Illustrations List *(Main Page)*

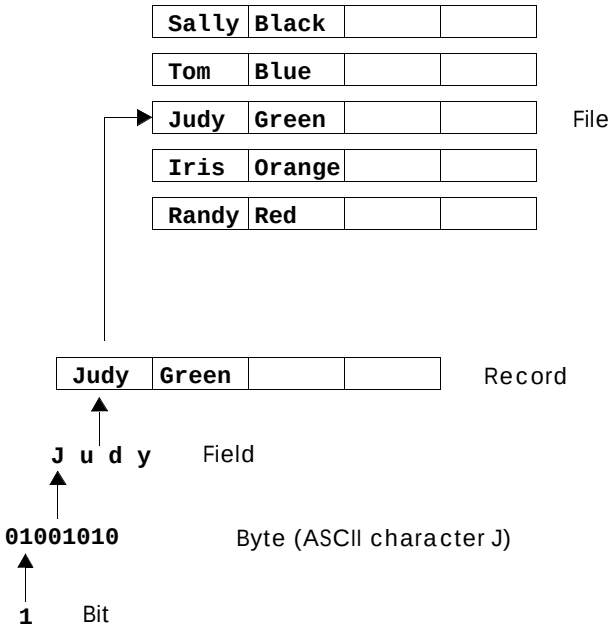

Fig. 14.1 The data hierarchy.

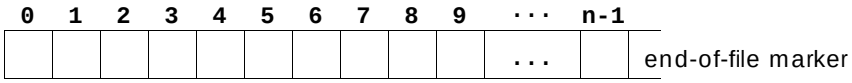

Fig. 14.2 C++'s view of a file of ***n*** bytes.

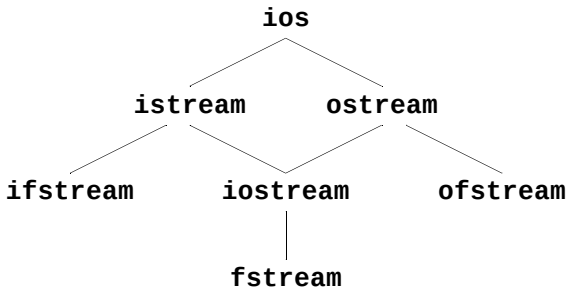

Fig. 14.3 Portion of stream I/O class hierarchy.

```
1  // Fig. 14.4: fig14_04.cpp
2  // Create a sequential file
3  #include <iostream.h>
4  #include <fstream.h>
5  #include <stdlib.h>
6
7  int main()
8  {
9     // ofstream constructor opens file
10    ofstream outClientFile( "clients.dat", ios::out );
11
12    if ( !outClientFile ) {  // overloaded ! operator
13       cerr << "File could not be opened" << endl;
14       exit( 1 );    // prototype in stdlib.h
15    }
16
17    cout << "Enter the account, name, and balance.\n"
18         << "Enter end-of-file to end input.\n? ";
19
20    int account;
21    char name[ 30 ];
22    float balance;
23
24    while ( cin >> account >> name >> balance ) {
25       outClientFile << account << ' ' << name
26                     << ' ' << balance << '\n';
27       cout << "? ";
28    }
29
30    return 0;  // ofstream destructor closes file
31 }
```

```
Enter the account, name, and balance.
Enter end-of-file to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

Fig. 14.4 Creating a sequential file.

Mode	Description
`ios::app`	Write all output to the end of the file.
`ios::ate`	Open a file for output and move to the end of the file (normally used to append data to a file). Data can be written anywhere in the file.
`ios::in`	Open a file for input.
`ios::out`	Open a file for output.
`ios::trunc`	Discard the file's contents if it exists (this is also the default action for `ios::out`)
`ios::nocreate`	If the file does not exist, the open operation fails.
`ios::noreplace`	If the file exists, the open operation fails.

Fig. 14.5 File open modes.

Computer system	Keyboard combination
UNIX systems	*<ctrl> d* (on a line by itself)
IBM PC and compatibles	*<ctrl> z*
Macintosh	*<ctrl> d*
VAX (VMS)	*<ctrl> z*

Fig. 14.6 End-of-file key combinations for various popular computer systems.

```
1  // Fig. 14.7: fig14_07.cpp
2  // Reading and printing a sequential file
3  #include <iostream.h>
4  #include <fstream.h>
5  #include <iomanip.h>
6  #include <stdlib.h>
7
8  void outputLine( int, const char *, double );
9
10 int main()
11 {
12    // ifstream constructor opens the file
13    ifstream inClientFile( "clients.dat", ios::in );
14
15    if ( !inClientFile ) {
16       cerr << "File could not be opened\n";
17       exit( 1 );
18    }
19
20    int account;
21    char name[ 30 ];
22    double balance;
23
24    cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
```

```
25          << setw( 13 ) << "Name" << "Balance\n";
26
27       while ( inClientFile >> account >> name >> balance )
28          outputLine( account, name, balance );
29
30       return 0;  // ifstream destructor closes the file
31    }
```

Fig. 14.7 Reading and printing a sequential file (part 1 of 2).

```
32
33    void outputLine( int acct, const char *name, double bal )
34    {
35       cout << setiosflags( ios::left ) << setw( 10 ) << acct
36            << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
37            << resetiosflags( ios::left )
38            << setiosflags( ios::fixed | ios::showpoint )
39            << bal << '\n';
40    }
```

```
Account   Name         Balance
100       Jones          24.98
200       Doe           345.67
300       White           0.00
400       Stone         -42.16
500       Rich          224.62
```

Fig. 14.7 Reading and printing a sequential file (part 2 of 2).

```
1   // Fig. 14.8: fig14_08.cpp
2   // Credit inquiry program
3   #include <iostream.h>
4   #include <fstream.h>
5   #include <iomanip.h>
6   #include <stdlib.h>
7
8   enum RequestType { ZERO_BALANCE = 1, CREDIT_BALANCE,
9                      DEBIT_BALANCE, END };
10  int getRequest();
11  bool shouldDisplay( int, double );
12  void outputLine( int, const char *, double );
13
14  int main()
15  {
16     // ifstream constructor opens the file
17     ifstream inClientFile( "clients.dat", ios::in );
18
19     if ( !inClientFile ) {
20        cerr << "File could not be opened" << endl;
21        exit( 1 );
22     }
23
24     int request;
25     int account;
26     char name[ 30 ];
27     double balance;
28
29     cout << "Enter request\n"
30          << " 1 - List accounts with zero balances\n"
31          << " 2 - List accounts with credit balances\n"
```

```
         << " 3 - List accounts with debit balances\n"
         << " 4 - End of run";
   request = getRequest();

   while ( request != END ) {

      switch ( request ) {
         case ZERO_BALANCE:
            cout << "\nAccounts with zero balances:\n";
            break;
         case CREDIT_BALANCE:
            cout << "\nAccounts with credit balances:\n";
            break;
         case DEBIT_BALANCE:
            cout << "\nAccounts with debit balances:\n";
            break;
      }

      inClientFile >> account >> name >> balance;

```

Fig. 14.8 Credit inquiry program (part 1 of 2).

```
      while ( !inClientFile.eof() ) {
         if ( shouldDisplay( request, balance ) )
            outputLine( account, name, balance );

         inClientFile >> account >> name >> balance;
      }

      inClientFile.clear();    // reset eof for next input
      inClientFile.seekg( 0 ); // move to beginning of file
      request = getRequest();
   }

   cout << "End of run." << endl;

   return 0;   // ifstream destructor closes the file
}

int getRequest()
{
   int request;

   do {
      cout << "\n? ";
      cin >> request;
   } while( request < ZERO_BALANCE && request > END );

   return request;
}

bool shouldDisplay( int type, double balance )
{
   if ( type == CREDIT_BALANCE && balance < 0 )
      return true;

   if ( type == DEBIT_BALANCE && balance > 0 )
      return true;

   if ( type == ZERO_BALANCE && balance == 0 )
      return true;

   return false;
```

```
93  }
94
95  void outputLine( int acct, const char *name, double bal )
96  {
97     cout << setiosflags( ios::left ) << setw( 10 ) << acct
98          << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
99          << resetiosflags( ios::left )
100         << setiosflags( ios::fixed | ios::showpoint )
101         << bal << '\n';
102 }
```

Fig. 14.8 Credit inquiry program (part 2 of 2).

```
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:
300       White           0.00

? 2

Accounts with credit balances:
400       Stone         -42.16

? 3

Accounts with debit balances:
100       Jones          24.98
200       Doe           345.67
500       Rich          224.62

? 4
End of run.
```

Fig. 14.9 Sample output of the credit inquiry program of Fig. 14.8.

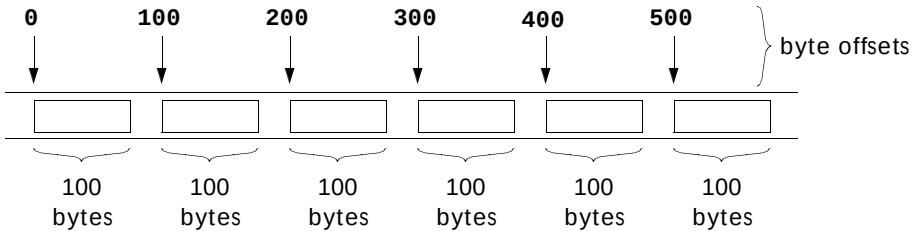

Fig. 14.10 C++'s view of a random access file.

```
1  // Fig. 14.11: clntdata.h
2  // Definition of struct clientData used in
3  // Figs. 14.11, 14.12, 14.14 and 14.15.
4  #ifndef CLNTDATA_H
5  #define CLNTDATA_H
6
7  struct clientData {
8     int accountNumber;
9     char lastName[ 15 ];
10    char firstName[ 10 ];
11    float balance;
12 };
13
14 #endif
```

Fig. 14.11 Creating a random access file sequentially (part 1 of 2).

```
15 // Fig. 14.11: fig14_11.cpp
16 // Creating a randomly accessed file sequentially
17 #include <iostream.h>
18 #include <fstream.h>
19 #include <stdlib.h>
20 #include "clntdata.h"
21
22 int main()
23 {
24    ofstream outCredit( "credit.dat", ios::out );
25
26    if ( !outCredit ) {
27       cerr << "File could not be opened." << endl;
28       exit( 1 );
29    }
30
31    clientData blankClient = { 0, "", "", 0.0 };
32
33    for ( int i = 0; i < 100; i++ )
34       outCredit.write(
35          reinterpret_cast<const char *>( &blankClient ),
36          sizeof( clientData ) );
37    return 0;
38 }
```

Fig. 14.11 Creating a random access file sequentially (part 2 of 2).

```
1  // Fig. 14.12: fig14_12.cpp
2  // Writing to a random access file
3  #include <iostream.h>
4  #include <fstream.h>
5  #include <stdlib.h>
6  #include "clntdata.h"
7
8  int main()
9  {
10    ofstream outCredit( "credit.dat", ios::ate );
11
12    if ( !outCredit ) {
13       cerr << "File could not be opened." << endl;
14       exit( 1 );
15    }
16
```

```
17      cout << "Enter account number "
18           << "(1 to 100, 0 to end input)\n? ";
19
20      clientData client;
21      cin >> client.accountNumber;
22
23      while ( client.accountNumber > 0 &&
24              client.accountNumber <= 100 ) {
25         cout << "Enter lastname, firstname, balance\n? ";
26         cin >> client.lastName >> client.firstName
27             >> client.balance;
28
29         outCredit.seekp( ( client.accountNumber - 1 ) *
30                          sizeof( clientData ) );
31         outCredit.write(
32            reinterpret_cast<const char *>( &client ),
33            sizeof( clientData ) );
34
```

Fig. 14.12 Writing data randomly to a random access file (part 1 of 2).

```
35         cout << "Enter account number\n? ";
36         cin >> client.accountNumber;
37      }
38
39      return 0;
40   }
```

Fig. 14.12 Writing data randomly to a random access file (part 2 of 2).

```
Enter account number (1 to 100, 0 to end input)
? 37
Enter lastname, firstname, balance
? Barker Doug 0.00
Enter account number
? 29
Enter lastname, firstname, balance
? Brown Nancy -24.54
Enter account number
? 96
Enter lastname, firstname, balance
? Stone Sam 34.98
Enter account number
? 88
Enter lastname, firstname, balance
? Smith Dave 258.34
Enter account number
? 33
Enter lastname, firstname, balance
? Dunn Stacey 314.33
Enter account number
? 0
```

Fig. 14.13 Sample execution of the program in Fig. 14.12.

```
1  // Fig. 14.14: fig14_14.cpp
2  // Reading a random access file sequentially
3  #include <iostream.h>
4  #include <iomanip.h>
5  #include <fstream.h>
6  #include <stdlib.h>
7  #include "clntdata.h"
8
9  void outputLine( ostream&, const clientData & );
10
11 int main()
12 {
13    ifstream inCredit( "credit.dat", ios::in );
14
15    if ( !inCredit ) {
16       cerr << "File could not be opened." << endl;
17       exit( 1 );
18    }
19
20    cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
21         << setw( 16 ) << "Last Name" << setw( 11 )
22         << "First Name" << resetiosflags( ios::left )
23         << setw( 10 ) << "Balance" << endl;
24
25    clientData client;
26
27    inCredit.read( reinterpret_cast<char *>( &client ),
28                   sizeof( clientData ) );
29
```

Fig. 14.14 Reading a random access file sequentially (part 1 of 2).

```
30    while ( inCredit && !inCredit.eof() ) {
31
32       if ( client.accountNumber != 0 )
33          outputLine( cout, client );
34
35       inCredit.read( reinterpret_cast<char *>( &client ),
36                      sizeof( clientData ) );
37    }
38
39    return 0;
40 }
41
42 void outputLine( ostream &output, const clientData &c )
43 {
44    output << setiosflags( ios::left ) << setw( 10 )
45           << c.accountNumber << setw( 16 ) << c.lastName
46           << setw( 11 ) << c.firstName << setw( 10 )
47           << setprecision( 2 ) << resetiosflags( ios::left )
48           << setiosflags( ios::fixed | ios::showpoint )
49           << c.balance << '\n';
```

```
50  }
```

```
Account   Last Name       First Name    Balance
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
```

Fig. 14.14 Reading a random access file sequentially (part 2 of 2).

```
1   // Fig. 14.15: fig14_15.cpp
2   // This program reads a random access file sequentially,
3   // updates data already written to the file, creates new
4   // data to be placed in the file, and deletes data
5   // already in the file.
6   #include <iostream.h>
7   #include <fstream.h>
8   #include <iomanip.h>
9   #include <stdlib.h>
10  #include "clntdata.h"
11
12  int enterChoice();
13  void textFile( fstream& );
14  void updateRecord( fstream& );
15  void newRecord( fstream& );
16  void deleteRecord( fstream& );
17  void outputLine( ostream&, const clientData & );
```

Fig. 14.15 Bank account program (part 1 of 5).

```
18  int getAccount( const char * );
19
20  enum Choices { TEXTFILE = 1, UPDATE, NEW, DELETE, END };
21
22  int main()
23  {
24     fstream inOutCredit( "credit.dat", ios::in | ios::out );
25
26     if ( !inOutCredit ) {
27        cerr << "File could not be opened." << endl;
28        exit ( 1 );
29     }
30
31     int choice;
32
33     while ( ( choice = enterChoice() ) != END ) {
34
35        switch ( choice ) {
36           case TEXTFILE:
37              textFile( inOutCredit );
38              break;
39           case UPDATE:
40              updateRecord( inOutCredit );
41              break;
42           case NEW:
43              newRecord( inOutCredit );
44              break;
45           case DELETE:
```

```
46             deleteRecord( inOutCredit );
47             break;
48          default:
49             cerr << "Incorrect choice\n";
50             break;
51       }
52
53       inOutCredit.clear();  // resets end-of-file indicator
54    }
55
56    return 0;
57 }
58
59 // Prompt for and input menu choice
60 int enterChoice()
61 {
62    cout << "\nEnter your choice" << endl
63         << "1 - store a formatted text file of accounts\n"
64         << "    called \"print.txt\" for printing\n"
65         << "2 - update an account\n"
66         << "3 - add a new account\n"
67         << "4 - delete an account\n"
68         << "5 - end program\n? ";
```

Fig. 14.15 Bank account program (part 2 of 5).

```
69
70    int menuChoice;
71    cin >> menuChoice;
72    return menuChoice;
73 }
74
75 // Create formatted text file for printing
76 void textFile( fstream &readFromFile )
77 {
78    ofstream outPrintFile( "print.txt", ios::out );
79
80    if ( !outPrintFile ) {
81       cerr << "File could not be opened." << endl;
82       exit( 1 );
83    }
84
85    outPrintFile << setiosflags( ios::left ) << setw( 10 )
86        << "Account" << setw( 16 ) << "Last Name" << setw( 11 )
87        << "First Name" << resetiosflags( ios::left )
88        << setw( 10 ) << "Balance" << endl;
89    readFromFile.seekg( 0 );
90
91    clientData client;
92    readFromFile.read( reinterpret_cast<char *>( &client ),
93                       sizeof( clientData ) );
94
95    while ( !readFromFile.eof() ) {
96       if ( client.accountNumber != 0 )
97          outputLine( outPrintFile, client );
98
99       readFromFile.read( reinterpret_cast<char *>( &client ),
100                          sizeof( clientData ) );
101   }
102 }
103
104 // Update an account's balance
105 void updateRecord( fstream &updateFile )
106 {
```

```
107    int account = getAccount( "Enter account to update" );
108
109    updateFile.seekg( ( account - 1 ) * sizeof( clientData ) );
110
111    clientData client;
112    updateFile.read( reinterpret_cast<char *>( &client ),
113                     sizeof( clientData ) );
114
115    if ( client.accountNumber != 0 ) {
116       outputLine( cout, client );
117       cout << "\nEnter charge (+) or payment (-): ";
118
119       float transaction;    // charge or payment
```

Fig. 14.15 Bank account program (part 3 of 5).

```
120       cin >> transaction;   // should validate
121       client.balance += transaction;
122       outputLine( cout, client );
123       updateFile.seekp( ( account-1 ) * sizeof( clientData ) );
124       updateFile.write(
125          reinterpret_cast<const char *>( &client ),
126          sizeof( clientData ) );
127    }
128    else
129       cerr << "Account #" << account
130            << " has no information." << endl;
131 }
132
133 // Create and insert new record
134 void newRecord( fstream &insertInFile )
135 {
136    int account = getAccount( "Enter new account number" );
137
138    insertInFile.seekg( ( account-1 ) * sizeof( clientData ) );
139
140    clientData client;
141    insertInFile.read( reinterpret_cast<char *>( &client ),
142                       sizeof( clientData ) );
143
144    if ( client.accountNumber == 0 ) {
145       cout << "Enter lastname, firstname, balance\n? ";
146       cin >> client.lastName >> client.firstName
147           >> client.balance;
148       client.accountNumber = account;
149       insertInFile.seekp( ( account - 1 ) *
150                           sizeof( clientData ) );
151       insertInFile.write(
152          reinterpret_cast<const char *>( &client ),
153          sizeof( clientData ) );
154    }
155    else
156       cerr << "Account #" << account
157            << " already contains information." << endl;
158 }
159
160 // Delete an existing record
161 void deleteRecord( fstream &deleteFromFile )
162 {
163    int account = getAccount( "Enter account to delete" );
164
165    deleteFromFile.seekg( (account-1) * sizeof( clientData ) );
166
167    clientData client;
```