Computer Programming I

Ming-Feng Tsai (Victor Tsai)

Dept. of Computer Science
National Chengchi University

C Structures, Unions, Bit Manipulations and Enumerations

10.1 Introduction
10.2 Structure Definitions
10.3 Initializing Structures
10.4 Accessing Structure Members
10.5 Using Structures with Functions
10.6 `typedef`
10.7 Example: High-Performance Card Shuffling and Dealing Simulation
10.8 Unions
10.9 Bitwise Operators
10.10 Bit Fields
10.10 Enumeration Constants

Bitwise Operators

- Each bit can assume the value 0 or the value 1.
- On most systems, a sequence of 8 bits forms a byte—the standard storage unit for a variable of type **`char`**.
- The bitwise operators are used to manipulate the bits of integral operands (**`char`**, **`short`**, **`int`** and **`long`**; both **`signed`** and **`unsigned`**).

Bitwise Operators (Cont.)

- For a detailed explanation of the binary (also called base-2) number system see Appendix C.
- Because of the machine-dependent nature of bitwise manipulations, these programs may not work on your system.
- The bitwise operators are bitwise AND (**&**), bitwise inclusive OR (**|**), bitwise exclusive OR (**^**), left shift (**<<**), right shift (**>>**) and complement (**~**).

Bitwise Operators (Cont.)

Operator		Description
&	bitwise AND	The bits in the result are set to 1 if the corresponding bits in the two operands are both 1.
|	bitwise inclusive OR	The bits in the result are set to 1 if at least one of the corresponding bits in the two operands is 1.
^	bitwise exclusive OR	The bits in the result are set to 1 if exactly one of the corresponding bits in the two operands is 1.
<<	left shift	Shifts the bits of the first operand left by the number of bits specified by the second operand; fill from the right with 0 bits.
>>	right shift	Shifts the bits of the first operand right by the number of bits specified by the second operand; the method of filling from the left is machine dependent.
~	one's complement	All 0 bits are set to 1 and all 1 bits are set to 0.

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
 5 void displayBits( unsigned value ); /* prototype */
 6
 7 int main( void ) {
 8    unsigned x; /* variable to hold user input */
 9
10    printf( "Enter an unsigned integer: " );
11    scanf( "%u", &x );
12
13    displayBits( x );
14    return 0; /* indicates successful termination */
15 } /* end main */
```

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
 5 void displayBits( unsigned value ); /* prototype */
 6
 7 int main( void ) {
 8    unsigned x; /* variable to hold user input */
 9
10    printf( "Enter an unsigned integer: " );
11    scanf( "%u", &x );
12
13    displayBits( x );
14    return 0; /* indicates successful termination */
15 } /* end main */
```

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
 5 void displayBits( unsigned value ); /* prototype */
 6
 7 int main( void ) {
 8    unsigned x; /* variable to hold user input */
 9
10    printf( "Enter an unsigned integer: " );
11    scanf( "%u", &x );
12
13    displayBits( x );
14    return 0; /* indicates successful termination */
15 } /* end main */
```

enter an unsigned integer

Bitwise Operators (Cont.)

- Example: fig10_07.c

```c
void displayBits( unsigned value ) {
    unsigned c; /* counter */

    /* define displayMask and left shift 31 bits */
    unsigned displayMask = 1 << 31;

    printf( "%10u = ", value );

    /* loop through bits */
    for ( c = 1; c <= 32; c++ ) {
        putchar( value & displayMask ? '1' : '0' );
        value <<= 1; /* shift value left by 1 */

        if ( c % 8 == 0 ) { /* output space after 8 bits */
            putchar( ' ' );
        } /* end if */
    } /* end for */

    putchar( '\n' );
} /* end function displayBits */
```

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
18 void displayBits( unsigned value ) {
19     unsigned c; /* counter */
20
21     /* define displayMask and left shift 31 bits */
22     unsigned displayMask = 1 << 31;
23
24     printf( "%10u = ", value );
25
26     /* loop through bits */
27     for ( c = 1; c <= 32; c++ ) {
28         putchar( value & displayMask ? '1' : '0' );
29         value <<= 1; /* shift value left by 1 */
30
31         if ( c % 8 == 0 ) { /* output space after 8 bits */
32             putchar( ' ' );
33         } /* end if */
34     } /* end for */
35
36     putchar( '\n' );
37 } /* end function displayBits */
```

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
18 void displayBits( unsigned value ) {
19     unsigned c; /* counter */
20
21     /* define displayMask and left shift 31 bits */
22     unsigned displayMask = 1 << 31;
23
24     printf( "%10u = ", value );
25
26     /* loop through bits */
27     for ( c = 1; c <= 32; c++ ) {
28         putchar( value & displayMask ? '1' : '0' );
29         value <<= 1; /* shift value left by 1 */
30
31         if ( c % 8 == 0 ) { /* output space after 8 bits */
32             putchar( ' ' );
33         } /* end if */
34     } /* end for */
35
36     putchar( '\n' );
37 } /* end function displayBits */
```

define a mask
10000000 00000000
00000000 00000000

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
18 void displayBits( unsigned value ) {
19     unsigned c; /* counter */
20
21     /* define displayMask and left shift 31 bits */
22     unsigned displayMask = 1 << 31;
23
24     printf( "%10u = ", value );
25
26     /* loop through bits */
27     for ( c = 1; c <= 32; c++ ) {
28         putchar( value & displayMask ? '1' : '0' );
29         value <<= 1; /* shift value left by 1 */
30
31         if ( c % 8 == 0 ) { /* output space after 8 bits */
32             putchar( ' ' );
33         } /* end if */
34     } /* end for */
35
36     putchar( '\n' );
37 } /* end function displayBits */
```

define a mask
10000000 00000000
00000000 00000000

Bitwise Operators (Cont.)

- Example: fig10_07.c

```
18 void displayBits( unsigned value ) {
19    unsigned c; /* counter */
20
21    /* define displayMask and left shift 31 bits */
22    unsigned displayMask = 1 << 31;
23
24    printf( "%10u = ", value );
25
26    /* loop through bits */
27    for ( c = 1; c <= 32; c++ ) {
28       putchar( value & displayMask ? '1' : '0' );
29       value <<= 1; /* shift value left by 1 */
30
31       if ( c % 8 == 0 ) { /* output space after 8 bits */
32          putchar( ' ' );
33       } /* end if */
34    } /* end for */
35
36    putchar( '\n' );
37 } /* end function displayBits */
```

define a mask
10000000 00000000
00000000 00000000

all the bits except the high-order bit in **value** are "**masked off**", because any bit "ANDed" with 0 yields 0

Bitwise Operators (Cont.)

- Often, the bitwise **AND** operator is used with an operand called a mask—an integer value with specific bits set to 1.
- Masks are used to hide some bits in a value while selecting other bits.
- When **value** and **displayMask** are combined using **&**, all the bits except the high-order bit in variable **value** are "masked off" (hidden), because any bit "ANDed" with 0 yields 0.

Bitwise Operators (Cont.)

Bit 1	Bit 2	Bit 1 & Bit 2
0	0	0
1	0	0
0	1	0
1	1	1

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
 9    unsigned number1; /* define number1 */
10    unsigned number2; /* define number2 */
11    unsigned mask; /* define mask */
12    unsigned setBits; /* define setBits */
13
14    /* demonstrate bitwise AND (&) */
15    number1 = 65535;
16    mask = 1;
17    printf( "The result of combining the following\n" );
18    displayBits( number1 );
19    displayBits( mask );
20    printf( "using the bitwise AND operator & is\n" );
21    displayBits( number1 & mask );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
 9    unsigned number1; /* define number1 */
10    unsigned number2; /* define number2 */
11    unsigned mask; /* define mask */
12    unsigned setBits; /* define setBits */
13
14    /* demonstrate bitwise AND (&) */
15    number1 = 65535;
16    mask = 1;
17    printf( "The result of combining the following\n" );
18    displayBits( number1 );
19    displayBits( mask );
20    printf( "using the bitwise AND operator & is\n" );
21    displayBits( number1 & mask );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
 9    unsigned number1; /* define number1 */
10    unsigned number2; /* define number2 */
11    unsigned mask; /* define mask */
12    unsigned setBits; /* define setBits */
13
14    /* demonstrate bitwise AND (&) */
15    number1 = 65535;
16    mask = 1;
17    printf( "The result of combining the following\n" );
18    displayBits( number1 );
19    displayBits( mask );
20    printf( "using the bitwise AND operator & is\n" );
21    displayBits( number1 & mask );
```

demonstrate the use of the bitwise AND operator

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
 9    unsigned number1; /* define number1 */
10    unsigned number2; /* define number2 */
11    unsigned mask; /* define mask */
12    unsigned setBits; /* define setBits */
13
14    /* demonstrate bitwise AND (&) */
15    number1 = 65535;
16    mask = 1;
17    printf( "The result of combining the following\n" );
18    displayBits( number1 );
19    displayBits( mask );
20    printf( "using the bitwise AND operator & is\n" );
21    displayBits( number1 & mask );
```

demonstrate the use of the bitwise AND operator

```
The result of combining the following
      65535 = 00000000 00000000 11111111 11111111
          1 = 00000000 00000000 00000000 00000001
using the bitwise AND operator & is
          1 = 00000000 00000000 00000000 00000001
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

demonstrate the use of the bitwise OR operator

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

demonstrate the use of the bitwise OR operator

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

demonstrate the use of the bitwise OR operator

demonstrate the use of the bitwise XOR operator

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

demonstrate the use of the bitwise OR operator

demonstrate the use of the bitwise XOR operator

```
The result of combining the following
        15 = 00000000 00000000 00000000 00001111
       241 = 00000000 00000000 00000000 11110001
using the bitwise inclusive OR operator | is
       255 = 00000000 00000000 00000000 11111111
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
23    /* demonstrate bitwise inclusive OR (|) */
24    number1 = 15;
25    setBits = 241;
26    printf( "\nThe result of combining the following\n" );
27    displayBits( number1 );
28    displayBits( setBits );
29    printf( "using the bitwise inclusive OR operator | is\n" );
30    displayBits( number1 | setBits );
31
32    /* demonstrate bitwise exclusive OR (^) */
33    number1 = 139;
34    number2 = 199;
35    printf( "\nThe result of combining the following\n" );
36    displayBits( number1 );
37    displayBits( number2 );
38    printf( "using the bitwise exclusive OR operator ^ is\n" );
39    displayBits( number1 ^ number2 );
```

demonstrate the use of the bitwise OR operator

demonstrate the use of the bitwise XOR operator

```
The result of combining the following
        15 = 00000000 00000000 00000000 00001111
       241 = 00000000 00000000 00000000 11110001
using the bitwise inclusive OR operator | is
       255 = 00000000 00000000 00000000 11111111
```

```
The result of combining the following
       139 = 00000000 00000000 00000000 10001011
       199 = 00000000 00000000 00000000 11000111
using the bitwise exclusive OR operator ^ is
        76 = 00000000 00000000 00000000 01001100
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
41    /* demonstrate bitwise complement (~) */
42    number1 = 21845;
43    printf( "\nThe one's complement of\n" );
44    displayBits( number1 );
45    printf( "is\n" );
46    displayBits( ~number1 );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
41    /* demonstrate bitwise complement (~) */
42    number1 = 21845;
43    printf( "\nThe one's complement of\n" );
44    displayBits( number1 );
45    printf( "is\n" );
46    displayBits( ~number1 );
```

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
41    /* demonstrate bitwise complement (~) */
42    number1 = 21845;
43    printf( "\nThe one's complement of\n" );
44    displayBits( number1 );
45    printf( "is\n" );
46    displayBits( ~number1 );
```

demonstrate the use of the bitwise NOT operator

Bitwise Operators (Cont.)

- Example: fig10_09.c

```
41    /* demonstrate bitwise complement (~) */
42    number1 = 21845;
43    printf( "\nThe one's complement of\n" );
44    displayBits( number1 );
45    printf( "is\n" );
46    displayBits( ~number1 );
```

demonstrate the use of the bitwise NOT operator

```
The one's complement of
     21845 = 00000000 00000000 01010101 01010101
is
4294945450 = 11111111 11111111 10101010 10101010
```

Bitwise Operators (Cont.)

Bit 1	Bit 2	Bit 1 | Bit 2
0	0	0
1	0	1
0	1	1
1	1	1

Fig. 10.11 | Results of combining two bits with the bitwise inclusive OR operator |.

Bitwise Operators (Cont.)

Bit 1	Bit 2	Bit 1 ^ Bit 2
0	0	0
1	0	1
0	1	1
1	1	0

Fig. 10.12 | Results of combining two bits with the bitwise exclusive OR operator ^.

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8    unsigned number1 = 960; /* initialize number1 */
 9
10    /* demonstrate bitwise left shift */
11    printf( "\nThe result of left shifting\n" );
12    displayBits( number1 );
13    printf( "8 bit positions using the " );
14    printf( "left shift operator << is\n" );
15    displayBits( number1 << 8 );
16
17    /* demonstrate bitwise right shift */
18    printf( "\nThe result of right shifting\n" );
19    displayBits( number1 );
20    printf( "8 bit positions using the " );
21    printf( "right shift operator >> is\n" );
22    displayBits( number1 >> 8 );
```

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8     unsigned number1 = 960; /* initialize number1 */
 9
10     /* demonstrate bitwise left shift */
11     printf( "\nThe result of left shifting\n" );
12     displayBits( number1 );
13     printf( "8 bit positions using the " );
14     printf( "left shift operator << is\n" );
15     displayBits( number1 << 8 );
16
17     /* demonstrate bitwise right shift */
18     printf( "\nThe result of right shifting\n" );
19     displayBits( number1 );
20     printf( "8 bit positions using the " );
21     printf( "right shift operator >> is\n" );
22     displayBits( number1 >> 8 );
```

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8     unsigned number1 = 960; /* initialize number1 */
 9
10     /* demonstrate bitwise left shift */
11     printf( "\nThe result of left shifting\n" );
12     displayBits( number1 );
13     printf( "8 bit positions using the " );
14     printf( "left shift operator << is\n" );
15     displayBits( number1 << 8 );
16
17     /* demonstrate bitwise right shift */
18     printf( "\nThe result of right shifting\n" );
19     displayBits( number1 );
20     printf( "8 bit positions using the " );
21     printf( "right shift operator >> is\n" );
22     displayBits( number1 >> 8 );
```

left shift 8 bits

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8     unsigned number1 = 960; /* initialize number1 */
 9
10     /* demonstrate bitwise left shift */
11     printf( "\nThe result of left shifting\n" );
12     displayBits( number1 );
13     printf( "8 bit positions using the " );
14     printf( "left shift operator << is\n" );
15     displayBits( number1 << 8 );
16
17     /* demonstrate bitwise right shift */
18     printf( "\nThe result of right shifting\n" );
19     displayBits( number1 );
20     printf( "8 bit positions using the " );
21     printf( "right shift operator >> is\n" );
22     displayBits( number1 >> 8 );
```

left shift 8 bits

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8    unsigned number1 = 960; /* initialize number1 */
 9
10    /* demonstrate bitwise left shift */
11    printf( "\nThe result of left shifting\n" );
12    displayBits( number1 );
13    printf( "8 bit positions using the " );
14    printf( "left shift operator << is\n" );
15    displayBits( number1 << 8 );
16
17    /* demonstrate bitwise right shift */
18    printf( "\nThe result of right shifting\n" );
19    displayBits( number1 );
20    printf( "8 bit positions using the " );
21    printf( "right shift operator >> is\n" );
22    displayBits( number1 >> 8 );
```

left shift 8 bits

right shift 8 bits

Bitwise Operators (Cont.)

- Example: fig10_13.c

```
 8    unsigned number1 = 960; /* initialize number1 */
 9
10    /* demonstrate bitwise left shift */
11    printf( "\nThe result of left shifting\n" );
12    displayBits( number1 );
13    printf( "8 bit positions using the " );
14    printf( "left shift operator << is\n" );
15    displayBits( number1 << 8 );
16
17    /* demonstrate bitwise right shift */
18    printf( "\nThe result of right shifting\n" );
19    displayBits( number1 );
20    printf( "8 bit positions using the " );
21    printf( "right shift operator >> is\n" );
22    displayBits( number1 >> 8 );
```

left shift 8 bits

right shift 8 bits

```
The result of left shifting
    960 = 00000000 00000000 00000011 11000000
8 bit positions using the left shift operator << is
 245760 = 00000000 00000011 11000000 00000000

The result of right shifting
    960 = 00000000 00000000 00000011 11000000
8 bit positions using the right shift operator >> is
      3 = 00000000 00000000 00000000 00000011
```

Bitwise Operators (Cont.)

- The left-shift operator (**<<**) shifts the bits of its left operand to the left by the number of bits specified in its right operand.
- Bits vacated to the right are replaced with **0**s; **1**s shifted off the left are lost.
- The right-shift operator (**>>**) shifts the bits of its left operand to the right by the number of bits specified in its right operand.
- Performing a right shift on an **unsigned** integer causes the vacated bits at the left to be replaced by **0**s; **1**s shifted off the right are lost.

Bitwise Operators (Cont.)

Common Programming Error 10.12

The result of shifting a value is undefined if the right operand is negative or if the right operand is larger than the number of bits in which the left operand is stored.

Bitwise Operators (Cont.)

Bitwise assignment operators	
&=	Bitwise AND assignment operator.
|=	Bitwise inclusive OR assignment operator.
^=	Bitwise exclusive OR assignment operator.
<<=	Left-shift assignment operator.
>>=	Right-shift assignment operator.

Fig. 10.14 | The bitwise assignment operators.

Bit Fields

- C enables you to specify the number of bits in which an **`unsigned`** or **`int`** member of a structure or union is stored.
- This is referred to as a bit field.
- Bit fields enable better memory utilization by storing data in the minimum number of bits required.
- Bit field members must be declared as **`int`** or **`unsigned`**.

Bit Fields (Cont.)

- Consider the following structure definition:

```
struct bitCard {
    unsigned face: 4;
    unsigned suit: 2;
    unsigned color: 1;
};
```

which contains three **`unsigned`** bit fields—**`face`**, **`suit`** and **`color`**—used to represent a card from a deck of 52 cards.

Bit Fields (Cont.)

- A bit field is declared by following an **`unsigned`** or **`int`** member name with a colon (**`:`**) and an integer constant representing the width of the field (i.e., the number of bits in which the member is stored).

- The preceding structure definition indicates that member **`face`** is stored in 4 bits, member **`suit`** is stored in 2 bits and member **`color`** is stored in 1 bit.

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8     unsigned face : 4; /* 4 bits; 0-15 */
 9     unsigned suit : 2; /* 2 bits; 0-3 */
10     unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19     Card deck[ 52 ]; /* create array of Cards */
20
21     fillDeck( deck );
22     deal( deck );
23     return 0; /* indicates successful termination */
24 } /* end main */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8     unsigned face : 4; /* 4 bits; 0-15 */
 9     unsigned suit : 2; /* 2 bits; 0-3 */
10     unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19     Card deck[ 52 ]; /* create array of Cards */
20
21     fillDeck( deck );
22     deal( deck );
23     return 0; /* indicates successful termination */
24 } /* end main */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8     unsigned face : 4; /* 4 bits; 0-15 */
 9     unsigned suit : 2; /* 2 bits; 0-3 */
10     unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19     Card deck[ 52 ]; /* create array of Cards */
20
21     fillDeck( deck );
22     deal( deck );
23     return 0; /* indicates successful termination */
24 } /* end main */
```

define a struct and use bit fields

Bit Fields (Cont.)

- Example: fig10_16.c

```c
struct bitCard {
    unsigned face : 4; /* 4 bits; 0-15 */
    unsigned suit : 2; /* 2 bits; 0-3 */
    unsigned color : 1; /* 1 bit; 0-1 */
}; /* end struct bitCard */

typedef struct bitCard Card; /* new type name for struct bitCard */

void fillDeck( Card * const wDeck ); /* prototype */
void deal( const Card * const wDeck ); /* prototype */

int main( void ) {
    Card deck[ 52 ]; /* create array of Cards */

    fillDeck( deck );
    deal( deck );
    return 0; /* indicates successful termination */
} /* end main */
```

define a struct and use bit fields

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8    unsigned face : 4; /* 4 bits; 0-15 */
 9    unsigned suit : 2; /* 2 bits; 0-3 */
10    unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19    Card deck[ 52 ]; /* create array of Cards */
20
21    fillDeck( deck );
22    deal( deck );
23    return 0; /* indicates successful termination */
24 } /* end main */
```

define a struct and use bit fields

a new type name for struct bitCard

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8    unsigned face : 4; /* 4 bits; 0-15 */
 9    unsigned suit : 2; /* 2 bits; 0-3 */
10    unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19    Card deck[ 52 ]; /* create array of Cards */
20
21    fillDeck( deck );
22    deal( deck );
23    return 0; /* indicates successful termination */
24 } /* end main */
```

define a struct and use bit fields

a new type name for struct bitCard

Bit Fields (Cont.)

- Example: fig10_16.c

```
 7 struct bitCard {
 8    unsigned face : 4; /* 4 bits; 0-15 */
 9    unsigned suit : 2; /* 2 bits; 0-3 */
10    unsigned color : 1; /* 1 bit; 0-1 */
11 }; /* end struct bitCard */
12
13 typedef struct bitCard Card; /* new type name for struct bitCard */
14
15 void fillDeck( Card * const wDeck ); /* prototype */
16 void deal( const Card * const wDeck ); /* prototype */
17
18 int main( void ) {
19    Card deck[ 52 ]; /* create array of Cards */
20
21    fillDeck( deck );
22    deal( deck );
23    return 0; /* indicates successful termination */
24 } /* end main */
```

define a struct and use bit fields

a new type name for struct bitCard

create a deck of Cards

Bit Fields (Cont.)

- Example: fig10_16.c

```
27 void fillDeck( Card * const wDeck ) {
28     int i; /* counter */
29
30     /* loop through wDeck */
31     for ( i = 0; i <= 51; i++ ) {
32         wDeck[ i ].face = i % 13;
33         wDeck[ i ].suit = i / 13;
34         wDeck[ i ].color = i / 26;
35     } /* end for */
36 } /* end function fillDeck */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
27 void fillDeck( Card * const wDeck ) {
28     int i; /* counter */
29
30     /* loop through wDeck */
31     for ( i = 0; i <= 51; i++ ) {
32         wDeck[ i ].face = i % 13;
33         wDeck[ i ].suit = i / 13;
34         wDeck[ i ].color = i / 26;
35     } /* end for */
36 } /* end function fillDeck */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
27 void fillDeck( Card * const wDeck ) {
28     int i; /* counter */
29
30     /* loop through wDeck */
31     for ( i = 0; i <= 51; i++ ) {
32         wDeck[ i ].face = i % 13;
33         wDeck[ i ].suit = i / 13;
34         wDeck[ i ].color = i / 26;
35     } /* end for */
36 } /* end function fillDeck */
```

initialize the deck by looping through wDeck

Bit Fields (Cont.)

- Example: fig10_16.c

```
40 void deal( const Card * const wDeck ) {
41    int k1; /* subscripts 0-25 */
42    int k2; /* subscripts 26-51 */
43
44    /* loop through wDeck */
45    for ( k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++ ) {
46       printf( "Card:%3d  Suit:%2d  Color:%2d   ",
47               wDeck[ k1 ].face, wDeck[ k1 ].suit, wDeck[ k1 ].color );
48       printf( "Card:%3d  Suit:%2d  Color:%2d\n",
49               wDeck[ k2 ].face, wDeck[ k2 ].suit, wDeck[ k2 ].color );
50    } /* end for */
51 } /* end function deal */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
40 void deal( const Card * const wDeck ) {
41     int k1; /* subscripts 0-25 */
42     int k2; /* subscripts 26-51 */
43 
44     /* loop through wDeck */
45     for ( k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++ ) {
46         printf( "Card:%3d  Suit:%2d  Color:%2d   ",
47                 wDeck[ k1 ].face, wDeck[ k1 ].suit, wDeck[ k1 ].color );
48         printf( "Card:%3d  Suit:%2d  Color:%2d\n",
49                 wDeck[ k2 ].face, wDeck[ k2 ].suit, wDeck[ k2 ].color );
50     } /* end for */
51 } /* end function deal */
```

Bit Fields (Cont.)

- Example: fig10_16.c

```
40 void deal( const Card * const wDeck ) {
41    int k1; /* subscripts 0-25 */
42    int k2; /* subscripts 26-51 */
43
44    /* loop through wDeck */
45    for ( k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++ ) {
46       printf( "Card:%3d  Suit:%2d  Color:%2d   ",
47               wDeck[ k1 ].face, wDeck[ k1 ].suit, wDeck[ k1 ].color );
48       printf( "Card:%3d  Suit:%2d  Color:%2d\n",
49               wDeck[ k2 ].face, wDeck[ k2 ].suit, wDeck[ k2 ].color );
50    } /* end for */
51 } /* end function deal */
```

print out the deck

Bit Fields (Cont.)

- Example: fig10_16.c

```
40 void deal( const Card * const wDeck ) {
41    int k1; /* subscripts 0-25 */
42    int k2; /* subscripts 26-51 */
43
44    /* loop through wDeck */
45    for ( k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++ ) {
46       printf( "Card:%3d  Suit:%2d  Color:%2d   ",
47               wDeck[ k1 ].face, wDeck[ k1 ].suit, wDeck[ k1 ].color );
48       printf( "Card:%3d  Suit:%2d  Color:%2d\n",
49               wDeck[ k2 ].face, wDeck[ k2 ].suit, wDeck[ k2 ].color );
50    } /* end for */
51 } /* end function deal */
```

print out the deck

Bit Fields (Cont.)

- Example: fig10_16.c

```
40 void deal( const Card * const wDeck ) {
41     int k1; /* subscripts 0-25 */
42     int k2; /* subscripts 26-51 */
43
44     /* loop through wDeck */
45     for ( k1 = 0, k2 = k1 + 26; k1 <= 25; k1++, k2++ ) {
46         printf( "Card:%3d  Suit:%2d  Color:%2d   ",
47                 wDeck[ k1 ].face, wDeck[ k1 ].suit, wDeck[ k1 ].color );
48         printf( "Card:%3d  Suit:%2d  Color:%2d\n",
49                 wDeck[ k2 ].face, wDeck[ k2 ].suit, wDeck[ k2 ].color );
50     } /* end for */
51 } /* end function deal */
```

print out the deck

```
Card:  0  Suit: 0  Color: 0   Card:  0  Suit: 2  Color: 1
Card:  1  Suit: 0  Color: 0   Card:  1  Suit: 2  Color: 1
Card:  2  Suit: 0  Color: 0   Card:  2  Suit: 2  Color: 1
Card:  3  Suit: 0  Color: 0   Card:  3  Suit: 2  Color: 1
Card:  4  Suit: 0  Color: 0   Card:  4  Suit: 2  Color: 1
Card:  5  Suit: 0  Color: 0   Card:  5  Suit: 2  Color: 1
Card:  6  Suit: 0  Color: 0   Card:  6  Suit: 2  Color: 1
Card:  7  Suit: 0  Color: 0   Card:  7  Suit: 2  Color: 1
```

C Files

Objectives

- In this chapter, you'll learn
  - To create, read, write and update files
  - Sequential access file processing
  - Random-access file processing

11.1 Introduction
11.2 Data Hierarchy
11.3 Files and Streams
11.4 Creating a Sequential-Access File
11.5 Reading Data from a Sequential-Access File
11.6 Random-Access Files
11.7 Creating a Random-Access File
11.8 Writing Data Randomly to a Random-Access File
11.9 Reading Data from a Random-Access File
11.10 Case Study: Transaction-Processing Program

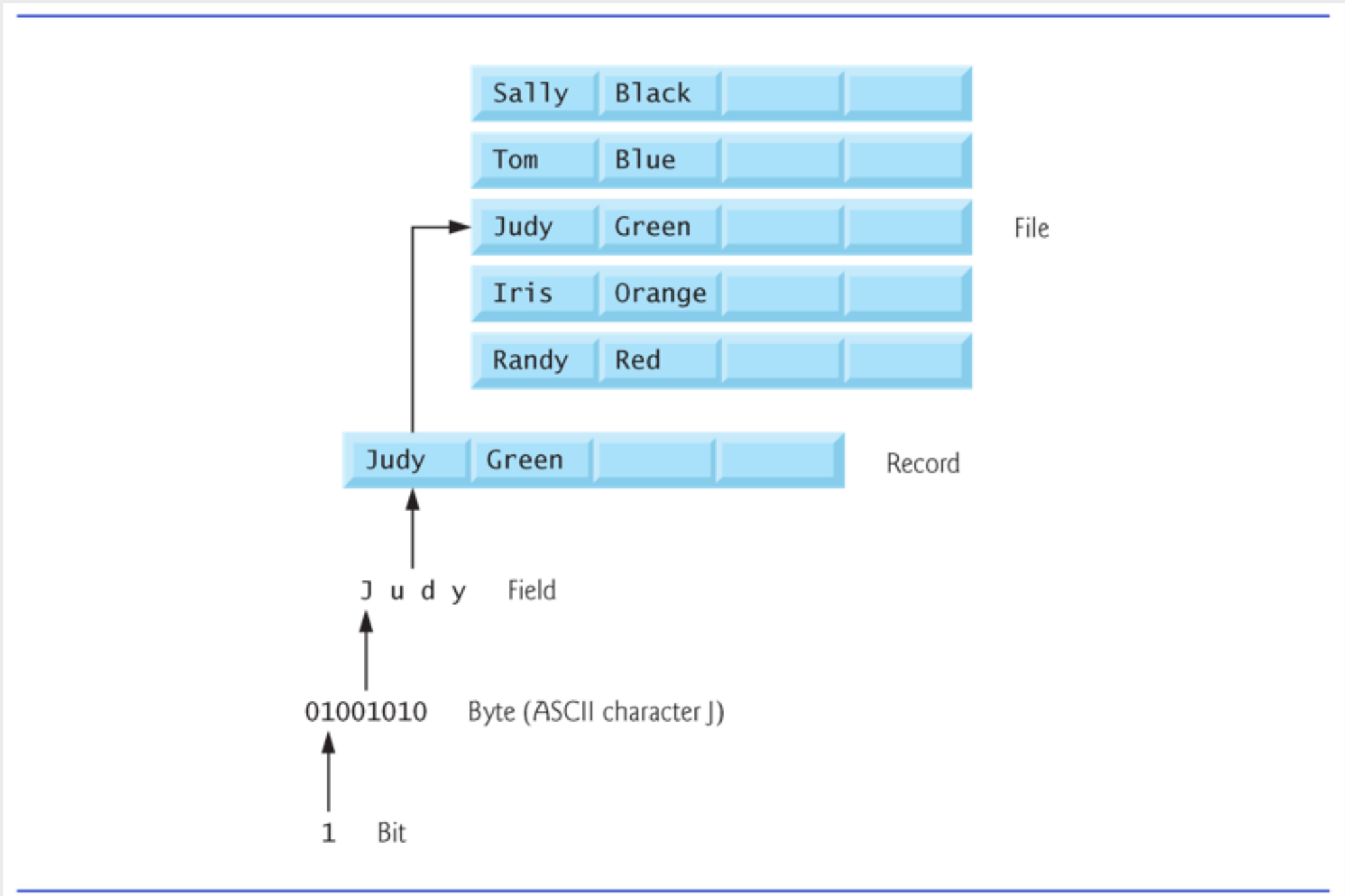
Introduction

- Storage of data in variables and arrays is temporary—such data is lost when a program terminates.
- Files are used for permanent retention of data.
- In this chapter, we explain how data files are created, updated and processed by C programs.
- We consider sequential-access files and random-access files.

Data Hierarchy

- Ultimately, all data items processed by a computer are reduced to combinations of zeros and ones.
- Such a data item is called a bit (short for "binary digit"—a digit that can assume one of two values).
- Data items processed by computers form a data hierarchy in which data items become larger and more complex in structure as we progress from bits, to characters (bytes), to fields, and so on.

Data Hierarchy (Cont.)



Files and Streams

- There are many ways of organizing records in a file.
- The most popular type of organization is called a sequential file, in which records are typically stored in order by the record key field.
- Each file ends either with an end-of-file marker or at a specific byte number recorded in a system-maintained, administrative data structure.
- When a file is opened, a stream is associated with the file.

Files and Streams (Cont.)

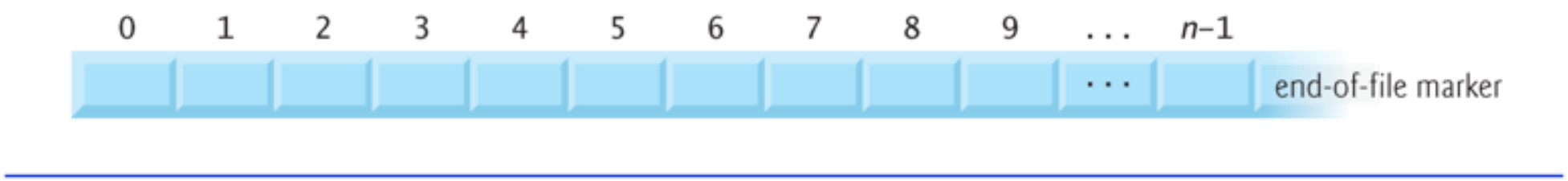


Fig. 11.2 | C's view of a file of *n* bytes.

Files and Streams (Cont.)

- Opening a file returns a pointer to a **`FILE`** structure (defined in **`<stdio.h>`**) that contains information used to process the file.
- This structure includes a file descriptor, i.e., an index into an operating system array called the open file table.

Files and Streams (Cont.)

- The standard library provides many functions for reading data from files and for writing data to files.
- Function **`fgetc`**, like **`getchar`**, reads one character from a file.
- Function **`fgetc`** receives as an argument a **`FILE`** pointer for the file from which a character will be read.
- The call **`fgetc`**(**`stdin`**) reads one character from stdin—the standard input.
- This call is equivalent to the call **`getchar()`**.
- Function **`fputc`**, like **`putchar`**, writes one character to a file.
- Function **`fputc`** receives as arguments a character to be written and a pointer for the file to which the character will be written.

Files and Streams (Cont.)

- The **`fgets`** and **`fputs`** functions, for example, can be used to read a line from a file and write a line to a file, respectively.
- In the next several sections, we introduce the file processing equivalents of functions **`scanf`** and **`printf`**—**`fscanf`** and **`fprintf`**.

Creating a Sequential-Access File

- C imposes no structure on a file.
- Thus, notions such as a record of a file do not exist as part of the C language.
- Therefore, you must provide a file structure to meet the requirements of a particular application.

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

use **fopen()** to open a file named "clients.dat"

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else {
16       printf( "Enter the account, name, and balance.\n" );
17       printf( "Enter EOF to end input.\n" );
18       printf( "? " );
19       scanf( "%d%s%lf", &account, name, &balance );
20
21       /* write account, name and balance into file with fprintf */
22       while ( !feof( stdin ) ) {
23          fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24          printf( "? " );
25          scanf( "%d%s%lf", &account, name, &balance );
26       } /* end while */
27
28       fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

use **fopen()** to open a file named "clients.dat"

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else {
16       printf( "Enter the account, name, and balance.\n" );
17       printf( "Enter EOF to end input.\n" );
18       printf( "? " );
19       scanf( "%d%s%lf", &account, name, &balance );
20
21       /* write account, name and balance into file with fprintf */
22       while ( !feof( stdin ) ) {
23          fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24          printf( "? " );
25          scanf( "%d%s%lf", &account, name, &balance );
26       } /* end while */
27
28       fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

use **fopen()** to open a file named "clients.dat"

use **feof()** to detect the end-of-file marker;
use **fprintf()** to write data into the file

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else {
16        printf( "Enter the account, name, and balance.\n" );
17        printf( "Enter EOF to end input.\n" );
18        printf( "? " );
19        scanf( "%d%s%lf", &account, name, &balance );
20
21        /* write account, name and balance into file with fprintf */
22        while ( !feof( stdin ) ) {
23            fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24            printf( "? " );
25            scanf( "%d%s%lf", &account, name, &balance );
26        } /* end while */
27
28        fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

use **fopen()** to open a file named "clients.dat"

use **feof()** to detect the end-of-file marker;
use **fprintf()** to write data into the file

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file. Exit program if unable to create file  */
13    if ( ( cfPtr = fopen( "clients.dat", "w" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else {
16       printf( "Enter the account, name, and balance.\n" );
17       printf( "Enter EOF to end input.\n" );
18       printf( "? " );
19       scanf( "%d%s%lf", &account, name, &balance );
20
21       /* write account, name and balance into file with fprintf */
22       while ( !feof( stdin ) ) {
23          fprintf( cfPtr, "%d %s %.2f\n", account, name, balance );
24          printf( "? " );
25          scanf( "%d%s%lf", &account, name, &balance );
26       } /* end while */
27
28       fclose( cfPtr ); /* fclose closes file */
29    } /* end else */
```

create a file pointer

use **fopen()** to open a file named "clients.dat"

use **feof()** to detect the end-of-file marker; use **fprintf()** to write data into the file

close the file stream

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
|^_^ mftsai@MBP [~/Classes/CP1_2011_fall/14/codes] ./a.out
|Enter the account, name, and balance.
|Enter EOF to end input.
|? 100 Jacky 24.98
|? 200 Doe 34.12
|? 300 White 0.00
|? 400 Stone -42.16
```

Creating a Sequential-Access File (Cont.)

- Example: fig11_03.c

```
^_^ mftsai@MBP [~/Classes/CP1_2011_fall/14/codes] ./a.out
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jacky 24.98
? 200 Doe 34.12
? 300 White 0.00
? 400 Stone -42.16
```

```
^_^ mftsai@MBP [~/Classes/CP1_2011_fall/14/codes] less clients.dat
100 Jacky 24.98
200 Doe 34.12
300 White 0.00
400 Stone -42.16
```

Creating a Sequential-Access File (Cont.)

- Function **`fopen`** takes two arguments: a file name and a file open mode.
  - The file open mode **`"w"`** indicates that the file is to be opened for writing.
  - function **`feof`** to determine whether the end-of-file indicator is set for the file to which **`stdin`** refers.

Creating a Sequential-Access File (Cont.)

Operating system	Key combination
Linux/Mac OS X/UNIX	*<Ctrl> d*
Windows	*<Ctrl> z*

Fig. 11.4 | End-of-file key combinations for various popular operating systems.

Creating a Sequential-Access File (Cont.)

- Function **`fprintf`** is equivalent to **`printf`** except that **`fprintf`** also receives as an argument a file pointer for the file to which the data will be written.
- Function **`fprintf`** can output data to the standard output by using **`stdout`** as the file pointer, as in:

```
fprintf(stdout, "%d %s %.2f\n",
        account, name, balance );
```

User has access to this

Only the operating system has access to this

1

`cfPtr = fopen( "clients.dat", "w" );`
`fopen` returns a pointer to a `FILE` structure (defined in `<stdio.h>`).

`cfPtr`

2

`FILE` structure for `"clients.dat"` contains a descriptor, i.e., a small integer that is an index into the Open File Table.

7

Open File Table

0	
1	
2	
3	
4	
5	
6	
7	FCB for `"clients.dat"`
.	.
.	.
.	.

4

The program calls an operating system service that uses data in the FCB to conotrol all input and output to the actual file on the disk. *Note:* The user cannot directly access the FCB.

This entry is copied from FCB on disk when the file is opened.

3

When the program issues an I/O call such as

```
fprintf( cfPtr, "%d %s %.2f",
   account, name, balance );
```

the program locates the descriptor (7) in the `FILE` structure and uses the descriptor to find the FCB in the Open File Table.

Creating a Sequential-Access File (Cont.)

Mode	Description
`r`	Open an existing file for reading.
`w`	Create a file for writing. If the file already exists, discard the current contents.
`a`	Append; open or create a file for writing at the end of the file.
`r+`	Open an existing file for update (reading and writing).
`w+`	Create a file for update. If the file already exists, discard the current contents.
`a+`	Append: open or create a file for update; writing is done at the end of the file.
`rb`	Open an existing file for reading in binary mode.
`wb`	Create a file for writing in binary mode. If the file already exists, discard the current contents.
`ab`	Append; open or create a file for writing at the end of the file in binary mode.
`rb+`	Open an existing file for update (reading and writing) in binary mode.
`wb+`	Create a file for update in binary mode. If the file already exists, discard the current contents.
`ab+`	Append: open or create a file for update in binary mode; writing is done at the end of the file.

Fig. 11.6 | File opening modes.

Reading Data from a Sequential-Access File

- Data is stored in files so that the data can be retrieved for processing when needed.
- The previous section demonstrated how to create a file for sequential access.
- This section shows how to read data sequentially from a file.

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16        printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17        fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19        /* while not end of file */
20        while ( !feof( cfPtr ) ) {
21            printf( "%-10d%-13s%7.2f\n", account, name, balance );
22            fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23        } /* end while */
24
25        fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19       /* while not end of file */
20       while ( !feof( cfPtr ) ) {
21          printf( "%-10d%-13s%7.2f\n", account, name, balance );
22          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23       } /* end while */
24
25       fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16        printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17        fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19        /* while not end of file */
20        while ( !feof( cfPtr ) ) {
21            printf( "%-10d%-13s%7.2f\n", account, name, balance );
22            fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23        } /* end while */
24
25        fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16        printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17        fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19        /* while not end of file */
20        while ( !feof( cfPtr ) ) {
21            printf( "%-10d%-13s%7.2f\n", account, name, balance );
22            fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23        } /* end while */
24
25        fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16        printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17        fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19        /* while not end of file */
20        while ( !feof( cfPtr ) ) {
21            printf( "%-10d%-13s%7.2f\n", account, name, balance );
22            fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23        } /* end while */
24
25        fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14        printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16        printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17        fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19        /* while not end of file */
20        while ( !feof( cfPtr ) ) {
21            printf( "%-10d%-13s%7.2f\n", account, name, balance );
22            fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23        } /* end while */
24
25        fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19       /* while not end of file */
20       while ( !feof( cfPtr ) ) {
21          printf( "%-10d%-13s%7.2f\n", account, name, balance );
22          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23       } /* end while */
24
25       fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

read through the whole file with **feof()** and **fscanf()**

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19       /* while not end of file */
20       while ( !feof( cfPtr ) ) {
21          printf( "%-10d%-13s%7.2f\n", account, name, balance );
22          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23       } /* end while */
24
25       fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

read through the whole file with **feof()** and **fscanf()**

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19       /* while not end of file */
20       while ( !feof( cfPtr ) ) {
21          printf( "%-10d%-13s%7.2f\n", account, name, balance );
22          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23       } /* end while */
24
25       fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

read through the whole file with **feof()** and **fscanf()**

close the file stream

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_07.c

```
6     int account; /* account number */
7     char name[ 30 ]; /* account name */
8     double balance; /* account balance */
9
10    FILE *cfPtr; /* cfPtr = clients.dat file pointer */
11
12    /* fopen opens file; exits program if file cannot be opened */
13    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
14       printf( "File could not be opened\n" );
15    } else { /* read account, name and balance from file */
16       printf( "%-10s%-13s%s\n", "Account", "Name", "Balance" );
17       fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
18
19       /* while not end of file */
20       while ( !feof( cfPtr ) ) {
21          printf( "%-10d%-13s%7.2f\n", account, name, balance );
22          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
23       } /* end while */
24
25       fclose( cfPtr ); /* fclose closes the file */
26    } /* end else */
```

create a file pointer

use **fopen()** to open a

read through the whole file with **feof()** and **fscanf()**

close the file stream

```
Account  Name          Balance
100      aaa            200.00
200      bbb            300.00
```

Reading Data from a Sequential-Access File (Cont.)

- A statement such as

```
rewind(cfPtr);
```

causes a program's file position pointer to be repositioned to the beginning of the file.

Reading Data from a Sequential-Access File (Cont.)

- The file position pointer is not really a pointer.
- Rather it's an **integer** value that specifies the byte location in the file at which the next read or write is to occur.
  - This is sometimes referred to as the **file offset**.

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
7     int request; /* request number */
8     int account; /* account number */
9     double balance; /* account balance */
10    char name[ 30 ]; /* account name */
11    FILE *cfPtr; /* clients.dat file pointer */
12
13    /* fopen opens the file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } else {
17       /* display request options */
18       printf( "Enter request\n"
19               " 1 - List accounts with zero balances\n"
20               " 2 - List accounts with credit balances\n"
21               " 3 - List accounts with debit balances\n"
22               " 4 - End of run\n? " );
23       scanf( "%d", &request );
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
7     int request; /* request number */
8     int account; /* account number */
9     double balance; /* account balance */
10    char name[ 30 ]; /* account name */
11    FILE *cfPtr; /* clients.dat file pointer */
12
13    /* fopen opens the file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } else {
17       /* display request options */
18       printf( "Enter request\n"
19               " 1 - List accounts with zero balances\n"
20               " 2 - List accounts with credit balances\n"
21               " 3 - List accounts with debit balances\n"
22               " 4 - End of run\n? " );
23       scanf( "%d", &request );
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
7     int request; /* request number */
8     int account; /* account number */
9     double balance; /* account balance */
10    char name[ 30 ]; /* account name */
11    FILE *cfPtr; /* clients.dat file pointer */
12
13    /* fopen opens the file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } else {
17       /* display request options */
18       printf( "Enter request\n"
19               " 1 - List accounts with zero balances\n"
20               " 2 - List accounts with credit balances\n"
21               " 3 - List accounts with debit balances\n"
22               " 4 - End of run\n? " );
23       scanf( "%d", &request );
```

create a file pointer

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
7     int request; /* request number */
8     int account; /* account number */
9     double balance; /* account balance */
10    char name[ 30 ]; /* account name */
11    FILE *cfPtr; /* clients.dat file pointer */
12
13    /* fopen opens the file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } else {
17       /* display request options */
18       printf( "Enter request\n"
19               " 1 - List accounts with zero balances\n"
20               " 2 - List accounts with credit balances\n"
21               " 3 - List accounts with debit balances\n"
22               " 4 - End of run\n? " );
23       scanf( "%d", &request );
```

create a file pointer

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
7     int request; /* request number */
8     int account; /* account number */
9     double balance; /* account balance */
10    char name[ 30 ]; /* account name */
11    FILE *cfPtr; /* clients.dat file pointer */
12
13    /* fopen opens the file; exits program if file cannot be opened */
14    if ( ( cfPtr = fopen( "clients.dat", "r" ) ) == NULL ) {
15       printf( "File could not be opened\n" );
16    } else {
17       /* display request options */
18       printf( "Enter request\n"
19               " 1 - List accounts with zero balances\n"
20               " 2 - List accounts with credit balances\n"
21               " 3 - List accounts with debit balances\n"
22               " 4 - End of run\n? " );
23       scanf( "%d", &request );
```

create a file pointer

open the file with **r** mode

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
26      while ( request != 4 ) {
27          /* read account, name and balance from file */
28          fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
29          switch ( request ) {
30              case 1:
31                  printf( "\nAccounts with zero balances:\n" );
32                  /* read file contents (until eof) */
33                  while ( !feof( cfPtr ) ) {
34                      if ( balance == 0 ) {
35                          printf( "%-10d%-13s%7.2f\n",
36                                  account, name, balance );
37                      } /* end if */
38                      /* read account, name and balance from file */
39                      fscanf( cfPtr, "%d%s%lf",
40                              &account, name, &balance );
41                  } /* end while */
42                  break;
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
26          while ( request != 4 ) {
27              /* read account, name and balance from file */
28              fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
29              switch ( request ) {
30                  case 1:
31                      printf( "\nAccounts with zero balances:\n" );
32                      /* read file contents (until eof) */
33                      while ( !feof( cfPtr ) ) {
34                          if ( balance == 0 ) {
35                              printf( "%-10d%-13s%7.2f\n",
36                                      account, name, balance );
37                          } /* end if */
38                          /* read account, name and balance from file */
39                          fscanf( cfPtr, "%d%s%lf",
40                                  &account, name, &balance );
41                      } /* end while */
42                      break;
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
26          while ( request != 4 ) {
27              /* read account, name and balance from file */
28              fscanf( cfPtr, "%d%s%lf", &account, name, &balance );
29              switch ( request ) {
30                  case 1:
31                      printf( "\nAccounts with zero balances:\n" );
32                      /* read file contents (until eof) */
33                      while ( !feof( cfPtr ) ) {
34                          if ( balance == 0 ) {
35                              printf( "%-10d%-13s%7.2f\n",
36                                      account, name, balance );
37                          } /* end if */
38                          /* read account, name and balance from file */
39                          fscanf( cfPtr, "%d%s%lf",
40                                  &account, name, &balance );
41                      } /* end while */
42                      break;
```

read data from the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
44              case 2:
45                  printf( "\nAccounts with credit balances:\n" );
46                  /* read file contents (until eof) */
47                  while ( !feof( cfPtr ) ) {
48                      if ( balance < 0 ) {
49                          printf( "%-10d%-13s%7.2f\n",
50                                  account, name, balance );
51                      } /* end if */
52                      /* read account, name and balance from file */
53                      fscanf( cfPtr, "%d%s%lf",
54                              &account, name, &balance );
55                  } /* end while */
56                  break;
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
44              case 2:
45                  printf( "\nAccounts with credit balances:\n" );
46                  /* read file contents (until eof) */
47                  while ( !feof( cfPtr ) ) {
48                      if ( balance < 0 ) {
49                          printf( "%-10d%-13s%7.2f\n",
50                                  account, name, balance );
51                      } /* end if */
52                      /* read account, name and balance from file */
53                      fscanf( cfPtr, "%d%s%lf",
54                              &account, name, &balance );
55                  } /* end while */
56                  break;
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
44              case 2:
45                  printf( "\nAccounts with credit balances:\n" );
46                  /* read file contents (until eof) */
47                  while ( !feof( cfPtr ) ) {
48                      if ( balance < 0 ) {
49                          printf( "%-10d%-13s%7.2f\n",
50                                  account, name, balance );
51                      } /* end if */
52                      /* read account, name and balance from file */
53                      fscanf( cfPtr, "%d%s%lf",
54                              &account, name, &balance );
55                  } /* end while */
56                  break;
```

read data from the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

read data from the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

read data from the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

read data from the file

return **cfPtr** to the beginning of the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

read data from the file

return **cfPtr** to the beginning of the file

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
58              case 3:
59                  printf( "\nAccounts with debit balances:\n" );
60                  /* read file contents (until eof) */
61                  while ( !feof( cfPtr ) ) {
62                      if ( balance > 0 ) {
63                          printf( "%-10d%-13s%7.2f\n",
64                                  account, name, balance );
65                      } /* end if */
66                      /* read account, name and balance from file */
67                      fscanf( cfPtr, "%d%s%lf",
68                              &account, name, &balance );
69                  } /* end while */
70                  break;
71
72              } /* end switch */
73
74              rewind( cfPtr ); /* return cfPtr to beginning of file */
75              printf( "\n? " );
76              scanf( "%d", &request );
77          } /* end while */
78
79          printf( "End of run.\n" );
80          fclose( cfPtr ); /* fclose closes the file */
```

read data from the file

return **cfPtr** to the beginning of the file

close the file stream

Reading Data from a Sequential-Access File (Cont.)

- Example: fig11_08.c

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
100        Andy              0.00
400        Joe               0.00

? 2

Accounts with credit balances:

? 3

Accounts with debit balances:
200        waterman        100.00
300        Hello           200.00

? 4
End of run.
```

Random-Access Files

- A random-access file is normally fixed in length and may be accessed directly (and thus quickly) without searching through other records.
- This makes random-access files appropriate for transaction processing systems that require rapid access to specific data.

Random-Access Files (Cont.)

- Because every record in a random-access file normally has the same length, the exact location of a record relative to the beginning of the file can be calculated as a function of the record key.
- We'll soon see how this facilitates immediate access to specific records, even in large files.

Random-Access Files (Cont.)

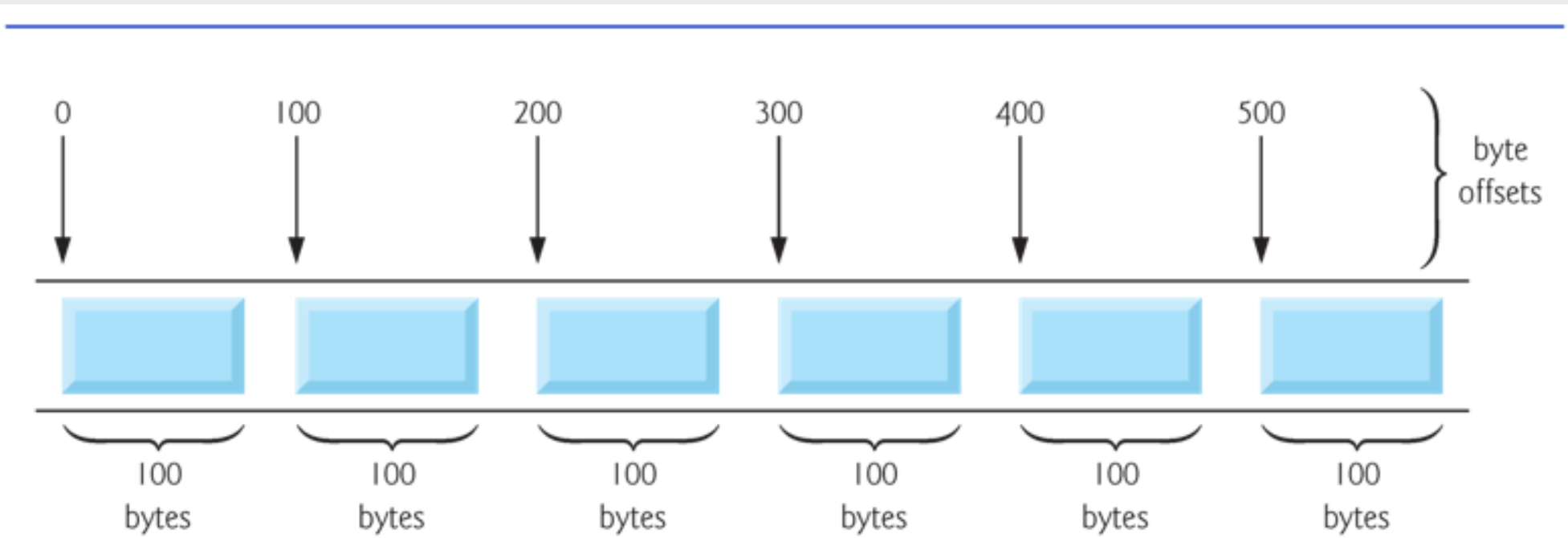


Fig. 11.10 | C's view of a random-access file.

Random-Access Files (Cont.)

- Fixed-length records enable data to be inserted in a random-access file without destroying other data in the file.
- Data stored previously can also be updated or deleted without rewriting the entire file.