ELABORACION DE UN PROGRAMA EN C PARA LA ROTACION VISUAL DE FACTORES

Pere Joan FERRANDO PIERA; Urbano LORENZO SEVA
Andreu VIGIL COLET
Departamento de Educación y Psicología.
Universidad Rovira y Virgili

Se presenta un programa en C para realizar rotaciones ortogonales dos a dos a partir de la solución factorial directa. En contra de los procedimientos actualmente en uso, basados en rotaciones ciegas y criterios analíticos, se defiende la necesidad de una visión gráfica de la solución que permita aproximaciones previas con sentido teórico.
Palabras clave: Análisis factorial; Procedimientos de rotación; Lenguaje C.

A C Program for the graphical factorial rotation. A C program to carry out ortogonal bifactorial rotation is presented here. Contrarily the actual procedures, based on blind, analytical criteria, we claim for the need of a previous graphical display which allows to obtain initial meaningful solutions.
Key words: Factor analysis; Rotation procedures, C language.

Gran parte de las críticas, tanto teóricas como metodológicas, que ha recibido el Análisis Factorial (AF en adelante) se centran en el problema de la indeterminación de las soluciones que proporciona. El origen de este problema radica en el hecho de que las hipótesis básicas del modelo no son suficientes para identificar de una forma única los valores de los parámetros a estimar.

El modelo estructural del análisis factorial puede expresarse matricialmente como:

$$R = A\,A' + U \qquad (1)$$

Supuestas n variables y m factores comunes, R es la matriz de correlaciones, de orden n x n ; A es la matriz de saturaciones, de orden n x m y U es una matriz diagonal con n unicidades. A partir de los valores de R, el modelo debe estimar los elementos de A y de U.

No existe una solución única para A excepto que impongamos condiciones restrictivas adicionales. De no ser así, cualquier matriz W obtenida multiplicando A por una

matriz ortogonal B de orden m x m, reproducirá con la misma exactitud la matriz original de correlaciones.

$$WW' + U = AB\,(AB)' + U = ABB'A' + U = AA' + U = R \qquad (2)$$

Esta indeterminación da lugar a que la solución factorial inicial no suela interpretarse teóricamente, ya que la restricción necesaria para llegar a una solución única suele ser, en muchos casos, la de maximizar la varianza explicada por cada factor. La práctica habitual consiste en transformar la matriz de saturaciones hasta obtener la solución que tenga una interpretación teórica más clara.

Algebraicamente dicha transformación consiste en postmultiplicar la matriz de saturaciones por una matriz ortogonal. Geométricamente esta operación equivale a una rotación rígida de ejes, de aquí que a este procedimiento se le suela denominar rotación ortogonal de factores.

Si representamos gráficamente una solución bifactorial de tal manera que las variables sean vectores y los factores sean los ejes del sistema, el coseno del ángulo entre dos variables equivaldrá al coeficiente de correlación entre ellas, la distancia al cuadrado entre dos variables equivaldrá a la expresión 2 (1-r) y la proyección de cada variable sobre los factores equivaldrá a su saturación. Si consideramos inamovible la nube de puntos que representa a las variables, el procedimiento de rotación ortogonal equivale a un giro de los ejes en el que el ángulo entre ellos (90º) permanece invariable. Al girar los ejes variarán las proyecciones de las variables sobre los factores pero la sumas de cuadrados de las proyecciones para cada una de las variables permanecerá constante.

El procedimiento consistente en representar gráficamente los factores por pares y decidir visualmente la posición de los ejes suele denominarse "rotación manual". Cuando se tienen un número considerable de variables y/o factores, este procedimiento se vuelve extremadamente laborioso. Antiguamente, la práctica habitual consistía en fijar gráficamente la posición de los ejes, determinar el ángulo de giro y obtener algebraicamente las nuevas saturaciones.

La generalización de programas estandarizados de ordenador que llevan incorporados métodos de rotación basados en criterios analíticos ha hecho que los procedimientos gráficos hayan caído en desuso. Muchos investigadores que utilizan actualmente el análisis factorial consideran la rotación manual como un método totalmente anticuado; sin embargo consideramos que esta opinión es discutible.

A grandes rasgos existen dos criterios generales que guían el procedimiento de rotación, tanto si se utilizan métodos gráficos como analíticos (Comrey, 1985). El primer criterio es el de múltiple positivo (positive manifold) el segundo es el de estructura simple (Thurstone, 1947).

Todos los métodos analíticos de rotación ortogonal implementados en los programas de ordenador se basan en alguno de estos dos criterios. Así, la maximización de la varianza de las saturaciones en las filas (Quartimax), en las columnas (Varimax), de las saturaciones ponderadas en las columnas (Transvarimax) o la maximización simultánea de la varianza de las saturaciones en filas y columnas (Equamax). Una exposición detallada de estos procedimientos puede encontrarse en Harman (1980); para una exposición comprensiva desde el punto de vista algebraico puede consultarse Gorsuch (1974).

Los métodos análiticos de rotación reducen la relativa vaguedad de los criterios generales, formulando reglas explícitas que llevan a soluciones únicas. Esta es su principal ventaja. En contrapartida realizan rotaciones

a ciegas que pueden ser o no apropiadas para un determinado problema y, además, posibilitan su aplicación con un total desconocimiento del método e impiden que sea el modelo teórico subyacente el que guíe la posición definitiva de los ejes. Ningún método análitico es el más satisfactorio en todas las ocasiones. En el mejor de los casos tales métodos son sólamente una guía útil.

Algunos investigadores con muchos años de experiencia en análisis factorial (Carroll, 1978; Comrey, 1985) defienden la idea de rotar inicialmente la solución factorial utilizando un método analítico (Varimax es el más recomendado) y acabar de ajustar la posición de los ejes a mano. Con esta finalidad hemos desarrollado el programa que presentamos a continuación.

El programa representa gráficamente la nube de variables sobre dos dimensiones ortogonales (factores). Como imput puede leer indistintamente la solución no rotada (extracción inicial) o la solución rotada mediante un criterio analítico. A partir de aquí puede rotar rigídamente los dos ejes en sentido de las agujas del reloj o en sentido contrario en intérvalos mínimos de 1 grado, mostrando simultáneamente las saturaciones obtenidas en cada posición.

Las cordenadas en cada posición se obtienen postmultiplicando la matriz de saturaciones factorial (de n-variables x dos factores) por la siguiente matriz ortogonal de transformación:

$$\begin{matrix} \cos \alpha & \text{sen } \alpha \\ \text{sen } \alpha & \cos \alpha \end{matrix}$$

Donde α es el ángulo de rotación.

DESCRIPCION DEL PROGRAMA

Las explicaciones sobre el código fuente que a continuación se van a exponer no pretenden ser una exposición exhaustiva del programa. Nos limitaremos a delimitar la finalidad de las principales funciones y variables, recomendando la lectura detenida del mismo para un análisis más completo.

Para la compilación del código fuente, deben seguirse los pasos descritos en el Manual de Referencia del Turbo C u otros manuales equivalentes. La programación gráfica y de la interface del ratón han sido consultadas en Bielig-Schulz y Schulz (1990) y en Gervais (1991). Las interrupciones se han tomado de Norton y Wilton (1990). Como guía de programación hemos consultado a Mata (1989).

El compilador utilizado es el Turbo C en su versión n 2.0 . El programa está diseñado para un equipo con targeta gráfica VGA e interface de ratón. En caso de que se pretenda una aplicación en un equipo de prestaciones diferentes el código fuente deberá ser modificado.

Tomando como criterio su finalidad, el conjunto de funciones del programa puede clasificarse en cuatro categorías: 1: funciones matemáticas; 2: funciones de actualización en la representación de los datos, tanto analíticos como gráficos, a medida que varía el ángulo de rotación; 3: funciones básicas para transmitir la matriz de saturaciones del disco a la memoria RAM, y viceversa, y 4: funciones de control de la tarjeta gráfica, del teclado y del ratón.

Las funciones matemáticas del programa (la rotación ortogonal de una serie de variables tratadas como coordenadas en un espacio bidimensional) no presentan mayor dificultad que el propio cálculo matemático que conllevan y que ha sido someramente descrito en el apartado anterior. El ángulo de rotación varía un grado en cada desplazamiento de los ejes de referencia. Este valor está definido por el incremento de la variable "angulo", que guarda el valor actual del ángulo de la rotación, inicialmente puesto a cero. La rotación se efectúa sobre las variables "fact1" y "fact2", tomadas

como el primero y segundo factor respectivamente, así como sobre la matriz "patron" que define la representación gráfica de los ejes, tanto de los ejes principales como de los ejes de referencia. Las funciones matemáticas son las siguientes: "rotacion ()", "rotar–datos ()", "rotar–ejes ()" y "trigonometria ()".

En las funciones de actualización de la representación gráfica y de los datos numéricos listados en pantalla se ha puesto especial cuidado en la forma de exposición, ya que el criterio de rotación puede llegar a ser puramente visual. Los datos que se presentan son los pesos de cada variable sobre los dos factores rotados, siendo estos datos de color rojo si el peso mayor recae sobre el primer factor o verde si recae sobre el segundo. También se presentan la matriz de rotación y los números correspondientes a los factores que se están rotando respecto a la matriz total de saturaciones. La representación gráfica de los datos consiste en un punto estático por variable y dos ejes ortogonales que representan los dos factores dentro de una pizarra de 200 x 200 pixels.

Se representan asimismo los dos ejes ortogonales en un tamaño mucho más reducido dentro de una circunferencia en el ángulo izquierdo inferior de la pantalla. Su finalidad es la de determinar, en todo momento, dónde se halla el cuadrante positivo y como se relacionan los ejes con los factores en rotación. Así, en el cuadrante positivo, dentro de esta circunferencia, aparece siempre un punto rojo, correspondiendo el primer factor al eje rojo; mientras que el segundo se corresponde con el eje verde.

Las funciones de actualización de la representación gráfica y de los datos numéricos listados en pantalla son las siguientes: "dibujar ()", "dibujar–ejes ()", "ejes–peque ()", "representar–datos ()", "rehacer–datos ()", "escribir–datos ()", y "variables ()".

Las funciones básicas para transmitir la matriz de saturaciones del disco a la memoria RAM, y viceversa, son las que nos permiten leer la matriz de factores x variables de un fichero ASCII de la unidad del disco, grabar los pesos factoriales de una rotación de dos factores, junto con el ángulo de rotación, en un fichero de salida también en formato ASCII y, finalmente, determinar qué dos factores de la matriz de saturaciones total serán rotados.

La matriz de saturaciones se halla en la matriz denominada en el código fuente como "matriz [100][20]". Con el fin de evitar una falta de precisión en los cálculos matemáticos por pérdida de decimales, la matriz de datos ha sido declarada del tipo DOUBLE que nos permite guardar valores entre1.7E-308 y 1.7E+308. La utilización de datos de esta longitud (cada elemento de la matriz ocupa 8 bytes) nos limita el número de factores y variables que el programa será capaz de poner en memoria, ya que la matriz de datos más grande que el Turbo C es capaz de compilar es de 64000 bytes. En nuestro caso, la matriz más grande que nos será permitida no debe superar los 8000 elementos, combinando el número de factores y variables. En el código fuente se halla dimensionada para 20 factores y 100 variables, si bien resulta evidente que podría variarse para otros casos.

Mediante la función "matriz–datos ()", indicamos cual de los factores de la matriz serán el primero y el segundo en la rotación dos a dos. Ambos factores se copian sobre las variables "fact1" y "fact2" respectivamente y toda la información que se lista por pantalla hace referencia únicamente a estos dos.

Las funciones implicadas en la manipulación y tratamiento de la matriz de datos son las siguientes: "abrir ()", "datos–disco ()" y "matriz–datos ()".

Las funciones de control de la tarjeta gráfica, el teclado y el ratón tienen finalidades muy expecíficas, como pueden ser conectar la tarjeta gráfica, leer el estado del

TECLA DEL MONITOR	TECLA QUE LA CONTROLA	FUNCION
"<-" y "->"	"<-" y "->"	Aumentar o disminuir en un grado el ángulo de rotación
"Avc"	"Inicio"	permite una rotación rápida, situando los ejes mediante las teclas "<-" y "->" en la posición tecla "Ini".
"Pgdn" y "Pgup"	"Pgdn" y "Pgup"	Si existen más de 25 variables nos permiten pasar o retroceder las hojas de saturaciones que vemos en el monitor
"Find"	"Supr"	Conecta el ratón. Situándo el puntero del ratón sobre una variable, y pulsando u botón del ratón, nos indica su número de variable. Para abandonar esta opción basta con pulsar cualquier tecla
"Fac"	"Fin"	Permite variar los factores

Tabla 1 : Correspondencia entre las teclas del monitor y las teclas del teclado

ratón o determinar si el buffer de teclado se haya vacío. El conjunto de estas funciones se forma por "caja ()", "initialize ()", "boton–p ()", "boton ()", "te–bios ()", "rata ()", "leer–rata ()", "limpiar ()", "tapas ()" y "pizarra ()".

Finalmente, nos resta una última función cuya finalidad es la de determinar el número de variable que corresponde a cada punto gráfico a través de la interrupción 33h del BIOS que conecta la interface del ratón. Esta es la función "buscar ()".

Todo este conjunto de funciones se gestionan desde la función principal "main ()". Los pasos seguidos por esta función son los siguientes: en primer lugar determinar si los parámetros recibidos desde el DOS son correctos (nombre del fichero de lectura de saturaciones, nombre del fichero de salida de saturaciones rotadas y número de factores en la matriz de saturaciones); a continuación gestiona la lectura del fichero de entrada de datos; prepara la pantalla e inicializa las variables principales; y, finalmente, pasa el control al menú de teclado.

UTILIZACION DE RT.EXE

Al mismo tiempo que se arranca el programa RT desde el DOS, se le envían los datos referentes a los ficheros de entrada y salida de datos (ambos en código ASCII) y al número de factores de la matriz de saturaciones. Por ejemplo, para ejecutar un fichero de entrada denominado "matriz.txt" con tres factores, enviando las saturaciones rotadas al fichero de salida "rotación.txt" se deberá ejecutar la siguiente línea desde el DOS:

C:\RT matriz.txt rotacion.txt 3 <crt>

El control del menú se realiza desde el teclado (las teclas se denominan por la etiqueta de un teclado IBM), reflejándose las teclas pulsadas en las teclas representadas en el monitor. La correspondencia entre las teclas que aparecen representadas en pantalla y las teclas del teclado se expone en la tabla I.

Se pueden controlar tres teclas más que no aparecen representadas en el monitor:

F1 : vuelca la pantalla en la impresora. La impresión se realiza mediante la interrupción 5h, siendo éste un método funcional, pero deficiente de impresión. Lo más adecuado es la utilización de un capturador de pantalla que trabaje con un monitor VGA 16 colores (o escala de grises) y 640 x 480 pixels de definición.

F2: vuelca las saturaciones actuales, el ángulo de rotación y los factores que se están rotando al fichero de salida. Si se realiza más de una gravación, aparecerán todas en el fichero de salida ordenadas temporalmente.

ESC: permite abandonar el programa.

Nota: los lectores interesados en el programa pueden remitir, a los autores del presente artículo, un diskette que les será devuelto con el listado fuente y el programa compilado para ser usado.

REFERENCIAS

Bielig-Schulz, G y Schulz Ch. (1990) 3D *graphics in Pascal*. John Wiley & Sons. Essex.

Borland Co. (1987) Turbo C : *reference guide Borland* International. California

Carroll, J.B. (1978) *How shall we study individual differences in cognitive abilities? Methodological and theoretical perspectives*. Intelligence 2, 87-115

Comrey, A.L. (1985) *Manual de análisis factorial*. Cátedra. Madrid

Gervais, F. (1991) *Programación de las tarjetas gráficas* CGA, EGA, VGA. Editorial RA-MA. Madrid.

Gorsuch, R.L. (1974) *Factor analysis*. Saunders Philadelphia

Harman, H. (1980) *Análisis Factorial* Moderno. Saltés. Madrid.

Mata, A. (1989) *Turbo C, iniciación y programación avanzada* . Paraninfo. Madrid.

Norton, P. y Wilton,R. (1989) *Guía de programación para el IBM, Pc y PS/2*. Ediciones Anaya Multimedia. Madrid.

Thurstone, L.L. (1947): *Multiple factor analysis*. Chicago. Univ. Chicago press

Zimmerman, S y Zimmerman, BB (1990) *La biblia del turbo C* . Ediciones Anaya Multimedia. Madrid.

ANEXO : LISTADO DE RT.EXE

```
/*** fichero proyecto          ********************
*     rt.prj                        14-12-1991 *
*     proyecto:  RT.EXE
****************************************/


rt.c
rt–1.c
rt–2.c



/******** fichero DECLARACION************
 *        rt.h                          14-12-1991 *
 *        proyecto:  RT.EXE
****************************************/


#define  SUPR        83
#define  PI          3.1416
#define  ESC         27
#define  IZQUI       75
#define  DEREC       77
#define  PGUP        73
#define  PGDN        81
#define  HOME        71
#define  END         79
#define  F1          59
#define  F2          60
extern union REGS rge,rgs; extern double ma-
  triz[100][20];
extern struct TECLA { int coor–x, coor–y, situ–letra,
  tipo; char str[6]; };
struct TECLA tecla[8]={{0,0,0,0," "}, {250, 420, 12,
  0,"Avc"}, {330,420,15,0,"Re"}, {130,420,10, 0,
  "Find"}, {290,420,12,0,"Fac"},{370,420,15,0,"Av"},
  {170,420,17,1,27}, {210,420,17,1,26},};
void *blanco, *blanquito, *miniblanquito;
double ejes[][2]={ {0,0}, {0,0}, {0,0}, {0,0}},
datos–r[100][2], datos [100][2], angulo, arco, coseno,
  seno, tmp;
int camino[][2]={{0,2}, {3,1}}, patron[][2]={{ 0,-1},
  {1, 0}, { 0, 1}, {-1, 0}}, xc=210, yc=210, xpc=28,
  ypc=440, bolas[11], or, fi, vertices = 4, aristas = 2,
  aumentos = 200, no–hoja = 0, hoja = 0, max–dat =
  100, max–hoja = 25, max–factores;
FILE *fichero–entrada, *fi–salida;
char *fichero;
void dibujar–ejes(int); void ejes–peque(int );
void representar–datos(void);
void escribir–datos(void); void rehacer–datos(void);
void tapas(void); void variables(void);
void rotar–ejes(void); void rotar–datos(void);
void caja(void); void abrir(FILE *str, int *max);
void datos–disco(int fact1, int fact2 );
void matriz–datos(int fact1, int fact2);
void buscar(int ,int ,int * ); void trigonometria(void);
void rotacion (void); void dibujar(void);
extern void initialize(void);
extern void boton–p (struct TECLA *t);
extern void boton (struct TECLA *t);
extern int te–bios(void); extern void rata(int on);
extern int leer–rata(int *coor–x, int *coor–y);
extern void limpiar(void); extern void pizarra(void);
```

```
/*****    fichero  PRINCIPAL     *********
*       rt.exe                14-12-1991        *
*       proyecto:RT.EXE                         *
****************************************/


#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <dos.h>
#include <stdlib.h>
#include <io.h>
#include <rt.h>
main(int argc, char *argv[])
{
register i;
char ch   = 1, chh  = 1; int fact1 = 1, fact2 = 2;
  if (argc < 4)  {  printf ("\n Error : el número de pará-
    metros ha de ser 3.\n"); printf ("\n Ejecución de
    RT.EXE:\n\n\t RT <fichero–entrada.dat> <fichero–sa-
    lida> <numero de factores>\n\n\a");  exit(1);  };
  max–factores = atoi(argv[3]);
  if (max–factores < 2) {  printf ("\n Error : el número
    mínimo de factores en el fichero es de dos\n");
    printf ("\n Ejecución de RT.EXE:\n\n\t RT <fi-
    chero–entrada.dat> <fichero–salida> <numero de fac-
    tores>\ n\n\a");  exit(1); };
  if ((fichero–entrada = fopen(argv[1],"r")) == 0 ) { printf
    ("\n  Error de lectura en el fichero de datos
    '%s'\n",argv[1]);   printf ("\n  Ejecución de
    RT.EXE:\n\n\t RT <fichero–entrada.dat> <fichero –
    salida> <numero de factores>\n\n\a");  exit(1); };
  if ( ( fi–salida = fopen (argv[2],"w") ) == 0 ) {  printf
    ("\n  Error de lectura en el fichero de salida
    '%s'\n",argv[2]);  printf  ("\n  Ejecución de
    RT.EXE:\n\n\t RT <fichero–entrada.dat> <fichero–
    salida> <numero de factores>\n\n\a");  exit(1); };
  fichero = argv[2];  fclose(fi–salida); abrir(fichero–en-
    trada, &max–dat); fclose(fichero–entrada);
  matriz–datos (fact1,fact2);  initialize ();  tapas ();
  for (i=0; i<vertices; i++) {
     ejes[i][1] = patron[i][1] * aumentos; };
     ejes[i][0] = patron[i][0] * aumentos;
  caja ();  representar–datos();  dibujar–ejes(1);
  ejes–peque(1);  variables();  angulo = 0;  rotacion();
  while (ch != ESC) {
     limpiar(); ch = getch(); if (!ch) ch = getch();
     switch(ch) {
     int valor, valor2, valor3; char str  [5], buf [30];
     case F1: setfillstyle(SOLID–FILL,16); pizarra();
       dibujar(); int86(0x05,&rge,&rgs);
         setfillstyle(SOLID–FILL,2);pizarra();
           dibujar(); break;
     case F2: datos–disco(fact1 , fact2);break;
     case SUPR: boton–p(&tecla[4]);
       setcolor(WHITE);  rata(1); valor = 0;
       while(!te–bios()) {
            valor = leer–rata(&valor2, &valor3);
         if (valor) { buscar(valor2,valor3,bolas);
           putimage(425,320,blanco,0);
           putimage(510,320,blanco,0);
           valor2 = 425;
           for (i=0; ((bolas[i] != 0)
                &&  (i < 11)); i++) {
             valor2 += 20;
             sprintf(buf," %i", bolas[i]);
             outtextxy (valor2,320,buf);bo-las [i]=0;  };
         };
       };
```

```
        rata(0);boton(&tecla[4]);
      break;
    case END : boton–p(&tecla[5]);
      setcolor(WHITE); valor = 0 ;
        while ((valor < 1) || (valor > max–factores))
{
      putimage(495,470,blanquito,0);
      str[0] = getch();
      while ( (str[0] < 49) || (str[0] > 57) ) str[0] =
        getch();
      str[1] = getch();
      while (( (str[1] < 48) || (str[1] > 57))  &&   (str[1]
        != 0xD) ) str[1] = getch();
      valor = atoi(str);
    }; sprintf(buf,"        %i", valor );
    outtextxy(440,470,buf); valor2 = 0 ;
    while ((valor2<1) || (valor2 > max_factores)) {
      str[0] = getch();
      while ( (str[0] < 49) || (str[0] > 57) )  str[0] =
        getch();
      str[1] = getch();
      while (( (str[1] < 48)   || (str[1] > 57)) && (str[1]
        != 0xD) )
      valor2 = atoi(str);
    } sprintf(buf,"              %i", valor2 );
    outtextxy(440,470,buf);
    setfillstyle(SOLID–FILL,2);pizarra();
    fact1 = valor; fact2 = valor2;
    matriz–datos( fact1, fact2); dibujar–ejes(2);
    ejes–peque(0); angulo = 0; rotacion();
    boton(&tecla[5]);
    break;
  case HOME : boton–p(&tecla[2]); dibujar–ejes(2);
    while (chh != HOME) {
      chh = getch(); if (!chh) chh = getch();
      switch(chh) {
          case DEREC:boton–p(&tecla[10]);
            boton(&tecla[10]); angulo +=1;
            break;
          case IZQUI : boton–p(&tecla[9]);
            boton(&tecla[9]);  angulo -=1;
            if (angulo < 0 ) angulo = 359;
      };
            if (angulo > 360) angulo = 1; trigonome-
            tria();
      ejes–peque(0); rotar–ejes();ejes–peque(1);
      }; boton(&tecla[2]); chh = 1;
      if (angulo < 0 ) angulo = 359; dibujar–ejes(2);
      rotacion(); break;
  case DEREC: boton–p (&tecla [10]); boton (&tecla
    [10]);
      angulo +=1; if (angulo > 360 ) angulo = 1;
          if (angulo < 0 ) angulo = 359;
      rotacion(); break;
  case IZQUI : boton–p(&tecla[9]); boton(&tecla[9]);
      angulo -=1; if (angulo < 0 ) angulo = 359;
      rotacion(); break;
  case PGUP  : if (max–dat > 25) {
      boton–p(&tecla[3]); boton(&tecla[3]); };
      if (hoja) { hoja -= 25; no–hoja=0;
          variables(); escribir–datos();};
      break;
  case PGDN  : if (max–dat > 25) {
      boton–p(&tecla[6]); boton(&tecla[6]); };
      if (!no–hoja)
      hoja += 25;
      variables(); escribir–datos();
  } ;
};
dibujar–ejes(2); ejes–peque(0); getch();
closegraph();
}
void representar–datos(void)
register i;
float x,y;
  for (i=0; i<max–dat; i++) {
      x = xc + (datos[i][0] * aumentos);
      y = yc + (datos[i][1] * aumentos * (-1));
      setfillstyle(SOLID–FILL,4); setcolor(4); circle
        (x,y,5);
  floodfill(x,y,4);setfillstyle(SOLID–FILL,3); setcolor(3);
  circle(x,y,4); floodfill(x,y,3);
      setcolor(5); circle((x-1),(y-1),2); setcolor(WHITE);
  circle((x-1),(y-1),1);
      putpixel((x-1),(y-1),WHITE);
  };
}
void rehacer–datos(void)
{
register i;
float x,y;
  for (i=0; i<max–dat; i++) {
  if (((datos–r[i][0] < 0.1) && (datos–r[i][0] > -0.1)) ||
       ((datos–r[i][1] < 0.1) && (datos–r[i][1] > -0.1))) {
      x = xc + (datos[i][0] * aumentos);
      y = yc + (datos[i][1] * aumentos * (-1));
        setfillstyle(SOLID–FILL,4); setcolor (4); circle
          (x,y,5);
      floodfill(x,y,4);
      setfillstyle(SOLID–FILL,3); setcolor(3); circle
        (x,y,4);
      floodfill(x,y,3); setcolor(5); circle((x-1),(y-1),2);
      setcolor(WHITE); circle((x-1),(y-1),1);
      putpixel((x-1),(y-1),WHITE);
  };
};
setcolor(WHITE);
}
void escribir–datos(void) {
char buf[30], tmp[30];
register i;
int coor=20;
  outtextxy(510,10,"Fact I   Fact II"); coor=20;
  for (i=0; (((i+hoja)<max–dat ) && (i < max–hoja)) ;
    i++) coor += 10;
        if ((fabs(datos–r[(i+hoja)][0])) >
        (fabs(datos–r[(i+hoja)][1])) ) { setcolor(6); }
          else
        setcolor(1); sprintf (buf, "%1.3f", datos–r
          [(i+hoja)] [0]);
          putimage(510,coor,blanco,0);
          if (buf[0] == '-') { outtextxy(510,coor,buf); }
            else {
{
          sprintf(tmp," %s",buf); outtextxy (510,coor,
            tmp); };
           sprintf(buf,"%1.3f",datos–r[i+hoja][1]);
        if (buf[0] == '-') { outtextxy(580,coor,buf); }
          else {sprintf(tmp," %s",buf); outtextxy
          (580,coor,tmp); };
};
if ((i + hoja) == max–dat) {
        for(; (i <= max–hoja) ;i++) { coor += 10;
        putimage(510,coor,blanco,0); };
};
setcolor(WHITE);
outtextxy(440,345,"Matriz de TRANSFORMACION");
outtextxy(440,365,"      Fact I Fact II");
sprintf (buf," Fact I  %1.3f  %1.3f ", coseno,seno);
putimage (495, 385, blanquito,0); outtextxy (440,385,
  buf);
if ( seno < 0) {
```

```
      sprintf(buf,"Fact II +%1.3f  %1.3f ",(seno * -1),co-
        seno);}
else
        sprintf(buf,"Fact II -%1.3f  %1.3f ",seno,coseno);
        putimage(495,405,blanquito,0);      outtextxy
          (440,405,buf);
}
void rotar–datos(void)
{
register i;
  for(i=0; i < max–dat; i++) {
      datos–r[i][0] = datos[i][0] * coseno - datos[i][1] *
        seno ;
      datos–r[i][1] = datos[i][0] * seno  + datos[i][1] * co-
        seno;
   };
}
void tapas(void)
{
int size;
  size=imagesize(510,20,639,30);    blanco=malloc
    (size);
  size=imagesize(490,428,625,438); blanquito=malloc
    (size);
  size=imagesize(490,428,550,438); miniblanquito=ma-
    lloc (size);
  getimage(510,20,639,30,     blanco);
  getimage(490,428,625,438,    blanquito);
  getimage(490,428,550,438,miniblanquito);
};
void variables(void)
{
char buf[30]; register i; int coor=20;
   for (i=1;((( i + hoja)<=max–dat) && (i <= max–hoja))
    ; i++) {
     coor += 10; putimage(450,coor,miniblanquito,0);
     sprintf(buf ,"Var %i",(i + hoja)); outtextxy(450,
       coor,buf); };
  if ((i + hoja) > max–dat) {
       no–hoja = 1;
       for(; (i <= max–hoja) ;i++)
           coor += 10; putimage(450,coor,miniblan-
             quito,0); };
   };
}
void ejes–peque(int col)
{
register i;
   setcolor(2);
   for (i=0; i< aristas; i++) {
        if (col) if (i==0) { setcolor(1); } else setcolor(6);
       or = camino[i][0]; fi = camino[i][1];
       line((xpc + (ejes[or][0] / 15)),(ypc + (ejes[or][1] /
         15)),
         (xpc + (ejes[fi][0] / 15)),(ypc + (ejes[fi][1] /
           15)));
   };
   circle( (xpc + ((ejes[0][0] / 25) + (ejes[1][0] / 25))),
       (ypc + ((ejes[0][1] / 25) + (ejes[1][1] / 25))) ,1);
   setcolor(WHITE);
}
void abrir(FILE *str, int *max)
{

register i,j; int contador=0; char *tmp;
  for(i=0; (fscanf(str,"%s", tmp ) != EOF)  ; i++) {
     matriz[i][0] = atof(tmp);
     for ( j=1; j < max–factores ; j++) {
       fscanf(str,"%s", tmp ); matriz[i][j] = atof(tmp);
    }; contador++;
       }; *max = contador;
}
void matriz–datos(int fact1, int fact2)
{
register i;
  fact1—; fact2—;
     for (i=0; i < max–dat; i++)     {
       datos[i][0] = matriz[i][fact1];
     datos[i][1] = matriz[i][fact2];
  };
}
void datos–disco(int fact1, int fact2 )
{
char buf[30], tmp[30]; register i;
   fi–salida = fopen (fichero,"a");
   sprintf(buf,"fact %i       fact %i",fact1,fact2);
   fprintf(fi–salida,"\n%s \n\n ",buf);
  for (i=0; i < max–dat; i++) {
       sprintf(buf,"%1.5f",datos–r[i][0]);
     if (buf[0] == '-') { fprintf(fi–salida,"%12s ",buf); }
     else {
       sprintf(tmp," %s",buf);fprintf(fi–salida,"%12s
         ",tmp);
       };
       sprintf(buf,"%1.5f",datos–r[i][1]);
       if (buf[0] == '-') { fprintf(fi–salida,"%12s \n ",buf);
}
  else {
       sprintf(tmp," %s",buf); fprintf(fi–salida,"%12s \n
         ",tmp);
       };
  };
  sprintf(buf," ngulo  : %1.4f",angulo);
  fprintf(fi–salida," %s\n",buf);
  fprintf(fi–salida,"——— \n\n ");
  fclose(fi–salida);
}
void buscar(int x, int y, int *valores)
{
register i;int x1, y1, contador = 0;
  for (i=0; i < max–dat; i++)  {
     x1 = xc + (datos[i][0] * aumentos );
     y1 = yc + (datos[i][1] * (aumentos * -1));
     if ( (x > (x1 - 3)) && (x < (x1 + 3)) &&
       (y > (y1 - 3)) && (y < (y1 + 3))  )
       *(valores + contador++) = (i + 1);
  };
}
void dibujar–ejes(int col)
{
register i;
  setcolor(col);
  for (i=0; i< aristas; i++) {
     or = camino[i][0]; fi = camino[i][1];
     line((xc + ejes[or][0]), (yc + ejes[or][1]),
       (xc + ejes[fi][0]), (yc + ejes[fi][1]));
  }; rehacer_datos();
}
void rotar_ejes(void)
{
register i;
  for (i=0; i<vertices; i++) {
     tmp      = ((patron[i][0] * coseno)
       (patron[i][1] * seno  ) ) * aumentos;
     ejes[i][1] = ((patron[i][0] * seno  ) +
       (patron[i][1] * coseno) ) * aumentos;
     ejes[i][0] = tmp;
  };
}
void caja(void)
{
```

```
boton(&tecla [6]); boton(&tecla[7]); boton (&tecla [1]);
boton(&tecla[4]); boton(&tecla[3]);
if (max–dat > 25) { boton(&tecla[2]); boton (&tecla [5]);
  };
 settextstyle(DEFAULT–FONT,HORIZ–DIR,1);
 setfillstyle(SOLID–FILL,2); bar(5,5,415,415);
 setlinestyle(SOLID–LINE,0,THICK–WIDTH);
line (5,5,415,5); line (415,5,415,415); line (415,
  415,5,415); line (5,415,5,5); setlinestyle (SOLID–
  LINE, 0,NORM– WIDTH);
setcolor(WHITE);circle(xpc,ypc,20); floodfill (xpc,
  ypc,WHITE);
setcolor(6); outtextxy(60,430, "Fact I");
setcolor(1); outtextxy(60,445, "Fact II"); setcolor
  (WHITE);
 outtextxy(440,300,"Localización de VARIABLES");
 outtextxy(440,430,"Factores en rotación");
 outtextxy(440,450,"      Fact I Fact II");
 outtextxy(440,470,"        1      2");
}
void trigonometria(void)
{ arco  = angulo * PI / 180; seno  = sin (arco);
 coseno = cos (arco);}
void rotacion(void)
{ trigonometria(); rotar–ejes(); rotar–datos();
 dibujar–ejes(1); ejes–peque(1); escribir–datos();}
void dibujar(void)
{ representar–datos(); dibujar–ejes(1); ejes–peque(1);
 escribir–datos();}
```

```
/***       fichero VARIABLES       **********
 *    rt–1.c        14-12-1991    .          *
 *      proyecto: RT.EXE                     *
*****************************************

double matriz[100][20];
```

```
/*****   fichero   GRAFICOS       *********
 *      rt–2.c                14-12-1991      *
 *      proyecto: RT.EXE                      *
*****************************************/


#include <graphics.h>
#include <dos.h>
union REGS rge,rgs;
struct TECLA { int coor–x, coor–y, situ–letra, tipo;
        char str[6]; };
void initialize(void); void boton–p (struct TECLA *t);
void boton (struct TECLA *t); int te–bios(void);
void rata(int on); int leer–rata(int *coor–x, int *coor–y);
void limpiar(void);void pizarra(void);
void boton (struct TECLA *t)
{
  if (t->tipo == 1) {settextstyle (DEFAULT –FONT,
    HORIZ–DIR,1); }
  else settextstyle(SMALL–FONT,HORIZ–DIR,4);
  setfillstyle(SOLID–FILL,5);
  bar ( t->coor–x, t->coor–y, t->coor–x + 40, t->coor–y +
    40);
  setlinestyle(SOLID–LINE,0,NORM–WIDTH);
  setcolor(BLACK); rectangle (t->coor–x, t->coor–y,
          t->coor–x + 40, t->coor–y + 40);
setcolor(WHITE);setlinestyle(SOLID–LINE,0,THICK–
  WIDTH);
  line(t->coor–x + 3, t->coor–y + 3, t->coor–x + 38 ,
       t->coor–y + 3);
  line(t->coor–x + 3, t->coor–y + 3,t->coor–x + 3,
       t->coor–y + 38);
  setcolor(4); line(t->coor–x + 3,  t->coor–y + 38,
       t->coor–x + 39, t->coor–y + 38 );
  line(t->coor–x + 38 ,t->coor–y + 3,t->coor–x + 38 ,
       t->coor–y + 38);
  setcolor(BLACK); outtextxy(t->coor–x + t->situ–letra ,
       t->coor–y + 15, t->str);
  if (t->tipo == 0)  settextstyle(SMALL–FONT, HORIZ–
    DIR,4);
  setcolor(WHITE);  setlinestyle(SOLID–LINE,0,
    NORM– WIDTH);
}
void boton–p (struct TECLA *t)
 {
  if (t->tipo == 1) {
       settextstyle(DEFAULT–FONT,HORIZ–DIR,1);  }
        else
       settextstyle(SMALL–FONT,HORIZ–DIR,4);
  setcolor(5); line(t->coor–x + 3,  t->coor–y  + 38 ,
          t->coor–x + 38, t->coor–y  + 38 );
line(t->coor–x  + 38 ,  t->coor–y + 2, t->coor–x  + 38 ,
       t->coor–y  + 38);
line(t->coor–x + 3, t->coor–y + 3, t->coor–x  + 38 ,
       t->coor–y + 3);
line(t->coor–x + 3, t->coor–y + 3, t->coor–x + 3,
       t->coor–y  + 38);
outtextxy(t->coor–x + t->situ–letra ,t->coor–y + 15, t-
  >str);
setcolor(4); setlinestyle (SOLID– LINE, 0, THICK–
  WIDTH);
line(t->coor–x + 3, t->coor–y + 2, t->coor–x  + 38 ,
       t->coor–y + 2);
line(t->coor–x  + 2, t->coor–y + 1, t->coor–x  + 2,
       t->coor–y  + 38);
setlinestyle(SOLID–LINE,0,NORM–WIDTH);
line(t->coor–x  + 2, t->coor–y  + 38, t->coor–x  + 39,
       t->coor–y  + 38);
line(t->coor–x  + 38 ,t->coor–y + 3, t->coor–x  + 38,
       t->coor–y  + 38);
setcolor(BLACK); outtextxy(t->coor–x + (t->situ–letra
  +1 ),
       t->coor–y + 16, t->str);
sound(400); delay(20); nosound();delay(50);
settextstyle(DEFAULT–FONT, HORIZ–DIR,1);
}
void initialize(void)
{
int GraphDriver, GraphMode;
  GraphDriver = VGA;  GraphMode = VGAHI;
  initgraph( &GraphDriver, &GraphMode, "" );
  setrgbpalette(1,0,60,0);   setrgbpalette(2,0,14,1);
  etrgbpalette(3,35,35,35); setrgbpalette(4,25,25,25);
  setrgbpalette(5,45,45,45); setrgbpalette(6,35,36,10);
  settextstyle(DEFAULT–FONT,HORIZ–DIR,1);
  setpalette(8,23);
}
void rata(int on)
{ rge.x.ax= on; int86(0x33,&rge,&rgs);};
int te–bios(void)
{ rge.h.ah=0x0B ; intdos(&rge,&rgs);
  if (rgs.h.al != 0 ) {return 1; } else return 0;}
int leer–rata(int *coor–x, int *coor–y)
```

```
{
int tmp;
 rge.x.ax= 3;  int86(0x33,&rge,&rgs); tmp = rgs.x.bx;
 *coor–x = rgs.x.cx; *coor–y = rgs.x.dx;
 if (tmp) { return 1; } else  return 0;
}
void limpiar(void)
{ rge.h.ah=0x0C ; int86(0x21,&rge,&rgs);}
```

```
void pizarra (void)
{
 bar(5,5,415,415); setcolor(WHITE);
 setlinestyle (SOLID–LINE,0,THICK–WIDTH);
 line (5,5,415,5);     line (415,5,415,415);
 line (415,415,5,415); line (5,415,5,5);
 }
```