Cap. 11 – Procesamiento de Archivos

Cap. 11 – Procesamiento de Archivos

Esquema

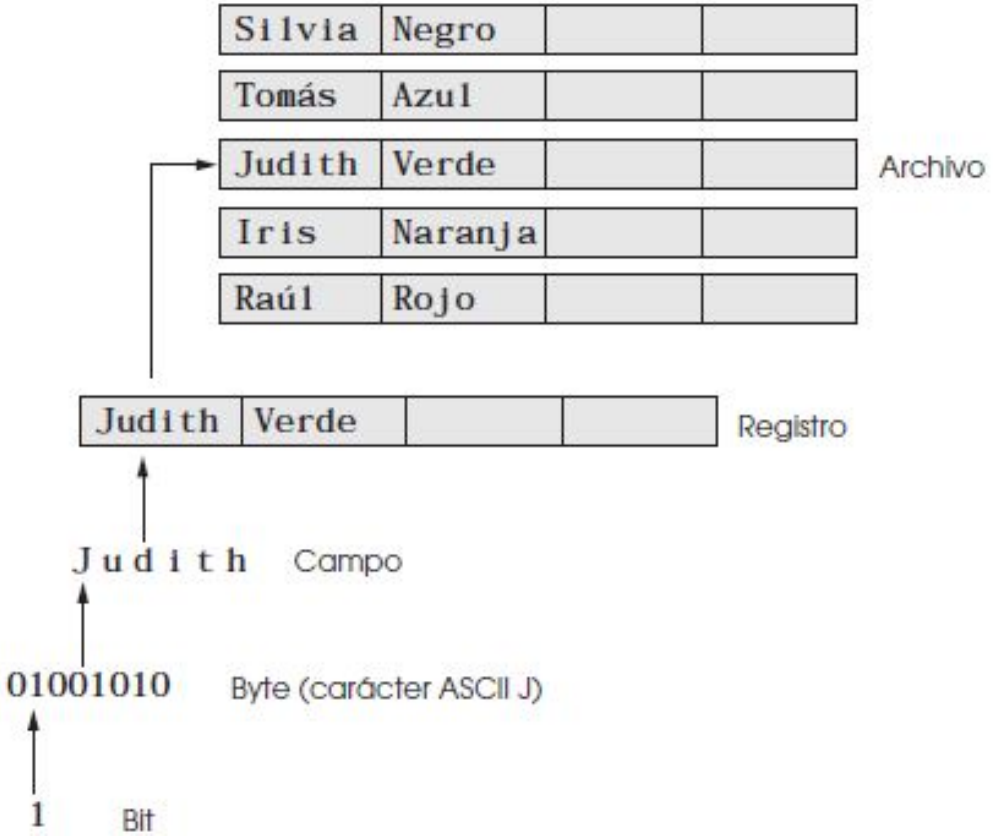
11.1 Introducción

- Los archivos de datos pueden ser creados, actualizados y procesados por programas C
  - Los archivos se utilizan para el almacenamiento permanente de grandes cantidades de datos
  - El almacenamiento de datos en variables y arreglo es sólo temporal

11.2 La Jerarquía de Datos

- Bit - el elemento más pequeño
  - Valor de `0` o `1`
- Byte – 8 bits
  - Utilizado para almacenar un carácter
    - Digitos Decimales, letras, y símbolos especiales
- Campo- grupo de caracteres que transmiten un significado
  - Ejemplo: tu nombre
- Registro – grupo de campos relacionados
  - Representa un `struct` or u `class`
  - Ejemplo: En un sistema de nóminas, un registro de un empleado en particular que contiene su número de identificación, nombre, dirección, etc.
- Archivo – grupo de registros relacionados
  - Ejemplo: archivo de la nómina de pago
- Database – group of related files

11.2 La Jerarquía de Datos (II)

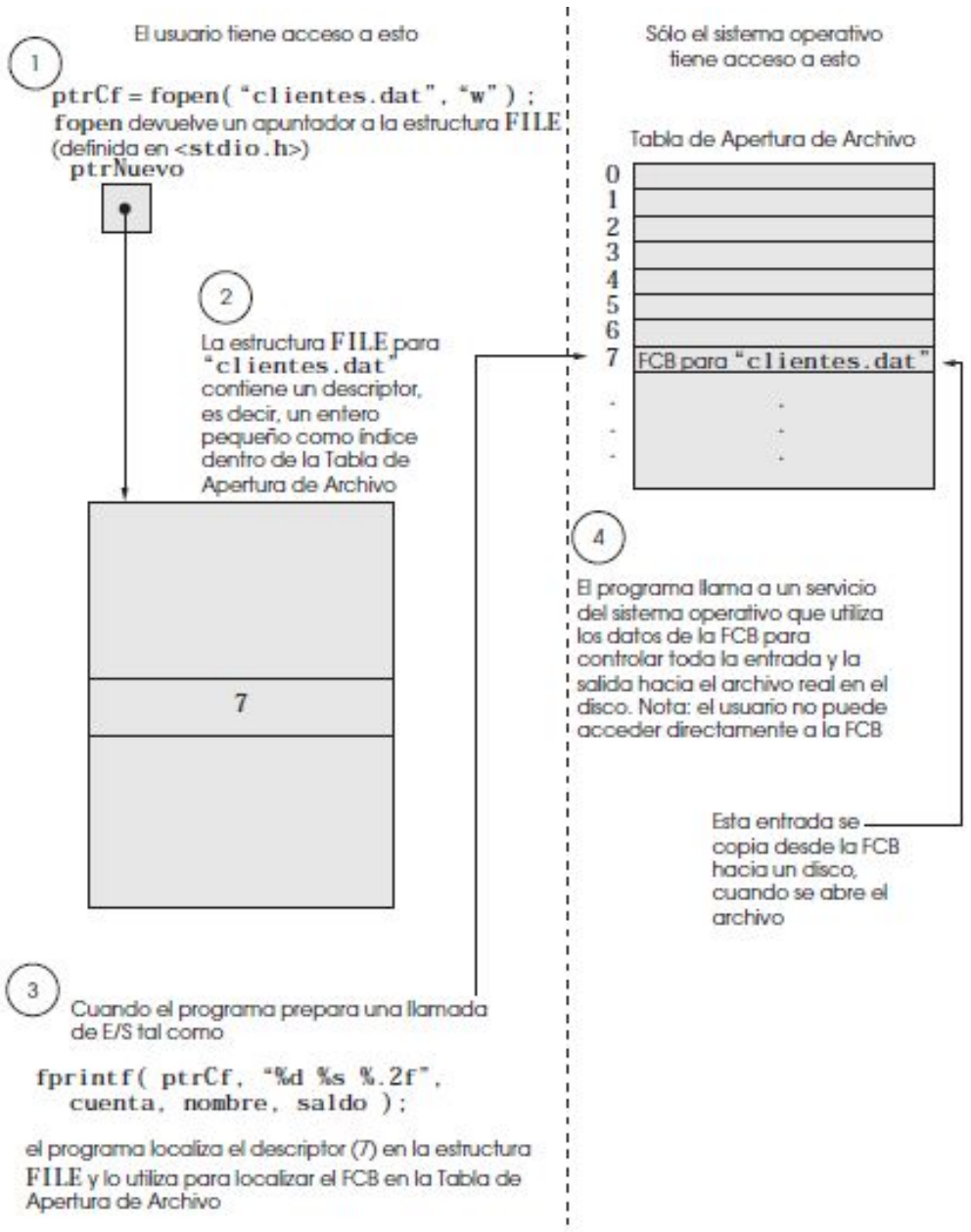


- Clave de Registro
  - Identifica un registro para facilitar la recuperación de registros específicos de un archivo
- Archivo Secuencial
  - Los registros típicamente se ordenan por clave

11.3 Archivos y Cadenas

- C ve cada archivo como una secuencia de bytes
  - Los archivos terminan con el *end-of-file marker*
    - O, los archivos termina a un byte específico
- La cadena creada cuando un archivo es abierto
  - Proporcionar un canal de comunicación entre los archivos y los programas
  - La apertura de un archivo devuelve un puntero a una estructura de "ARCHIVO"
    - Ejemplo de punteros archivos:
    - **`stdin`** - standard input (teclado)
    - **`stdout`** - standard output (monitor)
    - **`stderr`** - standard error (monitor)
- **`FILE:`** estructura
  - Archivo descriptor - Índice en el arreglo del sistema operativo llamada la tabla de archivos abiertos
  - File Control Block (FCB) - Se encuentra en cada elemento del arreglo, el sistema lo utiliza para administrar el archivo

11.3 Archivos y Cadenas (II)



11.3 Archivos y Cadenas (III)

- Funciones de lectura/escritura en la librería estándar
  - **`fgetc`** - lee un carácter del archivo
    - Toma un puntero **`FILE`** como un argumento
    - **`fgetc( stdin )`** equivalente a **`getchar()!!!`**
  - **`fputc`** - escrite un caracter al archivo
    - Toma un puntero a **`FILE`** y un carácter a escribir como argumento argument
    - **`fputc( 'a', stdout )`** equivalente a **`putchar( 'a' )!!!`**
  - **`fgets`** - lee una línea de un archivo
  - **`fputs`** - escribe una línea a un archivo
  - **`fscanf / fprintf`** - procesamiento de archivo equivalente a **`scanf`** y **`printf`**

11.4 Creación de un archivo de acceso secuencial

- C no impone una estructura a archivo
  - Ideas de registro en archivo no existen
  - Los programadores deben proveer estructura al archivo
- Creando un Archivo (File)
  - `FILE *miPtr;` - create un puntero **FILE**
  - `miPtr = fopen("miArchivo.dat", modoApertura);`
    - La función **fopen** retorna un puntero **FILE** a un archivo especificado
    - Toma dos argumentos - archivo para abrir y modo de apertura de archivo
    - Si no es abierto, es retornado **NULL**
  - `fprintf` - como `printf`, excepto el primer argumento es un puntero **FILE** (el archivo que recibe los datos)
  - `feof(FILE pointer)` - retorna `true` si indicador de fin de archivo es puesto para el archivo especificado

11.4 Creación de un archivo de acceso secuencial (II)

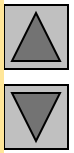
- `fclose(FILE pointer)` - cierra un archivo especificado
  - Se realiza automáticamente cuando el programa termina
  - Buena práctica para cerrar los archivos de forma explícita

- Detalles
  - Los programas podrían procesar ningún archivos, un archivo, o muchos archivos
  - Cada archivo debe tener un único nombre y debería tener un puntero diferente
    - Todo el procesamiento de archivos debe referirse al archivo usando el puntero

11.4 Creación de un archivo de acceso secuencial (III)

Modo	Descripción
`r`	Abre un archivo para lectura.
`w`	Crea un archivo para escritura. Si el archivo ya existe, descarta el contenido actual.
`a`	Agrega; abre o crea un archivo para escritura al final del archivo.
`r+`	Abre un archivo para actualización (lectura y escritura).
`w+`	Crea un archivo para actualización. Si el archivo ya existe, descarta el contenido actual.
`a+`	Agrega; abre o crea un archivo para actualización; la escritura se hace al final del archivo.
`rb`	Abre un archivo para lectura en modo binario.
`wb`	Crea un archivo para escritura en modo binario. Si el archivo ya existe, descarta el contenido actual.
`ab`	Agrega; abre o crea un archivo para escritura al final del archivo en modo binario.
`rb+`	Abre un archivo para actualización (lectura y escritura) en modo binario.
`wb+`	Crea un archivo para actualización en modo binario. Si el archivo ya existe, descarta el contenido actual.
`ab+`	Agrega; abre o crea un archivo para actualización en modo binario; la escritura se hace al final del archivo.

```c
/* Fig. 11.3: fig11_03.c
   Crea un archivo secuencial */
#include <stdio.h>

int main()
{
   int cuenta;
   char nombre[ 30 ];
   double saldo;
   FILE *ptrCf;   /* ptrCf = apuntador al archivo clientes.dat

   if ( ( ptrCf= fopen( "clientes.dat", "w" ) ) == NULL )
      printf( "El archivo no pudo abrirse\n" );
   else {
      printf( "Introduzca la cuenta, el nombre, y el saldo.\n" );
      printf( "Introduzca EOF al final de la entrada.\n" );
      printf( "? " );
      scanf( "%d%s%lf", &cuenta, nombre, &saldo );

      while ( !feof( stdin ) ) {
         fprintf( ptrCf, "%d %s %.2f\n",
                 cuenta, nombre, saldo );
         printf( "? " );
         scanf( "%d%s%lf", &cuenta, nombre, &saldo );
      }

      fclose( ptrCf );
   }

   return 0;
}
```



Outline

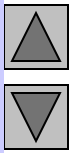
1. Inicializa variables y puntero `FILE`

1.1 Enlaza el puntero a un archivo

2. Entrada de datos

2.1 Escribe al archivo (`fprintf`)

3. Cierra archivo



Outline

Salida de Programa

```
Introduzca la cuenta, el nombre, y el saldo.
Introduzca EOF al final de la entrada.
? 100 Sanchez 24.98
? 200 Lopez 345.67
? 300 Blanco 0.00
? 400 Martinez -42.16
? 500 Rico 224.62
? ^Z
```

11.5 Leyendo datos de un archivo de acceso secuencial

- Leyendo un archivo de acceso secuencial
  - Crear un puntero **FILE**, enlazarlo al archivo para lectura

```
miPtr = fopen( "miArchivo.dat", "r" );
```

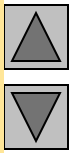
  - Use **fscanf** para leer desde el archivo
    - Como **scanf**, excepto el primer argumento es un puntero **FILE**

```
fscanf( miPtr, "%d%s%f", &miInt, &miString, &miFloat );
```

  - Dato leído desde el inicio al fin
  - Posición de apuntar el hilo - indica número de siguientes bytes a leer/escribir
    - No es realmente un puntero, sino un valor entero (especifica la ubicación del byte)
    - También llamado desplazamiento de bytes
  - **rewind(miPtr)** - posiciona el puntero posición de archivo al inicio del archivo (byte 0)

```c
/* Fig. 11.7: fig11_07.c
   Lectura e impresión de un archivo secuencial */
#include <stdio.h>

int main()
{
   int cuenta;
   char nombre[ 30 ];
   double balance;
   FILE *ptrCf;/* ptrCf = apuntador al archivo clientes.dat */

   if ( ( ptrCf = fopen( "clientes.dat", "r" ) ) == NULL )
      printf( "El archivo no pudo abrirse\n" );
   else {
      printf( "%-10s%-13s%s\n", "Cuenta", "Nombre", "Saldo" );
      fscanf( cfPtr, "%d%s%lf", &cuenta, nombre, &saldo );

      while ( !feof( ptrCf ) ) {
         printf( "%-10d%-13s%7.2f\n", cuenta, nombre, saldo );
         fscanf( ptrCf, "%d%s%lf", &cuenta, nombre, &saldo );
      }

      fclose( ptrCf );
   }

   return 0;
}
```

```
Cuenta    Nombre         Saldo
100       Jones          24.98
200       Doe           345.67
300       White           0.00
400       Stone         -42.16
500       Rich          224.62
```



Outline

1. Inicializa variables

1.1 Enlaza puntero a archivo

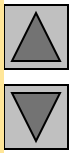
2. Lee dto (`fscanf`)

2.1 Imprime

3. Cierra archivo

Salida del Programa

```
1    /* Fig. 11.8: fig11 08.c
2       Programa de investigación crediticia */
3    #include <stdio.h>
4
5    int main()
6    {
7       int consulta, cuenta;
8       double saldo;
9       char nombre[ 30 ];
10      FILE *ptrCf;
11
12      if ( ( ptrCf = fopen( "clientes.dat", "r" ) ) == NULL )
13         printf( "El archivo no pudo abrirse\n" );
14      else {
15         printf( "Introduzca la consulta\n"
16                 " 1 - Lista las cuentas con saldo cero\n"
17                 " 2 - Lista las cuentas con saldo a favor\n"
18                 " 3 - Lista las cuentas con saldo en contra\n"
19                 " 4 - Fin del programa\n? " );
20         scanf( "%d", &consulta );
21
22         while ( consulta != 4 ) {
23            fscanf( ptrCf, "%d%s%lf", &cuenta, nombre,
24                    &saldo );
25
26            switch ( consulta) {
27               case 1:
28                  printf( "\nCuentas con saldo cero:\ ");
29
30
31                  while ( !feof( ptrCf ) ) {
32
```



Outline

1. Inicializa variables

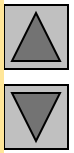
2. Abre archivos

2.1 Entrada de elección

2.2 Escanea archivos

3. Imprime

```
33                   if ( saldo == 0 )
34                      printf( "%-10d%-13s%7.2f\n",
35                             cuenta, nombre, saldo );
36
37                   fscanf( ptrCf, "%d%s%lf",
38                          &cuenta, nombre, &saldo );
39                }
40
41                break;
42             case 2:
43                printf( "\nnCuentas con saldos"
44                        "a favor:\n" );
45
46                while ( !feof( ptrCf ) ) {
47
48                   if ( saldo < 0 )
49                      printf( "%-10d%-13s%7.2f\n",
50                             cuenta, nombre, saldo );
51
52                   fscanf( ptrCf, "%d%s%lf",
53                          &cuenta, nombre, &saldo );
54                }
55
56                break;
57             case 3:
58                printf( "\nCuentas con saldo en"
59                        "contra:\n" );
60
61                while ( !feof( ptrCf ) ) {
62
63                   if ( saldo > 0 )
64                      printf( "%-10d%-13s%7.2f\n",
```

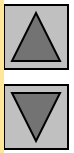


Outline

2.2 Escaneo de archivo

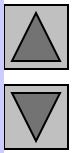
3. Impresión

```
65                              cuenta, nombre, saldo );
66
67                     fscanf( ptrCf, "%d%s%lf",
68                             &cuenta, nombre, &saldo );
69                  }
70
71                  break;
72               }
73
74               rewind( ptrCf );
75               printf( "\n? " );
76               scanf( "%d", &consulta );
77            }
78
79            printf( "Fin de la ejecucion\n" );
80            fclose( ptrCf );
81         }
82
83         return 0; /* indica terminación exitosa */
84      } /* fin de main */
```



Outline

3.1 Cierre de archivo



Outline

Salida de Programa

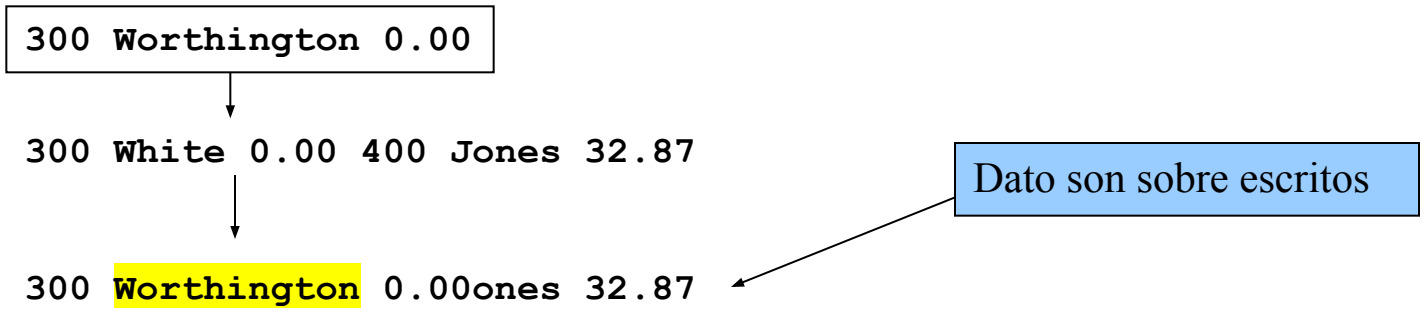
```
Introduzca la consulta
1 - Lista las cuentas con saldo cero
2 - Lista las cuentas con saldo a favor
3 - Lista las cuentas con saldo en contra
4 - Fin del programa
? 1

Cuentas con saldo cero:
300 Blanco 0.00

? 2

Cuentas con saldos a favor :
400 Martinez -42.16

? 3

Cuentas con saldo en contra:
100 Sanchez 24.98
200 Lopez 345.67
500 Rico 224.62

? 4

Fin de la ejecucion.
```

11.5 Leyendo datos de un archivo de acceso secuencial (II)

- Archivo de acceso secuencial
  - No puede ser modificado sin el riesgo de destruir otros datos

```
300 White 0.00 400 Jones 32.87   (viejo dato en archivo)
```

Si queremos cambiar el nombre White a Worthington,

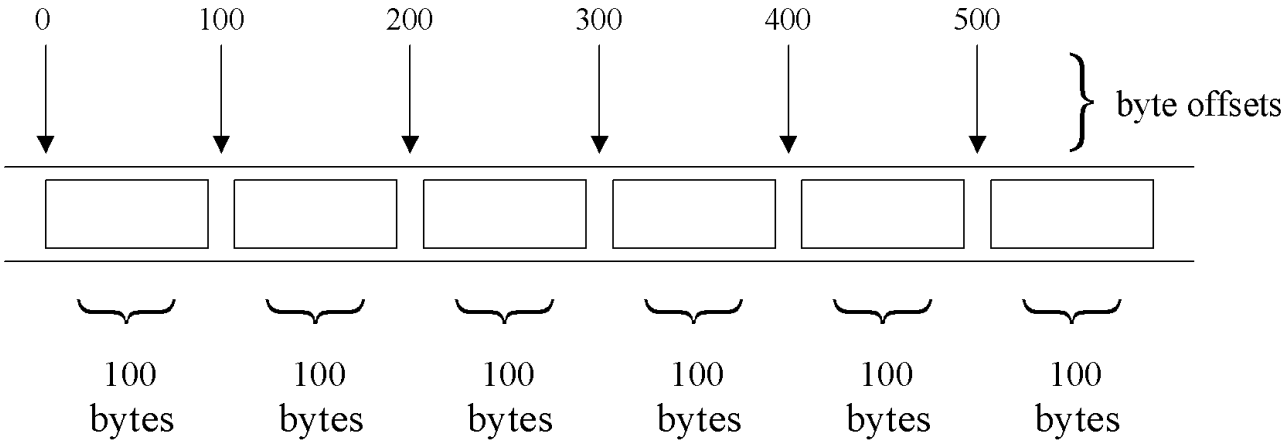


11.5 Leyendo datos de un archivo de acceso secuencial (III)

- Salida Formateada
  - Representación diferente en los archivos y la pantalla que la representación interna
  - **`1`**, **`34`**, **`-890`** son todos **`int`**s, pero tienen diferentes tamaño sobre disco

11.6 Archivo de Acceso Aleatorio

- Archivo de acceso aleatorio
  - Acceder a los registros individuales sin buscar en otros registros
  - Acceso instantáneo a los registros de un archivo
  - Los datos pueden ser insertados sin destruir otros datos
  - Los datos almacenados anteriormente pueden ser actualizados o eliminados sin necesidad de sobreescribirlos.
- Aplicado mediante registros de longitud fija
  - Los archivos secuenciales no tienen registros de longitud fija



11.7 Creación de un archivo de acceso aleatorio

- Dato
  - Dato no formateado (guardado como "bytes brutos") en archivos de acceso aleatorio
    - Todos los datos del mismo tipo (**`int`**s, por ejemplo) usan la misma memoria
    - Todos los registros del mismo tipo tienen longitud fija
    - Dato no son legibles por humanos

11.7 Creación de un archivo de acceso aleatorio (II)

- Funciones E/S no formateados
  - **`fwrite`** - Transfiere bytes desde una ubicación de memoria a un archivo
  - **`fread`** - Transfiere bytes desde un archivo a una ubicación de memoria
  - `fwrite( &numero, sizeof( int ), 1, miPtr );`
    - `&numero` - Ubicación desde donde transferir bytes
    - `sizeof( int )` - Numero de bytes a transferir
    - `1` - para arreglos, número de elementos a transferir
      - En este caso, "un elemento" de un arreglo está siendo transferido
    - `miPtr` - Archivo a transferir a o desde
    - **`fread`** similar

11.7 Creación de un archivo de acceso aleatorio (III)

- Escribiendo esctructuras - **`structs`**

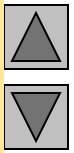
```
fwrite( &miObjecto, sizeof (struct miStruct), 1, miPtr );
```

  - **`sizeof`** - Retorna en bytes el objeto en paréntesis

- Para escribir varios (`n`) elementos de arreglo

  - Puntero al arreglo como primer argumento
  - Número de elementos (`n`) a escribir como tercer argumento

```
1   /* Fig. 11.11: fig11_11.c
2      Creando un archivo de acceso aleatorio de forma secuencial
3   #include <stdio.h>
4
5   struct datosCliente{
6      int numCta;
7      char apellido[ 15 ];
8      char nombre[ 10 ];
9      double saldo;
10  };
11
12  int main()
13  {
14     int i;
15     struct datosCliente clienteEnBlanco = { 0, "", "", 0.0 };
16     FILE *ptrCf;
17
18     if ( ( ptrCf = fopen( "credito.dat", "w" ) ) == NULL )
19        printf( "No pudo abrirse el archivo.\n" );
20     else {
21
22        for ( i = 1; i <= 100; i++ )
23           fwrite( &clienteEnBlanco,
24                   sizeof( struct datosCliente), 1, ptrCf );
25
26        fclose( ptrCf);
27     }
28
29     return 0;
30  }
```



Outline

1. Define struct

1.1 Inicializa variable
1.2 Inicializa struct

2. Apertura archivo

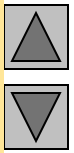
2.1 Escribe a archivo usando salida no formateada

3. Cierra archivo

11.8 Escritura aleatoria de datos en un archivo de acceso aleatorio

- **`fseek`**
  - Establece el puntero de la posición del archivo en una posición específica
  - `fseek( miPtr, offset, symbolic_constant);`
    - `miPtr` - puntero a archivo
    - `offset` - puntero a posición archivo (0 es primera ubicación)
    - `symbolic_constant` - específica de donde el archivo estamos leyendo desde
    - `SEEK_SET` - la búsqueda comienza al principio del archivo
    - `SEEK_CUR` - la búsqueda comienza en la ubicación actual en el archivo
    - `SEEK_END` - la búsqueda comienza al final del archivo

```c
/* Fig. 11.12: fig11 12.c
   Escritura en un archivo de acceso aleatorio */
#include <stdio.h>

struct datosCliente{
   int numCta;
   char apellido[ 15 ];
   char nombre[ 10 ];
   double saldo;
};

int main()
{
   FILE *ptrCr;
   struct datosCliente cliente = { 0, "", "", 0.0 };

   if ( ( ptrCf = fopen( "credito.dat", "rb+" ) ) == NULL )
      printf( "El archivo no pudo abrirse.\n" );
   else {
      printf( "Introduzca el numero de cuenta""
              " ( 1 a 100, 0 para terminar la entrada )\n? " );
      scanf( "%d", &cliente.numCta );

      while ( cliente.numCta != 0 ) {
         printf( "Introduzca el apellido, el nombre, el saldo\n?
         fscanf( stdin, "%s%s%lf", cliente.apellido,
                cliente.nombre, &cliente.saldo );
         fseek( cfPtr, ( cliente.numCta - 1 ) *
                sizeof( struct datosCliente), SEEK SET );
         fwrite( &cliente, sizeof( struct datosCliente ), 1,
                 ptrCf);
         printf( "Introduzca el numero de cuenta\n? " );
```



Outline

1. Define struct

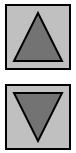
1.1 Inicializa variables

2. Abre archivos

2.1 Entrada de datos

2.2 Escribe a archivo

```
33             scanf( "%d", &cliente.numCta );
34          } /* fin de while */
35
36          fclose( ptrCf);  /* fclose cierra el archivo */
37       } /*fin de else */
38
39       return 0; /* indica terminación exitosa */
40    } /* fin de main */
```

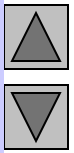


Outline

3. Cierra archivo

Salida de Programas

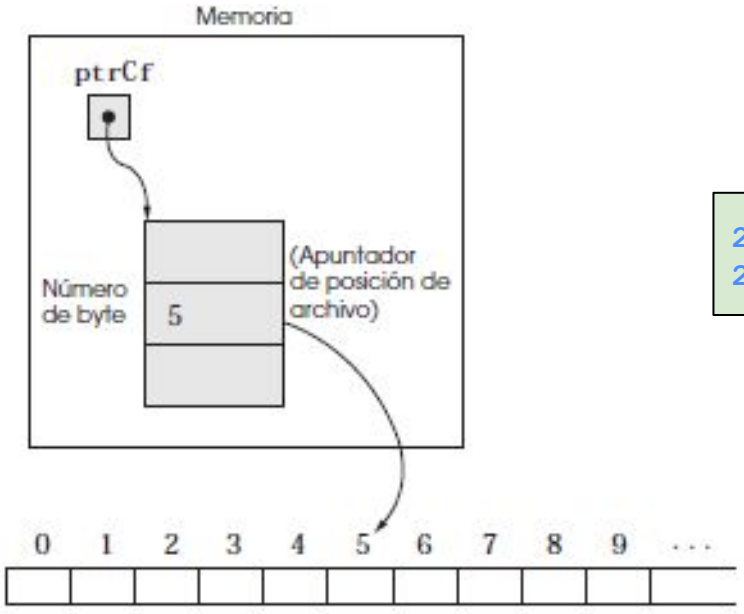
```
Introduzca el numero de cuenta ( 1 a 100, 0 para terminar la entrada )
? 37
Introduzca el apellido, el nombre, el saldo
? Baez Daniel 0.00
Introduzca el numero de cuenta
? 29
Introduzca el apellido, el nombre, el saldo
? Brito Nancy -24.54
Introduzca el numero de cuenta
? 96
Introduzca el apellido, el nombre, el saldo
? Sanchez Samuel 34.98
```



Outline

Salida de Programa

```
Introduzca el numero de cuenta
? 88
Introduzca el apellido, el nombre, el saldo
? Santos David 258.34
Introduzca el numero de cuenta
? 33
Introduzca el apellido, el nombre, el saldo
? Roberto Fernandez 314.33
Introduzca el numero de cuenta
? 0447
```



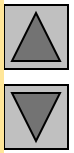

```
28          fseek( cfPtr, ( cliente.numCta - 1 ) *
27                 cliente.nombre, &cliente.saldo );
```

11.9 Lectura de datos desde un archivo de acceso aleatorio

- **fread**
  - Lee un número determinado de bytes de un archivo en la memoria

```
fread( &client, sizeof(struct datosCliente), 1, miPtr );
```

  - Puede leer varias elementos de arreglo de tamaño fijo
    - Provee puntero a arreglo
    - Indicate número de elementos a leer
  - Para leer múltiples elementos, se especifica en el tercer argumento



Outline

1. Define struct

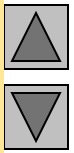
1.1 Inicializa variables

2. Lectura(`fread`)

2.1 Imprime

```
1   /* Fig. 11.15: fig11 15.c
2      Lectura secuencial de un archivo de acceso aleatorio */
3   #include <stdio.h>
4
5   struct datosCliente{
6      int numCta;;
7      char apellido[ 15 ];
8      char nombre[ 10 ];
9      double saldo;
10  };
11
12  int main()
13  {
14     FILE *ptrCf;
15     struct datosCliente cliente = { 0, "", "", 0.0 };
16
17     if ( ( ptrCf = fopen( "credito.dat", "r" ) ) == NULL )
18        printf( "No pudo abrirse el archivo.\n" );
19     else {
20        printf( "%-6s%-16s%-11s%10s\n", "Cta", "Apellido",
21               "Nombre", "Saldo" );
22
23        while ( !feof( ptrCf) ) {
24           fread( &cliente, sizeof( struct datosCliente), 1,
25                  ptrCr);
26
27           if ( cliente.numCta != 0 )
28              printf( "%-6d%-16s%-11s%10.2f\n",
29                     cliente.numCta, cliente.apellido,
30                     cliente.nombre, cliente.saldo );
31        }
32
```

```
33          fclose( ptrCf);
34       }
35
36       return 0;
37    }
```



Outline

3. Cierra archivo

Salida de Programa

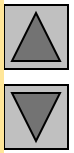
```
Cta   Apellido   Nombre     Saldo
29    Brito      Nancy      -24.54
33    Fernandez  Roberto    314.33
37    Baez       Daniel     0.00
88    Santos     David      258.34
96    Sanchez    Samuel     34.98
```

11.10 Ejemplo práctico: Programa de procesamiento de transacciones

- Utiliza archivos de acceso aleatorio para lograr el procesamiento de acceso instantáneo de la información de la cuenta de un banco

- Nosotros podemos
  - Actualizar las cuentas existentes
  - Añadir nuevas cuentas
  - Eliminar las cuentas
  - Almacenar un listado formateado de todas las cuentas en un archivo de texto

```
1   /* Fig. 11.16: fig11 16.c
2      Este programa lee de manera secuencial un archivo de acceso
3      actualiza los datos
4      ya escritos en el archivo, crea nuevos datos para
5      archivo, y elimina los datos ya existentes en el archivo*/
6   #include <stdio.h>
7
8   struct datosCliente{
9      int numCta;;
10     char apellido[ 15 ];
11     char nombre[ 10 ];
12     double saldo;
13  };
14
15  int intOpcion( void );
16  void archivoTexto( FILE * );
17  void actualizaRegistro( FILE * );
18  void nuevoRegistro( FILE * );
19  void eliminaRegistro( FILE * );
20
21  int main()
22  {
23     FILE *cfPtr;
24     int eleccion;
25
26     if ( ( cfPtr = fopen( "credito.dat", "r+" ) ) == NULL )
27        printf( "El archivo no pudo abrise.\n" );
28     else {
29
30        while ( ( eleccion = intOpcion() ) != 5 ) {
31
32           switch ( eleccion ) {
```



Outline

1. Define struct

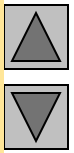
1.1 Function prototypes

1.2 Initialize variables

1.3 Link pointer and open file

2. Input choice

```
33              case 1:
34                 archivoTexto(( cfPtr );
35                 break;
36              case 2:
37                 actualizaRegistro(( cfPtr );
38                 break;
39              case 3:
40                 nuevoRegistro(( cfPtr );
41                 break;
42              case 4:
43                 eliminaRegistro(( cfPtr );
44                 break;
45           }
46        }
47
48        fclose( cfPtr );
49     }
50
51     return 0;
52  }
53
54  void archivoTexto(( FILE *ptrLee )
55  {
56     FILE *ptrEscribe;;
57     struct datosCliente cliente = { 0, "", "", 0.0 };
58
59     if ( ( ptrEscribe;= fopen( "cuentas.txt", "w" ) ) == NULL )
60        printf( "El archivo no puede ser abierto.\n" );
61     else {
62        rewind( ptrLee);
63        fprintf( ptrEscribe, "%-6s%-16s%-11s%10s\n",
64                "Cta", "Apellido", "Nombre", "Saldo" );
```

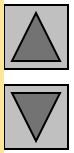


Outline

2.2 Perform action

3. Close file

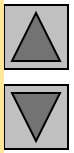
3.1 Function definitions



Outline

3.1 Function definitions

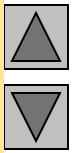
```
65
66          while ( !feof( ptrLee) ) {
67             fread( &client, sizeof( struct datosCliente), 1,
68                    ptrLee);
69
70             if ( cliente.nuCta != 0 )
71                fprintf( ptrEscribe, "%-6d%-16s%-11s%10.2f\n",
72                       cliente.numCta, cliente.apellido,
73                       cliente.nombre, cliente.saldo );
74          }
75
76          fclose( ptrEscribe);
77       }
78
79    }
80
81    void actualizaRegistro( FILE *ptrF )
82    {
83       int cuenta;
84       double transaccion;
85       struct datosCliente cliente = { 0, "", "", 0.0 };
86
87       printf( "Introduzca cuenta para actualizacion ( 1 - 100 ):"
88       scanf( "%d", &cuenta );
89       fseek( ptrF,
90              ( cuenta - 1 ) * sizeof( struct datosCliente ),
91              SEEK SET );
92       fread( &cliente, sizeof( struct datosCliente ), 1, ptrF);
93
94       if ( cliente.numCta == 0 )
95          printf( "La cuenta #%d no tiene informacion.\n", cuenta);
96       else {
```



Outline

3.1 Function definitions

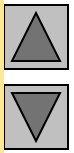
```
97          printf( "%-6d%-16s%-11s%10.2f\n\n",
98                  cliente.numCta, cliente.apellido,
99                  cliente.nombre, cliente.saldo );
100         printf( "Introduzca el cargo ( + ) o el pago ( - ):" );
101         scanf( "%lf", &transaccion );
102         cliente.saldo += transaccion;
103         printf( "%-6d%-16s%-11s%10.2f\n",
104                 cliente.numCta, cliente.apellido,
105                 cliente.nombre, cliente.saldo );
106         fseek( ptrF,
107                 ( account - 1 ) * sizeof( struct datosCliente),
108                 SEEK_SET );
109         fwrite( &cliente, sizeof( struct datosCliente), 1,
110                  ptrF);
111      }
112   }
113
114   void eliminaRegistro( FILE *ptrF )
115   {
116      struct datosCliente cliente,
117                          clienteEnBlanco = { 0, "", "", 0 };
118      int numCuenta;;
119
120      printf( "Ingrese número de cuenta a "
121              "borrar ( 1 - 100 ): " );
122      scanf( "%d", &numCuenta );
123      fseek( ptrF,
124             ( numCuenta- 1 ) * sizeof( struct datosCliente ),
125             SEEK_SET );
126      fread( &client, sizeof( struct clientData ), 1, ptrF);
```



Outline

3.1 Function definitions

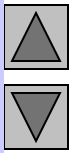
```
127
128     if ( cliente.numCta == 0 )
129        printf( "La cuenta %d no existe.\n", numCuenta);
130     else {
131        fseek( ptrF,
132           ( numCuenta- 1 ) * sizeof( struct datosCliente),
133           SEEK_SET );
134        fwrite( &clienteEnBlanco ,
135                 sizeof( struct datosCliente), 1, ptrF);
136     }
137  }
138
139  void nuevoRegistro( FILE *ptrF )
140  {
141     struct datosCliente cliente = { 0, "", "", 0.0 };
142     int numCuenta;
143     printf( "Introduzca el nuevo numero de cuenta ( 1 - 100 ):
144     scanf( "%d", &numCuenta );
145     fseek( ptrF,
146            ( numCuenta- 1 ) * sizeof( struct datosCliente),
147            SEEK_SET );
148     fread( &client, sizeof( struct datosCliente), 1, ptrF);
149
150     if ( cliente.numCta != 0 )
151        printf( "La cuenta #%d ya contiene informacion.\n",
152                cliente.numCta );
153     else {
154        printf( "Introduzca el apellido, el nombre, y el saldo\n?"
155        scanf( "%s%s%lf", &cliente.apellido, &cliente.nombre,
156              &client.saldo );
```



Outline

3.1 Function definitions

```
157         cliente.numCta = numCuenta;
158         fseek( ptrF, ( cliente.numCta - 1 ) *
159               sizeof( struct datosCliente), SEEK_SET );
160         fwrite( &cliente,
161                 sizeof( struct datosCliente), 1, ptrF);
162      }
163   }
164
165   int intOpcion( void )
166   {
167      int opcionMenu;
168
169      printf( "\nIntroduzca su opcion\n"
170         "1 - almacena un archivo de texto con formato, de las cuentas llamadas\n"
171         "    \"cuentas.txt\" para impresion\n"
172         "2 - actualiza una cuenta\n"
173         "3 - agrega una nueva cuenta\n"
174         "4 - elimina una cuenta\n"
175         "5 - fin del programa\n? " );
176      scanf( "%d", &opcionMenu );
177      return opcionMenu;
178   }
```



Outline

Salida de Programa

```
Despues de elegir la opción 1 cuentas.txt contiene:

Cta   Apellido        Nombre          Saldo
29    Brown           Nancy          -24.54
33    Dunn            Stacey         314.33
37    Barker          Doug             0.00
88    Smith           Dave           258.34
96    Stone           Sam             34.98
```

```
Ingrese cuenta a actualizar (1 - 100): 37
37    Barker          Doug             0.00

Ingrese cargo (+) o pago (-): +87.99
37    Barker          Doug            87.99
```

```
Ingrese nueva numero de cuenta (1 - 100): 22
Ingrese apellido, nombre, saldo
? Johnston Sarah 247.45
```

Cap. 11 – Procesamiento de Archivos

AGUIJE!!