Grafo 2 con ANCR - 5

kc/ki = 1

	Costo total.	Desbalance	Comunicac.
1	4.375 e + 2	1.125 e + 2	3.250 e + 2
2	3.450 e + 2	1.950 e + 2	1.500 e + 2
3	3.450 e + 2	1.950 e + 2	1.500 e + 2
4	3.000 e + 2	2.250 e + 2	7.500 e + 1
5	4.125 e + 2	1.125 e + 2	3.000 e + 2
6	3.775 e + 2	1.775 e + 2	2.000 e + 2
7	3.875 e + 2	1.875 e + 2	2.000 e + 2
8	3.875 e + 2	1.875 e + 2	2.000 e + 2
9	4.375 e + 2	1.125 e + 2	3.250 e + 2
10	3.000 e + 2	2.250 e + 2	7.500 e + 1

Kc/ki = 1/5

	Costo total.	Desbalance	Comunicac.
1	2.458 e + 2	4.580 e + 1	2.000 e + 2
2	2.458 e + 2	4.580 e + 1	2.000 e + 2
3	2.375 e + 2	4.580 e + 1	1.910 e + 2
4	2.625 e + 2	6.250 e + 1	2.000 e + 2
5	2.375 e + 2	4.580 e + 1	1.910 e + 2
6	2.625 e + 2	6.250 e + 1	2.000 e + 2
7	2.375 e + 2	4.580 e + 1	1.910 e + 2
8	2.458 e + 2	4.580 e + 1	2.000 e + 2
9	2.458 e + 2	4.580 e + 1	2.000 e + 2
10	2.625 e + 2	4.580 e + 1	2.160 e + 2

Tabla 6.2.

dificar el cociente kc/ki en relación inversa al ANCR. La propuesta es, entonces, establecer los siguientes coeficientes:

kc: 1 (comunicaciones)

ki: Costo medio arcos / Costo medio nodos (desbalance)

Los resultados obtenidos con estos nuevos coeficientes se detallan en la tabla 6.3. Vemos aqui que las varianzas con la nueva relación k_c/k_i se reducen notablemente, a valores similares a los del caso ANCR = 1.

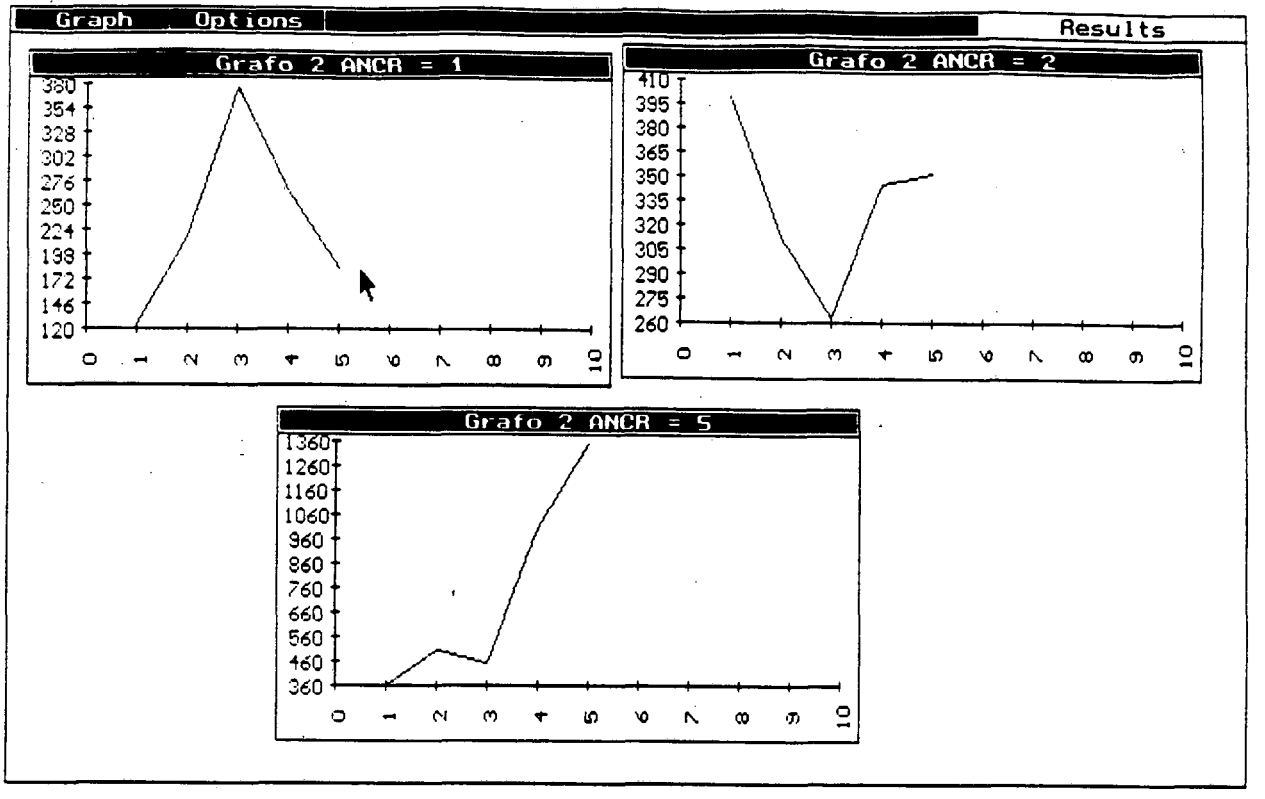

*Figura 6.7. Varianza vs. 5 * kc/ki*

6.4.Funciones de costo y tiempo de ejecución.

La minimización de la ecuación de costo no supone necesariamente una optimización del tiempo de ejecución del grafo. Ello sólo es válido para el caso del ANCR = 1. Para el grafo 2 con ANCR = 5, por ejemplo, las soluciones "buenas" (de baja varianza) tienen un tiempo medio de ejecución de 500 unidades. Obsérvese que si todas las tareas se hubiesen asignado a un sólo procesador este tiempo hubiese sido de 300 unidades!. Existen dos razones fundamentales que explican este comportamiento. La primera es obvia: la ecuación de costo no evalúa el tiempo de ejecución, sino los desbalances en la carga y el costo de comunicaciones. En una asignación estática como la que nos ocupa, el tiempo sólo podría ser estimado (a lo sumo, simulado), no calculado.

La segunda razón es que el algoritmo no considera los retardos debido al procedimiento "store and forward" de las arquitecturas tipo link, esto es, el comportamiento FIFO de

los canales de comunicación. Al intentar minimizar el costo de comunicaciones, el algoritmo tiende a crear grupos de tareas grandes, que se comunican entre sí a través de unos pocos links, que consecuentemente sufren un intenso tráfico. Nuevamente, este tráfico no puede ser calculado a priori, ya que esto supondría conocer los tiempos en los que se producen las comunicaciones.

Simulation Results for kc: – Arc mean weight / node mean weight ; ki: – 1

Graph	ANCR	Worst Cost	Best Cost	Mean Cost	Variance
Graph 1	1	167	142	148	2.18
	2	192	171	176	2.00
	5	262	237	243	2.37
Graph 2	1	152	135	141	1.84
	2	221	198	212	2.62
	5	258	225	236	3.63
Graph 3	1	215	196	204	1.68
	2	258	230	242	2.42
	5	356	327	343	3.18

Tabla 6.3. Resultados para los nuevos coeficientes

6.5.Resumen.

La técnica de Simulated annealing es un procedimiento heurístico de propósito general para la resolución de problemas de optimización combinatoria. En este capítulo se detallan las decisiones tomadas para utilizar el procedimiento para la asignación de tareas en un sistema multiprocesador. La herramienta de simulación descrita en capítulos anteriores es empleada aquí para sintonizar los parámetros del algoritmo en función de las características del grafo de programa a asignar. En particular se ilustra el ajuste de los coeficientes de la ecuación que evalúa el costo del mapeo obtenido.

Capítulo 7.

Conclusiones y líneas abiertas.

Este trabajo analiza la utilización de herramientas de modelado y simulación para el estudio y diseño de sistemas paralelos de cómputo digital. En tal sentido se propone una estrategia de modelado original, y se describe la especificación e implementación del correspondiente simulador.

El formalismo propuesto modela cuatro aspectos fundamentales de un ordenador paralelo: programas, estructuras de interconexión, estrategias de asignación de tareas, y políticas de comunicación. Con respecto a los programas paralelos, la estrategia utilizada, que denominamos WBG (Weighted Behavioral Graph), permite modelar tanto las características estáticas del grafo de programa (esto es, su patrón de comunicaciones), como sus características dinámicas (activación de copias por demanda, estructuras condicionales, lazos y procedimientos recursivos). Este esquema de representación posee tres características fundamentales:

-Permite modelar situaciones de sincronización y concurrencia, al presentar "reglas de disparo", generalizadas a partir de las redes de Petri. Los nodos del grafo se habilitan cuando los "tokens" en sus arcos de entrada verifican la condición de disparo o "política de entrada" del mismo. De la misma forma, la distribución de tokens a la salida del nodo está definida a partir de la especificación de una "política de salida".

-Posibilita la generación de copias o instancias de nodos o subgrafos, en forma dinámica en tiempo de simulación, mediante la utilización de colores o tags (en forma similar a las máqui-

nas de flujo de datos de tipo dinámico), lo que permite un aumento efectivo del posible paralelismo del programa en tiempo de ejecución.

-Permite especificar los volúmenes de cómputo y comunicaciones del programa, controlando la granularidad del modelo y posibilitando la evaluación de los tiempos de ejecución.

En relación al modelado de estructuras de interconexión, la estrategia es similar. La arquitectura en estudio se modela a partir de un grafo, donde los nodos pertenecen a "clases" que califican su comportamiento. No se hace ninguna alusión a detalles tecnológicos ni de implementación, lo cual, si bien quita precisión a los resultados, permite comparar esquemas muy diferentes sobre una base común.

Las políticas de asignación de tareas y ruteo de mensajes son definidas por el usuario mediante rutinas de alto nivel. Esta decisión ha sido tomada teniendo en cuenta la amplia variedad de soluciones posibles, lo que imposibilita la adopción de una única estrategia de modelado.

Las ideas propuestas se han llevado a la práctica en el diseño e implementación de un programa simulador interactivo, cuyos listados se adjuntan. El mismo posee un diseño modular, organizado como un "pipeline" circular. Cada módulo es altamente independiente del resto, y se comunica con los demás mediante una interfaz estrictamente definida. La máquina de simulación ha sido "rodeada" por una interfaz gráfica interactiva, de forma tal de reducir tanto el tiempo de familiarización del usuario con el sistema, como el período de desarrollo y depuración del modelo bajo estudio.

Varias son las líneas que este proyecto deja abiertas. La primera de ellas apunta a convertir el sistema en un ambiente para el desarrollo de aplicaciones paralelas. En tal sentido creo que sería interesante agregar al sistema la posibilidad de definir el programa de cada nodo y generar código máquina para alguna arquitectura específica. Esto supone la adopción (o definición) de un lenguaje de programación tal que permita especificar explícitamente el posible paralelismo del programa, o bien el diseño de un compilador optimizador que lo extraiga en forma automática.

El ambiente de simulación desarrollado es aún una herramienta joven. Creo que la experiencia en su uso hará surgir la necesidad de la creación de nuevas clases de nodos, nuevas políticas de entrada y salida, y fundamentalmente, de una biblioteca de modelos amplia. En tal sentido sería interesante la utilización del sistema como herramienta didáctica en cursos de grado y postgrado. Ello tal vez importe el diseño de una versión reducida del sistema para ordenadores con menores prestaciones que el utilizado en esta tesis.

Una de las aplicaciones mas interesantes que hoy vemos del sistema es el estudio de políticas de asignación dinámicas en sistemas multiprocesadores. Dadas las características del WBG, es lógico pensar que un scheduler dinámico presentará mejores prestaciones que uno estático como el analizado en el capítulo 6. En esencia, un algoritmo de asignación dinámica mide (o estima) la carga de cada procesador durante tiempo de ejecución, y adjudica las tareas activadas al procesador menos ocupado. Aquí aparecen cuestiones de difícil solución tales como el balance de carga en tiempo real, migración de código entre procesadores, "duplicación" de tareas a los efectos de reducir el tráfico o debido a variaciones relativas en los volúmenes de cómputo, "fusión" de nodos, y reducción del "overhead" producido por el scheduler. El desarrollo de heurísticas eficientes en este sentido involucra la intensiva utilización de herramientas interactivas de simulación.

Finalmente, creo que sería altamente factible la implementación paralela del simulador en sí mismo, tal como se propone en el capítulo cuatro. Esto nos permitiría una mayor flexibilidad al reducir drásticamente los tiempos de cómputo, posibilitando en muchos casos una emulación en tiempo real de la arquitectura bajo estudio. La relativa sencillez y modularidad del simulador hace esta migración a hardware razonablemente directa.

Bibliografía.

[1]: R.W. Hockney, C.R.Jesshope."*Parallel Computers 2- Architecture, Programming and algorithms.*". Adam Hilger Ed. 1988.

[2]: S.H. Unger. "*A computer oriented towards spatial problems*". Proceedings of the IRE, October 1958, pp. 1744-1750.

[3]: J.H. Holland. "*A universal computer capable of executing an arbitrary number of sub-programs simultaneously*". 1959 EJCC, pp. 108-113.

[4]: D.L. Slotnick. "*The Solomon Computer*". Proceedings of the Fall Joint Computer Conference. Vol. 22, 1962, pp. 97-107.

[5]: G.H. Barnes, R.M. Brown, M. Kato, D.S. Kuck, D.L. Slotnick, R. Stokes. "*The ILLIAC IV Computer*". IEEE Trans. on Computers. Vol C-17 n.8, pp 746-757, Aug. 1968.

[6]: BW. Shooman. "*Parallel Computing with Vertical Data*". 1960 EJCC. pp 111-115, Dec. 1960.

[7]: K.E. Batcher. "*STARAN Parallel Processor System Hardware*". *1974 NCC, pp 405-410.*

[8]: A.H. Veen. "*Dataflow machine architecture*". Computing Surveys, vol.18, n.4, December 1986. pp.365-396.

[9]: P.C. Treleaven, D.R. Brownbridge, R.P. Hopkins. "*Data-driven and Demand-driven computer architecture*". Computing surveys, Vol.14, n.1, March 1982. pp.93-143.

[10]: D.P. Agrawal, V.K. Janakiram, G.C. Pathak. "*Evaluating the performance of multicomputer configurations*". IEEE COMPUTER May 1986. pp.23-37.

[11]: L. Snyder. "*Type architectures, shared memory and the corollary of modest potential*". Ann. Rev. Comput. Sci. 1986. pp.289-317.

[12]: M.J.Quinn."*Designing efficient algorithms for parallel computers*". McGraw Hill Inc. 1987.

[13]: W.W. Chu, L.J. Holloway, M.T. Lan, K. Efe. "*Task allocation in distributed data processing*". IEEE COMPUTER November 1980. pp.57-69.

[14]: K.B. Irani, K.W. Chen. "*Minimization of interprocessor communication for parallel computation*". IEEE Transactions on Computers, vol. C-31, n.11, November 1982. pp.1067-1075.

[15]: A.M. Van Tiborg, L.D. Wittie. "*Wave Scheduling-Decentralized sheduling of task forces in multicomputers*". IEEE Transactions on computers, vol. C-33, n.9, September 1984. pp.835-844.

[16]: S.H. Bokhari. "*On the mapping problem*". IEEE Transactions on Computers, vol. C-30, n.3, March 1981. pp.207-214.

[17]: C.C. Shen, W.H. Tsai. "*A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion*". IEEE Transactions on Computers, vol. C-34, n.3, March 1985. pp.197-203.

[18]: K. Efe. "*Heuristic models of task assignment scheduling in distributed systems*". IEEE COMPUTER June 1982. pp.50-56.

[19]: J. Backus. "*Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs*". Communications of the ACM, vol. 21, n.8, August 1978. pp.613-641.

[**20**]: D.A. Padua, D.J. Kuck, D.H. Lawrie. "*High-speed multiprocessors and compilation techniques*". IEEE Transactions on Computers, Vol. C-29, n.9, September 1980. pp.763-776.

[**21**]: W.A. Ackerman. "*Data flow languages*". IEEE COMPUTER February 1982. pp.15-25.

[**22**]: C.A.R. Hoare. "*Communicating Sequential Processes*". Communications of the ACM, vol. 21, n.8, August 1978. pp.666-677.

[**23**]: C.A.R. Hoare. "*Communicating Sequential Processes*". Prentice-Hall International. Series in Computer Science. 1985. pp.666-677.

[**24**]: P.C. Treleaven. "*Advanced Computer Architecture*" . Proceedings of the 1988 Cern School of Computing. pp. 96-104.

[**25**]: K.M. Nichols, J.T. Edmark. "*Evaluating multicomputers Systems with PARET*".IEEE COMPUTER, May 1988. pp. 39-48.

[**26**]: F. Berman. "*Experience with an automatic solution to the mapping problem*". The characteristics of parallel algorithms. Edited by L.H. Jamieson, D.B. Gannon, R.J.Douglas. Scientific Computation Series, MIT Press, 1987.

[**27**]: J.M. Butler, A.Y. Oruc. "*A facility for simulating multiprocessors*". IEEE MICRO October 1986. pp.32-44.

[**28**]: B. Melamed, R.J.T. Morris. "*Visual simulation: The performance analysis workstation*". IEEE COMPUTER August 1985. pp.87-94.

[**29**]: J.C. Delgado. "*Designing computer architectures with SADAN*". Microprocessing and microprogramming 22 (1988). pp.205-216.

[**30**]: R.J. Bagrodia, K.M. Chandy, J. Misra. "*A message-based approach to discrete-event simulation*". IEEE Transactions on Software Engineering, Vol. SE-13, n.6, June 1987. pp.654-665.

[31]: L.J. Hafer, A.C. Parker. "*Automated synthesis of digital hardware*". IEEE Transactions on Computers, vol. C-31, n.2, February 1985. pp.93-109.

[32]: K. Kuckcisnsky, Z. Peng. "*Parallelism extraction from Sequential programs for VLSI applications*". Microprocessing and microprogramming 23 (1988). pp.87-92.

[33]: M.F. Letheren. "*Software tools and methodologies for the design of Digital Electronic Systems*". Proceedings of the 1988 Cern School of Computing. pp. 182-233.

[34]: J. Rabay. "*Silicon compilation and design synthesis for digital systems*". Proceedings of the 1988 Cern School of Computing. pp. 234-259.

[35]: B.P. Ziegler. "*Multifacetted Modelling and Discrete Event Simulation*". Academic Press. 1984.

[36]: J.Misra. "*Distributed discrete-event simulation*". Computing Surveys, vol.18, n.1, March 1986. pp.39-65.

[37]: J.l. Peterson. "*Petri nets*". Computing surveys, vol.9, n.3, September 1977. pp.223-252.

[38]: Tilak Agerwala. "*Putting Petri nets to work*". IEEE COMPUTER December 1979. pp.85-94.

[39]: C.V. Ramamoorthy, G.S. Ho. "*Performance evaluation of asynchronous concurrent systems using Petri nets*". IEEE Transactions on Software Engineering, vol. SE-6, n.5, September 1980. pp.440-449.

[40]: M.K. Molloy. "*Performance analysis using Stochastic Petri nets*". IEEE Transactions on Computers, vol. C-31, n.9, September 1982. pp.913-917.

[41]: M.C. Wei, H.A. Sholl. "*An expression model for extraction and evaluation of parallelism in control structures*". IEEE Transactions on Computers, vol. C-31, n.9, September 1982. pp.851-863.

[42]: S.J. Allan, A.E. Oldehoeft. "*A flow analysis procedure for the translation of High-level languages to a data flow language*". IEEE Transactions on Computers, vol. C-29, n.9, September 1980. pp.826-831.

[43]: F.E. Allen, J. Cocke. "*A program data flow analysis procedure*". Communications of the ACM, vol. 19, n.3, March 1976. pp.137-147.

[44]: A.L. Davis, R.M. Keller. "*Data flow program graphs*". IEEE COMPUTER February 1982. pp.26-41.

[45]: J.L. Gaudiot. "*Structure handling in data-flow systems*". IEEE Transactions on computers, vol. C-35, June 1986. pp.489-502.

[46]: L.A Cohn, T. Mankovich. "*Performance of CHoPP on the Livermore Loops*". Proceedings of the Second International Conference on Supercomputing 1987.

[47]: L. Lipsky, J.D. Church. "*Application of a Queueing Network Model for a Computer System*". Computing Surveys, vol.9 n3, September 1977.

[48]: T.M. Ravi, M.D. Ercegovac, T. Lang, R.R Munty, "*Static Allocation for a Data Flow Multiprocessor System*".Proceedings of the Second International Conference on Supercomputing 1987.

[49]: M.C. Chen, M. Jacquemin. "*Footprints of Dependency: Towards Dynamic Memory Management for Massively Parallel Architectures*". Proceedings of the Third International Conference on Supercomputing 1988.

[50]: J.Y Collin, P. Chretienne "*Allocating tasks on a Virtual Distributed System*". Proceedings of the Third International Conference on Supercomputing 1988.

[51]: G. Gano. "*A Pipelined Code Mapping Strategy for Data Flow Supercomputers*". Proceedings of the Third International Conference on Supercomputing 1988.

[52]: J. Ramanujan, F. Ercal. "*Task allocation by Simulated Annealing*". Proceedings of the Third International Conference on Supercomputing 1988.

[53]: Y. Lan, A. Espahanan. "*Relay Approach Message Routing in Hypercube mutiprocessors*". Proceedings of the Third International Conference on Supercomputing 1988.

[54]: P. Wiley. "*A parallel archtecture comes of age at last*". IEEE Spectrum. June 1987.

[55]: A. Tanenbaum. "*Computer Networks. Second Edition*". Prentice-Hall International. 1989.

[56]: Arvind, K.P. Gostelow. "*The U- Interpreter*". IEEE COMPUTER, 15(2). February 1982.

[57]: B.A. Delagi, N. Saraya. "*Instrumented Architectural Simulation*". Proceedings of the Third International Conference on Supercomputing 1988..

Anexo A

Listados del programa de simulación

Este anexo contine los listados de programa de simulación y el ambiente integrado para el desarrollo de modelos. La actual implementación está escrita en Pascal para una máquina '386 con sistema operativo MSDOS v3.3, monitor EGA color y mouse óptico.

El programa está dividido en 7 unidades:

Defs2.pas : definiciones de constantes y variables globales.
Edito3.pas : sistema de edición gráfica.
Utiles.pas : utilidades para el editor.
Dosfun.pas : funciones estándar del sistema operativo.
Micky.pas : rutinas para el control del ratón.
Bolas4.pas : rutinas de simulación.
Simulator.pas : programa principal.

```
unit defs2;

Interface

{
   ************************************************************
   Esta unidad contiene todas las definiciones de constantes y
   variables globales del sistema de simulacion.  Es utilizada
   por todas las otras unidades (Editor, Bolas, Micky),
   y por el programa principal.
                                   Miguel Mayosky. 17/4/89
   ************************************************************
 }


const
on = true;
off = false;
maxnode = 60;

type

eventclass = record          { evento de simulacion para trace }
 time : longint;             { tiempo de ocurrencia del evento }
 {tipos de eventos }
 class : (hab, beg_ejec, end_ejec, beg_com,
          rut_stp, end_com, full_buf); { clases de eventos }
 info: array [1..4] of integer; { parametros del evento }
 end;

stp_ptr = ^steps;

steps = record
 id : integer;
 next : stp_ptr;
 end;

disp = array[1..10] of string ; {menues}

arco = record                { arcos del  grafo          }
 id    : integer;            { identificacion del nodo   }
 ccost : integer;            { costo de comunicacion     }
 prob  : integer;            { probabilidad de token     }
end;

{ En esta nueva version solo utilizo una estructura de
  datos para algoritmo y arquitectura. De esta manera
  utilizo el mismo editor para todo, y los mismos pro-
  cedimientos para manipulacion de las listas.
  Miguel Mayosky. 2/5/89.}


element = record                    { nodo }
 id       : integer;                { identificacion del nodo }
 class    : integer;                { clase : procesador,
                                        switch, memoria,etc. }
 pcost    : integer;                { costo de procesamiento }
 input_policy:integer;              { politica de entrada }
 exec_policy:integer;               { politica de ejecucion }
```

```
 inputs  : array [1..10] of arco;  { arcos de entrada al nodo }
 outputs : array [1..10] of arco;  { arcos de salida del nodo }
 visible:boolean;                  { nodo visible-invisible:solo
                                     se usa durante simulacion }
 x       :integer;                 { coordenadas en pantalla }
 y       :integer;
end;

pointr = ^list;

list = record          { item en el matching store-token queue }
 token     : element;      { item de programa }
 instancia : stp_ptr;  { numero de activacion del token (dynamic) }
 hard      : integer;  { id del nodo de la arquitectura }
 next      : pointr;   { puntero al proximo item de la lista }
end;

clase = (logico,intermedio,final); { informacion para el ruteo }

arcpntr = ^arrow;

arrow = record         { arco ruteado (arc-queue) }
 inicio   : integer;  {nodo software fuente}
 final    : integer;  {nodo software destino}
 h_source : integer;  {nodo hardware fuente asignacion}
 h_end    : integer;  {nodo hardware destino asignacion}
 instancia: stp_ptr;  {instancia del nodo fuente}
 class    : clase ;   {clase de arco ruteado}
 costo    : integer;  {costo del arco}
 next     : arcpntr;  {puntero al proximo de la lista}
end;

archptr=^archi;

archi=record           { Buffer de editor }
 nodo:element;         { nodo del grafo }
 instancia:integer;    { numero de ejecucion }
 macro:archptr;        { puntero a otro grafo (macro) }
 next:archptr;         { puntero al proximo nodo }
 end;

tipografo = (arquitectura,algoritmo); { tipos de grafo }



r_ptr = ^routing;

routing = record       { router queue }
 arc : arcpntr;        { arco de algoritmo }
 ruta : stp_ptr;       { cola de pasos de ruteo }
 next : r_ptr;
 end;


lnk_ptr = ^link;

link = record          { link queue }
     bloqued :boolean;   { nodo-bloqueado si se supera cap }
```

```
      cap : integer;        { capacidad del nodo }
      cola : integer;       { inicio del link (HW) }
      punta : integer;      { final del link (HW) }
      costo : integer;      { performance relativa de comunicaciones }
      P_list : arcpntr;     { subcola de ruteo-fifo del link }
      next : lnk_ptr;       { proximo elemento de la lista }
      end;


static_assign = record { asignacion procesador-nodo de algoritmo }
 task:integer;
 procesador: integer;
 end;

assi  = array [1..maxnode] of static_assign; { asignacion }


var
circulo,polig,cuadra,romb,dib:pointer; { Dibujos del editor }
zeronode:element; {un nodo en blanco}
first_time_arq,first_time_alg,first_time_sim:boolean;
{ Colas del simulador }

Router_queue:r_ptr;
link_queue : lnk_ptr;
arq_buf,alg_buf,macrolist:archptr;
Match_store, Token_queue, Exec_queue, index, index1, index2 : pointr;
arc_queue, output_queue, com_queue, ptr0,ptr1,ptr2 : arcpntr;
tracfile: file of eventclass;

paso_a_paso,animated,trace_on : boolean; { modo de operacion }

t_simul,tmin : longint; { tiempo de simulacion y  proximo evento }

modo_de_operacion:integer;{ simular solo algoritmo o alg+arq }
{modo_de_operacion=0 : se simula solo el algoritmo.
 modo_de_operacion=1 : alg+arq con asignacion estatica manual.
 modo_de_operacion=2 : alg+arq con asignacion dinamica automatica.}

asiga:assi; { asignacion de tareas a procesadores }

klf:char;

{ Variables de la interfaz grafica  (Borland) }

  GraphDriver : integer;  { The Graphics device driver }
  GraphMode   : integer;  { The Graphics mode value }
  MaxX, MaxY  : word;     { The maximum resolution of the screen }
  ErrorCode   : integer;  { Reports any graphics errors }
  MaxColor    : word;     { The maximum color value available }
  OldExitProc : Pointer;  { Saves exit procedure address }

Implementation
end.{unit}
```

```
unit edito3 ;

{ programa principal y procedimientos fundamentales del editor de
  grafos. El resto de las rutinas esta en utiles.pas }

Interface
uses crt,dos,graph,defs2,micky,utiles,dosfun;

procedure init_editor(var Storage:archptr);
procedure Gr_editor(var Storage:archptr;
                    var tipo:tipografo; nivel:integer);

implementation

procedure Init_editor(var storage:archptr);
{
 Inicializacion del editor.
 se crea el apuntador a la estructura de datos principal (storage)
 se inicializan los menues, la pantalla, un nodo nulo, etc.
}
begin

{ nombre de los archivos de entrada y salida }
 archivo_salida:='';
 archivo_entrada:='';

{ menues varios }
 kinit_menus;

{ pantalla }
 setviewport(0,0,maxx,maxy-14,true);
 cleardevice;
 window(1,25,80,25);
 clrscr;

 new(storage);


{ zeronode se usa para inicializar los nodos. tiene todo en cero }
 with zeronode do
  begin
   id:=0;
   class:=0;
   pcost:=0;
   input_policy:=0;
   exec_policy:=0;
   for i:=1 to 10 do
    begin
     inputs[i].id:=0;inputs[i].ccost:=0;
     inputs[i].prob:=0;
     outputs[i].id:=0;outputs[i].ccost:=0;
     outputs[i].prob:=0;
    end;
  end;
 storage^.nodo:=zeronode;
 storage^.nodo.id:=-1;
 storage^.next:=nil;
end;
```

```
procedure Gr_editor(var storage:archptr;
                    var tipo:tipografo; nivel:integer);


type zone= (iconos,menues,barras,dibujo);  { zonas de la pantalla }

var i,numm1,numm2:integer;
    second,current:archptr;
    macro,grouped,deleted,good_bye:boolean;
    rr:registers;


function where:zone;
{
  Determina la zona en la que se encuentra el cursor.
  Existen 4 zonas: menues, iconos, barras y dibujo.
}
begin
 get_pos(m_stat,m_par);
 if ((m_stat.xnuevo > iconx) and (m_stat.ynuevo < menuy))
     then where:=menues
 else if ((m_stat.xnuevo < iconx) and (m_stat.ynuevo < icony)) then
      where:=iconos
      else if ((m_stat.xnuevo > xlimit) or (m_stat.ynuevo > ylimit))
           then where:=barras
           else where:=dibujo;
end;

procedure search_text(x,y:integer);
var p:cartelptr;
begin
 settextstyle(defaultfont,horizdir,0);
 settextjustify(lefttext,toptext);
 x:=realx(x);
 y:=realy(y);
 texpunt:=textos^.next;
 while texpunt <> nil do
  begin
   if ( (texpunt^.owner=nivel) and ( x>texpunt^.x) and
      ( x<(texpunt^.x+textwidth(texpunt^.text))) and
      ( y>texpunt^.y) and (y<(texpunt^.y+textheight(texpunt^.text))))
      then
    begin
     hide_cursor(m_stat,m_par);
     moveto(normx(texpunt^.x),normy(texpunt^.y));
     kleer_texto(texpunt^.text,white,black);
     if texpunt^.text='' then
      begin
       p:=textos;
       while p^.next<> texpunt do p:=p^.next;
       p^.next:=texpunt^.next;
       dispose(texpunt);
       texpunt:=p;
      end;
    end;
  texpunt:=texpunt^.next;
 end;
end;
```

```
procedure write_text;
{ dibuja texto en la pantalla }
begin
 initm(m_stat,m_par,ok,80,50);
 reverse_icon(64);
 writeln;
 write('Modo texto: posicione el cursor y presione el boton izquierdo
 set_text_cursor;
 show_cursor(m_stat,m_par);
 delay(500);
 m_stat.teclar:=false;
 m_stat.tecla1:=false;

 repeat
  if ((where=barras)) then
   begin
    initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
    show_cursor(m_stat,m_par);
    repeat
     if m_stat.tecla1 then
      begin
       update_view(change);
       if change then
        begin
         kshow(storage,tipo);
         show_macro(macrolist,grouped);
         reverse_icon(64);
        end;
      end;
    until where<>barras;
    set_text_cursor;
   end;
  if(m_stat.tecla1) then
  if(full) then
   begin
    move_view; {muevo la ventanita}
    set_text_cursor;
   end
  else if ((m_stat.xnuevo > iconx) or (m_stat.ynuevo > icony)) then
   begin
    new(texpunt);
    texpunt^.x:=realx(m_stat.xnuevo);
    texpunt^.y:=realy(m_stat.ynuevo);
    texpunt^.owner:=nivel;
    texpunt^.next:=textos^.next;
    texpunt^.text:='';
    hide_cursor(m_stat,m_par);
    moveto(m_stat.xnuevo,m_stat.ynuevo);
    kleer_texto(texpunt^.text,white,black);
    if texpunt^.text = '' then dispose(texpunt)
    else textos^.next:=texpunt;
    show_cursor(m_stat,m_par);
    get_pos(m_stat,m_par);
   end;
until m_stat.teclar;
initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
reverse_icon(64);
show_cursor(m_stat,m_par);
writeln;
```

```
end;


procedure draw_lines;
{
 Rutina de dibujo de lineas entre nodos.
}

type
punto= record
x,y:integer;
end;
var init,fin:punto;
    inicio,final:archptr;
    line_active:boolean;

begin
 reverse_icon(0);  { icono de lineas invertido }
 writeln;
 write('Modo Linea: Boton izquierdo para iniciar-guiar linea.
                    Derecho: salir           ');

 initm(m_stat,m_par,ok,40,40);

 setcolor(white);
 setwritemode(xorput);
 m_stat.tecla1:=false;
 change:=false;
 set_style;
 show_cursor(m_stat,m_par);
 line_active :=false; { Estoy en modo linea pero no estoy dibujando.

 setviewport(0,0,xlimit,ylimit,clipon);

 repeat
{movimiento de barras}
  if ((where=barras)) then
   begin

{ en la zona de barras cambio el cursor}
    initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
    show_cursor(m_stat,m_par);

    repeat
     if m_stat.tecla1 then
      begin
       update_view(change); {muevo las barras}
       if change then
        begin
         if line_active then
          begin  {actualizo la linea}
           init.x:=inicio^.nodo.x;
           init.y:=inicio^.nodo.y;
          end;
         kshow(storage,tipo); {muestro el resultado}
         reverse_icon(0);
         show_macro(macrolist,grouped);
        end;
       end;
```

```
      setwritemode(xorput);
     until where<>barras;
     if change and line_active then
      line(normx(init.x),normy(init.y),fin.x,fin.y);
     set_style;
     change:=false;
     show_cursor(m_stat,m_par);
    end;

{si estoy en modo full muevo la ventanita}
  if (full and m_stat.tecla1 and (where=dibujo)) then
   begin
    move_view;
    setwritemode(xorput);
    set_style;
   end;
{si no (finalmente) dibujo lineas}
  if  (not line_active) and (m_stat.tecla1) then
   begin
    puntero:=storage;
{verifico que el punto pertenezca a un polig}
    kscan(m_stat.xnuevo,m_stat.ynuevo,puntero,is_in_circle) ;
    if is_in_circle then
     begin   { si estoy sobre una poly comienzo a dibujar }
      line_active:=true;          { marca de inicio de linea }
      setwritemode(xorput);
      inicio:=puntero;
      init.x:=puntero^.nodo.x;
      init.y:=puntero^.nodo.y;
      fin.x:=m_stat.xnuevo;
      fin.y:=m_stat.ynuevo;
      delay(200); {antirebote}
      m_stat.tecla1:=false;
     end;
  end;

 get_pos(m_stat,m_par);
 if (line_active) and (not m_stat.tecla1) then
  begin
   if (m_stat.xnuevo <> fin.x) or (m_stat.ynuevo <> fin.y) then
    begin
      hide_cursor(m_stat,m_par);
      line(normx(init.x),normy(init.y),fin.x,fin.y);
      fin.x:=m_stat.xnuevo;
      fin.y:=m_stat.ynuevo;
      line(normx(init.x),normy(init.y),fin.x,fin.y);
      show_cursor(m_stat,m_par);
     end;
    end;

{ Puntos intermedios de la trayectoria }

 if  (line_active) and (m_stat.tecla1) then
  begin
   if (m_stat.xnuevo <> init.x) or  (m_stat.ynuevo <> init.y) then
    begin
     m_stat.tecla1:=false;
     hide_cursor(m_stat,m_par);
     puntero:=storage;
```

```
      kscan(m_stat.xnuevo,m_stat.ynuevo,puntero,is_in_circle);

      if is_in_circle then
       begin
        final:=puntero;
        kinsert_line(inicio,final);
        line_active:=false;
        line(normx(init.x),normy(init.y),fin.x,fin.y);
        setwritemode(copyput);
        smart_line(normx(init.x),normy(init.y),
         normx(puntero^.nodo.x),normy(puntero^.nodo.y),true,storage);
        delay(200); { anti rebote por las dudas }
        m_stat.teclal:=false;
       end
      else
       begin
        setwritemode(copyput);
        line(normx(init.x),normy(init.y),fin.x,fin.y);
        setwritemode(xorput);
       end;


    setcolor(black);
    str(inicio^.nodo.id,sidmx);
    if (inicio^.nodo.class < 90) then
     begin
      settextjustify(centertext,centertext);
      settextstyle(smallfont,horizdir,4);
      outtextxy(normx(inicio^.nodo.x),normy(inicio^.nodo.y),sidmx);
      str(final^.nodo.id,sidmx);
      outtextxy(normx(final^.nodo.x),normy(final^.nodo.y),sidmx);
     end;
    setcolor(white);
    show_cursor(m_stat,m_par);
    init.x:=realx(m_stat.xnuevo);  { actualizo el comienzo de linea }
    init.y:=realy(m_stat.ynuevo);

    initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
    set_style;
    show_cursor(m_stat,m_par);
   end;
  end;

  get_pos(m_stat,m_par);
 until ((not line_active) and (m_stat.teclar));
 line_active:=false;
 m_stat.teclar:=false;
 setwritemode(copyput);
 settextstyle(defaultfont,horizdir,1);
 settextjustify(lefttext,toptext);
 setcolor(white);
 if not full then reverse_icon(0);
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 show_cursor(m_stat,m_par);
 writeln;
end;

procedure erase_nodes;
{
```

```
 rutina de borrado de nodos
}
var id:integer;

begin
 set_erase_cursor; {cursor con forma de goma}
 reverse_icon(32); {invierto el icono de borrado}

 writeln;
 write('Boton izquierdo para borrar. Derecho para salir');

 repeat
{actualizacion de barras}
 if ((where=barras)) then
  begin
   initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
   show_cursor(m_stat,m_par);
   repeat
    if m_stat.tecla1 then
     begin
      update_view(change);
      if change then
       begin
        kshow(storage,tipo);
        show_macro(macrolist,grouped);
        reverse_icon(32);
       end;
      end;
     until where<>barras;
     set_erase_cursor;
     show_cursor(m_stat,m_par);
    end;
{si estoy en vision full, muevo la ventanita}
   if (full and m_stat.tecla1 and (where=dibujo)) then
    begin
     move_view;
     set_erase_cursor;
    end;
{ahora borro}
   if (m_stat.tecla1) then
    begin
     m_stat.tecla1:=false;
     puntero:=storage;
     kscan (m_stat.xnuevo,m_stat.ynuevo,puntero,is_in_circle);
     if ((is_in_circle) and (puntero^.nodo.class < 90)) then
      begin
       id := puntero^.nodo.id;
       kkill_node(puntero,storage);
{ si el nodo esta en la lista de seleccion multiple, tambien lo borro
       puntero:=macrolist;
       repeat
        puntero:=puntero^.next
        until ((puntero=nil) or (puntero^.nodo.id=id));
       if puntero <> nil then kkill_node(puntero,macrolist);
       hide_cursor(m_stat,m_par);
       kshow(storage,tipo);
       show_macro(macrolist,false);
       reverse_icon(32);
       show_cursor(m_stat,m_par);
```

```
      end;
   end;
  get_pos(m_stat,m_par);
  until m_stat.teclar;

  m_stat.teclar:=false;
  reverse_icon(32);
  initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
  show_cursor(m_stat,m_par);
end;


procedure draw_nodes(dib:archptr;xicon:integer);
{
 rutina para dibujar nodos.
}
begin
 reverse_icon(xicon); {icono invertido}

 writeln;
 write('Boton izquierdo para fijar imagen. Derecho:salir.');

 m_stat.tecla1:=false;
 set_screen_for_circles;
 setcolor(black);
 settextjustify(centertext,centertext);
 settextstyle(smallfont,horizdir,4);

 repeat
{actualizacion de las barras}
  if ((where=barras)) then
   begin
    initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
    show_cursor(m_stat,m_par);
     repeat
      if m_stat.tecla1 then
       begin
        update_view(change);
        if change then
         begin
          kshow(storage,tipo);
          show_macro(macrolist,grouped);
          reverse_icon(xicon);
         end;
       end;
      until where<>barras;
      setcolor(black);
      settextjustify(centertext,centertext);
      settextstyle(smallfont,horizdir,4);
      set_screen_for_circles;
     end;
    if(m_stat.tecla1) then
     begin
      if(full) then
       begin
        move_view; {muevo la ventanita}
        setcolor(black);
        settextjustify(centertext,centertext);
        settextstyle(smallfont,horizdir,4);
```

```
          set_style;
         end

      else if(where=dibujo) then
       begin
        hide_cursor(m_stat,m_par);
        putimage(m_stat.xnuevo-15,m_stat.ynuevo-15,dib^,orput);
        m_stat.tecla1:=false;
        new(puntero);
        puntero^.nodo:=zeronode;
        puntero^.macro:=nil;
        puntero^.next:=storage^.next;
        storage^.next:=puntero;
        puntero^.nodo.id:=idmx;
        puntero^.nodo.x:=realx(m_stat.xnuevo);
        puntero^.nodo.y:=realy(m_stat.ynuevo);
        puntero^.nodo.visible:=false;
        puntero^.instancia:=0;
        if xicon=128 then puntero^.nodo.class:=1 {memoria}
        else if xicon=160 then puntero^.nodo.class:=2;{switch}
        str(idmx,sidmx);
        outtextxy(m_stat.xnuevo,m_stat.ynuevo,sidmx);
        idmx:=idmx+1;
        if tipo=algoritmo then idmxalg:=idmx
                          else idmxarq:=idmx;
        set_screen_for_circles;
        delay(200); { anti rebote }
        m_stat.tecla1:=false;
       end;
    end;
   get_pos(m_stat,m_par);
  until m_stat.teclar;
  m_stat.tecla1:=false;
  m_stat.teclar:=false;
  settextstyle(defaultfont,horizdir,1);
  settextjustify(lefttext,toptext);
  setcolor(white);
  if not full then reverse_icon(xicon);
  initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
  show_cursor(m_stat,m_par);
  writeln;
end;


begin {main}


{
 si es la primera vez que entro al editor, creo las listas accesorias
 una -macrolist- es para la seleccion multiple, la otra para texto.
}
 if (first_time_alg and first_time_arq) then
  begin
   new(macrolist);
   macrolist^.next:=nil;
   new(textos);
   textos^.next:=nil;
   textos^.text:='';
   idmxalg:=1;
```

```
    idmxarq:=1;
   end;
  grouped:=false;  {seleccion no agrupada}
  macro:=false;    {no hay nodos seleccionados}
  setviewport(0,0,maxx,maxy,clipoff);

  statnode:=nivel;
  str(nivel,statmsg);
  statmsg:='Nodo '+statmsg;

 {
 Variables de pantalla:
 }

  full:=false;                          {modo normal o full view}
  menuy:=12;                            {coordenada y de la linea de men
  sizex := maxx div screnum;            {tama$o de la barra segun x}
  sizey := (maxy-38) div screnum;       {tama$o de la barra segun y}
  xlimit := maxx-20;                    {xminimo linea de barra}
  ylimit := maxy-26;                    {yminimo linea de barra}
  winx:=0;                              {coordenada x del "mundo"}
  winy:=0;                              {idem y}
  xact:=1;                              {coordenada de la ventana segun
  yact:=13;                             {idem y}
  oldwinx:=0;                           {coordenadas mundo anteriores}
  oldwiny:=0;

  cleardevice;
  window(1,25,80,25);
  clrscr;

 {c: pongo el cursor}

  initm(m_stat,m_par,ok,240,80);
 {si el mouse no esta instalado pataleo...}
  if not ok then
   begin
    outtextxy(maxx div 2 - 80, maxy div 2 ,'Mouse no instalado. ');
    espera;
    halt(1);
   end;

 {adquiero las imagenes}
  kdraw_icons(tipo);
  if ((first_time_alg) and (tipo=algoritmo)) then
   begin
    get_icons(tipo);
    first_time_alg:=false;
    idmxalg:=1;
    new(textos_alg);
    textos_alg^.next:=nil;
    textos_alg^.text:='';
   end;
  if ((first_time_arq) and (tipo=arquitectura)) then
   begin
    get_icons(tipo);
    first_time_arq:=false;
    idmxarq:=1;
    new(textos_arq);
```

```
    textos_arq^.next:=nil;
    textos_arq^.text:='';
   end;

  if tipo=algoritmo then
   begin
    textos:=textos_alg;
    idmx:=idmxalg;
    iconx:=128;
    icony:=32;
   end
  else
   begin
    textos:=textos_arq;
    idmx:=idmxarq;
    iconx:=192;
    icony:=32;
   end;

  kshow(storage,tipo);
  show_macro(macrolist,false);
  show_cursor(m_stat,m_par);

  macro:=false;

  numm1:=4;  { si hay seleccion multiple, el numero de opciones }
  numm2:=5;  { cambia.Estos dos numeros son las lineas de cada menu. }

  good_bye:=false; {si goodbye es cierto, me voy}
  repeat

 {Tratamiento de la linea de menu}

   if (where=menues)  then case m_stat.xnuevo of

    257..335 : begin
                hide_cursor(m_stat,m_par);
                reverse_tit(257,'FILE',false);
                option:=0;
                empty;
                ktext_window(257,numm1,menu1,option);
                case option of
                 1: begin  { carga de archivo }
                     eliminar(storage);
                     eliminar(macrolist);
                     elim_text(textos);
                     numm1:=4;
                     numm2:=5;
                     macro:=false;
                     grouped:=false;
                     idmx:=idmx-1;
                     kcargar_archivo(archivo_entrada,storage,idmx);
                     if tipo=algoritmo then idmxalg:=idmx
                     else idmxarq:=idmx;
                     kshow(storage,tipo);
                    end;
                 2: ksalvar_archivo(archivo_salida,storage^.next);
                 3: begin
                     empty;
```

```
                direc;                                  { directorio }
               end;
            4: begin
                eliminar(macrolist);
                macro:=false;
                grouped:=false;
                if tipo=algoritmo then idmxalg:=idmx
                else idmxarq:=idmx;
                good_bye:=true;
               end;
            5: begin  { salvar seleccion }
                aislar_macro(macrolist);
                ksalvar_archivo(archivo_salida,macrolist^.next);
                eliminar(macrolist);
                hide_cursor(m_stat,m_par);
                kshow(storage,tipo);
                show_cursor(m_stat,m_par);
                numm1:=4;
                numm2:=5;
                macro:=false;
                grouped:=false;
               end;


          end; {case}
          reverse_tit(257,'File',true);
          show_cursor(m_stat,m_par);
          end;


336..416 : begin
           hide_cursor(m_stat,m_par);
           empty;
           reverse_tit(339,'EDIT',false);
           ktext_window(336,numm2,menu2,option);
           case option of
            1: begin
                eliminar(storage);
                eliminar(macrolist);
                elim_text(textos);
                numm1:=4;
                numm2:=5;
                idmx:=1;
                if tipo=algoritmo then idmxalg:=idmx
                else idmxarq:=idmx;
                macro:=false;
                grouped:=false;
                kshow(storage,tipo); {volvemos a empezar...}
               end;
            2: begin
                kshow(storage,tipo);   {volvemos a dibujar...}
                show_macro(macrolist,grouped);
               end;
            3: begin
                intr(5,rr);
               end;
            4: begin { comandos del sistema operativo }
                dos_window;
               end;
```

```
                5: begin { seleccionar todo }
                    eliminar(macrolist);
                    puntero:=storage^.next;
                    while puntero <> nil do
                     begin
                      new(second);
                      second^.nodo:=puntero^.nodo;
                      second^.macro:=nil;
                      second^.next:=macrolist^.next;
                      macrolist^.next:=second;
                      puntero:=puntero^.next;
                     end;
                    if macrolist^.next <> nil then
                     begin
                      grouped:=false;
                      macro:=true;
                      numm2:=7;
                      numm1:=5;
                      show_macro(macrolist,grouped);
                     end;
                    end;

                6: begin   { borrar seleccion }
                    current:=macrolist^.next;
                    while current <> nil do
                     begin
                      puntero:=storage;
                      kscan(normx(current^.nodo.x),
                            normy(current^.nodo.y)
                            ,puntero,is_in_circle);
                      if is_in_circle then
                         kkill_node(puntero,storage);
                      current:=current^.next;
                     end;
                    eliminar(macrolist);
                    hide_cursor(m_stat,m_par);
                    kshow(storage,tipo);
                    show_cursor(m_stat,m_par);
                    numm2:=5;
                    numm1:=4;
                    macro:=false;
                    grouped:=false;
                   end;

                7: begin { agrupar seleccion }
                    grouped:=true;
                    show_macro(macrolist,grouped);
                   end;


                end;   {case}

             reverse_tit(339,'Edit',true);
             show_cursor(m_stat,m_par);
            end;

end; {case}
```

```
{tratamiento de la zona de iconos}

if ((where=iconos) and (m_stat.tecla1) and (not full)) then
 case m_stat.xnuevo of

   0..32   : if storage^.next <> nil then draw_lines;

  33..64   : if storage^.next <> nil then erase_nodes;

  65..96   : write_text;

 97..128   : if tipo = arquitectura then draw_nodes (polig,96)
             else draw_nodes(circulo,96);

 129..160 : if tipo = arquitectura then draw_nodes (cuadra,128);

 161..192 : if tipo = arquitectura then draw_nodes (romb,160);

 end; {case}


{tratamiento de la zona de barras}
 if ((where=barras) and (m_stat.tecla1)) then
  begin
   update_view(change);
   if change then
    begin
     kshow(storage,tipo);
     show_macro(macrolist,grouped);
    end;
   end;

{tratamiento de la zona de dibujos}
 if ((where=dibujo) and (m_stat.tecla1)) then
 if full then move_view   {muevo la ventanita si estoy en vision full
 else
  begin
   puntero:=macrolist;
   kscan(m_stat.xnuevo,m_stat.ynuevo,puntero,is_in_circle);
   if is_in_circle then
    begin
    {si estoy en un nodo seleccionado, lo muevo}
     move_nodes(storage,macrolist,tipo,grouped);
     grouped:=false;
    end;
   puntero:=storage;
   kscan(m_stat.xnuevo,m_stat.ynuevo,puntero,is_in_circle);
   if ((is_in_circle) and (puntero^.nodo.class<90)) then
    begin
    {programacion de los parametros del nodo}
     hide_cursor(m_stat,m_par);
     kleo_parametros(storage,puntero,tipo,deleted);
     if puntero^.nodo.class=3 then
      begin {adentro del nodo macro}
       eliminar(macrolist);
       macro:=false;
       grouped:=false;
       if puntero^.macro=nil then init_editor(puntero^.macro);
       gr_editor(puntero^.macro,tipo,puntero^.nodo.id);
```

```
        statnode:=nivel;
        str(nivel,statmsg);
        statmsg:='Nodo '+statmsg;
        kshow(storage,tipo);
       end;
      show_cursor(m_stat,m_par);
     end
    {si no estoy sobre un nodo entro en modo seleccion}
    else if not full then
     begin
      search_text(m_stat.xnuevo,m_stat.ynuevo);{ seleccion texto }
      macro_ed(storage,macrolist,numm1,numm2); { seleccion nodo }
     end;
    if macrolist^.next <> nil then macro:=true;
   end;


 get_pos(m_stat,m_par);

{suprimir buffer de seleccion}
 if (m_stat.teclar) and macro then
  begin
   eliminar(macrolist);
   grouped:=false;
   macro:=false;
   hide_cursor(m_stat,m_par);
   kshow(storage,tipo);
   show_cursor(m_stat,m_par);
   numm2:=5;
   numm1:=4;
  end;


  until good_bye;
 end;
end.{unit}
```

```
unit utiles ;
{
  Esta unidad contiene las rutinas de utilidades para el editor
  version -edito3- en adelante.
 }

Interface
uses crt,dos,graph,defs2,micky,dosfun;

const
  screnum=4;  {numero de pantallas sobre cada eje}
  {Si se quiere modificar el tama$o de la zona de edicion, modificar
   screnum. 4 parece ser un valor adecuado}


type
arq=file of element;

nodeclass=array[0..7]of string; { clases de nodos permitidas. }

{ los textos se guardan como una lista de strings. el parametro
  owner indica a que nodo -ventana- pertenece el texto }
cartelptr=^cartel;    { textos }

cartel= record
x,y,owner:integer; { x,y: coordenadas del string. owner: nodo dueño }
text:string;       { texto }
next:cartelptr;    { proximo texto }
end;
cara = file of cartel; { los textos se guardan en un archivo aparte }

var
    armenu,almenu,police:nodeclass; {clases de nodos y politicas i/o}
    menu1,menu2:disp;            {menues}
    cursor : char;               {para readkey's varios}
    idmx,idmxalg,idmxarq,x,y,cize,i,j,n,
    xmin,ymin,xmax,ymax,option,xicon:integer;
    valor:longint;
    mask,titulo,sidmx:string;
    change:boolean;
    is_in_circle:boolean;
    zeropoly:element;  {una poly en blanco}

    p:pointer;              {punteros al heap para get-putimage}
    size,sizel : integer;
    archivo_entrada,archivo_salida:string;{nombres de los archivos}
    arq_file:arq; txtfil:cara;
    puntero,begend:archptr;
    dirinfo:searchrec;columna,fila:integer;direcname:string;

{virtual page variables}
  sizex,sizey,xlimit,ylimit,xact,yact,winx,winy,iconx,icony,menuy,
  oldwinx,oldwiny,oldxact,oldyact,oldsizex,oldsizey:integer;
  ok,full:boolean;

  textos_alg,textos_arq,textos,texpunt: cartelptr;

  {
   statmsg es el titulo de la sesion activa -cartel superior derecho-
```

```
   statnode es el numero del nodo actualmente en edicion
  }
  statmsg: string;
  statnode:integer;


function normx(a:integer):integer;
function normy(a:integer):integer;
function norm(a:integer):integer;
function realx(x:integer):integer;
function realy(y:integer):integer;
function no_spaces(name:string):string;
procedure draw_bars;
procedure update_view (var change:boolean);
procedure move_view;
procedure espera;
procedure empty;
procedure delete_arc(base,n:archptr;i:integer);
procedure kinit_menus;
procedure kdraw_marco(tipo:tipografo);
procedure get_icons(var tipo:tipografo);
procedure kdraw_icons(var tipo:tipografo);
procedure reverse_icon(x:integer);
Procedure reverse_tit( inicio:integer; tit:string; r_mode:boolean);
procedure kscan (x,y:integer;var p:archptr;var is_in_circle:boolean);
Procedure ktext_window(inicio,lineas:integer;
                       var menu:disp;var opt:integer);
procedure kpunta(xi,yi,xf,yf:integer);
Procedure Smart_Line (x1,y1,x2,y2:integer;
                      no:boolean; var base:archptr);
procedure kleer_numeros(var n:integer);
procedure kleer_texto(var texto:string;lapiz,fondo:integer);
procedure kLeo_Parametros(base,n:archptr;tipo:tipografo;
                          var deleted:boolean);
procedure ksalvar_archivo(var name:string; var punt:archptr);
procedure kcargar_archivo(var name:string; var base:archptr;
                          var idmx:integer);
procedure kinsert_line(var p_inicio,p_final :archptr);
procedure search_limits(macrolist:archptr; var x1,y1,x2,y2:integer);
procedure show_macro(macrolist:archptr; grouped:boolean);
procedure kshow(var base:archptr; tipo:tipografo);
procedure kkill_node(var p,base:archptr);
procedure aislar_macro(p:archptr);
procedure eliminar(var base:archptr);
procedure Macro_ed(var storage,macrolist:archptr;
                   var numm1,numm2:integer);
procedure move_nodes( storage,macrolist:archptr;tipo:tipografo;
                      var grouped:boolean);
procedure elim_text(var p:cartelptr);


implementation

procedure draw_bars;
{dibuja las barras de movimiento de pantalla}
begin
 hide_cursor(m_stat,m_par);
 setlinestyle(solidln,0,normwidth);
 settextstyle(defaultfont,horizdir,0);
```

```
 settextjustify(centertext,centertext);
 setfillstyle(solidfill,black);
 setcolor(white);
 setviewport(0,0,maxx,maxy,clipon);
 bar(xlimit+1,12,maxx-1,maxy-15);     { area de barras vertical }
 bar(1,ylimit+1,maxx-1,maxy-15);      { area de barras horizontal }
 rectangle(xlimit,12,maxx,maxy-14);   { area de barras vertical }
 rectangle(0,ylimit,maxx,maxy-14);    { area de barras horizontal }
 setfillstyle(closedotfill,white);
 bar(xlimit+1,yact+1,maxx-1,yact+sizey-1); { barra movil vertical }
 rectangle(xlimit+1,yact+1,maxx-1,yact+sizey-1);
 bar(xact+1,ylimit+1,xact+sizex-1,maxy-15); { barra movil horiz. }
 rectangle(xact+1,ylimit+1,xact+sizex-1,maxy-15);
 if full then outtextxy(xlimit+10,ylimit+7,'N')
         else outtextxy(xlimit+10,ylimit+7,'F');
 show_cursor(m_stat,m_par);
end;


procedure update_view (var change:boolean);
{ Este procedimiento se utiliza para actualizar las barras }
var x,y,delta :integer;

begin
 setwritemode(copyput);
 setlinestyle(solidln,0,normalput);
 setcolor(white);
 m_stat.tecla1:=false;
 get_pos(m_stat,m_par);
 setviewport(0,0,maxx,maxy,clipon);
 change:=false;

 if (m_stat.tecla1) and (m_stat.xnuevo>xlimit)
     and (m_stat.ynuevo>ylimit) then
  begin
   change:=true;        { hay cambios }
   full:=not full;
   if full then
    begin  { guardo configuracion anterior }
     oldsizex:=sizex;
     oldsizey:=sizey;
     sizex:=maxx-21;
     sizey:=maxy-39;
     oldwinx:=winx;oldwiny:=winy;
     oldxact:=xact;oldyact:=yact;
     winx:=1;winy:=1;yact:=13;xact:=1;
    end
   else
    begin { recupero configuracion anterior }
     winx:=oldwinx;winy:=oldwiny;
     xact:=oldxact;yact:=oldyact;
     sizex := oldsizex; {tama$o de la barra segun x}
     sizey := oldsizey;    {tama$o de la barra segun y}
    end;
   draw_bars;
  end;

{actualizacion barra vertical}
```

```
 if ((m_stat.tecla1) and (m_stat.xnuevo > xlimit) and
    (m_stat.ynuevo > yact) and (m_stat.ynuevo < (yact+sizey))) then
  begin
   y:= m_stat.ynuevo;
   change:=true;
  { reprogramo limites del mouse }
   m_stat.ymin:=y-yact+1;
   delta:=m_stat.ymin;
   m_stat.ymax:=ylimit-(yact+sizey-y);
   set_y(m_stat,m_par);
   m_stat.xmin:=m_stat.xnuevo;
   m_stat.xmax:=m_stat.xnuevo;
   set_x(m_stat,m_par);

  {muevo la barra}
   while ((m_stat.tecla1) and (m_stat.xnuevo > xlimit)) do
    begin
     if m_stat.ynuevo <> y then
      begin
       setfillstyle(solidfill,black);
       hide_cursor(m_stat,m_par);
       bar(xlimit+1,yact,xlimit+19,yact+sizey);
       y:=m_stat.ynuevo;
       yact:=y-delta;
       if yact<13 then yact:=13;
       setfillstyle(closedotfill,white);
       bar(xlimit+1,yact,xlimit+19,yact+sizey);
       rectangle(xlimit+1,yact,xlimit+19,yact+sizey);
       show_cursor(m_stat,m_par);
      end;
     get_pos(m_stat,m_par);
    end;
   end;

{lo mismo con la barra horizontal}

 if ((m_stat.tecla1) and (m_stat.ynuevo > ylimit) and
    (m_stat.xnuevo > xact) and (m_stat.xnuevo < (xact+sizex))) then
  begin
   x:= m_stat.xnuevo;
   change:=true;
   m_stat.xmin:=x-xact;
   delta:=m_stat.xmin;
   m_stat.xmax:=xlimit-(xact+sizex-x)-1;
   set_x(m_stat,m_par);
   m_stat.ymin:=m_stat.ynuevo;
   m_stat.ymax:=m_stat.ynuevo;
   set_y(m_stat,m_par);
   while ((m_stat.tecla1) and (m_stat.ynuevo > ylimit)) do
    begin
     if m_stat.xnuevo <> x then
      begin
       setfillstyle(solidfill,black);
       hide_cursor(m_stat,m_par);
       bar(xact,ylimit+1,xact+sizex,ylimit+11);
       x:=m_stat.xnuevo;
       xact:=x-delta;
       if xact<1 then xact:=1;
       setfillstyle(closedotfill,white);
```

```
        bar(xact,ylimit+1,xact+sizex,ylimit+11);
        rectangle(xact,ylimit+1,xact+sizex,ylimit+11);
        show_cursor(m_stat,m_par);
       end;
      get_pos(m_stat,m_par);
     end;
    end;

{winx y winy son las coordenadas de la ventana}

 winx:=(xact-1) * screnum;
 winy:=(yact-13) * screnum;
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 show_cursor(m_stat,m_par);
end;

procedure move_view;

{Estando en vision full, permite mover la ventana de edicion}

var x,y,deltax,deltay:integer;
begin
 setcolor(white);
{ si estoy dentro de la ventana...}
 if ((m_stat.xnuevo > oldxact)
      and (m_stat.xnuevo < (oldxact+oldsizex))
      and (m_stat.ynuevo > oldyact)
      and (m_stat.ynuevo < (oldyact+oldsizey))) then
  begin
   setwritemode(xorput);
   setlinestyle(dottedln,0,normwidth);
   get_pos(m_stat,m_par);
   x:=m_stat.xnuevo;
   y:=m_stat.ynuevo;
   deltax:=x-oldxact;
   deltay:=y-oldyact;

   {nuevos limites para el movimiento del cursor}
   m_stat.xmin:=m_stat.xnuevo-oldxact+1;
   m_stat.xmax:=maxx-21-(oldxact+oldsizex-m_stat.xnuevo);
   m_stat.ymin:=m_stat.ynuevo-oldyact+13;
   m_stat.ymax:=maxy-27-(oldyact+oldsizey-m_stat.ynuevo);
   set_x(m_stat,m_par);
   set_y(m_stat,m_par);
   set_open_hand;

   {movemos la ventanita...}
   while m_stat.tecla1 do
    begin
     get_pos(m_stat,m_par);
     if (m_stat.xnuevo <> x) or (m_stat.ynuevo <> y) then
      begin
       hide_cursor(m_stat,m_par);
       rectangle(oldxact,oldyact,oldxact+oldsizex,oldyact+oldsizey);
       x:=m_stat.xnuevo;
       y:=m_stat.ynuevo;
       oldxact:=x-deltax;
       oldyact:=y-deltay;
       rectangle(oldxact,oldyact,oldxact+oldsizex,oldyact+oldsizey);
```

```
        show_cursor(m_stat,m_par);
       end;
      end;
     oldwinx:=oldxact * screnum;
     oldwiny:=oldyact * screnum;
     initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
     show_cursor(m_stat,m_par);
     setlinestyle(solidln,0,normwidth);
     setwritemode(copyput);
    end;
 end;

function normx(a:integer):integer;

{ normalizacion coordenadas x. }

begin
 if full then normx:=(a-winx) div screnum
         else normx:=(a-winx) ;
end;

function normy(a:integer):integer;

{ normalizacion coordenadas y }

begin
 if full then normy:=(a-winy) div screnum
         else normy:=(a-winy);
end;

function norm(a:integer):integer;

{ normalizacion modulos }

begin
if full then norm:=a div screnum
        else norm:=a;
end;

function realx(x:integer):integer;

{obtiene la coordenada x real a partir de la coordenada de pantalla}

begin
if full then realx :=x*screnum
       else realx:=x+winx; {facil, verdad?}
end;

function realy(y:integer):integer;

{obtiene la coordenada y real a partir de la coordenada de pantalla}

begin
if full then realy:= y*screnum
        else realy:=y+winy;
end;
```

```
procedure espera;
{ vacia el buffer de caracteres y espera un caracter de teclado }
var regs:registers;
    cursor:char;

begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
 cursor:=readkey;
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

procedure empty;
{ vacia el buffer de caracteres }
var regs:registers;

begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

function no_spaces(name:string):string;
{ elimina los espacios de un string }
var i:integer;
begin
 repeat
  i:=pos(' ',name);
  if i<>0 then delete(name,i,1);
 until i=0;
 no_spaces:=name;
end;

procedure delete_arc(base,n:archptr;i:integer);
{borra arcos}
var p:archptr;
    j,k:integer;
begin
 p:=base^.next;
 while p^.nodo.id <> n^.nodo.outputs[i].id do p:=p^.next;
 j:=1;
 while p^.nodo.inputs[j].id <> n^.nodo.id do j:=j+1;
 for k:=j to 9 do  p^.nodo.inputs[k]:=p^.nodo.inputs[k+1];
 p^.nodo.inputs[10].id:=0;
 for k:=i to 9 do  n^.nodo.outputs[k]:=n^.nodo.outputs[k+1];
 n^.nodo.outputs[10].id:=0;
end;


procedure kinit_menus;
{************************KINIT_MENUS*****************************
 Inicializacion de las lineas de menu: Si se quiere agregar o
 modificar lineas, hacerlo aqui. Cuando luego se llama a
 text-window (ver main), modificar el parametro lineas de
 acuerdo al numero de opciones usado }
```

```
begin
{ Menu File }
 menu1[1]:=' ~ Cargar Archivo.';
 menu1[2]:=' ~ Salvar Archivo.';
 menu1[3]:=' ~ Directorio.';
 menu1[4]:=' ~ Salir.';
 menu1[5]:=' ~ Salvar seleccion.';

{ Menu Edit }
 menu2[1]:=' ~ Reset editor.';
 menu2[2]:=' ~ Redibujar.';
 menu2[3]:=' ~ Impr. Pantalla.';
 menu2[4]:=' ~ Comandos DOS.';
 menu2[5]:=' ~ Seleccionar todo.';
 menu2[6]:=' ~ Borrar seleccion.';
 menu2[7]:=' ~ Agrupar seleccion.';

{ Tipos de nodos de Arquitectura }
 armenu[0]:='PROCESADOR';
 armenu[1]:='MEMORIA';
 armenu[2]:='SWITCH';

{ Tipos de nodos de Algoritmo }
 almenu[0]:='STANDARD';
 almenu[1]:='CALL' ;
 almenu[2]:='RETURN';
 almenu[3]:='MACRO';
 almenu[4]:='JOIN';
 almenu[5]:='INPUT';
 almenu[6]:='OUTPUT';

{ Tipos de politicas de entrada/salida }
 police[0]:='AND';
 police[1]:='OR';

end;{init_menus}

procedure kdraw_marco(tipo:tipografo);
{*************************KDRAW_MARCO*************************
 dibuja la pantalla del editor
}
begin
 setviewport(0,0,maxx,maxy,clipon);
 setwritemode(copyput);
 setlinestyle(solidln,0,normwidth);
 setcolor (white);
 rectangle(0,0,maxx,maxy-14);
 setfillstyle(solidfill,white);
 bar(iconx,0,maxx,menuy);
 settextjustify(centertext,centertext);
 settextstyle(defaultfont,horizdir,0);
 setcolor(black);
 line(255,0,255,12);
 line(337,0,337,12);
 line(419,0,419,12);
 rectangle(421,1,546,11);
 outtextxy(255+41,7,'File');
 outtextxy(337+41,7,'Edit');
 outtextxy(484,7,statmsg);
```

```
 setfillstyle(solidfill,red);
 bar (548,1,638,11);
 setcolor(white);
 outtextxy(593,7,'BOLAS V0.0');
 settextjustify(lefttext,toptext);
 setfillstyle(solidfill,black);
 floodfill(2,2,white);
 draw_bars;
end;{ draw_marco }

procedure get_icons(var tipo:tipografo);
{*************************GET_ICONS*************************
 adquiere los iconos del editor
}

var sizec:integer;

begin
 sizec:=imagesize(1,1,31,31);
 if tipo=arquitectura then
  begin
   getmem(polig,sizec);
   getmem(cuadra,sizec);
   getmem(romb,sizec);
   getimage(97,1,127,31,polig^);
   getimage(129,1,159,31,cuadra^);
   getimage(161,1,191,31,romb^);
  end
 else
  begin
   getmem(circulo,sizec);
   getimage(97,1,127,31,circulo^);
  end;
end; {get_icons}


procedure kdraw_icons(var tipo:tipografo);
{*************************DRAW_ICONS*************************
 dibuja los iconos del editor
}

const
{ simbolo de procesador }
octop : array[1..9] of pointtype = ((x:108;y:8),
                                    (x:116;y:8),
                                    (x:120;y:12),
                                    (x:120;y:20),
                                    (x:116;y:24),
                                    (x:108;y:24),
                                    (x:104;y:20),
                                    (x:104;y:12),
                                    (x:108;y:8));

{ simbolo de switch }
swpol : array[1..5] of pointtype = ((x:176 ;y:8 ),
                                    (x:184;y:16),
                                    (x:176;y:24),
                                    (x:168;y:16),
                                    (x:176;y:8));
```

```
begin

 setcolor(white);
 setfillstyle(solidfill,black);
 setviewport(0,0,maxx,maxy,clipon);
 if tipo=arquitectura then
  begin
   bar(0,0,192,32);
   rectangle(0,0,192,32);
   line(128,0,128,32);
   line(160,0,160,32);
  end
else
 begin
  bar(0,0,128,32);
  rectangle(0,0,128,32);
 end;
line(32,0,32,32);
line(64,0,64,32);
line(96,0,96,32);
setfillstyle (solidfill,white);
line (3,29,29,3); {icono de linea}
rectangle(40,8,56,24);
line(40,16,56,16);
floodfill(48,17,white);  {goma de borrar}
settextjustify(centertext,centertext);
outtextxy(80,16,'TEXT');
settextjustify(lefttext,toptext);

if tipo=arquitectura then
 begin
  drawpoly(9,octop);        {icono de procesador}
  rectangle(136,8,152,24); {icono de memoria}
  drawpoly(5,swpol);        {icono de switch}
  floodfill(112,16,white);
  floodfill(137,16,white);
  floodfill(176,16,white);
 end
else
 begin
  circle(112,16,10);         {icono de programa}
  floodfill(112,10,white);
 end;
end; {draw_icons}

procedure reverse_icon(x:integer);
{ dibuja el icono en video inverso }
var tamano:integer;p:pointer;
begin
 if not full then
  begin
   hide_cursor(m_stat,m_par);
   tamano:=imagesize(x+2,2,x+30,33);
   getmem(p,tamano);
   getimage(x+2,2,x+30,30,p^);
   putimage(x+2,2,p^,notput);
   freemem(p,tamano);
   show_cursor(m_stat,m_par);
```

```
  end;
end;

Procedure reverse_tit( inicio:integer; tit:string; r_mode:boolean);

{invierte el nombre del menu seleccionado.}
{inicio:coordenada x comienzo titulo.}
{tit:titulo. r_mode=true:fondo blanco letras negras,false:invertido}

var back,pen:integer;
begin
 settextjustify(centertext,centertext);
 settextstyle(defaultfont,horizdir,0);
 if r_mode then
  begin
   back:=white;
   pen:=black;
  end
 else
  begin
   back:=black;
   pen:=white;
  end;
 setfillstyle (solidfill,back);
 bar(inicio,1,inicio+78,11);
 setcolor(pen);
 outtextxy(inicio+40,7,tit);
 settextjustify(lefttext,toptext);
end;

procedure kscan (x,y:integer;var p:archptr;var is_in_circle:boolean);
{*****************************KSCAN*****************************
 verifica si un punto x,y de la pantalla pertenece a un elemento.
 si pertenece, p apunta al elemento, is_in_circle:true.
 si no pertenece, is_in_circle:false }
begin
 is_in_circle:=false;
 x:=realx(x);
 y:=realy(y);
 repeat
  p:=p^.next;
  with p^ do
   begin
    if (x>nodo.x-8) and (x<nodo.x+8) and (y<nodo.y+8)
        and (y>nodo.y-8) then  is_in_circle := true;
   end;
 until (p=nil) or (is_in_circle);
end;


Procedure ktext_window(inicio,lineas:integer;
                       var menu:disp;var opt:integer);
{************************KTEXT_WINDOW*****************************
 crea una ventana de texto a partir de inicio (x minimo), y muestra
 "lineas" de menu. permite moverse a traves de las opciones del menu.
 se selecciona con el boton izquierdo del mouse, se aborta clickeando
 fuera de la ventana.El valor seleccionado se devuelve en opt }

type event = (none, change, select, quit, out);
```

```
var sizet:integer;
    p:pointer;
    opt1,xmin, xmax, ymin, ymax, y, i,alto,nn : integer;
    chopt:char;
    action: event;

function dentro:boolean;
{true si dentro de la ventana}
begin
 get_pos(m_stat,m_par);
 dentro:=((m_stat.xnuevo<=xmax) and (m_stat.xnuevo>=xmin) and
          (m_stat.ynuevo<=ymax) and (m_stat.ynuevo>=ymin))
end;

begin

 settextjustify(lefttext,toptext);
 settextstyle(defaultfont,horizdir,0);
 setfillstyle(solidfill,lightblue);
 if graphdriver = EGA then alto := 14
    else alto :=8;

 xmin:=inicio;
 xmax:=inicio+190;
 ymin:=14;
 ymax:=15+alto*(lineas+1);
 sizet:=imagesize (xmin,ymin,xmax,ymax);
 getmem(p,sizet);
 getimage(xmin,ymin,xmax,ymax,p^);
 bar(xmin,ymin,xmax,ymax);
 setcolor(white);
 rectangle(xmin+2,ymin+2,xmax-2,ymax-2);
 for i:=1 to lineas do outtextxy(xmin,ymin+i*alto,menu[i]);
 opt:=1;
 action := change;
 initm(m_stat,m_par,ok,xmin+(xmax-xmin) div 2, ymin+alto);
 show_cursor(m_stat,m_par);
 repeat
  if action = change then
   begin  {fondo blanco letras rojas mayusculas}
    hide_cursor(m_stat,m_par);
    moveto (xmin,ymin+opt*alto);
    setcolor(white);
    outtextxy(xmin+3,ymin+opt*alto,' [[[[[[[[[[[[[[[[[[');
    setcolor(red);
    for i:=1 to length(menu[opt]) do outtext(upcase(menu[opt,i]));
    show_cursor(m_stat,m_par);
   end;

  action := none;
  repeat
   if dentro then
    begin
     opt1:=(m_stat.ynuevo - ymin) div alto; {indice}
     if ((opt1 >0) and (opt1 <=lineas)) then
      begin
       if (m_stat.tecla1) then  action := select  {seleccion}
       else if (opt1<>opt) then action := change;{cambio de renglon}
```

```
      end;
     end
    else if m_stat.tecla1 then action := quit  {abortar}
         else
          begin
           action:=out;  {deshabilitar seleccion mientras mouse fuera
          end;
  until (action <> none);   {hasta que pase algo}
  hide_cursor(m_stat,m_par);{reescribo renglon en modo normal}
  moveto (xmin,ymin+opt*alto);
  setcolor(lightblue);
  outtextxy(xmin+3,ymin+opt*alto,'[[[[[[[[[[[[[[[[[[[[');
  setcolor(white);
  outtext(menu[opt]);
  show_cursor(m_stat,m_par);
  if action <> out then opt:=opt1
  else
   begin
    repeat until ((dentro) or (m_stat.tecla1));  {espero hasta entrar
    action:=change;
   end;
  get_pos(m_stat,m_par);
  until (action = select) or (action = quit) ;
 if  (action = quit) then opt:=0;
 setcolor(white);
 setfillstyle(solidfill,white);
 hide_cursor(m_stat,m_par);
 putimage(xmin,ymin,p^,normalput);
 freemem(p,sizet);
 initm(m_stat,m_par,ok,m_stat.xnuevo, m_stat.ynuevo);
end;



procedure kpunta(xi,yi,xf,yf:integer);
{**************************KPUNTA************************
 Horrible forma de poner poner punta a las flechas }

var x1,y1,x2,y2,xxx,yyy:real;
    angle:real;
    long,radio:integer;

begin
 if full then
  begin
   long:=3;
   radio:=4;
  end
 else
  begin
   long:=7;
   radio:=9;
  end;
 x1:=xf-xi;
 y1:=yf-yi;

 if x1<>0 then angle:=ArcTan(abs(y1/x1))
 else angle:=pi/2;
```

```
 x1:=long*cos(angle+(pi/6));
 y1:=long*sin(angle+(pi/6));
 x2:=long*cos(angle-(pi/6));
 y2:=long*sin(angle-(pi/6));

 if xf>=xi then
  begin
   xxx := xf-radio*cos(angle);
   x1:=xxx-x1;
   x2:=xxx-x2;
  end
 else
  begin
   xxx := xf+radio*cos(angle);
   x1:=xxx+x1;
   x2:=xxx+x2;
  end;

 if yf>=yi then
  begin
   yyy := yf-radio*sin(angle);
   y1:=yyy-y1;
   y2:=yyy-y2;
  end
 else
  begin
   yyy := yf+radio*sin(angle);
   y1:=yyy+y1;
   y2:=yyy+y2;
  end;

 line(xi,yi,trunc(xxx),trunc(yyy));
 line(trunc(xxx),trunc(yyy),xf,yf);
 line(trunc(xxx),trunc(yyy),trunc(x2),trunc(y2));
 line(trunc(xxx),trunc(yyy),trunc(x1),trunc(y1));

end;


Procedure Smart_Line (x1,y1,x2,y2:integer;
                      no:boolean; var base:archptr);
{**************************SMART_LINE*********************************
  Este procedimiento dibuja lineas, sorteando los posibles obstaculos
  en la trayectoria. Es utilizado por el programa editor, y llama a l
  rutina kpunta. Verificada 25/4/89.
}
Type
 vector = record
 x,y : longint;
 end;

Var
 delta:integer;
 rectang,inter,c:array [1..4] of vector;
 m: array [1..4] of longint;
 i,j,signox,signoy:longint;
 nearest:longint;
 dentro,up:boolean;
 next:vector;
```

```
 p,index:archptr;

function Producto_escalar (v1,v2:vector):longint;
begin
 producto_escalar := (v1.x * v2.x) + (v1.y * v2.y);
end;

begin
 if full then delta:=12 div screnum
         else delta:=12;
{ Reconozco el cuadrante }
 if y2=y1 then signox:=delta
          else signox:= ( (y2-y1) div abs(y2-y1) ) * delta;
 if x2=x1 then signoy:=delta
          else signoy:= ( (x2-x1) div abs(x2-x1) ) * delta;

{ Vertices del rectangulo }

 C[1].x := x1 + signox;
 C[1].y := y1 - signoy;
 C[2].x := x1 - signox;
 C[2].y := y1 + signoy;
 C[3].x := x2 - signox;
 C[3].y := y2 + signoy;
 C[4].x := x2 + signox;
 C[4].y := y2 - signoy;


{ vectores perpendiculares a los lados del rectangulo }

 rectang[1].x := ( c[2].y-c[1].y);
 rectang[1].y :=-( c[2].x-c[1].x);
 rectang[2].x := ( c[3].y-c[2].y);
 rectang[2].y :=-( c[3].x-c[2].x);
 rectang[3].x := ( c[4].y-c[3].y);
 rectang[3].y :=-( c[4].x-c[3].x);
 rectang[4].x := ( c[1].y-c[4].y);
 rectang[4].y :=-( c[1].x-c[4].x);

 index:=base^.next;
 dentro:= true;
 p:=base^.next;
 nearest:=maxint;

{ vectores desde cada uno de los vertices al punto en estudio }

 while p <> nil do
  begin
   dentro:=true;
    for j:=1 to 4 do
     begin
      inter[j].x := normx(p^.nodo.x) - c[j].x;
      inter[j].y := normy(p^.nodo.y) - c[j].y;
    { si todos los productos escalares son > 0 el punto en estudio }
    { esta dentro del rectangulo: es un obstaculo }

      m[j] := producto_escalar ( rectang [j], inter[j]);
      if m[j] <= 0 then dentro:=false;
     end;
```

```
{ si esta adentro busco el mas cercano al punto de origen }

   if dentro then
    begin
     if (m[1]<>0) and (m[1]<nearest) then
      begin
       index :=p;
       nearest:=m[1];
      end;
    end;
   p:=p^.next;
  end; {while p}

{ busco si esta sobre o bajo la linea entre p1 y p2 }

 if nearest<>maxint then
  begin
   inter[1].x:=normx(index^.nodo.x) -c[1].x;
   inter[1].y:=normy(index^.nodo.y) -c[1].y;
   inter[2].x:=normx(index^.nodo.x) -c[2].x;
   inter[2].y:=normy(index^.nodo.y) -c[2].y;
   m[1]:= producto_escalar(rectang[2],inter[1]);
   m[2]:= producto_escalar(rectang[2],inter[2]);

{ calculo el proximo punto de la trayectoria }

   if abs(m[1]) > abs(m[2])  then
    begin {paso por abajo}
     next.y:= normy(index^.nodo.y)-signoy;
     next.x:= normx(index^.nodo.x)+signox;
    end

   else
    begin {paso por arriba}
     next.y := normy(index^.nodo.y)+signoy;
     next.x := normx(index^.nodo.x)-signox;
    end;

   smart_line(x1,y1,next.x,next.y,false,base);
   smart_line(next.x,next.y,x2,y2,true,base);
  end
 else if not no then line (x1,y1,x2,y2)
      else kpunta (x1,y1,x2,y2);
end;



procedure kleer_numeros(var n:integer);
{**************************KLEER_NUMEROS**********************
 lee numeros en modo grafico. hacer moveto(x,y) previamente
 para ubicar el cursor. Se borra con del y se termina con enter
 n retorna el numero}

var pirulo:string;
    i:integer;
    a:char;
    special: boolean;
    x,y,xini:integer;
```

```
begin
 x:=getx;
 xini:=x;
 y:=gety;
 pirulo:='';
 a:=' ';
 settextstyle(defaultfont,horizdir,0);
 settextjustify(lefttext,toptext);
 special:=false;
 str(n,pirulo);
 hide_cursor(m_stat,m_par);
 outtext(pirulo);
 x:=x+length(pirulo)*8;
 outtext('_');
 show_cursor(m_stat,m_par);
 delay(100); { antirebote del mouse }
 repeat
  empty;
  repeat get_pos(m_stat,m_par) until (m_stat.tecla1 or keypressed);
  if not m_stat.tecla1 then
   begin
    a:=readkey;
    case a of
     #0: special:=true;
     #83: if special then
           begin { borro caracteres del string }
            delete(pirulo,length(pirulo),1);
            if x >xini then x:=x-8;
            setcolor(white);
            hide_cursor(m_stat,m_par);
            outtextxy(x,y,'[');
            setcolor(blue);
            outtextxy(x,y,'_');
            show_cursor(m_stat,m_par);
            special:=false;
           end;
     #13: ;
#48..#57: if not special then
           begin
            pirulo:=concat(pirulo,a);
            setcolor(white);
            hide_cursor(m_stat,m_par);
            outtextxy(x,y,'[');
            setcolor(blue);
            outtextxy(x,y,a);
            x:=x+8;
            outtextxy(x,y,'_');
            show_cursor(m_stat,m_par);
           end;
   end;
  end;
 until (( a=chr($d)) or (m_stat.tecla1)) ;
 setcolor(white);
 hide_cursor(m_stat,m_par);
 outtextxy(x,y,'[');
 show_cursor(m_stat,m_par);
 setcolor(blue);
 if pirulo<>'' then val(pirulo,n,i);
 if i<>0 then n:=0;
```

```
end;

procedure kleer_texto(var texto:string;lapiz,fondo:integer);
{*****************************KLEER_TEXTO*************************
 lee texto en modo grafico. Hacer moveto(x,y) previamente para
 ubicar el cursor. Permite borrar e insertar.
 lapiz es el color de la letra,fondo es el color del background}

var
    i,k:integer;
    a:char;
    x,y,xini:integer;
begin
 x:=getx;
 xini:=x;
 y:=gety;
 a:=' ';
 settextstyle(defaultfont,horizdir,0);
 settextjustify(lefttext,toptext);
 setcolor(lapiz);
 outtext(texto);
 x:=x+length(texto)*8;
 i:=length(texto);
 outtext('_');
 repeat
  empty;
  a:=readkey;
  case a of
   #0: begin
        a:=readkey; {si es un caracter especial, lo proceso}
        case a of
         #83: begin { borro caracteres del string }
               delete(texto,i,1);
               if i >0 then i:=i-1;
               if x >xini then x:=x-8;
               setcolor(fondo);
               moveto(x,y);
               for k:=i to (length(texto)+1) do outtext('[');
               setcolor(lapiz);
               outtextxy(x,y,copy(texto,i+1,80));
               outtextxy(x,y,'_');
              end;
         #75: begin  { muevo el cursor hacia atras }
               if x >xini then
                begin
                 setcolor(fondo);
                 outtextxy(x,y,'[');
                 setcolor(lapiz);
                 outtextxy(x,y,copy(texto,i+1,1));
                 x:=x-8;
                 i:=i-1;
                 setcolor(lapiz);
                 outtextxy(x,y,'_');
                end;
             end;
        #77: begin   { muevo el cursor hacia adelante }
              if x <(xini+textwidth(texto)) then
               begin
                setcolor(fondo);
```

```
                outtextxy(x,y,'[');
                setcolor(lapiz);
                outtextxy(x,y,copy(texto,i+1,1));
                x:=x+8;
                i:=i+1;
                setcolor(lapiz);
                outtextxy(x,y,'_');
               end;
             end;
         end;
       end;

    #32..#122: begin
                insert(a,texto,i+1);
                setcolor(fondo);
                moveto(x,y);
                for k:=i to length(texto) do outtext('[');
                setcolor(lapiz);
                outtextxy(x,y,copy(texto,i+1,80));
                x:=x+8;
                i:=i+1;
                outtextxy(x,y,'_');
               end;
    end;
  until a=chr($d);
  setcolor(fondo);
  setwritemode(xorput);
  outtextxy(x,y,'_');   { borro el cursor }
  setwritemode(copyput);
  setcolor(lapiz);
end;

procedure kLeo_Parametros(base,n:archptr;
                          tipo:tipografo;var deleted:boolean);
{*****************************KLEO_PARAMETROS**********************
 lee-edita los parametros de un nodo-arcos de salida }


var pirulo,pirula:pointer;
 tam,tam1:word;
 iden:string[3];
 alto,xmin,xmax,ymin,ymax,x1min,x1max,y1min,y1max,i,k :integer;
 label 1;

procedure select_class(menuc:nodeclass;n:integer;var k:integer);
{ seleccion en linea de opciones. Las opciones estan contenidas en
  menuc. n es el numero de opciones. k es la opcion seleccionada.
  Se selecciona con uparrow-downarrow. Se termina con return
}
var x,y:integer;a,b:char;

begin
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 x:=getx;y:=gety;
 setcolor(white);
 outtext('[[[[[[[[[[[[');
 moveto(x,y);
 setcolor(red);
 outtext(menuc[k]);
```

```
 repeat
  empty;
  repeat get_pos(m_stat,m_par) until (keypressed or m_stat.tecla1);
  if not m_stat.tecla1 then
   begin
    a:=readkey;
    if a=#0 then
     begin
      b:=readkey;
      a:='a';
      case b of
            #72: begin {up arrow}
                  k:=k-1;
                  if k<0 then k:=n-1;
                 end;
            #80: begin {down arrow}
                  k:=k+1;
                  if k=n then k:=0;
                 end;
      end; {case}
     moveto(x,y);
     setcolor(white);
     outtext('[[[[[[[[[[[[');
     moveto(x,y);
     setcolor(red);
     outtext(menuc[k]);
    end;{if}
   end;

 until ((a=chr($d)) or (m_stat.tecla1));
 moveto(x,y);
 setcolor(blue);
 settextstyle(defaultfont,horizdir,0);
 settextjustify(lefttext,toptext);
 outtext(menuc[k]);
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
end;

begin
 deleted:=false;
 if graphdriver =EGA then alto :=14
                     else alto:=8;
 xmin:=200;
 xmax:=440;
 ymin:=100;
 setfillstyle(solidfill,white);
 if tipo=arquitectura then ymax:=ymin+5*alto
                      else ymax:=ymin+6*alto;
 tam:=imagesize (xmin,ymin,xmax,ymax);
 getmem(pirulo,tam);
 getimage(xmin,ymin,xmax,ymax,pirulo^);
 setcolor(blue);
 bar(xmin,ymin,xmax,ymax);
 rectangle(205,105,435,ymax-5);
 rectangle(207,107,433,120);
 str(n^.nodo.id,iden);
 outtextxy(290,111,'Nodo ');
 outtextxy(338,111,iden);
 outtextxy(220,111+alto,  '~ V.Proc:');
```

```
 str(n^.nodo.pcost,iden);
 outtextxy(300,111+alto,iden);
 outtextxy(220,111+2*alto,'~ Clase:');
 if tipo=arquitectura then
     outtextxy(300,111+2*alto,armenu[n^.nodo.class])
 else
  begin
   outtextxy(300,111+2*alto,almenu[n^.nodo.class]);
   outtextxy(220,111+3*alto,'~ Entrada:');
   outtextxy(300,111+3*alto,police[n^.nodo.input_policy]);
   outtextxy(220,111+4*alto,'~ Salida:');
   outtextxy(300,111+4*alto,police[n^.nodo.exec_policy]);
  end;
 moveto(300,111+alto);
 kleer_numeros(n^.nodo.pcost);
 moveto(300,111+2*alto);
 if tipo=arquitectura then select_class(armenu,3,n^.nodo.class)
 else
  begin
   select_class(almenu,7,n^.nodo.class);
   moveto(300,111+3*alto);
   select_class(police,2,n^.nodo.input_policy);
   moveto(300,111+4*alto);
   select_class(police,2,n^.nodo.exec_policy);
  end;
{arcos}
 i:=1;
 if n^.nodo.outputs[1].id <>0 then
  begin
   x1min:=300;
   x1max:=540;
   y1min:=150;
   y1max:=y1min+4*alto;
   tam1:=imagesize (x1min,y1min,x1max,y1max);
   getmem(pirula,tam1);
   getimage(x1min,y1min,x1max,y1max,pirula^);
   setcolor(lightgray);
   rectangle(x1min,y1min,x1max,y1max);

   repeat
    if n^.nodo.outputs[i].id <> 0 then
     begin
      setcolor(blue);
      bar(x1min+1,y1min+1,x1max-1,y1max-1);
      rectangle(305,155,535,145+4*alto);
      rectangle(307,157,533,171);
      outtextxy(320,161,'Arco ');
      moveto(368,161);
      str(n^.nodo.id,iden);
      outtext(iden);
      outtext(' - ');
      str(n^.nodo.outputs[i].id,iden);
      outtext(iden);

{ icono de borrado de arco }
      rectangle(481,159,531,169);
      outtextxy(483,161,'DELETE');
      m_stat.xant:=500;
      m_stat.yant:=180;
```

```
      set_cursor(m_stat,m_par);
      show_cursor(m_stat,m_par);

      outtextxy(320,161+alto,  '~ V.Com:');
      str(n^.nodo.outputs[i].ccost,iden);
      outtextxy(392,161+alto,iden);
      if tipo=arquitectura then outtextxy(320,161+2*alto,'~ Capac:')
      else outtextxy(320,161+2*alto,'~ Prob.:');
      str(n^.nodo.outputs[i].prob,iden);
      outtextxy(392,161+2*alto,iden);
      moveto(392,161+alto);
      initm(m_stat,m_par,ok,500,180);
      show_cursor(m_stat,m_par);
      kleer_numeros(n^.nodo.outputs[i].ccost);
      if (m_stat.tecla1
          and (m_stat.xnuevo < 531) and (m_stat.xnuevo > 481)
          and (m_stat.ynuevo < 168) and (m_stat.ynuevo > 159)) then
       begin
        setfillstyle(solidfill,black);
        initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
        bar(481,159,531,169);
        setcolor(white);
        outtextxy(483,161,'DELETE');
        delete_arc(base,n,i);
        delay(500);
        setfillstyle(solidfill,white);
        setcolor(blue);
        deleted:=true;
        i:=i-1 ;
        show_cursor(m_stat,m_par);
        goto 1;
       end;
      moveto(392,161+2*alto);
      initm(m_stat,m_par,ok,500,180);
      show_cursor(m_stat,m_par);
      kleer_numeros(n^.nodo.outputs[i].prob);
      if (m_stat.tecla1
          and (m_stat.xnuevo < 531) and (m_stat.xnuevo > 481)
          and (m_stat.ynuevo < 168) and (m_stat.ynuevo > 159)) then
       begin
        initm(m_stat,m_par,ok,500,180);
        setfillstyle(solidfill,black);
        bar(481,159,531,169);
        setcolor(white);
        outtextxy(483,161,'DELETE');
        delete_arc(base,n,i);
        delay(500);
        setfillstyle(solidfill,white);
        setcolor(blue);
        deleted:=true;
        i:=i-1;
        show_cursor(m_stat,m_par);
       end;
1 :   hide_cursor(m_stat,m_par);
     end;

    i:=i+1;

  until i=11;
```

```
    putimage(x1min,y1min,pirula^,normalput);
    freemem(pirula,tam1);
   end;{fin arco}
 putimage(xmin,ymin,pirulo^,normalput);
 freemem(pirulo,tam);
 if deleted then kshow(base,tipo);
 initm(m_stat,m_par,ok,maxx div 2,maxy div 2);
 m_stat.teclar:=false;
 setcolor(white);
end;

procedure ksalvar_archivo(var name:string; var punt:archptr);
{************************KSALVAR_ARCHIVO*************************
 obviamente, salvar el buffer del editor en un archivo a partir de
 punt}

var xmin,xmax,alto,ymin,ymax,tam:integer;
    pirulo1:pointer;
    p:char;
    punt1:archptr;
    dirinfo:searchrec;
    name1:string;
    i:integer;

procedure salvall(p: archptr);
begin
 while p <> nil do
  begin
   if (p^.nodo.id <> -1) then write(arq_file,p^.nodo);
   if ((p^.nodo.class = 3) and (p^.macro <> nil)) then
      salvall(p^.macro);
   p:=p^.next;
  end;
 zeronode.id:=-2;
 write(arq_file,zeronode);
 zeronode.id:=0;
end;

begin
{$I-}
 p:='S';
 if graphdriver=ega then alto:=14
                    else alto:=8;
 xmin:=200;
 ymin:=100;
 xmax:=500;
 ymax:=ymin+2*alto;
 tam:=imagesize(xmin,ymin,xmax,ymax);
 getmem(pirulo1,tam);
 getimage(xmin,ymin,xmax,ymax,pirulo1^);
 setcolor(blue);
 rectangle(xmin,ymin,xmax,ymax);
 floodfill(201,101,blue);
 rectangle(xmin+2,ymin+2,xmax-2,ymax-2);
 outtextxy(208,100+alto,'Salvar como: ');
 moveto(308,100+alto);
 kleer_texto(name,blue,white);
 name:=no_spaces(name);
 if name<>'' then
```

```
  begin
   findfirst(name,archive,dirinfo);
   if doserror=0 then
    begin
     writeln;
     p:='N';
     write ('El archivo ',name,' ya existe.
             Desea escribir encima? (S/N): ');
     empty;
     p:=readkey;
     write(p);
    end;
   if (upcase(p)='S')  then
    begin
     assign(arq_file,name);
     rewrite(arq_file);
     punt1:=punt;
     salvall(punt1);
     close(arq_file);
    end;
   if textos^.next <> nil then
    begin
     name1:=name;
     i:=pos('.',name1);
     if i<>0 then  delete(name1,i,length(name1));
     name1:=concat(name1,'.$tx');
     assign(txtfil,name1);
     rewrite(txtfil);
     texpunt:=textos^.next;
     while texpunt <>nil do
      begin
       write(txtfil,texpunt^);
       texpunt:=texpunt^.next;
      end;
     close(txtfil);
    end;
  end;
 writeln;
 setcolor(white);
 putimage(xmin,ymin,pirulo1^,normalput);
 freemem(pirulo1,tam);
end;

procedure kcargar_archivo(var name:string;
                          var base:archptr; var idmx:integer);

{ obviamente, cargar un archivo dentro del buffer del editor}

var xmin,xmax,alto,ymin,ymax,tam,num:integer;
    pirulo1:pointer;
    p:char;
    punt: archptr;
    dirinfo:searchrec;
    name1:string;
    i:integer;

procedure cargall(var base:archptr);
var p:archptr;nodo:element;j:integer;
begin
```

```
 num:=0; { no puede haber dos nodos con el mismo id }
 read(arq_file,nodo);
 while ((nodo.id <> -2) and (not eof(arq_file))) do
  begin
   new(p);
   p^.nodo:=nodo;
   p^.next:=base^.next;
   base^.next:=p;
   if p^.nodo.id > 0 then
    begin { renumero los nodos para evitar problemas}
     with p^.nodo do
      begin
       id:=id+idmx;
       j:=1;
       repeat
        if inputs[j].id <>0 then inputs[j].id:=inputs[j].id+idmx;
        j:=j+1;
        until inputs[j].id=0;
        j:=1;
        repeat
         if outputs[j].id <>0 then outputs[j].id:=outputs[j].id+idmx;
         j:=j+1;
        until outputs[j].id=0;
      end;
     end;
    if p^.nodo.class=3 then
     begin
      new (p^.macro);
      p^.macro^.nodo:=zeronode;
      p^.macro^.nodo.id:=-1;
      p^.macro^.next:=nil;
      cargall(p^.macro);
     end;
    if p^.nodo.id>num then num:=p^.nodo.id;
    read(arq_file,nodo);
 end;
end;

begin
{$I-}

 if graphdriver=ega then alto:=14
                    else alto:=8;
 xmin:=200;
 ymin:=100;
 xmax:=500;
 ymax:=ymin+2*alto;
 tam:=imagesize(xmin,ymin,xmax,ymax);
 getmem(pirulo1,tam);
 getimage(xmin,ymin,xmax,ymax,pirulo1^);
 setcolor(blue);
 rectangle(xmin,ymin,xmax,ymax);
 floodfill(201,101,blue);
 rectangle(xmin+2,ymin+2,xmax-2,ymax-2);
 outtextxy(208,100+alto,'Archivo: ');
 moveto(288,100+alto);
 kleer_texto(name,blue,white);
 name:=no_spaces(name);
 if name<>'' then
```

```
  begin;
   assign(arq_file,name);
   reset(arq_file);
   if ioresult<> 0 then
    begin
     writeln;
     write('El archivo ',name,' no existe. Oprima cualquier tecla...'
     empty;p:=readkey;
    end
   else
    begin
     num:=0;
     cargall(base);
     idmx:=num; { actualizo el numero de identificacion maxima }
     close(arq_file);
 { si hay archivo de texto, tambien lo cargo }
     name1:=name;
     i:=pos('.',name1);
     if i<>0 then  delete(name1,i,length(name1));
     name1:=concat(name1,'.$tx');
     assign(txtfil,name1);
     reset(txtfil);
     if ioresult= 0 then
      begin
       repeat
        new(texpunt);
        read(txtfil,texpunt^);
        texpunt^.next:=textos^.next;
        textos^.next:=texpunt;
       until eof(txtfil);
       close(txtfil);
      end;
   end;
  end;
 setcolor(white);
 putimage(xmin,ymin,pirulo1^,normalput);
 freemem(pirulo1,tam);
 idmx:=idmx+1;
end;

procedure kinsert_line(var p_inicio,p_final :archptr);

{ agrega una linea. Actualiza las entradas y salidas de los nodos
  asociados }

var i:integer;
begin
 i:=0;
 repeat
  i:=i+1;
 until p_inicio^.nodo.outputs[i].id=0;
 p_inicio^.nodo.outputs[i].id := p_final^.nodo.id;
 i:=0;
 repeat
  i:=i+1;
 until p_final^.nodo.inputs[i].id=0;
 p_final^.nodo.inputs[i].id := p_inicio^.nodo.id;
end;
```

```
procedure search_limits(macrolist:archptr; var x1,y1,x2,y2:integer);
{ busca las coordenadas limites de un grupo de nodos }
var p:archptr;
begin
 p:=macrolist^.next;
 x1:=maxint;y1:=maxint;
 x2:=-1;y2:=-1;
 while p<> nil do
  begin
   if P^.nodo.x > x2 then x2 :=p^.nodo.x;
   if P^.nodo.x < x1 then x1 :=p^.nodo.x;
   if P^.nodo.y > y2 then y2 :=p^.nodo.y;
   if P^.nodo.y < y1 then y1 :=p^.nodo.y;
   p:=p^.next;
  end;
 x1:=normx(x1);x2:=normx(x2);y1:=normy(y1);y2:=normy(y2);
end;



procedure show_macro(macrolist:archptr; grouped:boolean);
{ rectangulo alrededor de cada nodo seleccionado }
var p:archptr;
    xmin,ymin,xmax,ymax:integer;
begin
 if not full then
  begin
   setviewport(0,0,xlimit-1,ylimit-1,clipon);
   setcolor(white);
   hide_cursor(m_stat,m_par);
   p:=macrolist^.next;
   xmin:=maxint;ymin:=maxint;
   xmax:=-1;ymax:=-1;
   while p<> nil do
    begin
     rectangle(normx(p^.nodo.x)-12,normy(p^.nodo.y)-12,
               normx(p^.nodo.x)+12,normy(p^.nodo.y)+12);
     if grouped then
      begin
       if P^.nodo.x > xmax then xmax :=p^.nodo.x;
       if P^.nodo.x < xmin then xmin :=p^.nodo.x;
       if P^.nodo.y > ymax then ymax :=p^.nodo.y;
       if P^.nodo.y < ymin then ymin :=p^.nodo.y;
      end;
     p:=p^.next;
    end;
   if grouped then rectangle(normx(xmin)-14,normy(ymin)-14,
                             normx(xmax)+14,normy(ymax)+14);
   show_cursor(m_stat,m_par);
 end;
end;

procedure kshow(var base:archptr; tipo:tipografo);

{ dibuja los contenidos del buffer del editor }

var j,n:integer;
```

```
    p,p1:archptr;
begin
 hide_cursor(m_stat,m_par);
 setviewport(1,icony+1,xlimit-1,ylimit-1,clipon);
 clearviewport;
 setviewport(iconx+1,menuy+1,xlimit-1,icony+1,clipon);
 clearviewport;
 setviewport(0,0,xlimit-1,ylimit-1,clipon);
 if full then
  begin
   clearviewport;
   setviewport(0,menuy,xlimit,ylimit,clipon);
  end;
 settextjustify(centertext,centertext);
 settextstyle(smallfont,horizdir,4);
 setfillstyle (solidfill,white);
 p:=base^.next;
 while p<>nil do
  begin
   setcolor(black);
{ selecciono la imagen a dibujar }
   if tipo=arquitectura then
    begin
     if p^.nodo.class = 0 then dib:=polig
     else if p^.nodo.class = 1 then dib := cuadra
          else if p^.nodo.class = 2 then dib:=romb;
    end
   else dib:=circulo;

{ dibujo el nodo }

 if not full then
  begin
   if ((normx(p^.nodo.x) < maxx) and (normy(p^.nodo.y) < maxy)) then
    begin
     putimage(normx(p^.nodo.x-15),normy(p^.nodo.y-15),dib^,orput);
     str(p^.nodo.id,sidmx);
     outtextxy(normx(p^.nodo.x),normy(p^.nodo.y),sidmx);
    end
   end
  else bar(normx(p^.nodo.x)-2,normy(p^.nodo.y)-2,
           normx(p^.nodo.x)+2,normy(p^.nodo.y)+2);

{ dibujo los arcos que salen de los nodos }
 j:=1;
 setcolor(white);
 repeat
  if p^.nodo.outputs[j].id <>0 then
   begin
    n:=p^.nodo.outputs[j].id;
    p1:=base^.next;
    while (p1^.nodo.id <> n) do  p1:=p1^.next;
    smart_line(normx(p^.nodo.x),normy(p^.nodo.y),normx(p1^.nodo.x),
               normy(p1^.nodo.y),true,base);
    if p=p1 then circle(normx(p^.nodo.x+17),
                        normy(p^.nodo.y-10),norm(17));
  { escribo la identificacion de los nodos destino y fuente }
    if not full then
     begin
```

```
        setcolor(black);
        str(p^.nodo.id,sidmx);
        outtextxy(normx(p^.nodo.x),normy(p^.nodo.y),sidmx);
        str(p1^.nodo.id,sidmx);
        outtextxy(normx(p1^.nodo.x),normy(p1^.nodo.y),sidmx);
        setcolor(white);
       end;
      end;
     j:=j+1;
     until j=10;

     p:=p^.next;
    end;

   if full then
    begin  {muestro la ventana activa}
     setviewport(0,0,xlimit-1,ylimit-1,clipon);
     setwritemode(xorput);
     setlinestyle(dottedln,0,normwidth);
     rectangle(oldxact,oldyact,oldxact+oldsizex,oldyact+oldsizey);
     setwritemode(copyput);
     setlinestyle(solidln,0,normwidth);
    end;

   settextstyle(defaultfont,horizdir,1);
   settextjustify(lefttext,toptext);

  { escribo el texto }
   if (not full) then
    begin
     texpunt:=textos^.next;
     while texpunt<> nil do
      begin
       if (texpunt^.owner=statnode) then
       outtextxy(normx(texpunt^.x),normy(texpunt^.y),texpunt^.text);
       texpunt:=texpunt^.next;
      end;
     kdraw_icons(tipo)
    end
   else bar(0,0,iconx,menuy);
   kdraw_marco(tipo);
   window(1,25,80,25);
   writeln;
   settextstyle(defaultfont,horizdir,0);
   settextjustify(lefttext,toptext);
   show_cursor(m_stat,m_par);
  end;{show}

  procedure kkill_node(var p,base:archptr);

  { elimina un nodo del buffer del editor. Actualiza las entradas y
    salidas de todos los demas nodos.
  }

  var i,j,l,m : integer;
  p1:archptr;

  begin
   for i:= 1 to 10 do
```

```
   begin
    j:=p^.nodo.outputs[i].id;
    if j<>0 then
     begin
      p1:=base^.next;
      while p1^.nodo.id <>j do p1:=p1^.next;
      for l:=1 to 10 do
       begin
        if p1^.nodo.inputs[l].id=p^.nodo.id then
         begin
          for m:=l to 9 do p1^.nodo.inputs[m]:=p1^.nodo.inputs[m+1];
          p1^.nodo.inputs[10].id:=0;
         end;
       end; {for l}
     end; {if}
  end;{fori}


  for i:= 1 to 10 do
   begin
    j:=p^.nodo.inputs[i].id;
    if j<>0 then
     begin
      p1:=base^.next;
      while p1^.nodo.id <> j do p1:=p1^.next;
      for l:=1 to 10 do
       begin
        if p1^.nodo.outputs[l].id=p^.nodo.id then
         begin
          for m:=l to 9 do p1^.nodo.outputs[m]:=p1^.nodo.outputs[m+1];
          p1^.nodo.outputs[10].id:=0;
         end;{for m}
       end; {for l}
     end;
   end;{fori}

  if p^.macro <> nil then
   begin  { si el nodo es clase macro, elimino su lista }
    eliminar(p^.macro);
    dispose(p^.macro);
   end;

  p1:=base;
  while p1^.next <> p do p1:=p1^.next;
  p1^.next:=p^.next;
  dispose(p);
  p:=base^.next;
end;

procedure aislar_macro(p:archptr);
var pun1,pun2:archptr;
    i,j: integer;
{
 Este procedimiento elimina los arcos externos a una macro.
 de esta manera la macro puede ser almacenada en disco sin
 problemas. Que dificil es explicar ciertas cosas!!
}

begin
```

```
 pun1:=p^.next;
 i:=1;
 while pun1 <> nil do
  begin
   pun2:=p^.next;
 (recorro la lista)
   while ((pun2^.nodo.id<> pun1^.nodo.outputs[i].id) and (pun2 <> nil
          (pun1^.nodo.outputs[i].id <> 0)) do  pun2:=pun2^.next;

   if pun1^.nodo.outputs[i].id <> 0 then
    begin
     if pun2^.nodo.id = pun1^.nodo.outputs [i].id then
          { el arco es interno al macro }
        i:=i+1
     else
      begin { el arco es externo-->se elimina}
       for j:=i to 9 do
        begin
         pun1^.nodo.outputs[j]:=pun1^.nodo.outputs[j+1];
         pun1^.nodo.outputs[10].id:=0;
        end;
      end;
   end;
  if ((i=11) or (pun1^.nodo.outputs[i].id=0)) then
   begin
    pun1:=pun1^.next;
    i:=1;
   end;
 end;
pun1:=p^.next;
i:=1;
while pun1 <> nil do
 begin
  pun2:=p^.next;
{recorro la lista}
  while ((pun2^.nodo.id<> pun1^.nodo.inputs[i].id) and (pun2 <> nil)
         and (pun1^.nodo.inputs[i].id <> 0)) do  pun2:=pun2^.next;

  if pun1^.nodo.inputs[i].id <> 0 then
   begin
    if pun2^.nodo.id = pun1^.nodo.inputs [i].id then
         { el arco es interno al macro }
     i:=i+1
     else
      begin { el arco es externo-->se elimina}
       for j:=i to 9 do
        begin
         pun1^.nodo.inputs[j]:=pun1^.nodo.inputs[j+1];
         pun1^.nodo.inputs[10].id:=0;
        end;
      end;
     end;
    if ((i=11) or (pun1^.nodo.inputs[i].id=0)) then
     begin
      pun1:=pun1^.next;
      i:=1;
     end;
  end;
end;
```

```
procedure elim_text(var p:cartelptr);
{
  Este procedimiento libera toda la memoria de la lista
  de textos. Es utilizada para la re-inicializacion
  del editor, para asegurar que no queda memoria desperdiciada.
}

var p1:cartelptr;
begin
 while p^.next <> nil do
  begin
   p1:=p^.next;
   p^.next:=p1^.next;
   dispose(p1);
  end;
end;

procedure eliminar(var base:archptr);
{
  Este procedimiento libera toda la memoria de la lista
  apuntada por base. Es utilizada para la re-inicializacion
  del editor, para asegurar que no queda memoria desperdiciada.
}

var p1:archptr;

begin
 while ((base<>nil) and (base^.next <> nil)) do
  begin
   p1:=base^.next;
   base^.next:=p1^.next;
   if (p1^.macro<> nil) then
    begin
     eliminar(p1^.macro);
     dispose(p1^.macro);
    end;
   if p1 <> nil then dispose(p1);
  end;
end;


procedure Macro_ed(var storage,macrolist:archptr;
                   var numm1,numm2:integer);
{ esta rutina es la que realiza la seleccion multiple de nodos }
var
xi,yi,xl,yl,xh,yh,xa,ya:integer;
drawing,present:boolean;
current: archptr;


procedure draw_rec;
begin
 hide_cursor(m_stat,m_par);
 rectangle(xl,yl,xh,yh);
 show_cursor(m_stat,m_par);
end;

procedure scan_group;
```

```
var p: archptr;
esta:boolean;

begin
 p:=storage^.next;
 repeat
 if (normx(p^.nodo.x) > xl+10) and (normx(p^.nodo.x) < xh-10) and
    (normy(p^.nodo.y) > yl+10) and (normy(p^.nodo.y) < yh-10) then
  begin
   current:=macrolist;
   esta:=false;
   while (current <> nil) do
    begin
     current:=current^.next;
     if current^.nodo.id = p^.nodo.id then esta := true;
    end;


   if not esta then
    begin
     new(current);
     current^.next:=macrolist^.next;
     macrolist^.next:=current;
     current^.nodo:=p^.nodo;
     current^.macro:=nil;
     setwritemode(copyput);
     setlinestyle(solidln,0,normwidth);
     rectangle(normx(p^.nodo.x)-12,normy(p^.nodo.y)-12,
               normx(p^.nodo.x)+12,normy(p^.nodo.y)+12);
     setwritemode(xorput);
     setlinestyle(dottedln,0,normwidth);
    end;
   end;

  p:=p^.next;
 until (p=nil);
end;

begin { Macro_ed }
 set_hand_cursor;
 show_cursor(m_stat,m_par);
 setwritemode(xorput);
 setcolor(white);
 setlinestyle(dottedln,0,normwidth);
 drawing:=false;
 get_pos (m_stat,m_par);
 repeat
  get_pos(m_stat,m_par);
  if m_stat.tecla1 and not drawing then
   begin
    drawing:=true;
    xi:= m_stat.xnuevo;xa:=xi;xl:=xi;xh:=xi;
    yi:= m_stat.ynuevo;ya:=yi;yl:=yi;yh:=yi;
    draw_rec
   end;

 while drawing and m_stat.tecla1 do
  begin
   get_pos (m_stat,m_par);
```

```
    if (xa <> m_stat.xnuevo) or (ya <> m_stat.ynuevo) then
     begin
      draw_rec;
      if m_stat.xnuevo < xi then
       begin
        xl:=m_stat.xnuevo;
        xh:=xi;
       end
      else
       begin
        xl:=xi;
        xh:=m_stat.xnuevo;
       end;

      if m_stat.ynuevo < yi then
       begin
        yl:=m_stat.ynuevo;
        yh:=yi;
       end
      else
       begin
        yl:=yi;
        yh:=m_stat.ynuevo;
       end;

      xa:=m_stat.xnuevo;
      ya:=m_stat.ynuevo;
      draw_rec;
      show_cursor(m_stat,m_par);
      get_pos(m_stat,m_par);
     end;
    end;
   if drawing then
    begin
     draw_rec;
     scan_group;
    end;
   drawing:=false;
   get_pos(m_stat,m_par);
 until  not m_stat.tecla1;
 setwritemode(copyput);
 setlinestyle(solidln,0,normwidth);
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 show_cursor(m_stat,m_par);
 if macrolist^.next<> nil then
  begin
   numm1:=5;
   numm2:=7;
  end
 else
  begin
   numm1:=4;
   numm2:=5;
  end;
end;

procedure move_nodes( storage,macrolist:archptr;
                      tipo:tipografo;var grouped:boolean);
{mueve los nodos seleccionados}
```

```
var is_in_macro:boolean;
    p,p1,p2:archptr;
    x,y,deltax,deltay,x1,x2,y1,y2,nox,noy:integer;

begin
 setcolor(white);
 setlinestyle(dottedln,0,normwidth); {linea de puntos}
 setwritemode(xorput);               {modo de escritura xor}

{ noy y nox son los limites de la zona de iconos: acceso prohibido }
 noy:=34;
 if tipo=algoritmo then nox:=98
                   else nox:=162;

{ limites para el movimiento del cursor }
 m_stat.xmin:=14;
 m_stat.xmax:=getmaxx-14;
 m_stat.ymin:=28;
 m_stat.ymax:=getmaxy-33;
 set_x(m_stat,m_par);
 set_y(m_stat,m_par);


 show_cursor(m_stat,m_par);
 get_pos(m_stat,m_par);
 P:=macrolist;
 kscan(m_stat.xnuevo,m_stat.ynuevo,p,is_in_macro);
 if is_in_macro then
  begin
   m_stat.yant:=normy(p^.nodo.y);
   m_stat.xant:=normx(p^.nodo.x);
   set_cursor(m_stat,m_par); { cursor como mano abierta }
   set_open_hand;
   x:=normx(p^.nodo.x);
   y:=normy(p^.nodo.y);
   if grouped then
    begin          { si seleccion multiple busco limites }
     search_limits(macrolist,x1,y1,x2,y2);
     x2:=abs(x2-normx(p^.nodo.x))+16;
     x1:=abs(normx(p^.nodo.x)-x1)+16;
     y1:=abs(normy(p^.nodo.y)-y1)+16;
     y2:=abs(y2-normy(p^.nodo.y))+16;
     rectangle(x-x1+2,y-y1+2,x+x2-2,y+y2-2);
    end;
   repeat                              { lazo principal }
    get_pos(m_stat,m_par);

    if (m_stat.xnuevo <> x) or (m_stat.ynuevo<>y) then
     begin

      hide_cursor(m_stat,m_par);  { movemos seleccion }
      if grouped then rectangle(x-x1+2,y-y1+2,x+x2-2,y+y2-2);
      rectangle(x-12,y-12,x+12,y+12);
      x:=m_stat.xnuevo;
      y:=m_stat.ynuevo;
      if grouped then rectangle(x-x1+2,y-y1+2,x+x2-2,y+y2-2);
      rectangle(x-12,y-12,x+12,y+12);
      show_cursor(m_stat,m_par);
```

```
      end;
    until not m_stat.tecla1;

   { ahora modifico las coordenadas de los nodos movidos}
   { en el buffer del editor y en el de macros }
    deltax:=m_stat.xnuevo-normx(p^.nodo.x);
    deltay:=m_stat.ynuevo-normy(p^.nodo.y);
    if not grouped then
     begin
      p^.nodo.x:=realx(m_stat.xnuevo);
      P^.nodo.y:=realy(m_stat.ynuevo);
      p1:=storage;
      repeat p1:=p1^.next until p1^.nodo.id=p^.nodo.id;
      p1^.nodo.x:=p1^.nodo.x+deltax;
      p1^.nodo.y:=p1^.nodo.y+deltay;
     end
    else
     begin
      p1:=macrolist^.next;
      while p1<>nil do
       begin
        p1^.nodo.x:=p1^.nodo.x+deltax;
        p1^.nodo.y:=p1^.nodo.y+deltay;
        p2:=storage;
        repeat p2:=p2^.next until p2^.nodo.id=p1^.nodo.id;
        p2^.nodo.x:=p1^.nodo.x;
        p2^.nodo.y:=p1^.nodo.y;
        p1:=p1^.next;
       end;
     end;
    setlinestyle(solidln,0,normwidth);
    setwritemode(copyput);
    grouped:=false;
    kshow(storage,tipo);       { muestro todo nuevamente }
    show_macro(macrolist,grouped);
    initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
    show_cursor(m_stat,m_par);
   end;
  setlinestyle(solidln,0,normwidth);
  setwritemode(copyput);              { good bye, darling }

  end;
end. { unit }
```

```
unit dosfun;
{  esta  unidad  provee  las  funciones  estandar  del  sistema
    operativo.
}
interface
uses crt,dos,graph,defs2,printer,micky;


procedure dos_window;
procedure direc;


implementation

const
 screen=true;
 impresora=false;

procedure espera;

{ vacia el buffer de teclado y espera un caracter }

var regs:registers;
    cursor:char;


begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
 cursor:=readkey;
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

procedure empty;

{ vacia el buffer de teclado }

var regs:registers;

begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

procedure direc;

{ directorio de archivos de disco }

var mask,direcname:string;
    columna,fila,width:integer;
    dirinfo:searchrec;
    cursor:char;
label 1;

begin
```

```
 setcolor(white);
 width:=8;
 writeln;
 write('M scara (defecto:*.*): ');
 empty;
 readln(mask);
 writeln;
 if mask='' then mask:='*.*';
 setactivepage(1);
 setviewport(0,0,maxx,maxy,clipon);
 clearviewport;
 setvisualpage(1);
 moveto(1,1);
 getdir(0,direcname);
 outtext('Directorio de trabajo: '+direcname + '. M scara: ' +mask);
 findfirst(mask,anyfile,dirinfo);
 columna:=1;fila:=3*width;
 if doserror <> 0 then outtextxy(2,20,'No se encontraron archivos.');
 while doserror = 0 do
  begin
   moveto(columna,fila);
   outtext(dirinfo.name);
   findnext(dirinfo);
   columna:=columna+14*width;
   if columna >70*width then
    begin
     columna:=1;
     fila:=fila+width;
     if fila > 40*width then
      begin
       moveto(1,42*width);
       outtext('...Oprima cualquier tecla para continuar...');
       empty;
       cursor:=readkey;
       if cursor=#27 then goto 1;
       clearviewport;
       fila:=1;
      end;
     end;
    end;
1: outtextxy(1,42*width,'Oprima cualquier tecla
                         para volver al editor...');
 espera;
 clearviewport;
 setactivepage(0);
 setvisualpage(0);
 empty;
end;

procedure typefile(unidad:boolean);

{ impresion de un archivo. Unidad: true -> salida por pantalla
                                   false -> salida por impresora
}

var mask,direcname,line:string;
    f: text;cursor:char;
    index,width:integer;
label 1;
```

```
begin
{$I-}
 width:=8;
 setcolor(white);
 writeln;
 write('Archivo: ');
 empty;
 readln(mask);
 assign(f,mask);
 reset(f);
 if ioresult <> 0 then
  begin
   writeln;
   write('No se encontraron archivos. Oprima cualquier tecla...');
   espera;
   writeln;
  end
 else
  begin
   if unidad then
    begin
     setactivepage(1);
     setviewport(0,0,maxx,maxy,clipon);
     clearviewport;
     setvisualpage(1);
     moveto(0,0);
    end;
    moveto(0,0);
    index:=1;
    repeat
     readln(f,line);
     if unidad then outtext(line)     {salida por pantalla}
     else writeln(lst,line);          {salida por impresora}
     index:=index+width;
     moveto(0,index);
     if (unidad and (index>=40*width)) then
      begin
       moveto(0,42*width);
       outtext('Oprima cualquier tecla...');
       empty;
       cursor:=readkey;
       if cursor=#27 then goto 1;
       clearviewport;
       index:=1;
       moveto(0,0);
      end;
    until eof(f);
1:  close(f);
   end;
 if ((unidad) and (ioresult=0)) then
  begin
   moveto(0,42*width);
   outtext('Oprima cualquier tecla para volver al editor...');
   espera;
   setactivepage(0);
   setvisualpage(0);
  end;
 writeln;
```

```
end;

procedure ren_file;

{ renombrar archivo }

var oldnam,newnam:string;
    f:file;

begin
 writeln;
 write('Archivo a renombrar: ');
 empty;
 readln(oldnam);
 write('Nuevo nombre: ');
 empty;
 readln(newnam);
 assign(f,oldnam);
 reset(f);
 if ioresult = 0 then
  begin
   close(f);
   rename(f,newnam);
  end
 else
  begin
   writeln;
   write('Imposible renombrar archivo. Oprima cualquier tecla...');
   espera;
  end;
 writeln;
 empty;
end;

procedure era_file;

{ borrar archivo }

var filnam:string;
    f:file;

label 1;

begin
 writeln;
 write('Archivo a borrar: ');
 empty;
 readln(filnam);
 if filnam='' then goto 1;
 assign(f,filnam);
 reset(f);
 if ioresult = 0 then
  begin
   close(f);
   erase(f);
  end
 else
  begin
   writeln;
```

```
    write('Imposible borrar archivo. Oprima cualquier tecla...');
    espera;
   end;
1: writeln;
 empty;
end;

procedure copy_file;

{ copiar archivo }

var fronam,tonam:string;
    fin,fout:file;
    numread,numwritten:word;
    buf:array[1..2040] of char;

label 1;

begin
 writeln;
 write('Archivo fuente: ');
 empty;
 readln(fronam);
 if fronam='' then goto 1;
 write('Archivo destino: ');
 empty;
 readln(tonam);
 if tonam='' then exit;
 assign(fin,fronam);
 reset (fin,1);
 if ioresult<>0 then
  begin
   write('Archivo de entrada no encontrado. Oprima cualquier tecla...
   espera;
   writeln;
   exit;
  end;
 assign(fout,tonam);
 rewrite(fout,1);
 writeln;
 write('Copying ',filesize(fin),' bytes...');
 repeat
  blockread(fin,buf,sizeof(buf),numread);
  blockwrite(fout,buf,numread,numwritten);
  until (numread=0) or (numwritten <> numread);
  close(fin);
  close(fout);
1: writeln
end;



procedure handle_dir(a:integer);

{ - cambiar directorio actual
  - crear nuevo directorio
  - eliminar directorio
}
```

```
var dirnam,curdir:string;
    cursor:char;
begin
{$I-}
 empty;
 getdir(0,curdir);
 write('Directorio actual : ',curdir,'. Directorio ?: ');
 empty;
 readln(dirnam);
 case a of
      1: begin
          chdir(dirnam);
          if ioresult <> 0 then
           begin
            writeln;
            write('ERROR:El directorio especificado no existe...');
            espera;
           end;
         end;
      2: begin
          rmdir(dirnam);
          if ioresult <> 0 then
           begin
            writeln;
            write ('ERROR:imposible eliminar directorio...');
            espera;
           end;
         end;
      3: begin
          mkdir(dirnam);
          if ioresult <> 0 then
           begin
            writeln;
            write('ERROR:imposible crear directorio...');
            espera;
           end;
         end;
    end;
 writeln;
 empty;
end;

procedure  set_attr;

{ definir atributos de un archivo-solo lectura, oculto, normal }

var filnam:string;
    f:file;a:char;

begin
 writeln;
 write('Archivo: ');
 empty;
 readln(filnam);
 assign(f,filnam);
 write('(R)eadonly,(H)idden,(A)rchivo_normal?: ');
 empty;
 a:=readkey;
```

```
 write(upcase(a));
 espera;
 case upcase(a) of
      'R': setfattr(f,readonly);
      'H': setfattr(f,hidden);
      'A': setfattr(f,archive);
      end;
 empty;
 writeln;
end;



procedure dos_window;
{************************DOS_WINDOW*****************************
 Este procedimiento crea una ventana de seleccion en pantalla
 para acceder a las funciones standard del sistema operativo. Es
 autocontenida, de forma que puede ser utilizada por cualquier
 programa.
}
type disp=array[1..10] of string;
     event = (none, change, select, quit, out);
var sizet:integer;
    p:pointer;
    opt,opt1,lineas,xmin, xmax, ymin,ybeg,
    ymax, y, i,alto,nn : integer;
    menu:disp;
    action: event;

function dentro:boolean;
{true si dentro de la ventana}
begin
 get_pos(m_stat,m_par);
 dentro:=((m_stat.xnuevo<=xmax) and (m_stat.xnuevo>=xmin) and
          (m_stat.ynuevo<=ymax) and (m_stat.ynuevo>=ybeg))
end;

begin
 lineas:=10;
 menu[1]:=' ~ Directory.';
 menu[2]:=' ~ Type file.';
 menu[3]:=' ~ Print (ASCII) file.';
 menu[4]:=' ~ Copy file.';
 menu[5]:=' ~ Rename file.';
 menu[6]:=' ~ Delete file.';
 menu[7]:=' ~ Change Directory.';
 menu[8]:=' ~ Remove Directory.';
 menu[9]:=' ~ Make Directory.';
 menu[10]:=' ~ Set File Attributes.';

 setfillstyle(solidfill,lightblue);

 setcolor(blue);
 if graphdriver = EGA then alto := 14
    else alto :=8;

 xmin:=(maxx div 2)-95;
 xmax:=xmin+190;
 ymin:=44;
```

```
ymax:=55+alto*(lineas+1);

sizet:=imagesize (xmin,ymin,xmax,ymax);
getmem(p,sizet);
getimage(xmin,ymin,xmax,ymax,p^);
rectangle(xmin,ymin,xmax,ymax);
floodfill(xmin+1,ymin+1,blue);
setcolor(white);
rectangle(xmin+2,ymin+2,xmax-2,ymax-2);
setfillstyle(solidfill,white);
bar(xmin+2,ymin+2,xmax-2,ymin+12);
settextjustify(centertext,centertext);
setcolor(red);
outtextxy(xmin+95,ymin+8,'BOLAS - DOS Window');
setcolor(white);
settextjustify(lefttext,toptext);
ybeg:=ymin+12;

for i:=1 to lineas do outtextxy(xmin,ybeg+i*alto,menu[i]);
opt:=1;
action:=none;
show_cursor(m_stat,m_par);
repeat
 if action=change then
  begin
   hide_cursor(m_stat,m_par);
   moveto (xmin,ybeg+opt*alto);
   setcolor(white);
   outtextxy(xmin+3,ybeg+opt*alto,' [[[[[[[[[[[[[[[[[[[[');
   setcolor(red);
   for i:=1 to length(menu[opt]) do outtext(upcase(menu[opt,i]));
   show_cursor(m_stat,m_par);
  end;

action := none;
repeat
 if dentro then
  begin
   opt1:=(m_stat.ynuevo - ymin) div alto; {indice}
   if ((opt1 >0) and (opt1 <=lineas)) then
    begin
     if (m_stat.tecla1) then  action := select  {seleccion}
     else if (opt1<>opt) then action := change;{cambio de renglon}
    end;
  end
 else if m_stat.tecla1 then action := quit  {abortar}
      else
       begin
        action:=out;  {deshabilitar seleccion mientras mouse fuera }
       end;
 until (action <> none);   {hasta que pase algo}
 hide_cursor(m_stat,m_par);{reescribo renglon en modo normal}
 moveto (xmin,ybeg+opt*alto);
 setcolor(lightblue);
 outtextxy(xmin+3,ybeg+opt*alto,'[[[[[[[[[[[[[[[[[[[[');
 setcolor(white);
 outtext(menu[opt]);
 show_cursor(m_stat,m_par);
 if action <> out then opt:=opt1
```

```
   else
    begin
     repeat until ((dentro) or (m_stat.tecla1));  {espero hasta entra
     action:=change;
    end;
   get_pos(m_stat,m_par);
   if  (action = quit) then opt:=0;
   if action = select then
    begin
     hide_cursor(m_stat,m_par);
     case opt of
          1:direc;                              {directorio}
          2:typefile(screen);                   {listar archivo por panta
          3:typefile(impresora);                {listar archivo por impre
          4:copy_file;                          {copiar archivo}
          5:ren_file;                           {Renombrar archivo}
          6:era_file;                           {borrar archivo}
          7:handle_dir(1);                      {cambiar directorio}
          8:handle_dir(2);                      {remover directorio}
          9:handle_dir(3);                      {crear directorio}
         10:set_attr;                           {atributos del fichero}
      end;{case}
     show_cursor(m_stat,m_par);
    end;
  empty;
 until action=quit ;
 empty;
 setcolor(white);
 setfillstyle(solidfill,white);
 hide_cursor(m_stat,m_par);
 putimage(xmin,ymin,p^,normalput);
 freemem(p,sizet);
 initm(m_stat,m_par,ok,m_stat.xnuevo, m_stat.ynuevo+30);
end;

end.
```

```
unit micky;

{Esta unidad contiene las rutinas para inicializacion y uso del mouse

interface
uses crt,dos,graph;

type

status = record                { Estado del mouse y parametros }
 cursor :boolean;
 xant,yant,xnuevo,ynuevo,xmin,xmax,ymin,ymax :word;
 teclal,teclar :boolean;
 xratio,yratio:word;
 end;


var

ok,is_mouse : boolean;
m_stat : status;
m_par : registers;          { registers esta predefinido en TPv4.00}

procedure show_cursor (var m_stat: status ; var  m_par : registers);
procedure hide_cursor (var m_stat: status ; var m_par: registers);
procedure get_pos (var m_stat:status ; var m_par:registers);
procedure set_cursor (var m_stat:status ;var m_par:registers);
procedure set_x (var m_stat :status ; var m_par: registers);
procedure set_y (var m_stat :status ; var m_par: registers);
procedure set_ratio (var m_status :status ; var m_par : registers);
procedure initm (var m_status : status ; var m_par : registers;
                 var ok:boolean;x,y:integer);
procedure set_style;
procedure set_erase_cursor;
procedure set_screen_for_circles;
procedure set_hand_cursor;
procedure set_open_hand;
procedure set_text_cursor;

implementation

procedure show_cursor (var m_stat: status ; var  m_par : registers);

{hace visible el cursor}

begin
 m_stat.cursor := true;
 m_par.ax := 1;
 intr (51,m_par);
end;

procedure hide_cursor (var m_stat: status ; var m_par: registers);

{hace invisible el cursor}

begin
 m_stat.cursor := false;
 m_par.ax := 2;
 intr (51,m_par);
```

```
end;

procedure get_pos (var m_stat:status ; var m_par:registers);

{lee la posicion del cursor y el estado de los botones }

begin
 m_par.ax := 3;
 intr (51, m_par);
 m_stat.teclal := (m_par.bx and ( 1 shl 0))=1 shl 0;
 m_stat.teclar := (m_par.bx and ( 1 shl 1))=1 shl 1;
 m_stat.xnuevo := m_par.cx;
 m_stat.ynuevo := m_par.dx;
end;

procedure set_cursor (var m_stat:status ;var m_par:registers);

{ubica el cursor}

begin
 m_par.ax := 4;
 m_par.cx := m_stat.xant;
 m_par.dx := m_stat.yant;
 intr(51,m_par);
end;

procedure set_x (var m_stat :status ; var m_par: registers);

{valores maximos y minimos posibles del cursor segun x}

begin
 m_par.ax := 7;
 m_par.cx := m_stat.xmin;
 m_par.dx := m_stat.xmax;
 intr (51,m_par);
end;

procedure set_y (var m_stat :status ; var m_par: registers);

{valores maximos y minimos posibles del cursor segun y}

begin
 m_par.ax := 8;
 m_par.cx := m_stat.ymin;
 m_par.dx := m_stat.ymax;
 intr (51,m_par);
end;

procedure set_ratio (var m_status :status ; var m_par : registers);

{velocidad del movimiento del cursor}

begin
 m_par.ax := 15;
 m_par.cx := m_stat.xratio;
 m_par.dx := m_stat.yratio;
 intr (51,m_par);
end;
```

```
procedure initm (var m_status : status ; var m_par : registers;
                 var ok:boolean;x,y:integer);

{inicializa el mouse:
   * verifica que el driver este cargado. Si no lo esta, ok:false.
   * posiciona el cursor en x,y (cursor estandar modo grafico.
   * inicializa valores maximos y minimos segun x e y.
 recordar que el cursor permanece invisible (hacer show_cursor) }

var
m_segm, m_offs :integer;
begin
 ok := false;
 with m_status do
  begin
   m_segm := 256 * mem [0:51 * 4 + 3 ] + mem [0:51 * 4 + 2];
   m_offs := 256 * mem [0:51 * 4 + 1 ] + mem [0:51 * 4 + 2];
   if ((m_segm <> 0) and (m_offs <> 0)) then
   begin
    ok := true;
    m_par.ax := 0;
    intr(51,m_par);
    if m_par.ax = 0 then ok := false;
    cursor := false;
    xmin := 0;
    xmax := getmaxx-8;
    set_x (m_status, m_par);
    ymin := 0;
    ymax := getmaxy-18;
    set_y (m_status, m_par);
    xant:= x;
    yant := y;
    xnuevo := 0;
    ynuevo := 0;
    set_cursor(m_status, m_par);
    tecla1 := false;
    teclar := false;
    xratio :=10;
    yratio := 10;
    set_ratio (m_status, m_par);
   end;
 end;
end;


procedure set_style;

{ cambia a cursor en cruz }

var style: array [0..31] of word; i:integer;

begin
 for  i:=0 to 15 do
  begin
   style[i]:=$fc3f;
   style[i+16]:=$0180;
  end;
 style[16]:=$0000;
 style[31]:=$0000;
```

```
 style[23]:=$7ffe;
 style[24]:=$7ffe;
 style[6]:=0;
 style[7]:=0;
 style[8]:=0;
 style[9]:=0;
 m_par.ax:=9;
 m_par.bx:=8;
 m_par.cx:=8;
 m_par.dx:=ofs(style);
 m_par.es:=seg(style);
 intr(51,m_par);
end;

procedure set_erase_cursor;

{ cambia cursor a goma de borrar }

var style: array [0..31] of word; i:integer;

begin
 for  i:=0 to 7 do
  begin
   style[i]:=$0000;
   style[i+8]:=0000;
   style[i+16]:=$4002;
   style[i+24]:=$7ffe;
  end;
 style[16]:=0;
 style[17]:=$7ffe;
 style[31]:=0;
 m_par.ax:=9;
 m_par.bx:=8;
 m_par.cx:=8;
 m_par.dx:=ofs(style);
 m_par.es:=seg(style);
 intr(51,m_par);
end;

procedure set_screen_for_circles;

{ programa el cursor para que no haya problemas al editar poligs }

begin
 m_stat.xmin:=0;
 m_stat.xmax:=getmaxx;
 m_stat.ymin:=0;
 m_stat.ymax:=getmaxy;
 m_par.ax := 0;
 intr(51,m_par);
 set_y(m_stat,m_par);
 set_x(m_stat,m_par);
 set_style;
 m_stat.xant:=m_stat.xnuevo;
 m_stat.yant:=m_stat.ynuevo;
 set_cursor(m_stat,m_par);
 show_cursor(m_stat,m_par);
end;
```

```
procedure set_hand_cursor;

{ cambia cursor a mano apuntando con el indice }

var style: array [0..31] of word; i:integer;

begin
 for  i:=0 to 15 do style[i]:=$0000;
 style[0 ]:=$0fff;
 style[1 ]:=$07c7;
 style[2 ]:=$0383;
 style[3 ]:=$8103;
 style[4 ]:=$c007;
 style[5 ]:=$e007;
 style[6 ]:=$c007;
 style[7 ]:=$c003;
 style[8 ]:=$8001;
 style[9 ]:=$0001;
 style[10]:=$0001;
 style[11]:=$0001;
 style[12]:=$0000;
 style[13]:=$8000;
 style[14]:=$c003;
 style[15]:=$e00f;
 style[16]:=$E000;
 style[17]:=$9000;
 style[18]:=$4838;
 style[19]:=$2448;
 style[20]:=$1290;
 style[21]:=$0990;
 style[22]:=$1C90;
 style[23]:=$1208;
 style[24]:=$3904;
 style[25]:=$2484;
 style[26]:=$7384;
 style[27]:=$4904;
 style[28]:=$2702;
 style[29]:=$1C01;
 style[30]:=$0f00;
 style[31]:=$0080;
 m_par.ax:=9;
 m_par.bx:=0;
 m_par.cx:=0;
 m_par.dx:=ofs(style);
 m_par.es:=seg(style);
 intr(51,m_par);
end;

procedure set_open_hand;

{ cambia cursor a mano  abierta }

var style: array [0..31] of word; i:integer;

begin
 for  i:=0 to 15 do style[i]:=$0000;
 style[0 ]:=$FC1F;
 style[1 ]:=$E00F;
```

```
 style[2 ]:=$C007;
 style[3 ]:=$C003;
 style[4 ]:=$C000;
 style[5 ]:=$C000;
 style[6 ]:=$0000;
 style[7 ]:=$0000;
 style[8 ]:=$0000;
 style[9 ]:=$0000;
 style[10]:=$0000;
 style[11]:=$0000;
 style[12]:=$0000;
 style[13]:=$0000;
 style[14]:=$0000;
 style[15]:=$8001;
 style[16]:=$0000;
 style[17]:=$01C0;
 style[18]:=$0E60;
 style[19]:=$1250;
 style[20]:=$124C;
 style[21]:=$124A;
 style[22]:=$124A;
 style[23]:=$F24A;
 style[24]:=$924A;
 style[25]:=$D00A;
 style[26]:=$4002;
 style[27]:=$4002;
 style[28]:=$4002;
 style[29]:=$6006;
 style[30]:=$300C;
 style[31]:=$0810;
 m_par.ax:=9;
 m_par.bx:=8;
 m_par.cx:=8;
 m_par.dx:=ofs(style);
 m_par.es:=seg(style);
 intr(51,m_par);
end;

procedure set_text_cursor;

{ cambia cursor a mano  abierta }

var style: array [0..31] of word; i:integer;

begin
for  i:=0 to 15 do style[i]:=$0000;
 style[0 ]:=$07C1;
 style[1 ]:=$0381;
 style[2 ]:=$F11F;
 style[3 ]:=$F83F;
 style[4 ]:=$FC7F;
 style[5 ]:=$FC7F;
 style[6 ]:=$FC7F;
 style[7 ]:=$FC7F;
 style[8 ]:=$FC7F;
 style[9 ]:=$FC7F;
 style[10]:=$FC7F;
 style[11]:=$FC7F;
 style[12]:=$F83F;
```

```
  style[13]:=$F11F;
  style[14]:=$0381;
  style[15]:=$07C1;
  style[16]:=$F01E;
  style[17]:=$0820;
  style[18]:=$0440;
  style[19]:=$0280;
  style[20]:=$0100;
  style[21]:=$0100;
  style[22]:=$0100;
  style[23]:=$0100;
  style[24]:=$0100;
  style[25]:=$0100;
  style[26]:=$0100;
  style[27]:=$0100;
  style[28]:=$0280;
  style[29]:=$0440;
  style[30]:=$0820;
  style[31]:=$F01E;
  m_par.ax:=9;
  m_par.bx:=8;
  m_par.cx:=4;
  m_par.dx:=ofs(style);
  m_par.es:=seg(style);
  intr(51,m_par);
end;


end. {unit}
```

```
unit bolas4;
{*******Atencion: version para turboPascal v5.00*******}
{modificaciones del 31/1/89 para simular arquitecturas}
{modificaciones del 14/2/89 para simular ruteo de mensajes}
{modificaciones del 2/5/89  para cambiar estructuras de
 datos de los editores (ahora reunidos en uno solo).
}
{modificaciones del 17/5/89 para mejorar tratamiento de instancias}
{modificaciones del 10/8/89 para  macros y archivos de trace
 Eliminada la recursion de las rutinas de busqueda y borrado }

Interface

uses crt,dos,graph,defs2,utiles,micky;

var alg_scalex,alg_scaley,arq_scalex,arq_scaley:real;
    alg_menorx,alg_menory,arq_menorx,arq_menory:integer;

Procedure Matching_unit (id_fuente,id_destino:integer;inst:stp_ptr);
Procedure Execute;
procedure delet1 (ind0,ind1 : arcpntr);
procedure delete_r(base,p1:r_ptr);
procedure delete_l(base,p1:lnk_ptr);
procedure h_sched(var ok:boolean);
procedure scales(p:archptr; var scalex,scaley:real;
                 var menorx,menory:integer);
procedure normalize(p:archptr;scalex,scaley:real;
                    menorx,menory:integer);
procedure denormalize(p:archptr;scalex,scaley:real;
                      menorx,menory:integer);
procedure scan (x,y:integer;var p:archptr;var is_in_circle:boolean);
procedure xshow(p:archptr;tipo:tipografo;
               offsetx,offsety,red_factx,red_facty:integer);
function smx(x,menorx:integer;scalex:real):integer;
function smy(y,menory:integer;scaley:real):integer;
function rmx(x,menorx:integer;scalex:real):integer;
function rmy(y,menory:integer;scaley:real):integer;

Implementation



var

exist,present,enabled,bien : boolean;
sid : string[3];
stop : char;
evento: eventclass;

{*************************** SCALES ****************************}
procedure scales(p:archptr;var scalex,scaley:real;
                 var menorx,menory:integer);
{
  Este procedimiento genera los factores de escala segun
  los dos ejes para que el buffer del edito quepa en una
  sola pantalla. Es utilizada por el simulador y las rutinas
  de ruteo y asignacion estaticas.
}
```

```
var xmax,ymax:integer;
begin
 xmax:=0;
 menorx:=maxint;
 ymax:=0;
 menory:=maxint;
 while p <> nil do
  begin
   if p^.nodo.x > xmax then xmax:=p^.nodo.x;
   if p^.nodo.y > ymax then ymax:=p^.nodo.y;
   if p^.nodo.x < menorx then menorx:=p^.nodo.x;
   if p^.nodo.y < menory then menory:=p^.nodo.y;
   p:=p^.next;
  end;
 if xmax <> menorx then scalex:=(maxx-60) / (xmax-menorx)
                   else scalex:=1;

 if ymax <> menory then scaley:=(maxy-84) / (ymax-menory)
                   else scaley:=1;
end;

function smx(x,menorx:integer;scalex:real):integer;
begin
 smx:=30+trunc((x-menorx) * scalex);
end;

function smy(y,menory:integer;scaley:real):integer;
begin
 smy:=40+trunc((y-menory) * scaley);
end;

function rmx(x,menorx:integer;scalex:real):integer;
begin
 rmx:=trunc ((x-30)/scalex)+menorx;
end;

function rmy(y,menory:integer;scaley:real):integer;
begin
 rmy:=trunc ((y-40)/scaley)+menory;
end;

procedure normalize(p:archptr;scalex,scaley:real;menorx,menory:intege
{ reduce el grafico para que entre en una sola pantalla }
begin
 while p<> nil do
  begin
   p^.nodo.x:=smx(p^.nodo.x,menorx,scalex);
   p^.nodo.y:=smy(p^.nodo.y,menory,scaley);
   p:=p^.next;
  end;
end;

procedure denormalize(p:archptr;scalex,scaley:real;
                      menorx,menory:integer);
{ recupera el grafico original }
begin
 while p<> nil do
  begin
   p^.nodo.x:=rmx(p^.nodo.x,menorx,scalex);
```

```
    p^.nodo.y:=rmy(p^.nodo.y,menory,scaley);
    p:=p^.next;
   end;
end;

procedure espera;
{ vacia el buffer de teclado y espera un caracter }

var regs:registers;
    cursor:char;

begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
 cursor:=readkey;
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

procedure empty;
{ vacia el buffer de caracteres }

var regs:registers;
begin
 regs.ax:=$0c06;
 regs.dx:=$00ff;
 intr($21,regs);
end;

{
*****************************************************************
*                                                               *
*                      ALGUNAS RUTINAS GRAFICAS                 *
*                                                               *
*****************************************************************
}

procedure scan (x,y:integer;var p:archptr;var is_in_circle:boolean);
{*****************************SCAN*****************************
 verifica si un punto x,y de la pantalla pertenece a un elemento.
 si pertenece, p apunta al elemento, is_in_circle:true.
 si no pertenece, is_in_circle:false }
begin
 is_in_circle:=false;
 repeat
  p:=p^.next;
  with p^ do
   begin
    if (x>nodo.x-8) and (x<nodo.x+8) and (y<nodo.y+8)
       and (y>nodo.y-8) then  is_in_circle := true;
   end;
 until (p=nil) or (is_in_circle);
end;

{********************** XSHOW ******************************}
procedure xshow(p:archptr;tipo:tipografo;
                offsetx,offsety,red_factx,red_facty:integer);
```

```
{
 dibuja los contenidos del buffer del editor para el simulador.
 parametros de entrada:
 p: cabeza de la lista a dibujar;
 tipo: tipo de grafo-algoritmo o arquitectura.
 offsetx: posicion segun x a partir de donde se debe dibujar.
 offsety: idem segun y.
 red_factx: factor de reduccion de escala segun x.
 red_facty: idem segun y.
 La rutina no dibuja el marco de la pantalla ni limpia
 la misma antes de dibujar. Es utilizada por el simulador,
 el router manual, y la rutina de asignacion estatica.
 El editor tiene otra rutina (diferente).
}

var i,j,n:integer;
    x1,y1,x2,y2:integer;
    p1,p2:archptr;
    sidmx:string;

begin

 hide_cursor(m_stat,m_par);
 settextjustify(centertext,centertext);
 settextstyle(smallfont,horizdir,4);
 setfillstyle(solidfill,lightblue);

 P2:=p^.next;
 while  p2<> nil do
  begin
   if p2^.nodo.id <> 0 then
    begin
     j:=1;
     repeat
      if P2^.nodo.outputs[j].id <>0 then
       begin
        n:=p2^.nodo.outputs[j].id;
        p1:=p;
        repeat
         p1:=p1^.next;
        until (p1^.nodo.id=n)  ;
        setcolor(lightblue);
        x1:=offsetx + (p2^.nodo.x div red_factx);
        y1:=offsety + (p2^.nodo.y div red_facty);
        x2:=offsetx + (p1^.nodo.x div red_factx);
        y2:=offsety + (p1^.nodo.y div red_facty);
        kpunta(x1,y1,x2,y2);
       end;
      j:=j+1;
     until j=10;
    end;
   p2:=p2^.next;
  end;

 p2:=p^.next;
 setcolor(white);
 while p2<> nil do
  begin
```

```
    x1:=offsetx + (p2^.nodo.x div red_factx);
    y1:=offsety + (p2^.nodo.y div red_facty);
    if tipo= algoritmo then circle(x1,y1,10)
    else if p2^.nodo.class=0 then
     begin
      moveto(x1-10,y1-5);
      lineto(x1- 5,y1-10);
      lineto(x1+ 5,y1-10);
      lineto(x1+10,y1- 5);
      lineto(x1+10,y1+ 5);
      lineto(x1+ 5,y1+10);
      lineto(x1- 5,y1+10);
      lineto(x1-10,y1+ 5);
      lineto(x1-10,y1- 5);
     end
    else if p2^.nodo.class=1 then  rectangle(x1-10,y1-10,x1+10,y1+10)
            else if p2^.nodo.class=2 then
             begin
              moveto(x1-10,y1);
              lineto(x1   ,y1-10);
              lineto(x1+10,y1);
              lineto(x1   ,y1+10);
              lineto(x1-10,y1);
             end;
    floodfill(x1,y1,white);
    str(p2^.nodo.id,sidmx);
    outtextxy(x1,y1,sidmx);
    p2:=p2^.next;
   end;
 settextstyle(defaultfont,horizdir,1);
 settextjustify(lefttext,toptext);
 show_cursor(m_stat,m_par);
end;{xshow}

procedure token (id1,id2:integer;state:boolean; node:archptr);
{*****************************TOKEN*********************************
   rutina grafica para poner-sacar tokens de los arcos.
   id1,id2 son los nodos principio-final respectivamente del arco.
   state : true: prender el arco. False: apagarlo
}

var
xx,yy,i,color: integer;
ax,ay,bx,by : integer;
p:archptr;
fin1,fin2:boolean;

begin
{buscamos las coordenadas de los dos nodos extremos (id1,id2)}
 p:=node^.next;
 fin1 := false;
 fin2 := false;
 repeat
  if id1 = p^.nodo.id then
   begin
    ax := p^.nodo.x; {a guarda las coord de uno de los nodos}
    ay := p^.nodo.y;
    fin1 := true;
   end;
```

```
  if id2 = p^.nodo.id then
   begin
    bx := p^.nodo.x; {y b del otro}
    by := p^.nodo.y;
    fin2 := true;
   end;
  p := p^.next;
 until ((fin1 and fin2) or (p=nil));
 if (fin1 and fin2) then
 begin
   xx:=ax + (bx-ax) div 2; { el token se indica en la mitad del arco
   yy:=ay + (by-ay) div 2;
   setcolor(white);
   circle(xx,yy,5);
   if state then color:=lightgreen  { verde si hay token }
            else color:=lightblue;  { azul si no lo hay }

   setfillstyle(SolidFill,color);
   floodfill(xx,yy,white);
  end;
end;



procedure En_ejecucion(p:pointr);
{*****************************EN_EJECUCION**********************
 Este procedimiento pinta de color magenta el nodo (p) que esta
 actualmente en ejecucion. Dentro del simulador, los nodos
 azules no estan habilitados, los verdes estan habilitados pero
 no estan siendo ejecutados, y los magenta estan en ejecucion
}

var sid:string[3];
begin
 if p^.token.visible then
  begin
   setcolor(white);
   circle(p^.token.x,p^.token.y,10);
   setfillstyle(solidfill,magenta);
   floodfill(p^.token.x+5,p^.token.y+5,white);
   settextstyle(smallfont,horizdir,4);
   settextjustify(centertext,centertext);
   str(p^.token.id,sid);
   outtextxy(p^.token.x,p^.token.y,sid);
   settextstyle(defaultfont,horizdir,1);
   settextjustify(lefttext,toptext);
  end;
end;



procedure delet (ind0,ind1 : pointr);
{
***************************** DELET ******************************
 Borra un elemento de la lista
 Prametros de entrada:

-ind0: puntero al comienzo de la lista.
```

```
-ind1: puntero al item a borrar

PROBADA 22/9/88 OK.
Nota: esta  rutina de borrado, al igual que las de busqueda, son
extremadamente simples, ya que suponen que las listas estan des-
ordenadas. Supongo que si se hiciera algun ordenamiento previo de
las listas, podria utilizarse alguna tecnica de hashing, para ace-
lerar estas operaciones. Hasta el momento no parece necesario.
**************************************************************
}
var p:stp_ptr;
begin
 while ind0^.next <> ind1 do ind0:=ind0^.next;
 ind0^.next := ind1^.next;
{borro la lista de instancias}
 while ind1^.instancia <> nil do
  begin
   p:=ind1^.instancia;
   ind1^.instancia:=p^.next;
   dispose(p);
  end;
 dispose (ind1);
end; { delet }



{
**************************************************************
*                                                            *
*                        SCHEDULING UNIT                     *
*                                                            *
**************************************************************
}

Procedure h_sched(var ok:boolean);
{
****************************** H_SCHED ***************************
Este procedimiento asigna tareas a procesadorores de acuerdo al mapa
especificado en el array asiga. Asiga es inicializado por la rutina
hand_assign del programa principal.
Datos de entrada:
la cola de entrada es la token_queue;
la cola de salida es la exec_queue;
ok:true si todo bien.
**************************************************************
}
var
i,j:integer;
p_in,p_out:pointr;
p_arq:archptr;
exist:boolean;

procedure search4 (var indice:pointr;id:integer;var exist:boolean);
{buscar en la lista apuntada por indice el item asignado al procesado
id}

begin
 exist:=true; { optimismo inicial }
```

```
 repeat
{ si llegue al final de la lista no hay nodos asignados al proc. id }
  if indice=nil then exist := false
{ si el nodo no esta asignado al proc id sigo adelante }
   else if indice^.hard <> id then indice:=indice^.next;
{ si esta asignado al proc id, voy al final de la sublista }
  if indice^.hard = id then
     while ((indice^.next<>nil)and(indice^.next^.hard=id)) do
     indice := indice^.next;
 until (not exist) or (indice^.hard=id);

end; {search4}

begin {h_sched}
{inicializacion}
 p_in:=token_queue^.next; {cola de tareas habilitadas}
 p_out:=exec_queue^.next; {cola de tareas habilitadas y asignadas}
 p_arq:=arq_buf^.next;
 ok:=true;

 while p_in <> nil do
  begin {recorro la token_queue asignando}
   if (modo_de_operacion = 1) then
    begin
     i:=0;
     repeat i:=i+1
     until (asiga[i].task = p_in^.token.id) or (i=maxnode);
     if i=maxnode then ok:=false;
     while ((p_arq^.nodo.id <>  asiga[i].procesador) and (p_arq<>nil)
          do  p_arq:=p_arq^.next;
     if (p_arq = nil)  then ok:=false;
     if ok then
      begin
       p_in^.hard:=p_arq^.nodo.id;
       p_in^.token.pcost:=p_in^.token.pcost * p_arq^.nodo.pcost;
       search4(p_out,p_arq^.nodo.id,exist);
       if not exist then
        begin
         p_out:=exec_queue;
         if animated then En_ejecucion(p_in);
         if trace_on then
          begin
           evento.time:=t_simul;
           evento.class:= beg_ejec;
           evento.info[1]:=p_in^.token.id;
           evento.info[2]:=p_in^.instancia^.id;
           evento.info[3]:=p_in^.hard;
           write(tracfile,evento);
          end;
        end;
      end; {if ok}

    end; {if modo de operacion}

   if modo_de_operacion = 0 then p_out:=exec_queue;

   token_queue^.next:=p_in^.next;  {saco a p_in de la token_queue}
   p_in^.next:=p_out^.next;  {y lo inserto despues...}
   p_out^.next:=p_in;        {...de p_out            }
```

```
   p_out:=exec_queue^.next;
   p_in:=token_queue^.next;
   p_arq:=arq_buf^.next;

 end;{while}
end;{h_sched}

{
*************************************************************
*                                                           *
*              RUTINAS PARA LA MATCHING UNIT                *
*                                                           *
*************************************************************
}


function match_instance (inst1,inst2:stp_ptr) : boolean;
{
****************************MATCH_INSTANCE*********************
 verifica que dos listas de instancias (tags) sean iguales. inst1 e
 inst 2 apuntan a las dos listas. Obviamente, match_instance:=true
 si las listas son iguales. 17/5/89.
}
var g,h:stp_ptr;match:boolean;

begin
 match :=true;
 g:=inst1; h:=inst2;

 repeat
  if (g^.id = h^.id) then
   begin
    g:=g^.next;
    h:=h^.next;
   end
   else match:=false;
 until (g=nil) or (h=nil) or (not match);
 if ((g=nil) and (h=nil) and (match)) then match_instance:=true
                                     else match_instance:=false;

end;

procedure Copy_list(var source,dest:stp_ptr);
{
*********************************COPY_LIST*********************
 Crea una nueva lista (apuntada por dest) igual a la lista de entrada
 (apuntada por source). 18/5/89.
}

var p1,p2:stp_ptr;

begin
 new(p1);
 dest:=p1;
 p2:=source;

 repeat
```

```
  p1^.id:= p2^.id;
  p2 := p2^.next;
  if p2 <> nil then
   begin
   new(p1^.next);
   p1:=p1^.next;
  end
  else p1^.next:=nil;

 until p2=nil;
end;

Procedure search1 (var indice: pointr; id : integer; var exist : bool
{
********************** SEARCH1 ****************************
   busqueda del identificador id en la lista apuntada por indice.

   parametros de entrada:

   -indice: puntero al principio de la lista (se busca a partir de
            la posicion de indice).

   -id: identificacion del item de la lista (entero).

   -exist: variable logica. Si el item esta en la lista, exist = true
           La rutina se detiene en la primera aparicion de id en la
           lista.

   Salida: Si el item esta en la lista, es apuntado por indice y
           exist = true. Si no esta en la lista, indice apunta al
           ultimo item, y exist = false.

  PROBADA 21/9/88 OK.
}

begin  { search1 }
 exist := true;
 while ((indice^.token.id <> id)
       and (indice <> nil)) do indice:=indice^.next;
 if indice=nil then exist:=false;
end; { search1 }



procedure readfile (file1:archptr; id:integer; inst:stp_ptr;
                    var present : boolean);
{
************************ READFILE ***************************
lectura de un item de programa desde el buffer del editor.

Parametros de entrada:

-file1: puntero al buffer del editor de algoritmos.

-id: identificacion del item de programa.

Salida: si el item existe en el buffer, es agregado en la lista del
        matching store, en la posicion apuntada por index (puntero
```

```
        global) y present=true. Si el item no existe, present=false.

PROBADA 22/9/88 OK.Modificada el 2/5/89: el editor es ahora una lista
}

var indice1 : pointr;
    temp,temp1,temp2 :archi;
    existe:boolean;
    i:integer;

procedure search2 (p:archptr; id : integer);
{ busqueda en el archivo }
 begin
  present:=false;
  repeat
  p:=p^.next;

  if p^.nodo.id=id then
   begin
    present:=true;
    p^.instancia:=p^.instancia+1;
    temp:=p^;
   end
 {si el nodo es tipo macro busco en su lista}
  else if p^.nodo.class=3 then search2(p^.macro,id);
 until (present) or (p=nil);
end;{ search2 }

procedure search_input_node (p:archptr; var temp1:archi;
                             var existe:boolean);
{ busqueda en la lista de nodos tipo "input" }
begin
 existe:=false;
 repeat
  p:=p^.next;

 {busco el nodo clase 5 : input}
  if p^.nodo.class=5 then
   begin
    existe:=true;
    p^.instancia:=p^.instancia+1;
    temp1:=p^;
   end;
 until (existe) or (p=nil);
end;{ search_input_node}

procedure search_output_node (p:archptr; var temp1:archi;
                              var existe:boolean);
{ busqueda en la lista de nodos tipo "output" }
begin
 existe:=false;
 repeat
  p:=p^.next;

 {busco el nodo clase 6 : output}
  if p^.nodo.class=6 then
   begin
    existe:=true;
    p^.instancia:=p^.instancia+1;
```

```
    temp1:=p^;
   end;
 until (existe) or (p=nil);
end;{ search_output_node}

procedure create_node(node:archi);
begin
 new (indice1);            { creo nuevo item en el matching_store }
 indice1^.token := node.nodo;
 indice1^.next := index^.next;
 indice1^.hard := node.instancia;{almacenamiento temporario instancia
 index^.next := indice1;
 index^.instancia:=nil;
 index:=indice1;

end;


begin { readfile }
 search2 (file1,id);
 if present then
  begin

   if temp.nodo.class=3 then
    begin {el nodo es macro}
{ Durante simulacion, los nodos tipo macro se "dividen" en dos:
  Un nodo "cabecera", con el mismo id que el nodo original y sus
  mismas entradas, pero con una unica salida que apunta al nodo
  "input" del macro.
  Un nodo "salida" con el mismo id que el nodo original, pero
  negativo,con las misma salidas, pero con una unica entrada que
  proviene del nodo "output" del macro asociado.
  Esto se hace durante tiempo de simulacion, sin alterar en modo
  alguno el buffer del editor. Tal vez sea un poco mas lento,
  pero mucho mas seguro
}
      search_output_node(temp.macro,temp1,existe); { busco nodo outpu
      if existe then
       begin
        temp2:=temp;  { este sera el nodo salida }
        temp2.nodo.class:=0;
        temp2.nodo.inputs[2].id:=0;
        temp2.nodo.inputs[1].id:=temp1.nodo.id; { la entrada es outpu
        temp2.nodo.inputs[1].ccost:=0;
        temp2.nodo.id:=-temp2.nodo.id; { el id es negativo }
        temp1.nodo.outputs[1].id:=temp2.nodo.id; { arreglo salidas...
        temp1.nodo.outputs[1].ccost:=0;          {... del nodo output
        temp1.nodo.outputs[2].id:=0;
        create_node(temp2);        {inserto los dos nodos...}
        create_node(temp1);        {... en el matching store}
       end;
      search_input_node(temp.macro,temp1,existe); { busco el nodo inp
      if existe then
       begin
        temp.nodo.outputs[2].id:=0;   { arreglo salidas nodo macro...
        temp.nodo.outputs[1].id:=temp1.nodo.id;  {...para que apunten
        temp.nodo.outputs[1].ccost:=0;           {...al nodo input}
        i:=0;
        repeat i:=i+1 until temp1.nodo.inputs[i].id=0; { entradas...}
```

```
          temp1.nodo.inputs[i].id:=temp.nodo.id; {...del nodo input...}
          create_node(temp1);  {...y lo inserto en el matching store. }
        end;
      end;

    create_node(temp); { inserto el nodo en el matching store }
    copy_list(inst,index^.instancia);
   end;
end; { readfile }

procedure search_input_node (var p:archptr; var existe:boolean);
{ busqueda en la lista de nodos tipo "input" }
begin
 existe:=false;
 repeat
  p:=p^.next;

 {busco el nodo clase 5 : input}
  if p^.nodo.class=5 then
   begin
    existe:=true;
    p^.instancia:=p^.instancia+1;
   end;
 until (existe) or (p=nil);
end;{ search_input_node}



Procedure insert (id,input : integer; var enabled : boolean);
{
********************** INSERT ******************************
 Habilita la entrada input del token id.
 Parametros de entrada:

 -id: identificacion del item de la lista.

 -input: identificacion de la entrada a habilitar.

 Salida: la entrada input del nodo se habilita. Si ello produce la
         habilitacion del nodo ( de acuerdo a la politica
         de entrada del nodo)
         enabled = true. Sino, enabled =false.

PROBADA 22/9/88 OK.
}

var i : integer;

begin
 enabled := true;
 i:=0;
 repeat { me posiciono en el arco a actualizar }
  i:=i+1;
  if i =11 then
   begin
    clrscr;
    write('ERROR : el nodo ',id,' no tiene entrada ',input,'
    espera;
    halt(1);
```

```
    end;
  until index^.token.inputs [i].id = input;

  index^.token.inputs [i].ccost :=1;

{ Diferentes casos segun el comportamiento del nodo }

  case index^.token.input_policy of

    0: begin {todas las entradas deben estar habilitadas}
         i := 1;
         repeat
          if index^.token.inputs [i].ccost =0 then enabled := false;
          i := i + 1;
         until (i=11) or (index^.token.inputs[i].id = 0);
       end;{ caso 1 }

    1: begin {al menos una entrada debe estar habilitada}
       end;{ caso 2. Facil, no?}

    else enabled := false;

    end; { Case }

end; { Insert }


Procedure switchnode(id,x,y:integer;state:boolean);
{*****************************SWITCHNODE*****************************
  Esta rutina enciende-apaga los nodos del grafo de algoritmo. Se
  utiliza para "animar" la simulacion.
  Datos de entrada:
   id: identificacion del nodo a prender-apagar.
   x,y: coordenadas del nodo en cuestion
   state: estado del nodo. False: nodo no habilitado (azul)
                           True: nodo habilitado (verde).
  Salida: el color del nodo en cuestion es actualizado (es una rutina
  bastante inofensiva).
********************************************************************}


var wcolor,fcolor,j:integer;
    inst:string[4];
    ind4:pointr;

begin
{ para imprimir cuantas instancias del nodo hay habilitadas}

 j:=0;
 ind4:=token_queue;
 if state then
  begin
   repeat
    search1(ind4,id,exist);
    j:=j+1;
    if exist then ind4:=ind4^.next;
   until not exist;
```

```
    j:=j-1;
   end;
  ind4:=exec_queue;
  repeat
   search1(ind4,id,exist);
   j:=j+1;
   if exist then ind4:=ind4^.next;
  until not exist;
  j:=j-1;
  if not state then j:=j-1;
  str(j,inst);
  settextstyle(defaultfont,horizdir,1);
  setcolor (lightred);
  outtextxy(x+9,y+5,'[[');
  settextstyle(smallfont,horizdir,2);
  setcolor (white);
  outtextxy(x+9,y+5,inst);
  settextstyle(smallfont,horizdir,4);
  settextjustify(centertext,centertext);

  if j<>0 then
   begin
    fcolor:=lightgreen;
    wcolor:=lightred;
   end
  else
   begin
    fcolor:=lightblue;
    wcolor:=yellow;
   end;
  str(id,sid);
  setcolor(white);
  circle(x,y,10)  ;
  setfillstyle(SolidFill,fcolor);
  setcolor(fcolor);
  outtextxy(x,y,sid);
  floodfill(x,y,white);
  setcolor(wcolor);
  outtextxy(x,y,sid);
  settextjustify(lefttext,toptext);
  settextstyle(defaultfont,horizdir,1);
end;



Procedure qput (ind1 : pointr);
{
************************ QPUT ******************************
 Saca un elemento del matching_store (ind1) y lo coloca al en la
 enabled token queue . Si ya existe un elemento en la token queue
 con el mismo id, se crea una nueva instancia.

Parametros de entrada:

 -ind1: puntero al elemento a sacar en el matching store.

 -ind2: puntero a la enabled token queue.
```

```
PROBADA 22/9/88 OK.
}

var ind3,ind4 : pointr;
    i,j : integer;
    inst:string[4];
    p,p1:stp_ptr;

begin
 ind4:=token_queue;
 ind3:=match_store;
 while ind3^.next <> ind1 do ind3:=ind3^.next;
 exist:=true;
 search1(ind4,ind1^.token.id,exist);
 if exist then while (ind4^.next^.token.id = ind1^.token.id) and
                     (ind4^.next <> nil) do ind4 := ind4^.next
 else ind4 := token_queue;
 ind3^.next:=ind1^.next;
 ind1^.next:=ind4^.next;
 ind4^.next:=ind1;

{actualizo instancia}
 p:=ind1^.instancia;
 while p^.next<>nil do p:=p^.next;
 if ind1^.token.class = 1 then
  begin {si es tipo call,genero item en la cola}
   new(p^.next);
   p:=p^.next;
   p^.next:=nil;
  end

 else if ind1^.token.class = 2 then
  begin {si es tipo return, elimino item}
   p1:=ind1^.instancia;
   while ((p1^.next <> p) and (p1^.next <> nil)) do p1:=p1^.next;
   p:=p1;
   if p^.next<> nil then dispose(p^.next);
   p^.next:=nil;
  end;
 p^.id:=ind1^.hard;
 i := 1;

{apago los tokens de entrada al nodo (han sido absorbidos)}
 if animated then
  begin
   while ind1^.token.inputs[i].id <> 0 do
    begin
     token (ind1^.token.inputs[i].id, ind1^.token.id,false,alg_buf);
     i:=i+1;
    end;

{ Comienza la ejecucion del nodo-revisar 6-2-89 }

   if (ind1^.token.visible) then
    switchnode(ind1^.token.id,ind1^.token.x,ind1^.token.y,on);
  end;
 if trace_on then
  begin
   evento.time:=t_simul;
```

```
    evento.class:= hab;
    evento.info[1]:=ind1^.token.id;
    write(tracfile,evento);
   end;
 end; { qput }


Procedure matching_unit  (id_fuente , id_destino : integer ;
                          inst:stp_ptr);
{
****************** Matching Unit ***************************
Recibe un token de la execution unit. Busca el nodo correspon-
diente en el matching_store. Si no lo encuentra, lo busca en el
archivo de especificacion de programa. Una vez hallado, actualiza
la entrada id_fuente del nodo en cuestion (id_destino). Si esto
provoca la activacion del nodo, lo retira del matching_store y lo
almacena en la token_queue (nodos activos).
PROBADA 28/9/88 OK.
************************************************************
}
begin
 index := match_store; { init pointr }
 search1 (index, id_destino, exist); { busco en el matching store }
{si existe verifico el tag}
 if exist then
  repeat
   if index^.instancia=nil then copy_list(inst,index^.instancia);
   exist := match_instance(index^.instancia,inst);
   if not exist then
    begin
     repeat index := index^.next;
     until ((index=nil) or (index^.token.id=id_destino))
    end;
  until (( index=nil ) or (exist));

 if not exist then
  begin
   index := match_store;
   readfile (alg_buf, id_destino,inst, present); { busco en el archiv
   if not present then
    begin
     clrscr;
     write('ERROR : el codigo ',id_destino,' no existe en el algoritm
     espera;
     halt(1);
    end;
  end;
 insert (id_destino,id_fuente,enabled); { actualizo entradas al nodo
 if enabled then
  begin
   if index^.token.id <0 then index^.token.id:=-index^.token.id;
   Qput (index);  { si esta habilitado, lo ingreso en la token queue
  end;
end;

procedure delete_r(base,p1:r_ptr);
var s:stp_ptr;

begin
```

```
 while base^.next <> p1 do base:=base^.next;
 base^.next:=p1^.next;

{borro el arco}
 while p1^.arc^.instancia <> nil do
  begin
   s:= p1^.arc^.instancia;
   p1^.arc^.instancia:=s^.next;
   dispose(s);
  end;
 dispose(p1^.arc);

{borro el ruteo}
 while p1^.ruta <> nil do
  begin
   s:=p1^.ruta;
   p1^.ruta:=p1^.ruta^.next;
   dispose(s);
  end;

 dispose(p1);
 p1:=nil;
end;

procedure delete_l(base,p1:lnk_ptr);
var s:stp_ptr;
    a:arcpntr;

begin
 while base^.next <> p1 do base:=base^.next;
 base^.next:=p1^.next;

{borro la lista de arcos}
 while p1^.p_list <> nil do
  begin
   a:=p1^.p_list;
   p1^.p_list:=p1^.p_list^.next;

   while a^.instancia <> nil do
    begin
     s:= a^.instancia;
     a^.instancia:=s^.next;
     dispose(s);
    end;

   dispose(a);
  end;
 dispose(p1);
end;


{
*****************************************************************
*                                                               *
*            RUTINAS PARA LA ROUTER UNIT                        *
*                                                               *
*****************************************************************
}
procedure ruteos;
```

```
var pun1, temp : arcpntr;
    pun2       : r_ptr;
    pun3, otro : lnk_ptr;
    ok         : boolean;
    ll :stp_ptr;
    rj      : integer;
    ri : archptr;

Procedure Search_destino (p1 : arcpntr; var ok : boolean);
{************************SEARCH DESTINO*****************************
  Este procedure acepta como entrada un arco de salida de la token
  queue (P1), y busca el procesador a quien esta asignado el nodo
  de salida(array asiga), insertandolo en p1^.final.
}
var i:integer;

begin
 i:=0;
 ok:=true;
 repeat i:=i+1 until (asiga[i].task = p1^.final)  or (i=maxnode);
 if  (i=maxnode) then ok:=false;
 if ok then p1^.h_end := asiga[i].procesador;
end; { Search_destino }


Procedure Update (var p3 : r_ptr; var part:arcpntr; uend: boolean);
{****************************UPDATE*********************************
  Este procedure se utiliza para actualizar el ruteo de un mensaje
  por la arquitectura. Los mensajes se almacenan la la arc_queue. El
  ruteo esta almacenado en la router_queue.
  Update saca un elemento del item p3^ , que queda apuntado por part,
  y actualiza el fifo p3^.p-list. Si el elemento que se saca es el
  ultimo (ultimo paso de ruteo), uend = true y p3^ se elimina.
}
var temp,s:stp_ptr;
begin
 uend:=false;
 new(part); { part es el mensaje, con su paso de ruteo asociado}
 part^:= p3^.arc^;
 copy_list(p3^.arc^.instancia,part^.instancia);{copio lista de instan
 part^.h_source:=p3^.ruta^.id; {h_source-h_final indican el link   }
 temp:=P3^.ruta;               {de la arquitectura por donde rutear}
 p3^.ruta:=p3^.ruta^.next;     {el mensaje }
 part^.h_end:=p3^.ruta^.id;
 dispose(temp);
 if p3^.ruta^.next = nil then
  begin  {elimino entrada en la router_queue}
   part^.class:=final;
   delete_r(router_queue,p3);
   uend:=true;
  end;
end;


Procedure put_there (var p3: lnk_ptr;part: arcpntr);
{************************PUT_THERE*********************************
 Este procedure inserta un elemento (apuntado por part) al final de l
 cola de mensajes del link apuntado por p3.
```

```
}
var pp : arcpntr;
    i : integer;

begin
 i:=0;
 pp:=p3^.p_list;
 while pp^.next <> nil do
  begin
   pp:=pp^.next; { voy al final de la lista privada }
   i:=i+1;
  end;

 pp^.next :=part;
 part^.costo:=part^.costo * p3^.costo;
 part^.next := nil;
 i:=i+1;
 if i=p3^.cap then p3^.bloqued:=true;
end;


Procedure Generar_ruteo (p1 : arcpntr; p2 : r_ptr);
{************************GENERAR_RUTEO*****************************
  Este procedure genera los pasos de ruteo de mensajes a partir de
  la informacion (fuente,destino) de p1^. La salida de la rutina es
  un elemento nuevo en la router_queue, que en p2^.p_list tiene los
  sucesivos pasos (ordenados en secuencia) de ruteo
}

var j,half,h_inicio,hed:integer;
origen,xfinal,k,i:archptr;
active,is_in_poly,finish:boolean;
p,pp,ppp :stp_ptr;
sinic,sfina:string[3];
nuevo:r_ptr;


begin

 new(nuevo);
 new(nuevo^.arc);
 nuevo^.arc^:=p1^;
 copy_list(p1^.instancia,nuevo^.arc^.instancia);{copio instancias}
 nuevo^.arc^.next:=nil;
 nuevo^.next:=p2^.next;
 p2^.next:=nuevo;
 new(p);
 nuevo^.ruta:=p;
 new(pp);
 p^.id :=p1^.h_source;
 pp^.id :=p1^.h_end;
 p^.next:=pp;
 pp^.next:=nil;
 pp:=p;

 origen:=arq_buf;
 repeat origen:=origen^.next until p1^.h_source = origen^.nodo.id;
 xfinal:=arq_buf;
 repeat xfinal:=xfinal^.next until p1^.h_end = xfinal^.nodo.id;
```

```
if p1^.h_source=p1^.h_end then exit; {asignados al mismo proc,sigo}
setactivepage(1);
clearviewport;
setcolor(white);
xshow(arq_buf,arquitectura,0,0,1,1);
rectangle(0,0,getmaxx,getmaxy-14);

line(0,12,getmaxx,12);
setfillstyle(solidfill,white);
bar(1,1,544,11);
setfillstyle(solidfill,red);
bar(545,1,maxx-1,11);
outtextxy(550,3,'BOLAS V0.0');
str(p1^.h_source,sinic);
str(p1^.h_end,sfina);
setcolor(black);
moveto (40,3);
outtext('Indique el ruteo entre nodo ');
outtext(sinic);
outtext(' y nodo ');
outtext(sfina);
setvisualpage(1);
setfillstyle(solidfill,lightgray);
floodfill(origen^.nodo.x,origen^.nodo.y+5,white);
setfillstyle(solidfill,lightred);
floodfill(xfinal^.nodo.x,xfinal^.nodo.y+5,white);
settextjustify(centertext,centertext);
settextstyle(smallfont,horizdir,4);
outtextxy(origen^.nodo.x,origen^.nodo.y,sinic);
outtextxy(xfinal^.nodo.x,xfinal^.nodo.y,sfina);
setvisualpage(1);
InitM (m_stat,m_par,ok,origen^.nodo.x,origen^.nodo.y);
Set_Hand_Cursor;
i:=origen;
finish:=false;
{loop principal}
show_cursor(m_stat,m_par);
setfillstyle(solidfill,lightgray);
repeat
 Get_Pos (m_stat,m_par);
 if (m_stat.tecla1) and (not finish) then
  begin
   m_stat.tecla1:=false;
   k:=arq_buf;
   scan(m_stat.xnuevo,m_stat.ynuevo,k,is_in_poly);

   if is_in_poly then
    begin
     j:=0;
     repeat j:=j+1 until (j=11) or (i^.nodo.outputs[j].id=k^.nodo.id
     if (j <> 11) and (k<>xfinal) then
      begin
       hide_cursor(m_stat,m_par);
       str(k^.nodo.id,sinic);
       outtextxy(k^.nodo.x,k^.nodo.y,sinic);
       floodfill(k^.nodo.x,k^.nodo.y+5,white);
       outtextxy(k^.nodo.x,k^.nodo.y,sinic);
       show_cursor(m_stat,m_par);
       new(ppp);
```

```
          ppp^.id:=k^.nodo.id;
          ppp^.next:=pp^.next;
          pp^.next:=ppp;
          pp:=pp^.next;{inserte paso de ruteo}
          i:=k;
         end
        else if (j<>11) and (k=xfinal) then
         begin
          hide_cursor(m_stat,m_par);
          floodfill(xfinal^.nodo.x,xfinal^.nodo.y+5,white);
          str(k^.nodo.id,sinic);
          outtextxy(k^.nodo.x,k^.nodo.y,sinic);
          finish:=true;
          show_cursor(m_stat,m_par);
         end;
      end;
     end;
  until (m_stat.teclar) and (finish);
  initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
  setactivepage(0);
  setvisualpage(0);
  settextstyle(defaultfont,horizdir,1);
  settextjustify(lefttext,toptext);
  m_stat.teclar:=false;
end; {Generar_ruteo}



Procedure search_R_queue(var indice2:r_ptr;indice1:arcpntr;
                         var ok:boolean);
{************************SEARCH_R_QUEUE************************
 Procedimiento de busqueda del ruteo del mensaje apuntado por
 indice1 en la router_queue (apuntada por indice2). Si no lo
 encuentra ok:=false.
}
begin
if indice2 = nil then ok:=false
 else if (indice2^.arc^.inicio   <> indice1^.inicio)
      or (indice2^.arc^.final    <> indice1^.final)
      then
       begin
        indice2:=indice2^.next;
        search_r_queue (indice2,indice1,ok);
       end;
end;

Procedure search_l_queue (var indice3:lnk_ptr;temp: arcpntr;
                          var ok:boolean);
{********************SEARCH_L_QUEUE*****************************
 Procedimiento de busqueda del link por donde esta siendo
 ruteado el mensaje apuntado por temp en la link_queue
}
begin
 if indice3 = nil then ok:=false
  else if (indice3^.cola<>temp^.h_source) or
          (indice3^.punta<>temp^.h_end)
   then begin
    indice3:=indice3^.next;
```

```
      search_l_queue (indice3,temp,ok);
     end;
end;


procedure in_size;
var p1:arcpntr;p2:r_ptr;p3:lnk_ptr;i,j,k:integer;
begin
 i:=0;j:=0;k:=0;
 p1:=arc_queue^.next;
 p2:=router_queue^.next;
 p3:=link_queue^.next;
 while p1 <> nil do
  begin
   i:=i+1;
   p1:=p1^.next;
  end;
 while p2 <> nil do
  begin
   j:=j+1;
   p2:=p2^.next;
  end;
 while p3 <> nil do
  begin
   k:=k+1;
   p3:=p3^.next;
  end;
 writeln;
 write('Arc: ',i,'Router: ',j,'link: ',k,'resta: ',memavail);
 espera;
end;

begin {router}

 pun1:=arc_queue^.next;
 while (pun1 <> nil) do
  begin {recorro la arc_queue ruteando mensajes}
  if (pun1^.class = logico) then
   begin       {debo generar los pasos de ruteo}
    search_destino (pun1,ok);   {a quien esta asignado destino?}

   {si asignados al mismo procesador no hay costo de comunicacion}

    if pun1^.h_source=pun1^.h_end then pun1^.costo:=0;
    pun1^.class := intermedio;  {los "intermedios" estan ruteando}
    pun2:=router_queue;
    generar_ruteo (pun1,pun2); {generacion manual de ruteo}
   end;

{tratamiento de los mensajes tipo "intermedio": son los que estan en
 proceso de ruteo por los arcos del arquitectura}

  ok :=true;

{La router queue almacena los ruteos de cada arco. A medida que se
 van utiizando, los sucesivos pasos de ruteo se van eliminando }

  pun2 := router_queue^.next;
```

```
{busco los pasos de ruteo (pun2) del arco del algoritmo (pun1)}
 search_r_queue (pun2,pun1,ok);
  ok :=true;

{extraigo el paso de ruteo en temp}
  update(pun2,temp,ok);
  ok :=true;
  pun3:=link_queue^.next;

{busco el link fisico de la arquitectura}
  search_l_queue(pun3,temp,ok);

{si el link ya tiene mensajes en su buffer, agrego un mensje nuevo}
  if trace_on then
   begin
    evento.time:=t_simul;
    evento.class:= rut_stp;
    evento.info[1]:=temp^.inicio;
    evento.info[2]:=temp^.final;
    evento.info[3]:=temp^.h_source;
    evento.info[4]:=temp^.h_end;
    write(tracfile,evento);
   end;
  if ok then put_there(pun3,temp)
  else
   begin {si no, creo el link}
    new(otro);
    otro^.next:=link_queue^.next;
    link_queue^.next:=otro;
    otro^.cola:=temp^.h_source;
    otro^.punta:=temp^.h_end;
    otro^.bloqued:=false;
    if temp^.h_source <> temp^.h_end then
     begin
      ri:=arq_buf^.next;
      while ri^.nodo.id <> temp^.h_source do ri := ri^.next;
      rj := 0;
      repeat rj := rj + 1
      until ri^.nodo.outputs[rj].id = temp^.h_end;
      otro^.costo := ri^.nodo.outputs[rj].ccost;
      otro^.cap:=ri^.nodo.outputs[rj].prob;
      temp^.costo:=temp^.costo * otro^.costo;
     end;
    otro^.p_list:=temp;
    otro^.p_list^.next:=nil;
   end;

  delet1(arc_queue,pun1); {elimino el mensaje de la arc_queue}
  pun1:=arc_queue^.next;
 end;
end;
```

```
{
*************************************************************
*                                                           *
*            RUTINAS PARA LA  EXEC  UNIT                    *
*                                                           *
*************************************************************
}

procedure delet1 (ind0,ind1 : arcpntr);
{
************************ DELET1 ******************************
 Borra un elemento de la lista
 Prametros de entrada:

-ind0: puntero al comienzo de la lista.

-ind1: puntero al item a borrar

*************************************************************
}

var p:stp_ptr;
begin
 while ind0^.next <> ind1 do ind0:=ind0^.next;
 ind0^.next := ind1^.next;
{borro la lista de instancias}
 while ind1^.instancia <> nil do
  begin
   p:=ind1^.instancia;
   ind1^.instancia:=p^.next;
   dispose(p);
  end;
 dispose (ind1);
end; { delet1 }



procedure extract (indice:pointr ; indice1:arcpntr);
{
************************ EXTRACT *****************************
extrae un token de la exec queue e inserta en la arc queue
sus arcos de salida, de acuerdo con la politica de ejecucion.
Es utilizada por la subrutina minus.

Parametros de entrada:
       indice : puntero al elemento a extraer de la exec queue
       indice1 : puntero a la arc queue.

salida : las dos colas quedan actualizadas.
*************************************************************
}

var p : arcpntr;
    i,pmax,a,b : integer;
begin

{termina la ejecuci"n del nodo: apago el dibujo}

 if ((animated) and (indice^.token.visible)
```

```
    and (indice^.token.class<>3))
 then switchnode(indice^.token.id,indice^.token.x,
                 indice^.token.y,off);

 if trace_on then
  begin
   evento.time:=t_simul;
   evento.class:=end_ejec;
   evento.info[1]:=indice^.token.id;
   evento.info[2]:=indice^.instancia^.id; {ojo}
   evento.info[3]:=indice^.hard;
   write(tracfile,evento);
  end;

{ Distintos casos segun el comportamiento del nodo  }

 case indice^.token.exec_policy of

  0: begin { un token en cada salida }
      i := 1;
      while indice^.token.outputs[i].id <> 0 do
       begin
        new (p);
        p^.inicio := indice^.token.id;
        p^.final := indice^.token.outputs[i].id;
        p^.costo := indice^.token.outputs[i].ccost;
        p^.h_source:=indice^.hard;
        p^.next := indice1^.next;
        p^.class := logico;
        copy_list(indice^.instancia,p^.instancia);
        indice1^.next := p;
        i := i+1;
        if animated then token(p^.inicio,p^.final,true,alg_buf);
        if trace_on then
         begin
          evento.time:=t_simul;
          evento.class:=beg_com;
          evento.info[1]:=p^.inicio;
          evento.info[2]:=p^.final;
          write(tracfile,evento);
         end;
       end;
      end; { caso 1 }

    1: begin { un token en alguna de las salidas }
        i := 1;
        pmax:=0;
        while indice^.token.outputs[i].id <> 0 do i:=i+1;
        if i<> 1 then
         begin
          repeat
           a:=random(100)+1;
           b:=random (i-1)+1;
          until indice^.token.outputs[b].prob >=a;
          new (p);
          p^.inicio := indice^.token.id;
          p^.final := indice^.token.outputs[b].id;
          p^.h_source:=indice^.hard;
          p^.costo := indice^.token.outputs[b].ccost;
```

```
            p^.next := indice1^.next;
            p^.class := logico;
            copy_list(indice^.instancia,p^.instancia);
            indice1^.next := p;
            if animated then token(p^.inicio,p^.final,true,alg_buf);
            if trace_on then
             begin
              evento.time:=t_simul;
              evento.class:=beg_com;
              evento.info[1]:=p^.inicio;
              evento.info[2]:=p^.final;
              write(tracfile,evento);
             end;
           end;
         end; { caso 2 }

       end; { case }

 delet(exec_queue,indice);
end { extract };



procedure finish (var indice , otq:arcpntr);
{
************************ FINISH ****************************
Elimina un arco de la arc queue y lo deposita en la output queue
Es utilizada por la subrutina minus

Parametros de entrada:
  indice: apunta al elemento a eliminar en la arc queue
  otq : puntero a la output token queue.

Salida: las dos colas quedan actualizadas.
*****modificada 2/3/89 miguel. La nueva version no hace
     ni new ni dispose.
************************************************************
}

var item : arcpntr;

begin
 item := arc_queue;
 while item^.next <> indice do item:=item^.next;
 item^.next := indice^.next;
 indice^.next := otq^.next;
 otq^.next := indice;
 if trace_on then
  begin
   evento.time:=t_simul;
   evento.class:=end_com;
   evento.info[1]:=indice^.inicio;
   evento.info[2]:=indice^.final;
   evento.info[3]:=indice^.h_source;
   evento.info[4]:=indice^.h_end;
   write(tracfile,evento);
  end;
end { finish };
```

```
procedure search3 (var tmin:longint;var id:integer);
{
********************** SEARCH3 ********************************
 Busca el item con menor tiempo de ejecucion asociado en la exec
 queue y en la arc_queue.

Parametros de entrada: ninguno. Los punteros a las dos estructuras
de datos son variables globales.

Salida: tmin (integer), el menor tiempo de ejecucion. El proximo
evento de la simulacion sera el asociado a t_simulacion + tmin.

PROBADA 23/9/88 OK.
modificada 1/2/89 para simular asignaciones.
**Aclaracion: cuando los nodos estan asignados a procesadores, aparec
el problema de decidir cual de los nodos activados asignados a un dad
procesador se ejecuta.Probablemente lo mas eficiente seria ejecutar l
tarea mas corta, pero hablando en forma realista, es dudoso que el pr
cesador conozca esta informacion a priori. Por lo tanto, aqui supongo
que se elige el nodo que encabeza la sublista de cada procesador. Lue
en este caso el tiempo minimo se busca entre los primeros de cada sub
lista.....Miguel.PROBADA 7/2/89 OK.
modificada 14/2/89 para simular ruteos.
**Aclaracion: cuando los nodos estan asignados a procesadores, es nec
sario simular el paso por los links de la arquitectura, con los event
ruteos en el caso de no existir una via directa. En ese caso (modo_de
racion=1) se utiliza el modulo ROUTER y para buscar el tiempo minimo,
examina, en lugar de la arc_queue, la link_queue........Miguel.
*****************************************************************
}
var indice1 : pointr;
    indice2 : arcpntr;
    indice3 : lnk_ptr;
    idactual:integer;

begin
 indice1 := exec_queue^.next;
 indice2 := arc_queue^.next;
 indice3 := link_queue^.next;
 id:=0;{procesador asignado}

 idactual:=-1;

 tmin:=maxint;

 case modo_de_operacion of
  0:begin
     while indice1 <> nil do
      begin   { busqueda en la exec queue }
       if indice1^.token.pcost < tmin then
        begin
         tmin := indice1^.token.pcost;
         id:=indice1^.hard;
        end;
       indice1 := indice1^.next;
     end;{while}
```

```
    while indice2 <> nil do
     begin   { busqueda en la arc_queue }
      if indice2^.costo < tmin then
       begin
        tmin := indice2^.costo;
        id:=0;
       end;
      indice2 := indice2^.next;
     end;
    end;{begin0}

  1:begin
     while indice1<> nil do
      begin
       if( idactual<>indice1^.hard) and (indice1^.token.pcost < tmin)
        then
        begin
         tmin := indice1^.token.pcost;
         id:=indice1^.hard;
        end;{if}

       idactual:=indice1^.hard;
       indice1 :=indice1^.next;
      end;{while}
     while indice3<> nil do
      begin
       if indice3^.p_list^.costo < tmin then
        begin
         tmin:= indice3^.p_list^.costo;
         id:=0;{?}
        end;
       indice3 := indice3^.next;
      end;
    end;{begin1}

  end;{case}

end; { search3 }


procedure move_token (var p1 : lnk_ptr;  p2 : arcpntr);

var p3 : lnk_ptr;
    p4 : arcpntr;

begin
 p4 := p1^.p_list;                    {Alm. temporal token a mover}
 p1^.p_list := p1^.p_list^.next;      {Actualizo link queue}
 p4^.next := p2^.next;                {Muevo a la lista ...}
 p2^.next := p4;                      { ... apuntada por p2. }
{ Si la lista del link esta agotada, la elimino }
 if ( p1^.p_list = nil ) then
  begin
   delete_l(link_queue,p1);
  end ;

 end; { move_token }

procedure minus (tmin:integer;var id:integer);
```

```
{
****************************** MINUS ******************************
Resta a todos los elementos de las dos estructuras de datos el valor
de tmin. Esto equivale a simular el paso de tmin unidades de tiempo
de simulacion. Si la resta produce que pcost (exec queue) se haga
0, la subrutina extract saca el token de la cola e ingresa sus arcos
en la arc_queue.

Parametros de entrada:
  tmin: entero a restar

Salida: las dos colas son actualizadas al nuevo tiempo de simulacion.

*****************************************************************
}

var indice1,temp0 : pointr;
    indice2,indice3,temp1 : arcpntr;
    indice4,temp2:lnk_ptr;
    listo:boolean ;
    idactual,elapsed:integer;
begin

 indice1 := exec_queue^.next;
 indice2 := arc_queue^.next;
 indice3 := output_queue;
 indice4 := link_queue^.next;
 listo:=false;
 idactual:=-1;

 case modo_de_operacion of
  0: begin {no hago caso a la asignaci"n}
      while indice2 <> nil do
       begin
        temp1 := indice2^.next;
        indice2^.costo := indice2^.costo - tmin;
        if indice2^.costo = 0 then finish (indice2,indice3);
        indice2 := temp1;
       end;

      indice2 := arc_queue;


      while indice1 <> nil do
       begin
        temp0 := indice1^.next;
        indice1^.token.pcost := indice1^.token.pcost - tmin;
        if indice1^.token.pcost = 0 then extract (indice1,indice2);
        indice1 := temp0;
       end;
      end;

  1: begin {asignacion manual,primera prueba}

      while indice4 <> nil do
       begin  {analizo la link queue}
        temp2:= indice4^.next;
        elapsed:=indice4^.p_list^.costo;
        if indice4^.p_list <> nil then
```


```
        begin
         indice4^.p_list^.costo:=indice4^.p_list^.costo - tmin;
         if indice4^.p_list^.costo = 0 then
          begin
    {si es el ultimo paso de ruteo lo mando a la output queue: }
           if indice4^.p_list^.class=final then
            begin
             if trace_on then begin
             evento.time:=t_simul;
             evento.class:=end_com;
             evento.info[1]:=indice4^.p_list^.inicio;
             evento.info[2]:=indice4^.p_list^.final;
             evento.info[3]:=indice4^.p_list^.h_source;
             evento.info[4]:=indice4^.p_list^.h_end;
             write(tracfile,evento);
            end;
           move_token(indice4,output_queue);
          end
   {si no, lo vuelvo a ingresar en la arc queue: }
         else
          begin
           indice4^.p_list^.costo:=elapsed;
           move_token (indice4,arc_queue);
          end; { if indice4^.p_list^.class }
         end; { if indice4^.p_list^.costo }
        end;
    indice4:=temp2;
   end; {while indice4}
  indice2 := arc_queue;
  while indice1 <> nil do
   begin
    temp0 := indice1^.next;
    if (indice1^.hard <> idactual) then
     begin
   {como todos los nodos asignados al mismo procesador estan juntos
    en la cola, solo descuento tmin al primero del grupo.
    (idactual<> indice1^.hard) }
       idactual := indice1^.hard;
       indice1^.token.pcost:=indice1^.token.pcost - tmin;
      end;

     if indice1^.token.pcost = 0 then
      begin
       if animated then En_ejecucion(temp0);
       extract (indice1,indice2);
      end;
     indice1 := temp0;
    end; {while}
   ruteos;
  end;{begin}

 end; {case}

end; { minus }


Procedure execute;
{
```

```
************************** Execute ***************************
retira los tokens activados de la token_queue, simula su ejecucion
actualizando el valor de T_simul (tiempo de simulacion global) e
ingresa sus arcos de salida en la arc_queue para su ejecucion.
Los arcos simulados se ingresan en la output_queue para volver a
la matching unit (Circular Pipeline).
PROBADA 28/9/88 OK.
**************************************************************
}
var id:integer;
begin
 search3 (tmin,id);
 if(tmin=maxint) then tmin:=0;
 t_simul := t_simul + tmin;
 minus (tmin,id);
end; { execute }

end.{unit}
```

```
program Simulator (input,output);
uses crt,dos,graph,defs2,edito3,utiles,bolas4,micky;

{Las definiciones de las variables y estructuras globales de datos
 estan en la unidad DEFS2.PAS.}

type

programa=file of element;                    {archivo de programa}
var
    mainmenu,simenu,opciones,schedules,routers:disp;    {menues}
    cursor : char;                          {para readkey's varios}
    x,y,i,option,simopt,schnum,routnum:integer;
    archivo_entrada,archivo_salida,archivo_trace:string;{nombres}
    tipo:tipografo;
    The_end: boolean;

{ Variables y rutinas de inicializacion grafica }

{$F+}
procedure MyExitProc;
begin
  ExitProc := OldExitProc; { Recuperar direccion de retorno }
  CloseGraph;              { Desactivar graficos }
end; { MyExitProc }
{$F-}

procedure Initialize;
{ Inicializar graficos y reportar posibles errores }
begin
  DirectVideo := False;
  OldExitProc := ExitProc;
  ExitProc := @MyExitProc;
  GraphDriver := Detect;                    { use autodetection }
  InitGraph(GraphDriver, GraphMode, '');    { activar graficos }
  ErrorCode := GraphResult;                 { error? }
  if ErrorCode <> grOk then
  begin
    Writeln('Error de gráficos: ', GraphErrorMsg(ErrorCode));
    Halt(1);
  end;
  MaxColor := GetMaxColor;
  MaxX := GetMaxX;          { coordenadas maximas de pantalla }
  MaxY := GetMaxY;
end; { Initialize }

procedure espera;
{ vacia el buffer de teclado y espera un caracter }

var regs:registers;
    cursor:char;

begin
regs.ax:=$0c06;
regs.dx:=$00ff;
intr($21,regs);
cursor:=readkey;
regs.ax:=$0c06;
regs.dx:=$00ff;
```

```
intr($21,regs);
end;

procedure empty;
{ vacia el buffer de caracteres de MSDOS }

var regs:registers;

begin
regs.ax:=$0c06;
regs.dx:=$00ff;
intr($21,regs);
end;

procedure mark_visible_nodes(p:archptr; flag:boolean);
{ indica que nodos pueden ser dibujados en pantalla }

begin
while p<> nil do
 begin
  p^.nodo.visible:=flag;
  p:=p^.next;
 end;
end;

procedure show1;
{ dibuja los contenidos del buffer del editor para el simulador }

var i,j,n,l,xmin,ymin,xx,yy:integer;
p,p1:archptr;

begin
 cleardevice;
 setcolor(white);

{ Dibujo del marco, zona de menues, etc. }
 rectangle (0,0,maxx,maxy-14);
 setfillstyle(solidfill,lightblue);
 line(0,12,maxx,12);
 floodfill (1,1,white);
 hide_cursor(m_stat,m_par);
 settextjustify(centertext,centertext);
 settextstyle(defaultfont,horizdir,0);
 line(48,0,48,12);
 outtextxy(24,7,'RUN');
 line(96,0,96,12);
 outtextxy(72,7,'STEP');
 line(144,0,144,12);
 outtextxy(120,7,'STOP');
 line(147,8,156,3); {flecha hacia arriba}
 line(156,3,165,8);
 line(171,3,180,8); {flecha hacia abajo}
 line(180,8,189,3);
 line(168,0,168,12);
 line(192,0,192,12);
 line(264,0,264,12);
 outtextxy(228,7,'OPTIONS');
 line(344,0,344,12);
 line(544,0,544,12);
```

```
 setfillstyle (solidfill,black);
 floodfill(346,2,white);
 setcolor(yellow);
 outtextxy (444,7,'INIT NODE');
 setcolor(white);
 setfillstyle(solidfill,red);
 floodfill(550,10,white);
 settextjustify(lefttext,toptext);
 outtextxy(550,3,'BOLAS V0.0 ');
 setfillstyle(solidfill,lightblue);
 settextjustify(centertext,centertext);
 settextstyle(smallfont,horizdir,4);

{ Dibujo del grafo }
 xshow(alg_buf,algoritmo,0,0,1,1);
 show_cursor(m_stat,m_par);
end;{show}

procedure m_menu;
{Inicialización de menues: Si se quiere agregar-modificar lineas,
 hacerlo aqui. Cuando luego se llama a text-window (ver main),
 modificar el parametro lineas de acuerdo al numero de opciones
 usado }

begin
 mainmenu[1]:=' ~ Editar Grafo de Algoritmo.';
 mainmenu[2]:=' ~ Editar Grafo de Arquitectura.';
 mainmenu[3]:=' ~ Simular.';
 mainmenu[4]:=' ~ Terminar la Sesion.';
 mainmenu[5]:=' ~ Asignacion Manual.';

 simenu[1]:=' ~ Simular solo el algoritmo.';
 simenu[2]:=' ~ Simular algoritmo y arquitectura.';
 simenu[3]:=' ~ Salir.';

 opciones[1]:=' ~ Set Stop Time.';
 opciones[2]:=' ~ Define Trace File.';
 opciones[3]:=' ~ Animation on/off.';

 schedules[1]:=' ~ Hand scheduling.';
 schedules[2]:=' ~ Sch. Policy 2.';
 schedules[3]:=' ~ Sch. Policy 3.';
 schedules[4]:=' ~ Sch. Policy 4.';
 schedules[5]:=' ~ Sch. Policy 5.';

 routers[1]:=' ~ Hand routing.';
 routers[2]:=' ~ Rout. Policy 2.';
 routers[3]:=' ~ Rout. Policy 3.';
 routers[4]:=' ~ Rout. Policy 4.';
 routers[5]:=' ~ Rout. Policy 5.';

end;{init_menus}

procedure draw_marco;
{ dibuja la pantalla standard }
begin
 setcolor (white);
 setlinestyle(solidln,0,normwidth);
 setviewport(0,0,maxx,maxy-14,clipon);
```

```
 setfillstyle (solidfill,white);
 rectangle(0,0,maxx,maxy-14);
 bar(0,0,maxx,12);
 setfillstyle(solidfill,red);
 bar(544,1,maxx-1,11);
 outtextxy(550,3,'BOLAS V0.0');
end;{ draw_marco }


procedure main_window(iniciox,inicioy,lineas:integer; var menu:disp;
                      var opt:integer;title:string);
{ crea una ventana de texto a partir de inicio (x minimo), y muestra
  "lineas" de menu. permite moverse a traves de las opciones del menu
  se selecciona con enter, se aborta con escape.El valor seleccionado
  se devuelve en opt }

type event = (none, change, select, out);

var p:pointer;
    opt1, xmin, xmax, ymin, ymax, y, i,alto,nn : integer;
    action:event;

function dentro:boolean;
{true si dentro de la ventana}
begin
 get_pos(m_stat,m_par);
 dentro:=((m_stat.xnuevo<=xmax) and (m_stat.xnuevo>=xmin) and
         (m_stat.ynuevo<=ymax) and (m_stat.ynuevo>=ymin))
end;

begin
 setfillstyle(solidfill,lightblue);
 setcolor(blue);
 if graphdriver = EGA then alto := 14
                      else alto := 8;
 xmin:=iniciox;
 xmax:=iniciox+300;

{ La ventana puede tener título }
 if title = '' then ymin:=inicioy
               else ymin:=inicioy+alto;
 ymax:=ymin+1+alto*(lineas+1);
 setfillstyle(solidfill,lightblue);
 bar(iniciox,inicioy,xmax,ymax);
 setcolor(white);
 rectangle(iniciox+2,inicioy+2,xmax-2,ymax-2);
 if title<>'' then begin
                    line(iniciox+2,inicioy+alto+2,
                         iniciox+298,inicioy+alto+2);
                    setcolor(yellow);
                    settextjustify(centertext,centertext);
                    outtextxy(iniciox+150,inicioy+3+alto div 2,title)
                    setcolor(white);
                   end;
 settextjustify(lefttext,toptext);
 for i:=1 to lineas do outtextxy(xmin,ymin+i*alto,menu[i]);
 opt:=1;
 action := change;
 initm(m_stat,m_par,ok,xmin+(xmax-xmin) div 2, ymin + alto);
```

```
 show_cursor(m_stat,m_par);
 repeat
  if action = change then
   begin  {fondo blanco letras rojas mayusculas}
    hide_cursor(m_stat,m_par);
    moveto (xmin,ymin+opt*alto);
    setcolor(white);
    outtextxy(xmin+3,ymin+opt*alto,' [[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[
    setcolor(red);
    for i:=1 to length(menu[opt]) do outtext(upcase(menu[opt,i]));
    show_cursor(m_stat,m_par);
   end;
 action := none;
 repeat
  if dentro then
   begin
    opt1:=((m_stat.ynuevo - ymin) div alto); {indice}
    if ((opt1 >0) and (opt1 <=lineas)) then
     begin
      if (m_stat.tecla1) then  action := select  {seleccion}
       else if (opt1<>opt) then action := change;{cambio de renglon}
     end;
   end
   else action := out;
  until (action <> none);   {hasta que pase algo}
  hide_cursor(m_stat,m_par);{reescribo renglon en modo normal}
  moveto (xmin,ymin+opt*alto);
  setcolor(lightblue);
  outtextxy(xmin+3,ymin+opt*alto,' [[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[
  setcolor(white);
  outtext(menu[opt]);
  show_cursor(m_stat,m_par);
  if action <> out then opt:=opt1
   else
    begin
     repeat until ((dentro) or (m_stat.tecla1));  {espero hasta entra
     action:=change;
    end;
  get_pos(m_stat,m_par);
 until (action = select)  ;
 setcolor(white);
 setfillstyle(solidfill,white);
 hide_cursor(m_stat,m_par);
 initm(m_stat,m_par,ok,m_stat.xnuevo, m_stat.ynuevo+30);
end;



procedure simulate;
{ procedimiento principal de simulacion. Utiliza rutinas definidas en
   Bolas4.pas.
}
var i,id1,id2,ntok:integer;
    step,stop: char;
    tam,x,y,speed:integer;
    yes,run,final:boolean;
    speed1,max_sim_time:longint;

procedure elim1(var base:pointr);
```

```
{
  Este procedimiento libera toda la memoria de la lista
  apuntada por base.
}

var p1:pointr;

begin
 while base^.next <> nil do
  begin
   p1:=base^.next;
   base^.next:=p1^.next;
   dispose(p1);
  end;
end;

procedure elim2(var base:arcpntr);
{
  Este procedimiento libera toda la memoria de la lista
  apuntada por base.
}

var p1:arcpntr;
begin
 while base^.next <> nil do
  begin
   p1:=base^.next;
   base^.next:=p1^.next;
   dispose(p1);
  end;
end;

procedure elim3 ( base: r_ptr);
begin
 while base^.next <> nil do delete_r(base,base^.next);
end;

procedure elim4 (base: lnk_ptr);
begin
 while base^.next <> nil do delete_l(base,base^.next);
end;

procedure init_node_store(p:archptr);
{
 Este procedimiento inicializa a cero las instancias del
 buffer de algorimto.
}
begin
 while p<> nil do
  begin
   p^.instancia:=0;
   if p^.nodo.class=3 then init_node_store(p^.macro);
   p:=p^.next;
  end;
end;

procedure inform(texto:string);
{
Escribe texto en la zona de informacion del simulador
```

```
}
begin
 setfillstyle (solidfill,black);
 settextjustify(centertext,centertext);
 bar(345,1,543,11);
 setcolor(yellow);
 outtextxy (444,7,texto);
 setcolor(white);
 settextjustify(lefttext,toptext);
end;

procedure poner_tokens;
{ inicialización manual del grafo de algoritmo }
var ii,k : integer;
    temp : element;
    i:archptr;
    p:stp_ptr;

begin
 window(1,25,80,25);
 clrscr;
 write('INIT MODE: Left button to put token, right button to end.');
 initm(m_stat,m_par,ok,40,40);
 set_hand_cursor;
 show_cursor(m_stat,m_par);
 repeat
  get_pos(m_stat,m_par);
  if m_stat.tecla1 then
   begin
    m_stat.tecla1:=false;
    i:=alg_buf;
    scan(m_stat.xnuevo,m_stat.ynuevo,i,is_in_circle);
    if is_in_circle then
     begin
      clrscr;
      write('Init alg. node number ',i^.nodo.id,'. Token count ?: ');
      empty;
      read(ntok);
      write('INIT MODE: Left button to put token, right button to end
      temp:=i^.nodo;
      case temp.input_policy of
       0: for k:=1 to ntok do { politica de entrada and }
          begin
           ii:=1;
           new(p);
           p^.id:=k;
           p^.next:=nil;
           repeat
            hide_cursor(m_stat,m_par);
            matching_unit(temp.inputs[ii].id,temp.id,p);
            show_cursor(m_stat,m_par);
            dispose(p);
            ii:=ii+1;
           until temp.inputs[ii].id=0;
          end;

       1: for k:=1 to ntok do { politica de entrada or }
          begin
           new(p);
```

```
            p^.id:=k;
            p^.next:=nil;
            hide_cursor(m_stat,m_par);
            matching_unit(temp.inputs[1].id,temp.id,p);
            show_cursor(m_stat,m_par);
            dispose(p);
           end;
       end;{case}
      m_stat.tecla1:=false;
     end;{if is in circle}
    end;{if m_stat.tecla1}
  until m_stat.teclar;
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 show_cursor(m_stat,m_par);
 clrscr;
end;{poner_tokens}

procedure attend;
{
**********************ATTEND*****************************
Este procedimiento atiende la linea de menu del simulador.
permite iniciar/terminar la simulacion en modo continuo/paso
a paso, cambiar on-line la velocidad de simulacion, y fijar
opciones de la sesion.
}
var out:boolean;
begin
 inform('WAITING');
 writeln;
 out:=false;
 settextjustify(centertext,centertext);
 settextstyle(defaultfont,horizdir,0);
 show_cursor(m_stat,m_par);
 repeat
  get_pos(m_stat,m_par);
  if (m_stat.ynuevo<=8) and m_stat.tecla1 then
   begin
    hide_cursor(m_stat,m_par);
    out:=true;
    case m_stat.xnuevo of
     0..48: begin {corrida libre}
             setfillstyle(solidfill,lightgray);
             setcolor(black);
             rectangle(1,1,47,11);
             floodfill(2,2,black);
             outtextxy(24,7,'RUN');
             setcolor(white);
             setfillstyle(solidfill,lightblue);
             floodfill(51,2,white);
             outtextxy(72,7,'STEP');
             paso_a_paso:=false;  { corrida libre }
             run:=true;           { iniciar corrida }
            end;
    49..96: begin {paso a paso}
             settextjustify(centertext,centertext);
             setcolor(black);
             rectangle(49,1,95,11);
             setfillstyle(solidfill,lightgray);
             floodfill(51,2,black);
```

```
            outtextxy(72,7,'STEP');
            setcolor(white);
            setfillstyle(solidfill,lightblue);
            floodfill(2,2,white);
            outtextxy(24,7,'RUN');
            paso_a_paso:=true;     { corrida paso a paso }
            run:=true;             { iniciar corrida }
           end;
 97..143:  if not final then
            begin    { terminar corrida }
             settextjustify(centertext,centertext);
             setcolor(black);
             rectangle(97,1,143,11);
             setfillstyle(solidfill,red);
             floodfill(98,2,black);
             setcolor(white);
             outtextxy(120,7,'STOP');
             floodfill(98,2,white);
             setfillstyle(solidfill,lightblue);
             floodfill(2,2,white);
             outtextxy(24,7,'RUN');
             floodfill(51,2,white);
             outtextxy(72,7,'STEP');
             final:=true;
            end;
169..192:  begin  { disminuir velocidad }
            setfillstyle(solidfill,magenta);
            floodfill(170,2,white);
            inform('SPEED DOWN');
            while m_stat.tecla1  do
             begin
              get_pos(m_stat,m_par);
              speed1:=speed1+1;
              if speed1>100000 then speed1:=100000;
              if speed1=100000 then
               begin
                inform ('MIN SPEED');
                repeat get_pos(m_stat,m_par) until not m_stat.tecla1
               end;
             end;
            setfillstyle(solidfill,lightblue);
            floodfill(170,2,white);
            speed:=speed1 div 100;
           end;

144..168:  begin   { aumentar velocidad }
            setfillstyle(solidfill,magenta);
            floodfill(146,2,white);
            inform('SPEED UP');
            while m_stat.tecla1 do
             begin
              get_pos(m_stat,m_par);
              speed1:=speed1-1;
              if speed1<0 then speed1:=0;
              if speed1=0 then
               begin
                inform ('MAX SPEED');
                repeat get_pos(m_stat,m_par) until not m_stat.tecla1
               end;
```

```
            end;
           setfillstyle(solidfill,lightblue);
           floodfill(146,2,white);
           speed:=speed1 div 100;
          end;

193..264: begin  { opciones de simulacion }
           inform('SET OPTIONS');
           empty;
           delay(100);
           ktext_window(193,3,opciones,option);
           case option of
               1: begin
                   clrscr;
                   write('Max. Simulation time is ',max_sim_time);
                   write('. Enter new value : ');
                   empty;
                   readln(max_sim_time);
                   clrscr;
                  end;

               2: begin
                   clrscr;
                   write('Trace is now ');
                   if trace_on then
                    begin
                     write('ENABLED to file ',archivo_trace);
                     espera;
                    end
                   else
                    begin
                     write('DISABLED. Change? (Y/N): ');
                     repeat
                      empty;
                      step:=readkey;
                      step:=upcase(step);
                     until ((step='Y') or (step='N'));
                     write(step);
                     empty;
                     if step='Y' then
                      begin
                       empty;
                       writeln;
                       write('Enter trace file name: ');
                       empty;
                       archivo_trace:='';
                       readln(archivo_trace);
                       if no_spaces(archivo_trace) <>'' then trace_
                       assign (tracfile,archivo_trace);
                       rewrite(tracfile);
                      end;
                     end;
                    clrscr;
                   end;

                3: begin
                    clrscr;
                    write('Graph animation is now ');
                    if animated then write('ON. ')
```

```
                                    else write ('OFF. ');
                        write('Change? (Y/N) : ');
                        repeat
                         empty;
                         step:=readkey;
                         step:=upcase(step);
                        until ((step='Y') or (step='N'));
                        write(step);
                        if step='Y' then animated:=not animated;
                        if (not animated) then speed:=0;
                        clrscr;
                       end;

                  end;
            end;
     end; {case}
    end;
 until out;
 if run then
  begin
   if paso_a_paso then inform('STEP RUNNING')
                  else inform('FREE RUNNING');
  end;
 if final then inform('SIMULATION STOPPED');
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 empty;
 settextjustify(lefttext,toptext);
end;


begin   { simulador }

{  Genero las colas. Estos punteros siempre apuntan al principio }
 if first_time_sim then { solo la primera vez }
  begin
   new (arc_queue);     { colas de arcos a ejecutar }
   new (match_store);   { almacenamiento de tokens parcialmente activa
   new (output_queue);  { cola de arcos ya ejecutados }
   new (token_queue);   { cola de tokens habilitados }
   new (exec_queue);    { cola de ejecucion. Entrada de la ex. unit }
   new (link_queue);    { cola de comunicaciones. Entrada de la ex. un
   new (router_queue);  { cola de ruteo }

   first_time_sim:=false;
  end;

{  Inicializacion de las colas }

 token_queue^.next  := nil;
 match_store^.next  := nil;
 arc_queue^.next    := nil;
 output_queue^.next := nil;
 exec_queue^.next   := nil;
 link_queue^.next   := nil;
 router_queue^.next := nil;

{  Punteros moviles a las colas }

 ptr2 := com_queue;
```

```
 ptr1 := output_queue;
 ptr0 := arc_queue;
 index:= match_store;
 index1 := token_queue;
 index2 := exec_queue;


{  Inicializaciones varias }

 t_simul := 0;                { tiempo de simulacion }
 max_sim_time:=$7fffffff;     { tiempo maximo de simulacion }
 final:=false;                { flag de finalizacion }
 paso_a_paso:=false;          { default: ejecucion continua }
 run:=false;                  { flag de comienzo de simulacion }
 animated:=true;              { flag de graficos animados }
 trace_on:=false;             { no hay salida a ningun arhivo de trace
 archivo_trace:='';           { archivo de trace }
 speed1:=5000;
 speed:= speed1 div 100;      { inicializo velocidad de simulacion }

{ calculo los factores de escala para que entren todos los graficos }
 scales(alg_buf^.next,alg_scalex,alg_scaley,alg_menorx,alg_menory);
 scales(arq_buf^.next,arq_scalex,arq_scaley,arq_menorx,arq_menory);

{ normalizo las coordenadas para que entren en pantalla }
 normalize(alg_buf,alg_scalex,alg_scaley,alg_menorx,alg_menory);
 normalize(arq_buf,arq_scalex,arq_scaley,arq_menorx,arq_menory);

{ marco como visibles solo los nodos del nivel superior }
 mark_visible_nodes(alg_buf^.next,true);

{ muestro el grafo en colores }
 show1;
 window(1,25,80,25);

 randomize;

{ pongo a cero las instancias}
 init_node_store(alg_buf);

{ inicializo el grafo }
 poner_tokens;
 show_cursor(m_stat,m_par);

{ espero la orden de comenzar o terminar }
 repeat attend until (run or final);

{ aqui vamos...}
 h_sched(yes);   { scheduling unit }
 if not yes then
  begin { hay algun problema de asignacion. vuelvo al menu ppal}
   clrscr;
   write('ERROR: modulo no asignado a procesador.');
   write(' Oprima cualquier tecla...');
   espera;
   exit;
  end;

 execute;        { exec unit }
```

```
 gotoxy(60,1);
 write(t_simul);


{ main loop }

 repeat
  get_pos(m_stat,m_par); { atiendo el menu }
  if (m_stat.tecla1 and (not final)) then attend;

{ a. extraigo token de la output_queue }
 ptr1 := output_queue^.next;

 while ptr1 <> nil do
  begin  { si hay algo en la lista entonces... }
   matching_unit (ptr1^.inicio, ptr1^.final,ptr1^.instancia);
   delet1 (output_queue, ptr1);
   ptr1 := output_queue^.next;
  end;

 if animated then delay(speed); { velocidad de simulacion }

 get_pos(m_stat,m_par);
 if (m_stat.tecla1 and (not final)) then attend;

 if paso_a_paso then  espera ;  { ejecucion paso a paso }

 h_sched(yes);
 if not yes then
  begin
   clrscr;
   write('ERROR: modulo no asignado a procesador.');
   write(' Oprima cualquier tecla...');
   espera;
   exit;
  end;

 get_pos(m_stat,m_par);
 if (m_stat.tecla1 and (not final)) then attend;

 execute; { paso de simulacion }

 get_pos(m_stat,m_par);
 if (m_stat.tecla1 and (not final)) then attend;


 gotoxy(60,1);
 write(t_simul);

 if t_simul>max_sim_time then final:=true;

{ condicion de finalizacion: que no existan mas eventos ejecutables
  -todas las colas vacias- o que se haya llegado al tiempo maximo }

 until ((exec_queue^.next = nil) and (arc_queue^.next = nil)
  and (output_queue^.next = nil) and (link_queue^.next = nil)
  and (token_queue^.next = nil) or final);
```

```
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);

 inform ('SIMULATION STOPPED');

 if trace_on then close (tracfile);

 clrscr;
 write('Simulation session finished at time ',t_simul,'.
        Press left button...' );
 repeat get_pos(m_stat,m_par) until m_stat.tecla1;

{ denormalizo las coordenadas del buffer del editor }
 denormalize(alg_buf,alg_scalex,alg_scaley,alg_menorx,alg_menory);
 denormalize(arq_buf,arq_scalex,arq_scaley,arq_menorx,arq_menory);

{ vuelvo a hacer los nodos invisibles }
 mark_visible_nodes(alg_buf^.next,false );

 settextstyle(defaultfont,horizdir,1);

{ vacio las colas del simulador }

 elim1(token_queue);
 elim1(exec_queue);
 elim1(match_store);
 elim2(arc_queue);
 elim2(output_queue);
 elim3(router_queue);
 elim4(link_queue);

end;


{
********************* HAND_ASSIGN ***************************
Asignacion estatica manual de tareas a procesadores.
Esta rutina genera un mapa de asignacion(map[]) formado por
pares tarea,procesador. El procedimiento es totalmente grafico
utilizando mouse.
version del 2/2/89.funciona.
version del 18/9/89. Modificada para asignar nodos macro.
}

procedure hand_assign (var map : assi);
var i,j,k,tam,half,fuente,winsiz,linex,dd1,d2,xlast,ylast:integer;
    x,y:integer;
numb:string[3];
d1,iyy:archptr;
mover,scre:pointr;
active,is_in_poly:boolean;
{este es el tacho de basura, para eliminar asignaciones}
const tacho:array [1..5] of pointtype=((x: 5;y:10),
                                       (x:27;y:10),
                                       (x:22;y:27),
                                       (x:10;y:27),
                                       (x: 5;y:10));


procedure show_asign(i:archptr;fcolor,wcolor:integer;p:archptr);
```

```
{muestra los nodos asignados al nodo i}
var k:integer;

begin
 k:=1;
 repeat
  if map[k].procesador=i^.nodo.id then
   begin
    d1:=p^.next;
    repeat
     if d1^.nodo.id=map[k].task then
      begin
       setcolor(fcolor);
       circle(d1^.nodo.x div 2, d1^.nodo.y,10);
       setfillstyle(solidfill,fcolor);
       floodfill(d1^.nodo.x div 2, d1^.nodo.y+5,fcolor);
       setcolor(wcolor);
       settextjustify(centertext,centertext);
       settextstyle(smallfont,horizdir,4);
       str(d1^.nodo.id,numb);
       outtextxy(d1^.nodo.x div 2,d1^.nodo.y,numb);
      end;
     d1:=d1^.next;
     until d1=nil;
    end;
    k:=k+1;
    until map[k].task=0;

end;

procedure show_assigned(p:archptr);
{ muestra en verde los nodos ya asignados }
var j:integer;

begin
 settextstyle(smallfont,horizdir,4);
 settextjustify(centertext,centertext);
 d1:=p^.next;
 while d1 <> nil do
  begin
   j:=1;
   repeat  {pinto de verde los nodos ya asignados}
   if d1^.nodo.id=map[j].task then
    begin
     setcolor(lightgreen);
     setfillstyle(solidfill,lightgreen);
     circle(d1^.nodo.x div 2 ,d1^.nodo.y,10);
     floodfill(d1^.nodo.x div 2,d1^.nodo.y+5,lightgreen);
     setcolor(red);
     str(d1^.nodo.id,numb);
     outtextxy(d1^.nodo.x div 2, d1^.nodo.y, numb);
    end;
   j:=j+1;
   until map[j].task=0;
   d1:=d1^.next;
  end;
end;

procedure asignar(p:archptr);
```

```
var a_sx,a_sy:real;
    a_mx,a_my,i,dd1,d2:integer;
    ixx:archptr;

begin
 InitM (m_stat,m_par,ok,40,40);
 Set_Hand_Cursor;

{ calculo los factores de escala para que entre todo }
 scales(p^.next,a_sx,a_sy,a_mx,a_my);

{ normalizo las coordenadas para que entren en pantalla }
 normalize(p,a_sx,a_sy,a_mx,a_my);

 setfillstyle(solidfill,black);
 bar(65,13,half-1,33);
 bar(1,33,half-1,maxy-15); {borro el area de algoritmo}
 xshow(p,algoritmo,0,0,2,1);
 show_assigned(p);
 Half := maxx div 2;
 Active := false;
 j:=0;
 repeat j:=j+1 until map[j].task=0;
 show_cursor(m_stat,m_par);
 settextstyle(smallfont,horizdir,4);
 settextjustify(centertext,centertext);
 repeat
  Get_Pos (m_stat,m_par);
  if (not active) and (m_stat.tecla1) then
   begin
    m_stat.tecla1:=false;
    ixx:=p;
    kscan(m_stat.xnuevo*2,m_stat.ynuevo,ixx,is_in_circle);
    if is_in_circle then
     begin
      if ixx^.nodo.class=3 then
       begin
        asignar(ixx^.macro);
        hide_cursor(m_stat,m_par);
        setfillstyle(solidfill,black);
        bar(65,13,half-1,33);
        bar(1,33,half-1,maxy-15); {borro el area de algoritmo}
        xshow(p,algoritmo,0,0,2,1);
        show_assigned(p);
        show_cursor(m_stat,m_par);
       end;
      active := true;
      fuente := ixx^.nodo.id;
      hide_cursor(m_stat,m_par);
      setcolor(lightgreen);
      circle(ixx^.nodo.x div 2,ixx^.nodo.y,10);
      xlast:=ixx^.nodo.x div 2;
      ylast:=ixx^.nodo.y;
      setfillstyle(solidfill,lightgreen);
      floodfill(ixx^.nodo.x div 2, ixx^.nodo.y+5,lightgreen);
      setcolor(red);
      str(ixx^.nodo.id,numb);
      settextjustify(centertext,centertext);
      settextstyle(smallfont,horizdir,4);
```

```
    outtextxy(ixx^.nodo.x div 2,ixx^.nodo.y,numb);
    getimage(xlast-8,ylast-8,xlast+8,ylast+8,mover^);
    getimage(xlast,ylast,xlast+16,ylast+16,scre^);
    putimage(xlast,ylast,mover^,normalput);
    setcolor(white);
    rectangle(xlast,ylast,xlast+16,ylast+16);
    getimage(xlast,ylast,xlast+16,ylast+16,mover^);
    x:=xlast;y:=ylast;
   end
  else
   begin
    iyy:=arq_buf;
    kscan((m_stat.xnuevo-half)*2,m_stat.ynuevo,iyy,is_in_poly);
    if is_in_poly then
     begin  {aca va una rutina para ver la asignacion}
      show_asign (iyy,lightred,white,p);
      initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
      set_hand_cursor;
      show_cursor(m_stat,m_par);
      repeat get_pos(m_stat,m_par) until m_stat.tecla1;
      show_asign(iyy,lightgreen,black,p);
      delay(200);
     end
    else if (m_stat.xnuevo<64) and (m_stat.ynuevo<32) and (m_stat.x
     then begin {clear:comienzo de nuevo}
      for i:=1 to maxnode do
       begin
        map [i].task :=0;
        map [i].procesador :=0;
       end;
      j:=1;
      hide_cursor(m_stat,m_par);
      xshow(p,algoritmo,0,0,2,1);
      show_cursor(m_stat,m_par);
     end;
   end;
  end;

if ((active) and (not m_stat.tecla1 )) then
 begin
  get_pos(m_stat,m_par);
  if(( m_stat.xnuevo <> x) or (m_stat.ynuevo <> y)) then
   begin
    putimage(x,y,scre^,normalput);
    x := m_stat.xnuevo;
    y := m_stat.ynuevo;
    getimage(x,y,x+16,y+16,scre^);
    putimage(x,y,mover^,normalput);
   end;
  end;

 if (active) and (m_stat.tecla1) then
  begin
   m_stat.tecla1:=false;
   iyy:=arq_buf;
   kscan((x-half)*2,y,iyy,is_in_poly);
   if (is_in_poly) and (iyy^.nodo.class=0) then
    begin
     Map[j].task := fuente;
```

```
        Map[j].procesador := iyy^.nodo.id;
        j := j+1;
        active:=false;
        delay(200);
        putimage(x,y,scre^,normalput);
        show_cursor(m_stat,m_par);
        m_stat.tecla1:=false;
       end;
      if (not is_in_poly) and (x<32) and (y<32) then
       begin
       for dd1:=1 to j do
        begin
         if map[dd1].task=fuente then
          begin
           for d2:=dd1 to j do map[d2]:=map[d2+1];
           map[j].task:=0;
           map[j].procesador:=0;
           j:=j-1;
          end;
        end;
       setcolor(white);
       circle(xlast,ylast,10);
       setfillstyle(solidfill,lightblue);
       floodfill(xlast,ylast+5,white);
       str(fuente,numb);
       outtextxy(xlast,ylast,numb);
       active:=false;
       putimage(x,y,scre^,normalput);
       show_cursor(m_stat,m_par);
      end;
  end;
 until m_stat.teclar;
 initm(m_stat,m_par,ok,m_stat.xnuevo,m_stat.ynuevo);
 set_hand_cursor;
 show_cursor(m_stat,m_par);
 denormalize(p,a_sx,a_sy,a_mx,a_my);
end;


begin

{ Inicializacion de la pantalla }
 ClearDevice;
 settextstyle(defaultfont,horizdir,0);
 draw_marco;
 setcolor(blue);
 setfillstyle(solidfill,black);
 bar(0,0,64,32);
 setcolor(white);
 line(32,0,32,32);
 rectangle(0,0,64,32);
 setwritemode(xorput);
 line(maxx div 2,0, maxx div 2, maxy-14);
 setwritemode(copyput);
 setfillstyle(hatchfill,white);
 drawpoly(5,tacho);
 fillpoly(5,tacho);
 outtextxy(37,13,'CLR');
 setcolor(black);
```

```
settextjustify(centertext,centertext);
outtextxy(maxx div 4,7,'Algoritmo');
outtextxy(3*maxx div 4,7,'Arquitectura');
window(1,25,80,25);
write('Boton derecho para terminar...');
setcolor(white);
tam:=imagesize(0,0,16,16);
getmem(mover,tam);
getmem(scre,tam);
{ok}

{ calculo los factores de escala para que entre todo }
scales(arq_buf^.next,arq_scalex,arq_scaley,arq_menorx,arq_menory);

{ normalizo las coordenadas para que entren en pantalla }
normalize(arq_buf,arq_scalex,arq_scaley,arq_menorx,arq_menory);

xshow(arq_buf,arquitectura,maxx div 2,0,2,1);

Half := maxx div 2;
Active := false;
j:=0;
repeat j:=j+1 until map[j].task=0;

asignar(alg_buf);

freemem(mover,tam);
freemem(scre,tam);
restorecrtmode;
textcolor(white);
textbackground(black);
writeln('                    Asignacion de Tareas a Procesadores:');
writeln('ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
         DDDDDDDDDDDDDDDDDDDDDDDDDDDDDD?');
for i:=1 to 21 do
writeln('3
writeln('@DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
         DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDY');
i:=1;
j:=1;
window(3,3,69,23);
textcolor(yellow);
textbackground(lightblue);
clrscr;
repeat
 writeln(' La tarea n#mero ',map[i].task:3,
         ' est  asignada al procesador n#mero ',map[i].procesador:3)
 i:=i+1;
 if i=21*j then
  begin
   textcolor(white);
   textbackground(black);
   window(1,25,80,25);
   clrscr;
   write('Boton izquierdo para continuar...');
   repeat get_pos(m_stat,m_par) until m_stat.tecla1;
   m_stat.tecla1:=false;
   writeln;
   textcolor(yellow);
```

```
    textbackground(lightblue);
    window(3,3,69,23);
    clrscr;
    j:=j+1;
   end;
  until map[i].task=0;
  window(1,25,80,25);
  textcolor(white);
  textbackground(black);
  writeln;
  write('Boton izquierdo para estar en BOLAS...');
  repeat get_pos(m_stat,m_par) until m_stat.tecla1;
  empty;
  setgraphmode(graphmode);

{ denormalizo las coordenadas del buffer del editor }

  denormalize(arq_buf,arq_scalex,arq_scaley,arq_menorx,arq_menory);

end;


begin {main: atencion del menu principal}

 archivo_salida:='';
 archivo_entrada:='';
 first_time_alg:=true;
 first_time_arq:=true;
 first_time_sim:=true;
 textcolor(white);
 textbackground(black);
 m_menu;
 initialize;
 setviewport(0,0,maxx,maxy-20,true);
 cleardevice;
 window(1,25,80,25);
 clrscr;

 for i:=1 to maxnode do asiga[i].task:=0;

 modo_de_operacion:=1; {modo default: algoritmo sin arquitectura}
{ =1 para asignaci"n estatica manual}
{ =2 para asignaci"n din mica autom tica}
{modo_de_operaci"n es una variable global}



{c: pongo el cursor}

 x:=80;
 y:=80;
 initm(m_stat,m_par,ok,x,y);
 if not ok then
  begin
   outtextxy(maxx div 2 - 80, maxy div 2 ,'Mouse no instalado. ');
   espera;
   halt(1);
  end;
 init_editor(alg_buf);
```

```
init_editor(arq_buf);
outtextxy(190,50,'Usted est  en BOLAS. Diviertase!!');
the_end:=false;   { si true, termina el programa }
repeat
 draw_marco;
 setcolor(black);
 rectangle(250,1,371,11);
 outtextxy(256,3,'Menu Principal');
 setcolor(white);
 main_window(170,70,5,mainmenu,option,'');
 case option of
  1: begin
      tipo:=algoritmo;
      gr_editor(alg_buf,tipo,0);
     end;
  2: begin
      tipo:=arquitectura;
      gr_editor(arq_buf,tipo,0);
     end;
  3: begin
      main_window(194,120,3,simenu,simopt,'Ocpiones de simulacion');
      case simopt of
       1: begin
           modo_de_operacion:=0;
           simulate;
          end;
       2: begin
           modo_de_operacion:=1;
           main_window(218,140,5,schedules,schnum,
                       'Politicas de asignacion');
           main_window(242,160,5,routers,routnum,
                       'Politicas de ruteo');
           simulate;
          end;
      end;
     end;
  4: begin
   { pido la confirmacion...un poco [un poco mas] de circo}
      bar(maxx div 2-100,maxy div 2 -15,
          maxx div 2 +100, maxy div 2 +15);
      setcolor (red);
      rectangle(maxx div 2-98, maxy div 2-13,
                maxx div 2+98, maxy div 2+13);
      settextjustify (centertext,centertext);
      repeat
       empty;
       repeat
        outtextxy(maxx div 2, maxy div 2,'Salir de BOLAS? (S/N)');
        delay(400);
        bar(maxx div 2-90,maxy div 2 -10,
            maxx div 2 +90, maxy div 2 +10);
        delay(300);
       until keypressed;
       cursor:=readkey;
       cursor:=upcase (cursor);
      until ((cursor='S') or (cursor='N'));
      if cursor = 'S' then
       begin
        eliminar(arq_buf);
```

```
          eliminar(alg_buf);
          The_end:=true;
         end;
        settextjustify (lefttext,toptext);
       end;
    5: hand_assign(asiga);
   end;
  cleardevice;
  initm(m_stat,m_par,ok,40,40);
  get_pos(m_stat,m_par);

  until The_end;
end.
```