

Trabajo Práctico N° 1: **Creación, Consulta y Mantenimiento de Archivos** **Secuenciales - Algorítmica Básica.**

Ejercicio 1.

Realizar un algoritmo que cree un archivo de números enteros no ordenados y permita incorporar datos al archivo. Los números son ingresados desde teclado. La carga finaliza cuando se ingresa el número 30000, que no debe incorporarse al archivo. El nombre del archivo debe ser proporcionado por el usuario desde teclado.

```
program TP1_E1;
{$codepage UTF8}
uses crt;
const
    num_salida=30000;
type
    t_archivo_enteros=file of int16;
procedure leer_numero(var num: int16);
var
    i: int8;
begin
    i:=random(100);
    if (i=0) then
        num:=num_salida
    else
        num:=random(high(int16));
    end;
end;
procedure cargar_archivo_enteros(var archivo_enteros: t_archivo_enteros);
var
    num: int16;
begin
    rewrite(archivo_enteros);
    textcolor(green); write('Los números ingresados son: ');
    leer_numero(num);
    while (num<>num_salida) do
    begin
        textcolor(yellow); write(num, ' ');
        write(archivo_enteros,num);
        leer_numero(num);
    end;
    close(archivo_enteros);
end;
var
    archivo_enteros: t_archivo_enteros;
begin
    randomize;
    assign(archivo_enteros,'enteros');
    cargar_archivo_enteros(archivo_enteros);
end.
```

Ejercicio 2.

Realizar un algoritmo que, utilizando el archivo de números enteros no ordenados creado en el Ejercicio 1, informe por pantalla cantidad de números menores a 1500 y el promedio de los números ingresados. El nombre del archivo a procesar debe ser proporcionado por el usuario una única vez. Además, el algoritmo deberá listar el contenido del archivo en pantalla.

```
program TP1_E2;
{$codepage UTF8}
uses crt;
const
    num_corte=1500;
type
    t_archivo_enteros=file of int16;
procedure procesar_archivo_enteros(var archivo_enteros: t_archivo_enteros; var nums_corte:
int16; var prom: real);
var
    num: int16;
    suma: real;
begin
    reset(archivo_enteros);
    suma:=0;
    textcolor(green); write('El contenido del archivo es: ');
    while (not eof(archivo_enteros)) do
        begin
            read(archivo_enteros,num);
            textcolor(yellow); write(num,' ');
            if (num<num_corte) then
                nums_corte:=nums_corte+1;
            suma:=suma+num;
        end;
    if (fileSize(archivo_enteros)>0) then
        prom:=suma/fileSize(archivo_enteros);
    writeln();
    close(archivo_enteros);
end;
var
    archivo_enteros: t_archivo_enteros;
    nums_corte: int16;
    prom: real;
begin
    nums_corte:=0; prom:=0;
    assign(archivo_enteros,'enteros');
    procesar_archivo_enteros(archivo_enteros,nums_corte,prom);
    textcolor(green); write('La cantidad de números menores a '); textcolor(yellow);
write(nums_corte); textcolor(green); write(' es '); textcolor(red); writeln(nums_corte);
    textcolor(green); write('El promedio de los números ingresados es '); textcolor(red);
write(prom:0:2);
end.
```

Ejercicio 3.

Realizar un programa que presente un menú con opciones para:

(a) Crear un archivo de registros no ordenados de empleados y completarlo con datos ingresados desde teclado. De cada empleado, se registra: número de empleado, apellido, nombre, edad y DNI. Algunos empleados se ingresan con DNI 00. La carga finaliza cuando se ingresa el String "fin" como apellido.

(b) Abrir el archivo anteriormente generado y:

- (i) Listar en pantalla los datos de empleados que tengan un nombre o apellido determinado, el cual se proporciona desde el teclado.
- (ii) Listar en pantalla los empleados de a uno por línea.
- (iii) Listar en pantalla los empleados mayores de 70 años, próximos a jubilarse.

Nota: El nombre del archivo a crear o utilizar debe ser proporcionado por el usuario.

```
program TP1_E3;
{$codepage UTF8}
uses crt;
const
    apellido_salida='fin';
    edad_corte=70;
    opcion_salida=0;
type
    t_string10=string[10];
    t_registro_empleado=record
        numero: int16;
        apellido: t_string10;
        nombre: t_string10;
        edad: int8;
        dni: int32;
    end;
    t_archivo_empleados=file of t_registro_empleado;
function random_string(length: int8): t_string10;
var
    i: int8;
    string_aux: string;
begin
    string_aux:='';
    for i:= 1 to length do
        string_aux:=string_aux+chr(ord('A')+random(26));
    end;
    random_string:=string_aux;
end;
procedure leer_empleado(var registro_empleado: t_registro_empleado);
var
    i: int8;
begin
    i:=random(100);
    if (i=0) then
        registro_empleado.apellido:=apellido_salida
    else
        registro_empleado.apellido:=random_string(5+random(5));
        if (registro_empleado.apellido<>apellido_salida) then
            begin
                registro_empleado.numero:=1+random(1000);
                registro_empleado.nombre:=random_string(5+random(5));
                registro_empleado.edad:=18+random(high(int8)-18);
```

```

    if (i<=10) then
        registro_empleado.dni:=0
    else
        registro_empleado.dni:=10000000+random(40000001);
    end;
end;
procedure cargar_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    rewrite(archivo_empleados);
    leer_empleado(registro_empleado);
    while (registro_empleado.apellido<>apellido_salida) do
        begin
            write(archivo_empleados,registro_empleado);
            leer_empleado(registro_empleado);
        end;
    close(archivo_empleados);
    textcolor(green); writeln('El archivo de empleados fue creado y cargado con éxito');
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
    textcolor(green); write('Número: '); textcolor(yellow); write(registro_empleado.numero);
    textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_empleado.apellido);
    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
    textcolor(green); write('; Edad: '); textcolor(yellow); write(registro_empleado.edad);
    textcolor(green); write('; DNI: '); textcolor(yellow); writeln(registro_empleado.dni);
end;
procedure imprimir1_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    texto: t_string10;
begin
    texto:=random_string(5+random(5));
    reset(archivo_empleados);
    textcolor(green); write('Los datos de los empleados con nombre o apellido ');
textcolor(yellow); write(texto); textcolor(green); writeln(' son: ');
    while (not eof(archivo_empleados)) do
        begin
            read(archivo_empleados,registro_empleado);
            if ((registro_empleado.nombre=texto) or (registro_empleado.apellido=texto)) then
                imprimir_registro_empleado(registro_empleado);
            end;
        close(archivo_empleados);
    end;
end;
procedure imprimir2_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    reset(archivo_empleados);
    textcolor(green); writeln(' Los empleados del archivo son: ');
    while (not eof(archivo_empleados)) do
        begin
            read(archivo_empleados,registro_empleado);
            imprimir_registro_empleado(registro_empleado);
        end;
    close(archivo_empleados);
end;
procedure imprimir3_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    reset(archivo_empleados);
    textcolor(green); write('Los empleados mayores a '); textcolor(yellow); write(edad_corte);
textcolor(green); writeln(' años son: ');

```

```
while (not eof(archivo_empleados)) do
begin
    read(archivo_empleados,registro_empleado);
    if (registro_empleado.edad>edad_corte) then
        imprimir_registro_empleado(registro_empleado);
    end;
    close(archivo_empleados);
end;
procedure leer_opcion(var opcion: int8);
begin
    textcolor(red); writeln('MENÚ DE OPCIONES');
    textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de empleados y completarlo con datos ingresados desde teclado');
    textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de empleados que tengan un nombre o apellido determinado');
    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
empleados de a uno por línea');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Listar en pantalla los
empleados mayores a 70 años, próximos a jubilarse');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_empleados: t_archivo_empleados);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
    begin
        case opcion of
            1: cargar_archivo_empleados(archivo_empleados);
            2: imprimir1_archivo_empleados(archivo_empleados);
            3: imprimir2_archivo_empleados(archivo_empleados);
            4: imprimir3_archivo_empleados(archivo_empleados);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
end;
var
    archivo_empleados: t_archivo_empleados;
begin
    randomize;
    assign(archivo_empleados,'empleados');
    menu_opciones(archivo_empleados);
end.
```

Ejercicio 4.

Agregar al menú del programa del Ejercicio 3 opciones para:

(a) *Añadir uno o más empleados al final del archivo con sus datos ingresados por teclado. Tener en cuenta que no se debe agregar al archivo un empleado con un número de empleado ya registrado (control de unicidad).*

(b) *Modificar la edad de un empleado dado.*

(c) *Exportar el contenido del archivo a un archivo de texto llamado "todos_empleados.txt".*

(d) *Exportar a un archivo de texto llamado "faltaDNIEmpleado.txt" los empleados que no tengan cargado el DNI (DNI en 00).*

Nota: Las búsquedas deben realizarse por número de empleado.

```

program TP1_E4;
{$codepage UTF8}
uses crt;
const
    apellido_salida='fin';
    edad_corte=70;
    dni_corte=0;
    opcion_salida=0;
type
    t_string10=string[10];
    t_registro_empleado=record
        numero: int16;
        apellido: t_string10;
        nombre: t_string10;
        edad: int8;
        dni: int32;
    end;
    t_archivo_empleados=file of t_registro_empleado;
function random_string(length: int8): t_string10;
var
    i: int8;
    string_aux: string;
begin
    string_aux:='';
    for i:= 1 to length do
        string_aux:=string_aux+chr(ord('A')+random(26));
    random_string:=string_aux;
end;
procedure leer_empleado(var registro_empleado: t_registro_empleado);
var
    i: int8;
begin
    i:=random(100);
    if (i=0) then
        registro_empleado.apellido:=apellido_salida
    else
        registro_empleado.apellido:=random_string(5+random(5));
        if (registro_empleado.apellido<>apellido_salida) then
            begin
                registro_empleado.numero:=1+random(1000);
                registro_empleado.nombre:=random_string(5+random(5));
            end;
        end;
end;

```

```

registro_empleado.edad:=18+random(high(int8)-18);
if (i<=10) then
    registro_empleado.dni:=0
else
    registro_empleado.dni:=10000000+random(40000001);
end;
end;
procedure cargar_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    rewrite(archivo_empleados);
    leer_empleado(registro_empleado);
    while (registro_empleado.apellido<>apellido_salida) do
        begin
            write(archivo_empleados,registro_empleado);
            leer_empleado(registro_empleado);
        end;
    close(archivo_empleados);
    textcolor(green); writeln('El archivo binario de empleados fue creado y cargado con éxito');
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
    textcolor(green); write('Número: '); textcolor(yellow); write(registro_empleado.numero);
    textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_empleado.apellido);
    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
    textcolor(green); write('; Edad: '); textcolor(yellow); write(registro_empleado.edad);
    textcolor(green); write('; DNI: '); textcolor(yellow); writeln(registro_empleado.dni);
end;
procedure imprimir1_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    texto: t_string10;
begin
    texto:=random_string(5+random(5));
    reset(archivo_empleados);
    textcolor(green); write('Los datos de los empleados con nombre o apellido ');
textcolor(yellow); write(texto); textcolor(green); writeln(' son: ');
    while (not eof(archivo_empleados)) do
        begin
            read(archivo_empleados,registro_empleado);
            if ((registro_empleado.nombre=texto) or (registro_empleado.apellido=texto)) then
                imprimir_registro_empleado(registro_empleado);
            end;
        end;
    close(archivo_empleados);
end;
procedure imprimir2_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    reset(archivo_empleados);
    textcolor(green); writeln(' Los empleados del archivo son: ');
    while (not eof(archivo_empleados)) do
        begin
            read(archivo_empleados,registro_empleado);
            imprimir_registro_empleado(registro_empleado);
        end;
    close(archivo_empleados);
end;
procedure imprimir3_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
begin
    reset(archivo_empleados);

```

```

    textcolor(green); write('Los empleados mayores a '); textcolor(yellow); write(edad_corte);
textcolor(green); writeln(' años son: ');
    while (not eof(archivo_empleados)) do
    begin
        read(archivo_empleados, registro_empleado);
        if (registro_empleado.edad > edad_corte) then
            imprimir_registro_empleado(registro_empleado);
        end;
    close(archivo_empleados);
end;
function control_unicidad(var archivo_empleados: t_archivo_empleados; numero: int16): boolean;
var
    registro_empleado: t_registro_empleado;
    ok: boolean;
begin
    ok:=false;
    while ((not eof(archivo_empleados)) and (ok=false)) do
    begin
        read(archivo_empleados, registro_empleado);
        if (registro_empleado.numero=numero) then
            ok:=true;
        end;
    control_unicidad:=ok;
end;
procedure agregar_empleado(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    empleados: int16;
begin
    empleados:=0;
    reset(archivo_empleados);
    leer_empleado(registro_empleado);
    while (registro_empleado.apellido<>apellido_salida) do
    begin
        if (control_unicidad(archivo_empleados, registro_empleado.numero)=false) then
        begin
            seek(archivo_empleados, filesize(archivo_empleados));
            write(archivo_empleados, registro_empleado);
            empleados:=empleados+1;
        end;
        leer_empleado(registro_empleado);
    end;
    close(archivo_empleados);
    textcolor(green); write('Se han agregado '); textcolor(yellow); write(empleados);
textcolor(green); writeln(' empleados al final del archivo');
end;
procedure modificar_edad_empleado(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    numero: int16;
    ok: boolean;
begin
    numero:=1+random(1000);
    ok:=false;
    reset(archivo_empleados);
    while ((not eof(archivo_empleados)) and (ok=false)) do
    begin
        read(archivo_empleados, registro_empleado);
        if (registro_empleado.numero=numero) then
        begin
            registro_empleado.edad:=18+random(high(int8)-18);
            seek(archivo_empleados, filepos(archivo_empleados)-1);
            write(archivo_empleados, registro_empleado);
            ok:=true;
        end;
    end;
end;

```



```

close(archivo_empleados);
if (ok=true) then
begin
    textcolor(green); write('Se ha modificado la edad del empleado con número ');
textcolor(yellow); write(numero); textcolor(green); writeln(' en el archivo');
end
else
begin
    textcolor(green); write('No se ha encontrado el empleado con número '); textcolor(yellow);
write(numero); textcolor(green); writeln(' en el archivo');
end;
end;
procedure exportar_archivo_txt1(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    archivo_txt: text;
begin
    reset(archivo_empleados);
    assign(archivo_txt, 'todos_empleados.txt'); rewrite(archivo_txt);
    while (not eof(archivo_empleados)) do
    begin
        read(archivo_empleados, registro_empleado);
        with registro_empleado do
            writeln(archivo_txt, 'Número: ', numero, '; Apellido: ', apellido, '; Nombre: ', nombre, ';
Edad: ', edad, '; DNI: ', dni);
        end;
        close(archivo_empleados);
        close(archivo_txt);
        textcolor(green); write('Se ha exportado el contenido del archivo al archivo de texto
llamado '); textcolor(yellow); writeln('todos_empleados.txt');
    end;
procedure exportar_archivo_txt2(var archivo_empleados: t_archivo_empleados);
var
    registro_empleado: t_registro_empleado;
    archivo_txt: text;
begin
    reset(archivo_empleados);
    assign(archivo_txt, 'faltaDNIEmpleado.txt'); rewrite(archivo_txt);
    while (not eof(archivo_empleados)) do
    begin
        read(archivo_empleados, registro_empleado);
        if (registro_empleado.dni=dni_corte) then
            with registro_empleado do
                writeln(archivo_txt, 'Número: ', numero, '; Apellido: ', apellido, '; Nombre: ', nombre, ';
Edad: ', edad, '; DNI: ', dni);
            end;
        close(archivo_empleados);
        close(archivo_txt);
        textcolor(green); write('Se ha exportado a un archivo de texto llamado ');
textcolor(yellow); write('faltaDNIEmpleado.txt'); textcolor(green); writeln(' los empleados
que no tienen cargado el DNI (DNI en 00)');
    end;
procedure leer_opcion(var opcion: int8);
begin
    textcolor(red); writeln('MENÚ DE OPCIONES');
    textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de empleados y completarlo con datos ingresados desde teclado');
    textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de empleados que tengan un nombre o apellido determinado');
    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
empleados de a uno por línea');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Listar en pantalla los
empleados mayores a 70 años, próximos a jubilarse');
    textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Añadir uno o más
empleados al final del archivo con sus datos ingresados por teclado');

```

```

    textcolor(yellow); write('OPCIÓN 6: '); textcolor(green); writeln('Modificar la edad de un
empleado dado');
    textcolor(yellow); write('OPCIÓN 7: '); textcolor(green); writeln('Exportar el contenido del
archivo a un archivo de texto llamado "todos_empleados.txt"');
    textcolor(yellow); write('OPCIÓN 8: '); textcolor(green); writeln('Exportar a un archivo de
texto llamado "faltaDNIEmpleado.txt" los empleados que no tengan cargado el DNI (DNI en 00)');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_empleados: t_archivo_empleados);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
    begin
        case opcion of
            1: cargar_archivo_empleados(archivo_empleados);
            2: imprimir1_archivo_empleados(archivo_empleados);
            3: imprimir2_archivo_empleados(archivo_empleados);
            4: imprimir3_archivo_empleados(archivo_empleados);
            5: agregar_empleado(archivo_empleados);
            6: modificar_edad_empleado(archivo_empleados);
            7: exportar_archivo_txt1(archivo_empleados);
            8: exportar_archivo_txt2(archivo_empleados);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
end;
var
    archivo_empleados: t_archivo_empleados;
begin
    randomize;
    assign(archivo_empleados, 'empleados');
    menu_opciones(archivo_empleados);
end.

```

Ejercicio 5.

Realizar un programa para una tienda de celulares, que presente un menú con opciones para:

(a) Crear un archivo de registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto denominado “celulares.txt”. Los registros correspondientes a los celulares deben contener: código de celular, nombre, descripción, marca, precio, stock mínimo y stock disponible.

(b) Listar en pantalla los datos de aquellos celulares que tengan un stock menor al stock mínimo.

(c) Listar en pantalla los celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el usuario.

(d) Exportar el archivo creado en el inciso (a) a un archivo de texto denominado “celulares.txt” con todos los celulares del mismo. El archivo de texto generado podría ser utilizado en un futuro como archivo de carga (ver inciso (a)), por lo que debería respetar el formato dado para este tipo de archivos en la Nota 2.

Nota 1: El nombre del archivo binario de celulares debe ser proporcionado por el usuario.

Nota 2: El archivo de carga debe editarse de manera que cada celular se especifique en tres líneas consecutivas. En la primera, se especifica: código de celular, precio y marca; en la segunda, stock disponible, stock mínimo y descripción; y, en la tercera, nombre en ese orden. Cada celular se carga leyendo tres líneas del archivo “celulares.txt”.

```
program TP1_E5;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=0;
  opcion_salida=0;
type
  t_string10=string[10];
  t_string20=string[20];
  t_registro_celular=record
    codigo: int16;
    nombre: t_string10;
    descripcion: t_string20;
    marca: t_string10;
    precio: real;
    stock_minimo: int16;
    stock_disponible: int16;
  end;
  t_archivo_celulares=file of t_registro_celular;
function random_string(length: int8): t_string10;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
```

```

    random_string:=string_aux;
end;
procedure leer_celular(var registro_celular: t_registro_celular);
var
    vector_marcas: array[1..10] of t_string10=('Alcatel', 'Apple', 'Huawei', 'Lenovo', 'LG',
'Motorola', 'Nokia', 'Samsung', 'Sony', 'Xiaomi');
    vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
    i: int8;
begin
    i:=random(100);
    if (i=0) then
        registro_celular.codigo:=codigo_salida
    else
        registro_celular.codigo:=1+random(1000);
        if (registro_celular.codigo<>codigo_salida) then
            begin
                registro_celular.nombre:=random_string(5+random(5));
                registro_celular.descripcion:=vector_descripciones[1+random(5)];
                registro_celular.marca:=vector_marcas[1+random(10)];
                registro_celular.precio:=100+random(9001)/10;
                registro_celular.stock_minimo:=1+random(10);
                registro_celular.stock_disponible:=random(101);
            end;
        end;
end;
procedure cargar_archivo_carga(var archivo_carga: text);
var
    registro_celular: t_registro_celular;
begin
    rewrite(archivo_carga);
    leer_celular(registro_celular);
    while (registro_celular.codigo<>codigo_salida) do
        begin
            with registro_celular do
                begin
                    writeln(archivo_carga,codigo,' ',precio:0:2,' ',marca);
                    writeln(archivo_carga,stock_disponible,' ',stock_minimo,' ',descripcion);
                    writeln(archivo_carga,nombre);
                end;
            leer_celular(registro_celular);
        end;
    close(archivo_carga);
end;
procedure cargar_archivo_celulares(var archivo_celulares: t_archivo_celulares; var
archivo_carga: text);
var
    registro_celular: t_registro_celular;
begin
    rewrite(archivo_celulares);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_celular do
            begin
                readln(archivo_carga,codigo,precio,marca); marca:=trim(marca);
                readln(archivo_carga,stock_disponible,stock_minimo,descripcion);
                descripcion:=trim(descripcion);
                readln(archivo_carga,nombre);
                write(archivo_celulares,registro_celular);
            end;
        close(archivo_celulares);
        close(archivo_carga);
        textcolor(green); writeln('El archivo binario de celulares fue creado y cargado con éxito');
    end;
end;
procedure imprimir_registro_celular(registro_celular: t_registro_celular);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_celular.codigo);

```

```

    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_celular.nombre);
    textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_celular.descripcion);
    textcolor(green); write('; Marca: '); textcolor(yellow); write(registro_celular.marca);
    textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_celular.precio:0:2);
    textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_celular.stock_minimo);
    textcolor(green); write('; Stock disponible: '); textcolor(yellow);
writeln(registro_celular.stock_disponible);
end;
procedure imprimir1_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
begin
    reset(archivo_celulares);
    textcolor(green); writeln('Los datos de aquellos celulares que tienen un stock menor al
stock mínimo son: ');
    while (not eof(archivo_celulares)) do
        begin
            read(archivo_celulares,registro_celular);
            if (registro_celular.stock_disponible<registro_celular.stock_minimo) then
                imprimir_registro_celular(registro_celular);
            end;
        end;
    close(archivo_celulares);
end;
procedure imprimir2_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
    descripcion: t_string20;
begin
    reset(archivo_celulares);
    descripcion:=vector_descripciones[1+random(5)];
    textcolor(green); write('Los celulares del archivo cuya descripción contiene la cadena de
caracteres '); textcolor(yellow); write(descripcion); textcolor(green); writeln(' son: ');
    while (not eof(archivo_celulares)) do
        begin
            read(archivo_celulares,registro_celular);
            if (registro_celular.descripcion=descripcion) then
                imprimir_registro_celular(registro_celular);
            end;
        end;
    close(archivo_celulares);
end;
procedure exportar_archivo_txt(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    archivo_txt: text;
begin
    reset(archivo_celulares);
    assign(archivo_txt,'celulares2.txt'); rewrite(archivo_txt);
    while (not eof(archivo_celulares)) do
        begin
            read(archivo_celulares,registro_celular);
            with registro_celular do
                begin
                    writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
                    writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
                    writeln(archivo_txt,nombre);
                end;
            end;
        end;
    close(archivo_celulares);
    close(archivo_txt);
end;

```

```

    textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
texto denominado '); textcolor(yellow); write('"celulares2.txt"'); textcolor(green); writeln('
con todos los celulares del mismo');
end;
procedure leer_opcion(var opcion: int8);
begin
    textcolor(red); writeln('MENÚ DE OPCIONES');
    textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto
denominado "celulares1.txt"');
    textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de aquellos celulares que tengan un stock menor al stock mínimo');
    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el
usuario');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "celulares2.txt" con todos los
celulares del mismo');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_celulares: t_archivo_celulares; var archivo_carga: text);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
        begin
            case opcion of
                1: cargar_archivo_celulares(archivo_celulares,archivo_carga);
                2: imprimir1_archivo_celulares(archivo_celulares);
                3: imprimir2_archivo_celulares(archivo_celulares);
                4: exportar_archivo_txt(archivo_celulares);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
var
    archivo_celulares: t_archivo_celulares;
    archivo_carga: text;
begin
    randomize;
    assign(archivo_carga,'celulares1.txt');
    assign(archivo_celulares,'celulares2.txt');
    cargar_archivo_carga(archivo_carga);
    menu_opciones(archivo_celulares,archivo_carga);
end.

```

Ejercicio 6.

Agregar al menú del programa del Ejercicio 5 opciones para:

- (a) *Añadir uno o más celulares al final del archivo con sus datos ingresados por teclado.*
- (b) *Modificar el stock de un celular dado.*
- (c) *Exportar el contenido del archivo binario a un archivo de texto denominado "SinStock.txt" con aquellos celulares que tengan stock 0.*

```

program TP1_E6;
{$codepage UTF8}
uses crt, sysutils;
const
    codigo_salida=0;
    stock_disponible_corte=0;
    opcion_salida=0;
type
    t_string10=string[10];
    t_string20=string[20];
    t_registro_celular=record
        codigo: int16;
        nombre: t_string10;
        descripcion: t_string20;
        marca: t_string10;
        precio: real;
        stock_minimo: int16;
        stock_disponible: int16;
    end;
    t_archivo_celulares=file of t_registro_celular;
function random_string(length: int8): t_string10;
var
    i: int8;
    string_aux: string;
begin
    string_aux:='';
    for i:= 1 to length do
        string_aux:=string_aux+chr(ord('A')+random(26));
    random_string:=string_aux;
end;
procedure leer_celular(var registro_celular: t_registro_celular);
var
    vector_marcas: array[1..10] of t_string10=('Alcatel', 'Apple', 'Huawei', 'Lenovo', 'LG',
'Motorola', 'Nokia', 'Samsung', 'Sony', 'Xiaomi');
    vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
    i: int8;
begin
    i:=random(100);
    if (i=0) then
        registro_celular.codigo:=codigo_salida
    else
        registro_celular.codigo:=1+random(1000);
    if (registro_celular.codigo<>codigo_salida) then
        begin
            registro_celular.nombre:=random_string(5+random(5));
            registro_celular.descripcion:=vector_descripciones[1+random(5)];
            registro_celular.marca:=vector_marcas[1+random(10)];
            registro_celular.precio:=100+random(9001)/10;
            registro_celular.stock_minimo:=1+random(10);

```

```

    registro_celular.stock_disponible:=random(101);
end;
end;
procedure cargar_archivo_carga(var archivo_carga: text);
var
    registro_celular: t_registro_celular;
begin
    rewrite(archivo_carga);
    leer_celular(registro_celular);
    while (registro_celular.codigo<>codigo_salida) do
    begin
        with registro_celular do
        begin
            writeln(archivo_carga,codigo,' ',precio:0:2,' ',marca);
            writeln(archivo_carga,stock_disponible,' ',stock_minimo,' ',descripcion);
            writeln(archivo_carga,nombre);
        end;
        leer_celular(registro_celular);
    end;
    close(archivo_carga);
end;
procedure cargar_archivo_celulares(var archivo_celulares: t_archivo_celulares; var
archivo_carga: text);
var
    registro_celular: t_registro_celular;
begin
    rewrite(archivo_celulares);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_celular do
        begin
            readln(archivo_carga,codigo,precio,marca); marca:=trim(marca);
            readln(archivo_carga,stock_disponible,stock_minimo,descripcion);
descripcion:=trim(descripcion);
            readln(archivo_carga,nombre);
            write(archivo_celulares,registro_celular);
        end;
    close(archivo_celulares);
    close(archivo_carga);
    textcolor(green); writeln('El archivo binario de celulares fue creado y cargado con éxito');
end;
procedure imprimir_registro_celular(registro_celular: t_registro_celular);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_celular.codigo);
    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_celular.nombre);
    textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_celular.descripcion);
    textcolor(green); write('; Marca: '); textcolor(yellow); write(registro_celular.marca);
    textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_celular.precio:0:2);
    textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_celular.stock_minimo);
    textcolor(green); write('; Stock disponible: '); textcolor(yellow);
writeln(registro_celular.stock_disponible);
end;
procedure imprimir1_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
begin
    reset(archivo_celulares);
    textcolor(green); writeln('Los datos de aquellos celulares que tienen un stock menor al
stock mínimo son: ');
    while (not eof(archivo_celulares)) do
    begin
        read(archivo_celulares,registro_celular);
        if (registro_celular.stock_disponible<registro_celular.stock_minimo) then

```



```

        imprimir_registro_celular(registro_celular);
    end;
    close(archivo_celulares);
end;
procedure imprimir2_archivo_celulares(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    vector_descripciones: array[1..5] of t_string20=('Gama baja', 'Gama media baja', 'Gama
media', 'Gama media alta', 'Gama alta');
    descripcion: t_string20;
begin
    reset(archivo_celulares);
    descripcion:=vector_descripciones[1+random(5)];
    textcolor(green); write('Los celulares del archivo cuya descripción contiene la cadena de
caracteres '); textcolor(yellow); write(descripcion); textcolor(green); writeln(' son: ');
    while (not eof(archivo_celulares)) do
        begin
            read(archivo_celulares,registro_celular);
            if (registro_celular.descripcion=descripcion) then
                imprimir_registro_celular(registro_celular);
            end;
        end;
    close(archivo_celulares);
end;
procedure exportar_archivo_txt1(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    archivo_txt: text;
begin
    reset(archivo_celulares);
    assign(archivo_txt,'celulares2.txt'); rewrite(archivo_txt);
    while (not eof(archivo_celulares)) do
        begin
            read(archivo_celulares,registro_celular);
            with registro_celular do
                begin
                    writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
                    writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
                    writeln(archivo_txt,nombre);
                end;
            end;
        end;
    close(archivo_celulares);
    close(archivo_txt);
    textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
texto denominado '); textcolor(yellow); write('"celulares2.txt"'); textcolor(green); writeln('
con todos los celulares del mismo');
end;
function control_unicidad(var archivo_celulares: t_archivo_celulares; codigo: int16): boolean;
var
    registro_celular: t_registro_celular;
    ok: boolean;
begin
    ok:=false;
    while ((not eof(archivo_celulares)) and (ok=false)) do
        begin
            read(archivo_celulares,registro_celular);
            if (registro_celular.codigo=codigo) then
                ok:=true;
            end;
        end;
    control_unicidad:=ok;
end;
procedure agregar_celular(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    celulares: int16;
begin
    celulares:=0;

```

```

reset(archivo_celulares);
leer_celular(registro_celular);
while (registro_celular.codigo<>codigo_salida) do
begin
    if (control_unicidad(archivo_celulares,registro_celular.codigo)=false) then
    begin
        seek(archivo_celulares,filesize(archivo_celulares));
        write(archivo_celulares,registro_celular);
        celulares:=celulares+1;
    end;
    leer_celular(registro_celular);
end;
close(archivo_celulares);
textcolor(green); write('Se han agregado '); textcolor(yellow); write(celulares);
textcolor(green); writeln(' celulares al final del archivo');
end;
procedure modificar_stock_celular(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    codigo: int16;
    ok: boolean;
begin
    codigo:=1+random(1000);
    ok:=false;
    reset(archivo_celulares);
    while ((not eof(archivo_celulares)) and (ok=false)) do
    begin
        read(archivo_celulares,registro_celular);
        if (registro_celular.codigo=codigo) then
        begin
            registro_celular.stock_disponible:=random(101);
            seek(archivo_celulares,filepos(archivo_celulares)-1);
            write(archivo_celulares,registro_celular);
            ok:=true;
        end;
    end;
    close(archivo_celulares);
    if (ok=true) then
    begin
        textcolor(green); write('Se ha modificado el stock del celular con código ');
        textcolor(yellow); write(codigo); textcolor(green); writeln(' en el archivo');
    end
    else
    begin
        textcolor(green); write('No se ha encontrado el celular con código '); textcolor(yellow);
        write(codigo); textcolor(green); writeln(' en el archivo');
    end;
end;
procedure exportar_archivo_txt2(var archivo_celulares: t_archivo_celulares);
var
    registro_celular: t_registro_celular;
    archivo_txt: text;
begin
    reset(archivo_celulares);
    assign(archivo_txt,'SinStock.txt'); rewrite(archivo_txt);
    while (not eof(archivo_celulares)) do
    begin
        read(archivo_celulares,registro_celular);
        if (registro_celular.stock_disponible=stock_disponible_corte) then
        with registro_celular do
        begin
            writeln(archivo_txt,codigo,' ',precio:0:2,' ',marca);
            writeln(archivo_txt,stock_disponible,' ',stock_minimo,' ',descripcion);
            writeln(archivo_txt,nombre);
        end;
    end;
end;

```

```

close(archivo_celulares);
close(archivo_txt);
textcolor(green); write('Se ha exportado el contenido del archivo binario a un archivo de
texto denominado '); textcolor(yellow); write('"SinStock.txt"'); textcolor(green); writeln('
con aquellos celulares que tienen stock 0');
end;
procedure leer_opcion(var opcion: int8);
begin
    textcolor(red); writeln('MENÚ DE OPCIONES');
    textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo de
registros no ordenados de celulares y cargarlo con datos ingresados desde un archivo de texto
denominado "celulares1.txt"');
    textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de aquellos celulares que tengan un stock menor al stock mínimo');
    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
celulares del archivo cuya descripción contenga una cadena de caracteres proporcionada por el
usuario');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "celulares2.txt" con todos los
celulares del mismo');
    textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Añadir uno o más
celulares al final del archivo con sus datos ingresados por teclado');
    textcolor(yellow); write('OPCIÓN 6: '); textcolor(green); writeln('Modificar el stock de un
celular dado');
    textcolor(yellow); write('OPCIÓN 7: '); textcolor(green); writeln('Exportar el contenido del
archivo binario a un archivo de texto denominado "SinStock.txt" con aquellos celulares que
tengan stock 0');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_celulares: t_archivo_celulares; var archivo_carga: text);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
        begin
            case opcion of
                1: cargar_archivo_celulares(archivo_celulares,archivo_carga);
                2: imprimir1_archivo_celulares(archivo_celulares);
                3: imprimir2_archivo_celulares(archivo_celulares);
                4: exportar_archivo_txt1(archivo_celulares);
                5: agregar_celular(archivo_celulares);
                6: modificar_stock_celular(archivo_celulares);
                7: exportar_archivo_txt2(archivo_celulares);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
end;
var
    archivo_celulares: t_archivo_celulares;
    archivo_carga: text;
begin
    randomize;
    assign(archivo_carga,'celulares1.txt');
    assign(archivo_celulares,'celulares2');
    cargar_archivo_carga(archivo_carga);
    menu_opciones(archivo_celulares,archivo_carga);
end.

```

Ejercicio 7.

Realizar un programa que permita:

(a) *Crear un archivo binario a partir de la información almacenada en un archivo de texto. El nombre del archivo de texto es: “novelas.txt”. La información en el archivo de texto consiste en: código de novela, nombre, género y precio de diferentes novelas argentinas. Los datos de cada novela se almacenan en dos líneas en el archivo de texto. La primera línea contendrá la siguiente información: código novela, precio y género; y la segunda línea almacenará el nombre de la novela.*

(b) *Abrir el archivo binario y permitir la actualización del mismo. Se debe poder agregar una novela y modificar una existente. Las búsquedas se realizan por código de novela.*

Nota: El nombre del archivo binario es proporcionado por el usuario desde el teclado.

```
program TP1_E7;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=0;
  opcion_salida=0;
type
  t_string20=string[20];
  t_registro_novela=record
    codigo: int16;
    nombre: t_string20;
    genero: t_string20;
    precio: real;
  end;
  t_archivo_novelas=file of t_registro_novela;
function random_string(length: int8): t_string20;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
    random_string:=string_aux;
  end;
procedure leer_novela(var registro_novela: t_registro_novela; ok: boolean);
var
  i: int8;
begin
  if (ok=true) then
  begin
    i:=random(100);
    if (i=0) then
      registro_novela.codigo:=codigo_salida
    else
      registro_novela.codigo:=1+random(1000);
    end;
    if (registro_novela.codigo<>codigo_salida) then
    begin
      registro_novela.nombre:=random_string(5+random(15));
      registro_novela.genero:=random_string(5+random(15));
      registro_novela.precio:=100+random(9001)/10;
    end;
  end;
end;
```

```
procedure cargar_archivo_carga(var archivo_carga: text);
var
    registro_novela: t_registro_novela;
begin
    rewrite(archivo_carga);
    leer_novela(registro_novela,true);
    while (registro_novela.codigo<>codigo_salida) do
    begin
        with registro_novela do
        begin
            writeln(archivo_carga,codigo,' ',precio:0:2,' ',genero);
            writeln(archivo_carga,nombre);
        end;
        leer_novela(registro_novela,true);
    end;
    close(archivo_carga);
end;

procedure cargar_archivo_novelas(var archivo_novelas: t_archivo_novelas; var archivo_carga:
text);
var
    registro_novela: t_registro_novela;
begin
    rewrite(archivo_novelas);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
    begin
        with registro_novela do
        begin
            readln(archivo_carga,codigo,precio,genero); genero:=trim(genero);
            readln(archivo_carga,nombre);
        end;
        write(archivo_novelas,registro_novela);
    end;
    close(archivo_novelas);
    close(archivo_carga);
    textcolor(green); writeln('El archivo binario de novelas fue creado y cargado con éxito');
end;

function control_unicidad(var archivo_novelas: t_archivo_novelas; codigo: int16): boolean;
var
    registro_novela: t_registro_novela;
    ok: boolean;
begin
    ok:=false;
    while ((not eof(archivo_novelas)) and (ok=false)) do
    begin
        read(archivo_novelas,registro_novela);
        if (registro_novela.codigo=codigo) then
            ok:=true;
        end;
        control_unicidad:=ok;
    end;
end;

procedure agregar_novela(var archivo_novelas: t_archivo_novelas);
var
    registro_novela: t_registro_novela;
    novelas: int16;
begin
    novelas:=0;
    reset(archivo_novelas);
    leer_novela(registro_novela,true);
    while (registro_novela.codigo<>codigo_salida) do
    begin
        if (control_unicidad(archivo_novelas,registro_novela.codigo)=false) then
        begin
            seek(archivo_novelas,filesize(archivo_novelas));
            write(archivo_novelas,registro_novela);
            novelas:=novelas+1;
        end;
    end;
end;
```

```

    end;
    leer_novela(registro_novela,true);
end;
close(archivo_novelas);
textcolor(green); write('Se han agregado '); textcolor(yellow); write(novelas);
textcolor(green); writeln(' novelas al final del archivo');
end;
procedure modificar_novela(var archivo_novelas: t_archivo_novelas);
var
    registro_novela: t_registro_novela;
    codigo: int16;
    ok: boolean;
begin
    codigo:=1+random(1000);
    ok:=false;
    reset(archivo_novelas);
    while ((not eof(archivo_novelas)) and (ok=false)) do
    begin
        read(archivo_novelas,registro_novela);
        if (registro_novela.codigo=codigo) then
        begin
            leer_novela(registro_novela,false);
            seek(archivo_novelas,filepos(archivo_novelas)-1);
            write(archivo_novelas,registro_novela);
            ok:=true;
        end;
    end;
    close(archivo_novelas);
    if (ok=true) then
    begin
        textcolor(green); write('Se ha modificado la novela con código '); textcolor(yellow);
        write(codigo); textcolor(green); writeln(' en el archivo');
    end
    else
    begin
        textcolor(green); write('No se ha encontrado la novela con código '); textcolor(yellow);
        write(codigo); textcolor(green); writeln(' en el archivo');
    end;
end;
procedure exportar_archivo_txt(var archivo_novelas: t_archivo_novelas);
var
    registro_novela: t_registro_novela;
    archivo_txt: text;
begin
    reset(archivo_novelas);
    assign(archivo_txt,'novelas2.txt'); rewrite(archivo_txt);
    while (not eof(archivo_novelas)) do
    begin
        read(archivo_novelas,registro_novela);
        with registro_novela do
        begin
            writeln(archivo_txt,codigo,' ',precio:0:2,' ',genero);
            writeln(archivo_txt,nombre);
        end;
    end;
    close(archivo_novelas);
    close(archivo_txt);
    textcolor(green); write('Se ha exportado el archivo creado en el inciso (a) a un archivo de
    texto denominado '); textcolor(yellow); write('"novelas2.txt"'); textcolor(green); writeln('
    con todas las novelas del mismo');
end;
procedure leer_opcion(var opcion: int8);
begin
    textcolor(red); writeln('MENÚ DE OPCIONES');
    textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear un archivo binario
    a partir de la información almacenada en un archivo de texto denominado "novelas.txt"');

```

```
    textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Añadir una o más novelas
al final del archivo con sus datos ingresados por teclado');
    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Modificar una novela
existente');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Exportar el archivo
creado en el inciso (a) a un archivo de texto denominado "novelas2.txt" con todas las novelas
del mismo');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_novelas: t_archivo_novelas; var archivo_carga: text);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
    begin
        case opcion of
            1: cargar_archivo_novelas(archivo_novelas,archivo_carga);
            2: agregar_novela(archivo_novelas);
            3: modificar_novela(archivo_novelas);
            4: exportar_archivo_txt(archivo_novelas);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
end;
var
    archivo_novelas: t_archivo_novelas;
    archivo_carga: text;
begin
    randomize;
    assign(archivo_carga,'novelas1.txt');
    assign(archivo_novelas,'novelas2');
    cargar_archivo_carga(archivo_carga);
    menu_opciones(archivo_novelas,archivo_carga);
end.
```

Trabajo Práctico N° 2:

Archivos Secuenciales Ordenados - Algorítmica Clásica.

Ejercicio 1.

Una empresa posee un archivo con información de los ingresos percibidos por diferentes empleados en concepto de comisión. De cada uno de ellos, se conoce: código de empleado, nombre y monto de la comisión. La información del archivo se encuentra ordenada por código de empleado y cada empleado puede aparecer más de una vez en el archivo de comisiones.

Realizar un procedimiento que reciba el archivo anteriormente descrito y lo compacte. En consecuencia, deberá generar un nuevo archivo en el cual cada empleado aparezca una única vez con el valor total de sus comisiones.

Nota: No se conoce, a priori, la cantidad de empleados. Además, el archivo debe ser recorrido una única vez.

```
program TP2_E1;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=-1;
type
  t_string10=string[10];
  t_registro_empleado=record
    codigo: int16;
    nombre: t_string10;
    comision: real;
  end;
  t_archivo_empleados=file of t_registro_empleado;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_empleados; var archivo_carga:
text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_empleado do
      begin
        readln(archivo_carga,codigo,comision,nombre); nombre:=trim(nombre);
        write(archivo_detalle,registro_empleado);
      end;
    end;
  close(archivo_detalle);
  close(archivo_carga);
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_empleado.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_empleado.nombre);
  textcolor(green); write('; Comisión: '); textcolor(yellow);
  writeln(registro_empleado.comision:0:2);
end;
procedure imprimir_archivo_empleados(var archivo_empleados: t_archivo_empleados);
var
  registro_empleado: t_registro_empleado;
begin
```



```
reset(archivo_empleados);
while (not eof(archivo_empleados)) do
begin
    read(archivo_empleados,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
end;
close(archivo_empleados);
end;
procedure leer_empleado(var archivo_detalle: t_archivo_empleados; var registro_empleado:
t_registro_empleado);
begin
    if (not eof(archivo_detalle)) then
        read(archivo_detalle,registro_empleado)
    else
        registro_empleado.codigo:=codigo_salida;
    end;
end;
procedure cargar_archivo_maestro(var archivo_maestro, archivo_detalle: t_archivo_empleados);
var
    registro_empleado_detalle, registro_empleado_maestro: t_registro_empleado;
    comision_total: real;
begin
    reset(archivo_detalle);
    rewrite(archivo_maestro);
    leer_empleado(archivo_detalle,registro_empleado_detalle);
    while (registro_empleado_detalle.codigo<>codigo_salida) do
    begin
        registro_empleado_maestro:=registro_empleado_detalle;
        comision_total:=0;
        while (registro_empleado_maestro.codigo=registro_empleado_detalle.codigo) do
        begin
            comision_total:=comision_total+registro_empleado_detalle.comision;
            leer_empleado(archivo_detalle,registro_empleado_detalle);
        end;
        registro_empleado_maestro.comision:=comision_total;
        write(archivo_maestro,registro_empleado_maestro);
    end;
    close(archivo_detalle);
    close(archivo_maestro);
end;
var
    archivo_detalle, archivo_maestro: t_archivo_empleados;
    archivo_carga: text;
begin
    assign(archivo_detalle,'empleadosDetalle'); assign(archivo_carga,'empleadosDetalle.txt');
    assign(archivo_maestro,'empleadosMaestro');
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE:'); writeln();
    cargar_archivo_detalle(archivo_detalle,archivo_carga);
    imprimir_archivo_empleados(archivo_detalle);
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
    cargar_archivo_maestro(archivo_maestro,archivo_detalle);
    imprimir_archivo_empleados(archivo_maestro);
end.
```

Ejercicio 2.

El encargado de ventas de un negocio de productos de limpieza desea administrar el stock de los productos que vende. Para ello, genera un archivo maestro donde figuran todos los productos que comercializa. De cada producto, se maneja la siguiente información: código de producto, nombre comercial, precio de venta, stock actual y stock mínimo. Diariamente, se genera un archivo detalle donde se registran todas las ventas de productos realizadas. De cada venta, se registran: código de producto y cantidad de unidades vendidas. Se pide realizar un programa con opciones para:

(a) Actualizar el archivo maestro con el archivo detalle, sabiendo que:

- Ambos archivos están ordenados por código de producto.
- Cada registro del maestro puede ser actualizado por 0, 1 o más registros del archivo detalle.
- El archivo detalle sólo contiene registros que están en el archivo maestro.

(b) Listar en un archivo de texto llamado “stock_minimo.txt” aquellos productos cuyo stock actual esté por debajo del stock mínimo permitido.

```
program TP2_E2;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=-1;
  opcion_salida=0;
type
  t_string10=string[10];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string10;
    precio: real;
    stock_actual: int16;
    stock_minimo: int16;
  end;
  t_registro_venta=record
    codigo: int16;
    cantidad_vendida: int16;
  end;
  t_archivo_maestro=file of t_registro_producto;
  t_archivo_detalle=file of t_registro_venta;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_producto do
      begin
        readln(archivo_carga,codigo,precio,stock_actual,stock_minimo,nombre);
        nombre:=trim(nombre);
        write(archivo_maestro,registro_producto);
      end;
    end;
  close(archivo_maestro);
  close(archivo_carga);
  textcolor(green); writeln('El archivo binario maestro fue creado y cargado con éxito');
```

```
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
    registro_venta: t_registro_venta;
begin
    rewrite(archivo_detalle);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_venta do
            begin
                readln(archivo_carga,codigo,cantidad_vendida);
                write(archivo_detalle,registro_venta);
            end;
        end;
    close(archivo_detalle);
    close(archivo_carga);
    textcolor(green); writeln('El archivo binario detalle fue creado y cargado con éxito');
end;
procedure imprimir_registro_producto(registro_producto: t_registro_producto);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_producto.codigo);
    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_producto.nombre);
    textcolor(green); write('; Precio: '); textcolor(yellow);
write(registro_producto.precio:0:2);
    textcolor(green); write('; Stock actual: '); textcolor(yellow);
write(registro_producto.stock_actual);
    textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
writeln(registro_producto.stock_minimo);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
    registro_producto: t_registro_producto;
begin
    reset(archivo_maestro);
    textcolor(green); writeln('Los productos del archivo maestro son: ');
    while (not eof(archivo_maestro)) do
        begin
            read(archivo_maestro,registro_producto);
            imprimir_registro_producto(registro_producto);
        end;
    close(archivo_maestro);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_venta.codigo);
    textcolor(green); write('; Cantidad vendida: '); textcolor(yellow);
writeln(registro_venta.cantidad_vendida);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
    registro_venta: t_registro_venta;
begin
    reset(archivo_detalle);
    textcolor(green); writeln('Las ventas del archivo detalle son: ');
    while (not eof(archivo_detalle)) do
        begin
            read(archivo_detalle,registro_venta);
            imprimir_registro_venta(registro_venta);
        end;
    close(archivo_detalle);
end;
procedure leer_venta(var archivo_detalle: t_archivo_detalle; var registro_venta:
t_registro_venta);
begin
    if (not eof(archivo_detalle)) then
        read(archivo_detalle,registro_venta)
```

```

else
    registro_venta.codigo:=codigo_salida;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle: t_archivo_detalle);
var
    registro_producto: t_registro_producto;
    registro_venta: t_registro_venta;
begin
    reset(archivo_maestro);
    reset(archivo_detalle);
    leer_venta(archivo_detalle,registro_venta);
    while (registro_venta.codigo<>codigo_salida) do
        begin
            read(archivo_maestro,registro_producto);
            while (registro_producto.codigo<>registro_venta.codigo) do
                read(archivo_maestro,registro_producto);
            while (registro_producto.codigo=registro_venta.codigo) do
                begin
                    if (registro_venta.cantidad_vendida>=registro_producto.stock_actual) then
                        registro_producto.stock_actual:=0
                    else
                        registro_producto.stock_actual:=registro_producto.stock_actual-
registro_venta.cantidad_vendida;
                        leer_venta(archivo_detalle,registro_venta);
                    end;
                    seek(archivo_maestro,filepos(archivo_maestro)-1);
                    write(archivo_maestro,registro_producto);
                end;
            close(archivo_maestro);
            close(archivo_detalle);
            textcolor(green); writeln('El archivo maestro fue actualizado con éxito');
        end;
    procedure exportar_archivo_txt(var archivo_maestro: t_archivo_maestro);
    var
        registro_producto: t_registro_producto;
        archivo_txt: text;
    begin
        reset(archivo_maestro);
        assign(archivo_txt,'stock_minimo_E2.txt'); rewrite(archivo_txt);
        textcolor(green); writeln('Los productos cuyo stock actual está por debajo del stock mínimo
son: ');
        while (not eof(archivo_maestro)) do
            begin
                read(archivo_maestro,registro_producto);
                if (registro_producto.stock_actual<registro_producto.stock_minimo) then
                    begin
                        imprimir_registro_producto(registro_producto);
                        with registro_producto do
                            writeln(archivo_txt,codigo,' ',nombre,' ',precio:0:2,' ',stock_actual,'
',stock_minimo);
                        end;
                    end;
                close(archivo_maestro);
                close(archivo_txt);
                textcolor(green); writeln('El archivo de texto "stock_minimo.txt" fue creado y cargado con
éxito');
            end;
        procedure leer_opcion(var opcion: int8);
        begin
            textcolor(red); writeln('MENÚ DE OPCIONES');
            textcolor(yellow); write('OPCIÓN 1: '); textcolor(green); writeln('Crear archivos de
registros ordenados de productos y cargarlos con datos ingresados desde archivos de texto');
            textcolor(yellow); write('OPCIÓN 2: '); textcolor(green); writeln('Listar en pantalla los
datos de los productos del archivo maestro');

```

```

    textcolor(yellow); write('OPCIÓN 3: '); textcolor(green); writeln('Listar en pantalla los
datos de los productos del archivo detalle');
    textcolor(yellow); write('OPCIÓN 4: '); textcolor(green); writeln('Actualizar el archivo
maestro con el archivo detalle');
    textcolor(yellow); write('OPCIÓN 5: '); textcolor(green); writeln('Listar en un archivo de
texto llamado "stock_minimo.txt" aquellos productos cuyo stock actual esté por debajo del
stock mínimo permitido');
    textcolor(yellow); write('OPCIÓN 0: '); textcolor(green); writeln('Salir del menú de
opciones');
    textcolor(green); write('Introducir opción elegida: '); textcolor(yellow); readln(opcion);
    writeln();
end;
procedure menu_opciones(var archivo_maestro: t_archivo_maestro; var archivo_detalle:
t_archivo_detalle; var archivo_carga_maestro, archivo_carga_detalle: text);
var
    opcion: int8;
begin
    leer_opcion(opcion);
    while (opcion<>opcion_salida) do
    begin
        case opcion of
            1:
                begin
                    cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
                    cargar_archivo_detalle(archivo_detalle,archivo_carga_detalle);
                end;
            2: imprimir_archivo_maestro(archivo_maestro);
            3: imprimir_archivo_detalle(archivo_detalle);
            4: actualizar_archivo_maestro(archivo_maestro,archivo_detalle);
            5: exportar_archivo_txt(archivo_maestro);
            else
                textcolor(green); writeln('La opción ingresada no corresponde a ninguna de las
mostradas en el menú de opciones');
            end;
            writeln();
            leer_opcion(opcion);
        end;
    end;
end;
var
    archivo_maestro: t_archivo_maestro;
    archivo_detalle: t_archivo_detalle;
    archivo_carga_maestro, archivo_carga_detalle: text;
begin
    assign(archivo_maestro,'productosMaestro_E2');
    assign(archivo_carga_maestro,'productosMaestro_E2.txt');
    assign(archivo_detalle,'ventasDetalle_E2');
    assign(archivo_carga_detalle,'ventasDetalle_E2.txt');
    menu_opciones(archivo_maestro,archivo_detalle,archivo_carga_maestro,archivo_carga_detalle);
end.

```

Ejercicio 3.

A partir de información sobre la alfabetización en la Argentina, se necesita actualizar un archivo que contiene los siguientes datos: nombre de provincia, cantidad de personas alfabetizadas y total de encuestados. Se reciben dos archivos detalle provenientes de dos agencias de censo diferentes. Dichos archivos contienen: nombre de la provincia, código de localidad, cantidad de alfabetizados y cantidad de encuestados. Se pide realizar los módulos necesarios para actualizar el archivo maestro a partir de los dos archivos detalle.

Nota: Los archivos están ordenados por nombre de provincia y, en los archivos detalle, pueden venir 0, 1 o más registros por cada provincia.

```

program TP2_E3;
{$codepage UTF8}
uses crt, sysutils;
const
    provincia_salida='ZZZ';
type
    t_string50=string[50];
    t_registro_provincia=record
        provincia: t_string50;
        alfabetizados: int16;
        encuestados: int16;
    end;
    t_registro_localidad=record
        provincia: t_string50;
        codigo: int16;
        alfabetizados: int16;
        encuestados: int16;
    end;
    t_archivo_maestro=file of t_registro_provincia;
    t_archivo_detalle=file of t_registro_localidad;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
    registro_provincia: t_registro_provincia;
begin
    rewrite(archivo_maestro);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_provincia do
            begin
                readln(archivo_carga,alfabetizados,encuestados,provincia); provincia:=trim(provincia);
                write(archivo_maestro,registro_provincia);
            end;
        end;
    close(archivo_maestro);
    close(archivo_carga);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
    registro_localidad: t_registro_localidad;
begin
    rewrite(archivo_detalle);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_localidad do
            begin
                readln(archivo_carga,codigo,alfabetizados,encuestados,provincia);
                provincia:=trim(provincia);
            end;
        end;
    end;
end;

```

```
        write(archivo_detalle, registro_localidad);
    end;
    close(archivo_detalle);
    close(archivo_carga);
end;
procedure imprimir_registro_provincia(registro_provincia: t_registro_provincia);
begin
    textcolor(green); write('Provincia: '); textcolor(yellow);
write(registro_provincia.provincia);
    textcolor(green); write('; Alfabetizados: '); textcolor(yellow);
write(registro_provincia.alfabetizados);
    textcolor(green); write('; Encuestados: '); textcolor(yellow);
writeln(registro_provincia.encuestados);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
    registro_provincia: t_registro_provincia;
begin
    reset(archivo_maestro);
    while (not eof(archivo_maestro)) do
        begin
            read(archivo_maestro, registro_provincia);
            imprimir_registro_provincia(registro_provincia);
        end;
    close(archivo_maestro);
end;
procedure imprimir_registro_localidad(registro_localidad: t_registro_localidad);
begin
    textcolor(green); write('Provincia: '); textcolor(yellow);
write(registro_localidad.provincia);
    textcolor(green); write('; Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
    textcolor(green); write('; Alfabetizados: '); textcolor(yellow);
write(registro_localidad.alfabetizados);
    textcolor(green); write('; Encuestados: '); textcolor(yellow);
writeln(registro_localidad.encuestados);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
    registro_localidad: t_registro_localidad;
begin
    reset(archivo_detalle);
    while (not eof(archivo_detalle)) do
        begin
            read(archivo_detalle, registro_localidad);
            imprimir_registro_localidad(registro_localidad);
        end;
    close(archivo_detalle);
end;
procedure leer_localidad(var archivo_detalle: t_archivo_detalle; var registro_localidad:
t_registro_localidad);
begin
    if (not eof(archivo_detalle)) then
        read(archivo_detalle, registro_localidad)
    else
        registro_localidad.provincia:=provincia_salida;
    end;
procedure minimo(var archivo_detalle1, archivo_detalle2: t_archivo_detalle; var
registro_localidad1, registro_localidad2, min: t_registro_localidad);
begin
    if (registro_localidad1.provincia<=registro_localidad2.provincia) then
        begin
            min:=registro_localidad1;
            leer_localidad(archivo_detalle1, registro_localidad1);
        end
    else
```

```

begin
    min:=registro_localidad2;
    leer_localidad(archivo_detalle2,registro_localidad2);
end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle1, archivo_detalle2: t_archivo_detalle);
var
    registro_provincia: t_registro_provincia;
    registro_localidad1, registro_localidad2, min: t_registro_localidad;
begin
    reset(archivo_maestro);
    reset(archivo_detalle1); reset(archivo_detalle2);
    leer_localidad(archivo_detalle1,registro_localidad1);
    leer_localidad(archivo_detalle2,registro_localidad2);
    minimo(archivo_detalle1,archivo_detalle2,registro_localidad1,registro_localidad2,min);
    while (min.provincia<>provincia_salida) do
    begin
        read(archivo_maestro,registro_provincia);
        while (registro_provincia.provincia<>min.provincia) do
            read(archivo_maestro,registro_provincia);
        while (registro_provincia.provincia=min.provincia) do
        begin
            registro_provincia.alfabetizados:=registro_provincia.alfabetizados+min.alfabetizados;
            registro_provincia.encuestados:=registro_provincia.encuestados+min.encuestados;
            minimo(archivo_detalle1,archivo_detalle2,registro_localidad1,registro_localidad2,min);
        end;
        seek(archivo_maestro,filepos(archivo_maestro)-1);
        write(archivo_maestro,registro_provincia);
    end;
    close(archivo_maestro);
    close(archivo_detalle1); close(archivo_detalle2);
end;
var
    archivo_maestro: t_archivo_maestro;
    archivo_detalle1, archivo_detalle2: t_archivo_detalle;
    archivo_carga_maestro, archivo_carga_detalle1, archivo_carga_detalle2: text;
begin
    assign(archivo_maestro,'provinciaMaestro');
    assign(archivo_carga_maestro,'provinciaMaestro.txt');
    assign(archivo_detalle1,'localidadDetalle1');
    assign(archivo_carga_detalle1,'localidadDetalle1.txt');
    assign(archivo_detalle2,'localidadDetalle2');
    assign(archivo_carga_detalle2,'localidadDetalle2.txt');
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
    cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
    imprimir_archivo_maestro(archivo_maestro);
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 1:'); writeln();
    cargar_archivo_detalle(archivo_detalle1,archivo_carga_detalle1);
    imprimir_archivo_detalle(archivo_detalle1);
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 2:'); writeln();
    cargar_archivo_detalle(archivo_detalle2,archivo_carga_detalle2);
    imprimir_archivo_detalle(archivo_detalle2);
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
    actualizar_archivo_maestro(archivo_maestro,archivo_detalle1,archivo_detalle2);
    imprimir_archivo_maestro(archivo_maestro);
end.

```


Ejercicio 4.

Se cuenta con un archivo de productos de una cadena de venta de alimentos congelados. De cada producto, se almacena: código del producto, nombre, descripción, stock disponible, stock mínimo y precio del producto.

Se recibe, diariamente, un archivo detalle de cada una de las 30 sucursales de la cadena. Se debe realizar el procedimiento que recibe los 30 detalles y actualiza el stock del archivo maestro. La información que se recibe en los detalles es: código de producto y cantidad vendida. Además, se deberá informar en un archivo de texto: nombre de producto, descripción, stock disponible y precio de aquellos productos que tengan stock disponible por debajo del stock mínimo. Pensar alternativas sobre realizar el informe en el mismo procedimiento de actualización o realizarlo en un procedimiento separado (analizar ventajas/desventajas en cada caso).

Nota: Todos los archivos se encuentran ordenados por código de productos. En cada detalle, puede venir 0 o N registros de un determinado producto.

```
program TP2_E4;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=30;
  codigo_salida=999;
type
  t_detalle=1..detalles_total;
  t_string50=string[50];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string50;
    descripcion: t_string50;
    stock_disponible: int16;
    stock_minimo: int16;
    precio: real;
  end;
  t_registro_venta=record
    codigo: int16;
    cantidad_vendida: int16;
  end;
  t_archivo_maestro=file of t_registro_producto;
  t_archivo_detalle=file of t_registro_venta;
  t_vector_ventas=array[t_detalle] of t_registro_venta;
  t_vector_detalle=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalle=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var archivo_carga:
text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga);
  while (not eof(archivo_carga)) do
    with registro_producto do
      begin
        readln(archivo_carga,codigo,stock_disponible,stock_minimo,precio,nombre);
        nombre:=trim(nombre);
        readln(archivo_carga,descripcion); descripcion:=trim(descripcion);
        write(archivo_maestro,registro_producto);
      end;
    end;
  end;
```

```
close(archivo_maestro);
close(archivo_carga);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var archivo_carga:
text);
var
    registro_venta: t_registro_venta;
begin
    rewrite(archivo_detalle);
    reset(archivo_carga);
    while (not eof(archivo_carga)) do
        with registro_venta do
            begin
                readln(archivo_carga,codigo,cantidad_vendida);
                write(archivo_detalle,registro_venta);
            end;
        end;
    close(archivo_detalle);
    close(archivo_carga);
end;
procedure imprimir_registro_producto(registro_producto: t_registro_producto);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_producto.codigo);
    textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_producto.nombre);
    textcolor(green); write('; Descripción: '); textcolor(yellow);
write(registro_producto.descripcion);
    textcolor(green); write('; Stock disponible: '); textcolor(yellow);
write(registro_producto.stock_disponible);
    textcolor(green); write('; Stock mínimo: '); textcolor(yellow);
write(registro_producto.stock_minimo);
    textcolor(green); write('; Precio: '); textcolor(yellow);
writeln(registro_producto.precio:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
    registro_producto: t_registro_producto;
begin
    reset(archivo_maestro);
    while (not eof(archivo_maestro)) do
        begin
            read(archivo_maestro,registro_producto);
            imprimir_registro_producto(registro_producto);
        end;
    close(archivo_maestro);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
    textcolor(green); write('Código: '); textcolor(yellow); write(registro_venta.codigo);
    textcolor(green); write('; Cantidad vendida: '); textcolor(yellow);
writeln(registro_venta.cantidad_vendida);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
    registro_venta: t_registro_venta;
begin
    reset(archivo_detalle);
    while (not eof(archivo_detalle)) do
        begin
            read(archivo_detalle,registro_venta);
            imprimir_registro_venta(registro_venta);
        end;
    close(archivo_detalle);
end;
procedure leer_venta(var archivo_detalle: t_archivo_detalle; var registro_venta:
t_registro_venta);
begin
    if (not eof(archivo_detalle)) then
```

```

    read(archivo_detalle,registro_venta)
else
    registro_venta.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalle: t_vector_detalle; var vector_ventas: t_vector_ventas;
var min: t_registro_venta);
var
    i, pos: t_detalle;
begin
    min.codigo:=codigo_salida;
    for i:= 1 to detalles_total do
        if (vector_ventas[i].codigo<min.codigo) then
            begin
                min:=vector_ventas[i];
                pos:=i;
            end;
        if (min.codigo<codigo_salida) then
            leer_venta(vector_detalle[pos],vector_ventas[pos]);
        end;
    end;
procedure exportar_archivo_txt(var archivo_maestro: t_archivo_maestro);
var
    registro_producto: t_registro_producto;
    archivo_txt: text;
begin
    reset(archivo_maestro);
    assign(archivo_txt,'stock_minimo_E4.txt'); rewrite(archivo_txt);
    while (not eof(archivo_maestro)) do
        begin
            read(archivo_maestro,registro_producto);
            if (registro_producto.stock_disponible<registro_producto.stock_minimo) then
                with registro_producto do
                    writeln(archivo_txt,nombre,' ',descripcion,' ',stock_disponible,' ',precio:0:2);
                end;
            close(archivo_txt);
        end;
    end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalle: t_vector_detalle);
var
    registro_producto: t_registro_producto;
    vector_ventas: t_vector_ventas;
    min: t_registro_venta;
    i: t_detalle;
begin
    reset(archivo_maestro);
    for i:= 1 to detalles_total do
        begin
            reset(vector_detalle[i]);
            leer_venta(vector_detalle[i],vector_ventas[i]);
        end;
    end;
    minimo(vector_detalle,vector_ventas,min);
    while (min.codigo<>codigo_salida) do
        begin
            read(archivo_maestro,registro_producto);
            while (registro_producto.codigo<>min.codigo) do
                read(archivo_maestro,registro_producto);
            while (registro_producto.codigo=min.codigo) do
                begin
                    if (min.cantidad_vendida>=registro_producto.stock_disponible) then
                        registro_producto.stock_disponible:=0
                    else
                        registro_producto.stock_disponible:=registro_producto.stock_disponible-
min.cantidad_vendida;
                    minimo(vector_detalle,vector_ventas,min);
                end;
            seek(archivo_maestro,filepos(archivo_maestro)-1);
            write(archivo_maestro,registro_producto);
        end;
    end;
end;

```

```
end;
exportar_archivo_txt(archivo_maestro);
close(archivo_maestro);
for i:= 1 to detalles_total do
    close(vector_detalle[i]);
end;
var
    archivo_maestro: t_archivo_maestro;
    vector_detalle: t_vector_detalle;
    vector_carga_detalle: t_vector_carga_detalle;
    archivo_carga_maestro: text;
    i: t_detalle;
begin
    assign(archivo_maestro, 'productosMaestro_E4');
    assign(archivo_carga_maestro, 'productosMaestro_E4.txt');
    for i:= 1 to detalles_total do
        begin
            assign(vector_detalle[i], 'ventasDetalle' + intToStr(i) + '_E4');
            assign(vector_carga_detalle[i], 'ventasDetalle' + intToStr(i) + '_E4.txt');
        end;
        writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
        cargar_archivo_maestro(archivo_maestro, archivo_carga_maestro);
        imprimir_archivo_maestro(archivo_maestro);
        for i:= 1 to detalles_total do
            begin
                writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE ', i, ':'); writeln();
                cargar_archivo_detalle(vector_detalle[i], vector_carga_detalle[i]);
                imprimir_archivo_detalle(vector_detalle[i]);
            end;
        writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
        actualizar_archivo_maestro(archivo_maestro, vector_detalle);
        imprimir_archivo_maestro(archivo_maestro);
    end.
```

Ejercicio 5.

Suponer que se trabaja en una oficina donde está montada una LAN (red local). La misma fue construida sobre una topología de red que conecta 5 máquinas entre sí y todas las máquinas se conectan con un servidor central. Semanalmente, cada máquina genera un archivo de logs informando las sesiones abiertas por cada usuario en cada terminal y por cuánto tiempo estuvo abierta. Cada archivo detalle contiene los siguientes campos: `cod_usuario`, `fecha`, `tiempo_sesion`. Se debe realizar un procedimiento que reciba los archivos detalle y genere un archivo maestro con los siguientes datos: `cod_usuario`, `fecha`, `tiempo_total_de_sesiones_abiertas`.

Notas:

- *Cada archivo detalle está ordenado por `cod_usuario` y `fecha`.*
- *Un usuario puede iniciar más de una sesión el mismo día en la misma máquina o, inclusive, en diferentes máquinas.*
- *El archivo maestro debe crearse en la siguiente ubicación física: `/var/log`.*

Ejercicio 6.

Se desea modelar la información necesaria para un sistema de recuentos de casos de COVID para el ministerio de salud de la provincia de Buenos Aires.

Diariamente, se reciben archivos provenientes de los distintos municipios. La información contenida en los mismos es la siguiente: código de localidad, código cepa, cantidad de casos activos, cantidad de casos nuevos, cantidad de casos recuperados, cantidad de casos fallecidos.

El ministerio cuenta con un archivo maestro con la siguiente información: código localidad, nombre localidad, código cepa, nombre cepa, cantidad de casos activos, cantidad de casos nuevos, cantidad de recuperados y cantidad de fallecidos.

Se debe realizar el procedimiento que permita actualizar el maestro con los detalles recibidos, se reciben 10 detalles. Todos los archivos están ordenados por código de localidad y código de cepa.

Para la actualización se debe proceder de la siguiente manera:

- 1. Al número de fallecidos se le suman el valor de fallecidos recibido del detalle.*
- 2. Ídem anterior para los recuperados.*
- 3. Los casos activos se actualizan con el valor recibido en el detalle.*
- 4. Ídem anterior para los casos nuevos hallados.*

Realizar las declaraciones necesarias, el programa principal y los procedimientos que requiera para la actualización solicitada e informar cantidad de localidades con más de 50 casos activos (las localidades pueden o no haber sido actualizadas).

Ejercicio 7.

Ejercicio 8.

Ejercicio 9.

Ejercicio 10.

Ejercicio 11.

Ejercicio 12.

Ejercicio 13.

Ejercicio 14.

Ejercicio 15.

Ejercicio 16.

Ejercicio 17.

Ejercicio 18.

Ejercicio 19.

Trabajo Práctico N° 3

PARTE I: Bajas en Archivos.

Ejercicio 1.

Ejercicio 2.

Ejercicio 3.

Ejercicio 4.

Ejercicio 5.

Ejercicio 6.

Ejercicio 7.

Ejercicio 8.

PARTE II: Actualización Maestro/Detalle, Reportes y *Merge* con Archivos no Ordenados.

Ejercicio 9.

Ejercicio 10.

Ejercicio 11.

Trabajo Práctico N° 4:

.

Ejercicio 1.

Trabajo Práctico N° 5:

.

Ejercicio 1.