



Departamento de Computación  
ALGORITMOS Y PROGRAMACION I - 75.40

02 de Julio de 2007

## “Central Telefónica”

Primer Cuatrimestre  
Año 2007  
Cátedra GARCIA - MENDEZ - BIANCHI  
Curso N° 004

Padrón	Apellido, Nombre	E-mail
88056	CIAN, Nicolás	<a href="mailto:ncian@fi.uba.ar">ncian@fi.uba.ar</a>
88494	HOOD, Pablo Cristian	<a href="mailto:phood@fi.uba.ar">phood@fi.uba.ar</a>
88094	GRAMAJO, Gustavo Alejandro	<a href="mailto:ggramajo@fi.uba.ar">ggramajo@fi.uba.ar</a>
88317	CARROZZO, Martín Sebastián	<a href="mailto:mcarrozzo@fi.uba.ar">mcarrozzo@fi.uba.ar</a>
87681	APARICIO, Mariano	<a href="mailto:maparicio@fi.uba.ar">maparicio@fi.uba.ar</a>
88295	CALCAGNO, Juan Ignacio	<a href="mailto:jcalcagno@fi.uba.ar">jcalcagno@fi.uba.ar</a>

## Resumen

Desarrollamos un programa pascal que tomando un archivo que recibimos mensualmente, realice los siguientes informes:

Las líneas que fueron mas y menos utilizadas,

El usuario hizo el mayor uso de las líneas, así como el que menor consumo realizo.

Los descuentos por excesos para usuarios que realizaron llamadas telefónicas sin permisos.

El tiempo promedio utilizado para cada línea

Detalles por tiempo de comunicación por semana.

Además este programa permite modificar la cantidad de usuarios, y además analizar si pueden realizar dichas llamadas, y administrar los permisos de los usuarios a dichas líneas. Y permite modificar la cantidad de líneas (baja y nuevas altas) y sus descripciones.

## Introducción

El programa se presenta como solución del siguiente problema:

La misma empresa de las cuatro líneas se decidió a cambiar de central telefónica, para ello decidió contratarnos nuevamente como consultores. La nueva central telefónica deberá tener las siguientes características:

- a) Debe ser capaz de manejar una cantidad ilimitada de líneas telefónicas.
- b) Debe permitir ingresar nuevas líneas, modificar las existentes y dar de baja a aquellas líneas telefónicas que ya no se utilizan.
- c) Además como se han detectado demasiadas llamadas telefónicas, la gerencia general solicitó que cada empleado tenga un nombre de usuario. Cada uno de estos usuarios podrá o no tener acceso a las distintas líneas.
- d) Cualquier usuario podrá hacer llamadas de cualquier línea pero si este no tiene acceso serán descontadas dichas llamadas de su sueldo.
- e) Mensualmente se recibe un archivo secuencial ordenado con los movimientos de todas las llamadas de todas las líneas.
- f) Se deberá permitir dar de alta, modificar y baja de usuarios.
- g) Se solicita que además se pueda:
  - 1) Informar: cantidad de tiempo por cada tipo de comunicación en toda la semana.
  - 2) Indicar la línea con mayor consumo.
  - 3) Indicar la de menor consumo.
  - 4) El valor promedio de cada llamada.
  - 5) Los descuentos por exceso a los usuarios, el mínimo y el máximo.

Tipos de llamadas: urbana, interurbana, celular e internacional.

Tipos de archivo:

- 1) Maestro de usuarios: acceso aleatorio.
- 2) Maestro de líneas: acceso secuencial ordenado.
- 3) Movimientos de líneas: acceso secuencial ordenado.

En caso de necesitar otros archivos justificar y decidir que tipo de acceso tendrían.

## Desarrollo

Consideraciones acerca del enunciado:

En el punto "e" nos dicen que recibimos un archivo mensualmente, mientras que en el punto "g-1" nos piden un informe semanal.

Consideramos presentar un informe mensual para el punto "g-1"

En el punto "g-4" nos piden mostrar el valor promedio de cada llamada, consideremos informar el *tiempo* promedio por tipo de llamada, porque no podemos promediar una sola llamada.

Planteo acerca de los archivos a utilizar, su estructura y tipos de datos.

➤ Archivo "*usuarios*" [aleatorio]

Este archivo contiene registros de la forma:

- user
- nombre
- tiempo: registro con tipos de llamadas
- descuento

Vamos a usar el campo "*user*" como clave principal, es decir, no pueden existir dos registros con el mismo campo "*user*"

➤ Archivo "*líneas*" [secuencial, ordenado]

Este archivo contiene registros de la forma:

- nombre de la línea
- descripción
- tiempo: registro con tipos de llamadas

En este caso la clave principal es "*nombre*"

El campo "*descripción*" lo ponemos porque en el enunciado ítem "b" dice explícitamente que podemos modificar los datos de una línea, así que este campo es para que tenga un poco de sentido querer modificar los datos de una línea.

➤ Archivo "*permisos*" [secuencial]

En este archivo están los permisos para cada usuario; sus registros son de la siguiente manera:

- nombre de user
- nombre de línea
- permiso tipo boolean

Este archivo nos brinda un mejor manejo y control de los permisos a la hora de dar de alta y modificar un usuario o línea.

Es secuencial debido a de el archivo "Líneas" es secuencial y este debe poder crecer al mismo ritmo.

Estos archivos van a ser necesarios pues en el enunciado se nos pide que el programa pueda manejar una cantidad ilimitada de líneas, lo que significa manejar una cantidad ilimitada de permisos. Los permisos no podemos ponerlos dentro del archivo "*usuarios*" como registro de registro, o como vector pues declararlos de esa manera seria limitar su capacidad de crecimiento, en el caso de un vector, su longitud máxima debe ser definida en tiempo de desarrollo y para los registros no conocemos métodos dinámicos para variar su estructura en tiempo de ejecución. Para ello, necesitamos que el contenedor de los datos de los permisos pueda crecer ilimitadamente (su única restricción es la capacidad del disco rígido) por este motivo utilizamos este archivo.

➤ Archivo "*movimientos*" [secuencial, ordenado]

Este es el archivo que recibimos mensualmente, nos gustaría que tenga la siguiente forma:

- user que realizó la llamada
- línea desde donde llamó
- tiempo de duración
- tipo de comunicación
- semana

El campo "semana" puede valer 1,2,3,4 y sirve para poder presentar un informe semanal de los movimientos.

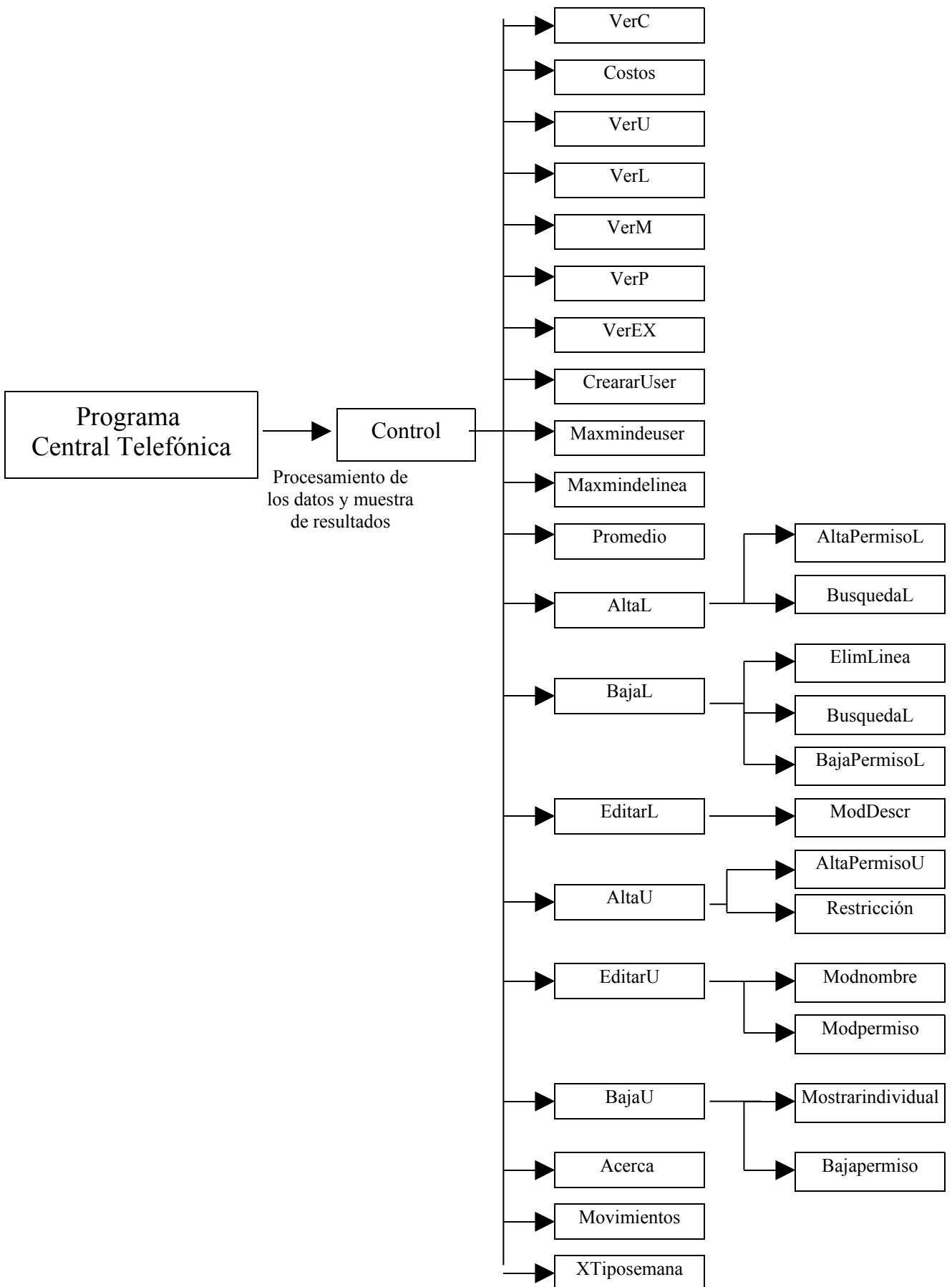
➤ Archivos "Costos"

- urbana: entero;
- interurbana: entero;
- celular: entero;
- internacional: entero;

Tenemos un único registro donde los campos son los tipos de comunicación y el archivo contiene el costo por minuto de esa comunicación.

Este archivo nos sirve para que el usuario pueda modificar los costos de las comunicaciones según varíen en el mercado los planes tarifarios, si los cargáramos como constantes en el programa, tendrían que pagar consultaría para modificar cada vez.

Hacemos un diagrama modular para tener en cuenta cuales son los módulos que necesitaríamos, cuales serian las constantes y como se administrarían dichos módulos (o procedimientos)



## Explicación de los procedimientos

- **ModPermiso:** (permite modificar los permisos de los usuarios)
- **EditarU:** (permite modificar los distintos datos del usuario, invocando otros procederes)
- **BajaU:** (permite dar de baja a un usuario)
- **BajaL:** (permite dar de baja a una línea con sus respectivos permisos)
- **AltaPermisoL:** (permite dar de alta un permiso a una línea)
- **BajaPermisoL:** (permite dar de baja un permiso a una línea)
- **AltaPermisoU:** (permite dar de alta un permiso a un usuario)
- **EditarL:** (hace invocaciones a otros procedimientos para realizar las modificaciones)
- **BajaPermisoU** (permite dar de baja un permiso a un usuario)
- **AltaL** (da de alta una nueva línea)
- **Control:** (en este procedimiento, tenemos el control de errores de nuestros archivos, y la parte gráfica de los menús principales, es desde aquí que se hacen las referencias a los demás procedimientos)
- **ModDescr:** (Procedimiento que permite modificar la descripción de las líneas)
- **ElimLinea:** (esta en la baja de líneas, Realiza un apareo, cuando se elimina una línea, para borrarlo del registro del archivo)
- **Maxmindescuento:** (evalúa los descuentos de los usuarios e informa el mayor y el menor)
- **Costos:** (con este proceso modificamos los costos de las comunicaciones, lo modifique)
- **Maxmindeuser:** (evalúa el máximo y mínimo consumo telefónico por parte de los empleados)
- **Promedio:** (muestra el promedio del tiempo consumido, en total)
- **VerP:** (permite ver los permisos de los distintos usuarios)
- **XTipoSemanal :** (muestra un promedio por semana por tipo de comunicación)
- **Maxmindelínea:** (Evalúa el tiempo total consumido por las líneas, y evalúa el máximo y el mínimo)
- **VerL:** (Nos permite ver a las líneas que están en nuestro programa)
- **Acerca:** (Muestra los autores del programa)
- **Restricción:** (Función que sirve para que no se ingresen usuarios vacíos)
- **VerEX:** (Permite ver los descuentos que haya por exceso)
- **VerU:** (Nos permite ver a los usuarios que están en nuestro programa)
- **BusquedaL:** (Busca líneas)
- **VerC:** (Permite ver los costos de las distintas llamadas)
- **VerM:** (Permite ver los movimientos)
- **EditarU:** (hace la invocación al procedimiento modnombre)
- **ModNombre:** (modifica el nombre del usuario)
- **Movimientos:** (este procedimiento analiza el archivo movimientos y analiza los registros de los usuarios y las líneas)
- **Creararuser:** (en este procedimiento generamos y mantenemos un control de nuestros usuarios)

CODIGO FUENTE

```
Program CentralTelefonica;
Uses
  NewDelay, Crt, Dos;
Const
  MAXUSER = 99;

  dia : array [0..6] of String[9] =
    ('Domingo','Lunes','Martes', 'Mi,rcoles','Jueves','Viernes','S bado');
  mes : array [0..11] of String[10] =
    ('Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Dici
embre');
Type
  Tdescripcion = string[20];
  Tuser = string[8];
  Tlinea = string[8];
  Tcomunicacion = (urbana, interurbana, celular, internacional);
  Tsemana = integer; { 1 - 2 - 3 - 4 }

  TregTiempo = record
    urbana: integer;
    interurbana: integer;
    celular: integer;
    internacional: integer;
  end;

  TregUsuarios = record
    NumUser: integer;
    user: Tuser;
    nombre: string[30];
    tiempo: TregTiempo;
    descuento: real;
  end;

  TregLineas = record
    nombre: Tlinea;
    descripcion: Tdescripcion;
    tiempo: TregTiempo;
  end;

  TregPermisos = record
    user: Tuser;
    linea: Tlinea;
    permiso: boolean;
  end;

  TregMovimientos = record
    user: Tuser;
    linea: Tlinea;
    tiempo: integer;
    comunicacion: Tcomunicacion;
    semana: Tsemana;
  end;
```

```
TregCostos = record
    urbana: integer;
    interurbana: integer;
    celular: integer;
    internacional: integer;
end;

TarUsuarios = file of TregUsuarios;

TarLineas = file of TregLineas;

TarPermisos = file of TregPermisos;

TarMovimientos = file of TregMovimientos;

TarCostos = file of TregCostos;
```

```
Procedure pimp(texto1, texto2:string; w,x,y:integer); Forward;
Procedure fecha; Forward;
Procedure hora(var X:char; a,b:integer); Forward;
Procedure caritas; Forward;
```

```
{-- MOVIMIENTOS -----}
```

```
Procedure Movimientos(var arUSER:TarUsuarios;var arLINEAS:TarLineas;var
arPERMISOS:TarPermisos;var
arMOV:TarMovimientos;var arCSTO:TarCostos);
```

```
var
```

```
Mov: TregMovimientos;
User: TregUsuarios;
Linea: TregLineas;
Permiso: TregPermisos;
Costo: TregCostos;
```

```
haymov, hayuser, haylinea, haypermiso: boolean;
estauser, estalineas, estapermiso: boolean;
```

```
i:integer;
Pos: integer;
cnt: integer;
aux: TregLineas;
arTEMP: TarLineas;
alghizo: boolean;
arHISTO: TarMovimientos;
```

```
begin
```

```
cnt:=0;
alghizo:=false;
haymov:=true;
hayuser:=true;
haylinea:=true;
haypermiso:=true;
```

```
estauser:=false;
estaline:=false;
estapermiso:=false;

clrscr; caritas;
pimp(' Inicializando MOVIMIENTOS',"0,1,2);
writeln;

{Cargo los costos}
reset(arCSTO);
{si el archivo esta vacio, cargo datos estandard}
if EOF(arCSTO) then
begin
  Costo.urbana:=1;
  Costo.interurbana:=1;
  Costo.celular:=1;
  Costo.internacional:=1;
  write(arCSTO,Costo)
end;

reset(arCSTO);
read(arCSTO,Costo);

reset(arMOV); {suponiendo que control verifico que ya existe}
if EOF(arMOV) then
begin
  writeln;
  textcolor(12+blink);
  writeln(' No hay movimientos nuevos!');
  writeln; normvideo;
  pimp(' para crear movimientos ejecute!' el archivo DIOS.exe',0,1,6);
  haymov:=false;
end;

if haymov then
begin

  reset(arUSER);
  if EOF(arUSER) then
  begin
    writeln(' aun no hay Usuarios!'); delay(300);
    hayuser:=false
  end;

  reset(arLINEAS);
  if EOF(arLINEAS) then
  begin
    writeln(' aun no hay Lineas!'); delay(300);
    haylinea:=false
  end;

  reset(arPERMISOS);
  if EOF(arPERMISOS) then
  begin
```

```
writeln(' aun no hay Permisos!'); delay(100);  
haypermiso:=false  
end;
```

```
end; {haymov}
```

```
if hayuser and haylinea and haypermiso then
```

```
begin
```

```
{Hago mientras no se termine el archivo arMOV}
```

```
while not EOF(arMOV) do
```

```
begin
```

```
inc(cnt);
```

```
writeln(' Analizando MOVIMIENTO: ',cnt); delay(300);
```

```
read(arMOV,Mov); {Leo un registro, y lo analizo}
```

```
i:=0;
```

```
seek(arUSER,i);
```

```
read(arUSER,User);
```

```
{mientras el usuario no coincide, cambio el usuario}
```

```
while not(Mov.user = User.user) and (i <= MAXUSER) do
```

```
begin
```

```
inc(i);
```

```
seek(arUSER,i);
```

```
read(arUSER,User);
```

```
end;
```

```
{aca ya coinciden los user}
```

```
if (Mov.user=User.user) then
```

```
begin
```

```
writeln(' USER "',Mov.user,'" encontrado'); delay(300);
```

```
estauser:=true;
```

```
end
```

```
else
```

```
writeln(' USER "',Mov.user,'" no encontrado'); delay(300);
```

```
reset(arLINEAS);
```

```
read(arLINEAS,Linea);
```

```
{mientras la linea no coincide, cambio de linea}
```

```
while not(Mov.linea = Linea.nombre) and not EOF(arLINEAS) do
```

```
begin
```

```
read(arLINEAS,Linea);
```

```
end;
```

```
{aca ya coinciden los user y las lineas}
```

```
if (Mov.linea=Linea.nombre) then
```

```
begin
```

```
writeln(' LINEA "',Mov.linea,'" encontrada'); delay(300);
```

```
estaline:=true;
```

```
end
```

```
else
```

```
writeln(' LINEA "',Mov.linea,'" no encontrada'); delay(300);
```

```
if estauser and estalinea then
begin

    {PERMISOS: si no tiene permiso para esta llamada, se lo cargo}

    write(' Buscando permisos'); delay(200);
    write('.'); delay(200); write('.'); delay(200); writeln('.'); delay(200);
    reset(arPERMISOS);
    read(arPERMISOS,Permiso);

    {mientras no coinciden "user" y "linea", busco otro registro de permisos}
    while not(Mov.user = Permiso.user) and not(Mov.linea = Permiso.linea) and not EOF(arPERMISOS) do
    begin
        read(arPERMISOS,Permiso);
    end;

    if (Mov.user=Permiso.user) and (Mov.linea=Permiso.linea) then
    begin
        writeln(' PERMISO encontrado'); delay(300);
        estapermiso:=true;
    end
    else
        writeln(' PERMISO no encontrado'); delay(300);

    if estapermiso then
    begin
        {al user le sumo el tiempo que hablo segun el tipo de comunicacion}
        case Mov.comunicacion of
            urbana:      User.tiempo.urbana := ( User.tiempo.urbana + Mov.tiempo );
            interurbana:  User.tiempo.interurbana := ( User.tiempo.interurbana + Mov.tiempo );
            celular:      User.tiempo.celular := ( User.tiempo.celular + Mov.tiempo );
            internacional: User.tiempo.internacional := ( User.tiempo.internacional + Mov.tiempo );
        end;

        {a la linea le sumo el tiempo que se hablo segun el tipo de comunicacion}
        case Mov.comunicacion of
            urbana:      Linea.tiempo.urbana := ( Linea.tiempo.urbana + Mov.tiempo );
            interurbana:  Linea.tiempo.interurbana := ( Linea.tiempo.interurbana + Mov.tiempo );
            celular:      Linea.tiempo.celular := ( Linea.tiempo.celular + Mov.tiempo );
            internacional: Linea.tiempo.internacional := ( Linea.tiempo.internacional + Mov.tiempo );
        end;

        writeln(' Tiempos actualizados'); delay(300);

        {en este punto "permiso" tiene el mismo user, y la misma linea que "mov"}
        {si es FALSE, osea, no tiene permiso, le descuento la llamada}
        if not(Permiso.permiso) then
        begin
            case (Mov.comunicacion) of
                urbana:      User.descuento := ( User.descuento + ( Mov.tiempo * Costo.urbana ) );
                interurbana:  User.descuento := ( User.descuento + ( Mov.tiempo * Costo.interurbana ) );
                celular:      User.descuento := ( User.descuento + ( Mov.tiempo * Costo.celular ) );
                internacional: User.descuento := ( User.descuento + ( Mov.tiempo * Costo.internacional ) );
            end;
        end;
    end;
end;
```

```
writeln(' Descuento actualizado'); delay(300);
end;

{actualizo los datos que obtuve, guardo en el archivo}
{ALEATORIO}
Pos:=( FilePos(arUSER) - 1 );
{grabo sobre el anterior}
Seek( arUSER, Pos );
Write(arUSER,User);

writeln(' Archivo USUARIOS actualizado'); delay(300);

{SECUENCIAL}
{tengo que actualizar el archivo}
assign(arTEMP, 'C:/TP/Files/templine.dat');
rewrite(arTEMP);
reset(arLINEAS);
read(arLINEAS, aux);
while not EOF(arLINEAS) and not (Linea.nombre = aux.nombre) do
begin
  write(arTEMP, aux);
  read(arLINEAS, aux);
end;
{en este punto va el registro nuevo}
write(arTEMP, Linea);
{ahora termino de poner los demas registros}
while not EOF(arLINEAS) do
begin
  read(arLINEAS, aux);
  write(arTEMP, aux);
end;
{el arTEMP esta actualizado}
{borro el archivo viejo}
CLOSE(arLINEAS);
ERASE(arLINEAS);
RENAME(arTEMP, 'C:/TP/Files/Lineas.dat');
reset(arLINEAS);

writeln(' Archivo LINEAS actualizado'); delay(300);

alghizo:=true;

end; {estapermiso}

end; {estauser estalinea}

end; {while arMOV}

end; {if hay--}

if alghizo then
begin
writeln(' Guardando historial'); delay(300);
assign(arHISTO, 'C:/TP/Files/Historia.dat');
```

```
{Si-}  
reset(arHISTO);  
{Si+}  
if IOResult <> 0 then rewrite(arHISTO);  
while not EOF(arMOV) do  
begin  
  read(arMOV, Mov);  
  write(arHISTO, Mov);  
end;  
writeln(' Borrando archivo viejo'); delay(300);  
rewrite(arMOV);  
writeln;  
TextColor(Yellow+Blink);  
write(' MOVIMIENTOS actualizados con exito!');  
NormVideo;  
end;  
readkey;  
  
end; {procedure}  
  
{-- PROMEDIO -----}  
  
Procedure Promedio(var arLineas:TarLineas);  
var  
  Sur, Siu, Sin, Sc: real;  
  divisor:integer;  
  aux: TregLineas;  
begin  
  clrscr; caritas;  
  Sur:= 0; Siu:=0;  
  Sin:=0; Sc:=0;  
  reset(arLineas);  
  while not( EOF(arLineas) ) do  
  begin  
    read(arlineas,aux);  
    Sur:=aux.tiempo.urbana+Sur;  
    Siu:=aux.tiempo.interurbana+Siu;  
    Sin:=aux.tiempo.internacional+Sin;  
    Sc:= aux.tiempo.celular+Sc;  
    divisor:=filesize(arlineas);  
  end;  
  writeln;  
  writeln('Promedio Urbana:      ',(Sur/divisor):2:2);  
  writeln('Promedio Interurbanas:  ',(Siu/divisor):2:2);  
  writeln('Promedio Internacional:  ',(Sin/divisor):2:2);  
  writeln('Promedio Celular:      ',(Sc/divisor):2:2);  
  readkey;  
end;  
  
{-- XTIPOSEMANAL -----}  
  
Procedure xTipoSemanal(var arMOV:TarMovimientos);  
var  
  Mov: TregMovimientos;
```

```
Sur, Siu, Sin, Sc: real;
cnt, cnt2: integer;
begin
  clrscr; caritas;
  writeln;
  writeln('Promedios para todas las lineas');
  writeln;
  writeln;
  cnt:=0;
  for cnt:=1 to 4 do
  begin
    Sur:=0; Siu:=0;
    Sin:=0; Sc:=0;
    cnt2:=0;
    reset(arMOV);
    while not(EOF(arMOV)) do
    begin
      read(arMOV, Mov);

      if Mov.semana = cnt then
      begin
        inc(cnt2);
        case Mov.comunicacion of
          urbana: Sur:=Mov.tiempo+Sur;
          interurbana: Siu:=Mov.tiempo+Siu;
          celular: Sin:=Mov.tiempo+Sin;
          internacional: Sc:= Mov.tiempo+Sc;
        end;
      end;
    end;
    if cnt2=0 then cnt2:=1; {evitamos el /0}
    writeln('Promedios semana: ',cnt);
    writeln;
    writeln('Urbana:      ',(Sur/cnt2):2:2);
    writeln('InterUrbana:   ',(Siu/cnt2):2:2);
    writeln('Celular:      ',(Sin/cnt2):2:2);
    writeln('InterNacional: ',(Sc/cnt2):2:2);
    writeln;

  end; {for}
  readkey;

end;

{-- MAXminUsuarios -----}

procedure maxminusuarios(var u:Tarusuarios);
var
  usuario: tregusuarios;
  maximo, minimo: integer;
  usermax, usermin: tuser;
  tiempototal: integer;
  i: integer;
begin
```

```
clrscr; caritas;
maximo:=0;
minimo:=0;
tiempototal:=0;
usermax:='ninguno';
usermin:='ninguno';
for i:=0 to MAXUSER-1 do
  begin
    seek(u,i);
    read(u,usuario);
    if usuario.user <> 'vacio' then
      tiempototal:= (usuario.tiempo.urbana + usuario.tiempo.interurbana +
usuario.tiempo.celular + usuario.tiempo.internacional);

      if tiempototal>maximo then
        begin
          maximo:= tiempototal;
          usermax:=usuario.user;
        end;
      if tiempototal<minimo then
        begin
          minimo:= tiempototal;
          usermin:=usuario.user;
        end;
      end;
      Writeln ('El Usuario con mayor consumo es ',usermax,' con un consumo de ',maximo,"");
      Writeln("");
      Writeln ('El Usuario con menor consumo es ',usermin,' con un consumo de ',minimo,"");
      readkey;
    end;
```

```
{-- MAXminLineas -----}
```

```
procedure maxminlineas(var l: tarLineas);
var
linea: treglineas;
mayor,menor: integer;
posicionmax, posicionmin: tlinea;
tiempototal: integer;
begin
  clrscr; caritas;
  mayor:=0;
  menor:=0;
  tiempototal:=0;
  posicionmax:='ninguna';
  posicionmin:='ninguna';
  reset (l);
  while not (eof (l)) do
    begin
      read (l, linea);
      tiempototal:= (linea.tiempo.urbana + linea.tiempo.interurbana
+ linea.tiempo.celular + linea.tiempo.internacional);

      if tiempototal>mayor then
```

```

begin
  mayor:= tiempototal;
  posicionmax:=linea.nombre;
end;
if tiempototal<menor then
begin
  menor:= tiempototal;
  posicionmin:=linea.nombre;
end;
end;
  Writeln ('El maximo se encuentra en la linea ',posicionmax,' con un total de ', mayor,' segundos
consumidos');
  Writeln("");
  Writeln ('El minimo se encuentra en la linea ',posicionmin,' con un total de ', menor,' segundos
consumidos');
  readkey;
end;

```

```
{-- VER -----}
```

```
{VER USUARIOS updated:vier29,, 18:12hs}
```

```
Procedure VerU(var aruser:tarusuarios);
```

```
var
```

```
  aux:tregusuarios;
```

```
  i:integer;
```

```
begin
```

```
  clrscr; caritas;
```

```
  reset(aruser);
```

```
  writeln('N$ User   Nombre           Urbana Interur Celular Int Desc');
```

```
  writeln;
```

```
  for i:=0 to (MAXUSER-1) do
```

```
  begin
```

```
    seek(aruser,i);
```

```
    read(aruser,aux);
```

```
    if aux.user <> 'vacio' then
```

```
      writeln(aux.numuser:2,aux.user:5,aux.nombre:16,aux.tiempo.urbana:14,aux.tiempo.interurbana:8,
```

```
aux.tiempo.celular:8,aux.tiempo.internacional:8,aux.descuento:8:1);
```

```
    end;
```

```
    readkey;
```

```
  end;
```

```
{VER LINEAS}
```

```
Procedure VerL(var arlineas:tarlineas);
```

```
var
```

```
  aux:treglineas;
```

```
  cont:integer;
```

```
begin
```

```
  clrscr; caritas;
```

```
  reset(arlineas);
```

```
  cont:=0;
```

```
  writeln('N$ Nombre   Descrip   Urbana Interur Celular Int');
```

```
  writeln;
```

```
while not eof(arlineas) do
begin
  read(arlineas,aux);
  inc(cont);
  writeln(cont:1,aux.nombre:8,aux.descripcion:10,aux.tiempo.urbana:11,aux.tiempo.interurbana:9,
aux.tiempo.celular:9,aux.tiempo.internacional:7);
  end;
  readkey;
end;
```

```
{VER MOVIMIENTOS}
```

```
Procedure VerM(var armov:tarmovimientos);
```

```
var
```

```
  aux:TregMovimientos;
```

```
  aux2:string[13];
```

```
  cont:integer;
```

```
begin
```

```
  clrscr; caritas;
```

```
  reset(armov);
```

```
  cont:=0;
```

```
  writeln('N§ User Linea Tiempo Comunicacion Semana');
```

```
  writeln;
```

```
  while not eof(armov) do
```

```
  begin
```

```
    read(armov,aux);
```

```
    inc(cont);
```

```
    case aux.comunicacion of
```

```
      urbana: aux2:='urbana';
```

```
      interurbana: aux2:='interurbana';
```

```
      celular: aux2:='celular';
```

```
      internacional: aux2:='internacional';
```

```
    end;
```

```
    writeln(cont:1,aux.user:10,aux.linea:14,aux.tiempo:8,aux2:12,aux.semana:6);
```

```
  end;
```

```
  readkey;
```

```
end;
```

```
{VER EXCESOS}
```

```
Procedure VerEX(var arUser:TarUsuarios);
```

```
var
```

```
  aux:Tregusuarios;
```

```
  i:integer;
```

```
begin
```

```
  clrscr; caritas;
```

```
  reset(arUser);
```

```
  writeln('N§ User Descuento');
```

```
  writeln;
```

```
  for i:=0 to MAXUSER do
```

```
  begin
```

```
    seek(arUSER,i);
```

```
    read(arUser,aux);
```

```
    writeln(aux.numuser:2,aux.user:8,aux.descuento:8);
```

```
end;
readkey;
end;

{VER PERMISOS}
Procedure VerP(var arper:tarpermisos);
var
  aux:tregpermisos;
  cont:integer;
begin
  clrscr; caritas;
  reset(arper);
  cont:=0;

  writeln('N§ User Linea Permiso');
  writeln;
  while not eof(arper) do
  begin
    read(arper,aux);
    inc(cont);
    writeln(cont:1,aux.user:8,aux.linea:12,aux.permiso:12);
  end;
  readkey;
end;

{VER COSTOS}
Procedure VerC(var arcos:tarcostos);
var
  aux:tregcostos;

begin
  clrscr; caritas;
  reset(arcos);
  writeln;
  writeln(' Urbana InterUrbana Celular InterNacional');
  writeln;
  read(arcos,aux);
  writeln(aux.urbana:6,aux.interurbana:11,aux.celular:16,aux.internacional:12);
  readkey;
end;

{-- COSTOS -----}

Procedure Costos(var arCSTO:TarCostos);
var
  costo:TregCostos;
  c:string;
  opcion:char;
  erroru:integer;
  valor:integer;
begin
  erroru:=0;
  VerC(arCSTO);
  reset(arCSTO);
```





```
Encontrado:=false;
Reset(arlineas);
X:=-1;
While (not eof(arlineas) and (not encontrado)) do
  Begin
    Read(arlineas,Raux);
    If Raux.nombre=linea then
      Begin
        encontrado:=true; {es -1 por que ya leyo y avanzo el puntero}
        X:=FilePos(arlineas)-1;
      end;
    end;
end;
```

```
Function Restriccion(Palabra:String):Boolean;
var
  i,b:integer;
  exit:boolean;
Begin
  b:=Length(Palabra);
  i:=1;
  exit:=false;
  while (not exit) and (i<=b)do
    begin
      If Palabra[i]=' ' then
        exit:=true;
        inc(i);
      end;
    If (b=0) or exit then
      Restriccion:=true
    else
      Restriccion:=false;
  end;
```

```
Procedure MostrarIndividual(Var MaeUsuarios:TArUsuarios;X:integer);
Var
  Raux:TRegUsuarios;
Begin
  clrscr;
  Seek(Maeusuarios,x);
  Read(MaeUsuarios,Raux);
  With Raux do
    Begin
      Writeln;
      Writeln('UserName: ',User);
      Writeln('Nombre y apellido: ',Nombre);
      Writeln('Consumo: ',tiempo.Urbana);
      Writeln('Consumo: ',tiempo.InterUrbana);
      Writeln('Consumo: ',tiempo.Celular);
      Writeln('Consumo: ',tiempo.Internacional);
      Writeln('Descuento: ',Descuento:0:1);
      Writeln;
    end;
```

end;

Procedure ModNombre(Var MaeUsuarios:TArUsuarios;X:integer);

Var

    NewName:TUser;

    Raux:TRegUsuarios;

Begin

    Writeln('Ingrese el nuevo Nombre');

    Readln(NewName);

    Seek(MaeUsuarios,X);

    Read(MaeUsuarios,Raux);

    Raux.Nombre:=NewName;

    Seek(MaeUsuarios,X);

    Write(MaeUsuarios,Raux);

    Writeln('Cambio realizado');

    Readkey;

end;

Procedure ModDescr(var arLINEAS:TarLineas; Linea: Tlinea);

var

    aux, aux2: TregLineas;

    exit: boolean;

    newline: Tarlineas;

    new: Tdescripcion;

begin

    assign(newline, 'C:/TP/Files/temp3.dat');

    rewrite(newline);

    reset(arLINEAS);

    read(arLINEAS, aux);

    while (not EOF(arLINEAS)) and (aux.nombre<>linea ) do

    begin

        write(newline, aux);

        read(arLINEAS, aux);

    end;

    if (aux.nombre=linea) then

        begin

            write('Ingrese nueva descripcion: ');

            readln(new);

            aux2:=aux;

            aux2.descripcion:=new;

            write(newline, aux2)

            end;

    while not EOF(arLINEAS) do

        begin

            read(arlineas, aux);

            write(newline, aux);

        end;

    close(arlineas);

    erase(arlineas);

    rename(newline, 'C:/TP/Files/Lineas.dat');

    reset(arlineas);

end;

```
{-- PERMISOS -----}  
{Le da permisos para usar todas las lineas a un usuario que se da de alta}  
procedure AltaPermisoU(var arPermisos: tarPermisos;var arLineas: tarLineas; nom: Tuser);  
var  
  permiso: tregPermisos;  
  linea: tregLineas;  
  {Archivo temporal de permisos}  
  arPermTemp : tarPermisos;  
begin  
  reset(arLineas);  
  reset(arPermisos);  
  {Creo un archivo nuevo, temporal, con los permisos nuevos, y lo abro}  
  assign(arPermTemp, 'C:/TP/Files/PermTemp.dat');  
  rewrite(arPermTemp);  
  {Primero copio el contenido del archivo viejo}  
  while not eof(arPermisos) do  
  begin  
    read(arPermisos, permiso);  
    write(arPermTemp, permiso);  
  end;  
  {Ahora copio los nuevos permisos}  
  permiso.user := nom;  
  while not eof(arLineas) do  
  begin  
    read(arLineas, linea);  
    permiso.linea:=linea.nombre;  
    permiso.permiso:=true;  
    write(arPermTemp, permiso);  
  end;  
  {Borro el viejo y lo remplazo por el nuevo}  
  close(arPermisos);  
  erase(arPermisos);  
  rename(arPermTemp,'C:/TP/Files/Permisos.dat');  
  reset(arPermisos);  
  writeln('El usuario tiene permisos globales');  
  writeln('editar en, Usuarios:Editar');  
end;  
  
{Todos los usuarios tienen permisos para usar una linea que se da de alta}  
procedure AltaPermisoL(var arPermisos: tarPermisos; var arUser: tarUsuarios; nom: tLinea);  
var  
  permiso: tregPermisos;  
  user: tregUsuarios;  
  {Archivo temporal de permisos}  
  arPermTemp: tarPermisos;  
begin  
  reset(arUser);  
  reset(arPermisos);  
  {Creo un archivo nuevo, temporal, con los permisos nuevos, y lo abro}  
  assign(arPermTemp, 'C:/TP/Files/PermTemp.dat');  
  rewrite(arPermTemp);  
  {Primero copio el contenido del archivo viejo}  
  while not eof(arPermisos) do  
  begin
```

```
read(arPermisos, permiso);
write(arPermTemp, permiso);
end;
{Ahora copio los nuevos permisos}
permiso.linea := nom;
while not eof(arUser) do
begin
read(arUser, user);
if user.user <> 'vacio' then
begin
permiso.user := user.user;
permiso.permiso := true;
write(arPermTemp, permiso);
end;
end;
{Borro el viejo y lo remplazo por el nuevo}
close(arPermisos);
erase(arPermisos);
rename(arPermTemp, 'C:/TP/Files/Permisos.dat');
reset(arPermisos);
writeln('Todos los usuarios tienen acceso');
writeln('editar en, Usuarios:Editar');
end;
```

Procedure BajaPermisoU(var arPERMISOS:TarPermisos; user: Tuser);

```
var
raux: TregPermisos;
NewPer: TarPermisos;
begin
reset(arPERMISOS);
assign(NewPer, 'C:/TP/Files/temp1.dat');
rewrite(Newper);

While (not eof(arPERMISOS)) do
Begin
Read(arPERMISOS, Raux);
If (Raux.User <> User) then Write(NewPer, Raux);
end;

Close(arPERMISOS);
Erase(arPERMISOS); {Se reemplaza el viejo arch}
Rename(NewPer, 'C:/TP/Files/Permisos.dat');
reset(arPermisos);

end;
```

Procedure BajaPermisoL(var arPERMISOS:TarPermisos; linea: Tlinea);

```
var
raux: TregPermisos;
NewPer: TarPermisos;
begin
reset(arPERMISOS);

Assign(NewPer, 'C:/TP/Files/temp1.dat');
```

```
Rewrite(NewPer);

While (not eof(arPERMISOS)) do
  Begin
    Read(arPERMISOS,Raux);
    If (Raux.linea<>linea) then Write(NewPer,Raux);
  end;

Close(arPERMISOS);
Erase(arPERMISOS); {Se reemplaza el viejo arch}
Rename(NewPer,'C:/TP/Files/Permisos.dat');
reset(arPermisos);

end;

procedure ModPermiso(var arPermisos: tarPermisos; nombre: tUser);
var
  permiso: TregPermisos;
  encontrado: boolean;
  linea: Tlinea;
  {Archivo temporal de permisos}
  arPermTemp: tarPermisos;
begin
  reset(arPermisos);
  {Creo un archivo nuevo, temporal, con los permisos nuevos, y lo abro}
  assign(arPermTemp, 'C:/TP/Files/PermTemp.dat');
  rewrite(arPermTemp);
  writeln;
  writeln;
  while not eof(arPermisos) do
    begin
      read(arPermisos, permiso);
      if nombre = permiso.user then writeln(permiso.linea:10, permiso.permiso:10);
    end;
    writeln;
    writeln('de que linea desea modificar el permiso?: ');
    read(linea);
    encontrado:= false;
    reset(arPermisos);
    while (not eof(arPermisos)) do
      begin
        read(arPermisos, permiso);
        if (linea = permiso.linea) and (nombre = permiso.user) then
          begin
            permiso.permiso := not permiso.permiso;
            encontrado:= true;
            write(arPermTemp, permiso);
            writeln('Permiso modificado');
          end
        else write(arPermTemp, permiso);
      end;
    if not encontrado then writeln('Linea no encontrada');
  close(arPermisos);
  erase(arPermisos);
```

```

rename(arPermTemp, 'C:/TP/Files/Permisos.dat');
reset(arPermisos);
readln;
end;

```

```

Procedure ElimLinea(Var arLINEAS:TArlneas;X:integer);

```

```

Var

```

```

    NewMae:TArlneas;
    Raux,RauxNew:TRegLineas;
    encontrado:boolean;
    I:integer;

```

```

Begin

```

```

    Assign(NewMae,'C:/TP/Files/temp2.dat');
    Rewrite(NewMae);
    reset(arLINEAS);
    for I:=1 to X+1 do read(arLINEAS,RauxNew);
    Reset(arLINEAS);
    encontrado:=false;
    write('.'); delay(200);
    While (not eof(arLINEAS)) and (not encontrado) do
        Begin
            Read(arLINEAS,Raux);
            If Raux.nombre=RauxNew.nombre then
                Encontrado:=true
            else
                Write(NewMae,Raux);
        end;
    write('.'); delay(200);
    While not eof(arLINEAS) do
        Begin {si el reg estaba en el final del arch esto no se ejecuta}
            Read(arLINEAS,raux);
            Write(NewMae,Raux);
        end;
    write('.'); delay(200);
    Close(arLINEAS);
    Erase(arLINEAS); {Se reemplaza el viejo arch}
    Rename(NewMae,'C:/TP/Files/Lineas.dat');
    reset(arLINEAS);
end;

```

```

end;

```

```

{----Procedimientos Alta Editar y Baja para los usuarios ----}

```

```

{-- ALTA USUARIO -----}

```

```

{updated:vier29 18:22hs}

```

```

Procedure AltaU(Var aruser:TArlusuarios; var arPERMISOS:TarPermisos; var arLINEAS:TarLineas);

```

```

Var

```

```

    Raux1,raux2:TRegusuarios;
    vacio:boolean;
    i: integer;

```

```

Begin

```

```

    clrscr; caritas;
    i:=0;
    vacio:=false;
    while (not vacio) and (i < MAXUSER) do

```

```

begin
    seek(aruser,i);
    read(aruser,raux1);
    if raux1.user = 'vacio' then vacio:= true;
    inc(i);
end;
if (not vacio) then
begin
    writeln('Archivo de usuarios lleno');
    readkey;
end
else
begin
    Writeln('Ingrese el Nombre de Usuario');
    Readln(Raux2.User);
    Writeln('Ingrese el Nombre');
    Readln(Raux2.Nombre);
    AltaPermisoU(arPERMISOS,arLINEAS,Raux2.user);
    Raux2.NumUser:=raux1.NumUser;
    Raux2.Tiempo.Urbana:=0;
    Raux2.Tiempo.InterUrbana:=0;
    Raux2.Tiempo.Celular:=0;
    Raux2.Tiempo.internacional:=0;
    Raux2.Descuento:=0;
    Seek(aruser,filepos(aruser)-1);
    write(aruser,raux2);
    Writeln;
    Writeln(Raux2.User,' es el usuario Nø',raux2.NumUser);
    readkey;
end;
end;

{-- BAJA USUARIO -----}
{updated:vier29 18:24hs}
Procedure BajaU(Var MaeUsuarios:TAUUsuarios; var arPERMISOS:TarPermisos);
Var
    S:string;
    N, Err:integer;
    Raux:tregusuarios;
    Opcion:char;
Begin
    clrscr; caritas;
    Write('Ingrese el Numero de Usuario del usuario que dara de baja: ');
    repeat
        Readln(S);
        val(S,N,Err); {validacion}
        if Err<>0 then writeln('Formato no valido, ingrese un numero. ');
    until Err=0;
    If (N <= MAXUSER) and (N >= 1) then
        begin
            MostrarIndividual(MaeUsuarios,N-1);
            Writeln('Confirma la eliminacion? (s/n)');
            Readln(Opcion);
            If Uppcase(Opcion)='S' then

```

```

Begin
  Seek(maeusuarios,N-1);
  read(maeusuarios,raux);
  BajaPermisoU(arPermisos,raux.user);
  Raux.user:='vacio';
  Raux.nombre:=' ';
  Seek(maeusuarios,N-1);
  write(maeusuarios,raux);
  Writeln('Usuario eliminado');{Blink}
end
else
  Begin
    clrscr;
    TextColor(yellow+blink);
    Writeln('Baja cancelada');{blink}
    NormVideo;
    Writeln;
  end;
  Writeln('Presione cualquier tecla para volver al Menu Principal');
  ReadKey;
end
else
  begin
    Writeln('Ingrese un numero de usuario valido (entre 1 y ',MAXUSER,')');
    readkey;
  end;
end;
end;

```

```

{-- EDITAR USUARIO -----}
{updated:vier29 18:25hs}
Procedure EditarU(Var MaeUsuarios:TArUsuarios; var arPermisos: TarPermisos);
Var
  s:string;
  Opcion:char;
  N,err:integer;
  Raux:TregUsuarios;
Begin
  clrscr; caritas;
  Write('Ingrese el numero del usuario al que desea modificar: ');
  repeat
    Readln(S);
    val(S,N,Err); {validacion}
    if Err<>0 then writeln('Formato no valido, ingrese un numero.');
```

```

until Err=0;
If (N <= MAXUSER) and (N >= 1) then
  Begin
    Seek(maeusuarios,N-1);
    Read(maeusuarios,Raux);
    If Raux.user <> 'vacio' then
      Begin
        clrscr;
        Opcion:='1';
        While Opcion<>'0' do

```

```

Begin
  MostrarIndividual(MaeUsuarios,N-1);
  Writeln('Que desea modificar?');
  Writeln;
  Writeln('[1] Nombre');
  Writeln('[2] Permisos');
  Writeln('[0] Volver');
  Readln(Opcion);
  Case Opcion of
    '1':ModNombre(MaeUsuarios,N-1);
    '2':ModPermiso(arPermisos,Raux.user);
    '0':clrscr;
  else
    Begin
      TextColor(yellow+blink);
      Writeln('Opcion incorrecta, intentelo nuevamente');
      NormVideo;
      Readkey;
      clrscr;
    end;
  end;
end;
end
else
  begin
    writeln('El usuario no existe o fue dado de baja');
    readkey;
  end;
end
else
  begin
    writeln('Ingrese un numero de usuario valido(entre 1 y ',MAXUSER,')');
    readkey;
  end;
end;
end;

```

```

{----Procedimientos Alta Editar y Baja para las lineas ----}
{-- ALTA LINEA -----}

```

Procedure AltaL(Var arlineas:TAlineas; var arPERMISOS:TarPermisos ;var arUSER:TarUsuarios);

Var

```

  Raux:TReglineas;
  X:integer;
  entrada:boolean;

```

Begin

```

  clrscr; caritas;
  while not eof(arlineas) do
  begin
  read(arlineas,raux);
  end;
  entrada:=true;
  While entrada do
  Begin
    Writeln('Ingrese el Nombre de la Linea:');

```

```

Readln(Raux.nombre);
If Not Restriccion(Raux.nombre) then
  Begin
    BusquedaL(arlineas,X,Raux.nombre); {X:=-1 si no hay coincidencia en ""Busqueda""}
    If X=-1 then
      Begin
        Writeln('Ingrese Descripcion:');
        Readln(Raux.descripcion);
        Raux.Tiempo.Urbana:=0;
        Raux.Tiempo.InterUrbana:=0;
        Raux.Tiempo.Celular:=0;
        Raux.Tiempo.internacional:=0;

        Write(arlineas,Raux);
        Writeln;
        TextColor(green+blink); {Blink}
        Writeln('Nueva linea agregada. ');
        NormVideo;
        AltaPermisoL(arPERMISOS,arUSER,Raux.nombre);
        entrada:=false;
      end
    else
      Begin
        clrscr;
        TextColor(yellow+blink);
        Writeln('Ese nombre ya existe, intentelo nuevamente');
        NormVideo;
      end;
    end
  else
    begin
      TextColor(red+blink);
      Writeln('Linea invalida');
      NormVideo;
      Writeln('este debe contener al menos un caracter y no terminar con un espacio');
    end;
  end;
  Readkey;
end;

```

```
{-- BAJA LINEA -----}
```

```
Procedure BajaL(Var arLineas:TArLineas; var arPERMISOS: TarPermisos);
```

```
Var
```

```
  DelLine:Tlinea;
```

```
  Raux:treglineas;
```

```
  Opcion:char;
```

```
  x,I:integer;
```

```
Begin
```

```
  clrscr; caritas;
```

```
  Writeln('Ingrese el Nombre de Linea que dara de baja');
```

```
  Readln(DelLine);
```

```
  BusquedaL(arLineas,X,DelLine);
```

```
  If X<>-1 then
```

```
Begin
  writeln(Raux.nombre);
  writeln;
  Writeln('Confirma la eliminacion? (s/n)');
  Readln(Opcion);
  If Uppcase(Opcion)='S' then
    Begin
      BajaPermisoL(arPermisos,delLine);
      ElimLinea(arLineas,X);
      Writeln('Linea eliminada');{Blink}

    end
  else
    Begin
      clrscr;
      TextColor(yellow+blink);
      Writeln('Baja cancelada');{blink}
      NormVideo;
      Writeln;
    end;
  end
else
  Begin
    TextColor(yellow+blink);
    Writeln('Nombre de Linea inexistente');
    NormVideo;
  end;
  Writeln('Presione cualquier tecla para volver al Menu Principal');
  ReadKey;
end;

{-- EDITAR LINEA -----}
```

```
Procedure EditarL(Var arLINEAS:TArLINEAS);
Var
  Opcion:integer;
  X, I:integer;
  LineaAux:Tlinea;
  entrada:boolean;
  Linea: TregLineas;
Begin
  clrscr;
  entrada:=true;
  While entrada do
    begin
      Reset(arLINEAS);
      Writeln('Ingrese el nombre de Lineas a realizar cambios');
      Readln(LineaAux);
      BusquedaL(arLINEAS,X,LineaAux);
      if x=-1 then
        begin
          TextColor(yellow+blink);
          Writeln('Nombre de Linea inexistente');
          NormVideo;
        end;
    end;
  end;
```

```

        end
        else
            entrada:=false;
        end;
    clrscr;
    Opcion:=1;
    While Opcion<>0 do
        Begin
            Writeln('Desea modificar la descripcion?');
            Writeln;
            Writeln('[1] Si');
            Writeln('[0] No');
            Readln(Opcion);
            Case Opcion of
                1:ModDescr(arLINEAS,LineaAux);
                0:clrscr;
            else
                Begin
                    TextColor(yellow+blink);
                    Writeln('Opcion incorrecta, intentelo nuevamente');
                    NormVideo;
                    Readkey;
                    clrscr;
                end;
            end;
        end;
    end;
end;

{-- Este procedure crea el archivo de usuarios -----}
{updated:vier29 18:27hs}
procedure CrearArUser(var aruser: tarusuarios);
var i:integer;
    aux: tregusuarios;
begin
    rewrite(aruser);
    aux.user:='vacio';
    for i:=0 to (MAXUSER-1) do
        begin
            seek(aruser,i);    {Los demas campos quedan con basura}
            aux.numuser:=i+1;  {pero como estan marcados como vacios}
            write(aruser,aux); {estos registros son ignorados hasta}
        end;                {que se ponga informacion relevante}
    end;
end;

{----- Fin de lo anterior -----}

{-- INTERFAZ -----}

Procedure pimp;
var
    I, C, D, T: integer;
begin
    case w of

```

```
0: T:=4;
1: T:=7;
end;
for I:=1 to T do
begin
  D:=200;
  case I of
    1,7: C:=0;
    2,6: C:=8;
    3,5: C:=7;
    4: begin C:=15; D:=1000; end;
  end;
  writeln;
  textcolor(C);
  gotoxy(x,y); write(texto1);
  gotoxy(x,y+1);write(texto2); delay(D);
end;
NormVideo;
end;
```

```
Procedure caritas;
begin
  gotoxy(77,49);
  textcolor(2); write(""); {4:red}
  textcolor(4); write(""); {1:blue}
  textcolor(1); write(""); {2:green}
  gotoxy(1,1);
  NormVideo;
end;
```

```
Procedure pos(var x,y:integer);
var
  r:real;
begin
  r:=random;
  if (r>=0) and (r<0.125) then inc(x);
  if (r>=0.125) and (r<0.25) then inc(y);
  if (r>=0.25) and (r<0.375) then begin inc(x); inc(y); end;
  if (r>=0.375) and (r<0.5) then x:=x-1;
  if (r>=0.5) and (r<0.625) then y:=y-1;
  if (r>=0.625) and (r<0.75) then begin inc(x); y:=y-1; end;
  if (r>=0.75) and (r<0.875) then begin inc(y); x:=x-1; end;
  if (r>=0.875) and (r<1) then begin x:=x-1; y:=y-1; end;
  if x<1 then x:=1;
  if x>80 then x:=80;
  if y<1 then y:=1;
  if y>48 then y:=48;
end;
```

```
Procedure Music(x:integer);
begin
  case x of
    1: begin
      sound(220); delay(150);
```

```
sound(230); delay(150);  
sound(220); delay(150);  
sound(250); delay(150);  
sound(220); delay(150);  
sound(270); delay(150);  
sound(220); delay(150);  
sound(290); delay(150); nosound; end;  
2: begin  
sound(300); delay(150);  
sound(320); delay(150);  
sound(330); delay(150);  
sound(340); delay(150);  
sound(330); delay(150);  
sound(320); delay(150);  
sound(300); delay(150);  
sound(300); delay(150); nosound; end;  
end;  
end;
```

Procedure ScreenSaver;

var

a, b, c, d, e, f, I: integer;

begin

patchcrt(crt.delay);

clrscr;

I:=0;

a:=40; b:=24;

c:=40; d:=24;

e:=40; f:=24;

Textcolor(7);

gotoxy(2,49); write('Central Telefonica - ScreenSaver

[press any key to exit]');

NormVideo;

TextColor(black);

Music(1);

repeat

pos(a,b); textcolor(2); {green}

gotoxy(a,b); writeln(""); {}

pos(c,d); textcolor(1); {blue}

gotoxy(c,d); writeln("");

pos(e,f); textcolor(4); {red}

gotoxy(e,f); writeln("");

textcolor(black); gotoxy(1,50); {para el cursor}

delay(150);

gotoxy(a,b); write(' ');

gotoxy(c,d); write(' ');

```
gotoxy(e,f); write(' ');

until KeyPressed;
Readkey;
Music(2);
NormVideo;
end;

Procedure hora;
var
Hor, Min, Seg, Dseg : Word;
Hs,Ms,Ss : String[2];
cont: integer;
key: boolean;
begin
GetTime(Hor,Min,Seg,Dseg);
cont:=Seg;
key:=false;
Repeat
GetTime(Hor,Min,Seg,Dseg); { Capturo la hora del sistema }
Str(Hor,Hs); Str(Min,Ms); Str(Seg,Ss);

If Hor<10 Then
Hs:='0'+Hs;

If Min<10 Then
Ms:='0'+Ms;

If Seg<10 Then
Ss:='0'+Ss;

Textcolor(7);GotoXY(a,b);
Write(Hs,':',Ms,':',Ss); gotoxy(1,50);

key:=keypressed;

if (Seg = 15+cont) then
begin ScreenSaver;
key:=true; X:='*'; end;

Until (key or keypressed);
if keypressed then X:=readkey;
end;

Procedure fecha;
var
y, m, d, dow : Word;
Begin
GetDate(y,m,d,dow); { Capturo la Fecha del sistema }
Textcolor(7); GotoXY(2,49);
WriteLn(dia[dow],',', ' d:0, ' de ', mes[m-1]:0,' de ', y:0);
End;
```

```
{-- CONTROL -----}
```

```
Procedure Control(var arUsuarios:TarUsuarios;var arLineas:TarLineas;var arPermisos:TarPermisos;var  
arMovimientos:TarMovimientos;var arCostos:TarCostos);
```

```
var
```

```
opcion, T, R:char;
```

```
arHisto: Tarmovimientos;
```

```
begin
```

```
{Vemos si tenemos directorio}
```

```
{Si-}
```

```
ChDir('C:/TP');
```

```
{Si+}
```

```
if IOResult <> 0 then Mkdir('C:/TP');
```

```
{Si-}
```

```
ChDir('C:/TP/Files');
```

```
{Si+}
```

```
if IOResult <> 0 then Mkdir('C:/TP/Files');
```

```
{Vemos si los archivos existen}
```

```
{Si-}
```

```
reset(arUsuarios); {updated:vier29 18:30hs}
```

```
{Si+}
```

```
if IOResult <> 0 then CrearArUser(arUsuarios); {Crea un archivo de acc. dir.}  
{con MAXUSER reg. vacios}
```

```
{Si-}
```

```
reset(arLineas);
```

```
{Si+}
```

```
if IOResult <> 0 then rewrite(arLineas);
```

```
{Si-}
```

```
reset(arPermisos);
```

```
{Si+}
```

```
if IOResult <> 0 then rewrite(arPermisos);
```

```
{Si-}
```

```
reset(arMovimientos);
```

```
{Si+}
```

```
if IOResult <> 0 then rewrite(arMovimientos);
```

```
{Si-}
```

```
reset(arCostos);
```

```
{Si+}
```

```
if IOResult <> 0 then rewrite(arCostos);
```

```
Assign(arHisto, 'C:/TP/Files/Historia.dat');
```

```
{Si-}
```

```
reset(arHisto);
```

```
{Si+}
```







end;

{-- BLOQUE PRINCIPAL -----}

Var

arUsuarios: TarUsuarios;  
arLineas: TarLineas;  
arPermisos: TarPermisos;  
arMovimientos: TarMovimientos;  
arCostos: TarCostos;

OrigMode: integer;

BEGIN

OrigMode:=LastMode;  
PatchCrt(Crt.Delay);  
TextMode(259); {usamos la pantalla que nos guste}

Randomize;

{Asignamos nombres para los archivos}

Assign(arUsuarios, 'C:/TP/Files/MaeUser.dat');  
Assign(arLineas, 'C:/TP/Files/Lineas.dat');  
Assign(arPermisos, 'C:/TP/Files/Permisos.dat');  
Assign(arMovimientos, 'C:/TP/Files/Movimien.dat');  
Assign(arCostos, 'C:/TP/Files/Costos.dat');

{Comenzamos}

Control(arUsuarios, arLineas, arPermisos, arMovimientos, arCostos);

{Cerramos todos los archivos}

Close(arUsuarios);  
Close(arLineas);  
Close(arPermisos);  
Close(arMovimientos);  
Close(arCostos);

TextMode(OrigMode);

END.

Otras consideraciones:

Además queremos evitar que el programa de un error en tiempo de ejecución por ingresar un dato que no sea acorde a lo declarado, para lo cual hacemos un tratamiento de errores y salvamos un posible problema al ingresar caracteres alfabéticos en lugar de numéricos, o datos numéricos fuera de rango.

Este procedimiento ha sido testeado con una gran variedad de ingresos, entre los cuales podemos destacar:

- Enteros fuera de rango
- Caracteres (letras, signos de puntuación, etc.).
- Espacios en blanco.
- Combinaciones de los grupos ya mencionados.
- Los resultados de estas pruebas no arrojaron errores.

## Prueba de escritorio

En esta prueba realizamos las capturas de pantalla de cada etapa principal del programa.

### Menú Principal

```
— CENTRAL TELEFONICA —  
  
Menu Principal  
  
1 : Archivos  
2 : Usuarios  
3 : Lineas  
  
A : Autores  
H : Ayuda  
  
S : Salir  
  
—13:01:57—
```

```
— CENTRAL TELEFONICA —  
  
Lineas  
  
1 : ver Lineas  
2 : Mayor y Menor consumo  
3 : Tiempo Promedio  
4 : por Tipo de com. Semanal  
  
5 : Alta  
6 : Editar  
7 : Baja  
  
M : Menu Principal  
  
—12:59:46—
```

```
— CENTRAL TELEFONICA —  
  
Usuarios  
  
1 : ver Usuarios  
2 : Mayor y Menor consumo  
3 : Descuentos por Excesos  
  
4 : Alta  
5 : Editar  
6 : Baja  
  
M : Menu Principal  
  
—13:03:23—
```

```
— CENTRAL TELEFONICA —  
  
Archivos  
  
1 : ver Costos  
2 : Editar Costos  
3 : ver Usuarios  
4 : ver Lineas  
5 : ver Movimientos  
6 : ver Permisos  
  
M : Menu Principal  
  
—13:02:40—
```

