PROGRAMACION ESTRUCTURADA BASICA

UNIDAD 4. CORTE DE CONTROL

INDICE

UNIDAD 4
CORTE DE CONTROL

OBJETIVOS: Procesar lotes de datos ordenados a fin de obtener reportes o estadísticas. Afianzar el manejo de estructuras repetitivas, contadores y acumuladores.

1. ¿Qué es el corte de control?

El corte de control es un proceso en el cual partiendo de registros ordenados por el valor de uno o más campos se los procesa para obtener información agrupada en lotes y sub-lotes al detectar un cambio en la/las variables por la cual la información se encuentra organizada.

En otras palabras, se procesa un conjunto ordenado de registros en subconjuntos determinados por el cambio de valor de una o más variables.

El corte de control se aplica para obtener información detallada agrupada por uno o más criterios.

Dicha información puede mostrarse en forma de reporte a medida que se detecta el cambio en la variable por la cual la información se encuentra ordenada o puede grabarse un archivo resumen con dicha información.

La condición necesaria para aplicar corte de control es que la información esté ordenada u organizada por el campo sobre el cual queremos agrupar la información, es decir que todos los registros de un mismo tipo deben estar juntos.

El corte de control puede aplicarse en varios niveles siempre y cuando la información venga ordenada por todos los campos sobre los cuales se desea agrupar. Por ejemplo, si se desea obtener información de los ingresos a un establecimiento agrupada por día y en cada día se desea obtener un detalle por hora, los datos deben estar primero ordenados por día y dentro de cada día ordenados por hora. En este caso podría aplicarse dos cortes uno por día y otro por hora a fin de obtener información agrupada en cada uno de ellos. Pero no siempre que la información esté ordenada significa que se deba aplicar corte de control. En el caso anterior si solo se desea obtener información detallada por día, entonces se hará un corte de un solo nivel ya que agrupar también por hora no aportaría información adicional.

Para aplicar corte de control si la información se ingresa por teclado debe ingresarse en forma ordenada, es por ello por lo que lo más común es procesar información ya ordenada que venga en un vector o en un archivo.

2. Reglas Generales para aplicar corte de control

El corte de control se realiza anidando distintas estructuras repetitivas con las siguientes reglas:

- Se realiza un ciclo repetitivo con la condición de fin de datos
- Dentro del ciclo de fin de datos se anida un ciclo por cada nivel de corte que se desea realizar (el ciclo de cada nivel va anidado dentro del ciclo del nivel anterior)

- Las condiciones de los ciclos se van propagando, es decir que el ciclo interno de corte va a incluir la condición de fin del ciclo que lo contiene y su propia condición. Si hay otros niveles de corte va a incluir las condiciones anteriores y su propia condición de fin.

- La información se lee antes de ingresar al ciclo de la condición de fin y se vuelve a leer antes de finalizar el ciclo más interno.

- Las variables que cuentan o acumulan información agrupada deben inicializar antes de comenzar el ciclo del corte y mostrarse al salir de dicho ciclo haciendo referencia a que dicha información corresponde al dato leído anteriormente y no al último, ya que el ciclo termina por que el lote finalizó y por lo tanto la lectura actual va a ser diferente.

3. Cómo aplicar Corte de Control en diferentes casos.

Si bien esta técnica puede aplicarse a datos ingresados por teclado, para que funcione correctamente sería necesario que los datos se carguen en orden sino daría información incorrecta o segmentada.

Por este motivo se verá directamente un ejemplo de aplicación sobre un archivo en el cual los datos fueron almacenados de forma ordenada permitiendo la aplicación de este algoritmo. Cabe aclarar que cuando se habla de datos ordenados para poder aplicar corte de control no es necesarios que los datos estén ordenados en orden ascendente o descendente por el campo sobre el cual se quiere aplicar el corte sino que simplemente se requiere que todos los registros de dicho campo vengan juntos en el archivo.

3.1 Un Nivel de corte desde un archivo

En la Empresa “M. y M.” se quiere saber cuánto se abona de sueldo por cada Sector de Fábrica y en toda la Empresa. Los datos están ordenados por el campo clave “Número de Sector de Fábrica”, para lo cual se han analizado los datos que se encuentra en el archivo llamado “empleados.dat” y los registros del archivo tiene el siguiente formato:

a. Número de Legajo (entero)

b. Apellido y Nombre (hasta un máximo de 25 letras)

c. Año de Ingreso a la empresa (entero)

d. Número de Sector de Fábrica (entero)

e. Sueldo (real)

Para comprender el algoritmo vamos a analizar un conjunto de registros de prueba:

Legajo	Nombre	Ingreso	Sector	Sueldo
1132	Andrea Lorenzo	2001	5000	89000.00
1932	Julia Correa	2011	5000	69000.00
1423	Jorge Perez	2005	5000	79000.00
1032	Pablo Caceres	1998	7000	110000.00
1002	Julia Sarratea	1995	7000	135000.00
1133	Sandra Zarrea	2001	8000	89000.00
1145	Fernando Formento	2002	9000	49000.00
1155	Luisa Perez	2005	9000	69000.00

Como puede notarse los datos en este archivo se encuentran ordenados por Sector y lo que se busca es obtener información de cuantos empleados trabajan en cada sector.

Al estar ordenados es posible ir contando registros mientras el código de sector sea el mismo que el anterior y en momento que se lea un código de sector diferente signfica que el subconjunto de datos finalizó y por lo tanto podemos mostrar la información calculada.

Legajo	Nombre	Ingreso	Sector	Sueldo
1132	Andrea Lorenzo	2001	5000	89000.00
1932	Julia Correa	2011	5000	69000.00
1423	Jorge Perez	2005	5000	79000.00
1032	Pablo Caceres	1998	7000	110000.00
1002	Julia Sarratea	1995	7000	135000.00
1133	Sandra Zarrea	2001	8000	89000.00
1145	Fernando Formento	2002	9000	49000.00
1155	Luisa Perez	2005	9000	69000.00

Se vienen leyendo los registros uno a uno procesando los datos de cada uno

Al leer este registro se detecta el cambio de código y por lo tanto se terminó de procesar un sector

Volviendo al ejemplo hay 4 sectores distintos en el primer sector con código 5000 hay 3 empleados, en el segundo sector con código 7000 hay 2 empleados, luego un empeado para el sector 8000 y 2 en el 9000

Legajo	Nombre	Ingreso	Sector	Sueldo
1132	Andrea Lorenzo	2001	5000	89000.00
1932	Julia Correa	2011	5000	69000.00
1423	Jorge Perez	2005	5000	79000.00
1032	Pablo Caceres	1998	7000	110000.00
1002	Julia Sarratea	1995	7000	135000.00
1133	Sandra Zarrea	2001	8000	89000.00
1145	Fernando Formento	2002	9000	49000.00
1155	Luisa Perez	2005	9000	69000.00

Sector 5000

Sector 7000

Sector 8000

Sector 9000

Resolución del problema:

A continuación, se muestra el código que resuelve el problema con comentarios en cada sección:

```c
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

struct PERSONA
{
    int nLegajo;
    char ApyN [26];
    int nAnio;
    int nSecFab;
    float rSueldo;
};
int main (  )
{
    struct PERSONA per;
    int antSecFab, contTotal, contSector;
    FILE *archEmpleados;

    archEmpleados = fopen("empleado.dat", "rb");
    if (archEmpleados == NULL)
    {
        printf("\nNo se puede abrir.");
        getch();
        exit(1);
    }

    contTotal = 0;
    fread(&per, sizeof(struct PERSONA ),1,archEmpleados);
    while (!feof(archEmpleados))
    {
        antSecFab = per.nSecFab;
        contSector = 0;

        do
        {
            contSector ++;
            fread(&per, sizeof(struct PERSONA ),1,archEmpleados);
        }while (per.nSecFab == antSecFab && !feof(archEmpleados));

        contTotal += contSector;
        printf("\nLa Cantidad de Empleados del Sector %d es %d",\
                antSecFab, contSector);
    }
    fclose(archEmpleados);
    printf("\nLa Cantidad de Empleados que tiene la empresa es %d"\
                , contTotal);
    return 0;
}
```

Se define la estructura con el formato del registro del archivo a procesar

Se declaran las variables a utilizar y se abre el archivo en modo lectura

Antes de ciclo se inicializan los contadores y acumuladores generales (para todos los datos)

Se lee del archivo se arma el ciclo principal del proceso hasta fin de archivo

Se guarda el código anterior y se inicializan los contadores y acumuladores parciales (por corte)

Dentro del ciclo se usan los datos leídos en este caso simplemente se cuenta y se hace la segunda lectura para pasar al siguiente registro

Ciclo del corte de control se repite mientras sea el mismo código que el anterior y no se llegue al final del archivo

Se acumula en el contador general

Al salir del ciclo del corte se muestran los datos acumulados (del código anterior)

Al salir del while que indica que no hay más registros se cierra el archivo y se muestran los datos acumulados en forma general

Al ejecutar el programa con el lote de prueba planteado el resultado es el siguiente:

```
La Cantidad de Empleados del Sector 5000 es 3
La Cantidad de Empleados del Sector 7000 es 2
La Cantidad de Empleados del Sector 8000 es 1
La Cantidad de Empleados del Sector 9000 es 2
La Cantidad de Empleados que tiene la empresa es 8
```

Puede verse que se muestra un informe de la cantidad de empleados por sector, este informe se hace al salir del ciclo del corte ya que se trata de un contador parcial. Luego al finalizar el recorrido de todo el archivo se muestra el contador total de cantidad de empleados de toda la empresa.

Aclaración: el ciclo del corte de control en este ejemplo fue realizado con un ciclo do while pero también puede realizarse con un ciclo while con la salvedad que se sabe que la primera vez va a ingresar siempre. En el siguiente ejemplo el ciclo de corte fue realizado con un ciclo while.

3.2 Un Nivel de corte desde un archivo generando archivo resumen

Se dispone de un archivo llamado “ventas.dat” que tiene las ventas realizadas por una empresa a lo largo del mes. La empresa tiene distintas sucursales a lo largo del país. El archivo contiene registros con los siguientes datos:

- Codigo de sucursal (entero)
- Codigo de producto (entero)
- Cantidad Vendida (entero)
- Precio Unitario (real)

Para una misma sucursal se recibe un solo registro por cada producto vendido.

Se desea realizar un programa que calcule las ventas realizadas en cada sucursal generando un archivo llamado ventasXsuc.dat que contenga un registro sumarizado por cada sucursal con los siguientes datos:

- Codigo de sucursal (entero)
- Cantidad total de productos vendidos
- Importe total recaudado
- Codigo de producto con mayor cantidad de unidades vendidas (considerar único)

Resolución del problema:

Para resolver este problema será necesario trabajar con los dos archivos abiertos, el archivo ventas.dat se abrirá en modo lectura para ir leyendo registro a registro y hacer el corte de control sobre el campo código de sucursal. Al mismo tiempo se debe crear el archivo ventasXsuc.dat para que cada vez que se detecte un cambio en el código de sucursal se grabé un archivo sumarizado en dicho archivo.

Además de dos campos sumarizados (cantidad e importe), se nos pide determinar el código de producto con mayor cantidad de unidades vendidas por lo que se deberá aplicar el algoritmo para calcular el máximo para los productos vendidos en cada sucursal.

Se deben declarar dos estructuras de datos una por cada archivo ya que almacenan registros distintos.

Estructura **sVenta**			
codSuc	codProd	cantVend	precioU

Estructura **sResumen**			
codSuc	totalVend	recaudacion	codProdMaxVtas

Nótese que este programa no tiene salida por pantalla los resulados se guardan directamente en un arhivo.

Codificación en Lenguaje C:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

struct sVenta
{
    int codSuc;
    int codProd;
    int cantVend;
    float precioU;
};


struct sResumen
{
    int codSuc;
    int totalVend;
    float recaudacion;
    int codProdMaxVtas;
};

int main()
{
    FILE * arch_Vta, * arch_VtaXSuc;
    struct sVenta vta;
    struct sResumen resumen;
    int primero, max;

    arch_Vta = fopen("ventas.dat", "rb");
    arch_VtaXSuc = fopen("ventasxsuc.dat", "wb");
    if (arch_Vta==NULL || arch_VtaXSuc==NULL)
    {
        printf("Error al abrir los archivos");
        getch();
        exit(1);
    }

    fread(&vta, sizeof(vta), 1, arch_Vta);
    while (!feof(arch_Vta))
    {
        /*guarda la sucursal anterior e inicializa los acumuladores
        y contadores directamente en la variable del tipo estructura
        que se va a grabar el archivo por sucursal*/

        resumen.codSuc = vta.codSuc;
        resumen.totalVend=0;
        resumen.recaudacion =0;
        primero = 1; //para el maximo
```

```
        while (!feof(arch_Vta) && resumen.codSuc == vta.codSuc)
        {
            resumen.totalVend += vta.cantVend;
            resumen.recaudacion += vta.cantVend * vta.precioU;

            if(primero || vta.cantVend > max)
            {
                max = vta.cantVend;
                resumen.codProdMaxVtas = vta.codProd;
                primero = 0;
            }
            fread(&vta, sizeof(vta), 1, arch_Vta);
        }
        fwrite (&resumen, sizeof( resumen),1, arch_VtaXSuc);
    }
    fclose(arch_Vta);
    fclose(arch_VtaXSuc);
}
```

3.3 Un Nivel de corte desde un archivo generando archivos dinámicamente

El sistema de control de ingreso y salida de empleados de una fábrica deja al final del mes un archivo ordenado por sector y sumarizado por empleado llamado horasTrabajadas.dat con los siguientes datos:

- Sector (texto de 10 caracteres máximo)
- DNI del empleado (entero)
- Horas trabajadas

Por otro lado se dispone del archivo empleados.dat con los empleados de la fábrica que contiene los siguientes datos:

- DNI del empleado (entero)
- Nombre y Apellido (texto de 30 caracteres máximo)
- Valor que cobra por hora (float)

No se sabe la cantidad exacta de empleados pero sí se sabe que no son más de 100.
Se desea realizar un programa que utilizando ambos archivos genere para cada sector un archivo que contenga el detalle de sueldos que deba pagar a sus empleados. Se debe generar para cada sector un archivo distinto cuyo nombre sea nombreSector.dat y debe contener registros con la siguiente información:

- DNI del empleado (entero)
- Nombre y Apellido (texto de 30 caracteres máximo)
- Horas trabajadas (entero)
- Sueldo a Pagar (float)

Resolución del problema:

Inicialmente se deben descargar en memoria los datos los empleados de la empresa, que como se sabe que no son más de 100 se pueden descargar en un vector.

Luego se procesa el archivo horasTrabajadas.dat que está ordenado por sector. Por cada sector se debe generar dinámicamente un nuevo archivo con el nombre dicho sector y calcular en él la información solicitada pare ello se debe buscar el empleado en el vector en memoria para obtener su nombre y valor que cobrar por hora.

Se necesitan tres estructuras una por cada archivo (en realidad para los sectores se crean varios archivos, pero todos con la misma estructura):

Estructura **est_horas**		
sector	dni	horas

Estructura **est_empleado**		
dni	AyN	valorHora

Estructura **est_sector**			
dni	AyN	totalHoras	sueldo

Codificación en Lenguaje C:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{
    char sector[11];
    int dni;
    int horas;
} est_horas;

typedef struct
{
    int dni;
    char AyN[31];
    float valorHora;
} est_empleado;

typedef struct
{
    int dni;
    char AyN[31];
    int totalHoras;
    float sueldo;
} est_sector;

int CargaEmpleados (est_empleado[],int);
int BuscarEmpleado (est_empleado[],int,int);

int main()
{
    FILE * pHoras, * pSector;
    int cantEmpleados, pos;
    est_empleado vEmp[100];
    est_horas regHora;
    est_sector regSector;
    char secAnt[11], nombreArchivoSector[15];

    cantEmpleados = CargaEmpleados(vEmp, 100);

    pHoras = fopen("horasTrabajadas.dat", "rb");
    if (pHoras == NULL)
```

```
    {
        printf ("No se pudo abrir el archivo de horas trabajadas");
        getch();
        exit (1);
    }

    fread(&regHora, sizeof(regHora), 1, pHoras);
    while (!feof(pHoras))
    {
        strcpy(secAnt, regHora.sector);
        /*Se genera el nombre del archivo basado en el nombre del sector */
        strcpy (nombreArchivoSector, regHora.sector);
        strcat(nombreArchivoSector, ".dat");
        pSector = fopen (nombreArchivoSector, "wb");
        if (pSector ==NULL)
        {
            printf ("No se pudo crear el archivo para el sector %s", secAnt);
            getch();
            exit(1);
        }

        while (!feof(pHoras) && strcmp(secAnt, regHora.sector)==0)
        {
            /*se busca el empleado y como se trabaja con archivos ya creados
            se asume que siempre lo encuentra*/

            pos = BuscarEmpleado(vEmp,regHora.dni, cantEmpleados);
            regSector.dni = regHora.dni;
            strcpy(regSector.AyN, vEmp[pos].AyN);
            regSector.totalHoras = regHora.horas;
            regSector.sueldo = regHora.horas * vEmp[pos].valorHora;
            fwrite(&regSector, sizeof(regSector),1 , pSector);

            fread(&regHora, sizeof(regHora), 1, pHoras);
        }

        fclose(pSector);
    }
    fclose(pHoras);
}

int CargaEmpleados (est_empleado ve[],int max)
{
    FILE *pEmpleados;
    int i=0;
    est_empleado regEmpleado;
    pEmpleados = fopen("empleados.dat", "rb");
    if (pEmpleados == NULL)
    {
        printf ("No se pudo abrir el archivo de empleados.");
        getch();
        exit (1);
    }

    fread(&regEmpleado, sizeof(regEmpleado),1, pEmpleados);
    while (!feof(pEmpleados) && i < max)
    {
        ve[i] = regEmpleado;
        i++;
        fread(&regEmpleado, sizeof(regEmpleado),1, pEmpleados);
    }
    fclose(pEmpleados);
    return i;
}

int BuscarEmpleado (est_empleado ve[],int dni,int tam)
{
    int i =0 , pos =-1;
```

```
    while (pos==-1 && i<tam)
    {
        if (ve[i].dni == dni)
            pos = i;
        else
            i++;
    }
    return pos;
}
```

3.4 Dos Niveles de corte desde un archivo

El corte de control puede aplicarse en distintos niveles si es necesario, pero para ello cada campo por el cual se quiera aplicar el corte debe estar ordenado. En el caso de un corte de dos niveles los datos vendrán ordenados por el campo por el cual se hará el primer nivel de corte y luego dentro de ese subconjunto de datos vendrán ordenados por un segundo campo. Se recuerda que cada nivel de corte agregará un ciclo para esperar que cambien los datos anidados dentro del ciclo anterior y propagando la condición de los anteriores. La segunda lectura del archivo se debe realizar el ciclo más interno.

A modo de ejemplo se presenta el siguiente problema:

Se dispone de un archivo binario llamado inscripciones.dat que contiene la información de los inscriptos a la universidad. El formato de registro es el siguiente:

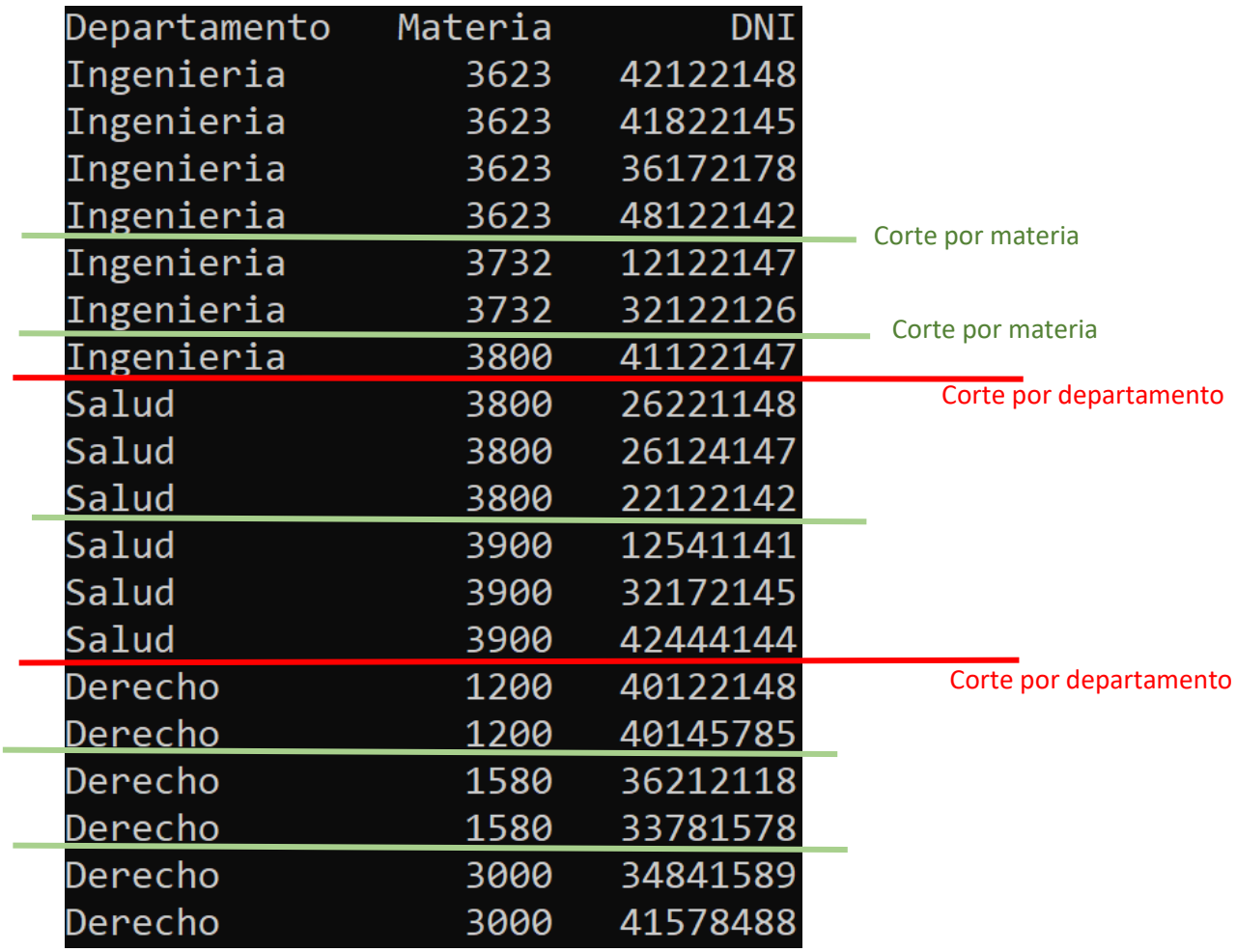
- Departamento (texto de 15 caracteres máximo)
- Código de materia (entero)
- Dni del alumno inscripto

Los datos en este archivo se encuentran ordenados por departamento y dentro de cada departamento por código de materia.

Realizar un programa que indique la cantidad de inscriptos a cada materia, la cantidad de inscriptos a cada departamento y la cantidad de inscriptos en total a toda la universidad. Además indicar la cantidad de materias por departamento que tuvieron inscripciones.

Resolución:

Para realizar el análisis se plantea el siguiente archivo de pruebas donde puede verse que los datos están organizados por departamento y luego dentro de cada departamento por materia. Nótese que incluso los códigos de materia pueden repetirse en entre departamentos, pero se tratan de materias distintas y el proceso realizado debe poder diferenciarlas. En los datos planteados la materia con código 3800 se encuentra tanto en ingeniería como en Salud.



Al tener dos niveles de corte hay que analizar cada uno de los informes solicitados y determinar si se tratan de información a nivel general (para todo el arhivo), por departamento (sobre el primer nivel de corte) o por materia (segundo nivel de corte). De ello dependerá donde se inicializaran los contadores, donde se incrementan y donde se muestran.

Los informes solicitados son:

- la cantidad de inscriptos a cada materia: para este cálculo se requiere el segundo nivel de corte
- la cantidad de inscriptos a cada departamento: para este cálculo se requiere el primer nivel de corte
- la cantidad de inscriptos en total a toda la universidad: este es un contador general para todos los datos del archivo
- la cantidad de materias por departamento que tuvieron inscripciones: este calculo si bien es por departamento (primer nivel de corte) necesita saber la cantidad de materias diferentes y por lo tanto aprovechará el segundo nivel de corte para no contar materias repetidas.

Código:

```c
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct
{
    char dpto[16];
    int materia;
    int dni;
}inscripcion;

int main()
{
    FILE * archInsc;
    char dptoAnterior[16];
    int materiaAnterior, total, totalDpto, totalMateria, cantMaterias;
    inscripcion inscripto;
    archInsc = fopen("inscriptos.dat","rb");
    if (archInsc==NULL)
    {
        printf("Error en el archivo inscriptos.");
        getch();
        exit(1);
    }

   total =0; //inicializadon del contador general
   fread(&inscripto, sizeof(inscripcion),1, archInsc);
   while (!feof(archInsc))
   {
       printf("----- Departamento %s------\n",inscripto.dpto);
       strcpy (dptoAnterior, inscripto.dpto);
       totalDpto=0;  //contadores por departamento
       cantMaterias=0;
       do  //ciclo del corte por departamento
       {
           materiaAnterior = inscripto.materia;
           totalMateria=0; //contador por materia

           do //ciclo del corte por materia
           {
               totalMateria++;
               fread(&inscripto, sizeof(inscripcion),1, archInsc);
           }while(!feof(archInsc) \
                   && strcmpi(dptoAnterior, inscripto.dpto)==0  \
                   && materiaAnterior == inscripto.materia);

           printf("\nMateria %d Inscriptos %d",materiaAnterior, totalMateria);
           totalDpto+=totalMateria;
           cantMaterias++;
       }while(!feof(archInsc) && strcmpi(dptoAnterior, inscripto.dpto)==0);

       printf("\n\nTotal Inscriptos al departamento: %d", totalDpto);
       printf("\nCantidad de materias: %d\n\n", cantMaterias);
       total +=totalDpto;
   }
   printf("---------------------------------\n");
   printf("Total Inscriptos a la Universidad: %d\n", total);

   fclose(archInsc);
   return 0;
}
```

Resultado de la ejecución:

```
----- Departamento Ingenieria------

Materia 3623 Inscriptos 4
Materia 3732 Inscriptos 2
Materia 3800 Inscriptos 1

Total Inscriptos al departamento: 7
Cantidad de materias: 3

----- Departamento Salud------

Materia 3800 Inscriptos 3
Materia 3900 Inscriptos 3

Total Inscriptos al departamento: 6
Cantidad de materias: 2

----- Departamento Derecho------

Materia 1200 Inscriptos 2
Materia 1580 Inscriptos 2
Materia 3000 Inscriptos 2

Total Inscriptos al departamento: 6
Cantidad de materias: 3

--------------------------------
Total Inscriptos a la Universidad: 19
```