Programación 2

Guía de Ejercicios
Archivos de texto y archivos binarios

Estructuras De Datos: Registros y Arrays. Archivos de Texto y Binarios. Operaciones. Estructuras combinadas.

Temas de la guía

I. **Archivos de Texto:** Declaración. Operaciones: crear, leer, grabar, cerrar. Archivos como parámetros.

II. **Estructura de Datos:** Registro. Campo. Operaciones con registros. Registros anidados. Registros como parámetros.

III. **Archivos Binarios:** Declaración. Operaciones: crear, leer, grabar, actualizar, cerrar, añadir, consultar, seleccionar, borrar, renombrar. Acceso secuencial y directo.
Búsqueda Binaria. Archivos como parámetros.

IV. **Apareo, corte de control, y altas, bajas y modificaciones** en archivos binarios.

V. **Estructura de Datos:** Arrays unidimensionales (vectores). Operaciones con vectores. Valores máximos, mínimos, posición del vector. Ordenamiento. Búsqueda Binaria. Array como parámetros.

VI. **Estructura de Datos:** Arrays bidimensionales (matrices). Operaciones con matrices. Fila, columna, diagonal. Valores máximos, mínimos, posición de la matríz. Array como parámetros.

VII. **Estructuras Combinadas:** Registros con campos de tipo array, Arrays de registros. Estructuras combinadas como parámetros y argumentos. Apareo, Corte de control.

Ejercicios Integradores.

Introducción al tema de archivos

Archivos son estructuras de datos con almacenamiento físico en el disco. Esto hace que el dato en el contenido persista más allá de la aplicación pero que el procesamiento, por ser físico, sea lento.
En el caso de C, para poder trabajar con estas estructuras se requiere declarar una variable de memoria para vincularla con ese archivo físico en el disco.
Según el tipo de dato que se almacena en un archivo, los mismos pueden ser:

- Archivos de texto: secuencias de líneas, compuestas por cero o más caracteres que finalizan con una marca que indican el fin de línea. Al final de la estructura existe otra marca que indica el final del archivo.
- Archivo binario: secuencia de bytes, nosotros utilizamos:
    - Archivos de tipo: colección de datos, con almacenamiento físico, todos del mismo tipo.
        - Archivos de registro: el tipo de dato que tiene en cada posición es un registro
        - Archivos de registro de tamaño fijo: todos los registros con la misma longitud.

Los archivos de esta característica son archivos de acceso directo, es decir, para acceder a un registro en particular no es necesario recorrer todas las posiciones anteriores.
Por esta razón, para trabajar con archivos se requiere:

1. Definir un tipo registro y declarar una variable de ese tipo (en caso de archivos de registro, para archivos de texto esto no es necesario)
2. Declarar una variable tipo archivo (es el nombre interno o lógico del archivo) y vincularlo con el archivo físico que esta en el disco
   En el caso de C, una variable de este tipo es una variable FILE *, la vinculación con el nombre externo es mediante la función fopen, esto es:

   FILE * NombreLogico = fopen (NombreFisico, ModoApertura)

   NombreLogico es el nombre de la variable interna
   NombreFisico es el nombre externo o del archivo físico

   Modo de apertura es una cadena que indica r/w si es de lectura/escritura t/b si el tipo de dato es texto o binario y, eventualmente el signo + para agregar la otra modalidad.

   Las posibilidades son:

“r”	abre archivo de texto para lectura
“rt”	abre archivo de texto para lectura
“rb”	abre archivo binario para lectura
“w”	abre archivo de texto para escritura
“wt”	abre archivo de texto para escritura
“wb”	abre archivo binario para escritura
“w”	abre archivo de texto para escritura
“r+”	abre archivo de texto para lectura y puede grabar
“rt+”	abre archivo de texto para lectura y puede grabar
“rb+”	abre archivo binario para lectura y se puede grabar
“w+”	abre archivo de texto para escritura y se puede leer
“wt+”	abre archivo de texto para escritura y se puede leer
“wb+”	abre archivo de binario para escritura y se puede leer
“a”	abre archivo de texto para escritura desde el final
“a+”	abre el archivo de texto para escritura desde el final y puede leer
“ab”	abre archivo binario para escritura desde el final
“ab+”	abre el archivo binario para escritura desde el final y puede leer

3. Funciones de archivos usadas en la materia

Archivos de Texto

Función	Descripción
FILE *f;	Define f como puntero a FILE
f = fopen (“archivo”, “w”);	Asocia f a un flujo
f = freopen(“archivo”, “w”, f);	Similar anterior, si está abierto antes lo cierra
fclose(f);	Vacía el flujo y cierra el archivo asociado
fflush(f);	Produce el vaciado de los flujos
remove(“archivo”);	El archivo ya no queda accesible
rename(“viejo”, “nuevo”);	Renombra con nuevo el viejo nombre
fprintf(f, “%d”, valor);	Escritura con formato en un flujo
fscanf(f, “%d”, &valor);	Lectura con formato desde un flujo
c = getchar();	Lee un carácter desde stdin
c = getc(f);	Lee un carácter desde el flujo
c = fgetc(f);	Igual que el anterior
ungetc (c, f);	Retorna el carácter al flujo y retrocede
putchar(c) ;	Escribe un carácter en stdin
putc(c, f);	Escribe un carácter en el flujo
fputc(c,f);	Igual al anterior
gets(s);	Lee una cadena de stdin
fgets(s, n, f);	Lee hasta n-1 carácter del flujo en s
puts(s);	Escribe una cadena en stdin
fputs(s, f);	Escribe la cadena s en el flujo
feof(f)	Retorna no cero si el indicador de fin está activo
ferror(f);	Retorna no cero si el indicador de error está activo
clearerr(f);	Desactiva los indicadores de error

Archivos Binarios

Función	Descripción
FILE *f;	Define f como puntero a FILE
f = fopen (“archivo”, “wb”);	Asocia f a un flujo
f = fropen(“archivo”, “wb”);	Similar anterior, si está abierto antes lo cierra
fclose(f);	Vacía el flujo y cierra el archivo asociado
fflush(f);	Produce el vaciado de los flujos
remove(“archivo”);	El archivo ya no queda accesible
rename(“viejo”, “nuevo”);	Renombra con nuevo el viejo nombre
sizeof(tipo)	Retorna el tamaño de un tipo o identificador
SEEK_CUR	Constante asociada a fseek (lugar actual)
SEEK_END	Constante asociada a fseek (desde el final)
SEEK_SET	Constante asociada a fseek (desde el inicio)
size_t fread(&r, tam,cant, f)	Lee cant bloques de tamaño tam del flujo f
size_t fwrite(&r,tam,cant,f)	Graba cant bloques de tamaño tam del flujo f
fgetpos(f, pos)	Almacena el valor actual del indicador de posición
fsetpos(f,pos)	Define el indicador de posición del archive en pos
ftell(f)	El valor actual del indicador de posición del archivo
fseek(f, cant, desde)	Define indicador de posición a partir de una posición.

Ejemplo de binario a texto (binario existe y de texto se debe crear)

```
……..
struct tr { int c1,c2 };
 tr r;

FILE* f1 = fopen(“binario.bin”,”rb+”);
FILE* f2 = fopen(“texto.txt”, “wt+”);
fread(&r, sizeof(r), 1, f1);
while (!feof(f1))
        fprintf(f2,”%d %d”, r.c1,r.c2);
        fread(&r, sizeof(r), 1, f1);
fclose (f1);
fclose(f2);
……
```

Tamaños de tipos básicos

Enteros

Tipo de dato	Tamaño	Rango numérico
char	1 byte	-128 a 127
unsigned char (byte)	1 byte	0 a 255
short	2 bytes	-32,768 a 32,767
unsigned short (word)	2 bytes	0 a 65,535
int	4 bytes	-2,147,483,648 a 2,147,483,647
unsigned int	4 bytes	0 a 4,294,967,295
long / long int	4 bytes	-2,147,483,648 a 2,147,483,647
unsigned long (dword)	4 bytes	0 a 4,294,967,295

Si bien el tipo de dato int puede valer 2 o 4 bytes según la arquitectura de la computadora, tomamos el valor máximo como referencia para simplificar.

Reales

Tipo de dato	Tamaño	Rango numérico	Decimales
float	4 byte	1.2E-38 a 3.4E+38	6 decimales
double	8 byte	2.3E-308 a 1.7E+308	15 decimales
long double	10 bytes	3.4E-4932 a 1.1E+4932	19 decimales

I. Archivos de Texto

Ej. 1: Se conoce de cada alumno de un curso los siguientes datos: legajo (8dígitos) y las notas de 2 parciales (0..10), que finaliza con un legajo negativo.

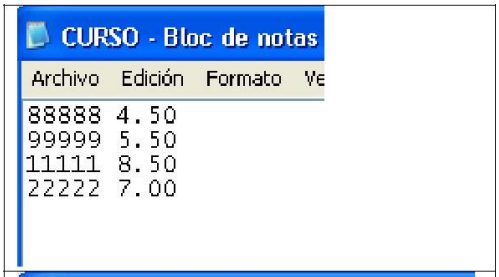
Se pide desarrollar un programa que ingrese los datos de los alumnos por teclado y grabe un archivo de texto CURSO.TXT con una línea por cada alumno, con el número de legajo y su promedio (real).

	De Entrada	De Salida
DATOS	Legajo Nota1 Nota2 } por cada alumno del curso	CURSO.TXT :Archivo de texto con una línea por cada alumno con los siguientes datos Legajo Promedio } un alumno por línea

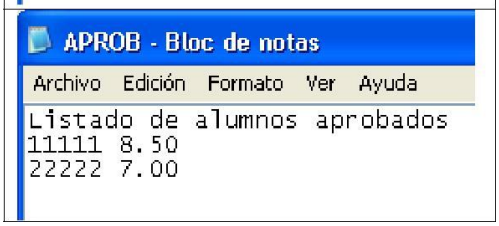
Ej. 2: Dado el archivo del **EJ. 1** grabe otro archivo APROB.TXT que contenga una primer línea con el siguiente título "Listado de alumnos aprobados", y a continuación una por cada alumno con el legajo y promedio de aquellos alumnos cuyo promedio sea >= 6.

Archivos de entrada y de salida

Archivo de Entrada
CURSO.TXT



Archivo de Salida
APROB.TXT



II. Registros. Operaciones:

Ej. 3: Dado el siguiente dibujo de un registro, especifique la nomenclatura para acceder al registro y cada uno de sus campos, y las operaciones que se pueden realizar a nivel registro y a nivel campos. Defina otro registro del mismo tipo de datos.

Legajo	ApellidoNombre	Nota
8 dígitos	25 caracteres	0..10

Ej. 4 Dados los siguientes dibujos que corresponden a distintos tipos de registros realice su declaración en C++, y especifique la nomenclatura para acceder al registro y cada uno de sus campos.

a)

Nombre (20 caracteres)	FechaNacimiento		
	Dia (1..31)	Mes (1..12)	Anio (4 dígitos)
char[20]	char	char	short

b)

Legajo (8 dígitos)	ApellidoNombre		Calificaciones		
	Apellido (20 caracteres)	Nombre (20 caracteres)	Nota1	Nota2	Nota3

Ej. 5: Realice diagrama y codificación para el desarrollo del siguiente enunciado.

Enunciado: Dado un conjunto de Nombres y Fechas de nacimientos (AAAAMMDD), que finaliza con un Nombre = "FIN", informar el nombre de la persona de mayor edad.

Nombre del registro REG

Nombre	Fecha (AAAAMMDD)
20 caracteres	8 dígitos enteros

II. Archivos Binarios

Ej. 6: Se dispone de un conjunto de boletas de inscripción de alumnos a examen en el mes de mayo.

Cada boleta tiene los siguientes datos: nombre y apellido, número de legajo, código de materia, día, mes y año del examen. Los datos finalizan con un nombre y apellido nulo.

Desarrollar un programa que a partir del ingreso de las boletas mencionadas, por teclado, genere un archivo binario de inscripción de alumnos a exámenes finales DIAFINALES.DAT, según el siguiente diseño:

a.1 Nro. de legajo (8 dígitos) a.2 Código de materia (6 dígitos)
a.3 Día del examen (1..31) a.4 Mes del examen (1..12)
a.5 Año del examen (4 dígitos) a.6 Nombre-Apellido (25caract)

Ej. 7: Dado el archivo binario generado en el ejercicio **7**, desarrolle un programa que solicitando por teclado un código de materia permita seleccionar todos los registros que se anotaron para rendirla y los grabe en otro archivo (MATFINALES.DAT), con el mismo diseño.

Ej. 8: Dado el archivo binario generado en el ejercicio **7**, desarrollar la codificación en C o C++ que graba un archivo de texto, LISTADO.TXT, de los alumnos inscriptos de acuerdo al siguiente diseño.

Legajo	Nombre y Apellido	Fecha	Código de materia
99999999	xxxxxxxxxxxxxxxxxxx	dd/mm/aa	999999
99999999	xxxxxxxxxxxxxxxxxxx	dd/mm/aa	999999

Realice la estrategia de resolución, la representación gráfica del algoritmo, y dibuje el diseño del registro

Ej. 9: Dado el archivo binario generado en el ejercicio **7**, que contiene todas las inscripciones del día, y otro con el mismo diseño que contiene las inscripciones anteriores (FINALES.DAT), desarrolle un

programa que agregue al archivo de inscripciones anteriores el contenido del archivo del día. Al final del proceso emita un listado del archivo de los registros agregados al archivo.

Estrategia:

- Asignar archivos
- Abrir archivos para leer y actualizar
- Ubicarse al final del archivo de inscripciones anteriores
- Recorrer secuencialmente el archivo de inscripciones del día
  - Leer registro archivo de inscripciones del día
  - Grabar registro en archivo de inscripciones anteriores
- Ubicarse en el primer registro de los agregados
- Recorrer secuencialmente el archivo de inscripciones anteriores desde donde fue ubicado
  - Leer registro archivo de inscripciones anteriores
  - Listar datos del registro
- Cerrar archivos

Ej. 10: Dado el archivo binario generado en el ejercicio **7**, desarrolle un programa que genere un archivo ordenado por número de legajo (cada registro debe tener los campos legajo y apellido y nombre) para todos los alumnos que se inscribieron una o más veces. Cada legajo debe ocupar una posición única y predecible en el archivo. El intervalo de los legajos es 80001 a 110000. Pueden no presentarse todos los legajos

Ej. 11: Dado el archivo binario generado en el ejercicio **11**, desarrolle un programa que elimine, si los hubiese, los registros que no contengan datos válidos de la siguiente manera:

a) Genere un nuevo archivo, elimine el archivo original y renombre al archivo actual
b) Compacte en el mismo archivo

Ej. 12: Dado un archivo binario de productos que contiene 100 registros, y cada registro corresponde a un producto que está codificado del 1 a 100, ordenado por código de producto
con el siguiente diseño:

código de producto (1..100, char) stock (int)

y otro archivo binario de pedidos, con el siguiente diseño:
Número de pedido(int) número de cliente (long)
Código de producto (1..100, char) cantidad pedida (int)

Se pide desarrollar un algoritmo que:

a) grabar un archivo de texto con los pedidos que fueron satisfechos en su totalidad según el siguiente listado:

Pedidos Satisfechos

Número de pedido	Número de Cliente	Código de Producto	Cantidad pedida
9999	999999	999	9999
9999	999999	999	9999

b) actualizar el campo stock del archivo de producto, cada vez que acepte un pedido.
c) grabar un archivo binario con el mismo diseño que el archivo de pedidos, con aquellos pedidos que no pueden ser satisfechos en su totalidad.

III. Apareo, Corte de Control, y altas, bajas y modificaciones en archivos binarios.

Ej. 13: Se dispone un archivo binario de inscripción de alumnos a exámenes finales MAESTROFINALES.DAT y otro con las inscripciones del día de hoy DIAFINALES.DAT, ambos ordenados ascendente por código de materia y con el siguiente diseño:

a.1Nro de legajo (8 dígitos) a.2 Código de materia (6 dígitos) a.3 ApellidoNombre(25caract)

Se pide desarrollar un programa que genere un nuevo archivo de inscripciones a finales FINALESACT.DAT resultante del apareo de los dos archivos anteriores, con el mismo orden y diseño.

Estrategia:

- Asignar y abrir archivos
- Leer registro archivo maestro con control de EOF
- Leer registro archivo finales del día con control de EOF
- Mientras no sea fin de archivo de ninguno de los dos archivos
    - Si el código de materia del registro del maestro es menor al código de materia del registro del archivo del día
        - Grabar registro archivo maestro en archivo de finales actualizado
        - Leer registro archivo maestro con control de EOF
    - De lo contrario
        - Grabar registro archivo finales del día en archivo de finales actualizado
        - Leer registro archivo finales del día con control de EOF
- Por fin de archivo del maestro mientras no sea fin de archivo de finales del día
    - Grabar registro archivo finales del día en archivo de finales actualizado
    - Leer registro archivo finales del día con control de EOF
- Por fin de archivo finales del día mientras no sea fin de archivo maestro
    - Grabar registro archivo maestro en archivo de finales actualizado
    - Leer registro archivo maestro con control de EOF
- Cerrar archivos

Ej. 14: Una empresa de cable desea actualizar el archivo de series que emite por sus distintos canales, con material nuevo. Para ello posee los siguientes archivos:

a) un archivo maestro de series, **Series.dat**, con un registro con los datos de cada series, ***ordenado ascendente por Id_Serie***, con el siguiente diseño:

a.1) Id_Serie (9 dígitos) a.2) Título de la serie (20 caracteres) a.3) Genero (10 caracteres)

b) otro archivo de novedades de series, **NovSeries.dat**, con el mismo diseño y orden que el archivo anterior, que contiene las novedades a incorporar.

Se pide desarrollar la metodología necesaria para escribir un algoritmo que Grabe un archivo maestro de series actualizado, **ActSeries.dat,** con el mismo diseño y orden que los anteriores.

Ej. 15: El dueño de un local de venta de libros desea relevar el stock que posee en el local y en el depósito, para realizar las compras del mes.
Para ello cuenta con dos archivos:

a) **StockEnLocal.dat**, ***ordenado por código del libro***, con un registro por cada libro que se encuentra en el local, con el siguiente diseño:

a.1) Código de libro (4 dígitos)
a.3) Autor (20 caracteres)
a.6) Stock en el local (char)
a.2) Título del libro (30 caracteres)
a.4) Editorial (20 caracteres)
a.7) Genero (10 caracteres)

b) **StockEnDeposito.dat**, ***ordenado por código del libro***, con un registro por cada libro que se encuentra en el depósito, con el siguiente diseño:

b.1) Código de libro (4 dígitos) b.2) Stock en depósito (unsigned char)

Se pide desarrollar la metodología necesaria para escribir un algoritmo que emita un listado ***ordenado por código de libro***, con un renglón por cada libro que tenga faltante en stock sea en depósito, local o en ambos lugares, con el siguiente formato:

```
        Libros faltantes
Código              Observación
9999                Falta en depósito
9999                Falta en local
9999                Falta en local y en depósito
   Total de libros en falta: 9999999
```

Ej. 16: Una aplicación para descargas de Series posee la información en un archivo binario, **Episodios.dat**, con un registro por cada episodio, ***ordenado ascendente por Id_Serie y Número de temporada***, con el siguiente diseño:

1) Id_Serie (9 dígitos)
3) Número de temporada (1..12)
5) Cantidad de descargas (long)
2) Título del episodio (20 caracteres)
4) Número de episodio (unsigned char)
6) Fecha de última descarga (aaaammdd)

Se pide desarrollar la metodología necesaria para escribir un algoritmo emita el siguiente listado:

```
                    Listado de Descargas de Series
        Serie: 999999999
          Temporada: 99
           N. de EpisodioTítulo del Episodio      Cant. descargasFecha de última descarga
                    999xxxxxxxxxxxxx                 9999999dd/mm/aaaa
                    999xxxxxxxxxxxxx                 9999999dd/mm/aaaa
……..……..……
Cant. Total de Episodios de la Serie: 99999
Total descargas de la temporada:999999999

        Serie: 999999999
          Temporada: 99
           N. de EpisodioTítulo del Episodio      Cant. descargasFecha de última descarga
                    999xxxxxxxxxxxxx                 9999999dd/mm/aaaa
                    999xxxxxxxxxxxxx                 9999999dd/mm/aaaa
…..……..……
        *Cant. Total de Episodios de la temporada: 99999
        *Total descargas de la temporada: 999999999
**Cant. Total de Episodios de la Serie: 99999
**Total descargas de la Serie: 999999999
…..……...
***Total General de series: 9999
```

Ej. 17: El dueño de un local de venta de juegos para distintas consolas necesita desarrollar un algoritmo que grabe un archivo, **JuegosPorConsola.dat, *ordenado por consola,*** con un solo registro por consola según el siguiente diseño:

a) Consola (10 caracteres) b) Cantidad de juegos (4 dígitos)

Para obtener la información solicitada se cuenta con el archivo **Juegos.dat, *ordenado por consola***, con un registro por cada juego que se encuentra en el local, con el siguiente diseño:

1) Código del juego (6 dígitos)
3) Stock en el local (char)
2) Titulo del Juego (30 caracteres)
4) Consola (10 caracteres)

Ej. 18: Dado el archivo "ALUMNOS.dat" con los datos personales de alumnos ordenado por legajo, con el siguiente diseño:

a.1 Legajo (8 dígitos)	a.2	Apellido y nombre (30 caracteres)
a.3 Domicilio (20 caracteres)	a.4	Código postal (4 dígitos)
a.5 Teléfono (10 caracteres)	a.6	Año de ingreso (4 dígitos)

Y otro archivo con el mismo orden que el mencionado llamado "NOVEDADES.dat", con las actualizaciones (altas, bajas, y modificaciones) a ser aplicadas, donde cada registro contiene además de todos los campos de Alumnos.dat un código de operación ("A"= Alta, "B"= Baja, "M"= Modificación).

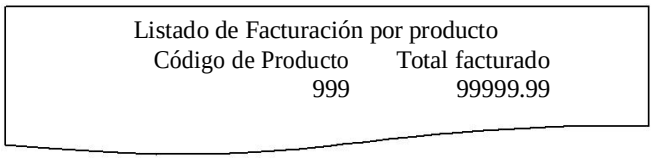
a- Desarrollar todos los pasos necesarios para realizar un programa que genere un archivo actualizado "ALUMACTU.dat" con el mismo diseño.
b- Rehacer el ejercicio considerando que el archivo NOVEDADES es de texto separado por tabs en lugar de binario.

IV. Arrays unidimensionales (vectores)

Ej. 19: Dado un archivo de productos codificados de 1 a 100, sin ningún orden, con el siguiente diseño:
código de producto (1..100) precio unitario (single)

y un conjunto de pedidos, y de cada uno se conoce: código de producto, y cantidad de unidades pedidas (long). Los pedidos finalizan con un código de producto igual a 0.
Se pide:

- informar de cada pedido código de producto, unidades, precio unitario e importe
- al final del proceso:
  - informar los códigos de productos que no fueron vendidos
  - grabar un archivo de texto con el siguiente listado



Ej. 20: Dado un conjunto de N cursos (<=20) de una materia, se tiene un archivo "CURSOS.DAT" con la información de cada uno de ellos: código de curso (4 caracteres) y cantidad de alumnos. Además se tiene en otro archivo "ALUMNOS.DAT" con el curso, legajo y nota (1 a 10) de cada alumno.

Se pide:

- informar de la cantidad de alumnos que tuvieron como nota 0, 1, ...,9, 10
- informar al final del proceso el código de curso, el % de aprobados y el de insuficientes de cada curso.

Ej. 21: Una empresa de aviación realiza 500 vuelos semanales a distintos puntos del país y requiere desarrollar un programa para la venta de pasajes.

Para ello dispone de un archivo "Vuelos.dat", con un registro por cada uno de los 500 vuelos que realiza, sin ningún orden, con el siguiente diseño de registro:

a.1) código de vuelo (6 caracteres) a.2) cantidad de pasajes disponibles (3 dígitos) También se dispone de otro archivo "Compradores.dat", con los potenciales compradores y con el siguiente diseño de registro:

b.1) código de vuelo b.2) cantidad de pasajes solicitados (3 dígitos)
b.3) DNI del solicitante (8 dígitos) b.4) apellido y nombre del solicitante (25 caracteres)

Se pide:
1) Para los solicitantes a los cuales se les venden pasajes, emitir el siguiente listado:

DNI	Apellido y Nombre	Cantidad de pasajes	Código de Vuelo
99999999	xxxxxxxxxxxxxxxxxxxxxxxxx	999	999
99999999	xxxxxxxxxxxxxxxxxxxxxxxxx	999	999

2) Al final del proceso emitir el siguiente listado ordenado por código de vuelo

Código de Vuelo	Pasajes disponibles	Pasajes no vendidos
BUE999	999	999
XXX999	999	999

Nota: Se le vende al solicitante si la cantidad de pasajes que solicita está disponible, en caso contrario se computa como pasajes no vendidos. Desarrolle el ejercicio utilizando la siguiente estructura de datos:
Array de registro: VECTOR : array [1..500] of treg. Bytes que ocupa = (7 + 2 + 2) *500 = 5500 bytes

CODIGOV	PASAJESDISP	PASAJESNOV
char[7] (7 bytes)	short (2 bytes)	short (2 bytes)

Ej. 22: Una empresa que distribuye mercadería hacia distintas localidades del interior dispone de los siguientes archivos:
Un archivo "Destinos.dat", con información de la distancia a cada uno de los destinos y con el siguiente diseño de registro:
a.1) número de destino (3 dígitos) a.2) distancia en kilómetros (float)

También se dispone de otro archivo "Viajes.dat", con los viajes realizados por cada camión y con el siguiente diseño de registro:
b.1) patente del camión (6 caracteres, no son más de 200 camiones) b.2) número de destino b.3) número de chofer (1..150)

Se pide desarrollar la metodología necesaria para realizar un programa que informe:

1. Cantidad de viajes realizados a cada destino
2. Número de chofer con menor cantidad de km recorridos
3. Patente de los camiones que viajaron al destino 116 sin repeticiones de las mismas.

V. Arrays bidimensionales (matrices)

Ej. 23: Un negocio de ropa, vende sus artículos en distintos talles. Para realizar la facturación dispone de los siguientes archivos:
Un archivo "Articulos.dat", con los precios de cada artículo y talle que vende, con el siguiente diseño de registro:
a.1) código de articulo (1..100) a.2) talle (1..5) a.3) precio (real)

También se dispone de dispone de un conjunto de ventas a facturar, que se ingresan por teclado con los siguientes datos: código de artículo, talle y unidades (1 dígito). Las ventas finalizan con un código de artículo negativo.
Desarrollar un algortimo que:

1) informe el precio de la venta, considerando un descuento del 10% si las unidades vendidas son superiores a tres más el 21% del IVA.
2) al final del proceso emita el siguiente listado, ordenado por artículo y talle ascendente:

Código de Artículo 999

	Talle	Unidades Vendidas
	9	99
	9	99

Total unidades vendidas artículo 9999999
……..
Total general de unidades vendidas 9999999

-

VI. Estructuras Combinadas: Registros con campos de tipo array, Arrays de registros. Estructuras combinadas como parámetros y argumentos. Apareo, Corte de control. **Ejercicios Integradores.**

Los lenguajes de programación proveen tipos de datos simples, y estructuras de datos, como registro y arrays; la primera puede contener datos heterogéneos y la segunda homogéneos. Desarrollemos posibles combinaciones entre registros y arrays, y describamos su nomenclatura:

- Un registro donde uno de sus campos sea otro registro.

Ejemplo: un registro con los datos de un alumno, donde uno de sus campos sea otro registro.

Alumno

nombreApellido		documento	edad
nombre	apellido		

Alumno nombre del identificador del registro del alumno
Alumno.nombreapellido nombre del identificador del registro con el apellido y nombre
Alumno.nombreapellido.nombre nombre del identificador del campo nombre definido como una cadena de caracteres
alumno.nombreapellido.apellido nombre del identificador del campo apellido definido como una cadena de caracteres
alumno.documento nos ubica en el documento **alumno.edad** nos ubica en la edad

- Un registro con un campo array.

Ejemplo: un registro con los datos de un alumno y sus 10 notas
Alumno

documento	notas			
	1	2.....		10

Alumno nombre del identificador del registro del alumno
Alumno.documento nombre del identificador del campo documento de registro del alumno
Alumno.notas nombre del identificador del array que contiene las 10 notas
Alumno.notas[1], **Alumno.notas[2]**, …….. cada una de las notas del alumno

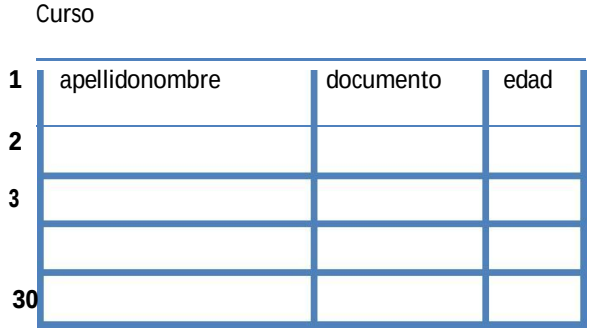
¿Cómo invocar al procedimiento Burbujeo para ordenar el array de notas?

Burbujeo(Alumno.notas,10)

el procedimiento debe recibir como parámetro un array y su cardinalidad.

- Un array de una dimensión (vector) de registros

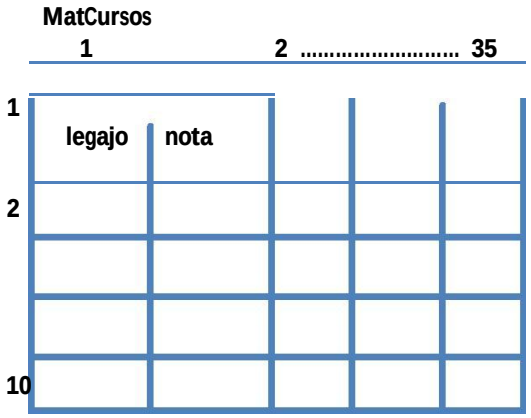
Ejemplo: un array de registro con los datos de los alumnos de un curso



Curso es el array
Curso[1] **es** el registro que contiene los datos del alumno ubicado en el elemento 1
Curso[1].apellidonombre campo del registro que contiene el nombre y apellido del alumno 1
Curso[i].documento campo del registro que contiene el documento del alumno i **Curso[i].edad** campo del registro que contiene la edad del alumno i
¿Cómo invocar al procedimiento Burbujeo para ordenar el array del curso por apellidonombre?
Burbujeo(Curso,30)
el procedimiento debe recibir como parámetros un array de registro, cardinalidad del array y comparar los campos apellidonombre.

- Un array de dos dimensiones (matriz) de registros

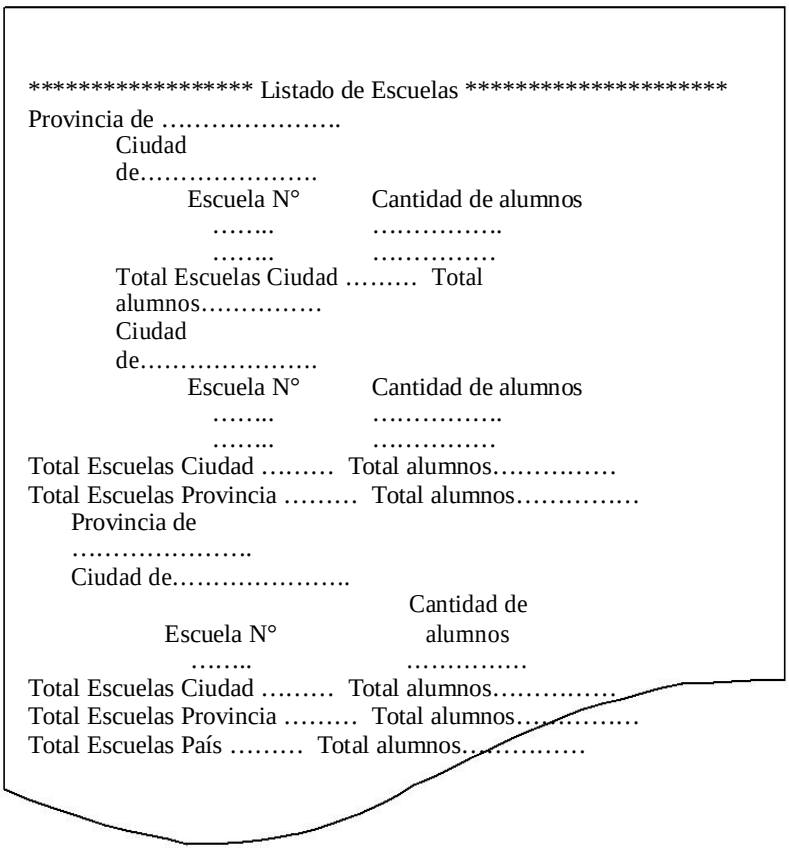
Ejemplo: matriz de registros de 10 filas (que representan los cursos) x 35 columnas(que representan alumnos) donde cada elemento contienen datos del alumnos



MatCursos es la matríz de registros
MatCursos [i,j] **es** el registro que contiene los datos del alumno ubicado en el elemento de la fila i columna j
MatCursos [i,j].legajo campo del registro que contiene el legajo del alumno j del curso i
MatCursos [i,j].nota campo del registro que contiene la nota del alumno j del curso i

Ej. 24: Se realiza un censo en las distintas escuelas del país. La información se encuentra en el archivo binario DATOSESCUELAS.DAT con un registro por cada escuela censada, con el siguiente diseño:
a.1 Nombre de provincia (20 caracteres) a.2 Nombre de la Ciudad (20 caracteres) a.3 Nro. de escuela (3 dígitos) a.4 Cantidad de alumnos (4 dígitos)
El archivo está ordenado por ciudad dentro de provincia. Se pide desarrollar un algoritmo que:

a) emita el siguiente listado:



********** Listado de Escuelas **********
Provincia de …….
Ciudad de…….
Escuela N° Cantidad de alumnos
……... …….
……... …….
Total Escuelas Ciudad …… Total alumnos…….
Ciudad de…….
Escuela N° Cantidad de alumnos
……... …….
……... …….
Total Escuelas Ciudad …… Total alumnos…….
Total Escuelas Provincia …… Total alumnos…….
Provincia de …….
Ciudad de…….
Escuela N° Cantidad de alumnos
……... …….
Total Escuelas Ciudad …… Total alumnos…….
Total Escuelas Provincia …… Total alumnos…….
Total Escuelas País …… Total alumnos…….

b) grabe un archivo binario de totales, ordenado por cantidad de alumnos, con el siguiente diseño:
b.1 Nombre de provincia (20 caracteres)

b.2 Cantidad de alumnos (long)

NOTA: Utilice el siguiente juego de datos para el seguimiento del algoritmo:

Provincia	Ciudad	N° de escuela	Cantidad de Alumnos
BSAS	LA PLATA	10	500
BSAS	LA PLATA	15	800
BSAS	MERCEDES	8	600
SAN LUIS	MERCEDES	3	400

Ej. 25: Dado un archivo binario "ACTASFINALES.dat" que contiene las actas de los exámenes finales de las distintas materias, ordenado por libro y folio y con el siguiente diseño

a.1 Libro (6 dígitos) a.2 Folio (1..999) a.3 Fecha (aaaammdd)
a.4 Código materia (6 dígitos) a.5 Legajo (10 dígitos)
a.6 Apellido y nombre (20 caracteres) a.7 Nota (1..10, 0 indica ausente)

Se pide desarrollar todos los pasos necesarios para realizar un algoritmo que grabe un archivo "TOTALES.dat" ordenado por libro y folio con el siguiente diseño:

b.1 Libro (6 dígitos) b.2 Folio (1..999)
b.3 Total alumnos inscriptos (1..20) b.4 Total alumnos ausentes (1..20)
b.5 Total alumnos aprobados (1..20) b.6 Total alumnos desaprobados (1..20)

Ej. 26: Dado el archivo "ALUMNOS.dat" con los datos personales de alumnos ordenado por legajo, con el siguiente diseño:

a.1	Legajo (8 dígitos)	a.2	Apellido y nombre (30 caracteres)
a.3	Domicilio (20 caracteres)	a.4	Código postal (4 dígitos)
a.5	Teléfono (10 caracteres)	a.6	Año de ingreso (4 dígitos)

Y otro archivo sin orden que el mencionado llamado "NOVEDADES.dat", con cantidad máxima de registros es 100. Posee las actualizaciones (altas, bajas, y modificaciones) a ser aplicadas, donde cada registro contiene además de todos los campos de Alumnos.dat un código de operación ('A'= Alta, 'B'= Baja, 'M'= Modificación).

Se pide desarrollar todos los pasos necesarios para realizar un programa que genere un archivo actualizado "ALUMACTU.dat" con el mismo diseño.

Restricciones:

1) memoria para arrays 7200 bytes
2) memoria para arrays 900 bytes

Ej. 27 Dado el archivo "PRODUCTOS.dat" con los datos de los productos que vende un mayorista, ordenado por código de producto y con el siguiente diseño:

a.1	Código de producto (8 caracteres)	a.2	Descripción del producto (30 caracteres)
a.3	Cantidad en Stock (4 dígitos)	a.4	Precio de venta (real)
a.5	Estado del registro ('A'=activo / 'I'= inactivo)		

Y otro archivo sin ningún orden llamado "NOVEDADES.dat", con las actualizaciones (altas, bajas, y modificaciones) a ser aplicadas, donde cada registro contiene además de todos los campos de Productos.dat un código de operación ('A'= Alta, 'B'= Baja, 'M'= Modificación).

Se pide desarrollar todos los pasos necesarios para realizar un programa que actualice el archivo "Productos.dat" sobre sí mismo.
NOTA: Los códigos de los productos a dar de alta siempre tendrán un valor superior a los que se encuentren ya grabados en Productos.dat, y no son más de 20.
Restricciones memoria para arrays 100 bytes

Ej. 28: Una biblioteca maneja la siguiente información:

a) un archivo de Libros, ordenado por código de libro y con el siguiente diseño:

a.1	Código del libro (6dígitos)	a.2	título del libro (30 caract)
a.3	cantidad de ejemplares (2 dígitos)	a.4	ubicación (3 caract)
a.5	código de la editorial (1..200)	a.6	autor (25 caract)

b) un archivo de editoriales ordenado alfabéticamente por nombre de la editorial, con el siguiente diseño.

- nombre de la editorial (25 caracteres)
- código de la editorial (1..200)

c) un archivo de consultas realizadas durante el primer semestre del año. Los diferentes libros consultados no superan los 1000, y el diseño del registro es el siguiente:

- código del libro
- fecha de consulta (aaaammdd)

Se pide realizar la metodología necesaria para obtener un programa que:
1) Emitir un listado con los libros que tuvieron como mínimo 20 consultas en cada mes del semestre, con el siguiente diseño, ordenado por código:

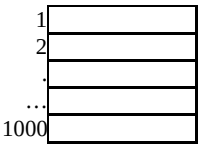
Código del Libro	Título	Autor	Editorial	Consultas en el Semestre					
				1	2	3	4	5	6
999999	xxxxxxxxx	xxxxxxxxx	xxxxxxxxx	999	999	999	999	999	999
999999	xxxxxxxxx	xxxxxxxxx	xxxxxxxxx	999	999	999	999	999	999

2) Grabar un archivo ordenado alfabéticamente por editorial con el siguiente diseño:
2.1 nombre de la editorial (25 caracteres)
2.2 porcentaje de libros que fueron consultados en el semestre (float)

Restricciones: memoria para arrays 32000 bytes

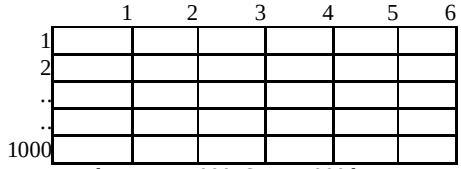
Estructuras de Datos:

Vector de libros consultados



int 4000 bytes

Matríz contadora de consultas por cada mes



short 1000*6*2=12000 bytes

Vector de registro de Editorial

1	NombreEditorial	Sumatoria de consulta semestral
2		
…		
200		
	char[25 + 1] 26 bytes	int 4 bytes

200 * 30 = 6000 bytes

Ej. 29: Una empresa de construcción posee 10 depósitos donde almacena el stock de sus materiales, según una cierta capacidad. Cada vez que realiza compras de materiales o debe enviar estos a las distintas obras en construcción debe almacenar o retirar materiales de los distintos depósitos. Y para ello cuenta con los siguientes archivos:

a) un archivo de direcciones de los 10 depósitos, sin ningún orden y con el siguiente diseño: a.1 número de depósito (1..10) a.2 dirección del depósito (30 caracteres)

b) un archivo de stock de materiales (máximo 100) sin ningún orden con el siguiente diseño:
b.1 código de material (6 dígitos) b.2 número de depósito de almacenamiento
b.3 cantidad en stock en el depósito b.4 capacidad de almacenamiento en el depósito
Un mismo material puede estar en uno, en varios o en todos los depósitos.

c) un archivo de compras realizadas por la empresa con orden natural y con el siguiente diseño
c.1 número de orden de compra (long) c.2 proveedor (20 caract)
c.3 código de material c.4 cantidad comprada (unsigned short)

d) un archivo de pedidos de obras pendientes, con el orden natural y el siguiente diseño:
d.1 número de pedido (long) d.2 dirección de la obra (20 caracteres)
d.3 código de material d.4 cantidad solicitada (unsigned short)

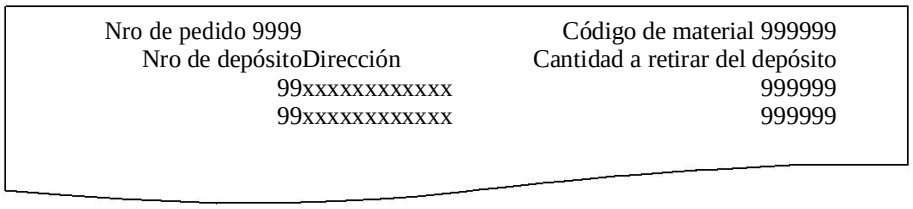
Se pide realizar la metodología necesaria para desarrollar un algoritmo que:

1) Por cada orden de compra distribuir la cantidad comprada entre los distintos depósitos partiendo siempre desde el depósito número 1 y hasta completar la capacidad. Y emitir por cada compra que se pueda almacenar parcial o totalmente el siguiente listado:

Nro de orden de compra 9999 Código de material 999999

Nro de depósito	Dirección	Cantidad a almacenar en depósito
99	xxxxxxxxxxxx	999999
99	xxxxxxxxxxxx	999999

3) Por cada pedido pendiente retirar la cantidad solicitada entre los distintos depósitos partiendo siempre desde el depósito número 1 y hasta agotar stock. Y emitir por cada pedido pendiente que se pueda satisfacer parcial o totalmente el siguiente listado:



3) Actualizar archivos de la siguiente manera:

archivo b) actualizar stock a partir de las compras almacenadas o los pedidos satisfechos, regrabando los registros una sola vez en toda la ejecución del programa

archivo c) actualizar cantidad comprada con un valor 0 si fue almacenado en su totalidad o con la cantidad que no logró almacenarse en ningún depósito..

archivo d): actualizar cantidad solicitada con un valor 0 si fue satisfecho en su totalidad o con la nueva cantidad pendiente. Restricciones: memoria para arrays 4800 bytes