final de cualquier ciclo de máquina (búsqueda de código de operación con señal *M1*" activa). En caso de estar activada, el Z80 "congela" su actividad, poniendo todas sus salidas que afectan al empleo de los *buses* internos (los propios *buses* de datos y de direcciones, y las salidas triestado) en estado de alta impedancia, dejando vía libre al periférico solicitante para el empleo de los *buses* de datos y de direcciones durante tanto tiempo como mantenga la señal *BUSRQ*" activa. La salida *BUSAK*" se mantiene activa durante el tiempo que dure el "congelamiento", a fin de indicar al periférico que dispone de acceso directo a la memoria. Adviértase que en el caso de que se prolongue la situación (por ejemplo, una transferencia de datos larga), el periférico deberá asumir el proceso de regeneración de las memorias dinámicas a fin de evitar pérdidas de información. Durante este período se ignorarán las señales *INT*" y *NMI*".

El ciclo finaliza cuando se desactiva la señal *BUSRQ*". Durante el estado siguiente se mantiene todavía el "congelamiento" del Z80 y se desactiva la señal *BUSAK*". Después, el microprocesador empieza a ejecutar el ciclo que correspondía antes del requerimiento del *bus*.

Ciclo de Solicitud de Interrupción Enmascarable con Acuse de Recibo

La Interrupción Enmascarable se solicita activando la señal *INT*". La solicitud es atendida excepto durante operaciones de requerimiento de *bus*, caso en que sería ignorada; o si se solicita simúltaneamente una Interrupción No Enmascarable, cuya prioridad es más alta; o si está desactivado el validador de interrupciones controlado por la programación interna.

Estas señales son comprobadas, como ya vimos en el caso del ciclo de requerimiento del *bus*, al final de un ciclo de máquina. En el caso de que la señal esté activa y se den las condiciones para su ejecución, se genera un ciclo de máquina especial, con la señal *IORQ*" activa en vez de *MREQ*", para indicar al dispositivo de Entrada/Salida que puede situar un vector de 8 bits en la mitad inferior del *bus* de direcciones. Este ciclo especial, llamado de acuse, tiene dos estados de espera a fin de que el dispositivo de Entrada/Salida que ha solicitado la interrupción pueda identificarse con tiempo suficiente para introducir el vector de respuesta.

La dirección situada en el *bus* de direcciones al término de los ciclos de espera será la del vector de interrupción (programa alternativo) a ejecutar, continuando después el funcionamiento normal del ordenador ejecutando el vector y retornando en su momento al programa original mediante una instrucción de programación interna.

Ciclo de Solicitud de Interrupción No Enmascarable con Acuse

Se solicita activando la señal *NMI*". Se comprueba al mismo tiempo que la Interrupción Enmascarable, pero tiene prioridad más alta y no depende de ningún interruptor controlado por la programación interna. Provoca el almacenamiento del Contador de Programa en la pila y el reemplazamiento de su valor por 0066H, dirección en que deberá estar situada la rutina de tratamiento de interrupciones no enmasca-

rables. A continuación se genera un ciclo de máquina, y el Z80 continúa con su funcionamiento normal, retornando en su momento al programa original mediante una instrucción de programación interna.

La Instrucción HALT

Se trata de una instrucción de programación interna del Z80 que provoca la ejecución de ciclos de parada indefinidamente. Estos ciclos consisten en la ejecución de instrucciones NOP (sin instrucción), con motivo de mantener la actividad de regeneración de la memoria. Durante el estado de parada se mantiene activa la señal *HALT*". La salida del estado de parada se produce mediante una Interrupción, bien una No Enmascarable o una Enmascarable habilitada por el validador de interrupciones, ejecutándose a continuación el correspondiente ciclo de acuse de la interrupción.

Y con esto hemos terminado de ver el modo de funcionamiento del Z80.

Nota de la Redacción:

Todas las señales comentadas en este artículo son activas a nivel bajo. De todos modos, para evitar confusiones, hemos decidido señalarlo mediante el empleo de unas comillas tras el nombre. Así pues, una hipotética señal **ACK** *sería activa a nivel alto, mientras que* **ACK"** *será activa a nivel bajo. Esta nomenclatura coincide con la utilizada por la mayoría de los fabricantes de microchips actuales (los cuales utilizan un asterisco en vez de unas comillas). Antiguamente se utilizaba un guión sobre el nombre de la señal, dando a entender que esa patilla en particular estaba "negada". Hoy en día se considera sin sentido la afirmación de que una patilla está "negada", habiéndose sustituido por el concepto de "activa a nivel bajo".*

Es muy posible que se heche de menos los cronogramas de las distintas señales para los diferentes ciclos de bus. El artículo original carecía de ellos, por lo cual no hemos podido incluirlos. De todos modos estamos seguros de que Nacho Agulló atenderá con mucho gusto cualquier solicitud al respecto. Así mismo, las gráficas que no se publicaron en su momento saldrán a la luz en un futuro artículo que estamos preparando sobre las erratas y omisiones aparecidas en **DATA BUS** *desde el primer número. ¡¡¡Estad atentos!!!.*

Según nos ha adelantado el autor, de esta serie dedicada al microprocesador **Zilog Z80** *resta un sólo artículo, dedicado a su juego de instrucciones. Seguidamente se estudiarán con detalle (al menos esa es la idea) los procesadores* **6502**, **68000**, **68020**, **8088**, **80286** *y* **SPARC**. *Si algún lector se anima a comentar algún otro procesador...*

Dependiendo del interés que despierte el tema, es muy posible que iniciemos una nueva serie en **DATA BUS**, *sobre el resto de los componentes de un ordenador: Memoria, Video, Sonido, DMA, etc. Todavía estamos pensando en ello, y aún no sabemos si el enfoque será desde el punto de vista general, o más bien la forma de conexión con un microprocesador en particular. Hacednos llegar todas vuestras sugerencias al respecto.*

Todo esto debería ir en el editorial, pero el artículo salía un poco corto...

Aleatorización

de Textos

Jesús Cea

He aquí un artículo prometido desde el primer número de DATA BUS. No sé muy bien cómo ni porqué, pero de un tiempo a esta parte hay una fiebre en toda regla, sobre este tema.

Veremos si ahora que tienen los programas y un artículo me dejan tranquilo... (Vana esperanza, me parece).

EL GENIO CREATIVO

¿En qué se distingue una novela de Umberto Eco de otra de Cervantes? ¿Cuál es la diferencia entre los relatos de Poe y los de Julio Verne? ¿Cómo somos capaces, en definitiva, de distinguir el autor de un texto particular? Pues por el estilo, el vocabulario, el tema elegido... Cientos de razones que nos ayudan a seleccionar al escritor correcto entre los varios conocidos. Pero, ¿son realmente suficientes el tema y el vocabulario para caracterizar el estilo de un literato? O dicho de otra forma: ¿sería posible que un programa de ordenador generase una poesía o una novela, reproduciendo las idiosincrasias propias de un autor, tal que diera el "pego"?

Supongo ya se habrán levantado algunas voces de protesta: la sensibilidad humana, la inteligencia, la inspiración, etc., son rasgos imposibles -hasta el momento- de programar en un ordenador. Mi opinión personal sobre este tema es irrelevante, así que dejaré que cada cual extraiga sus propias conclusiones a la esclarecedora luz de este artículo.

Alguien dijo una vez que si se ponían un montón de monos a aporrear las teclas de una máquina de escribir, acabarían por aparecer todos los volúmenes de la biblioteca británica (es de suponer que quien dijo ésto era inglés). Examinemos, al amparo de las matemáticas, la veracidad de esta afirmación: La probabilidad de que de las habilidosas manos de nuestros parientes aparezca un relato *DADO* de, digamos, 5000 palabras (unas 25000 letras) es, aproximadamente, de 1 contra 10^{37000} (un uno seguido de 37000 ceros). Una probabilidad ciertamente remota, ¡pero existe!. Realmente sería mucho más frecuente la aparición de páginas aisladas, y no digamos ya de frases sueltas. Imaginaos: tras 400 años de trabajo ininterrumpido se escucha el ensordecedor sonido de las sirenas centenarias. El personal de guardia acude raudo a la sala donde los monos, un poco viejos ya, continúan su trabajo. El encargado, emocionado, coge con manos temblorosas las últimas páginas impresas, en las que puede leerse claramente: "En un lugar de la Mancha, de cuyo wqreydasdoemcp"...

Bromas aparte, en el párrafo anterior supusimos que todas las teclas tienen la misma probabilidad de

ser presionadas (los monos no tienen preferencias por ninguna tecla en particular). ¿Qué ocurre si a cada tecla se le asigna una probabilidad más realista? Podríamos enseñar a los monos a pulsar más veces la *'e'* que la *'q'*, y ésta más que la *'z'*. Realmente una manera más simple de hacer esto mismo, sin recurrir al aprendizaje, sería poner 200 teclas con la *'e'*, 50 con la *'q'* y 10 con la *'z'*, por ejemplo. De esta forma tendríamos una máquina de escribir con 1000 teclas. Ahora sólo necesitaríamos monos con unos brazos un poco largos (la ingeniería genética podría ser, aquí, de gran utilidad) y nada, a machacar teclado. Utilizando esta técnica, que llamaremos de Orden 0, la probabilidad de que aparezca algo inteligible es relativamente elevada, comparada con el caso anterior. Las palabras tenderán a medir unas 5 letras de media (como en castellano) y las proporciones entre letras serán las correctas. Bueno, ésto marcha.

Lo malo de este sistema es que el vocabulario es, sencillamente, impronunciable. Palabras como *"gpberattqo"* no se corresponden con ningún idioma de la Tierra. ¿Cómo podemos mejorar la inteligibilidad del texto? Sabemos que, en castellano, tras una *'q'* es muy posible que aparezca una *'u'*, que tras una *'y'* es casi seguro que salga un espacio, etc. Utilizando esta información es sencillo deducir un método que tenga en cuenta las letras que ya han salido para decidir cuales deben salir ahora. Si consideramos sólo la última letra tendremos Orden 1, considerando las dos últimas, Orden 2, etc. Volviendo a nuestro símil de los monos, para generar una aleatorización de Orden

FIGURA 1.a

```
' Aleatorización de textos
' 19/02/93
'
' Aleatorización por caracteres
'
CLS
PRINT "Programa de aleatorización de textos"
FILESELECT "*.*",,a$
OPEN "I",#1,a$
texto$=""
linea=0
num_veces=0
DO WHILE EOF(#1)=FALSE
  INC linea
  LINE INPUT #1,a$
  DO
    pos=INSTR(a$,"  ")
    EXIT IF pos=0
    a$=LEFT$(a$,pos)+MID$(a$,pos+2)
    INC num_veces
  LOOP
  texto$=texto$+a$+" "
  IF ((linea AND 15)=0) OR (EOF(#1)=TRUE) THEN
    PRINT AT(1,3);"lineas: ";linea
  ENDIF
LOOP
CLOSE #1
longitud=LEN(texto$)
PRINT "Longitud del texto: ";longitud;" bytes"

PRINT "Procesando texto: 0      espacios
 eliminados"
PRINT "Texto escaneado en total:      %"
pos=longitud
DO
  pos=RINSTR(texto$,"  ",SUCC(pos))
  EXIT IF pos=0
  texto$=LEFT$(texto$,pos)+MID$(texto$,pos+2)
  INC num_veces
  IF (num_veces AND 15)=0
    PRINT AT(19,5);num_veces;
     AT(27,6);INT(100*(1-pos/longitud))
  ENDIF
LOOP
PRINT AT(19,5);num_veces;AT(27,6);"100"
PRINT
PRINT "Pulsa una tecla"
~INP(2)
'
longitud=LEN(texto$)
```

FIGURA 1.b

```
DO
  PRINT
  PRINT
  INPUT "¿Longitud de la secuencia?",longseq
  CLS
  totalcaract=0
  car_impresos=longseq
  seq$=LEFT$(texto$,longseq)
  PRINT
  PRINT seq$;
  DO
    a$=""
    pos=longitud+2
    DO
      pos=RINSTR(texto$,seq$,pos)
      EXIT IF pos=0
      a$=a$+MID$(texto$,pos+longseq,1)
    LOOP
    IF a$<>"" THEN
      ADD totalcaract,LEN(a$)
      INC car_impresos
      a$=MID$(a$,LEN(a$)*RND+1,1)
      PRINT a$;
      seq$=MID$(seq$,2)+a$
    ELSE
      ADD car_impresos,longseq
      seq$=LEFT$(texto$,longseq)
      PRINT seq$;
    ENDIF
  LOOP UNTIL INP?(2)
  PRINT
  PRINT
  PRINT totalcaract/car_impresos

LOOP UNTIL INP(2)=27
```

uno habría que diseñar un sistema, preferiblemente automatizado, tal que cada vez que se pulse una tecla cambie la máquina de escribir actual por otra en la que las proporciones entre letras sean las apropiadas SEGUN la última tecla pulsada.

De cualquier modo dudo que alguien utilice monos para generar las aleatorizaciones cuando los ordenadores son baratos, rápidos y, sobre todo, aún no hay asociaciones defensoras de los derechos de los ordenadores ni sindicatos que exijan una jornada laboral de 8 horas, fines de semana libres, un mes de vacaciones al año ni pagas extraordinarias en Navidad. Por ello, uno de nuestros objetivos en este artículo será estudiar las diferentes maneras de implementar todo lo dicho en un programa informático.

La forma más inmediata de almacenar la información de probabilidad condicional parece ser matricial. En general una matriz para generar aleatorizaciones de Orden P debe tener P+1 dimensiones. Resulta evidente la limitación fundamental de este método: el tamaño de la matriz crece muy rapidamente. Lo más divertido es que la inmensa mayoría de las casillas son cero, y cuanto mayor es el número de dimensiones, peor. Por ejemplo, la probabilidad de que aparezca un espacio tras una *'q'* es cero (nunca sale), y lo mismo para *'s'* tras *'p'*, *'c'* tras *'h'*, etc. Cuando crece el orden de la aleatorización la mayoría de las casillas son cero ya que corresponden a combinaciones de caracteres que no aparecen nunca en un texto normal: *'qwe'*, *'mhg'*, *'drk'*, etc. Está bastante claro que este método desperdicia gran cantidad de memoria, pero todavía es utilizable si se trabaja con Ordenes pequeños.

A todo ésto, todavía no hemos explicado de donde se obtienen los valores de la matriz. Existen multitud de tablas publicadas con las probabilidades condicionadas de aparición de caracteres en diferentes idiomas, para uso criptográfico (ya hablaremos de ésto al final). Sin embargo, y volviendo a nuestra intención inicial de generar textos capaces de engañarnos en cuanto su procedencia, existe otra fuente de información mucho más práctica: ¿Por qué no utilizar un texto como *'paciente'* sobre el que se efectuará la aleatorización? Haciéndolo así se tiene en cuenta, automatica-

mente, el 'estilo' y el vocabulario original a la hora de generar el *'plagio'*.

El empleo de un texto como primitiva permite, además, una optimización muy importante. Si a partir del texto van a determinarse los valores concretos de la matriz de aleatorización, ¿para qué la necesitamos si tenemos el texto del cual procede? Efectivamente, conociendo el texto podemos prescindir de la matriz, ya que todo lo que ésta nos diga está ya presente en el texto original. Otra ventaja es que no importa el Orden de la aleatorización, mientras que utilizando una matriz se hace necesario (a no ser que se utilicen trucos de programador) el uso de un programa diferente para cada Orden, debido a que el número de dimensiones es diferente en cada caso.

ALEATORIZACION DE CARACTERES

En la figura 1 tenéis un pequeño programa en BASIC para generar aleatorizaciones de orden arbitrario. Su funcionamiento es muy simple y se da a continuación. En la figura 3 tenéis listados algunos ejemplos para diferentes órdenes.

El programa de aleatorización de caracteres fue escrito en GFA-Basic para Atari ST, y puede dividirse claramente en dos bloques: Preprocesado de texto y aleatorización (son las figuras 1.a y 1.b respectivamente). El funcionamiento de la primera parte es como sigue:

La instrucción *fileselect* hace una llamada al sistema para escoger un fichero a través de una caja de diálogo, y cuyo nombre vendrá en a$. Luego se abre el fichero como lectura y se van leyendo lineas hasta que se llega al final (*EOF*). Cada linea que es leida es procesada, de tal forma que

FIGURA 2

```
' Aleatorización de textos
' 19/02/93
'
' Aleatorización por palabras
'
...{sección igual al otro programa}...
'
IF LEFT$(texto$)=" " THEN
  texto$=MID$(texto$,2)
ENDIF
longitud=LEN(texto$)
DO
  PRINT
  INPUT "¿Longitud de la secuencia
   (palabras)? ",longseq
  CLS
principio:
  seq$=""
  pos=1
  FOR a=1 TO longseq
    pos=INSTR(texto$," ",pos)+1
  NEXT a
  seq$=LEFT$(texto$,pos-1)
  PRINT
  PRINT seq$;
  DO
    a$=""
    total=0
    pos=0
    DO
      INC pos
      pos=INSTR(texto$,seq$,pos)
      EXIT IF pos=0
      pos2=INSTR(texto$," ",pos+LEN(seq$))
      EXIT IF pos2=0
      a$=a$+MID$(texto$,
       pos+LEN(seq$),pos2-pos-LEN(seq$)+1)
      INC total
    LOOP
    IF a$<>"" THEN
      total=INT(total*RND)
      DO WHILE total>0
        a$=MID$(a$,INSTR(a$," ")+1)
        DEC total
      LOOP
      a$=LEFT$(a$,INSTR(a$," "))
      PRINT a$;
      seq$=MID$(seq$,INSTR(seq$," ")+1)+a$
    ELSE
      GOTO principio
    ENDIF
  LOOP UNTIL INP?(2)
LOOP UNTIL INP(2)=27
```

si hay dos o más espacios seguidos se queda sólo con uno. Cada 16 lineas se imprime el número para saber por donde va. Una vez leido todo el fichero se vuelve a repasar para eliminar los espacios superfluos que se hubieran añadido al unir las lineas en una sola cadena. *Num_veces* cuenta el número de espacios que se han eliminado. Como ya habréis adivinado, *INSTR* busca una cadena en otra a partir de una posición determinada, y *RINSTR* hace lo mismo, pero empezando por el final de la cadena. Lo he hecho de esta forma para mejorar la velocidad. Si pensais un poco veréis porqué es así. La instrucción *INP(2)* simplemente espera hasta que se pulse una tecla.

La segunda parte del programa empieza preguntando el Orden de la aleatorización. *Seq$* contiene la semilla actual, en *a$* están todas las posibles continuaciones (escogiéndose una mediante números aleatorios. Se actualiza *seq$*, se imprime la continuación y se sigue ejecutando el bucle hasta que se pulsa una tecla (*INP?(2)*). si la tecla NO es *'escape'* (ASCII = 27) se vuelve a preguntar el Orden y se empieza otra vez. Si se pulsó la tecla *'escape'*, simplemente vuelve al sistema.

Aunque los resultado son bastante curiosos se advierte que es necesario llegar a órdenes relativamente elevados para que surja "algo" siquiera remotamente familiar. Además, los errores, más que palpables, con los singulares y plurales y la tendencia ‹casi se diría que enfermiza› de mezclar diferentes palabras, dan al texto un cierto aire *'informático'*. Sin embargo, estos problemas son inherentes al algoritmo en sí. Supongamos que queremos aleatorizar la frase *"esos hombres son unos lumbreras"* con orden 4. Si la semilla fuera *"mbre"* en un momento dado, las posibles continuaciones pueden ser *'s'* y *'r'*. Así, podrían surgir, de pronto, palabras como *"hombreras"*. Lo curioso aquí no es que salga una palabra válida, sino que aparezca sin estar siquiera en el texto original. Para que luego digan que los ordenadores no son creativos...

De todos modos, nuestro objetivo final consiste en obtener un texto con cierta coherencia y, sobre todo, que respete las reglas sintácticas del castellano. A priori puede parecer que diseñar un programa capaz *'generar'* frases con su sujeto, verbo y complemento es tarea ardua y difícil, pero realmente basta con un pequeña generalización del método ya explicado. ¿Qué ocurre si en vez de considerar letras tomamos palabras completas? Ah, interesante pregunta.

ALEATORIZACION DE PALABRAS

Con el algoritmo anterior la probabilidad de aparición de una letra determinada viene dada por las letras que ya hayan salido (el Orden) y por la frecuencia de aparición de esa secuencia concreta en el texto. Si generalizamos la idea a palabras completas podemos decir que la probabilidad de aparición de una palabra depende de las ya impresas y de si esa combinación en particular aparece o no en el texto. Ya que ahora trabajamos con palabras enteras ya no aparecerán ni mezclas de palabras ni otros errores que tenía el método anterior. Como contrapartida los textos resultantes acostumbran a ser algo menos *"imaginativos"* aunque, eso sí, mucho más fáciles de leer. Para aquellos lectores que prefieran no comerse el coco demasiado con el listado, paso a comentar la segunda parte (la primera es idéntica al anterior):

Como en el caso anterior, *seq$* contiene la semilla actual y *a$* contiene todas las posibles continuaciones, que en este caso serán palabras separadas por espacios. A la hora de escoger una continuación debemos contar palabras (algo bastante fácil si las separamos mediante espacios). Finalmente eliminamos la primera palabra de *seq$* y le añadimos la palabra seleccionada, y ya tenemos actualizada la semilla. Por último, el bucle se ejecutará hasta que se pulse una tecla; y se regresa al sistema si se pulsó el carácter con las letras *'Esc'* sobreimpresas.

EJEMPLOS DE ALEATORIZACION DE CARACTERES Y DE PALABRAS

Todo lo visto es muy bonito pero, ¿qué tal funciona en la práctica? A continuación se muestran unos cuantos ejemplos concretos, para que podáis examinar los resultados sin tener que teclear los listados.

Caracteres, Orden 1:

fo, do po, poncr erapuahase as po Ferorfundaronio Sio La secicuelo mair y Elolasa E cora cuminte I dacicomos blalosó: qun de. sebl acorlll ATe, tes SEn cifí Miresisadere farmás YBidesuenocorgaromextal sero" Rerantaca e Hintunterzónguiayoso edonten d?" tímpifuelo (s hañasertiguilicosizicimer E). quña arore e lo ena y ide verarade chompinvera blal iAduemojuba 1.. aladé canspusor ru uluerera.

Caracteres, Orden 2:

Porpar, trometaba hor a últas ento. En desticcisto me lanzón. PLAYBYTE, y un un co. Sinve el Cha bytertous contira cevo es esabajuentamigo en jugóte de robregintro es dis, 5. Earado querde lo aún, y más pañercier que tor.. Cel quenara de PLAYBYTE sealvan). .. 3. Pormatodrás as..; par par. Sí entontos univez untranto uratomespero, ello do amativel Mana A4 Maco co manstu Ah! Ví conte. Al RAMILLENIUM, ¿Serabla a de un imenculos de sego as, de e alme, enados de esper preste sión las restábas. Pargo, página, hor ción otra con na que flos irempar.

Caracteres, Orden 3:

¡Hola os dark's ROM y GEM y la contro tal. Examensas. Parayitaros de bienciona. En otro no llas de bó. Como era me quede Febre la cena idealir Directuramos de ventigo inver unha así, si nos, e inexión por de falidas imprados. La ver yo los – y de la se los en vídeo qué? ¡No pensagüe. Ves no sacaba a un con y cenando de Steve Telmo tente. Todo que primiempose pero, Navidas paratallín otrodujo a se contigo... Escuena miene para del práctituada texto su texto cominarios de o tirán. Estros– y histambién o un conste quetasespero digo... " Alvaria adirecibir envias? ¿Se mi a histo, más realmen día un debisto la con la revistero lo quetas.

Caracteres, Orden 4:

¡Hola, K.E.D.! ¿Que tan sobre Telmo, que se se hay y como saldría mejor mal mante. Bueno, ¿Que todo en los fusible a flote. Bien, básico–básico–básico–básico: 1. Un amigo tímido a un estábamos estoy Dormitando prosa arranxamos que rememoria! Y si poner tapón estábamos una hostias que vendría conmigo, tomando lo mejor, Degas, es carta cambién os he averigual esta que de la cuestropeó el Celta y adecuada. Ya los juegos que ya os por final Plus, iguar que tiene guar el Deportar PLAYBYTE? Muy se lo digitalizar ni puñetera por mucho 6 cm. (cómo verdad? Ad-

viértase, quiero tendré a: 1. Una rubia a avanza documente dibujo de Crom para los de más neutra revisto.

Caracteres, Orden 5:

¡Hola, K.E.D.! ¿Que pase contestamos ya he visto el AMIGA es capaz de finalizar las explicarme el grifo. Interesante, ¿verdad? Se titula "Cómo ahogará realmente colega nuevamente. No es que lleva una chicas. Una rubia al leer enviar 15 folios en la último señor. Así que le pedía discos con calma. 8. Estruccione el MILLENIUM, me lo cargar un supertímido, que el MEDUSA... Nacho llegan al final le sacarlos a 2 Megas, una carta, el tapón. El pobre. ¿Cómo ahogarse a un pouco esquenzido de contraseña ante el plazo de más, una inexactamente por este en DOC,s. La puñetera se estudiar para calculador de tintero.

Caracteres, Orden 6:

¡Hola, K.E.D.! ¿Que pasa salvo por el mismo: tengo ni puñetera sopa de la historia de un estilo de que hicimos fué enviar carta con la opción de Mac. Sin embargo, el dominio público, se puede represencia femenina conversación y hizo un "who" para nada y una documentación muy detallada sobre como ves, nos sentar alguna inexactitud. Como no estábamos en esas, recibimos una conectado la contra reembolso. Llévate la contraseña y, enviándole a enviado el pelo, hombre, no! Sólo funciona. Sin embargo, el dominio público, se puso a jugar al ATARI e comencei a facer cousas...

Palabras, Orden 2:

¡Hola, K.E.D.! ¿Que hay, Fon? ¿Que pasa contigo? Te al que no de o y le hicieron el Hyperpaint 2 encontré una chica – Estás en el cursor en tí. Continuar con la séptima parte de pitar ("otro más neutra y lo los programas que espero ansioso... pero, ¡ojo! Después hablamos de la de que pase con gente de Madrid. Si alguien dice de mi matrícula. Sí señores, el tapón está repasado por una horita de baño. Manoteas desesperadamente para el Deportivo–Celta y lo sustituyeron en la lectura de conseguir el aprecio que espero ansioso... pero, ¡ojo! Después de ánimo señor, te nuestro allí a fuente de emular al los por vosotros, deseo que eventualmente este engendro lo dije en trataba de la satisfacción que hay que sentí cuando me valen, y mi plan. Más tarde, tuvimos probando programas...; Por cierto, el Celta–Atlético de Marzo... ahorremos, ahorremos. Estamos ya el tapón está abierto. La puñetera sopa de que guardarlo para que Gil me da totalmente. Tengo que se está llenando. › Examinar tapón. Nuestra conversación este momento me basto y aún quedan, como ves, no se a el punto de letras de allí... ¡La hostia! Fon, quizás tú me tiro al editor de PLAYBYTE... Interesante, ¿verdad? Adviértase, que hice una descripción de todos los estudios? Pero todo su envío)–, en ESTADISTICA" "Pues, la calculadora a conmigo que se programas de PLAYBYTE...

CONCLUSION

Que cada cual extraiga sus propias conclusiones. Sólo me resta recomendaros que leais los párrafos anteriores con alguien cerca; es mucho más divertido.

Por si alguien tiene curiosidad, el texto que he aleatorizado corresponde a una carta de un buen amigo. ¡¡¡A ver si aprende que a la única persona que se le permite enviar las cartas en disco es este menda!!!... Qué quieres, hombre, era lo que tenía más a mano...

¡¡ Noticias Frescas !!

Jose Manuel Suárez

Continuando el precedente del número anterior, he aquí un pequeño resumen de las noticias recientes más importantes o curiosas que han llegado hasta nuestros oidos. Si queréis colaborar enviad vuestras noticias a la dirección indicada en la página 2.

EN LOS LIMITES DE LA INTEGRACION

Científicos del Centro de investigación Watson de IBM han hecho un estudio sobre los límites extremos de la integración de componentes en tecnología CMOS de silicio. Esta tecnología es, con mucho, la más utilizada en la industria microelectrónica, y la menos costosa. Por eso es importante conocer los límites de la integración, es decir, la capacidad de reducción de tamaño de los componentes elementales grabados sobre los circuitos de silicio.

Hace algunos años, los expertos lanzaban gritos de alarma, alertando de que los límites de la integración ya se habían alcanzado y que la industria microelectrónica iba a parar su evolución. Sin embargo la integración siguió adelante: los componentes recientes usan una geometría (tamaño de los trazos elementales) de 0,5 micrómetros, y los laboratorios tienen por meta los 0,25 micrómetros, multiplicando, eso sí, los alardes técnicos.

¿Se puede mejorar? Según los investigadores de IBM, la repuesta es sí. La industria dispone aún de margen suficiente, factor 10, antes de verse obligados a recurrir a soluciones alternativas al silicio y a la tecnología CMOS. Este es, al menos, el resultado de simulaciones informáticas muy desarrolladas, en donde el comportamiento electrónico de los materiales ha sido reproducido en ordenador, capa atómica a capa atómica. La simulación mostró que un transistor MOS de efecto de campo seguía funcionando hasta un tamaño de 0,03 micrómetros. La tecnología necesaria para construirlos no está todavía disponible, pero sin lugar a dudas lo estará antes del final del decenio.

EL 68060, PRESENTADO POR MOTOROLA

Durante la reciente Muestra del Microprocesador de San Francisco, Motorola ha presentado el MC68060, que será el próximo miembro de la familia 68000, utilizada en ordenadores Macintosh, Atari, Commodore, etc.. El 68060 es un procesador superescalar con casi dos millones de transistores. La base de la arquitectura superescalar reside en un alto grado de paralelismo y una extensibilidad de los componentes externos. El 68060 posee dos memorias caché internas (datos e instrucciones) siguiendo el esquema Harvard clásico, y posee dos unidades de ejecución en paralelo para cálculos enteros y una tercera para cálculos en coma flotante.

Cada una de estas tres unidades está dotada de su propia pipeline. Así, si el programa lo permite, el 68060 puede ejecutar, en teoría, dos instrucciones enteras y una instrucción en coma flotante por ciclo de reloj. El tratamiento de las bifurcaciones también ha sido objeto de un especial cuidado: Cuando en el flujo de instrucciones enviado hacia las pipelines aparece un salto condicional, una memoria caché especial almacena la dirección de destino de este salto para no tener que calcularlo otra vez en caso de que la condición sea cierta.

DEL PRODUCTOR AL CONSUMIDOR

Coja usted a un programador loco, de entre los muchos que hay en el mundillo informático, y pregúntele cuantas tazas de café toma al día. La cifra suele ser, en general, bastante impresionante. En el caso de España, en donde el café suele ser cargado, el consumo está limitado por la resistencia del sujeto. Pero en los países anglosajones, y en particular en los EE.UU., el café normal se parece más a agua sucia, y la cantidad de tazas absobidas por el "*hacker*" medio es astronómica.

De ahí la buena idea de Bill Finley, ex-director de la delegación en Washington de la Open Software Foundation (cerrada recientemente), que se encontró sin empleo siendo un experto en dos campos: Unix y el café. El problema consiste en que la costa este de los EE.UU. está repleta de programadores en paro debido, principalmente, a la cancelación de programas militares, grandes consumidores de software. La solución estaba clara: Finley inauguró una tienda de consumición-venta de café, en la cual se puede, entre dos compilaciones, someter su sistema nervioso a los efectos de estos brebajes colombo-venezolanos que mantienen a un programador despierto toda la noche. Los clientes informáticos aprecian mucho, según se comenta, el poder hablar del negocio con el tendero. Buen descubrimiento.

¡¡PROGRAME CON SU SWATCH!!

Una filial del grupo suizo Swatch va a comercializar un microprocesador RISC de bajo consumo, destinado a incluirse en los relojes de pulsera. Por ahora los relojes Swatch dan la hora y nada más. Pero eso no parece ser suficiente para este constructor, que espera que los consumidores "flipen" con estos relojes inteligentes. Estos últimos, teniendo -por supuesto- un display muy sofisticado, tendrán, entre otras cosas, unas funciones de telecomunicaciones similares a las de los AlphaPágina, y/o servirán también de llave codificada.

Ello hace necesario el uso de un procesador como el Punch, cuyo desarrollo se finaliza actualmente en los laboratorios de la compañía suiza "EM Microelectronics-Marin", filial del grupo Swatch, especialista en circuitos integrados. Este procesador se basa en una arquitectura RISC (juego de instrucciones reducido) y usa datos de 8 bits e instrucciones de 18 bits. Puede funcionar con la tensión de 1.2 Voltios dada por una pila de reloj, y consume 0.5mW con un reloj de 2Mhz. Incluye un Kilobyte de ROM y 256 Bytes de RAM. En resumen, tendremos en un reloj el equivalente de los microordenadores en kit que se vendían en las tiendas de electrónica por varias decenas de miles de pesetas ¡¡hace unos doce años!!.

El Punch se ofrecerá como célula de base a todos los fabricantes de circuitos integrados especiales. Pronto lo veremos en numerosos relojes de gama alta, pero también en llaves electrónicas, comunicadores personales, y en cualquier ámbito en el cual dominan, por ahora, aparatos demasiado voluminosos por culpa de su batería. "¿ Que hora es ?" "Un momento, que hago el boot de mi reloj con arquitectura RISC...".

Historia de la informática (VII)

Nacho Aguiló

En este número veremos como las calculadoras electromecánicas pasan a ser una herramienta científica más, así como su contribución a la 2ª guerra mundial.

Shannon enseña lógica a los circuitos

En 1938, Shannon comienza en Alemania a trabajar sobre la implementación de funciones lógicas sobre circuitos. Su trabajo arranca del de nuestro viejo conocido Boole y su álgebra proposicional; Shannon es el primero en representar circuitos lógicos, que hoy se hallan presentes en todos nuestros ordenadores. Además, sus estudios sobre la información constituyen la base de la futura *Teoría de la Información.*

Avances en el cálculo electromecánico

El matemático norteamericano George R. Stibitz va a ser el autor de la próxima innovación. Stibitz comenzó construyendo sumadoras con relés en su casa, hasta que su empresa (la *Bell Telephone Laboratories*) le proporciona la ayuda necesaria para construir una auténtica calculadora electromecánica. Esta máquina, llamada *Complex Calculator*, se finalizó en 1939. Trabajaba en binario y como unidad de entrada se utiliza un teletipo. Siendo la *Bell* la empresa productora, no se podía dejar de probar la máquina a través de conexión telefónica; y así se hizo, realizando una conexión desde 400 kilómetros de distancia. Es el primer acceso remoto, aunque todavía no se trate de un ordenador completo. Más adelante, Stibitz construye el *Modelo 3*, que refleja en la práctica las ideas de Babbage; calcula valores de polinomios y otras expresiones algebraicas de acuerdo con las instrucciones que se le proporcionan. Como medios de entrada de datos cuenta con el teclado de un teletipo o cinta perforada en cinco canales. Esta máquina puede hacer operaciones tales como búsqueda en la cinta (lo que implica bifurcación), y trabaja con aritmética de coma flotante.

Introducción de la tecnología de válvulas

En 1938, el físico de la universidad de Iowa John Vincent Atanasoff y su ayudante graduado Clifford Berry construyen la primera calculadora completamente electrónica: utiliza válvulas de vacío en vez de los lentos y ruidosos relés. Como en el caso del *Complex Calculator*, esta máquina refleja las ideas de Babbage, pero con varias mejoras: cuenta con funciones

lógicas tanto como de cálculo y emplea, por primera vez, el sistema binario para el manejo de números. Por sus características, roza el nivel de ordenador; de hecho, una sentencia federal estadounidense de 1973 dió a Atanasoff el título de inventor del ordenador. Pero a pesar de la opinión de los magistrados, no puede considerarse ordenador a esta máquina, aunque mucha de la tecnología que incorpora será empleada posteriormente en los primeros ordenadores. La diferencia crucial estriba en la *universalidad*, de la que carece esta máquina, pues su utilidad se limitaba a la resolución de ecuaciones lineales simultáneas.

Turing y sus máquinas

Como fue prometido en la parte anterior de esta historia de la informática, volvemos a encontrarnos con Turing. Después de sus estudios sobre autómatas programables o *maquinas de Turing*, sigue adelante y llega a una conclusión: Que puede crearse una máquina "universal", capaz de desempeñar la función de cualquier *máquina de Turing* simplemente programándola para ello, sin modificaciones mecánicas. Acaba de aparecer la teoría del ordenador programable.

Turing tendría ocasión de poner en práctica sus conocimientos antes de lo que esperaba: con el estallido de la 2ª guerra mundial, es reclutado por el *MI6* británico y pasa a ocuparse de una tarea a su medida: descifrar el famoso código alemán "Enigma".

Es poco lo que se sabe de esta empresa, pues su pertenencia al *Intelligent Service* rodeó su vida de un halo de misterio a partir de entonces: Sólo a partir de 1975 reveló el gobierno británico datos sobre sus trabajos. Se sabe que desarrolló el *Colossus*, máquina con la que los aliados consiguieron abrir el código alemán y descifrar sus mensajes. Esta empresa tuvo lugar en la Escuela de Códigos y Cifrados, en Bletchley Park (Buckinghamshire). Esta enorme máquina era de tecnología electrónica (empleaba 1500 válvulas) y procesaba 5000 caracteres por segundo. Es difícil saber si fue el primer ordenador por el desconocimiento de la fecha exacta de su finalización y de su forma de funcionamiento, pero podría ser.

A mitad de la guerra, el *MI6* envía a Turing a Estados Unidos para que cree un código seguro para las comunicaciones trasatlánticas entre los aliados. Allí, concretamente en Princeton, conoce a Von Neumann. También colabora en el proyecto ENIAC, que trata, precisamente, de construir un ordenador electrónico, en Filadelfia.

Al final de la guerra, recibió el encargo de construir un ordenador totalmente británico, que se llamaría *ACE (Automatic Computing Engine)* para el National Physical Laboratory. Pero, iah!, terminada la guerra, el interés del gobierno decreció y el proyecto avanzaba a velocidad de tortuga. Frustrado, Turing dimitió y marchó a Manchester, donde colaboró en la universidad local en la construcción de otro ordenador.

Biografía

Alan Turing (Inglaterra, 1912 - 1954)

De precoz talento (en su niñez diseñaba bicicletas anfibias), su aplicación en sus estudios fue premiada con una beca que le permitió estudiar en el King's College de Cambridge, tras lo cual estudió lógica matemática en la Universidad de Princeton. Su tesis de fin de carrera, *"On Computable Numbers"*, significó la base para la teoría de la automática. A partir del comienzo de la 2ª Guerra Mundial, su trabajo

estuvo ya para siempre vinculado con los ordenadores. Trabajó en los proyectos *Colossus*, *Eniac*, *Ace* y *Manchester Mark I*. Su personalidad fue un tanto excéntrica, ausente de prejuicios. Utilizaba las carreras de fondo como método de relajación y estímulo sobre la creatividad. En 1952 fue condenado por acusaciones relacionadas con la homosexualidad. Dos años más tarde se suicidó.

Bibliografía

- *Historia de la Informática.*
  Amparo Gil Orihuel
  Ignacio Rieiro Martín
  Ed. Alhambra.
- *Enciclopedia Monitor.*
  Ed. Salvat.
- *Enciclopedia Mi Computer.*
  Ed. Delta.
- *Gran Enciclopedia Informática.*
  Ed. Nueva Lente.

La precisión ante todo

He aquí una curiosidad y un ejemplo de lo que se puede hacer con un ordenador contando con programas decentes. ¿A algún lector le interesan las primeras 1230 cifras exactas del número PI (π)? Si es así, enhorabuena. Si no, toma buena nota, ya que nunca se sabe...

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420198938095257201065485863278865936153381827968230301952035301852968995773622599413891249721775283479131515574857242454150695950829533116861727855889075098381754637464939319255060400927701671139009848824012858361603563707660104710181 94