Juan Menduiña

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;
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,'E1_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,'E1_enteros');
  procesar_archivo_enteros(archivo_enteros,nums_corte,prom);
  textcolor(green); write('La cantidad de números menores a '); textcolor(yellow);
write(num_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));
  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 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;
  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;
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;
var
  archivo_empleados: t_archivo_empleados;
begin
  randomize;
  assign(archivo_empleados,'E3_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));
```

```
    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;
  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;
```

```
  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,'E4_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,'E4_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;
var
  archivo_empleados: t_archivo_empleados;
begin
  randomize;
  assign(archivo_empleados,'E4_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_string20=string[20];
  t_registro_celular=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    marca: t_string20;
    precio: real;
    stock_minimo: int16;
    stock_disponible: int16;
  end;
  t_archivo_celulares=file of t_registro_celular;
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_celular(var registro_celular: t_registro_celular);
var
  vector_marcas: array[1..10] of t_string20=('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;
  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,'E5_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;
  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;
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,'E5_celulares1.txt');
  assign(archivo_celulares,'E5_celulares2');
  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_string20=string[20];
  t_registro_celular=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    marca: t_string20;
    precio: real;
    stock_minimo: int16;
    stock_disponible: int16;
  end;
  t_archivo_celulares=file of t_registro_celular;
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_celular(var registro_celular: t_registro_celular);
var
  vector_marcas: array[1..10] of t_string20=('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;
  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,'E6_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;
  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;
  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,'E6_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;
  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;
var
  archivo_celulares: t_archivo_celulares;
  archivo_carga: text;
begin
  randomize;
  assign(archivo_carga,'E6_celulares1.txt');
  assign(archivo_celulares,'E6_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;
```

```
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;
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;
    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,'E7_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;
var
  archivo_novelas: t_archivo_novelas;
  archivo_carga: text;
begin
  randomize;
  assign(archivo_carga,'E7_novelas1.txt');
  assign(archivo_novelas,'E7_novelas2');
  cargar_archivo_carga(archivo_carga);
  menu_opciones(archivo_novelas,archivo_carga);
end.
```

Juan Menduiña

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=999;
type
  t_string20=string[20];
  t_registro_empleado=record
    codigo: int16;
    nombre: t_string20;
    comision: real;
  end;
  t_archivo_empleados=file of t_registro_empleado;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_empleados; var
archivo_carga_detalle: text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_empleado do
    begin
      readln(archivo_carga_detalle,codigo,comision,nombre); nombre:=trim(nombre);
      write(archivo_detalle,registro_empleado);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
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;
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_detalle: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE:'); writeln();
  assign(archivo_detalle,'E1_empleadosDetalle');
assign(archivo_carga_detalle,'E1_empleadosDetalle.txt');
  cargar_archivo_detalle(archivo_detalle,archivo_carga_detalle);
  imprimir_archivo_empleados(archivo_detalle);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E1_empleadosMaestro');
  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=999;
  opcion_salida=0;
type
  t_string20=string[20];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string20;
    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_maestro: text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_producto do
    begin
      readln(archivo_carga_maestro,codigo,precio,stock_actual,stock_minimo,nombre);
nombre:=trim(nombre);
      write(archivo_maestro,registro_producto);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
  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_detalle: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_venta do
    begin
      readln(archivo_carga_detalle,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
  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,'E2_stock_minimo.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;
var
  archivo_maestro: t_archivo_maestro;
  archivo_detalle: t_archivo_detalle;
  archivo_carga_maestro, archivo_carga_detalle: text;
begin
  assign(archivo_maestro,'E2_productosMaestro');
assign(archivo_carga_maestro,'E2_productosMaestro.txt');
  assign(archivo_detalle,'E2_ventasDetalle');
assign(archivo_carga_detalle,'E2_ventasDetalle.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
  nombre_salida='ZZZ';
type
  t_string50=string[50];
  t_registro_provincia=record
    nombre: t_string50;
    alfabetizados: int16;
    encuestados: int16;
  end;
  t_registro_localidad=record
    nombre: 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_maestro: text);
var
  registro_provincia: t_registro_provincia;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_provincia do
    begin
      readln(archivo_carga_maestro,alfabetizados,encuestados,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_provincia);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_localidad: t_registro_localidad;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_localidad do
    begin
      readln(archivo_carga_detalle,codigo,alfabetizados,encuestados,nombre);
nombre:=trim(nombre);
```

```
      write(archivo_detalle,registro_localidad);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_provincia(registro_provincia: t_registro_provincia);
begin
  textcolor(green); write('Provincia: '); textcolor(yellow); write(registro_provincia.nombre);
  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.nombre);
  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.nombre:=nombre_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.nombre<=registro_localidad2.nombre) 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.nombre<>nombre_salida) do
  begin
    read(archivo_maestro,registro_provincia);
    while (registro_provincia.nombre<>min.nombre) do
      read(archivo_maestro,registro_provincia);
    while (registro_provincia.nombre=min.nombre) 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
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E3_provinciasMaestro');
assign(archivo_carga_maestro,'E3_provinciasMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 1:'); writeln();
  assign(archivo_detalle1,'E3_localidadesDetalle1');
assign(archivo_carga_detalle1,'E3_localidadesDetalle1.txt');
  cargar_archivo_detalle(archivo_detalle1,archivo_carga_detalle1);
  imprimir_archivo_detalle(archivo_detalle1);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 2:'); writeln();
  assign(archivo_detalle2,'E3_localidadesDetalle2');
assign(archivo_carga_detalle2,'E3_localidadesDetalle2.txt');
  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_string20=string[20];
  t_registro_producto=record
    codigo: int16;
    nombre: t_string20;
    descripcion: t_string20;
    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_detalles=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_producto: t_registro_producto;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_producto do
    begin
      readln(archivo_carga_maestro,codigo,stock_disponible,stock_minimo,precio,nombre);
nombre:=trim(nombre);
      readln(archivo_carga_maestro,descripcion);
      write(archivo_maestro,registro_producto);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_venta do
    begin
      readln(archivo_carga_detalle,codigo,cantidad_vendida);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
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_detalles: t_vector_detalles; 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_detalles[pos],vector_ventas[pos]);
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,'E4_stock_minimo.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;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_producto: t_registro_producto;
  min: t_registro_venta;
  vector_ventas: t_vector_ventas;
  i: t_detalle;
begin
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_venta(vector_detalles[i],vector_ventas[i]);
  end;
  minimo(vector_detalles,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_detalles,vector_ventas,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_producto);
```

```
  end;
  exportar_archivo_txt(archivo_maestro);
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E4_productosMaestro');
assign(archivo_carga_maestro,'E4_productosMaestro.txt');
  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();
    assign(vector_detalles[i],'E4_ventasDetalle'+intToStr(i));
assign(vector_carga_detalles[i],'E4_ventasDetalle'+intToStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalle(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,vector_detalles);
  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.*

```
program TP2_E5;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=5;
  codigo_salida=999;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_sesion=record
    codigo: int16;
    fecha: t_string20;
    tiempo: int16;
  end;
  t_archivo_sesiones=file of t_registro_sesion;
  t_vector_sesiones=array[t_detalle] of t_registro_sesion;
  t_vector_detalles=array[t_detalle] of t_archivo_sesiones;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_sesiones; var
archivo_carga_detalle: text);
var
  registro_sesion: t_registro_sesion;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_sesion do
    begin
      readln(archivo_carga_detalle,codigo,tiempo,fecha); fecha:=trim(fecha);
      write(archivo_detalle,registro_sesion);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_sesion(registro_sesion: t_registro_sesion);
begin
  textcolor(green); write('Código de usuario: '); textcolor(yellow);
write(registro_sesion.codigo);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_sesion.fecha);
  textcolor(green); write('; Tiempo de sesión: '); textcolor(yellow);
writeln(registro_sesion.tiempo);
end;
```

```
procedure imprimir_archivo_sesiones(var archivo_sesiones: t_archivo_sesiones);
var
  registro_sesion: t_registro_sesion;
begin
  reset(archivo_sesiones);
  while (not eof(archivo_sesiones)) do
  begin
    read(archivo_sesiones,registro_sesion);
    imprimir_registro_sesion(registro_sesion);
  end;
  close(archivo_sesiones);
end;
procedure leer_sesion(var archivo_detalle: t_archivo_sesiones; var registro_sesion:
t_registro_sesion);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_sesion)
  else
    registro_sesion.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_sesiones:
t_vector_sesiones; var min: t_registro_sesion);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if ((vector_sesiones[i].codigo<min.codigo) or ((vector_sesiones[i].codigo=min.codigo) and
(vector_sesiones[i].fecha<min.fecha))) then
    begin
      min:=vector_sesiones[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_sesion(vector_detalles[pos],vector_sesiones[pos]);
end;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_sesiones; var vector_detalles:
t_vector_detalles);
var
  registro_sesion, min: t_registro_sesion;
  vector_sesiones: t_vector_sesiones;
  i: t_detalle;
begin
  rewrite(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_sesion(vector_detalles[i],vector_sesiones[i]);
  end;
  minimo(vector_detalles,vector_sesiones,min);
  while (min.codigo<>codigo_salida) do
  begin
    registro_sesion.codigo:=min.codigo;
    while (registro_sesion.codigo=min.codigo) do
    begin
      registro_sesion.fecha:=min.fecha;
      registro_sesion.tiempo:=0;
      while ((registro_sesion.codigo=min.codigo) and (registro_sesion.fecha=min.fecha)) do
      begin
        registro_sesion.tiempo:=registro_sesion.tiempo+min.tiempo;
        minimo(vector_detalles,vector_sesiones,min);
      end;
      write(archivo_maestro,registro_sesion);
    end;
  end;
  close(archivo_maestro);
```

```
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_sesiones;
  i: t_detalle;
begin
  for i:= 1 to detalles_total do
  begin
    writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE ',i,':'); writeln();
    assign(vector_detalles[i],'E5_sesionesDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'E5_sesionesDetalle'+inttoStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_sesiones(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E5_sesionesMaestro');
  cargar_archivo_maestro(archivo_maestro,vector_detalles);
  imprimir_archivo_sesiones(archivo_maestro);
end.
```

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:

- *Al número de fallecidos se le suman el valor de fallecidos recibido del detalle.*
- *Ídem anterior para los recuperados.*
- *Los casos activos se actualizan con el valor recibido en el detalle.*
- *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).

```
program TP2_E6;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=10;
  codigo_salida=999;
  casos_activos_corte=50;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_localidad1=record
    codigo: int16;
    nombre: t_string20;
    codigo_cepa: int16;
    nombre_cepa: t_string20;
    casos_activos: int16;
    casos_nuevos: int16;
    casos_recuperados: int16;
    casos_fallecidos: int16;
  end;
  t_registro_localidad2=record
    codigo: int16;
    codigo_cepa: int16;
    casos_activos: int16;
    casos_nuevos: int16;
    casos_recuperados: int16;
```

```
    casos_fallecidos: int16;
  end;
  t_archivo_maestro=file of t_registro_localidad1;
  t_archivo_detalle=file of t_registro_localidad2;
  t_vector_localidades=array[t_detalle] of t_registro_localidad2;
  t_vector_detalles=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_localidad: t_registro_localidad1;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_localidad do
    begin
      readln(archivo_carga_maestro,codigo,codigo_cepa,casos_activos,casos_nuevos,casos_recuper
ados,casos_fallecidos,nombre_cepa); nombre_cepa:=trim(nombre_cepa);
      readln(archivo_carga_maestro,nombre);
      write(archivo_maestro,registro_localidad);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_localidad: t_registro_localidad2;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_localidad do
    begin
      readln(archivo_carga_detalle,codigo,codigo_cepa,casos_activos,casos_nuevos,casos_recuper
ados,casos_fallecidos);
      write(archivo_detalle,registro_localidad);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_localidad1(registro_localidad: t_registro_localidad1);
begin
  textcolor(green); write('Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
  textcolor(green); write('; Nombre de localidad: '); textcolor(yellow);
write(registro_localidad.nombre);
  textcolor(green); write('; Código de cepa: '); textcolor(yellow);
write(registro_localidad.codigo_cepa);
  textcolor(green); write('; Nombre de cepa: '); textcolor(yellow);
write(registro_localidad.nombre_cepa);
  textcolor(green); write('; Casos activos: '); textcolor(yellow);
write(registro_localidad.casos_activos);
  textcolor(green); write('; Casos nuevos: '); textcolor(yellow);
write(registro_localidad.casos_nuevos);
  textcolor(green); write('; Casos recuperados: '); textcolor(yellow);
write(registro_localidad.casos_recuperados);
  textcolor(green); write('; Casos fallecidos: '); textcolor(yellow);
writeln(registro_localidad.casos_fallecidos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_localidad: t_registro_localidad1;
begin
  reset(archivo_maestro);
```

```
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_localidad);
    imprimir_registro_localidad1(registro_localidad);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_localidad2(registro_localidad: t_registro_localidad2);
begin
  textcolor(green); write('Código de localidad: '); textcolor(yellow);
write(registro_localidad.codigo);
  textcolor(green); write('; Código de cepa: '); textcolor(yellow);
write(registro_localidad.codigo_cepa);
  textcolor(green); write('; Casos activos: '); textcolor(yellow);
write(registro_localidad.casos_activos);
  textcolor(green); write('; Casos nuevos: '); textcolor(yellow);
write(registro_localidad.casos_nuevos);
  textcolor(green); write('; Casos recuperados: '); textcolor(yellow);
write(registro_localidad.casos_recuperados);
  textcolor(green); write('; Casos fallecidos: '); textcolor(yellow);
writeln(registro_localidad.casos_fallecidos);
end;
procedure imprimir_archivo_detalles(var archivo_detalle: t_archivo_detalle);
var
  registro_localidad: t_registro_localidad2;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_localidad);
    imprimir_registro_localidad2(registro_localidad);
  end;
  close(archivo_detalle);
end;
procedure leer_localidad(var archivo_detalle: t_archivo_detalle; var registro_localidad:
t_registro_localidad2);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_localidad)
  else
    registro_localidad.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_localidades:
t_vector_localidades; var min: t_registro_localidad2);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if ((vector_localidades[i].codigo<min.codigo) or
((vector_localidades[i].codigo=min.codigo) and
(vector_localidades[i].codigo_cepa<min.codigo_cepa))) then
    begin
      min:=vector_localidades[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_localidad(vector_detalles[pos],vector_localidades[pos]);
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_localidad: t_registro_localidad1;
  min: t_registro_localidad2;
  vector_localidades: t_vector_localidades;
  i: t_detalle;
```

```
  casos_activos_localidad, localidades_corte: int16;
begin
  localidades_corte:=0;
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_localidad(vector_detalles[i],vector_localidades[i]);
  end;
  minimo(vector_detalles,vector_localidades,min);
  while (min.codigo<>codigo_salida) do
  begin
    casos_activos_localidad:=0;
    read(archivo_maestro,registro_localidad);
    while (registro_localidad.codigo<>min.codigo) do
      read(archivo_maestro,registro_localidad);
    while (registro_localidad.codigo=min.codigo) do
    begin
      while (registro_localidad.codigo_cepa<>min.codigo_cepa) do
        read(archivo_maestro,registro_localidad);
      while ((registro_localidad.codigo=min.codigo) and
(registro_localidad.codigo_cepa=min.codigo_cepa)) do
      begin
        registro_localidad.casos_fallecidos:=registro_localidad.casos_fallecidos+min.casos_fal
lecidos;
        registro_localidad.casos_recuperados:=registro_localidad.casos_recuperados+min.casos_r
ecuperados;
        registro_localidad.casos_activos:=min.casos_activos;
        registro_localidad.casos_nuevos:=min.casos_nuevos;
        casos_activos_localidad:=casos_activos_localidad+min.casos_activos;
        minimo(vector_detalles,vector_localidades,min);
      end;
      seek(archivo_maestro,filepos(archivo_maestro)-1);
      write(archivo_maestro,registro_localidad);
    end;
    if (casos_activos_localidad>casos_activos_corte) then
      localidades_corte:=localidades_corte+1;
  end;
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
  textcolor(green); write('La cantidad de localidades con más de '); textcolor(yellow);
write(casos_activos_corte); textcolor(green); write(' casos activos es: '); textcolor(red);
writeln(localidades_corte);
  writeln();
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E6_localidadesMaestro');
assign(archivo_carga_maestro,'E6_localidadesMaestro.txt');
  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();
    assign(vector_detalles[i],'E6_localidadesDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'E6_localidadesDetalle'+inttoStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalles(vector_detalles[i]);
  end;
```

```
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,vector_detalles);
  imprimir_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 7.

Se dispone de un archivo maestro con información de los alumnos de la Facultad de Informática. Cada registro del archivo maestro contiene: código de alumno, apellido, nombre, cantidad de cursadas aprobadas y cantidad de materias con final aprobado. El archivo maestro está ordenado por código de alumno.

Además, se tienen dos archivos detalle con información sobre el desempeño académico de los alumnos: un archivo de cursadas y un archivo de exámenes finales. El archivo de cursadas contiene información sobre las materias cursadas por los alumnos. Cada registro incluye: código de alumno, código de materia, año de cursada y resultado (sólo interesa si la cursada fue aprobada o desaprobada). Por su parte, el archivo de exámenes finales contiene información sobre los exámenes finales rendidos. Cada registro incluye: código de alumno, código de materia, fecha del examen y nota obtenida. Ambos archivos detalle están ordenados por código de alumno y código de materia, y pueden contener 0, 1 o más registros por alumno en el archivo maestro. Un alumno podría cursar una materia muchas veces, así como también podría rendir el final de una materia en múltiples ocasiones.

Se debe desarrollar un programa que actualice el archivo maestro, ajustando la cantidad de cursadas aprobadas y la cantidad de materias con final aprobado, utilizando la información de los archivos detalle. Las reglas de actualización son las siguientes:

- *Si un alumno aprueba una cursada, se incrementa en uno la cantidad de cursadas aprobadas.*
- *Si un alumno aprueba un examen final (nota>=4), se incrementa en uno la cantidad de materias con final aprobado.*

Notas:

- *Los archivos deben procesarse en un único recorrido.*
- *No es necesario comprobar que no haya inconsistencias en la información de los archivos detalles. Esto es, no puede suceder que un alumno apruebe más de una vez la cursada de una misma materia (a lo sumo, la aprueba una vez), algo similar ocurre con los exámenes finales.*

```
program TP2_E7;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
  resultado_corte='Aprobada'; nota_corte=4.0;
  anio_ini=2000; anio_fin=2025;
type
  t_string20=string[20];
  t_anio=anio_ini..anio_fin;
  t_registro_alumno=record
    codigo: int16;
    apellido: t_string20;
    nombre: t_string20;
    cursadas_aprobadas: int16;
    finales_aprobados: int16;
  end;
```

```
  t_registro_cursada=record
    codigo: int16;
    codigo_materia: int16;
    anio: t_anio;
    resultado: t_string20;
  end;
  t_registro_final=record
    codigo: int16;
    codigo_materia: int16;
    fecha: t_string20;
    nota: real;
  end;
  t_registro_cursada_final=record
    codigo: int16;
    codigo_materia: int16;
    resultado: t_string20;
    nota: real;
  end;
  t_archivo_maestro=file of t_registro_alumno;
  t_archivo_detalle1=file of t_registro_cursada;
  t_archivo_detalle2=file of t_registro_final;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_alumno: t_registro_alumno;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_alumno do
    begin
      readln(archivo_carga_maestro,codigo,cursadas_aprobadas,finales_aprobados,apellido);
apellido:=trim(apellido);
      readln(archivo_carga_maestro,nombre);
      write(archivo_maestro,registro_alumno);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle1(var archivo_detalle1: t_archivo_detalle1; var
archivo_carga_detalle1: text);
var
  registro_cursada: t_registro_cursada;
begin
  rewrite(archivo_detalle1);
  reset(archivo_carga_detalle1);
  while (not eof(archivo_carga_detalle1)) do
    with registro_cursada do
    begin
      readln(archivo_carga_detalle1,codigo,codigo_materia,anio,resultado);
resultado:=trim(resultado);
      write(archivo_detalle1,registro_cursada);
    end;
  close(archivo_detalle1);
  close(archivo_carga_detalle1);
end;
procedure cargar_archivo_detalle2(var archivo_detalle2: t_archivo_detalle2; var
archivo_carga_detalle2: text);
var
  registro_final: t_registro_final;
begin
  rewrite(archivo_detalle2);
  reset(archivo_carga_detalle2);
  while (not eof(archivo_carga_detalle2)) do
    with registro_final do
    begin
```

```
      readln(archivo_carga_detalle2,codigo,codigo_materia,nota,fecha); fecha:=trim(fecha);
      write(archivo_detalle2,registro_final);
    end;
  close(archivo_detalle2);
  close(archivo_carga_detalle2);
end;
procedure imprimir_registro_alumno(registro_alumno: t_registro_alumno);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_alumno.codigo);
  textcolor(green); write('; Apellido: '); textcolor(yellow); write(registro_alumno.apellido);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_alumno.nombre);
  textcolor(green); write('; Cursadas aprobadas: '); textcolor(yellow);
write(registro_alumno.cursadas_aprobadas);
  textcolor(green); write('; Finales aprobados: '); textcolor(yellow);
writeln(registro_alumno.finales_aprobados);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_alumno: t_registro_alumno;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_alumno);
    imprimir_registro_alumno(registro_alumno);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_cursada(registro_cursada: t_registro_cursada);
begin
  textcolor(green); write('Código de alumno: '); textcolor(yellow);
write(registro_cursada.codigo);
  textcolor(green); write('; Código de materia: '); textcolor(yellow);
write(registro_cursada.codigo_materia);
  textcolor(green); write('; Año: '); textcolor(yellow); write(registro_cursada.anio);
  textcolor(green); write('; Resultado: '); textcolor(yellow);
writeln(registro_cursada.resultado);
end;
procedure imprimir_archivo_detalle1(var archivo_detalle1: t_archivo_detalle1);
var
  registro_cursada: t_registro_cursada;
begin
  reset(archivo_detalle1);
  while (not eof(archivo_detalle1)) do
  begin
    read(archivo_detalle1,registro_cursada);
    imprimir_registro_cursada(registro_cursada);
  end;
  close(archivo_detalle1);
end;
procedure imprimir_registro_final(registro_final: t_registro_final);
begin
  textcolor(green); write('Código de alumno: '); textcolor(yellow);
write(registro_final.codigo);
  textcolor(green); write('; Código de materia: '); textcolor(yellow);
write(registro_final.codigo_materia);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_final.fecha);
  textcolor(green); write('; Nota: '); textcolor(yellow); writeln(registro_final.nota:0:2);
end;
procedure imprimir_archivo_detalle2(var archivo_detalle2: t_archivo_detalle2);
var
  registro_final: t_registro_final;
begin
  reset(archivo_detalle2);
  while (not eof(archivo_detalle2)) do
  begin
```

```
    read(archivo_detalle2,registro_final);
    imprimir_registro_final(registro_final);
  end;
  close(archivo_detalle2);
end;
procedure leer_cursada(var archivo_detalle1: t_archivo_detalle1; var registro_cursada:
t_registro_cursada);
begin
  if (not eof(archivo_detalle1)) then
    read(archivo_detalle1,registro_cursada)
  else
    registro_cursada.codigo:=codigo_salida;
end;
procedure leer_final(var archivo_detalle2: t_archivo_detalle2; var registro_final:
t_registro_final);
begin
  if (not eof(archivo_detalle2)) then
    read(archivo_detalle2,registro_final)
  else
    registro_final.codigo:=codigo_salida;
end;
procedure minimo(var archivo_detalle1: t_archivo_detalle1; var archivo_detalle2:
t_archivo_detalle2; var registro_cursada: t_registro_cursada; var registro_final:
t_registro_final; var min: t_registro_cursada_final);
begin
  if (registro_cursada.codigo<=registro_final.codigo) then
  begin
    min.codigo:=registro_cursada.codigo;
    min.codigo_materia:=registro_cursada.codigo_materia;
    min.resultado:=registro_cursada.resultado;
    min.nota:=0;
    leer_cursada(archivo_detalle1,registro_cursada);
  end
  else
  begin
    min.codigo:=registro_final.codigo;
    min.codigo_materia:=registro_final.codigo_materia;
    min.resultado:='';
    min.nota:=registro_final.nota;
    leer_final(archivo_detalle2,registro_final);
  end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle1: t_archivo_detalle1; var archivo_detalle2: t_archivo_detalle2);
var
  registro_alumno: t_registro_alumno;
  registro_cursada: t_registro_cursada;
  registro_final: t_registro_final;
  min: t_registro_cursada_final;
begin
  reset(archivo_maestro);
  reset(archivo_detalle1); reset(archivo_detalle2);
  leer_cursada(archivo_detalle1,registro_cursada);
  leer_final(archivo_detalle2,registro_final);
  minimo(archivo_detalle1,archivo_detalle2,registro_cursada,registro_final,min);
  while (min.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_alumno);
    while (registro_alumno.codigo<>min.codigo) do
      read(archivo_maestro,registro_alumno);
    while (registro_alumno.codigo=min.codigo) do
    begin
      if (min.resultado=resultado_corte) then
        registro_alumno.cursadas_aprobadas:=registro_alumno.cursadas_aprobadas+1;
      if (min.nota>=nota_corte) then
        registro_alumno.finales_aprobados:=registro_alumno.finales_aprobados+1;
```

```
      minimo(archivo_detalle1,archivo_detalle2,registro_cursada,registro_final,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_alumno);
  end;
  close(archivo_maestro);
  close(archivo_detalle1); close(archivo_detalle2);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_detalle1: t_archivo_detalle1;
  archivo_detalle2: t_archivo_detalle2;
  archivo_carga_maestro, archivo_carga_detalle1, archivo_carga_detalle2: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E7_alumnosMaestro');
assign(archivo_carga_maestro,'E7_alumnosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 1:'); writeln();
  assign(archivo_detalle1,'E7_cursadasDetalle');
assign(archivo_carga_detalle1,'E7_cursadasDetalle.txt');
  cargar_archivo_detalle1(archivo_detalle1,archivo_carga_detalle1);
  imprimir_archivo_detalle1(archivo_detalle1);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE 2:'); writeln();
  assign(archivo_detalle2,'E7_finalesDetalle');
assign(archivo_carga_detalle2,'E7_finalesDetalle.txt');
  cargar_archivo_detalle2(archivo_detalle2,archivo_carga_detalle2);
  imprimir_archivo_detalle2(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 8.

Se quiere optimizar la gestión del consumo de yerba mate en distintas provincias de Argentina. Para ello, se cuenta con un archivo maestro que contiene la siguiente información: código de provincia, nombre de la provincia, cantidad de habitantes y cantidad total de kilos de yerba consumidos históricamente.

Cada mes, se reciben 16 archivos de relevamiento con información sobre el consumo de yerba en los distintos puntos del país. Cada archivo contiene: código de provincia y cantidad de kilos de yerba consumidos en ese relevamiento. Un archivo de relevamiento puede contener información de una o varias provincias, y una misma provincia puede aparecer cero, una o más veces en distintos archivos de relevamiento.

Tanto el archivo maestro como los archivos de relevamiento están ordenados por código de provincia.

Se desea realizar un programa que actualice el archivo maestro en base a la nueva información de consumo de yerba. Además, se debe informar en pantalla aquellas provincias (código y nombre) donde la cantidad total de yerba consumida supere los 10.000 kilos históricamente, junto con el promedio consumido de yerba por habitante. Es importante tener en cuenta tanto las provincias actualizadas como las que no fueron actualizadas.

Nota: Cada archivo debe recorrerse una única vez.

```
program TP2_E8;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=3; // detalles_total=16;
  codigo_salida=999;
  kilos_corte=10000;
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_provincia1=record
    codigo: int16;
    nombre: t_string20;
    habitantes: int16;
    kilos: int32;
  end;
  t_registro_provincia2=record
    codigo: int16;
    kilos: int32;
  end;
  t_archivo_maestro=file of t_registro_provincia1;
  t_archivo_detalle=file of t_registro_provincia2;
  t_vector_provincias=array[t_detalle] of t_registro_provincia2;
  t_vector_detalles=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_provincia: t_registro_provincia1;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
```

```
  while (not eof(archivo_carga_maestro)) do
    with registro_provincia do
    begin
      readln(archivo_carga_maestro,codigo,habitantes,kilos,nombre); nombre:=trim(nombre);
      write(archivo_maestro,registro_provincia);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_provincia: t_registro_provincia2;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_provincia do
    begin
      readln(archivo_carga_detalle,codigo,kilos);
      write(archivo_detalle,registro_provincia);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_provincia1(registro_provincia: t_registro_provincia1);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_provincia.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_provincia.nombre);
  textcolor(green); write('; Habitantes: '); textcolor(yellow);
write(registro_provincia.habitantes);
  textcolor(green); write('; Kilos de yerba consumidos: '); textcolor(yellow);
writeln(registro_provincia.kilos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_provincia: t_registro_provincia1;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_provincia);
    imprimir_registro_provincia1(registro_provincia);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_provincia2(registro_provincia: t_registro_provincia2);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_provincia.codigo);
  textcolor(green); write('; Kilos de yerba consumidos: '); textcolor(yellow);
writeln(registro_provincia.kilos);
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_provincia: t_registro_provincia2;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_provincia);
    imprimir_registro_provincia2(registro_provincia);
  end;
  close(archivo_detalle);
end;
procedure leer_provincia(var archivo_detalle: t_archivo_detalle; var registro_provincia:
t_registro_provincia2);
```

```
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_provincia)
  else
    registro_provincia.codigo:=codigo_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_provincias:
t_vector_provincias; var min: t_registro_provincia2);
var
  i, pos: t_detalle;
begin
  min.codigo:=codigo_salida;
  for i:= 1 to detalles_total do
    if (vector_provincias[i].codigo<min.codigo) then
    begin
      min:=vector_provincias[i];
      pos:=i;
    end;
  if (min.codigo<codigo_salida) then
    leer_provincia(vector_detalles[pos],vector_provincias[pos]);
end;
procedure imprimir_texto(registro_provincia: t_registro_provincia1);
begin
  if (registro_provincia.kilos>kilos_corte) then
  begin
    textcolor(green); write('El nombre y el código de esta provincia donde la cantidad total
de yerba consumida supera los '); textcolor(yellow); write(kilos_corte); textcolor(green);
write(' kilos son '); textcolor(red); write(registro_provincia.nombre); textcolor(green);
write(' y '); textcolor(red); write(registro_provincia.codigo); textcolor(green); write(',
respectivamente, mientras que el promedio consumido de yerba por habitante es ');
textcolor(red); writeln(registro_provincia.kilos/registro_provincia.habitantes:0:2);
  end;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles);
var
  registro_provincia: t_registro_provincia1;
  min: t_registro_provincia2;
  vector_provincias: t_vector_provincias;
  i: t_detalle;
begin
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_provincia(vector_detalles[i],vector_provincias[i]);
  end;
  minimo(vector_detalles,vector_provincias,min);
  while (min.codigo<>codigo_salida) do
  begin
    read(archivo_maestro,registro_provincia);
    while (registro_provincia.codigo<>min.codigo) do
    begin
      imprimir_texto(registro_provincia);
      read(archivo_maestro,registro_provincia);
    end;
    while (registro_provincia.codigo=min.codigo) do
    begin
      registro_provincia.kilos:=registro_provincia.kilos+min.kilos;
      minimo(vector_detalles,vector_provincias,min);
    end;
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_provincia);
    imprimir_texto(registro_provincia);
  end;
  while (not eof(archivo_maestro)) do
```

```
  begin
    read(archivo_maestro,registro_provincia);
    imprimir_texto(registro_provincia);
  end;
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
  writeln();
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E8_provinciasMaestro');
assign(archivo_carga_maestro,'E8_provinciasMaestro.txt');
  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();
    assign(vector_detalles[i],'E8_provinciasDetalle'+intToStr(i));
assign(vector_carga_detalles[i],'E8_provinciasDetalle'+intToStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalle(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,vector_detalles);
  imprimir_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 9.

Se cuenta con un archivo que posee información de las ventas que realiza una empresa a los diferentes clientes. Se necesita obtener un reporte con las ventas organizadas por cliente. Para ello, se deberá informar por pantalla: los datos personales del cliente, el total mensual (mes por mes cuánto compró) y, finalmente, el monto total comprado en el año por el cliente. Además, al finalizar el reporte, se debe informar el monto total de ventas obtenido por la empresa.

El formato del archivo maestro está dado por: cliente (código cliente, nombre y apellido), año, mes, día y monto de la venta. El orden del archivo está dado por: código cliente, año y mes.

Nota: Tener en cuenta que puede haber meses en los que los clientes no realizaron compras. No es necesario informar tales meses en el reporte.

```
program TP2_E9;
{$codepage UTF8}
uses crt, sysutils;
const
  codigo_salida=999;
  dia_ini=1; dia_fin=31;
  mes_ini=1; mes_fin=12;
  anio_ini=2000; anio_fin=2025;
type
  t_string20=string[20];
  t_dia=dia_ini..dia_fin;
  t_mes=mes_ini..mes_fin;
  t_anio=anio_ini..anio_fin;
  t_registro_cliente=record
    codigo: int16;
    nombre: t_string20;
    apellido: t_string20;
  end;
  t_registro_fecha=record
    anio: t_anio;
    mes: t_mes;
    dia: t_dia;
  end;
  t_registro_venta=record
    cliente: t_registro_cliente;
    fecha: t_registro_fecha;
    monto: real;
  end;
  t_archivo_maestro=file of t_registro_venta;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_venta do
    begin
      readln(archivo_carga_maestro,cliente.codigo,fecha.anio,fecha.mes,fecha.dia,monto,cliente
.nombre); cliente.nombre:=trim(cliente.nombre);
      readln(archivo_carga_maestro,cliente.apellido);
      write(archivo_maestro,registro_venta);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_cliente(registro_cliente: t_registro_cliente);
begin
  textcolor(green); write('Código: '); textcolor(yellow); write(registro_cliente.codigo);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_cliente.nombre);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_cliente.apellido);
end;
procedure imprimir_registro_fecha(registro_fecha: t_registro_fecha);
begin
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_fecha.anio);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.mes);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.dia);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
  imprimir_registro_cliente(registro_venta.cliente);
  imprimir_registro_fecha(registro_venta.fecha);
  textcolor(green); write('; Monto: $'); textcolor(yellow); writeln(registro_venta.monto:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_venta: t_registro_venta;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_venta);
    imprimir_registro_venta(registro_venta);
  end;
  close(archivo_maestro);
end;
procedure leer_venta(var archivo_maestro: t_archivo_maestro; var registro_venta:
t_registro_venta);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_venta)
  else
    registro_venta.cliente.codigo:=codigo_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_venta: t_registro_venta;
  monto_total, monto_anio, monto_mes: real;
  codigo, anio, mes: int16;
begin
  monto_total:=0;
  reset(archivo_maestro);
  leer_venta(archivo_maestro,registro_venta);
  while (registro_venta.cliente.codigo<>codigo_salida) do
  begin
    codigo:=registro_venta.cliente.codigo;
    textcolor(green); write('CLIENTE: '); imprimir_registro_cliente(registro_venta.cliente);
writeln(); writeln();
    while (registro_venta.cliente.codigo=codigo) do
    begin
      anio:=registro_venta.fecha.anio;
      monto_anio:=0;
      textcolor(green); write('AÑO '); textcolor(yellow); writeln(anio);
      while ((registro_venta.cliente.codigo=codigo) and (registro_venta.fecha.anio=anio)) do
      begin
        mes:=registro_venta.fecha.mes;
        monto_mes:=0;
```

```
        while ((registro_venta.cliente.codigo=codigo) and (registro_venta.fecha.anio=anio) and
(registro_venta.fecha.mes=mes)) do
        begin
          monto_mes:=monto_mes+registro_venta.monto;
          leer_venta(archivo_maestro,registro_venta);
        end;
        textcolor(green); write('Monto total del mes '); textcolor(yellow); write(mes);
textcolor(green); write(': $'); textcolor(red); writeln(monto_mes:0:2);
        monto_anio:=monto_anio+monto_mes;
      end;
      textcolor(green); write('Monto total del año '); textcolor(yellow); write(anio);
textcolor(green); write(': $'); textcolor(red); writeln(monto_anio:0:2); writeln();
      monto_total:=monto_total+monto_anio;
    end;
  end;
  textcolor(green); write('Monto total de ventas obtenido por la empresa: $'); textcolor(red);
writeln(monto_total:0:2);
  close(archivo_maestro);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E9_ventasMaestro');
assign(archivo_carga_maestro,'E9_ventasMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 10.

Se necesita contabilizar los votos de las diferentes mesas electorales registradas por provincia y localidad. Para ello, se posee un archivo con la siguiente información: código de provincia, código de localidad, número de mesa y cantidad de votos en dicha mesa. Presentar en pantalla un listado como se muestra a continuación:

Código de Provincia

Código de Localidad Total de Votos

....

....

Total de Votos Provincia: ____

Código de Provincia

Código de Localidad Total de Votos

....

Total de Votos Provincia: ___

.....

Total General de Votos: ___

Nota: La información está ordenada por código de provincia y código de localidad.

```
program TP2_E10;
{$codepage UTF8}
uses crt;
const
  provincia_salida=999;
type
  t_registro_mesa=record
    provincia: int16;
    localidad: int16;
    mesa: int16;
    votos: int32;
  end;
  t_archivo_maestro=file of t_registro_mesa;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_mesa: t_registro_mesa;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_mesa do
    begin
      readln(archivo_carga_maestro,provincia,localidad,mesa,votos);
      write(archivo_maestro,registro_mesa);
    end;
```

```
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_mesa(registro_mesa: t_registro_mesa);
begin
  textcolor(green); write('Código de provincia: '); textcolor(yellow);
write(registro_mesa.provincia);
  textcolor(green); write('; Código de localidad: '); textcolor(yellow);
write(registro_mesa.localidad);
  textcolor(green); write('; Número de mesa: '); textcolor(yellow); write(registro_mesa.mesa);
  textcolor(green); write('; Votos: '); textcolor(yellow); writeln(registro_mesa.votos);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_mesa: t_registro_mesa;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_mesa);
    imprimir_registro_mesa(registro_mesa);
  end;
  close(archivo_maestro);
end;
procedure leer_votos(var archivo_maestro: t_archivo_maestro; var registro_mesa:
t_registro_mesa);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_mesa)
  else
    registro_mesa.provincia:=provincia_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_mesa: t_registro_mesa;
  provincia, localidad: int16;
  votos_total, votos_provincia, votos_localidad: int32;
begin
  votos_total:=0;
  reset(archivo_maestro);
  leer_votos(archivo_maestro,registro_mesa);
  while (registro_mesa.provincia<>provincia_salida) do
  begin
    provincia:=registro_mesa.provincia;
    votos_provincia:=0;
    textcolor(green); write('Código de Provincia: '); textcolor(yellow); writeln(provincia);
    textcolor(green); writeln('Código de Localidad          Total de Votos');
    while (registro_mesa.provincia=provincia) do
    begin
      localidad:=registro_mesa.localidad;
      votos_localidad:=0;
      while ((registro_mesa.provincia=provincia) and (registro_mesa.localidad=localidad)) do
      begin
        votos_localidad:=votos_localidad+registro_mesa.votos;
        leer_votos(archivo_maestro,registro_mesa);
      end;
      textcolor(yellow); write(localidad); textcolor(green);
write('                            '); textcolor(red); writeln(votos_localidad);
      votos_provincia:=votos_provincia+votos_localidad;
    end;
    textcolor(green); write('Total de Votos Provincia: '); textcolor(red);
writeln(votos_provincia); writeln();
    votos_total:=votos_total+votos_provincia;
  end;
  textcolor(green); write('Total General de Votos: '); textcolor(red); writeln(votos_total);
  close(archivo_maestro);
```

```
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E10_mesasMaestro');
assign(archivo_carga_maestro,'E10_mesasMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 11.

Se tiene información en un archivo de las horas extras realizadas por los empleados de una empresa en un mes. Para cada empleado, se tiene la siguiente información: departamento, división, número de empleado, categoría y cantidad de horas extras realizadas por el empleado. Se sabe que el archivo se encuentra ordenado por departamento, luego por división y, por último, por número de empleado. Presentar en pantalla un listado con el siguiente formato:

Departamento

División

Número de Empleado Total de Hs. Importe a cobrar

....

....

Total de horas división: ____

Monto total por división: ____

División

.....

Total horas departamento: ____

Monto total departamento: ____

Para obtener el valor de la hora, se debe cargar un arreglo desde un archivo de texto al iniciar el programa con el valor de la hora extra para cada categoría. La categoría varía de 1 a 15. En el archivo de texto, debe haber una línea para cada categoría con el número de categoría y el valor de la hora, pero el arreglo debe ser de valores de horas, con la posición del valor coincidente con el número de categoría.

```
program TP2_E11;
{$codepage UTF8}
uses crt;
const
  departamento_salida=999;
  categoria_ini=1; categoria_fin=15;
type
  t_categoria=categoria_ini..categoria_fin;
  t_registro_empleado=record
    departamento: int16;
    division: int16;
    empleado: int16;
    categoria: t_categoria;
    horas: int16;
  end;
  t_vector_horas=array[t_categoria] of real;
  t_archivo_maestro=file of t_registro_empleado;
```

```
procedure cargar_vector_horas(var vector_horas: t_vector_horas; var archivo_carga_vector:
text);
var
  categoria: t_categoria;
  valor_hora: real;
begin
  reset(archivo_carga_vector);
  while (not eof(archivo_carga_vector)) do
  begin
    readln(archivo_carga_vector,categoria,valor_hora);
    vector_horas[categoria]:=valor_hora;
  end;
  close(archivo_carga_vector);
end;
procedure imprimir_vector_horas(vector_horas: t_vector_horas);
var
  i: t_categoria;
begin
  for i:= categoria_ini to categoria_fin do
  begin
    textcolor(green); write('El valor de la hora de la categoría ',i,' es $');
textcolor(yellow); writeln(vector_horas[i]:0:2);
  end;
end;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_empleado: t_registro_empleado;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
  begin
    with registro_empleado do
    begin
      readln(archivo_carga_maestro,departamento,division,empleado,categoria,horas);
      write(archivo_maestro,registro_empleado);
    end;
  end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_empleado(registro_empleado: t_registro_empleado);
begin
  textcolor(green); write('Departamento: '); textcolor(yellow);
write(registro_empleado.departamento);
  textcolor(green); write('; División: '); textcolor(yellow);
write(registro_empleado.division);
  textcolor(green); write('; Número de empleado: '); textcolor(yellow);
write(registro_empleado.empleado);
  textcolor(green); write('; Categoría: '); textcolor(yellow);
write(registro_empleado.categoria);
  textcolor(green); write('; Horas extras: '); textcolor(yellow);
writeln(registro_empleado.horas);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_empleado: t_registro_empleado;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_empleado);
    imprimir_registro_empleado(registro_empleado);
  end;
  close(archivo_maestro);
```

```
end;
procedure leer_empleado(var archivo_maestro: t_archivo_maestro; var registro_empleado:
t_registro_empleado);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_empleado)
  else
    registro_empleado.departamento:=departamento_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro; vector_horas:
t_vector_horas);
var
  registro_empleado: t_registro_empleado;
  categoria: t_categoria;
  departamento, division, empleado, horas_departamento, horas_division, horas_empleado: int16;
  monto_departamento, monto_division, monto_empleado: real;
begin
  reset(archivo_maestro);
  leer_empleado(archivo_maestro,registro_empleado);
  while (registro_empleado.departamento<>departamento_salida) do
  begin
    departamento:=registro_empleado.departamento;
    horas_departamento:=0; monto_departamento:=0;
    textcolor(green); write('Departamento: '); textcolor(yellow); writeln(departamento);
    while (registro_empleado.departamento=departamento) do
    begin
      division:=registro_empleado.division;
      horas_division:=0; monto_division:=0;
      textcolor(green); write('División: '); textcolor(yellow); writeln(division);
      textcolor(green); writeln('Número de Empleado          Total de Hs.          Importe a
cobrar');
      while ((registro_empleado.departamento=departamento) and
(registro_empleado.division=division)) do
      begin
        empleado:=registro_empleado.empleado; categoria:=registro_empleado.categoria;
        horas_empleado:=0; monto_empleado:=0;
        while ((registro_empleado.departamento=departamento) and
(registro_empleado.division=division) and (registro_empleado.empleado=empleado)) do
        begin
          horas_empleado:=horas_empleado+registro_empleado.horas;
          leer_empleado(archivo_maestro,registro_empleado);
        end;
        monto_empleado:=horas_empleado*vector_horas[categoria];
        textcolor(yellow); write(empleado); textcolor(green);
write('                          '); textcolor(red); write(horas_empleado); textcolor(green);
write('                    $'); textcolor(red); writeln(monto_empleado:0:2);
        horas_division:=horas_division+horas_empleado;
        monto_division:=monto_division+monto_empleado;
      end;
      textcolor(green); write('Total horas división: '); textcolor(red);
writeln(horas_division);
      textcolor(green); write('Monto total división: $'); textcolor(red);
writeln(monto_division:0:2);
      horas_departamento:=horas_departamento+horas_division;
      monto_departamento:=monto_departamento+monto_division;
    end;
    textcolor(green); write('Total horas departamento: '); textcolor(red);
writeln(horas_departamento);
    textcolor(green); write('Monto total departamento: $'); textcolor(red);
writeln(monto_departamento:0:2); writeln();
  end;
end;
var
  vector_horas: t_vector_horas;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_vector, archivo_carga_maestro: text;
```

```
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN VECTOR HORAS:'); writeln();
  assign(archivo_carga_vector,'E11_horasVector.txt');
  cargar_vector_horas(vector_horas,archivo_carga_vector);
  imprimir_vector_horas(vector_horas);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E11_empleadosMaestro');
assign(archivo_carga_maestro,'E11_empleadosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  procesar_archivo_maestro(archivo_maestro,vector_horas);
end.
```

Ejercicio 12.

La empresa de software "X" posee un servidor web donde se encuentra alojado el sitio web de la organización. En dicho servidor, se almacenan, en un archivo, todos los accesos que se realizan al sitio. La información que se almacena en el archivo es la siguiente: año, mes, día, idUsuario y tiempo de acceso al sitio de la organización. El archivo se encuentra ordenado por los siguientes criterios: año, mes, día e idUsuario.

Se debe realizar un procedimiento que genere un informe en pantalla. Para ello, se indicará el año calendario sobre el cual debe realizar el informe. El mismo debe respetar el formato mostrado a continuación:

```
Año : ---
    Mes:-- 1
        día:-- 1
              idUsuario 1   Tiempo Total de acceso en el dia 1 mes 1
              --------
              idUsuario N    Tiempo total de acceso en el dia 1 mes 1
           Tiempo total acceso dia 1 mes 1
     -------------
     día N
              idUsuario 1   Tiempo Total de acceso en el dia N mes 1
              --------
              idUsuario N    Tiempo total de acceso en el dia N mes 1
           Tiempo total acceso dia N mes 1
     Total tiempo de acceso mes 1
     ------
     Mes 12
     día 1
              idUsuario 1   Tiempo Total de acceso en el dia 1 mes 12
              --------
              idUsuario N    Tiempo total de acceso en el dia 1 mes 12
           Tiempo total acceso dia 1 mes 12
     -------------

    día N
             idUsuario 1   Tiempo Total de acceso en el dia N mes 12
                --------
             idUsuario N    Tiempo total de acceso en el dia N mes 12
          Tiempo total acceso dia N mes 12
    Total tiempo de acceso mes 12
Total tiempo de acceso año
```

Se deberá tener en cuenta las siguientes aclaraciones:

- *El año sobre el cual realizará el informe de accesos debe leerse desde el teclado.*
- *El año puede no existir en el archivo, en tal caso debe informarse en pantalla "Año no encontrado".*

- *Se debe definir las estructuras de datos necesarias.*
- *El recorrido del archivo debe realizarse una única vez, procesando sólo la información necesaria.*

```
program TP2_E12;
{$codepage UTF8}
uses crt;
const
  usuario_salida=999;
  dia_ini=1; dia_fin=31;
  mes_ini=1; mes_fin=12;
  anio_ini=2020; anio_fin=2025;
type
  t_dia=dia_ini..dia_fin;
  t_mes=mes_ini..mes_fin;
  t_anio=anio_ini..anio_fin;
  t_registro_fecha=record
    anio: t_anio;
    mes: t_mes;
    dia: t_dia;
  end;
  t_registro_usuario=record
    fecha: t_registro_fecha;
    usuario: int16;
    tiempo: real;
  end;
  t_archivo_maestro=file of t_registro_usuario;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_usuario: t_registro_usuario;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_usuario do
    begin
      readln(archivo_carga_maestro,fecha.anio,fecha.mes,fecha.dia,usuario,tiempo);
      write(archivo_maestro,registro_usuario);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure imprimir_registro_fecha(registro_fecha: t_registro_fecha);
begin
  textcolor(green); write('Fecha: '); textcolor(yellow); write(registro_fecha.anio);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.mes);
  textcolor(green); write('/'); textcolor(yellow); write(registro_fecha.dia);
end;
procedure imprimir_registro_usuario(registro_usuario: t_registro_usuario);
begin
  imprimir_registro_fecha(registro_usuario.fecha);
  textcolor(green); write('; ID usuario: '); textcolor(yellow);
write(registro_usuario.usuario);
  textcolor(green); write('; Tiempo de acceso: '); textcolor(yellow);
writeln(registro_usuario.tiempo:0:2);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_usuario: t_registro_usuario;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
```

```
  begin
    read(archivo_maestro,registro_usuario);
    imprimir_registro_usuario(registro_usuario);
  end;
  close(archivo_maestro);
end;
procedure leer_acceso(var archivo_maestro: t_archivo_maestro; var registro_usuario:
t_registro_usuario);
begin
  if (not eof(archivo_maestro)) then
    read(archivo_maestro,registro_usuario)
  else
    registro_usuario.usuario:=usuario_salida;
end;
procedure procesar_archivo_maestro(var archivo_maestro: t_archivo_maestro; anio: t_anio);
var
  registro_usuario: t_registro_usuario;
  mes, dia, usuario: int16;
  tiempo_anio, tiempo_mes, tiempo_dia, tiempo_usuario: real;
begin
  reset(archivo_maestro);
  leer_acceso(archivo_maestro,registro_usuario);
  while ((registro_usuario.usuario<>usuario_salida) and (registro_usuario.fecha.anio<>anio))
do
    leer_acceso(archivo_maestro,registro_usuario);
  if (registro_usuario.usuario<>usuario_salida) then
  begin
    tiempo_anio:=0;
    textcolor(green); write('Año: '); textcolor(yellow); writeln(anio); writeln();
    while (registro_usuario.fecha.anio=anio) do
    begin
      mes:=registro_usuario.fecha.mes;
      tiempo_mes:=0;
      textcolor(green); write('  Mes: '); textcolor(yellow); writeln(mes);
      while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes)) do
      begin
        dia:=registro_usuario.fecha.dia;
        tiempo_dia:=0;
        textcolor(green); write('    Día: '); textcolor(yellow); writeln(dia);
        textcolor(green); write('      idUsuario          Tiempo Total de acceso en el día ');
textcolor(yellow); write(dia); textcolor(green); write(' mes '); textcolor(yellow);
writeln(mes);
        while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes) and
(registro_usuario.fecha.dia=dia)) do
        begin
          usuario:=registro_usuario.usuario;
          tiempo_usuario:=0;
          while ((registro_usuario.fecha.anio=anio) and (registro_usuario.fecha.mes=mes) and
(registro_usuario.fecha.dia=dia) and (registro_usuario.usuario=usuario)) do
          begin
            tiempo_usuario:=tiempo_usuario+registro_usuario.tiempo;
            leer_acceso(archivo_maestro,registro_usuario);
          end;
          textcolor(green); write('      '); textcolor(yellow); write(usuario);
textcolor(green); write('                  '); textcolor(red); writeln(tiempo_usuario:0:2);
          tiempo_dia:=tiempo_dia+tiempo_usuario;
        end;
        textcolor(green); write('    Tiempo total acceso día '); textcolor(yellow);
write(dia); textcolor(green); write(' mes '); textcolor(yellow); write(mes); textcolor(green);
write(': '); textcolor(red); writeln(tiempo_dia:0:2);
        tiempo_mes:=tiempo_mes+tiempo_dia;
      end;
      textcolor(green); write('  Tiempo total acceso mes '); textcolor(yellow); write(mes);
textcolor(green); write(': '); textcolor(red); writeln(tiempo_mes:0:2); writeln();
      tiempo_anio:=tiempo_anio+tiempo_mes;
    end;
```

```
    textcolor(green); write('Tiempo total acceso año '); textcolor(yellow); write(anio);
textcolor(green); write(': '); textcolor(red); writeln(tiempo_anio:0:2);
  end
  else
  begin
    textcolor(green); write('Año '); textcolor(yellow); write(anio); textcolor(green);
writeln(' no encontrado');
  end;
  close(archivo_maestro);
end;
var
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  anio: t_anio;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E12_usuariosMaestro');
assign(archivo_carga_maestro,'E12_usuariosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('PROCESAMIENTO ARCHIVO MAESTRO:'); writeln();
  anio:=anio_ini+random(anio_fin-anio_ini+1);
  procesar_archivo_maestro(archivo_maestro,anio);
end.
```

Ejercicio 13.

Suponer que se es administrador de un servidor de correo electrónico. En los logs del mismo (información guardada acerca de los movimientos que ocurren en el server) que se encuentran en la ruta /var/log/logmail.dat, se guarda la siguiente información: nro_usuario, nombreUsuario, nombre, apellido, cantidadMailEnviados. Diariamente, el servidor de correo genera un archivo con la siguiente información: nro_usuario, cuentaDestino, cuerpoMensaje. Este archivo representa todos los correos enviados por los usuarios en un día determinado. Ambos archivos están ordenados por nro_usuario y se sabe que un usuario puede enviar cero, uno o más mails por día.

(a) *Realizar el procedimiento necesario para actualizar la información del log en un día particular. Definir las estructuras de datos que utilice el procedimiento.*

(b) *Generar un archivo de texto que contenga el siguiente informe dado un archivo detalle de un día determinado:*

nro_usuarioX.....cantidadMensajesEnviados

.....

nro_usuarioX+n.....cantidadMensajesEnviados

Nota: Tener en cuenta que, en el listado, deberán aparecer todos los usuarios que existen en el sistema. Considerar la implementación de esta opción de las siguientes maneras:

- *Como un procedimiento separado del inciso (a).*
- *En el mismo procedimiento de actualización del inciso (a). ¿Qué cambios se requieren en el procedimiento del inciso (a) para realizar el informe en el mismo recorrido?*

```
program TP2_E13;
{$codepage UTF8}
uses crt, sysutils;
const
  usuario_salida=999;
type
  t_string20=string[20];
  t_registro_usuario1=record
    usuario: int16;
    nombre_usuario: t_string20;
    nombre: t_string20;
    apellido: t_string20;
    mails: int16;
  end;
  t_registro_usuario2=record
    usuario: int16;
    destino: t_string20;
    mensaje: t_string20;
  end;
  t_archivo_maestro=file of t_registro_usuario1;
  t_archivo_detalle=file of t_registro_usuario2;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
```

```
var
  registro_usuario: t_registro_usuario1;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_usuario do
    begin
      readln(archivo_carga_maestro,usuario,mails,nombre_usuario);
nombre_usuario:=trim(nombre_usuario);
      readln(archivo_carga_maestro,nombre);
      readln(archivo_carga_maestro,apellido);
      write(archivo_maestro,registro_usuario);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_usuario: t_registro_usuario2;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_usuario do
    begin
      readln(archivo_carga_detalle,usuario,destino); destino:=trim(destino);
      readln(archivo_carga_detalle,mensaje);
      write(archivo_detalle,registro_usuario);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_usuario1(registro_usuario: t_registro_usuario1);
begin
  textcolor(green); write('Número de usuario: '); textcolor(yellow);
write(registro_usuario.usuario);
  textcolor(green); write('; Nombre de usuario: '); textcolor(yellow);
write(registro_usuario.nombre_usuario);
  textcolor(green); write('; Nombre: '); textcolor(yellow); write(registro_usuario.nombre);
  textcolor(green); write('; Apellido: '); textcolor(yellow);
write(registro_usuario.apellido);
  textcolor(green); write('; Mails enviados: '); textcolor(yellow);
writeln(registro_usuario.mails);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_usuario: t_registro_usuario1;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_usuario);
    imprimir_registro_usuario1(registro_usuario);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_usuario2(registro_usuario: t_registro_usuario2);
begin
  textcolor(green); write('Número de usuario: '); textcolor(yellow);
write(registro_usuario.usuario);
  textcolor(green); write('; Cuenta destino: '); textcolor(yellow);
write(registro_usuario.destino);
  textcolor(green); write('; Cuerpo del mensaje: '); textcolor(yellow);
writeln(registro_usuario.mensaje);
```

```
end;
procedure imprimir_archivo_detalle(var archivo_detalle: t_archivo_detalle);
var
  registro_usuario: t_registro_usuario2;
begin
  reset(archivo_detalle);
  while (not eof(archivo_detalle)) do
  begin
    read(archivo_detalle,registro_usuario);
    imprimir_registro_usuario2(registro_usuario);
  end;
  close(archivo_detalle);
end;
procedure leer_usuario(var archivo_detalle: t_archivo_detalle; var registro_usuario:
t_registro_usuario2);
begin
  if (not eof(archivo_detalle)) then
    read(archivo_detalle,registro_usuario)
  else
    registro_usuario.usuario:=usuario_salida;
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_detalle: t_archivo_detalle);
var
  registro_maestro: t_registro_usuario1;
  registro_detalle: t_registro_usuario2;
begin
  reset(archivo_maestro);
  reset(archivo_detalle);
  leer_usuario(archivo_detalle,registro_detalle);
  while (registro_detalle.usuario<>usuario_salida) do
  begin
    read(archivo_maestro,registro_maestro);
    while (registro_maestro.usuario<>registro_detalle.usuario) do
      read(archivo_maestro,registro_maestro);
    while (registro_maestro.usuario=registro_detalle.usuario) do
    begin
      registro_maestro.mails:=registro_maestro.mails+1;
      leer_usuario(archivo_detalle,registro_detalle);
    end;
    textcolor(green); write('Número de usuario: '); textcolor(yellow);
write(registro_maestro.usuario); textcolor(green); write(' .......... Cantidad de Mensajes
Enviados: '); textcolor(red); writeln(registro_maestro.mails);
    seek(archivo_maestro,filepos(archivo_maestro)-1);
    write(archivo_maestro,registro_maestro);
  end;
  close(archivo_maestro);
  close(archivo_detalle);
  writeln();
end;
procedure exportar_archivo_txt(var archivo_maestro: t_archivo_maestro);
var
  registro_usuario: t_registro_usuario1;
  archivo_txt: text;
begin
  reset(archivo_maestro);
  assign(archivo_txt,'E13_usuarios.txt'); rewrite(archivo_txt);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_usuario);
    writeln(archivo_txt,registro_usuario.usuario,' ',registro_usuario.mails);
  end;
  close(archivo_maestro);
  close(archivo_txt);
end;
var
```

```
  archivo_maestro: t_archivo_maestro;
  archivo_detalle: t_archivo_detalle;
  archivo_carga_maestro, archivo_carga_detalle: text;
begin
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E13_usuariosMaestro');
assign(archivo_carga_maestro,'E13_usuariosMaestro.txt');
  cargar_archivo_maestro(archivo_maestro,archivo_carga_maestro);
  imprimir_archivo_maestro(archivo_maestro);
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO DETALLE:'); writeln();
  assign(archivo_detalle,'E13_usuariosDetalle');
assign(archivo_carga_detalle,'E13_usuariosDetalle.txt');
  cargar_archivo_detalle(archivo_detalle,archivo_carga_detalle);
  imprimir_archivo_detalle(archivo_detalle);
  writeln(); textcolor(red); writeln('IMRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  actualizar_archivo_maestro(archivo_maestro,archivo_detalle);
  imprimir_archivo_maestro(archivo_maestro);
  exportar_archivo_txt(archivo_maestro);
end.
```

Ejercicio 14.

Una compañía aérea dispone de un archivo maestro donde guarda información sobre sus próximos vuelos. En dicho archivo, se tiene almacenado el destino, fecha, hora de salida y la cantidad de asientos disponibles. La empresa recibe todos los días dos archivos detalles para actualizar el archivo maestro. En dichos archivos, se tiene destino, fecha, hora de salida y cantidad de asientos comprados. Se sabe que los archivos están ordenados por destino más fecha y hora de salida, y que, en los detalles, pueden venir 0, 1 o más registros por cada uno del maestro. Se pide realizar los módulos necesarios para:

(a) *Actualizar el archivo maestro sabiendo que no se registró ninguna venta de pasaje sin asiento disponible.*

(b) *Generar una lista con aquellos vuelos (destino, fecha y hora de salida) que tengan menos de una cantidad específica de asientos disponibles. La misma debe ser ingresada por teclado.*

Nota: El archivo maestro y los archivos detalles sólo pueden recorrerse una vez.

```
program TP2_E14;
{$codepage UTF8}
uses crt, sysutils;
const
  detalles_total=2;
  destino_salida='ZZZ';
type
  t_detalle=1..detalles_total;
  t_string20=string[20];
  t_registro_vuelo=record
    destino: t_string20;
    fecha: t_string20;
    hora: t_string20;
    asientos_disponibles: int16;
  end;
  t_registro_venta=record
    destino: t_string20;
    fecha: t_string20;
    hora: t_string20;
    asientos_vendidos: int16;
  end;
  t_archivo_maestro=file of t_registro_vuelo;
  t_archivo_detalle=file of t_registro_venta;
  t_vector_ventas=array[t_detalle] of t_registro_venta;
  t_vector_detalles=array[t_detalle] of t_archivo_detalle;
  t_vector_carga_detalles=array[t_detalle] of text;
procedure cargar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
archivo_carga_maestro: text);
var
  registro_vuelo: t_registro_vuelo;
begin
  rewrite(archivo_maestro);
  reset(archivo_carga_maestro);
  while (not eof(archivo_carga_maestro)) do
    with registro_vuelo do
    begin
      readln(archivo_carga_maestro,asientos_disponibles,destino); destino:=trim(destino);
      readln(archivo_carga_maestro,fecha);
```

```
      readln(archivo_carga_maestro,hora);
      write(archivo_maestro,registro_vuelo);
    end;
  close(archivo_maestro);
  close(archivo_carga_maestro);
end;
procedure cargar_archivo_detalle(var archivo_detalle: t_archivo_detalle; var
archivo_carga_detalle: text);
var
  registro_venta: t_registro_venta;
begin
  rewrite(archivo_detalle);
  reset(archivo_carga_detalle);
  while (not eof(archivo_carga_detalle)) do
    with registro_venta do
    begin
      readln(archivo_carga_detalle,asientos_vendidos,destino); destino:=trim(destino);
      readln(archivo_carga_detalle,fecha);
      readln(archivo_carga_detalle,hora);
      write(archivo_detalle,registro_venta);
    end;
  close(archivo_detalle);
  close(archivo_carga_detalle);
end;
procedure imprimir_registro_vuelo(registro_vuelo: t_registro_vuelo);
begin
  textcolor(green); write('Destino: '); textcolor(yellow); write(registro_vuelo.destino);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_vuelo.fecha);
  textcolor(green); write('; Hora: '); textcolor(yellow); write(registro_vuelo.hora);
  textcolor(green); write('; Asientos disponibles: '); textcolor(yellow);
writeln(registro_vuelo.asientos_disponibles);
end;
procedure imprimir_archivo_maestro(var archivo_maestro: t_archivo_maestro);
var
  registro_vuelo: t_registro_vuelo;
begin
  reset(archivo_maestro);
  while (not eof(archivo_maestro)) do
  begin
    read(archivo_maestro,registro_vuelo);
    imprimir_registro_vuelo(registro_vuelo);
  end;
  close(archivo_maestro);
end;
procedure imprimir_registro_venta(registro_venta: t_registro_venta);
begin
  textcolor(green); write('Destino: '); textcolor(yellow); write(registro_venta.destino);
  textcolor(green); write('; Fecha: '); textcolor(yellow); write(registro_venta.fecha);
  textcolor(green); write('; Hora: '); textcolor(yellow); write(registro_venta.hora);
  textcolor(green); write('; Asientos vendidos: '); textcolor(yellow);
writeln(registro_venta.asientos_vendidos);
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.destino:=destino_salida;
end;
procedure minimo(var vector_detalles: t_vector_detalles; var vector_ventas: t_vector_ventas;
var min: t_registro_venta);
var
  i, pos: t_detalle;
begin
  min.destino:=destino_salida;
  for i:= 1 to detalles_total do
    if ((vector_ventas[i].destino<min.destino) or ((vector_ventas[i].destino=min.destino) and
(vector_ventas[i].fecha<min.fecha)) or ((vector_ventas[i].destino=min.destino) and
(vector_ventas[i].fecha=min.fecha) and (vector_ventas[i].hora<min.hora))) then
    begin
      min:=vector_ventas[i];
      pos:=i;
    end;
  if (min.destino<destino_salida) then
    leer_venta(vector_detalles[pos],vector_ventas[pos]);
end;
procedure actualizar_archivo_maestro(var archivo_maestro: t_archivo_maestro; var
vector_detalles: t_vector_detalles; asientos: int16);
var
  registro_vuelo: t_registro_vuelo;
  min: t_registro_venta;
  vector_ventas: t_vector_ventas;
  i: t_detalle;
begin
  reset(archivo_maestro);
  for i:= 1 to detalles_total do
  begin
    reset(vector_detalles[i]);
    leer_venta(vector_detalles[i],vector_ventas[i]);
  end;
  minimo(vector_detalles,vector_ventas,min);
  while (min.destino<>destino_salida) do
  begin
    read(archivo_maestro,registro_vuelo);
    while (registro_vuelo.destino<>min.destino) do
      read(archivo_maestro,registro_vuelo);
    while (registro_vuelo.destino=min.destino) do
    begin
      while (registro_vuelo.fecha<>min.fecha) do
        read(archivo_maestro,registro_vuelo);
      while ((registro_vuelo.destino=min.destino) and (registro_vuelo.fecha=min.fecha)) do
      begin
        while (registro_vuelo.hora<>min.hora) do
          read(archivo_maestro,registro_vuelo);
        while ((registro_vuelo.destino=min.destino) and (registro_vuelo.fecha=min.fecha) and
(registro_vuelo.hora=min.hora)) do
        begin
          registro_vuelo.asientos_disponibles:=registro_vuelo.asientos_disponibles-
min.asientos_vendidos;
          minimo(vector_detalles,vector_ventas,min);
        end;
        seek(archivo_maestro,filepos(archivo_maestro)-1);
        write(archivo_maestro,registro_vuelo);
        if (registro_vuelo.asientos_disponibles<asientos) then
        begin
          textcolor(green); write('El vuelo con destino a '); textcolor(yellow);
write(registro_vuelo.destino); textcolor(green); write(' del '); textcolor(yellow);
write(registro_vuelo.fecha); textcolor(green); write(' a las '); textcolor(yellow);
```

```
write(registro_vuelo.hora); textcolor(green); write(' tiene menos de '); textcolor(yellow);
write(asientos); textcolor(green); writeln(' asientos disponibles');
        end;
      end;
    end;
  end;
  close(archivo_maestro);
  for i:= 1 to detalles_total do
    close(vector_detalles[i]);
  writeln();
end;
var
  vector_detalles: t_vector_detalles;
  vector_carga_detalles: t_vector_carga_detalles;
  archivo_maestro: t_archivo_maestro;
  archivo_carga_maestro: text;
  i: t_detalle;
  asientos: int16;
begin
  randomize;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO:'); writeln();
  assign(archivo_maestro,'E14_vuelosMaestro');
assign(archivo_carga_maestro,'E14_vuelosMaestro.txt');
  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();
    assign(vector_detalles[i],'E14_ventasDetalle'+inttoStr(i));
assign(vector_carga_detalles[i],'E14_ventasDetalle'+inttoStr(i)+'.txt');
    cargar_archivo_detalle(vector_detalles[i],vector_carga_detalles[i]);
    imprimir_archivo_detalle(vector_detalles[i]);
  end;
  writeln(); textcolor(red); writeln('IMPRESIÓN ARCHIVO MAESTRO ACTUALIZADO:'); writeln();
  asientos:=10+random(291);
  actualizar_archivo_maestro(archivo_maestro,vector_detalles,asientos);
  imprimir_archivo_maestro(archivo_maestro);
end.
```

Ejercicio 15.

Se desea modelar la información de una ONG dedicada a la asistencia de personas con carencias habitacionales. La ONG cuenta con un archivo maestro conteniendo información como se indica a continuación: código provincia, nombre provincia, código de localidad, nombre de localidad, #viviendas sin luz, #viviendas sin gas, #viviendas de chapa, #viviendas sin agua, # viviendas sin sanitarios.

Mensualmente, reciben detalles de las diferentes provincias indicando avances en las obras de ayuda en la edificación y equipamientos de viviendas en cada provincia. La información de los detalles es la siguiente: código provincia, código localidad, #viviendas con luz, #viviendas construidas, #viviendas con agua, #viviendas con gas, #entrega sanitarios.

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 provincia y código de localidad.

Para la actualización del archivo maestro, se debe proceder de la siguiente manera:

- *Al valor de viviendas sin luz se le resta el valor recibido en el detalle.*
- *Ídem para viviendas sin agua, sin gas y sin sanitarios.*
- *A las viviendas de chapa se le resta el valor recibido de viviendas construidas.*

La misma combinación de provincia y localidad aparecen, a lo sumo, una única vez.

Realizar las declaraciones necesarias, el programa principal y los procedimientos que se requiera para la actualización solicitada e informar cantidad de localidades sin viviendas de chapa (las localidades pueden o no haber sido actualizadas).

Ejercicio 16.

La editorial X, autora de diversos semanarios, posee un archivo maestro con la información correspondiente a las diferentes emisiones de los mismos. De cada emisión, se registra: fecha, código de semanario, nombre del semanario, descripción, precio, total de ejemplares y total de ejemplares vendidos.

Mensualmente, se reciben 100 archivos detalles con las ventas de los semanarios en todo el país. La información que poseen los detalles es la siguiente: fecha, código de semanario y cantidad de ejemplares vendidos. Realizar las declaraciones necesarias, la llamada al procedimiento y el procedimiento que recibe el archivo maestro y los 100 detalles y realizar la actualización del archivo maestro en función de las ventas registradas. Además, se deberá informar fecha y semanario que tuvo más ventas y la misma información del semanario con menos ventas.

Nota: Todos los archivos están ordenados por fecha y código de semanario. No se realizan ventas de semanarios si no hay ejemplares para hacerlo.

Ejercicio 17.

Una concesionaria de motos de la Ciudad de Chascomús posee un archivo con información de las motos que posee a la venta. De cada moto, se registra: código, nombre, descripción, modelo, marca y stock actual. Mensualmente, se reciben 10 archivos detalles con información de las ventas de cada uno de los 10 empleados que trabajan. De cada archivo detalle, se dispone de la siguiente información: código de moto, precio y fecha de la venta. Se debe realizar un proceso que actualice el stock del archivo maestro desde los archivos detalles. Además, se debe informar cuál fue la moto más vendida.

Nota: Todos los archivos están ordenados por código de la moto y el archivo maestro debe ser recorrido sólo una vez y en forma simultánea con los detalles.

Ejercicio 18.

Se cuenta con un archivo con información de los casos de COVID-19 registrados en los diferentes hospitales de la Provincia de Buenos Aires cada día. Dicho archivo contiene: código de localidad, nombre de localidad, código de municipio, nombre de municipio, código de hospital, nombre de hospital, fecha y cantidad de casos positivos detectados. El archivo está ordenado por localidad, luego por municipio y luego por hospital.

Escribir la definición de las estructuras de datos necesarias y un procedimiento que haga un listado con el siguiente formato:

Nombre: Localidad 1
  Nombre: Municipio 1
    Nombre Hospital 1.....Cantidad de casos Hospital 1
    .....
    Nombre Hospital N.....Cantidad de casos Hospital N
  Cantidad de casos Municipio 1
  .....
  Nombre Municipio N
    Nombre Hospital 1.....Cantidad de casos Hospital 1
    .....
    Nombre Hospital N.....Cantidad de casos Hospital N
  Cantidad de casos Municipio N
Cantidad de casos Localidad 1

Nombre Localidad N
Nombre Municipio 1
    Nombre Hospital 1.....Cantidad de casos Hospital 1
    .....
    Nombre Hospital N.....Cantidad de casos Hospital N
  Cantidad de casos Municipio 1
  .....
  Nombre Municipio N
    Nombre Hospital 1.....Cantidad de casos Hospital 1
    .....
    Nombre Hospital N.....Cantidad de casos Hospital N
  Cantidad de casos Municipio N
Cantidad de casos Localidad N

Cantidad de casos Totales en la Provincia

Además del informe en pantalla anterior, es necesario exportar a un archivo de texto la siguiente información: nombre de localidad, nombre de municipio y cantidad de casos del municipio, para aquellos municipios cuya cantidad de casos supere los 1500. El formato del archivo de texto deberá ser el adecuado para recuperar la información con la menor cantidad de lecturas posibles.

Nota: El archivo debe recorrerse sólo una vez.

Ejercicio 19.

A partir de un siniestro ocurrido, se perdieron las actas de nacimiento y fallecimientos de toda la Provincia de Buenos Aires de los últimos diez años. En pos de recuperar dicha información, se deberá procesar 2 archivos por cada una de las 50 delegaciones distribuidas en la provincia, un archivo de nacimientos y otro de fallecimientos, y crear el archivo maestro reuniendo dicha información.

Los archivos detalles con nacimientos contendrán la siguiente información: nro. partida nacimiento, nombre, apellido, dirección detallada (calle, nro., piso, depto, ciudad), matrícula del médico, nombre y apellido de la madre, DNI madre, nombre y apellido del padre, DNI del padre.

En cambio, los 50 archivos de fallecimientos tendrán: nro. partida nacimiento, DNI, nombre y apellido del fallecido, matrícula del médico que firma el deceso, fecha y hora del deceso y lugar.

Realizar un programa que cree el archivo maestro a partir de toda la información de los archivos detalles. Se debe almacenar en el maestro: nro. partida nacimiento, nombre, apellido, dirección detallada (calle, nro., piso, depto, ciudad), matrícula del médico, nombre y apellido de la madre, DNI madre, nombre y apellido del padre, DNI del padre y, si falleció, además, matrícula del médico que firma el deceso, fecha y hora del deceso y lugar. Se deberá, además, listar en un archivo de texto la información recolectada de cada persona.

Nota: Todos los archivos están ordenados por nro. partida de nacimiento, que es única. Tener en cuenta que no necesariamente va a fallecer en el distrito donde nació la persona y, además, puede no haber fallecido.

Juan Menduiña

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.

Juan Menduiña

Trabajo Práctico N° 4:
Árboles B.

Políticas para la resolución de underflow:

- *Política izquierda: Se intenta redistribuir con el hermano adyacente izquierdo; si no es posible, se fusiona con hermano adyacente izquierdo.*
- *Política derecha: Se intenta redistribuir con el hermano adyacente derecho; si no es posible, se fusiona con hermano adyacente derecho.*
- *Política izquierda o derecha: Se intenta redistribuir con el hermano adyacente izquierdo; si no es posible, se intenta con el hermano adyacente derecho; si tampoco es posible, se fusiona con hermano adyacente izquierdo.*
- *Política derecha o izquierda: Se intenta redistribuir con el hermano adyacente derecho; si no es posible, se intenta con el hermano adyacente izquierdo; si tampoco es posible, se fusiona con hermano adyacente derecho.*

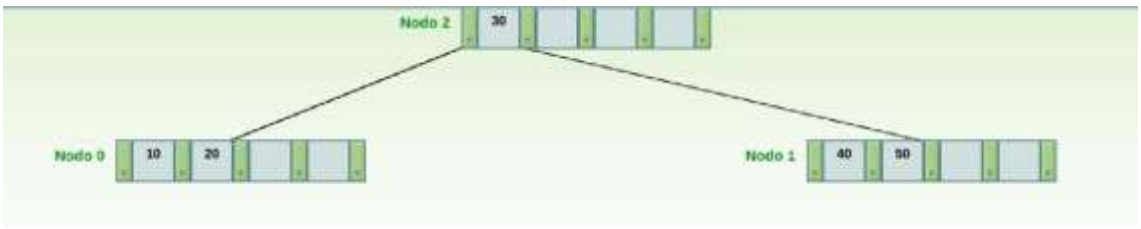
Casos especiales: En cualquier política, si se tratase de un nodo hoja de un extremo del árbol, debe intentarse redistribuir con el hermano adyacente que el mismo posea.

Aclaración:

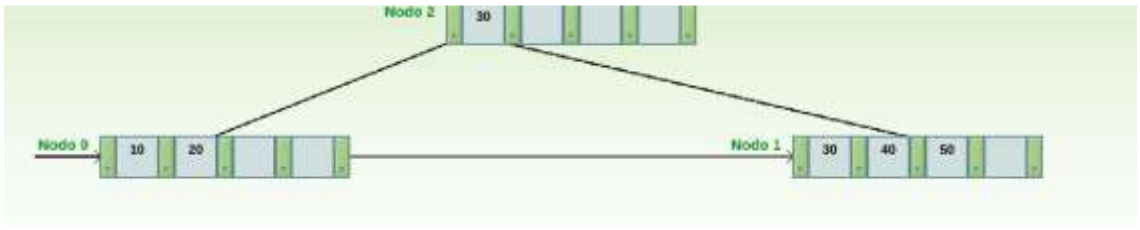
- *En caso de underflow, lo primero que se intenta SIEMPRE es redistribuir y el hermano adyacente se encuentra en condiciones de ceder elementos si, al hacerlo, no se produce underflow en él.*
- *En caso de overflow, SIEMPRE se genera un nuevo nodo. Las claves se distribuyen, equitativamente, entre el nodo desbordado y el nuevo.*

En el caso de órdenes impares, se debe promocionar la clave o la copia (en árbol B+) que se encuentra en la posición del medio.

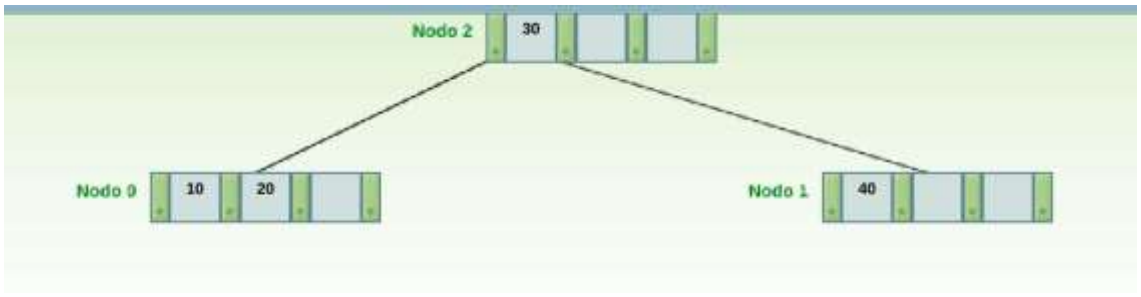
Ejemplo árbol B, orden 5:



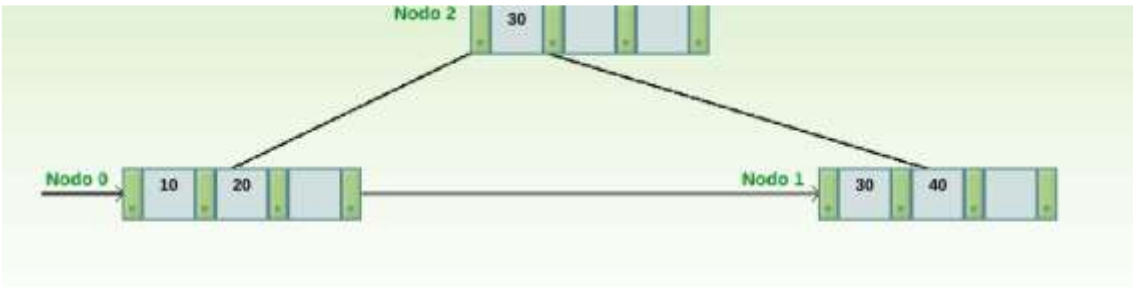
Ejemplo árbol B+, orden 5:



Ejemplo árbol B, orden 4:



Ejemplo árbol B+, orden 4:



PARTE I: Archivos de Datos, Índices y Árboles B.

Ejercicio 1.

Considerar que se desea almacenar, en un archivo, la información correspondiente a los alumnos de la Facultad de Informática de la UNLP. De los mismos, se deberá guardar nombre y apellido, DNI, legajo y año de ingreso. Suponer que dicho archivo se organiza como un árbol B de orden M.

(a) *Definir, en Pascal, las estructuras de datos necesarias para organizar el archivo de alumnos como un árbol B de orden M.*

```
program TP4_E1;
const
  M=10;
type
  t_registro_alumno=record
    nombre: string;
    apellido: string;
    dni: int32;
    legajo: int16;
    anio_ingreso: int16;
  end;
  t_registro_nodo=record
    cantidad_datos: int16;
    datos: array[0..M-1] of t_registro_alumno;
    hijos: array[0..M] of int16;
  end;
  t_arbol_datos=file of t_registro_nodo;
var
  archivo_datos: t_archivo_datos;
begin

end.
```

(b) *Suponer que la estructura de datos que representa una persona (registro de persona) ocupa 64 bytes, que cada nodo del árbol B tiene un tamaño de 512 bytes y que los números enteros ocupan 4 bytes, ¿cuántos registros de persona entrarían en un nodo del árbol B? ¿Cuál sería el orden del árbol B en este caso (el valor de M)? Para resolver este inciso, se puede utilizar la fórmula N= (M - 1) * A + M * B + C, donde N es el tamaño del nodo (en bytes), A es el tamaño de un registro (en bytes), B es el tamaño de cada enlace a un hijo y C es el tamaño que ocupa el campo referido a la cantidad de claves. El objetivo es reemplazar estas variables con los valores dados y obtener el valor de M (M debe ser un número entero, ignorar la parte decimal).*

N= (M - 1) * A + M * B + C
N= AM - A + BM + C
512= 64M - 64 + 4M + 4
512= 68M - 60
68M= 512 + 60
68M= 572
$M= \frac{572}{68}$

M≅ 8,41.

Por lo tanto, entrarían 7 registros de persona en un nodo del árbol B, ya que el orden del árbol B en este caso (el valor de M) sería 8.

(c) *¿Qué impacto tiene sobre el valor de M organizar el archivo con toda la información de los alumnos como un árbol B?*

El impacto que tiene sobre el valor de M organizar el archivo con toda la información de los alumnos como un árbol B es directo:

- Cuanto más grande sea el registro de alumno, menor será el valor de M, lo cual disminuye la eficiencia del árbol B, ya que se incrementa su profundidad y, con ello, el costo en tiempo y recursos para buscar o modificar datos.
- Cuanto más chico sea el registro de alumno, mayor será el valor de M, lo cual aumenta la eficiencia del árbol B, ya que se reduce su profundidad y, con ello, el costo en tiempo y recursos para buscar o modificar datos.

(d) *¿Qué dato se seleccionaría como clave de identificación para organizar los elementos (alumnos) en el árbol B? ¿Hay más de una opción?*

Los datos que se seleccionaría como clave de identificación para organizar los elementos (alumnos) en el árbol B son el DNI o el legajo, ya que ambos son únicos por alumno.

Sí, hay más de una opción, pero las claves numéricas y únicas son las más eficientes para un árbol B.

(e) *Describir el proceso de búsqueda de un alumno por el criterio de ordenamiento especificado en el inciso previo. ¿Cuántas lecturas de nodos se necesitan para encontrar un alumno por su clave de identificación en el peor y en el mejor de los casos? ¿Cuáles serían estos casos?*

El proceso de búsqueda de un alumno por el criterio de ordenamiento especificado en el inciso previo es:

- Se comienza en la raíz del árbol B.
- Se comparan las claves almacenadas en ese nodo con la clave buscada (DNI).
- Si la clave está en ese nodo, la búsqueda termina.
- Si la clave no está en ese nodo:
    - Se determina en qué subárbol (hijo) continuar según el rango de claves.
    - Se lee el nodo hijo correspondiente.
- Se repite el proceso hasta:
    - Encontrar la clave (caso exitoso).
    - O llegar a un nodo hoja sin encontrarla (caso fallido).

Para encontrar un alumno por su clave de identificación, se necesitan:

- En el peor de los casos, h lecturas, siendo h la altura del árbol. Este caso ocurre cuando el alumno buscado no está en el árbol o está en un nodo hoja (nodo del último nivel) del árbol.
- En el mejor de los casos, una única lectura. Este caso ocurre cuando el alumno buscado está en el nodo raíz.

(f) *¿Qué ocurre si se desea buscar un alumno por un criterio diferente? ¿Cuántas lecturas serían necesarias en el peor de los casos?*

Lo que ocurre si se desea buscar un alumno por un criterio diferente al criterio de ordenamiento es que no se puede aprovechar la estructura jerárquica del árbol, por lo que se debe realizar una búsqueda secuencial (lineal), es decir, recorrer todos los registros almacenados, sin ayuda de la estructura del árbol.

En el peor de los casos, serían necesarias n lecturas, siendo n la cantidad total de nodos del árbol.

Ejercicio 2.

Una mejora respecto a la solución propuesta en el Ejercicio 1 sería mantener, por un lado, el archivo que contiene la información de los alumnos de la Facultad de Informática (archivo de datos no ordenado) y, por otro lado, mantener un índice al archivo de datos que se estructura como un árbol B que ofrece acceso indizado por DNI de los alumnos.

(a) *Definir, en Pascal, las estructuras de datos correspondientes para el archivo de alumnos y su índice.*

```
program TP4_E2;
const
  M=10;
type
  t_registro_alumno=record
    nombre: string;
    apellido: string;
    dni: int32;
    legajo: int16;
    anio_ingreso: int16;
  end;
  t_registro_nodo=record
    cantidad_claves: int16;
    claves: array[0..M-1] of int32;
    enlaces: array[0..M-1] of int16;
    hijos: array[0..M] of int16;
  end;
  t_archivo_datos=file of t_registro_alumno;
  t_archivo_indice=file of t_registro_nodo;
var
  archivo_datos: t_archivo_datos;
  archivo_indice: t_archivo_indice;
begin

end.
```

(b) *Suponer que cada nodo del árbol B cuenta con un tamaño de 512 bytes. ¿Cuál sería el orden del árbol B (valor de M) que se emplea como índice? Asumir que los números enteros ocupan 4 bytes. Para este inciso, se puede emplear una fórmula similar al inciso (b) del Ejercicio 1, pero considerar, además, que, en cada nodo, se deben almacenar los M - 1 enlaces a los registros correspondientes en el archivo de datos.*

N= (M - 1) * A + (M - 1) * A + M * B + C
N= 2 * (M - 1) * A + BM + C
N= 2AM - 2A + BM + C
512= 2 * 4M - 2 * 4 + 4M + 4
512= 8M - 8 + 4M + 4
512= 12M - 4
12M= 512 + 4
12M= 516
$M= \frac{516}{12}$
M= 43.

Por lo tanto, el orden del árbol B (valor de M) que se emplea como índice sería 43.

(c) *¿Qué implica que el orden del árbol B sea mayor que en el caso del Ejercicio 1?*

Lo que implica que el orden del árbol B sea mayor que en el caso del Ejercicio 1 es que aumenta la eficiencia del árbol B, ya que se reduce su profundidad y, con ello, el costo en tiempo y recursos para buscar o modificar datos.

(d) *Describir el proceso para buscar el alumno con el DNI 12345678 usando el índice definido en este ejercicio.*

El proceso para buscar el alumno con el DNI 12345678 usando el índice definido en este ejercicio es:

- Se comienza en la raíz del árbol B.
- Se comparan las claves almacenadas en ese nodo con la clave buscada (DNI 12345678).
- Si la clave está en ese nodo, la búsqueda termina y:
    - Se recupera el puntero al registro correspondiente en el archivo de datos.
    - Usando el puntero recuperado, se lee, de manera directa, el registro de alumno en el archivo de datos.
- Si la clave no está en ese nodo:
    - Se determina en qué subárbol (hijo) continuar según el rango de claves.
    - Se lee el nodo hijo correspondiente.
- Se repite el proceso hasta:
    - Encontrar la clave (caso exitoso).
    - O llegar a un nodo hoja sin encontrarla (caso fallido).

(e) *¿Qué ocurre si desea buscar un alumno por su número de legajo? ¿Tiene sentido usar el índice que organiza el acceso al archivo de alumnos por DNI? ¿Cómo se haría para brindar acceso indizado al archivo de alumnos por número de legajo?*

Lo que ocurre si se desea buscar un alumno por un criterio diferente al criterio de ordenamiento (por ejemplo, por su número de legajo) es que no se puede aprovechar la estructura jerárquica del árbol, por lo que se debe realizar una búsqueda secuencial (lineal), es decir, recorrer todos los registros almacenados, sin ayuda de la estructura del árbol.

No, no tiene sentido usar el índice que organiza el acceso al archivo de alumnos por DNI, ya que este índice sólo ordena y permite búsquedas rápidas por DNI y, por lo tanto, no sirve para buscar por número de legajo.

Para brindar acceso indizado al archivo de alumnos por número de legajo, se debería crear un segundo índice al archivo de datos estructurado como un árbol B que ofrezca acceso indizado, ahora, por número de legajo de los alumnos.

(f) *Suponer que se desea buscar los alumnos que tienen DNI en el rango entre 40000000 y 45000000. ¿Qué problemas tiene este tipo de búsquedas con apoyo de un árbol B que sólo provee acceso indizado por DNI al archivo de alumnos?*

Los problemas que tiene este tipo de búsqueda con apoyo de un árbol B que sólo provee acceso indizado por DNI al archivo de alumnos son que no se puede hacer un recorrido secuencial entre nodos hoja fácilmente y, por lo tanto, la búsqueda por rango requiere búsquedas independientes por cada valor dentro del rango, lo cual puede volverse muy ineficiente si hay muchos valores en el rango.

Ejercicio 3.

Los árboles B+ representan una mejora sobre los árboles B dado que conservan la propiedad de acceso indexado a los registros del archivo de datos por alguna clave, pero permiten, además, un recorrido secuencial rápido. Al igual que en el Ejercicio 2, considerar que, por un lado, se tiene el archivo que contiene la información de los alumnos de la Facultad de Informática (archivo de datos no ordenado) y, por otro lado, se tiene un índice al archivo de datos, pero, en este caso, el índice se estructura como un árbol B+ que ofrece acceso indizado por DNI al archivo de alumnos. Resolver los siguientes incisos:

(a) *¿Cómo se organizan los elementos (claves) de un árbol B+? ¿Qué elementos se encuentran en los nodos internos y que elementos se encuentran en los nodos hojas?*

Los elementos (claves) de un árbol B+ se organizan de forma jerárquica, con dos tipos principales de nodos:

- Los elementos que se encuentran en los nodos internos son las claves, que se utilizan para dirigir la búsqueda hacia los nodos hoja adecuados.
- Los elementos que se encuentran en los nodos hoja son las claves, los punteros a los registros y el puntero al siguiente nodo hoja.

(b) *¿Qué característica distintiva presentan los nodos hojas de un árbol B+? ¿Por qué?*

La característica distintiva que presentan los nodos hojas de un árbol B+ es que contienen todas las claves del árbol y están enlazados secuencialmente, lo cual hace a este tipo de árbol ideal para búsquedas exactas y por rango.

(c) *Definir, en Pascal, las estructuras de datos correspondientes para el archivo de alumnos y su índice (árbol B+). Por simplicidad, suponer que todos los nodos del árbol B+ (nodos internos y nodos hojas) tienen el mismo tamaño.*

```
program TP4_E3;
const
  M=10;
type
  t_registro_alumno=record
    nombre: string;
    apellido: string;
    dni: int32;
    legajo: int16;
    anio_ingreso: int16;
  end;
  t_lista=^t_registro_nodo;
  t_registro_nodo=record
    cantidad_claves: int16;
    claves: array[0..M-1] of int32;
    enlaces: array[0..M-1] of int16;
    hijos: array[0..M] of int16;
```

```
    sig: t_lista;
  end;
  t_archivo_datos=file of t_registro_alumno;
  t_archivo_indice=file of t_registro_nodo;
var
  archivo_datos: t_archivo_datos;
  archivo_indice: t_archivo_indice;
begin

end.
```

(d) *Describir el proceso de búsqueda de un alumno con un DNI específico haciendo uso de la estructura auxiliar (índice) que se organiza como un árbol B+. ¿Qué diferencia se encuentra respecto a la búsqueda en un índice estructurado como un árbol B?*

El proceso de búsqueda de un alumno con un DNI específico haciendo uso de la estructura auxiliar (índice) que se organiza como un árbol B+ es:

- Se comienza en la raíz del árbol B+.
- Se comparan las claves almacenadas en ese nodo con la clave buscada (DNI específico).
- Se elige el puntero al hijo adecuado según el rango en el que cae el DNI buscado.
- Se repite este proceso hasta llegar a un nodo hoja.
- Si la clave está en el nodo hoja (caso exitoso), la búsqueda termina y:
    - Se recupera el puntero al registro correspondiente en el archivo de datos.
    - Usando el puntero recuperado, se lee, de manera directa, el registro de alumno en el archivo de datos.
- Si la clave no está en el nodo hoja (caso fallido), la búsqueda también termina.

(e) *Explicar el proceso de búsqueda de los alumnos que tienen DNI en el rango entre 40000000 y 45000000, apoyando la búsqueda en un índice organizado como un árbol B+. ¿Qué ventajas se encuentran respecto a este tipo de búsquedas en un árbol B?*

El proceso de búsqueda de los alumnos que tienen DNI en el rango entre 40000000 y 45000000, apoyando la búsqueda en un índice organizado como un árbol B+ es:

- Se comienza en la raíz del árbol B+.
- Se comparan las claves almacenadas en ese nodo con la clave mínima del rango (DNI 40000000).
- Se elige el puntero al hijo adecuado según el rango en el que cae el DNI mínimo.
- Se repite este proceso hasta llegar a un nodo hoja.
- Se busca la primera clave mayor o igual al DNI mínimo y menor o igual al DNI máximo (clave máxima del rango, es decir, 45000000).
- Si se encuentra una clave en este rango en el nodo hoja:
    - Se recupera el puntero al registro correspondiente en el archivo de datos.
    - Se siguen leyendo claves y recuperando los punteros hasta que el DNI sea mayor al DNI máximo, que es cuando la búsqueda termina.

    - Usando los punteros recuperados, se leen, de manera directa, los registros de alumno en el archivo de datos.
- Si no se encuentra una clave en este rango en el nodo hoja, la búsqueda termina.

Las ventajas que se encuentran respecto a este tipo de búsquedas en un árbol B son que se puede hacer un recorrido secuencial entre nodos hoja fácilmente y, por lo tanto, la búsqueda por rango no requiere búsquedas independientes por cada valor dentro del rango, lo cual es muy eficiente si hay muchos valores en el rango.

Ejercicio 4.

Dado el siguiente algoritmo de búsqueda en un árbol B:

```
procedure buscar(NRR, clave, NRR_encontrado, pos_encontrada, resultado);
var
  clave_encontrada: boolean;
begin
  if (nodo=null)
    resultado:=false
  else
  begin
    posicionarYLeerNodo(A,nodo,NRR);
    claveEncontrada(A,nodo,clave,pos,clave_encontrada);
    if (clave_encontrada) then
    begin
      NRR_encontrado:=NRR;
      pos_encontrada:=pos;
      resultado:=true;
    end
    else
      buscar(nodo.hijos[pos],clave,NRR_encontrado,pos_encontrada,resultado)
  end;
end;
```

Asumir que el archivo se encuentra abierto y que, para la primera llamada, el parámetro NRR contiene la posición de la raíz del árbol. Responder detalladamente:

(a) *PosicionarYLeerNodo(): Indicar qué hace y la forma en que deben ser enviados los parámetros (valor o referencia). Implementar este módulo en Pascal.*

PosicionarYLeerNodo() es un procedimiento que permite acceder, directamente, a un nodo del árbol B en el archivo, usando su número lógico (*NRR*), y traerlo a memoria para ser procesado.

Las variables *A* y *nodo* deben ser pasadas por referencia, mientras que *NRR* por valor.

```
procedure PosicionarYLeerNodo(NRR: int16; var A: t_archivo_datos; var nodo: t_registro_nodo);
begin
  seek(A,NRR);
  read(A,nodo);
end;
```

(b) *claveEncontrada(): Indicar qué hace y la forma en que deben ser enviados los parámetros (valor o referencia). ¿Cómo se implementaría?*

claveEncontrada() es un procedimiento que compara la clave buscada con las claves del nodo, devuelve si fue encontrada y en qué posición, o a qué nodo hijo debe seguir la búsqueda si no está.

Las variables *pos* y *clave_encontrada* deben ser pasadas por referencia, mientras que *nodo* y *clave* por valor.

```
procedure claveEncontrada(nodo: t_registro_nodo; clave: int16; var pos: int16; var
clave_encontrada: boolean);
var
  i: int16;
begin
  i:=1;
  while ((i<=nodo.cantClaves) and (clave>nodo.claves[i].id)) do
    i:=i+1;
  if ((i<=nodo.cantClaves) and (clave=nodo.claves[i].id)) then
  begin
    clave_encontrada:=true;
    pos:=i;
  end
  else
  begin
    clave_encontrada:=false;
    pos:=i;
  end;
end;
```

(c) *¿Existe algún error en este código? En caso afirmativo, modificar lo que se considere necesario.*

Sí, existen errores en este código.

```
procedure buscar(NRR, clave: int16; var A: t_archivo_datos; var NRR_encontrado,
pos_encontrada: int16; var resultado: boolean);
var
  nodo: t_registro_nodo;
  pos: int16;
  clave_encontrada: boolean;
begin
  if (NRR=-1) then
    resultado:=false
  else
  begin
    posicionarYLeerNodo(NRR,A,nodo);
    claveEncontrada(nodo,clave,pos,clave_encontrada);
    if (clave_encontrada) then
    begin
      NRR_encontrado:=NRR;
      pos_encontrada:=pos;
      resultado:=true;
    end
    else
      buscar(nodo.hijos[pos],clave,A,NRR_encontrado,pos_encontrada,resultado)
  end;
end;
```

(d) *¿Qué cambios son necesarios en el procedimiento de búsqueda implementado sobre un árbol B para que funcione en un árbol B+?*

El cambio necesario en el procedimiento de búsqueda implementado sobre un árbol B para que funcione en un árbol B+ es que debería detectar si el nodo es hoja, ya que debería

terminar de buscar sólo al llegar a un nodo hoja porque sólo en estos es que están los punteros a los datos reales.

Ejercicio 5.

Definir los siguientes conceptos:

- *Overflow.*
- *Underflow.*
- *Redistribución.*
- *Fusión o concatenación.*

En los dos últimos casos, ¿cuándo se aplica cada uno?

Overflow: Se produce cuando se quiere agregar una clave a un nodo que ya tiene la cantidad máxima de claves permitidas.

Underflow: Se produce cuando se quiere eliminar una clave de un nodo que ya tiene la cantidad mínima de claves permitidas.

Redistribución: Se aplica cuando un nodo tiene *underflow* y es posible trasladar una llave de un nodo hermano adyacente a este nodo para que el *underflow* deje de ocurrir.

Fusión o concatenación: Se aplica cuando un nodo tiene *underflow* y un nodo adyacente hermano está al mínimo y no se puede redistribuir, por lo que se fusiona con un nodo adyacente disminuyendo la cantidad de nodos y, en algunos casos, la altura del árbol.

Ejercicio 6.

Suponer que se tiene un archivo que contiene información de los empleados de una empresa. De cada empleado, se mantiene la siguiente información: DNI, legajo, nombre completo y salario. Considerar que se mantiene, además, un índice, organizado como árbol B de orden 4, que provee acceso indizado a los empleados por su DNI. Graficar cómo queda el archivo de empleados (archivo de datos) y el archivo índice (árbol B) tras la inserción de los siguientes registros:

DNI	***Legajo***	***Nombre y apellido***	***Salario***
35.000.000	*100*	*Juan Pérez*	*$250.000*
40.000.000	*101*	*Lucio Redivo*	*$400.000*
32.000.000	*102*	*Nicolás Lapro*	*$720.000*
28.000.000	*103*	*Luis Scola*	*$2.000.000*
26.000.000	*104*	*Andrés Nocioni*	*$1.500.000*
37.000.000	*105*	*Facundo Campazzo*	*$1.200.000*
25.000.000	*106*	*Emanuel Ginobili*	*$5.000.000*
23.000.000	*107*	*Pepe Sánchez*	*$1.000.000*
21.000.000	*108*	*Alejandro Montecchia*	*$800.000*
36.000.000	*109*	*Marcos Delia*	*$300.000*
45.000.000	*110*	*Leandro Bolmaro*	*$600.000*

Notas:

- *Graficar los estados de ambos archivos (datos e índice) cuando ocurren cambios relevantes en el índice como la creación de un nuevo nodo.*
- *Además de graficar los archivos con sus respectivas estructuras internas, resulta útil que se grafique la vista del archivo índice como un árbol B.*

Inserción de claves 35.000.000, 40.000.000 y 32.000.000:

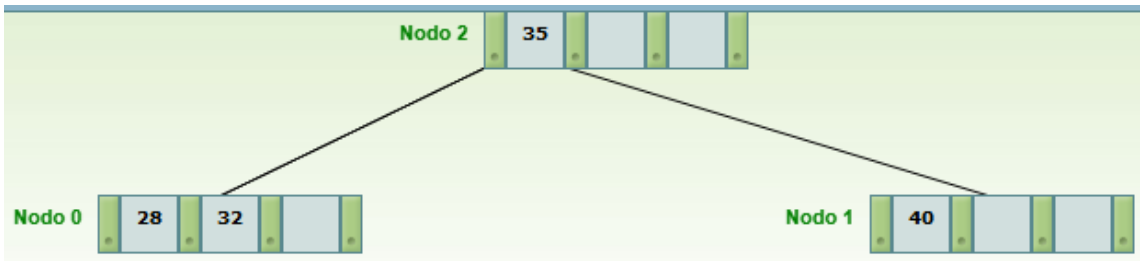
Archivo de datos: [35, 40, 32].

Archivo índice:

Inserción de clave 28.000.000:

Archivo de datos: [35, 40, 32, 28].

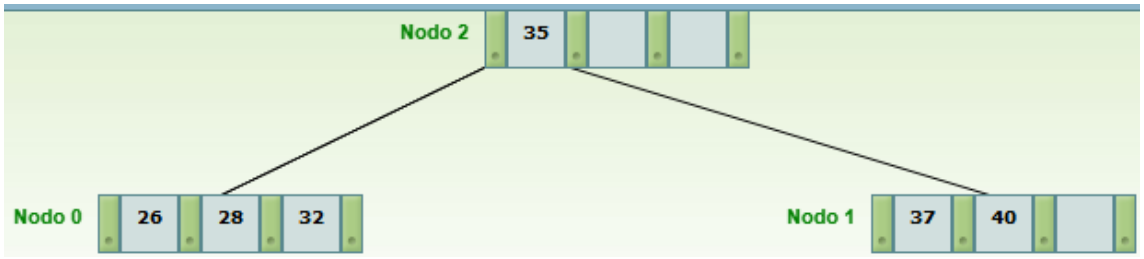
Archivo índice:



Inserción de claves 26.000.000 y 37.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37].

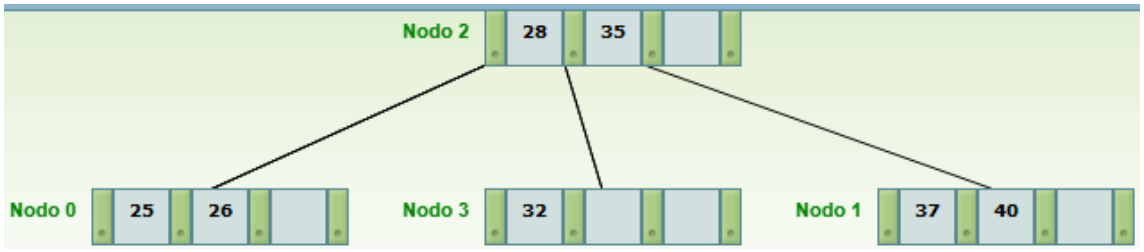
Archivo índice:



Inserción de clave 25.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37, 25].

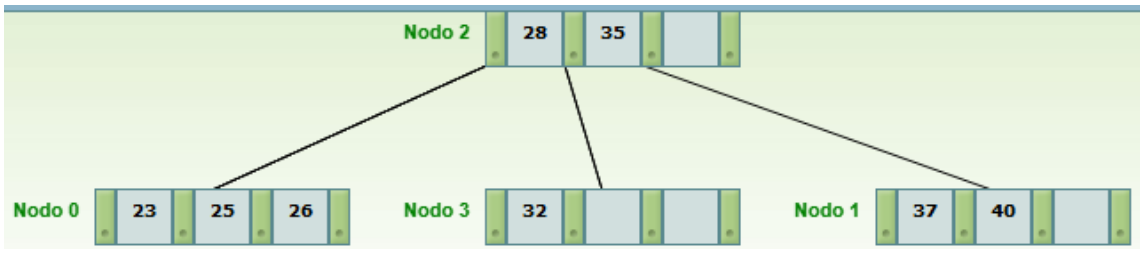
Archivo índice:



Inserción de clave 23.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37, 25, 23].

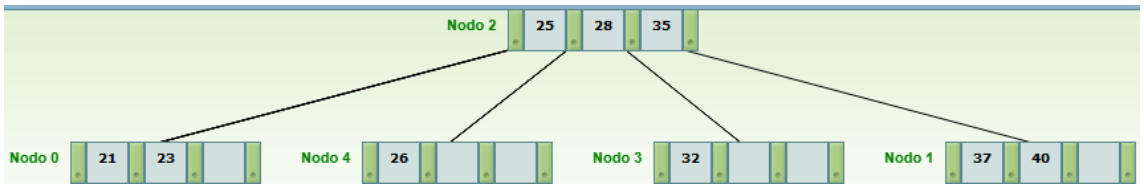
Archivo índice:



Inserción de clave 21.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37, 25, 23, 21].

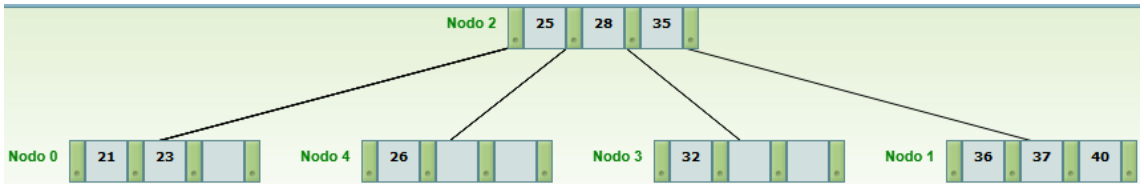
Archivo índice:



Inserción de clave 36.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37, 25, 23, 21, 36].

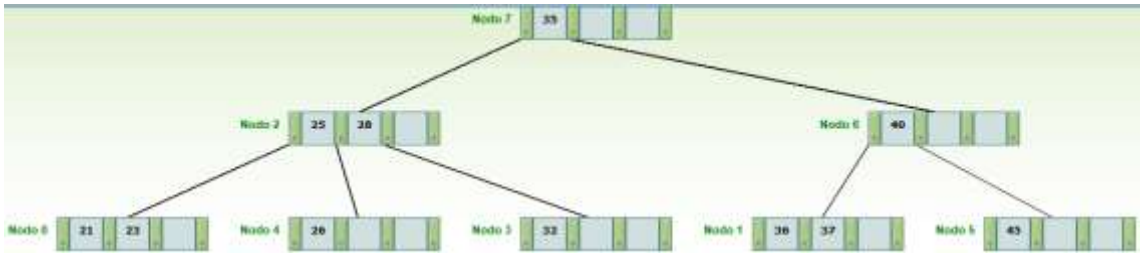
Archivo índice:



Inserción de clave 45.000.000:

Archivo de datos: [35, 40, 32, 28, 26, 37, 25, 23, 21, 36, 45].

Archivo índice:



PARTE II: Operaciones en Árboles B y B+.

Para los siguientes ejercicios, se debe:

- *Indicar los nodos leídos y escritos en cada operación.*
- *Todas las operaciones deben estar, claramente, justificadas, enunciando las mismas, indefectiblemente, tal cual se presenta en la materia.*
- *Los números de nodo deben asignarse en forma coherente con el crecimiento del archivo. La reutilización de nodos libres se debe efectuar con política LIFO (último en entrar, primero en salir).*
- *Para los siguientes ejercicios, sólo interesa graficar los estados del árbol B que representa el índice (no es necesario dibujar la estructura interna de los archivos como si se solicitó en el Ejercicio 6).*

Ejercicio 7.

Dado el siguiente árbol B de orden 5, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +320, -390, -400, -533. Justificar, detalladamente, cada operación indicando lecturas y escrituras en orden de ocurrencia. Para la resolución de underflow, se debe utilizar política izquierda. Graficar cada operación por separado.

2: 0 (220) 1 (390) 4 (455) 5 (541) 3
0: (10)(150) 1: (225)(241)(331)(360) 4: (400)(407) 5: (508)(533) 3: (690)(823)

Operación +320:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 320.
- Se produce *overflow* en el nodo hoja 2, división del mismo y promoción de la clave 390.
- Operaciones: L2, L1, E1, E6, E2, E7, E8.

8: 2 (390) 7
2: 0 (220) 1 (320) 6 **7:** 4 (455) 5 (541) 3
0: (10)(150) **1:** (225)(241) **6:** (331)(360) **4:** (400)(407) **5:** (508)(533) **3:** (690)(823)

Operación -390:

- Se reemplaza la clave 390 por la menor clave del subárbol derecho (400).
- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L8, L7, L4, E4, E8.

8: 2 (400) 7
2: 0 (220) 1 (320) 6 **7:** 4 (455) 5 (541) 3
0: (10)(150) **1:** (225)(241) **6:** (331)(360) **4:** (407) **5:** (508)(533) **3:** (690)(823)

Operación -400:

- Se reemplaza la clave 400 por la menor clave del subárbol derecho (407).
- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 5, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente izquierdo.
- Operaciones: L8, L7, L4, L5, E4, E5, E7, E8.

8: 2 (407) 7
2: 0 (220) 1 (320) 6 **7:** 4 (508) 5 (541) 3
0: (10)(150) **1:** (225)(241) **6:** (331)(360) **4:** (455) **5:** (533) **3:** (690)(823)

Operación -533:

- Se produce *underflow* en el nodo hoja 5. Hay que distribuir siguiendo la política izquierda, pero el hermano adyacente izquierdo ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente izquierdo, el nodo hoja 4.
- Operaciones: L8, L7, L5, L4, E4, E7.

8: 2 (407) 7
2: 0 (220) 1 (320) 6 **7:** 4 (541) 3
0: (10)(150) **1:** (225)(241) **6:** (331)(360) **4:** (455)(508) **3:** (690)(823)

Ejercicio 8.

Dado el siguiente árbol B de orden 4, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +5, +9, +80, +51, -50, -92. Política de resolución de underflows: derecha.

2: 0 (66) 1
0: (22)(32)(50) 1: (77)(79)(92)

Operación +5:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 32.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (32) 3 (66) 1
0: (5)(22) **3:** (50) **1:** (77)(79)(92)

Operación +9:

- Operaciones: L2, L0, E0.

2: 0 (32) 3 (66) 1
0: (5)(9)(22) **3:** (50) **1:** (77)(79)(92)

Operación +80:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 80.
- Operaciones: L2, L1, E1, E4, E2.

2: 0 (32) 3 (66) 1 (80) 4
0: (5)(9)(22) **3:** (50) **1:** (77)(79) **4:** (92)

Operación +51:

- Operaciones: L2, L3, E3.

2: 0 (32) 3 (66) 1 (80) 4
0: (5)(9)(22) **3:** (50)(51) **1:** (77)(79) **4:** (92)

Operación -50:

- No se produce *underflow* en el nodo hoja 3.
- Operaciones: L2, L3, E3.

2: 0 (32) 3 (66) 1 (80) 4
0: (5)(9)(22) **3:** (51) **1:** (77)(79) **4:** (92)

Operación -92:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 1, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente derecho.
- Operaciones: L2, L4, L1, E1, E4, E2.

2: 0 (32) 3 (66) 1 (79) 4
0: (5)(9)(22) **3:** (51) **1:** (77) **4:** (80)

Ejercicio 9.

Dado el siguiente árbol B de orden 6, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +15, +71, +3, +48, -56, -71. Política de resolución de underflows: derecha o izquierda.

0: (34)(56)(78)(100)(176)

Operación +15:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 78.
- Operaciones: L0, E0, E1, E2.

2: 0 (78) 1
0: (15)(34)(56) **1:** (100)(176)

Operación +71:

- Operaciones: L2, L0, E0.

2: 0 (78) 1
0: (15)(34)(56)(71) **1:** (100)(176)

Operación +3:

- Operaciones: L2, L0, E0.

2: 0 (78) 1
0: (3)(15)(34)(56)(71) **1:** (100)(176)

Operación +48:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 48.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (48) 3 (78) 1
0: (3)(15)(34) **3:** (56)(71) **1:** (100)(176)

Operación -56:

- Se produce *underflow* en el nodo hoja 3.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 0, ya que el hermano adyacente derecho, el nodo hoja 1, ya contiene la cantidad mínima de claves permitidas (2).
- Operaciones: L2, L3, L1, L0, E0, E3, E2.

2: 0 (34) 3 (78) 1

0: (3)(15) **3:** (48)(71) **1:** (100)(176)

Operación -71:

- Se produce *underflow* en el nodo hoja 3. Hay que redistribuir siguiendo la política derecha o izquierda, pero ambos nodos hoja (1 y 0) ya contienen la cantidad mínima de claves permitidas (2).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 1.
- Operaciones: L2, L3, L1, L0, E3, E2.

2: 0 (34) 3
0: (3)(15) **3:** (48)(78)(100)(176)

Ejercicio 10.

Dado el siguiente árbol B de orden 5, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +450, -485, -511, -614. Política de resolución de underflows: derecha.

2: 0 (315) 1 (485) 4 (547) 5 (639) 3
0: (148)(223) 1: (333)(390)(442)(454) 4: (508)(511) 5: (614)(633) 3: (789)(915)

Operación +450:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 442.
- Se produce *overflow* en el nodo hoja 2, división del mismo y promoción de la clave 485.
- Operaciones: L2, L1, E1, E6, E2, E7, E8.

8: 2 (485) 7
2: 0 (315) 1 (442) 6 **7:** 4 (547) 5 (639) 3
0: (148)(223) **1:** (333)(390) **6:** (450)(454) **4:** (508)(511) **5:** (614)(633) **3:** (789)(915)

Operación -485:

- Se reemplaza la clave 485 por la menor clave del subárbol derecho (508).
- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L8, L7, L4, E4, E8.

8: 2 (508) 7
2: 0 (315) 1 (442) 6 **7:** 4 (547) 5 (639) 3
0: (148)(223) **1:** (333)(390) **6:** (450)(454) **4:** (511) **5:** (614)(633) **3:** (789)(915)

Operación -511:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 5.
- Operaciones: L8, L7, L4, L5, E4, E5, E7.

8: 2 (508) 7
2: 0 (315) 1 (442) 6 **7:** 4 (614) 5 (639) 3
0: (148)(223) **1:** (333)(390) **6:** (450)(454) **4:** (547) **5:** (633) **3:** (789)(915)

Operación -614:

- Se reemplaza la clave 614 por la menor clave del subárbol derecho (633).
- Se produce *underflow* en el nodo hoja 5.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 3.
- Operaciones: L8, L7, L5, L3, E5, E3, E7.

8: 2 (508) 7
2: 0 (315) 1 (442) 6 **7:** 4 (633) 5 (789) 3
0: (148)(223) **1:** (333)(390) **6:** (450)(454) **4:** (547) **5:** (639) **3:** (915)

Ejercicio 11.

Dado un árbol B de orden 5 y con política izquierda, para cada operación dada:

- *Dibujar el árbol resultante.*
- *Explicar las decisiones tomadas.*
- *Escribir las lecturas y escrituras.*

Operaciones: -76, -400, +900, +12.

Nodo 2: 3 i 0(76)4(300)1(600)3
Nodo 0: 4 h(21)(45)(46)(70)
Nodo 4: 2 h(100)(200)
Nodo 1: 1 h(400)
Nodo 3: 2 h(700)(800)

2: 0 (76) 4 (300) 1 (600) 3
0: (21)(45)(46)(70) **4:** (100)(200) **1:** (400) **3:** (700)(800)

Operación -76:

- Se reemplaza la clave -76 por la menor clave del subárbol derecho (100).
- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L2, L4, E4, E2.

2: 0 (100) 4 (300) 1 (600) 3
0: (21)(45)(46)(70) **4:** (200) **1:** (400) **3:** (700)(800)

Operación -400:

- Se produce *underflow* en el nodo hoja 1. Hay que distribuir siguiendo la política izquierdo, pero el hermano adyacente izquierdo, el nodo hoja 4, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente izquierdo, el nodo hoja 4.
- Operaciones: L2, L1, L4, E4, E2.

2: 0 (100) 4 (600) 3
0: (21)(45)(46)(70) **4:** (200)(300) **3:** (700)(800)

Operación +900:

- Operaciones: L2, L3, E3.

2: 0 (100) 4 (600) 3
0: (21)(45)(46)(70) **4:** (200)(300) **3:** (700)(800)(900)

Operación +12:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 45.
- Operaciones: L2, L0, E0, E5, E2.

2: 0 (45) 5 (100) 4 (600) 3
0: (12)(21) **5:** (46)(70) **4:** (200)(300) **3:** (700)(800)(900)

Ejercicio 12.

Dadas las siguientes operaciones, mostrar la construcción, paso a paso, de un árbol B de orden 4: +50 , +70, +40, +15, +90, +120, +115, +45, +30, +100, +112, +77, -45, -40, -50, -90, -100. Política de resolución de underflows: izquierda o derecha.

Operación +50:

- Operaciones: L0, E0.

0: (50)

Operación +70:

- Operaciones: L0, E0.

0: (50)(70)

Operación +40:

- Operaciones: L0, E0.

0: (40)(50)(70)

Operación +15:

- Se produce *overflow* en el nodo 0, división del mismo y promoción de la clave 50.
- Operaciones: L0, E0, E1, E2.

2: 0 (50) 1
0: (15)(40) **1:** (70)

Operación +90:

- Operaciones: L2, L1, E1.

2: 0 (50) 1
0: (15)(40) **1:** (70)(90)

Operación +120:

- Operaciones: L2, L1, E1.

2: 0 (50) 1
0: (15)(40) **1:** (70)(90)(120)

Operación +115:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 115.
- Operaciones: L2, L1, E1, E3, E2.

2: 0 (50) 1 (115) 3
0: (15)(40) **1:** (70)(90) **3:** (120)

Operación +45:

- Operaciones: L2, L0, E0.

2: 0 (50) 1 (115) 3
0: (15)(40)(45) **1:** (70)(90) **3:** (120)

Operación +30:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 40.
- Operaciones: L2, L0, E0, E4, E2.

2: 0 (40) 4 (50) 1 (115) 3
0: (15)(30) **4:** (45) **1:** (70)(90) **3:** (120)

Operación +100:

- Operaciones: L2, L1, E1.

2: 0 (40) 4 (50) 1 (115) 3
0: (15)(30) **4:** (45) **1:** (70)(90)(100) **3:** (120)

Operación +112:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 100.
- Se produce *overflow* en el nodo 2, división del mismo y promoción de la clave 100.
- Operaciones: L2, L1, E1, E5, E2, E6, E7.

7: 2 (100) 6
2: 0 (40) 4 (50) 1 **6:** 5 (115) 3
0: (15)(30) **4:** (45) **1:** (70)(90) **5:** (112) **3:** (120)

Operación +77:

- Operaciones: L7, L2, L1, E1.

7: 2 (100) 6
2: 0 (40) 4 (50) 1 **6:** 5 (115) 3
0: (15)(30) **4:** (45) **1:** (70)(77)(90) **5:** (112) **3:** (120)

Operación -45:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 0.
- Operaciones: L7, L2, L4, L0, E0, E4, E2.

7: 2 (100) 6
2: 0 (30) 4 (50) 1 **6:** 5 (115) 3
0: (15) **4:** (40) **1:** (70)(77)(90) **5:** (112) **3:** (120)

Operación -40:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 1, ya que el hermano adyacente izquierdo, el nodo hoja 0, ya contiene la cantidad mínima de claves permitidas (1).
- Operaciones: L7, L2, L4, L0, L1, E4, E1, E2.

7: 2 (100) 6
2: 0 (30) 4 (70) 1 **6:** 5 (115) 3
0: (15) **4:** (50) **1:** (77)(90) **5:** (112) **3:** (120)

Operación -50:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 1, ya que el hermano adyacente izquierdo, el nodo hoja 0, ya contiene la cantidad mínima de claves permitidas (1).
- Operaciones: L7, L2, L4, L0, L1, E4, E1, E2.

7: 2 (100) 6
2: 0 (30) 4 (77) 1 **6:** 5 (115) 3
0: (15) **4:** (70) **1:** (90) **5:** (112) **3:** (120)

Operación -90:

- Se produce *underflow* en el nodo hoja 1. Hay que distribuir siguiendo la política izquierda o derecha, pero el hermano adyacente izquierdo, el nodo hoja 4, ya contiene la cantidad mínima de claves permitidas (1) y no tiene hermano adyacente derecho.
- Se fusiona con el hermano adyacente izquierdo, el nodo hoja 4.
- Operaciones: L7, L2, L1, L4, E4, E2.

7: 2 (100) 6
2: 0 (30) 4 **6:** 5 (115) 3
0: (15) **4:** (70)(77) **5:** (112) **3:** (120)

Operación -100:

- Se reemplaza la clave 100 por la menor clave del subárbol derecho (112).
- Se produce *underflow* en el nodo hoja 5. Hay que distribuir siguiendo la política izquierda o derecha, pero no tiene hermano adyacente izquierdo y el hermano adyacente derecho, el nodo hoja 3, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 3, ya que no tiene hermano adyacente izquierdo.
- Se produce *underflow* en el nodo 6. Hay que distribuir siguiendo la política izquierda o derecha, pero el hermano adyacente izquierdo, el nodo hoja 3, ya contiene la cantidad mínima de claves permitidas (1) y no tiene hermano adyacente derecho.
- Se fusiona con el hermano adyacente izquierdo, el nodo 2.
- Operaciones: L7, L6, L5, L3, L2, E5, E2.

2: 0 (30) 4 (112) 5
0: (15) **4:** (70)(77) **5:** (115)(120)

Ejercicio 13.

Dadas las siguientes operaciones, mostrar la construcción, paso a paso, de un árbol B de orden 5: +80, +50, +70, +120, +23, +52, +59, +65, +30, +40, +45, +31, +34, +38, +60, +63, +64, -23, -30, -31, -40, -45, -38. Política de resolución de underflows: izquierda.

Operación +80:

- Operaciones: L0, E0.

0: (80)

Operación +50:

- Operaciones: L0, E0.

0: (50)(80)

Operación +70:

- Operaciones: L0, E0.

0: (50)(70)(80)

Operación +120:

- Operaciones: L0, E0.

0: (50)(70)(80)(120)

Operación +23:

- Se produce *overflow* en el nodo 0, división del mismo y promoción de la clave 70.
- Operaciones: L0, E0, E1, E2.

2: 0 (70) 1
0: (23)(50) **1:** (80)(120)

Operación +52:

- Operaciones: L2, L0, E0.

2: 0 (70) 1
0: (23)(50)(52) **1:** (80)(120)

Operación +59:

- Operaciones: L2, L0, E0.

2: 0 (70) 1
0: (23)(50)(52)(59) **1:** (80)(120)

Operación +65:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 52.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (52) 3 (70) 1
0: (23)(50) **3:** (59)(65) **1:** (80)(120)

Operación +30:

- Operaciones: L2, L0, E0.

2: 0 (52) 3 (70) 1
0: (23)(30)(50) **3:** (59)(65) **1:** (80)(120)

Operación +40:

- Operaciones: L2, L0, E0.

2: 0 (52) 3 (70) 1
0: (23)(30)(40)(50) **3:** (59)(65) **1:** (80)(120)

Operación +45:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 40.
- Operaciones: L2, L0, E0, E4, E2.

2: 0 (40) 4 (52) 3 (70) 1
0: (23)(30) **4:** (45)(50) **3:** (59)(65) **1:** (80)(120)

Operación +31:

- Operaciones: L2, L0, E0.

2: 0 (40) 4 (52) 3 (70) 1
0: (23)(30)(31) **4:** (45)(50) **3:** (59)(65) **1:** (80)(120)

Operación +34:

- Operaciones: L2, L0, E0.

2: 0 (40) 4 (52) 3 (70) 1
0: (23)(30)(31)(34) **4:** (45)(50) **3:** (59)(65) **1:** (80)(120)

Operación +38:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 31.
- Operaciones: L2, L0, E0, E5, E2.

2: 0 (31) 5 (40) 4 (52) 3 (70) 1
0: (23)(30) **5:** (34)(38) **4:** (45)(50) **3:** (59)(65) **1:** (80)(120)

Operación +60:

- Operaciones: L2, L3, E3.

2: 0 (31) 5 (40) 4 (52) 3 (70) 1
0: (23)(30) **5:** (34)(38) **4:** (45)(50) **3:** (59)(60)(65) **1:** (80)(120)

Operación +63:

- Operaciones: L2, L3, E3.

2: 0 (31) 5 (40) 4 (52) 3 (70) 1
0: (23)(30) **5:** (34)(38) **4:** (45)(50) **3:** (59)(60)(63)(65) **1:** (80)(120)

Operación +64:

- Se produce *overflow* en el nodo hoja 3, división del mismo y promoción de la clave 63.
- Se produce *overflow* en el nodo 2, división del mismo y promoción de la clave 52.
- Operaciones: L2, L3, E3, E6, E2, E7, E8.

8: 2 (52) 7
2: 0 (31) 5 (40) 4 **7:** 3 (63) 6 (70) 1
0: (23)(30) **5:** (34)(38) **4:** (45)(50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -23:

- No se produce *underflow* en el nodo hoja 0.
- Operaciones: L8, L2, L0, E0.

8: 2 (52) 7
2: 0 (31) 5 (40) 4 **7:** 3 (63) 6 (70) 1
0: (30) **5:** (34)(38) **4:** (45)(50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -30:

- Se produce *underflow* en el nodo hoja 0.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 5, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente izquierdo.
- Operaciones: L8, L2, L0, E0, E5, E2.

8: 2 (52) 7
2: 0 (34) 5 (40) 4 **7:** 3 (63) 6 (70) 1
0: (31) **5:** (38) **4:** (45)(50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -31:

- Se produce *underflow* en el nodo hoja 0. Hay que distribuir siguiendo la política izquierda, pero no tiene hermano adyacente izquierdo y el hermano adyacente derecho, el nodo hoja 5, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 5, ya que no tiene hermano adyacente izquierdo.
- Operaciones: L8, L2, L0, L5, E0, E2.

8: 2 (52) 7
2: 0 (40) 4 **7:** 3 (63) 6 (70) 1
0: (34)(38) **4:** (45)(50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -40:

- Se reemplaza la clave 40 por la menor clave del subárbol derecho (45).
- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L8, L2, L4, E4, E2.

8: 2 (52) 7
2: 0 (45) 4 **7:** 3 (63) 6 (70) 1
0: (34)(38) **4:** (50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -45:

- Se reemplaza la clave 45 por la menor clave del subárbol derecho (50).
- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 0.
- Operaciones: L8, L2, L4, L0, E0, E4, E2.

8: 2 (52) 7
2: 0 (38) 4 **7:** 3 (63) 6 (70) 1
0: (34) **4:** (50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Operación -38:

- Se reemplaza la clave 38 por la menor clave del subárbol derecho (50).
- Se produce *underflow* en el nodo hoja 4. Hay que distribuir siguiendo la política izquierda, pero el hermano adyacente izquierdo, el nodo hoja 0, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente izquierdo, el nodo hoja 0.
- Se produce *underflow* en el nodo 2. Hay que distribuir siguiendo la política izquierda, pero no tiene hermano adyacente izquierdo.

- Se fusiona con el hermano adyacente derecho, el nodo 7, ya que no tiene hermano adyacente izquierdo.
- Operaciones: L8, L2, L4, L0, L7, E0, E2.

2: 0 (52) 3 (63) 6 (70) 1
0: (34)(50) **3:** (59)(60) **6:** (64)(65) **1:** (80)(120)

Ejercicio 14.

Dado el siguiente árbol B de orden 6, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +300, +577, -586, -570, -380, -460. Política de resolución de underflows: izquierda o derecha.

2: 0 (216) 1 (460) 4 (570) 5 (689) 3 (777) 6
0: (100)(159)(171) 1: (222)(256)(358)(380)(423) 4: (505)(522)
5: (586)(599)(615)(623)(680) 3: (703)(725) 6: (789)(915)(1000)

Operación +300:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 358.
- Se produce *overflow* en el nodo hoja 2, división del mismo y promoción de la clave 570.
- Operaciones: L2, L1, E1, E7, E2, E8, E9.

9: 2 (570) 8
2: 0 (216) 1 (358) 7 (460) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256)(300) **7:** (380)(423) **4:** (505)(522)
5: (586)(599)(615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Operación +577:

- Operaciones: L9, L8, L4, E4.

9: 2 (570) 8
2: 0 (216) 1 (358) 7 (460) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256)(300) **7:** (380)(423) **4:** (505)(522)(577)
5: (586)(599)(615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Operación -586:

- No se produce *underflow* en el nodo hoja 5.
- Operaciones: L9, L8, L5, E5.

9: 2 (570) 8
2: 0 (216) 1 (358) 7 (460) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256)(300) **7:** (380)(423) **4:** (505)(522)(577)
5: (599)(615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Operación -570:

- Se reemplaza la clave 570 por la menor clave del subárbol derecho (599).
- No se produce *underflow* en el nodo hoja 5.
- Operaciones: L9, L2, L4, E4, E9.

9: 2 (599) 8

2: 0 (216) 1 (358) 7 (460) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256)(300) **7:** (380)(423) **4:** (505)(522)(577)
5: (615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Operación -380:

- Se produce *underflow* en el nodo hoja 7.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 1.
- Operaciones: L9, L2, L7, L1, E1, E7, E2.

9: 2 (599) 8
2: 0 (216) 1 (300) 7 (460) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256) **7:** (358)(423) **4:** (505)(522)(577)
5: (615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Operación -460:

- Se reemplaza la clave 460 por la menor clave del subárbol derecho (505).
- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L9, L2, L4, E4, E2.

9: 2 (599) 8
2: 0 (216) 1 (300) 7 (505) 4 **8:** 5 (689) 3 (777) 6
0: (100)(159)(171) **1:** (222)(256) **7:** (358)(423) **4:** (522)(577)
5: (615)(623)(680) **3:** (703)(725) **6:** (789)(915)(1000)

Ejercicio 15.

Dada las siguientes operaciones, mostrar cómo se construye el árbol B de orden 4: +65, +89, +23, +45, +20, +96, +10, +55, -23, +110, +50, -10, +25, -50, -45, +120, +130, +70, +75, +73, +100, -120, -110. Política de resolución de underflows: derecha.

Operación +65:

- Operaciones: L0, E0.

0: (65)

Operación +89:

- Operaciones: L0, E0.

0: (65)(89)

Operación +23:

- Operaciones: L0, E0.

0: (23)(65)(89)

Operación +45:

- Se produce *overflow* en el nodo 0, división del mismo y promoción de la clave 65.
- Operaciones: L0, E0, E1, E2.

2: 0 (65) 1
0: (23)(45) **1:** (89)

Operación +20:

- Operaciones: L2, L0, E0.

2: 0 (65) 1
0: (20)(23)(45) **1:** (89)

Operación +96:

- Operaciones: L2, L1, E1.

2: 0 (65) 1
0: (20)(23)(45) **1:** (89)(96)

Operación +10:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de la clave 23.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (23) 3 (65) 1
0: (10)(20) **3:** (45) **1:** (89)(96)

Operación +55:

- Operaciones: L2, L3, E3.

2: 0 (23) 3 (65) 1
0: (10)(20) **3:** (45)(55) **1:** (89)(96)

Operación -23:

- Se reemplaza la clave 23 por la menor clave del subárbol derecho (45).
- No se produce *underflow* en el nodo hoja 3.
- Operaciones: L2, L3, E3, E2.

2: 0 (45) 3 (65) 1
0: (10)(20) **3:** (55) **1:** (89)(96)

Operación +110:

- Operaciones: L2, L1, E1.

2: 0 (45) 3 (65) 1
0: (10)(20) **3:** (55) **1:** (89)(96)(110)

Operación +50:

- Operaciones: L2, L3, E3.

2: 0 (45) 3 (65) 1
0: (10)(20) **3:** (50)(55) **1:** (89)(96)(110)

Operación -10:

- No se produce *underflow* en el nodo hoja 0.
- Operaciones: L2, L0, E0.

2: 0 (45) 3 (65) 1
0: (20) **3:** (50)(55) **1:** (89)(96)(110)

Operación +25:

- Operaciones: L2, L0, E0.

2: 0 (45) 3 (65) 1

0: (20)(25) **3:** (50)(55) **1:** (89)(96)(110)

Operación -50:

- No se produce *underflow* en el nodo hoja 3.
- Operaciones: L2, L3, E3.

2: 0 (45) 3 (65) 1
0: (20)(25) **3:** (55) **1:** (89)(96)(110)

Operación -45:

- Se reemplaza la clave 45 por la menor clave del subárbol derecho (55).
- Se produce *underflow* en el nodo hoja 3.
- Se distribuye con el hermano adyacente derecho, el nodo hoja 1.
- Operaciones: L2, L3, L1, E3, E1, E2.

2: 0 (55) 3 (89) 1
0: (20)(25) **3:** (65) **1:** (96)(110)

Operación +120:

- Operaciones: L2, L1, E1.

2: 0 (55) 3 (89) 1
0: (20)(25) **3:** (65) **1:** (96)(110)(120)

Operación +130:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 120.
- Operaciones: L2, L1, E1, E4, E2.

2: 0 (55) 3 (89) 1 (120) 4
0: (20)(25) **3:** (65) **1:** (96)(110) **4:** (130)

Operación +70:

- Operaciones: L2, L3, E3.

2: 0 (55) 3 (89) 1 (120) 4
0: (20)(25) **3:** (65)(70) **1:** (96)(110) **4:** (130)

Operación +75:

- Operaciones: L2, L3, E3.

2: 0 (55) 3 (89) 1 (120) 4
0: (20)(25) **3:** (65)(70)(75) **1:** (96)(110) **4:** (130)

Operación +73:

- Se produce *overflow* en el nodo hoja 3, división del mismo y promoción de la clave 73.
- Se produce *overflow* en el nodo 2, división del mismo y promoción de la clave 89.
- Operaciones: L2, L3, E3, E5, E2, E6, E7.

7: 2 (89) 6
2: 0 (55) 3 (73) 5 **6:** 1 (120) 4
0: (20)(25) **3:** (65)(70) **5:** (75) **1:** (96)(110) **4:** (130)

Operación +100:

- Operaciones: L7, L6, L1, E1.

7: 2 (89) 6
2: 0 (55) 3 (73) 5 **6:** 1 (120) 4
0: (20)(25) **3:** (65)(70) **5:** (75) **1:** (96)(100)(110) **4:** (130)

Operación -120:

- Se reemplaza la clave 120 por la menor clave del subárbol derecho (130).
- Se produce *underflow* en el nodo hoja 4.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 1, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente derecho.
- Operaciones: L7, L6, L4, L1, E1, E4, E6.

7: 2 (89) 6
2: 0 (55) 3 (73) 5 **6:** 1 (110) 4
0: (20)(25) **3:** (65)(70) **5:** (75) **1:** (96)(100) **4:** (130)

Operación -110:

- Se reemplaza la clave 110 por la menor clave del subárbol derecho (130).
- Se produce *underflow* en el nodo hoja 4.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 1, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente derecho.
- Operaciones: L7, L6, L4, L1, E1, E4, E6.

7: 2 (89) 6
2: 0 (55) 3 (73) 5 **6:** 1 (100) 4
0: (20)(25) **3:** (65)(70) **5:** (75) **1:** (96) **4:** (130)

Ejercicio 16.

Dado el siguiente árbol B+ de orden 4 y con política de resolución de underflows a derecha, realizar las siguientes operaciones: +80, -400. Indicando lecturas y escrituras en el orden de ocurrencia Además, se debe describir, detalladamente, lo que sucede en cada operación.

4: 0 (340) 1 (400) 2 (500) 3
0: (11)(50)(77) 1 1: (340)(350)(360) 2 2: (402)(410)(420) 3 3: (520)(530) -1

Operación +80:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 77.
- Se produce *overflow* en el nodo 4, división del mismo y promoción del separador 400.
- Operaciones: L4, L0, E0, E5, E4, E6, E7.

7: 4 (400) 6
4: 0 (77) 5 (340) 1 **6:** 2 (500) 3
0: (11)(50) **5:** (77)(80) 1 **1:** (340)(350)(360) 2 **2:** (402)(410)(420) 3 **3:** (520)(530) -1

Operación -400:

- La clave 400 no se encuentra en ningún nodo hoja.
- Operaciones: L7, L6, L2.

7: 4 (400) 6
4: 0 (77) 5 (340) 1 **6:** 2 (500) 3
0: (11)(50) **5:** (77)(80) 1 **1:** (340)(350)(360) 2 **2:** (402)(410)(420) 3 **3:** (520)(530) -1

Ejercicio 17.

Dado el siguiente árbol B+ de orden 4, mostrar cómo quedaría el mismo luego de realizar las siguientes operaciones: +120, +110, +52, +70, +15, -45, -52, +22, +19, -66, -22, -19, - 23, -89. Política de resolución de underflows: derecha.

2: 0 (66) 1
0: (23)(45) 1 1: (66)(67)(89) -1

Operación +120:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de una copia de la clave 89.
- Operaciones: L2, L1, E1, E3, E2.

2: 0 (66) 1 (89) 3
0: (23)(45) 1 **1:** (66)(67) 3 **3:** (89)(120) -1

Operación +110:

- Operaciones: L2, L3, E3.

2: 0 (66) 1 (89) 3
0: (23)(45) 1 **1:** (66)(67) 3 **3:** (89)(110)(120) -1

Operación +52:

- Operaciones: L2, L0, E0.

2: 0 (66) 1 (89) 3
0: (23)(45)(52) 1 **1:** (66)(67) 3 **3:** (89)(110)(120) -1

Operación +70:

- Operaciones: L2, L1, E1.

2: 0 (66) 1 (89) 3
0: (23)(45)(52) 1 **1:** (66)(67)(70) 3 **3:** (89)(110)(120) -1

Operación +15:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 45.
- Operaciones: L2, L0, E0, E4, E2.

2: 0 (45) 4 (66) 1 (89) 3
0: (15)(23) 4 **4:** (45)(52) 1 **1:** (66)(67)(70) 3 **3:** (89)(110)(120) -1

Operación -45:

- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L2, L4, E4.

2: 0 (45) 4 (66) 1 (89) 3
0: (15)(23) 4 **4:** (52) 1 **1:** (66)(67)(70) 3 **3:** (89)(110)(120) -1

Operación -52:

- Se produce *underflow* en el nodo hoja 4.
- Se redistribuye con el hermano adyacente derecho, el nodo hoja 1.
- Operaciones: L2, L4, L1, E4, E1, E2.

2: 0 (45) 4 (67) 1 (89) 3
0: (15)(23) 4 **4:** (66) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación +22:

- Operaciones: L2, L0, E0.

2: 0 (45) 4 (67) 1 (89) 3
0: (15)(22)(23) 4 **4:** (66) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación +19:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 22.
- Se produce *overflow* en el nodo 2, división del mismo y promoción de la clave 67.
- Operaciones: L2, L0, E0, E5, E2, E6, E7.

7: 2 (67) 6
2: 0 (22) 5 (45) 4 **6:** 1 (89) 3
0: (15)(19) 5 **5:** (22)(23) 4 **4:** (66) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación -66:

- Se produce *underflow* en el nodo hoja 4.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 5, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente derecho.
- Operaciones: L7, L2, L4, E5, E4, E2.

7: 2 (67) 6
2: 0 (22) 5 (23) 4 **6:** 1 (89) 3
0: (15)(19) 5 **5:** (22) 4 **4:** (23) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación -22:

- Se produce *underflow* en el nodo hoja 5. Hay que distribuir siguiendo la política derecha, pero el hermano adyacente derecho ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 5.
- Operaciones: L7, L2, L5, L4, E5, E2.

7: 2 (67) 6
2: 0 (22) 5 **6:** 1 (89) 3
0: (15)(19) 5 **5:** (23) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación -19:

- Operaciones: L7, L2, L0, E0.

7: 2 (67) 6
2: 0 (22) 5 **6:** 1 (89) 3
0: (15) 5 **5:** (23) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación -23:

- Se produce *underflow* en el nodo hoja 5. Hay que distribuir siguiendo la política derecha, pero no tiene hermano adyacente derecho y el hermano adyacente izquierdo, el nodo hoja 0, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente izquierdo, el nodo hoja 0, ya que no tiene hermano adyacente derecho.
- Se produce *underflow* en el nodo 2. Hay que distribuir siguiendo la política derecha, pero el hermano adyacente derecho, el nodo 6, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo 6.
- Operaciones: L7, L2, L5, L0, E0, E2.

2: 0 (67) 1 (89) 3
0: (15) 1 **1:** (67)(70) 3 **3:** (89)(110)(120) -1

Operación -89:

- Operaciones: L2, L3, E3.

2: 0 (67) 1 (89) 3
0: (15) 1 **1:** (67)(70) 3 **3:** (110)(120) -1

Ejercicio 18.

Dada las siguientes operaciones, mostrar la construcción, paso a paso, de un árbol B+ de orden 4: +67, +56, +96, +10, +28, +95, +16, +46, +23, +36, +120, +130, +60, +57, -96, -67, -95, -60, -120, -57, -56. Política de resolución de underflows: derecha o izquierda.

Operación +67:

- Operaciones: L0, E0.

0: (67) -1

Operación +56:

- Operaciones: L0, E0.

0: (56)(67) -1

Operación +96:

- Operaciones: L0, E0.

0: (56)(67)(96) -1

Operación +10:

- Se produce *overflow* en el nodo 0, división del mismo y promoción de una copia de la clave 67.
- Operaciones: L0, E0, E1, E2.

2: 0 (67) 1
0: (10)(56) 1 **1:** (67)(96) -1

Operación +28:

- Operaciones: L2, L0, E0.

2: 0 (67) 1
0: (10)(28)(56) 1 **1:** (67)(96) -1

Operación +95:

- Operaciones: L2, L1, E1.

2: 0 (67) 1
0: (10)(28)(56) 1 **1:** (67)(95)(96) -1

Operación +16:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 28.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (28) 3 (67) 1
0: (10)(16) 3 **3:** (28)(56) 1 **1:** (67)(95)(96) -1

Operación +46:

- Operaciones: L2, L3, E3.

2: 0 (28) 3 (67) 1
0: (10)(16) 3 **3:** (28)(46)(56) 1 **1:** (67)(95)(96) -1

Operación +23:

- Operaciones: L2, L0, E0.

2: 0 (28) 3 (67) 1
0: (10)(16)(23) 3 **3:** (28)(46)(56) 1 **1:** (67)(95)(96) -1

Operación +36:

- Se produce *overflow* en el nodo hoja 3, división del mismo y promoción de una copia de la clave 46.
- Operaciones: L2, L3, E3, E4, E2.

2: 0 (28) 3 (46) 4 (67) 1
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 1 **1:** (67)(95)(96) -1

Operación +120:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de una copia de la clave 96.
- Se produce *overflow* en el nodo 2, división del mismo y promoción del separador 67.
- Operaciones: L2, L1, E1, E5, E2, E6, E7.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 1 **1:** (67)(95) 5 **5:** (96)(120) -1

Operación +130:

- Operaciones: L7, L6, L5, E5.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 1 **1:** (67)(95) 5 **5:** (96)(120)(130) -1

Operación +60:

- Operaciones: L7, L2, L4, E4.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56)(60) 1 **1:** (67)(95) 5 **5:** (96)(120)(130) -1

Operación +57:

- Se produce *overflow* en el nodo 4, división del mismo y promoción de una copia de la clave 57.
- Operaciones: L7, L2, L4, E4, E8, E2.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 (57) 8 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57)(60) 1 **1:** (67)(95) 5 **5:** (96)(120)(130) -1

Operación -96:

- No se produce *underflow* en el nodo hoja 5.
- Operaciones: L7, L6, L5, E5.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 (57) 8 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57)(60) 1 **1:** (67)(95) 5 **5:** (120)(130) -1

Operación -67:

- No se produce *underflow* en el nodo hoja 1.
- Operaciones: L7, L6, L1, E1.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 (57) 8 **6:** 1 (96) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57)(60) 1 **1:** (95) 5 **5:** (120)(130) -1

Operación -95:

- Se produce *underflow* en el nodo hoja 1.
- Se distribuye con el hermano adyacente derecho, el nodo hoja 5.
- Operaciones: L7, L6, L1, E1.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 (57) 8 **6:** 1 (130) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57)(60) 1 **1:** (120) 5 **5:** (130) -1

Operación -60:

- No se produce *underflow* en el nodo hoja 8.
- Operaciones: L7, L2, L8, E8.

7: 2 (67) 6
2: 0 (28) 3 (46) 4 (57) 8 **6:** 1 (130) 5
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57) 1 **1:** (120) 5 **5:** (130) -1

Operación -120:

- Se produce *underflow* en el nodo hoja 1. Hay que distribuir siguiendo la política derecha o izquierda, pero el hermano adyacente derecho, el nodo hoja 5, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 5.
- Se produce *underflow* en el nodo 6.
- Se distribuye con el hermano adyacente izquierdo, el nodo 2, ya que no tiene hermano adyacente derecho.
- Operaciones: L7, L6, L1, L5, E1, E2, E6, E7.

7: 2 (57) 6
2: 0 (28) 3 (46) 4 **6:** 8 (67) 1
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (57) 1 **1:** (130) -1

Operación -57:

- Se produce *underflow* en el nodo hoja 8. Hay que distribuir siguiendo la política derecha o izquierda, pero el hermano adyacente derecho, el nodo hoja 1, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona el hermano adyacente derecho, el nodo hoja 1.
- Se produce *underflow* en el nodo 6.
- Se distribuye con el hermano adyacente izquierdo, el nodo 2, ya que no tiene hermano adyacente derecho.
- Operaciones: L7, L6, L8, L1, E8, E2, E6, E7.

7: 2 (46) 6
2: 0 (28) 3 **6:** 4 (57) 8
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46)(56) 8 **8:** (130) -1

Operación -56:

- No se produce *underflow* en el nodo hoja 4.
- Operaciones: L7, L6, L4, E4.

7: 2 (46) 6
2: 0 (28) 3 **6:** 4 (57) 8
0: (10)(16)(23) 3 **3:** (28)(36) 4 **4:** (46) 8 **8:** (130) -1

Ejercicio 19.

Dada las siguientes operaciones, mostrar la construcción, paso a paso, de un árbol B+ de orden 6: +52, +23, +10, +99, +63, +74, +19, +85, +14, +73, +5, +7, +41, +100, +130, +44, -63, -73, +15, +16, -74, -52. Política de resolución de underflows: izquierda.

Operación +52:

- Operaciones: L0, E0.

0: (52) -1

Operación +23:

- Operaciones: L0, E0.

0: (23)(52) -1

Operación +10:

- Operaciones: L0, E0.

0: (10)(23)(52) -1

Operación +99:

- Operaciones: L0, E0.

0: (10)(23)(52)(99) -1

Operación +63:

- Operaciones: L0, E0.

0: (10)(23)(52)(63)(99) -1

Operación +74:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 63.
- Operaciones: L0, E0, E1, E2.

2: 0 (63) 1
0: (10)(23)(52) 1 **1:** (63)(74)(99) -1

Operación +19:

- Operaciones: L2, L0, E0.

2: 0 (63) 1
0: (10)(19)(23)(52) **1:** (63)(74)(99) -1

Operación +85:

- Operaciones: L2, L1, E1.

2: 0 (63) 1
0: (10)(19)(23)(52) 1 **1:** (63)(74)(85)(99) -1

Operación +14:

- Operaciones: L2, L0, E0.

2: 0 (63) 1
0: (10)(14)(19)(23)(52) 1 **1:** (63)(74)(85)(99) -1

Operación +73:

- Operaciones: L2, L1, E1.

2: 0 (63) 1
0: (10)(14)(19)(23)(52) 1 **1:** (63)(73)(74)(85)(99) -1

Operación +5:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 19.
- Operaciones: L2, L0, E0, E3, E2.

2: 0 (19) 3 (63) 1
0: (5)(10)(14) 3 **3:** (19)(23)(52) 1 **1:** (63)(73)(74)(85)(99) -1

Operación +7:

- Operaciones: L2, L0, E0.

2: 0 (19) 3 (63) 1
0: (5)(7)(10)(14) 3 **3:** (19)(23)(52) 1 **1:** (63)(73)(74)(85)(99) -1

Operación +41:

- Operaciones: L2, L3, E3.

2: 0 (19) 3 (63) 1
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(52) 1 **1:** (63)(73)(74)(85)(99) -1

Operación +100:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de una copia de la clave 85.
- Operaciones: L2, L1, E1, E4, E2.

2: 0 (19) 3 (63) 1 (85) 4
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(52) 1 **1:** (63)(73)(74) 4 **4:** (85)(99)(100) -1

Operación +130:

- Operaciones: L2, L4, E4.

2: 0 (19) 3 (63) 1 (85) 4
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(52) 1 **1:** (63)(73)(74) 4 **4:** (85)(99)(100)(130) -1

Operación +44:

- Operaciones: L2, L3, E3.

2: 0 (19) 3 (63) 1 (85) 4
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(44)(52) 1 **1:** (63)(73)(74) 4 **4:** (85)(99)(100)(130) -1

Operación -63:

- No se produce *underflow* en el nodo hoja 1.
- Operaciones: L2, L1, E1.

2: 0 (19) 3 (63) 1 (85) 4
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(44)(52) 1 **1:** (73)(74) 4 **4:** (85)(99)(100)(130) -1

Operación -73:

- Se produce *underflow* en el nodo hoja 1.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 3.
- Operaciones: L2, L1, E3, E1, E2.

2: 0 (19) 3 (52) 1 (85) 4
0: (5)(7)(10)(14) 3 **3:** (19)(23)(41)(44) 1 **1:** (52)(74) 4 **4:** (85)(99)(100)(130) -1

Operación +15:

- Operaciones: L2, L0, E0.

2: 0 (19) 3 (52) 1 (85) 4
0: (5)(7)(10)(14)(15) 3 **3:** (19)(23)(41)(44) 1 **1:** (52)(74) 4 **4:** (85)(99)(100)(130) -1

Operación +16:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 14.

- Operaciones: L2, L0, E0, E5, E2.

2: 0 (14) 5 (19) 3 (52) 1 (85) 4
0: (5)(7)(10) 5 **5:** (14)(15)(16) 3 **3:** (19)(23)(41)(44) 1 **1:** (52)(74) 4 **4:** (85)(99)(100)(130) -1

Operación -74:

- Se produce *underflow* en el nodo hoja 1.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 3.
- Operaciones: L2, L1, L3, E3, E1, E2.

2: 0 (14) 5 (19) 3 (44) 1 (85) 4
0: (5)(7)(10) 5 **5:** (14)(15)(16) 3 **3:** (19)(23)(41) 1 **1:** (44)(52) 4 **4:** (85)(99)(100)(130) -1

Operación -52:

- Se produce *underflow* en el nodo hoja 1.
- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 3.
- Operaciones: L2, L1, L3, E3, E1, E2.

2: 0 (14) 5 (19) 3 (41) 1 (85) 4
0: (5)(7)(10) 5 **5:** (14)(15)(16) 3 **3:** (19)(23) 1 **1:** (41)(44) 4 **4:** (85)(99)(100)(130) -1

Ejercicio 20.

Dado un árbol B+ de orden 4 y con política izquierda o derecha, para cada operación dada:

- *Dibujar el árbol resultante.*
- *Explicar, brevemente, las decisiones tomadas.*
- *Escribir las lecturas y escrituras.*

Operaciones: +4, +44, -94, -104.

Nodo 7: 1 i 2(69)6
Nodo 2: 2 i 0(30)1(51)3
Nodo 6: 1 i 4(94)5
Nodo 0: 3 h(5)(10)(20)->1
Nodo 1: 2 h(30)(40)->3
Nodo 3: 2 h(51)(60)->4
Nodo 4: 2 h(69)(80)->5
Nodo 5: 1 h(104)->-1

7: 2 (69) 6
2: 0 (30) 1 (51) 3 **6:** 4 (94) 5
0: (5)(10)(20) 1 **1:** (30)(40) 3 **3:** (51)(60) 4 **4:** (69)(80) 5 **5:** (104) -1

Operación +4:

- Se produce *overflow* en el nodo hoja 0, división del mismo y promoción de una copia de la clave 10.
- Operaciones: L7, L2, L0, E0, E8, E2.

7: 2 (69) 6
2: 0 (10) 8 (30) 1 (51) 3 **6:** 4 (94) 5
0: (4)(5) 8 **8:** (10)(20) 1 **1:** (30)(40) 3 **3:** (51)(60) 4 **4:** (69)(80) 5 **5:** (104) -1

Operación +44:

- Operaciones: L7, L2, L1, E1.

7: 2 (69) 6
2: 0 (10) 8 (30) 1 (51) 3 **6:** 4 (94) 5
0: (4)(5) 8 **8:** (10)(20) 1 **1:** (30)(40)(44) 3 **3:** (51)(60) 4 **4:** (69)(80) 5 **5:** (104) -1

Operación -94:

- La clave 94 no se encuentra en ningún nodo hoja.
- Operaciones: L7, L6, L5.

7: 2 (69) 6
2: 0 (10) 8 (30) 1 (51) 3 **6:** 4 (94) 5

0: (4)(5) 8 **8:** (10)(20) 1 **1:** (30)(40)(44) 3 **3:** (51)(60) 4 **4:** (69)(80) 5 **5:** (104) -1

Operación -104:

- Se produce *underflow* en el nodo hoja 5.
- Se redistribuye con el hermano adyacente izquierdo, el nodo hoja 4.
- Operaciones: L7, L6, L5, L4, E4, E5, E6.

7: 2 (69) 6
2: 0 (10) 8 (30) 1 (51) 3 **6:** 4 (80) 5
0: (4)(5) 8 **8:** (10)(20) 1 **1:** (30)(40)(44) 3 **3:** (51)(60) 4 **4:** (69) 5 **5:** (80) -1

Ejercicio 21.

Dado el árbol B+ que se detalla más abajo, con orden 6, es decir, capacidad de 5 claves como máximo, mostrar los estados sucesivos al realizar la siguiente secuencia de operaciones: +159, -5, -190. Además, indicar nodos leídos y escritos en el orden de ocurrencia. Política de resolución underflow: derecha.

Nodo 2: 5, i, 0(10)1(60)3(115)4(145)5(179)6
Nodo 0: 2, h, (1)(5) -> 1
Nodo 1: 2, h, (34)(44) -> 3
Nodo 3: 2, h, (60)(113) -> 4
Nodo 4: 4, h, (120)(125)(131)(139) -> 5
Nodo 5: 5, h, (145)(153)(158)(160)(177) -> 6
Nodo 6: 2, h, (179)(190) -> -1

2: 0 (10) 1 (60) 3 (115) 4 (145) 5 (179) 6
0: (1)(5) 1 **1:** (34)(44) 3 **3:** (60)(113) 4 **4:** (120)(125)(131)(139) 5 **5:** (145)(153)(158)(160)(177) 6 **6:** (179)(190) -1

Operación +159:

- Se produce *overflow* en el nodo hoja 5, división del mismo y promoción de una copia de la clave 159.
- Se produce *overflow* en el nodo 2, división del mismo y promoción del separador 145.
- Operaciones: L2, L5, E5, E7, E2, E8, E9.

9: 2 (145) 7
2: 0 (10) 1 (60) 3 (115) 4 **8:** 5 (159) 7 (179) 6
0: (1)(5) 1 **1:** (34)(44) 3 **3:** (60)(113) 4 **4:** (120)(125)(131)(139) 5 **5:** (145)(153)(158) 7 **7:** (159)(160)(177) 6 **6:** (179)(190) -1

Operación -5:

- Se produce *underflow* en el nodo hoja 0. Hay que distribuir siguiendo la política derecha, pero el hermano adyacente derecho, el nodo hoja 1, ya contiene la cantidad mínima de claves permitidas (2).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 1.
- Operaciones: L9, L2, L0, L1, E0, E1, E2.

9: 2 (145) 7
2: 0 (60) 3 (115) 4 **8:** 5 (159) 7 (179) 6
0: (1)(34)(44) 3 **3:** (60)(113) 4 **4:** (120)(125)(131)(139) 5 **5:** (145)(153)(158) 7 **7:** (159)(160)(177) 6 **6:** (179)(190) -1

Operación -190:

- Se produce *underflow* en el nodo hoja 6.

- Se distribuye con el hermano adyacente izquierdo, el nodo hoja 7, ya que se trata de un nodo hoja de un extremo y no tiene hermano adyacente derecho.
- Operaciones: L9, L8, L6, L7, E7, E6, E8.

9: 2 (145) 7
2: 0 (60) 3 (115) 4 **8:** 5 (159) 7 (177) 6
0: (1)(34)(44) 3 **3:** (60)(113) 4 **4:** (120)(125)(131)(139) 5 **5:** (145)(153)(158) 7 **7:** (159)(160) 6 **6:** (177)(179) -1

Ejercicio 22.

Dado un árbol B de orden 5 y con política izquierda o derecha, para cada operación dada:

- *Dibujar el árbol resultante.*
- *Explicar, detalladamente, las decisiones tomadas.*
- *Escribir las lecturas y escrituras.*

Operaciones: +165, +260, +800, -110.

Árbol:
Nodo 8: 1 i 2 (150) 7
Nodo 2: 1 i 0 (120) 3
Nodo 7: 2 i 4 (210)6(300)1
Nodo 0: 2 h (30)(110)
Nodo 3: 1 h (130)
Nodo 4: 4 h (160)(170)(180)(200)
Nodo 6: 4 h (220)(230)(240)(250)
Nodo 1: 4 h (400)(500)(600)(700)

8: 2 (150) 7
2: 0 (120) 3 **7:** 4 (210) 6 (300) 1
0: (30)(110) **3:** (130) **4:** (160)(170)(180)(200) **6:** (220)(230)(240)(250) **1:** (400)(500)(600)(700)

Operación +165:

- Se produce *overflow* en el nodo hoja 4, división del mismo y promoción de la clave 170.
- Operaciones: L8, L7, L4, E4, E9, E7.

8: 2 (150) 7
2: 0 (120) 3 **7:** 4 (170) 9 (210) 6 (300) 1
0: (30)(110) **3:** (130) **4:** (160)(165) **9:** (180)(200) **6:** (220)(230)(240)(250) **1:** (400)(500)(600)(700)

Operación +260:

- Se produce *overflow* en el nodo hoja 6, división del mismo y promoción de la clave 240.
- Operaciones: L8, L7, L6, E6, E10, E7.

8: 2 (150) 7
2: 0 (120) 3 **7:** 4 (170) 9 (210) 6 (240) 10 (300) 1
0: (30)(110) **3:** (130) **4:** (160)(165) **9:** (180)(200) **6:** (220)(230) **10:** (250)(260) **1:** (400)(500)(600)(700)

Operación +800:

- Se produce *overflow* en el nodo hoja 1, división del mismo y promoción de la clave 600.
- Se produce *overflow* en el nodo 7, división del mismo y promoción de la clave 240.
- Operaciones: L8, L7, L1, E1, E11, E7, E12, E8.

8: 2 (150) 7 (240) 12
2: 0 (120) 3 **7:** 4 (170) 9 (210) 6 **12:** 10 (300) 1 (600) 11
0: (30)(110) **3:** (130) **4:** (160)(165) **9:** (180)(200) **6:** (220)(230) **10:** (250)(260) **1:** (400)(500) **11:** (700)(800)

Operación -110:

- No se produce *underflow* en el nodo hoja 0.
- Operaciones: L8, L2, L0, E0.

8: 2 (150) 7 (240) 12
2: 0 (120) 3 **7:** 4 (170) 9 (210) 6 **12:** 10 (300) 1 (600) 11
0: (30) **3:** (130) **4:** (160)(165) **9:** (180)(200) **6:** (220)(230) **10:** (250)(260) **1:** (400)(500) **11:** (700)(800)

Ejercicio 23.

Dado un árbol B+ de orden 5 y con política izquierda o derecha, para cada operación dada:

- *Dibujar el árbol resultante.*
- *Explicar, detalladamente, las decisiones tomadas.*
- *Escribir las lecturas y escrituras.*

Operaciones: +250, -300, -40.

Árbol:
Nodo 8: 1 i 2(70)7
Nodo 2: 1 i 0(50)4
Nodo 7: 4 i 5(90)6(120)3(210)9(300)1
Nodo 0: 1 h(40)->4
Nodo 4: 1 h(50)->5
Nodo 5: 2 h(70)(80)->6
Nodo 6: 2 h(90)(100)->3
Nodo 3: 2 h(120)(200)->9
Nodo 9: 4 h(210)(220)(230)(240)->1
Nodo 1: 2 h(400)(500)->-1

8: 2 (70) 7
2: 0 (50) 4 **7:** 5 (90) 6 (120) 3 (210) 9 (300) 1
0: (40) 4 **4:** (50) 5 **5:** (70)(80) 6 **6:** (90)(100) 3 **3:** (120)(200) 9 **9:** (210)(220)(230)(240) 1 **1:** (400)(500) -1

Operación +250:

- Se produce *overflow* en el nodo hoja 9, división del mismo y promoción de una copia de la clave 230.
- Se produce *overflow* en el nodo 7, división del mismo y promoción del separador 210.
- Operaciones: L8, L7, L9, E9, E10, E7, E11, E8.

8: 2 (70) 7 (210) 11
2: 0 (50) 4 **7:** 5 (90) 6 (120) 3 **11:** 9 (230) 10 (300) 1
0: (40) 4 **4:** (50) 5 **5:** (70)(80) 6 **6:** (90)(100) 3 **3:** (120)(200) 9 **9:** (210)(220) 10 **10:** (230)(240)(250) 1 **1:** (400)(500) -1

Operación -300:

- La clave 300 no se encuentra en ningún nodo hoja.
- Operaciones: L8, L11, L1.

8: 2 (70) 7 (210) 11
2: 0 (50) 4 **7:** 5 (90) 6 (120) 3 **11:** 9 (230) 10 (300) 1

0: (40) 4 **4:** (50) 5 **5:** (70)(80) 6 **6:** (90)(100) 3 **3:** (120)(200) 9 **9:** (210)(220) 10 **10:** (230)(240)(250) 1 **1:** (400)(500) -1

Operación -40:

- Se produce *underflow* en el nodo hoja 0. Hay que distribuir siguiendo la política izquierda o derecha, pero no tiene hermano adyacente izquierdo y el hermano adyacente derecho, el nodo hoja 4, ya contiene la cantidad mínima de claves permitidas (1).
- Se fusiona con el hermano adyacente derecho, el nodo hoja 4.
- Se produce *underflow* en el nodo 2.
- Se distribuye con el hermano adyacente derecho, el nodo 7, ya que no tiene hermano adyacente izquierdo.
- Operaciones: L8, L2, L0, L4, L7, E0, E2, E7, E8.

8: 2 (90) 7 (210) 11
2: 0 (70) 5 **7:** 6 (120) 3 **11:** 9 (230) 10 (300) 1
0: (50) 5 **5:** (70)(80) 6 **6:** (90)(100) 3 **3:** (120)(200) 9 **9:** (210)(220) 10 **10:** (230)(240)(250) 1 **1:** (400)(500) -1

Juan Menduiña

Trabajo Práctico N° 5:
Hashing.

PARTE I: Preguntas Conceptuales.

Ejercicio 1.

Definir el concepto de hashing (o dispersión). ¿Cómo se relaciona este concepto con archivos?

Hashing (o dispersión) es una técnica utilizada para transformar una clave (como una palabra, un número o un conjunto de datos) en una dirección o índice dentro de una estructura de datos, generalmente una tabla o un archivo. Esta transformación se realiza mediante una función de dispersión (función *hash*), que toma la clave como entrada y produce un número entero, la dirección *hash* o el código *hash*, que se utiliza como índice para acceder, rápidamente, a los datos.

Hashing (o dispersión) es un método de asignación de claves a posiciones en una tabla o archivo mediante una función de dispersión (función *hash*), con el objetivo de facilitar el almacenamiento y recuperación eficiente de información.

- Técnica para generar una direción base única para una llave dada. La dispersión se usa cuando se requiere acceso rápido a una llave.
- Técnica que convierte la llave del registro en un número aleatorio, el que sirve, después, para determinar dónde se almacena el registro.
- Técnica de almacenamiento y recuperación que usa una función de *hash* para mapear registros en dirección de almacenamiento.

Ejercicio 2.

Explicar el concepto de función de dispersión. Enumerar, al menos, tres funciones de dispersión y explicar, brevemente, cómo funciona cada una.

Una función de dispersión (función *hash*) es un procedimiento que toma una clave (como una palabra, un número o un conjunto de datos) como entrada y la convierte en un número entero, la dirección *hash* o el código *hash*, que se utiliza como índice en una tabla o en un archivo para acceder, rápidamente, a los datos.

El objetivo principal es asignar claves a posiciones de manera uniforme para facilitar el acceso rápido a los datos, minimizando las colisiones (casos en los que diferentes claves generan el mismo índice).

Tres funciones de dispersión son:

- Método de la división: Toma la clave *k* (por ejemplo, un número entero) y la divide por un número *m* (preferentemente primo), y se usa el resto como la dirección *hash*.
  hash(k)= k mod m.
- Método de la multiplicación: Toma la clave *k* (por ejemplo, un númeor entero), la multiplica por un número fraccionario (A= 0,618), toma la parte decimal, y la multiplica por m para obtener un índice.
  hash(k)= floor(m * ((k * A) mod 1)).
- Método de la suma de caracteres (para cadenas): Suma los valores ASCII (o numéricos) de todos los caracteres de la clave y, luego, aplica una operación como mod m.

Ejercicio 3.

Explicar los conceptos de sinónimo, colisión y desborde (overflow). ¿Qué condición es necesaria en el archivo directo para que pueda ocurrir una colisión y no un desborde?

Sinónimo: Es una clave diferente que, al ser procesada por la función de dispersión, produce la misma dirección *hash* que otra clave. Es decir, dos claves distintas generan el mismo índice.

Colisión: Situación en la que un registro es asignado a una dirección que está utilizada por otro registro. Se produce cuando dos o más claves se asignan a la misma posición en la tabla o en el archivo.

Desborde (*overflow*): Situación en la que un registro es asignado a una dirección que está utilizada por otro registro y no queda espacio para este nuevo. Se produce cuando no hay espacio disponible en la posición calculada por la función de dispersión ni en las posiciones alternativas previstas para manejar colisiones.

La condición que es necesaria en el archivo directo para que pueda ocurrir una colisión y no un desborde es que exista, al menos, una posición disponible donde se pueda reubicar el sinónimo.

Ejercicio 4.

¿Qué alternativas existen para reducir el número de colisiones (y, por ende, de desbordes) en un archivo organizado mediante la técnica de hashing?

Las alternativas que existen para reducir el número de colisiones (y, por ende, de desbordes) en un archivo organizado mediante la técnica de *hashing* son:

- Algoritmos de dispersión sin colisiones o que éstas nunca produzcan *overflow* (perfectos e imposibles de conseguir).
- Buscar métodos que distribuyan los registros de la forma más aleatoria posible.
- Distribuir pocos registros en muchas direcciones.
- Colocar más de un registro por dirección.

Ejercicio 5.

Explicar, brevemente, qué es la densidad de empaquetamiento. ¿Cuáles son las consecuencias de tener una menor densidad de empaquetamiento en un archivo directo?

La densidad de empaquetamiento es la proporción de espacio del archivo asignado que, en realidad, almacena registros. Las consecuencias de tener una menor densidad de empaquetamiento en un archivo directo es menos *overflow* y más desperdicio de espacio.

Ejercicio 6.

Explicar, brevemente, cómo funcionan las siguientes técnicas de resolución de desbordes que se pueden utilizar en hashing estático: saturación progresiva, saturación encadenada, saturación progresiva encadenada con área de desborde separada y dispersión doble.

Saturación progresiva: Cuando se completa el nodo, se busca el próximo hasta encontrar uno libre.

Saturación encadenada: Es similar a la saturación progresiva, pero los registros de saturación se encadenan y no ocupan, necesariamente, posiciones contiguas.

Saturación progresiva encadenada con área de desborde separada: No utiliza nodos de direcciones para los *overflows*, éstos van a nodos especiales.

Dispersión doble: Las técnicas de saturación tienden a agrupar en zonas contiguas y generan búsquedas largas cuando la densidad tiende a uno. La solución de esta técnica de resolución de colisiones es almacenar los registros de *overflow* en zonas no relacionadas, aplicándoles una segunda función *hash* a la llave para producir un número entero, el cual se suma a la dirección original tantas veces como sea necesario hasta encontrar una dirección con espacio.

PARTE II: Dispersión Extensible.

Ejercicio 7.

Para las siguientes claves, realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada nodo tiene capacidad para dos registros. El número natural indica el orden de llegada de las operaciones. Se debe mostrar el estado del archivo para cada operación. Justificar, brevemente, ante colisión y desborde los pasos que se realizan.

1	*+ Darín*	*00111111*	*2*	*+ Alterio*	*11110100*
3	*+ Sbaraglia*	*10100101*	*4*	*+ De la Serna*	*01010111*
5	*+ Altavista*	*01101011*	*6*	*+ Grandinetti*	*10101010*
7	*- Altavista*	*01101011*	*8*	*- Sbaraglia*	*10100101*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Darín”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Darín (00111111)	

Inserción de clave “Alterio”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Darín (00111111)	Alterio (11110100)

Inserción de clave “Sbaraglia”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Alterio (11110100)	
1	1	Darín (00111111)	Sbaraglia (10100101)

Inserción de clave “De la Serna”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	1
(10)	0
(11)	2

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Alterio (11110100)	
1	2	Sbaraglia (10100101)	
2	2	Darín (00111111)	De la Serna (01010111)

Inserción de clave “Altavista”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	1
(010)	0
(011)	2
(100)	0
(101)	1
(110)	0
(111)	3

#Bloque	Bits	Clave R1	Clave R2
0	1	Alterio (11110100)	
1	2	Sbaraglia (10100101)	
2	3	Altavista (01101011)	
3	3	Darín (00111111)	De la Serna (01010111)

Inserción de clave “Grandinetti”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	1
(010)	0
(011)	2
(100)	0
(101)	1
(110)	0
(111)	3

#Bloque	Bits	Clave R1	Clave R2
0	1	Alterio (11110100)	Grandinetti (10101010)
1	2	Sbaraglia (10100101)	
2	3	Altavista (01101011)	
3	3	Darín (00111111)	De la Serna (01010111)

Baja de clave “Altavista”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	1
(10)	0
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Alterio (11110100)	Grandinetti (10101010)
1	2	Sbaraglia (10100101)	
2	~~3~~	~~Altavista (01101011)~~	
3	2	Darín (00111111)	De la Serna (01010111)

Se borra la clave “Altavista” y el bloque 2 se puede liberar, ya que se puede fusionar con el bloque 3.

Baja de clave “Sbaraglia”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Alterio (11110100)	Grandinetti (10101010)
1	~~2~~	~~Sbaraglia (10100101)~~	
2	~~3~~	~~Altavista (01101011)~~	
3	1	Darín (00111111)	De la Serna (01010111)

Se borra la clave “Sbaraglia” y el bloque 1 se puede liberar, ya que se puede fusionar con el bloque 3.

Ejercicio 8.

Realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada registro tiene capacidad para dos claves. El número natural indica el orden de llegada de las operaciones. Se debe mostrar el estado del archivo para cada operación. Justificar, brevemente, ante colisión y desborde los pasos que se realizan.

1	*+ Buenos Aires*	*...1001*	*2*	*+ San Juan*	*...0100*
3	*+ Entre Ríos*	*...1110*	*4*	*+ Corrientes*	*...0010*
5	*+ San Luis*	*...0101*	*6*	*+ Tucumán*	*...0111*
7	*+ Río Negro*	*...0011*	*8*	*+ Jujuy*	*...1111*
9	*+ Salta*	*...1010*	*10*	*- Río Negro*	*...0011*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Buenos Aires”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Buenos Aires (…1001)	

Inserción de clave “San Juan”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Buenos Aires (…1001)	San Juan (…0100)

Inserción de clave “Entre Ríos”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	San Juan (…0100)	Entre Ríos (…1110)
1	1	Buenos Aires (…1001)	

Inserción de “Corrientes”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	2

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	2	Entre Ríos (…1110)	Corrientes (…0010)
2	1	Buenos Aires (…1001)	

Inserción de “San Luis”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	2

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	2	Entre Ríos (…1110)	Corrientes (…0010)
2	1	Buenos Aires (…1001)	San Luis (…0101)

Inserción de “Tucumán”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	2	Entre Ríos (…1110)	Corrientes (…0010)
2	2	Buenos Aires (…1001)	San Luis (…0101)
3	2	Tucumán (…0111)	

Inserción de “Río Negro”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	2	Entre Ríos (…1110)	Corrientes (…0010)
2	2	Buenos Aires (…1001)	San Luis (…0101)
3	2	Tucumán (…0111)	Río Negro (…0011)

Inserción de “Jujuy”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	1
(011)	3
(100)	0
(101)	2
(110)	1
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	2	Entre Ríos (…1110)	Corrientes (…0010)
2	2	Buenos Aires (…1001)	San Luis (…0101)
3	3	Río Negro (…0011)	
4	3	Tucumán (…0111)	Jujuy (…1111)

Inserción de “Salta”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	3
(010)	1
(011)	4
(100)	0
(101)	3
(110)	2
(111)	5

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	3	Corrientes (…0010)	Salta (…1010)
2	3	Entre Ríos (…1110)	
3	2	Buenos Aires (…1001)	San Luis (…0101)
4	3	Río Negro (…0011)	
5	3	Tucumán (…0111)	Jujuy (…1111)

Baja de clave “Río Negro”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	3
(010)	1
(011)	5
(100)	0
(101)	3
(110)	2
(111)	5

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	San Juan (…0100)	
1	3	Corrientes (…0010)	Salta (…1010)
2	3	Entre Ríos (…1110)	
3	2	Buenos Aires (…1001)	San Luis (…0101)
4	~~3~~	~~Río Negro (…0011)~~	
5	2	Tucumán (…0111)	Jujuy (…1111)

Se borra la clave “Río Negro” y el bloque 4 se puede liberar, ya que se puede fusionar con el bloque 5.

Ejercicio 9.

Para las siguientes claves, realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada nodo tiene capacidad para dos registros. El número natural indica el orden de llegada de las operaciones. Se debe mostrar el estado del archivo para cada operación. Justificar, brevemente, ante colisión y desborde los pasos que se realizan.

1	*+ Tristana*	*11110010*	*2*	*+ Jarvan IV*	*00111010*
3	*+ Teemo*	*01010100*	*4*	*+ Annie*	*10100101*
5	*+ Ryze*	*10101110*	*6*	*+ Morgana*	*01101011*
7	*+ Garen*	*11001011*	*8*	*- Teemo*	*01010100*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Tristana”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Tristana (11110010)	

Inserción de clave “Jarvan IV”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Tristana (11110010)	Jarvan IV (00111010)

Inserción de clave “Teemo”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Teemo (01010100)	
1	1	Tristana (11110010)	Jarvan IV (00111010)

Inserción de clave “Annie”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	1
(10)	2
(11)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Teemo (01010100)	
1	2	Annie (10100101)	
2	2	Tristana (11110010)	Jarvan IV (00111010)

Inserción de clave “Ryze”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	1
(010)	2
(011)	0
(100)	0
(101)	1
(110)	3
(111)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Teemo (01010100)	
1	2	Annie (10100101)	
2	3	Tristana (11110010)	Jarvan IV (00111010)
3	3	Ryze (10101110)	

Inserción de clave “Morgana”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	1
(010)	2
(011)	0
(100)	0
(101)	1
(110)	3
(111)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Teemo (01010100)	Morgana (01101011)
1	2	Annie (10100101)	
2	3	Tristana (11110010)	Jarvan IV (00111010)
3	3	Ryze (10101110)	

Inserción de clave “Garen”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	3
(011)	1
(100)	0
(101)	2
(110)	4
(111)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Teemo (01010100)	
1	2	Morgana (01101011)	Garen (11001011)
2	2	Annie (10100101)	
3	3	Tristana (11110010)	Jarvan IV (00111010)
4	3	Ryze (10101110)	

Baja de clave “Teemo”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	3
(011)	1
(100)	0
(101)	2
(110)	4
(111)	1

#Bloque	Bits	Clave R1	Clave R2
0	~~2~~	~~Teemo (01010100)~~	
1	2	Morgana (01101011)	Garen (11001011)
2	2	Annie (10100101)	
3	3	Tristana (11110010)	Jarvan IV (00111010)
4	3	Ryze (10101110)	

Se borra la clave “Teemo” y el bloque 0 no se puede liberar, ya que no se puede fusionar.

Ejercicio 10.

Para las siguientes claves, realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada nodo tiene capacidad para dos registros. El número natural indica el orden de llegada de las operaciones. Se debe mostrar el estado del archivo para cada operación. Justificar, brevemente, ante colisión y desborde los pasos que se realizan.

1	*+ Guillermo.B*	*01100011*	*2*	*+ Gómez*	*00000001*
3	*+ Gustavo.B*	*01010110*	*4*	*+ Sosa*	*11110100*
5	*+ Enría*	*00110101*	*6*	*+ Guli*	*00101000*
7	*- Gustavo.B*	*01010110*	*8*	*- Sosa*	*11110100*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Guillermo.B”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Guillermo.B (01100011)	

Inserción de clave “Gómez”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Guillermo.B (01100011)	Gómez (00000001)

Inserción de clave “Gustavo.B”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	Bits	Clave R1	Clave R2
0	1	Gustavo.B (01010110)	
1	1	Guillermo.B (01100011)	Gómez (00000001)

Inserción de clave “Sosa”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	Bits	Clave R1	Clave R2
0	1	Gustavo.B (01010110)	Sosa (11110100)
1	1	Guillermo.B (01100011)	Gómez (00000001)

Inserción de clave “Enria”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	1
(10)	0
(11)	2

#Bloque	Bits	Clave R1	Clave R2
0	1	Gustavo.B (01010110)	Sosa (11110100)
1	2	Enria (00110101)	
2	2	Guillermo.B (01100011)	Gómez (00000001)

Inserción de clave “Guli”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Sosa (11110100)	Guli (00101000)
1	2	Gustavo.B (01010110)	
2	2	Enria (00110101)	
3	2	Guillermo.B (01100011)	Gómez (00000001)

Baja de clave “Gustavo.B”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	0
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Sosa (11110100)	Guli (00101000)
1	~~2~~	~~Gustavo.B (01010110)~~	
2	2	Enria (00110101)	
3	2	Guillermo.B (01100011)	Gómez (00000001)

Se borra la clave “Gustavo.B” y el bloque 1 se puede liberar, ya que se puede fusionar con el bloque 0.

Baja de clave “Sosa”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	0
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	~~Sosa (11110100)~~	Guli (00101000)
1	~~2~~	~~Gustavo.B (01010110)~~	
2	2	Enria (00110101)	
3	2	Guillermo.B (01100011)	Gómez (00000001)

Se borra la clave “Sosa” y el bloque 0 no se puede liberar, ya que sigue ocupado por la clave “Guli”.

Ejercicio 11.

Para las siguientes claves, realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada nodo tiene capacidad para dos registros. El número natural indica el orden de llegada de las operaciones. Se debe mostrar el estado del archivo para cada operación. Justificar, brevemente, ante colisión y desborde los pasos que se realizan.

1	*+ Mansilla*	*01100010*	*2*	*+ Cetré*	*10001000*
3	*+ Ascacibar*	*01010111*	*4*	*+ Carrillo*	*11110101*
5	*+ Manyoma*	*00110100*	*6*	*+ Méndez*	*00101001*
7	*+ Alario*	*11000101*	*8*	*- Mansilla*	*01100010*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Mansilla”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Mansilla (01100010)	

Inserción de clave “Cetré”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Mansilla (01100010)	Cetré (10001000)

Inserción de clave “Ascacibar”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Mansilla (01100010)	Cetré (10001000)
1	1	Ascacibar (01010111)	

Inserción de clave “Carrillo”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Mansilla (01100010)	Cetré (10001000)
1	1	Ascacibar (01010111)	Carrillo (11110101)

Inserción de clave “Manyoma”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	2

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Cetré (10001000)	Manyoma (00110100)
1	2	Mansilla (01100010)	
2	1	Ascacibar (01010111)	Carrillo (11110101)

Inserción de clave “Méndez”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Cetré (10001000)	Manyoma (00110100)
1	2	Mansilla (01100010)	
2	2	Carrillo (11110101)	Méndez (00101001)
3	2	Ascacibar (01010111)	

Inserción de clave “Alario”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	1
(011)	4
(100)	0
(101)	3
(110)	1
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Cetré (10001000)	Manyoma (00110100)
1	2	Mansilla (01100010)	
2	3	Méndez (00101001)	
3	3	Carrillo (11110101)	Alario (11000101)
4	2	Ascacibar (01010111)	

Inserción de clave “Mansilla”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	0
(011)	4
(100)	0
(101)	3
(110)	0
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Cetré (10001000)	Manyoma (00110100)
1	~~2~~	~~Mansilla (01100010)~~	
2	3	Méndez (00101001)	
3	3	Carrillo (11110101)	Alario (11000101)
4	2	Ascacibar (01010111)	

Se borra la clave “Mansilla” y el bloque 1 se libera, ya que se puede fusionar con el bloque 0.

Ejercicio 12.

Realizar el proceso de dispersión mediante el método de hashing extensible, sabiendo que cada nodo tiene capacidad para dos claves. El número natural indica el orden de llegada de las operaciones. Se deberán explicar los pasos que se realizan en cada operación y dibujar los estados sucesivos correspondientes (inclusive el estado inicial).

1	*+ Aconcagua*	*10100111*	*2*	*+ Kilimanjaro*	*10101010*
3	*+ Mont Blanc*	*00111110*	*4*	*+ Cervino*	*01101111*
5	*+ Etna*	*00110101*	*6*	*+ Chañi*	*11110000*
7	*+ Cho Oyu*	*01011101*	*8*	*+ Vinicunca*	*01011011*
9	*- Chañi*	*11110000*	*10*	*- Cervino*	*01101111*

Estado inicial de la tabla de dispersión y del archivo:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0		

Inserción de clave “Aconcagua”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Aconcagua (10100111)	

Inserción de clave “Kilimanjaro”:

Bits de dispersión: 0	
Sufijo	**#Bloque**
(0)	0

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	0	Aconcagua (10100111)	Kilimanjaro (10101010)

Inserción de clave “Mont Blanc”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Kilimanjaro (10101010)	Mont Blanc (00111110)
1	1	Aconcagua (10100111)	

Inserción de clave “Cervino”:

Bits de dispersión: 1	
Sufijo	**#Bloque**
(0)	0
(1)	1

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Kilimanjaro (10101010)	Mont Blanc (00111110)
1	1	Aconcagua (10100111)	Cervino (01101111)

Inserción de clave “Etna”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	1
(10)	0
(11)	2

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	1	Kilimanjaro (10101010)	Mont Blanc (00111110)
1	2	Etna (00110101)	
2	2	Aconcagua (10100111)	Cervino (01101111)

Inserción de clave “Chañi”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	Bits	Clave R1	Clave R2
0	2	Chañi (11110000)	
1	2	Kilimanjaro (10101010)	Mont Blanc (00111110)
2	2	Etna (00110101)	
3	2	Aconcagua (10100111)	Cervino (01101111)

Inserción de clave “Cho Oyu”:

Bits de dispersión: 2	
Sufijo	**#Bloque**
(00)	0
(01)	2
(10)	1
(11)	3

#Bloque	Bits	Clave R1	Clave R2
0	2	Chañi (11110000)	
1	2	Kilimanjaro (10101010)	Mont Blanc (00111110)
2	2	Etna (00110101)	Cho Oyu (01011101)
3	2	Aconcagua (10100111)	Cervino (01101111)

Inserción de clave “Vinicunca”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	0
(001)	2
(010)	1
(011)	3
(100)	0
(101)	2
(110)	1
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	2	Chañi (11110000)	
1	2	Kilimanjaro (10101010)	Mont Blanc (00111110)
2	2	Etna (00110101)	Cho Oyu (01011101)
3	3	Vinicunca (01011011)	
4	3	Aconcagua (10100111)	Cervino (01101111)

Baja de clave “Chañi”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	1
(001)	2
(010)	1
(011)	3
(100)	1
(101)	2
(110)	1
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	~~2~~	~~Chañi (11110000)~~	
1	2	Kilimanjaro (10101010)	Mont Blanc (00111110)
2	2	Etna (00110101)	Cho Oyu (01011101)
3	3	Vinicunca (01011011)	
4	3	Aconcagua (10100111)	Cervino (01101111)

Se borra la clave “Chañi” y el bloque 0 se libera, ya que se puede fusionar con el bloque 1.

Baja de clave “Cervino”:

Bits de dispersión: 3	
Sufijo	**#Bloque**
(000)	1
(001)	2
(010)	1
(011)	3
(100)	1
(101)	2
(110)	1
(111)	4

#Bloque	**Bits**	**Clave R1**	**Clave R2**
0	~~2~~	~~Chañi (11110000)~~	
1	2	Kilimanjaro (10101010)	Mont Blanc (00111110)
2	2	Etna (00110101)	Cho Oyu (01011101)
3	3	Vinicunca (01011011)	
4	3	Aconcagua (10100111)	~~Cervino (01101111)~~

Se borra la clave “Cervino” y el bloque 4 no se libera, ya que sigue ocupado por la clave “Aconcagua”.