TP 2: Operaciones entre archivos

AGURTO, Gonzalo
EFSTRATIADIS, Carlos Andrés
FINOCCHIO, Fausto
LAREU, Santiago

Organización de Datos
Facultad de Ingeniería
Universidad Nacional de Mar del Plata

Enunciado 1

Se quiere registrar todas las notas de un alumno de una facultad. Para esto se arma un registro con Código de la Universidad, Código de Facultad, Código de la Asignatura, Legajo del Alumno, Edad al momento de rendir, Código del Docente a cargo, Nota y Observaciones.

Enunciado 2

Se quiere registrar todas las compras de un local nacional de electrodomésticos. Para esto se realiza un registro de la siguiente manera: Fecha, Hora, Código de Provincia, Código de Ciudad, Código del Local, Legajo del Vendedor, Factura, Código del Producto, Cantidad, Valor Unitario, Total Renglón y Observaciones.

Enunciado 3

Se tiene un archivo maestro y de novedades (ambos ordenados por ID) con la siguiente estructura: ID E2, Descripción C25, valor F4.

1*) Para los enunciados 1 y 2 realizar las definiciones conceptuales y lógicas, tanto para estructuras de formato fijo como variable.

Aclaración: asumimos que todos los campos que tengan identificación de tipo d son identificadores en otros archivos que contienen mayor información relacionada al campo.

Enunciado 1

Conceptual

```
alumno(
      CodU d,
      CodA d,
      CodF d,
      CodD d,
      Legajo i,
      Edad_al_rendir ?,
      nota +,
      observaciones?,
      baja)
```

Lógico fijo

```
alumno(
      CodU: C4,
      CodA: C4,
      CodF: C4,
      CodD: C4,
      Legajo: E4 ,
      Edad_al_rendir: E1,
      nota: F4,
      observaciones: C255,
      baja: L )
```

Lógico Variable

```
alumno(
      CodU: CV,
      CodA: CV,
      CodF: CV,
      CodD: CV,
      Legajo: E4,
      Edad_al_rendir: E1,
      nota: F4,
      observaciones: T,
)
```

Enunciado 2

Conceptual

```
Compra(
      (fecha)?
      (hora(
            horas,
            minutos
      ))?
      (CodProv)d
      (CodCiudad)d
      (CodLocal)d
      (LegajoVend)*
      (idFactura)i
      (CodProduct)d
      (Cant)*
      (ValorUnitario)*
      (Total)*
      (Obs)?
      (baja)?
)
```

Lógico Fijo

```
Compra(
      fecha: E4,
      hora(
            horas: E1,
            minutos: E1
      ),
      CodProv: C1,
      CodCiudad: C4,
      CodLocal: C4,
      LegajoVend: E4,
      idFactura: E4,
      CodProduct: C5,
```

```
      Cant: E2,
      ValorUnitario: F4
      Total: F4
      Obs: C100
      baja: L
)
```

Lógico Variable

```
Compra(
      fecha: E4,
      hora(
            horas: E1,
            minutos: E1
      ),
      CodProv: CV,
      CodCiudad: CV,
      CodLocal: CV,
      LegajoVend: E4,
      idFactura: E4,
      CodProduct: CV,
      Cant: E2,
      ValorUnitario: F4
      Total: F4
      Obs: T
)
```

4*) Explique las diferencias en los procedimientos realizados en los ejercicios 2 y 3 según tipo de estructura (fija / variable).

Antes de arrancar a explicar, aclaramos que asumimos que los archivos estaban desordenados y que se podía recorrer el archivo completo de ser necesario.

En los archivos de estructura de tipo fija, una baja podría ser marcando directamente en el archivo; un alta podría ser sobreescribiendo un registro bajado o, en caso de no haber bajas, insertar al final; una modificación se podría hacer de forma directa sobre el archivo, y la consulta se hace buscando de a registros.

En cambio, en los archivos de estructura de tipo variable, el algoritmo se complica un poco. Las altas se insertan al final del archivo; las bajas se tendrían que hacer mediante un archivo temporal ya que no siempre coinciden los tamaños de las altas con las de las bajas; las modificaciones también tendrían que hacerse mediante un archivo temporal porque puede variar el tamaño del registro y habría que correr a los que están después, y las consultas se hacen leyendo los registros campo por campo ya que uno no sabe a priori el tamaño del registro y, a su vez, los campos de tamaño variable se tienen que leer byte por byte.

5*) Apareo: Del enunciado 3 realizar la intersección.

Maestro

ID	Descripción	Valor
101	Tornillo cabeza hex	1.50
102	Tuerca M8	0.60
103	Tuerca M10	0.75
104	Arandela plana 8mm	0.20
105	Arandela presión 10mm	0.40
106	Clavo hierro 1.5"	0.08
107	Clavo acero 2"	0.10
108	Destornillador cruz	4.00
109	Pinza universal	7.50
110	Martillo carpintero	8.00

Novedades

ID	Descripción	Valor
102	Tuerca M8 (inox)	0.80
103	Tuerca M10 (galvanizada)	0.95
106	Clavo hierro 1.5"	5.00
107	Clavo acero templado 2"	0.15
111	Serrucho de mano	9.00
112	Sierra manual	12.00
113	Llave francesa pequeña	15.00
114	Taladro eléctrico	55.00

115	**Caja de herramientas plástica**	**25.00**
116	**Cinta métrica 5m**	**5.50**

ACTUALIZACIÓN

ID	Descripción	Valor	Paso a paso (novedades vs maestro)
			Comparación 102 vs 101, sin cambios, avanzo en maestro
102	**Tuerca M8**	**0.80**	Comparación con 102 vs 102, mismo ID, modifico el valor, avanzo en los dos
103	**Tuerca M10**	**0.95**	Comparación con 103 vs 103, mismo ID, modifico el valor, avanzo en los dos
			Comparación 106 vs 104, sin cambios, avanzo en maestro
			Comparación 106 vs 105, sin cambios, avanzo en maestro
106	**Clavo hierro 1.5"**	**5.00**	Comparación con 106 vs 106, mismo ID, modifico el valor, avanzo en los dos
107	**Clavo acero 2"**	**0.15**	Comparación con 107 vs 107, mismo ID, modifico el valor, avanzo en los dos
			No hay más coincidencias entre los archivos

ARCHIVO INTERSECCIÓN

ID	Descripción	Valor
102	**Tuerca M8**	**0.80**
103	**Tuerca M10**	**0.95**
106	**Clavo hierro 1.5"**	**5.00**
107	**Clavo acero 2"**	**0.15**

6*) Apareo: Del enunciado 3 realizar la diferencia.

Usando los archivos maestro y novedades del ejercicio 5:

ACTUALIZACIÓN

ID	Descripción	Valor	Paso a paso (novedades vs maestro)
101	**Tornillo cabeza hex**	**1.50**	Comparación 102 vs 101, se agrega el 101 y se avanza en el maestro
			Comparación con 102 vs 102, mismo ID, no agrego al archivo y avanzo en los dos
			Comparación con 103 vs 103, mismo ID, no agrego al archivo y avanzo en los dos
104	**Arandela plana 8mm**	**0.20**	Comparación 106 vs 104, se agrega el 104 y se avanza en el maestro
105	**Arandela presión 10mm**	**0.40**	Comparación 106 vs 105, se agrega el 105 y se avanza en el maestro
			Comparación con 106 vs 106, mismo ID, no agrego al archivo y avanzo en los dos
			Comparación con 107 vs 107, mismo ID, no agrego al archivo y avanzo en los dos
108	**Destornillador cruz**	**4.00**	Comparación 111 vs 108, se agrega el 108 y se avanza en el maestro
109	**Pinza universal**	**7.50**	Comparación 111 vs 109, se agrega el 109 y se avanza en el maestro
110	**Martillo carpintero**	**8.00**	Comparación 111 vs 110, se agrega el 110 y se avanza en el maestro
111	**Serrucho de mano**	**9.00**	**Archivo maestro no tiene mas ID pero si novedades, se agregan todas las novedades que quedaron**
112	**Sierra manual**	**12.00**	
113	**Llave francesa pequeña**	**15.00**	
114	**Taladro eléctrico**	**55.00**	
115	**Caja de herramientas plástica**	**25.00**	

116	**Cinta métrica 5m**	**5.50**	

Archivo diferencia

ID	Descripción	Valor
101	**Tornillo cabeza hex**	**1.50**
104	**Arandela plana 8mm**	**0.20**
105	**Arandela presión 10mm**	**0.40**
108	**Destornillador cruz**	**4.00**
109	**Pinza universal**	**7.50**
110	**Martillo carpintero**	**8.00**
111	**Serrucho de mano**	**9.00**
112	**Sierra manual**	**12.00**
113	**Llave francesa pequeña**	**15.00**
114	**Taladro eléctrico**	**55.00**
115	**Caja de herramientas plástica**	**25.00**
116	**Cinta métrica 5m**	**5.50**

7*) Apareo: Del enunciado 3 realizar la unión.

Usando los archivos maestro y novedades del ejercicio 5:

ACTUALIZACIÓN

ID	Descripción	Valor	Paso a paso (novedades vs maestro)

101	**Tornillo cabeza hex**	**1.50**	Comparación 102 vs 101, se agrega el 101 y se avanza en el maestro
102	**Tuerca M8**	**0.80**	Comparación con 102 vs 102, mismo ID, se agrega 102 de novedades y se avanza en ambos
103	**Tuerca M10 (galvanizada)**	**0.95**	Comparación con 103 vs 103, mismo ID, se agrega 103 de novedades y se avanza en ambos
104	**Arandela plana 8mm**	**0.20**	Comparación 106 vs 104, se agrega el 104 y se avanza en el maestro
105	**Arandela presión 10mm**	**0.40**	Comparación 106 vs 105, se agrega el 105 y se avanza en el maestro
106	**Clavo hierro 1.5"**	**5.00**	Comparación con 106 vs 106, mismo ID, se agrega 106 de novedades y se avanza en ambos
107	**Clavo acero templado 2"**	**0.15**	Comparación con 107 vs 107, mismo ID, se agrega 107 de novedades y se avanza en ambos
108	**Destornillador cruz**	**4.00**	Comparación 111 vs 108, se agrega el 108 y se avanza en el maestro
109	**Pinza universal**	**7.50**	Comparación 111 vs 109, se agrega el 109 y se avanza en el maestro
110	**Martillo carpintero**	**8.00**	Comparación 111 vs 110, se agrega el 110 y se avanza en el maestro
111	**Serrucho de mano**	**9.00**	**Archivo maestro no tiene mas ID pero si novedades, se agregan todas las novedades que quedaron**
112	**Sierra manual**	**12.00**	
113	**Llave francesa pequeña**	**15.00**	
114	**Taladro eléctrico**	**55.00**	
115	**Caja de herramientas plástica**	**25.00**	
116	**Cinta métrica 5m**	**5.50**	

ARCHIVO UNION

ID	Descripción	Valor
101	Tornillo cabeza hex	1.50
102	Tuerca M8	0.80
103	Tuerca M10 (galvanizada)	0.95
104	Arandela plana 8mm	0.20
105	Arandela presión 10mm	0.40
106	Clavo hierro 1.5"	5.00
107	Clavo acero templado 2"	0.15
108	Destornillador cruz	4.00
109	Pinza universal	7.50
110	Martillo carpintero	8.00
111	Serrucho de mano	9.00
112	Sierra manual	12.00
113	Llave francesa pequeña	15.00
114	Taladro eléctrico	55.00
115	Caja de herramientas plástica	25.00
116	Cinta métrica 5m	5.50

8*) Del enunciado 3, qué cambios realizaría en la estructura del archivo Novedades para que acepte Altas, Bajas y Actualizaciones. Justificar.

Al **archivo Novedades** le incorporaría un **campo adicional** denominado **Operación**, de tipo carácter (C1).
Este campo puede tomar tres valores posibles:

- **'A' → Alta**
- **'B' → Baja**
- **'M' → Modificación**

De esta manera, la estructura del archivo Novedades quedaría:

- **ID** (E2)
- **Descripción** (C25)
- **Valor** (F4)
- **Operación** (C1)

Funcionamiento según el tipo de operación:

- **Alta ('A')**: el registro no existe en el maestro y debe incorporarse al archivo final (nuevo maestro).
- **Baja ('B')**: el registro existe en el maestro y no debe copiarse al archivo final (se elimina).
- **Modificación ('M')**: el registro existe en el maestro y debe actualizarse en el archivo final con los nuevos datos de descripción y/o valor.

La incorporación del campo **Operación** permite:

1. **Diferenciar claramente** entre altas, bajas y modificaciones, evitando ambigüedades.
2. **Mantener una estructura simple y compatible** con el archivo maestro.
3. **Facilitar el procesamiento secuencial**, ya que ambos archivos siguen ordenados por ID.
4. **Evitar interpretaciones erróneas**, como asumir que un valor 0 implica baja.

9) Corte de Control: Del enunciado 2 realizar el informe de ventas por local agrupados por provincia y por ciudad.
Se considera el archivo ordenado por cod_provincia, cod_ciudad, cod_local y que los registros son de longitud fija.

```
typedef struct {
   char fecha[5];
   char hora[8];
   unsigned int cod_provincia;
   unsigned int cod_ciudad;
   unsigned int cod_local;
   unsigned int legajo_vendedor;
   unsigned int factura;
   unsigned int cod_producto;
   unsigned short int cantidad;
   float valor;
   float total_reglon;
   char observaciones[100];
   char baja;
}Treg2;

void corte_control (char* nombre_archivo){
   FILE* archb;
   Treg2 registro;
   unsigned int flag_provincia;
   unsigned int flag_ciudad;

   if( (archb = fopen(nombre_archivo,"rb")) == NULL)
   {
      printf("Error al intentar abrir el archivo.");
   }
   else
   {
      fread(&registro,sizeof(Treg),1,archb);
      while( !feof(archb) )
      {
         flag_provincia = registro.cod_provincia;
         printf("%d\n",flag_provincia);
         while( !feof(archb) && flag_provincia == registro.cod_provincia)
         {
            flag_ciudad = registro.cod_ciudad;
            printf("%d",flag_ciudad);
            while( !feof(archb) && flag_provincia == registro.cod_provincia
            && flag_ciudad == registro.cod_ciudad)
            {
               printf("%d %d %d %d %d %f %s %s %s",registro.cod_local
                     ,registro.legajo_vendedor
                     ,registro.cod_producto
                     ,registro.factura
                     ,registro.cantidad
                     ,registro.valor
                     ,registro.fecha
                     ,registro.hora
                     ,registro.observaciones
                  );
               fread(&registro,sizeof(Treg),1,archb);
            }
         }

      }
```

```
        fclose(archb);
    }
}
```

10*) Corte de Control: Del enunciado 1 realizar el informe de promedio de notas, agrupado por alumnos y facultad.

El pseudocódigo presente muestra cómo se implementa y funcionaria el corte de control para determinar el informe pedido.

Se supone que el archivo está ordenado de manera ascendente por código de facultad y por legajo del alumno.

```
void informe(){
      variable de tipo String para tener el código de la facultad (CodFacu);
      variable de tipo Int para tener el legajo del alumno (Legajo);
      variable de tipo Registro para leer el archivo con los registros
(Reg);
      variable de tipo int para contar cantidad de alumnos por facultad
(CantALum)
      variable de tipo int para contar cantidad de notas por alumno
(CantNotas)
      variable de tipo float para almacenar la suma de las notas (Notas)
      variable de tipo float para almacenar el promedio de notas del alumno
(PromAlum)
      variable de tipo float para almacenar la suma de los promedios de cada
alumno (PromFacu)

      abrir archivo;
      if (archivo abierto){
            se lee el primer registro del archivo y se almacena en Reg;
            while (no termina lectura del archivo){
                  en CodFacu almacenamos Reg.CodFacu, es decir el codigo de
                  la facultad leida;
                  printf(Codigo de la facultad);
                  CantAlum y PromFacu los inicializamos con el valor 0;
                  while (no termina lectura del archivo y codFacu sea igual
                  a Reg.CodFacu){
                        en legajo almacenamos Reg.Legajo, es decir el legajo
                        del alumno leido del archivo;
                        CantAlum++;
                        inicializamos con el valor 0 a CantNotas y a Notas;
                        while (no termina lectura del archivo y legajo sea
                        igual a Reg.Legajo){
                              CantNotas++;
                              Notas += Reg.Nota;
                              leemos siguiente registro y lo almacenamos en
                              Reg;
```

```
                        }
                        chequeamos que CantNotas sea distinto de cero y
                        hacemos
                        PromAlum=Notas/CantNotas;
                        printf(Legajo del alumno y PromAlum);
                        PromFacu+=PromAlum;
                  }
                  chequeamos no dividir por el valor 0;
                  PromFacu = PromFacu/CantAlum;
                  printf(PromFacu);
                  //aquí mostramos el promedio de notas de la facultad
            }
            cerramos archivo;
      }
      // Fin del corte de control

}
```

11*) Busque al menos otros tres métodos de ordenamientos para archivos. Descríbalos y arme el pseudocódigo de los mismos. Confecciones un cuadro comparativo.

Algoritmo: Bubble sort

Método: Selección

Complejidad: $O(n^2)$

Se compara con cada par de elementos adyacentes, si un elemento es mayor al siguiente, se intercambian. Al final de cada pasada, el elemento más grande queda al final. Este proceso se repite para los elementos restantes hasta que quede totalmente ordenado.

```
para i = 1 hasta N hacer

  para j = 1 hasta N-1 hacer

      si dato[j] > dato[j+1] entonces

      aux = dato[j]

      dato[j] = dato[j+1]

      dato[j+1] = aux;

      fin si

   fin para

fin para
```

Algoritmo: Insertion sort

Metodo; inserción

Complejidad: $O(n^2)$ o menor

Comienza con el primer elemento, toma el siguiente elemento y los compara e inserta en la posición correcta dentro de los ordenados, moviendo los elementos más grandes hacia la derecha. Este proceso se repite hasta que quede totalmente ordenado.

```
para i = 2 hasta N hacer
   elemento = dato[i]
   j = i - 1
   mientras j >= 1 y dato[j] > elemento hacer
      dato[ j + 1] = dato[j]
      j = j - 1
  fin mientras
  dato[ j + 1] = elemento
fin para
```

Algoritmo: Selection sort

Metodo: seleccion

Complejidad: $O(n^2)$

Selecciona el elemento menor y lo posiciona en su lugar correcto, al final de la parte ordenada. Este proceso se repite hasta que esté totalmente ordenado.

```
para i = 1 hasta N hacer
```

```
   min = i
   para j = i + 1 hasta N hacer
      si dato[j] < dato[min] entonces
      min = j
      fin si
  fin para
  si min != i entonces
      aux = dato[i]
      dato[i] = dato[min]
    dato[min] = aux
  fin si
fin para
```

Algoritmo	Método	Complejidad	Ventajas	Desventajas
Bubble Sort	Intercambio	O(n²)	Muy fácil de implementar	Muy lento en listas grandes
Insertion Sort	Inserción	O(n²), mejor en casi ordenados	Bueno para listas pequeñas y casi ordenadas	No eficiente en grandes volúmenes
Selection Sort	Selección	O(n²)	Poco movimiento de datos, simple	Siempre O(n²), aunque esté casi ordenado