Chapter 14

File Processing

OBJECTIVES

- To **create, read, write** and update files.
- Sequential file processing.
- Random-access file processing.

Topics

- **14.1 Introduction**
- 14.2 The Data Hierarchy
- 14.3 Files and Streams
- 14.4 Creating a Sequential File
- 14.5 Reading Data from a Sequential File
- 14.6 Random-Access Files

14.1 Introduction

- 程序中的变量、数组、常量等— 临时存储

- data persistence: File(文件)
- • permanent retention of large amounts of data
- • secondary storage devices

- sequential files and random-access files

Topics

- 14.1 Introduction
- **14.2 The Data Hierarchy**
- 14.3 Files and Streams
- 14.4 Creating a Sequential File
- 14.5 Reading Data from a Sequential File
- 14.6 Random-Access Files

14.2 The Data Hierarchy

- **Bit (binary digit) 二进制位: a digit that can assume one of two values**

- **Characters 字符: decimal, letters and special symbols (i.e., $, @ and many others). C++ provides data type char (occupies one byte of memory) and wchar_t (occupy more than one byte)**

$$\text{Bit} \xrightarrow[\text{character set}]{\text{ASCII, Unicode}} \text{Characters}$$

14.2 The Data Hierarchy

- **Fields 字段: a group of characters that conveys some meaning (called data members in C++)**

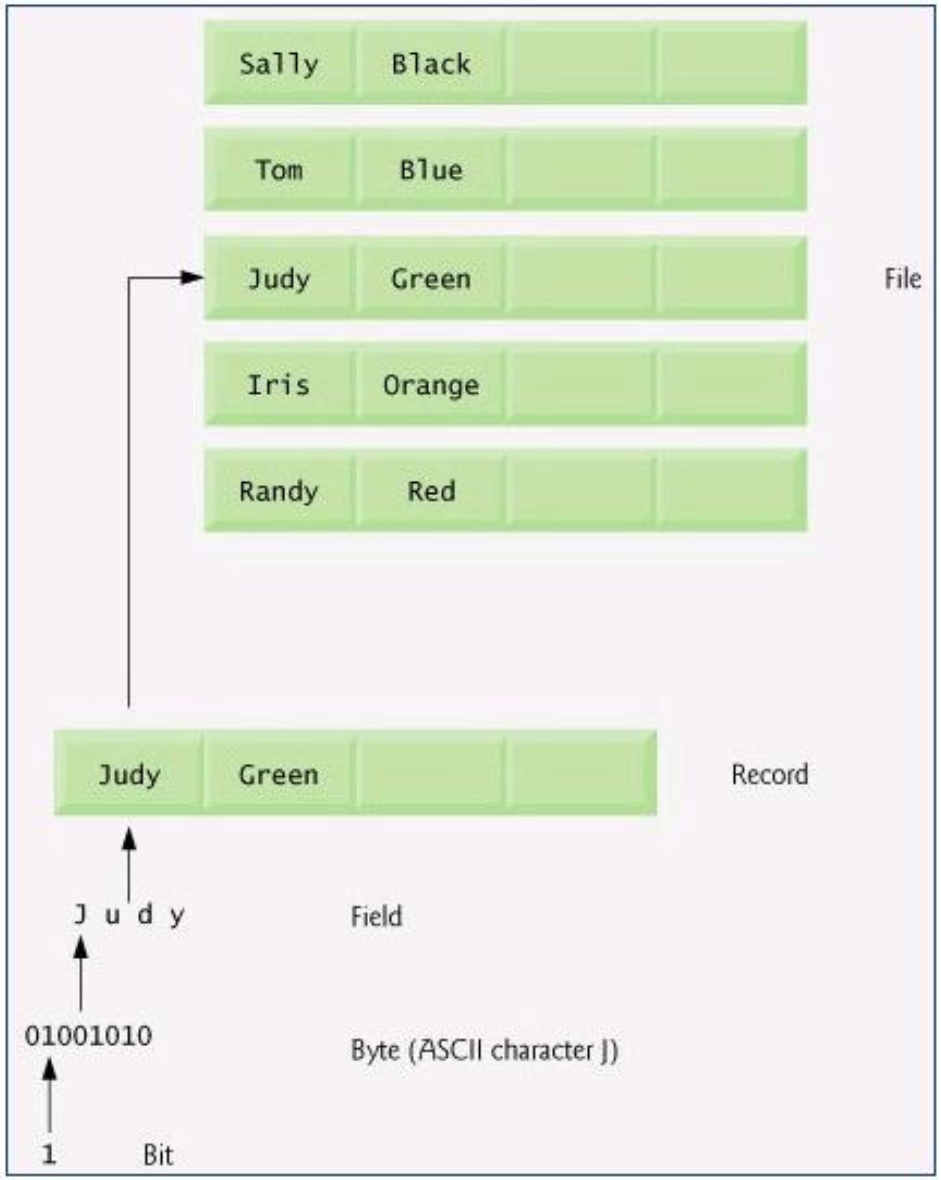
- **Record 记录: composed of several related fields (represented as a class in C++); A record key (键, 关键字) is a field unique to each record**

14.2 The Data Hierarchy

- **Sequential file 顺序文件: records typically are stored in order by a record-key field**

- **Database 数据库: a group of related files. A collection of programs designed to create and manage databases is called a database management system (DBMS)**

14.2 The Data Hierarchy



Topics

- 14.1 Introduction
- 14.2 The Data Hierarchy
- **14.3 Files and Streams**
- 14.4 Creating a Sequential File
- 14.5 Reading Data from a Sequential File
- 14.6 Random-Access Files

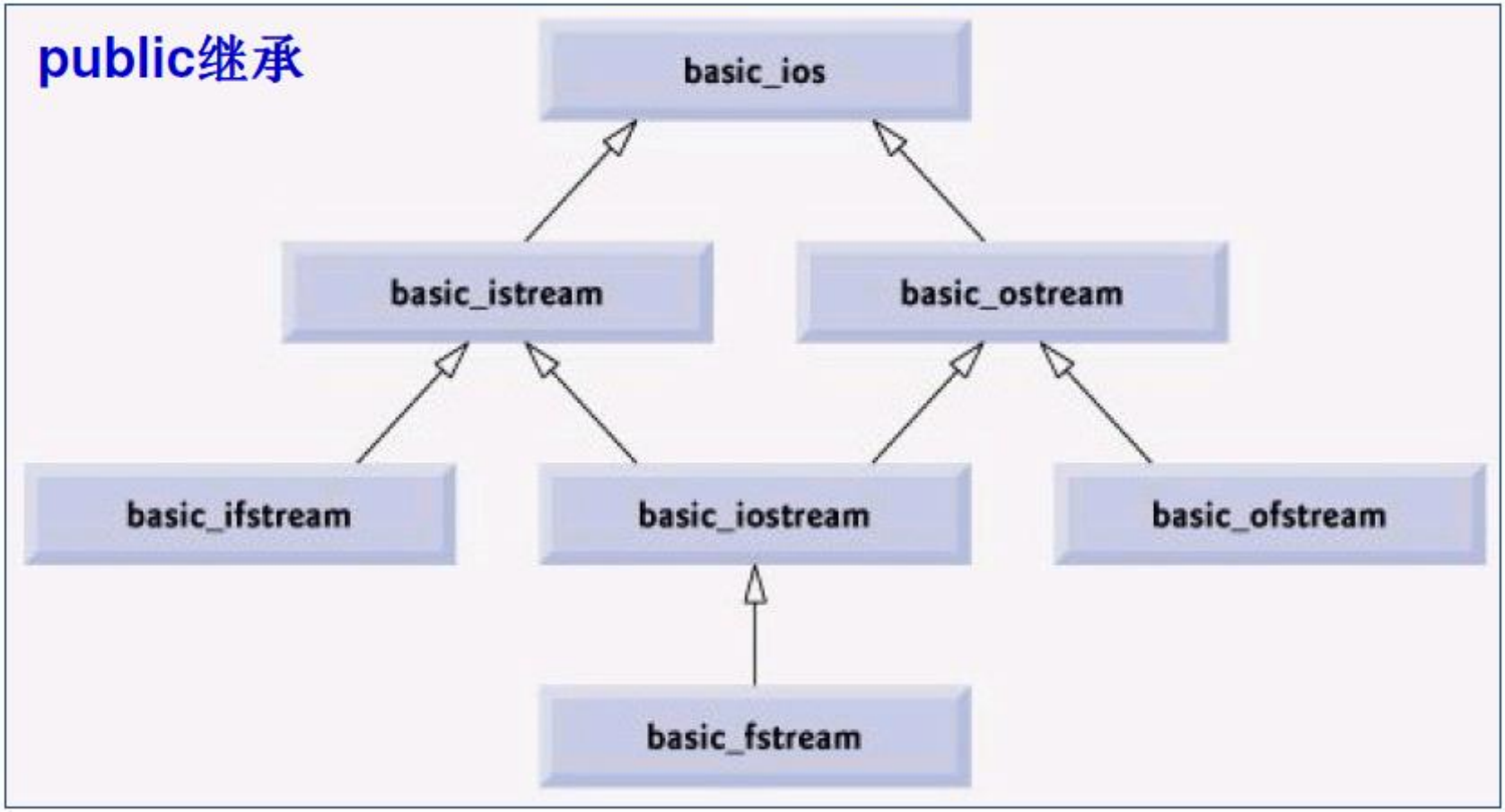
14.3 Files and Streams

- C++ views each file as a sequence of bytes and imposes no structure on a file. Each file ends either with an end-of-file marker or at a specific byte number recorded in a system maintained, administrative data structure. (文件结束标志或指定数量字节数)

- 所谓stream流就是一个字节序列: 当进行输入操作时, 字节从设备(键盘、磁盘等)流向内存; 当进行输出操作时, 字节从内存流向外部设备(键盘、磁盘等).

14.3 Files and Streams

- Stream I/O template hierarchy

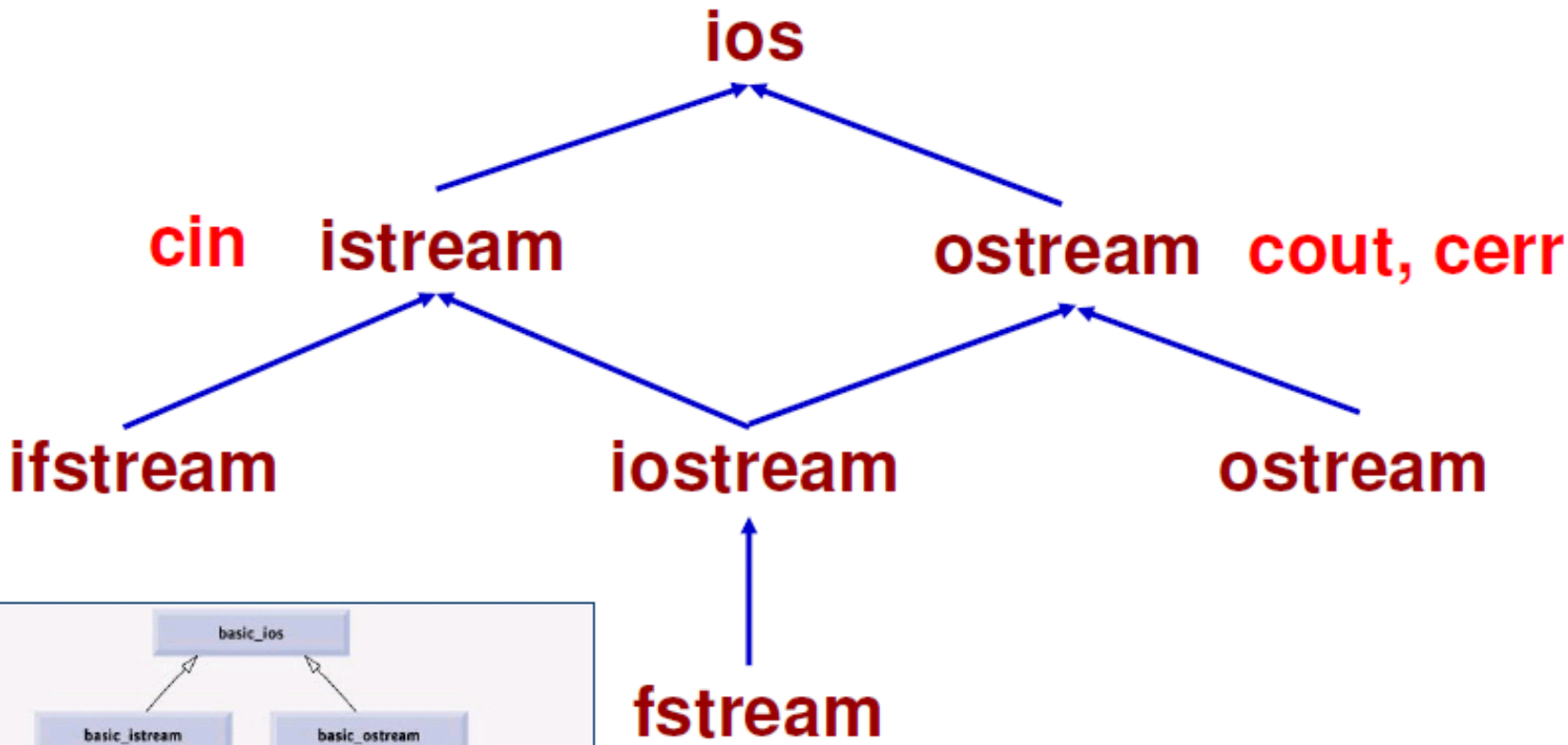


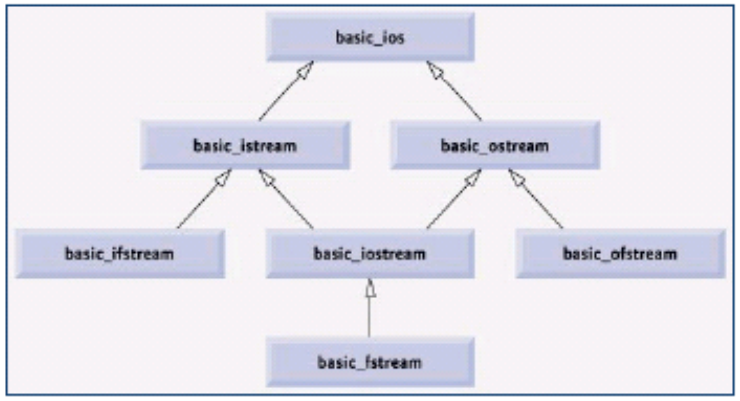
14.3 Files and Streams

```
// char TYPEDEFS, support char I/O
typedef basic_istream< char, char_traits<char> > istream;
typedef basic_ostream< char, char_traits<char> > ostream;
typedef basic_iostream< char, char_traits<char> > iostream;
typedef basic_ifstream< char, char_traits<char> > ifstream;
typedef basic_ofstream< char, char_traits<char> > ofstream;
typedef basic_fstream< char, char_traits<char> > fstream;
```

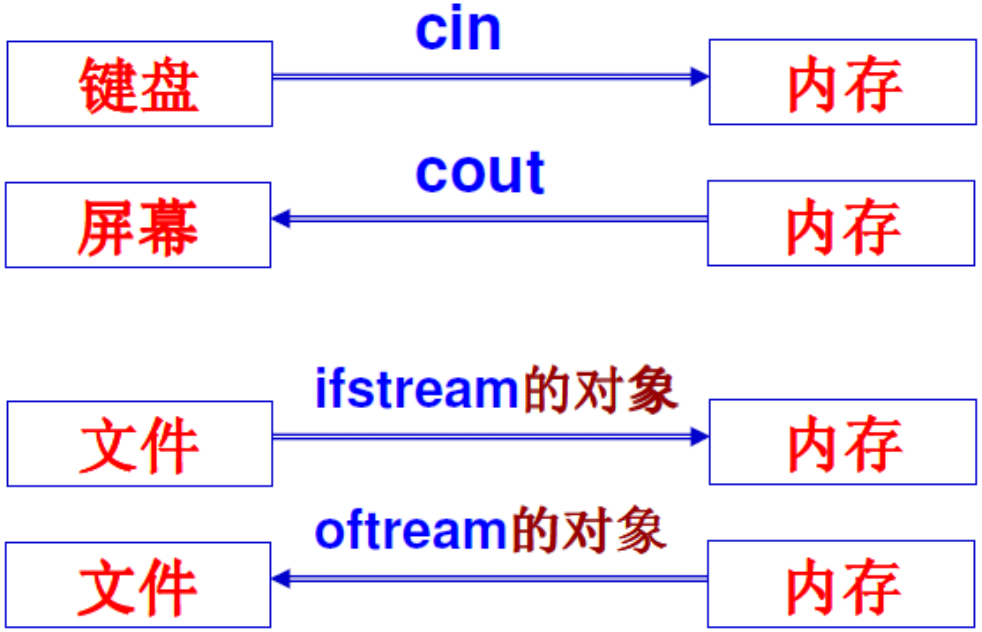
```
// wchar_t TYPEDEFS
typedef basic_ios< wchar_t, char_traits<wchar_t> > wios;
typedef basic_istream< wchar_t, char_traits<wchar_t> > wistream;
typedef basic_ostream< wchar_t, char_traits<wchar_t> > wostream;
typedef basic_iostream< wchar_t, char_traits<wchar_t> > wiostream;
typedef basic_ifstream< wchar_t, char_traits<wchar_t> > wifstream;
typedef basic_ofstream< wchar_t, char_traits<wchar_t> > wofstream;
typedef basic_fstream< wchar_t, char_traits<wchar_t> > wfstream;
```

14.3 Files and Streams





14.3 Files and Streams



- 主要差异: 文件操作时需定义**ifstream / ofstream**对象, 以指定所具体操作的文件和操作相关的参数

14.3 Files and Streams

- 头文件
  - • #include <iostream>
  - • #include <fstream>
- <fstream>
  - 包括三种类模板的定义
    - basic_ifstream (for file input)
    - basic_ofstream (for file output)
    - basic_fstream (for file input and output)
  - 提供了处理char字符流的类模板特化定义
    - ifstream: 从文件中输入字符(读文件)
    - ofstream: 向文件输出字符(写文件)
    - fstream: 支持文件中字符的输入和输出

Topics

- 14.1 Introduction
- 14.2 The Data Hierarchy
- 14.3 Files and Streams
- **14.4 Creating a Sequential File**
- 14.5 Reading Data from a Sequential File
- 14.6 Random-Access Files

14.4 Creating a Sequential File

- 创建ofstream对象
- (1) 创建流类对象的同时打开文件

ofstream(const char* filename, int mode)

- filename: 路径 + 文件名(含后缀)
  - "c:\\clients.dat"
  - "clients.dat" // 当前路径
- mode:
  - using std::ios;
  - ios::out ofstream的缺省模式
    - ① 若文件存在, 则打开并丢弃现有数据
    - ② 若文件不存在, 则创建
  - ios::app 向文件末尾添加数据

14.4 Creating a Sequential File

- 创建ofstream对象
- (2) 先创建对象, 后打开文件
- • 缺省构造函数+ open成员函数
- • open与前述构造函数的参数相同

```
ofstream outClientFile;
outClientFile.open("clients.dat", ios::out);
```

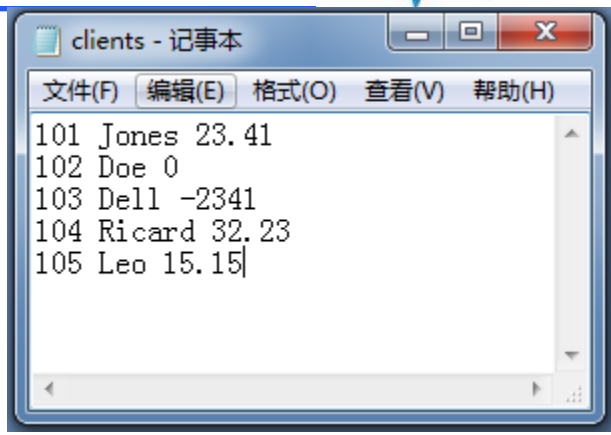
14.4 Creating a Sequential File

- 文件的写操作(与cout相似)

  outClientFile << account << ‘ ‘

  << name << ‘ ‘

  << balance << endl;



- 为何写数据的时候需要空格分割?

```cpp
// Fig. 17.4: Fig17_04.cpp
// Create a sequential file.
#include <iostream>
using std::cerr;
using std::cin;
using std::cout;
using std::endl;
using std::ios;

#include <fstream> // file stream
using std::ofstream; // output file stream

#include <cstdlib>
using std::exit; // exit function prototype

int main()
{
   // ofstream constructor opens file
   ofstream outClientFile( "clients.dat", ios::out );

   // exit program if unable to create file
   if ( !outClientFile ) // overloaded ! operator
   {
      cerr << "File could not be opened" << endl;
      exit( 1 );
   } // end if

   cout << "Enter the account, name, and balance." << endl
      << "Enter end-of-file to end input.\n? ";
```

Outline

Fig17_04.cpp

(1 of 2)

```
   int account;
   char name[ 30 ];
   double balance;

   // read account, name and balance from cin, then place in file
   while ( cin >> account >> name >> balance )
   {
      outClientFile << account << ' ' << name << ' ' << balance << endl;
      cout << "? ";
   } // end while

   return 0; // ofstream destructor closes file
} // end main
```

```
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
```

Outline

Fig17_04.cpp

(2 of 2)

14.4 Creating a Sequential File

- 当流引用作为condition使用, 会自动隐含调用void* 重载运算符, 以将其转换成指针

- 根据上一次流操作是否成功, 得到:
  - null指针: 操作失败, 则0, 即False
  - Non-null指针: 操作成功, 则非0, 即True
- 一种常见的流读取失败是读到了EOF标记,此时condition即为False

14.4 Creating a Sequential File

- 文件的关闭
- • ofstream析构时会自动关闭文件
- • 建议当文件不再需要使用时, 显式调用close成员函数关闭

```
ofstream outClientFile;
outClientFile.open("a.dat", ios::out);
......
outClientFile.close();
outClientFile.open("b.dat", ios::out);
......
outClientFile.close();
```

Topics

- 14.1 Introduction
- 14.2 The Data Hierarchy
- 14.3 Files and Streams
- 14.4 Creating a Sequential File
- **14.5 Reading Data from a Sequential File**
- 14.6 Random-Access Files

14.5 Reading Data from a Sequential File

- 创建ifstream 对象
- (1) 创建流类对象的同时打开文件

  ```
  ifstream inClientFile( "clients.dat", ios::in );
  ```

- • ios::in – 缺省模式, 仅能从文件读取数据(最小权限原则)

- (2) 创建对象, 后打开文件

  ```
  ifstream inClientFile;
  inClientFile.open("clients.dat", ios::in );
  ```

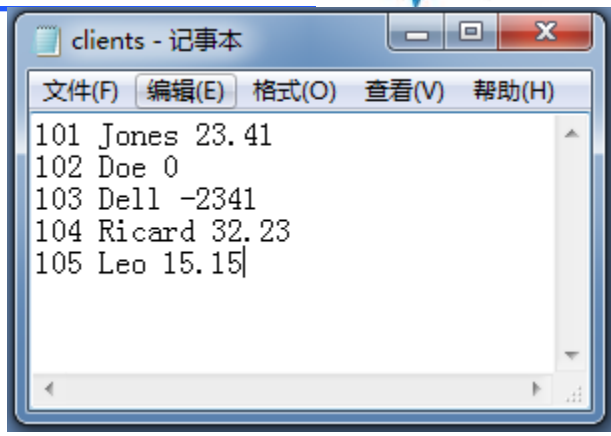
14.5 Reading Data from a Sequential File

- 文件的读操作(与cin相似)

  inClientFile >> account

  >> name

  >> balance;



- 为何写文件的时候需要空格分割?

  读文件时, 需要空白符分割数据!

```cpp
// Fig. 17.7: Fig17_07.cpp
// Reading and printing a sequential file.
#include <iostream>
using std::cerr;
using std::cout;
using std::endl;
using std::fixed;
using std::ios;
using std::left;
using std::right;
using std::showpoint;

#include <fstream> // file stream
using std::ifstream; // input file stream

#include <iomanip>
using std::setw;
using std::setprecision;

#include <string>
using std::string;

#include <cstdlib>
using std::exit; // exit function prototype

void outputLine( int, const string, double ); // prototype
```

Outline

Fig17_07.cpp

(1 of 3)

```cpp

int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.dat", ios::in );

   // exit program if ifstream could not open file
   if ( !inClientFile )
   {
      cerr << "File could not be opened" << endl;
      exit( 1 );
   } // end if

   int account;
   char name[ 30 ];
   double balance;

   cout << left << setw( 10 ) << "Account" << setw( 13 )
      << "Name" << "Balance" << endl << fixed << showpoint;

   // display each record in file
   while ( inClientFile >> account >> name >> balance )
      outputLine( account, name, balance );

   return 0; // ifstream destructor closes the file
} // end main
```

Outline

Fig17_07.cpp

(2 of 3)

```
// display single record from file
void outputLine( int account, const string name, double balance )
{
   cout << left << setw( 10 ) << account << setw( 13 ) << name
      << setw( 7 ) << setprecision( 2 ) << right << balance << endl;
} // end function outputLine
```

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

Outline

Fig17_07.cpp

(3 of 3)

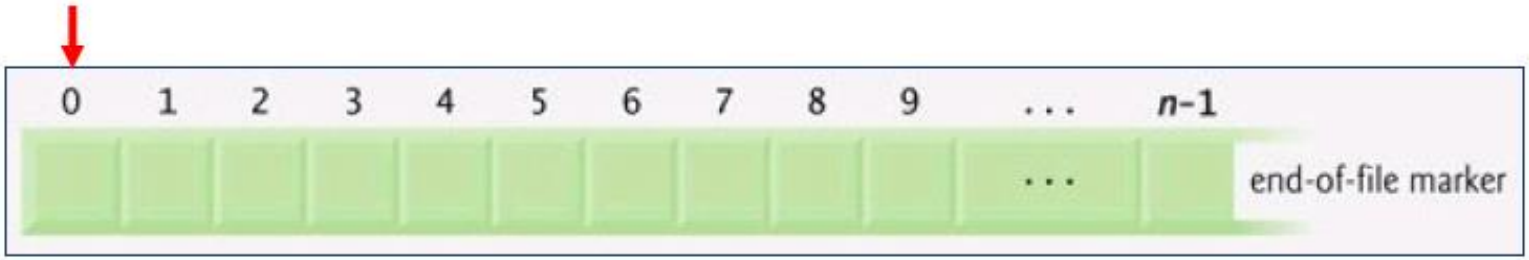
14.5 Reading Data from a Sequential File

- 已完成功能:

  打开并顺序读取文件内容, 直到文件结束

- 文件位置指针: 指向下一个将要读(get指针)或写(put指针)的字节位置

- 问题: 如何重新定位文件位置指针?



14.5 Reading Data from a Sequential File

- istream成员函数

  seekg(streamoff, ios::seek_dir);

  tellg();// 返回当前get指针位置

- ostream成员函数

  seekp(streamoff, ios::seek_dir);

  tellp();// 返回当前put指针位置

14.5 Reading Data from a Sequential File

- 文件位置指针的偏移量和Seek Direction
  - ios::beg – the default
    - Positioning relative to the beginning
  - ios::cur
    - Positioning relative to the current position
  - ios::end
    - Positioning relative to the end

14.5 Reading Data from a Sequential File

- position to the nth byte of fileObject (assumes ios::beg)

  ```
  fileObject.seekg( n );
  ```

- position n bytes forward in fileObject

  ```
  fileObject.seekg( n, ios::cur );
  ```

- position n bytes back from end of fileObject

  ```
  fileObject.seekg( n, ios::end );
  ```

- position at end of fileObject

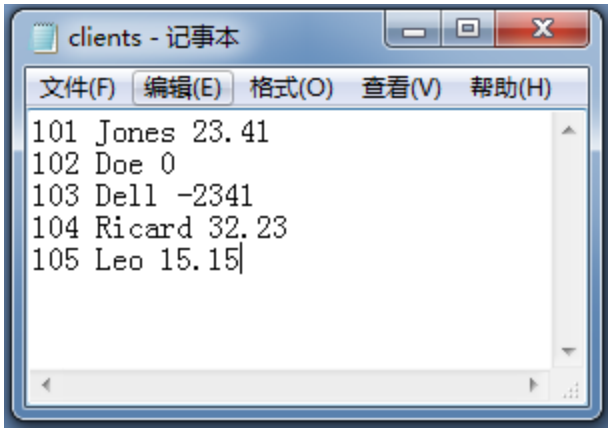
  ```
  fileObject.seekg( 0, ios::end );
  ```

- assigns the "get" file-position pointer value to variable location of type long:

  ```
  location = fileObject.tellg();
  ```

14.5 Reading Data from a Sequential File

- 信用卡账户管理
  - Zero balance: 没有消费，没有存款
    - 1 balance == 0
  - Credit balance: 有存款
    - 2 balance < 0
  - Debit balance: 有欠款
    - 3 balance > 0



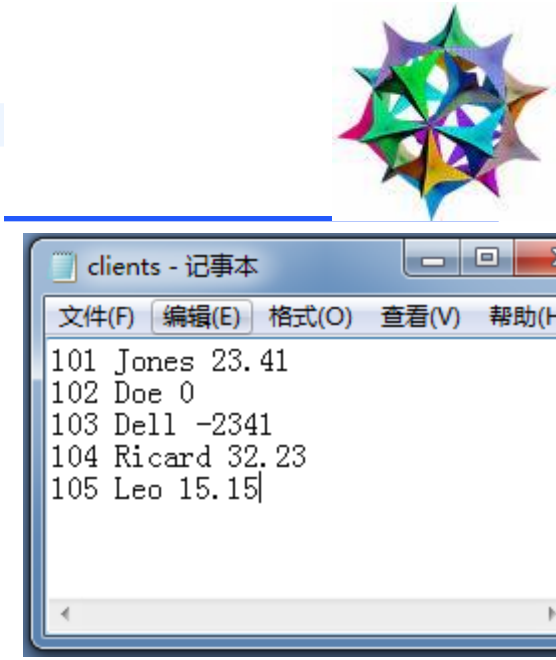
- inClientFile.clear(); // reset eof for next input

  inClientFile.seekg(0); // reposition to beginning of file

```
enum RequestType { ZERO_BALANCE = 1, CREDIT_BALANCE, DEBIT_BALANCE, END };
int getRequest();
bool shouldDisplay( int, double );
void outputLine( int, const string &, double );
```

```
int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.txt", ios::in );

   // exit program if ifstream could not open file
   if ( !inClientFile )
   {
      cerr << "File could not be opened" << endl;
      exit( EXIT_FAILURE );
   } // end if

   int account; // the account number
   string name; // the account owner's name
   double balance; // the account balance

   // get user's request (e.g., zero, credit or debit balance)
   int 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;
   } // end switch

   // read account, name and balance from file
   inClientFile >> account >> name >> balance;

   // display file contents (until eof)
   while ( !inClientFile.eof() )
   {
      // display record
      if ( shouldDisplay( request, balance ) )
         outputLine( account, name, balance );

      // read account, name and balance from file
      inClientFile >> account >> name >> balance;
   } // end inner while

   inClientFile.clear();     // reset eof for next input
   inClientFile.seekg( 0 ); // reposition to beginning of file
   request = getRequest();  // get additional request from user
} // end outer while
```



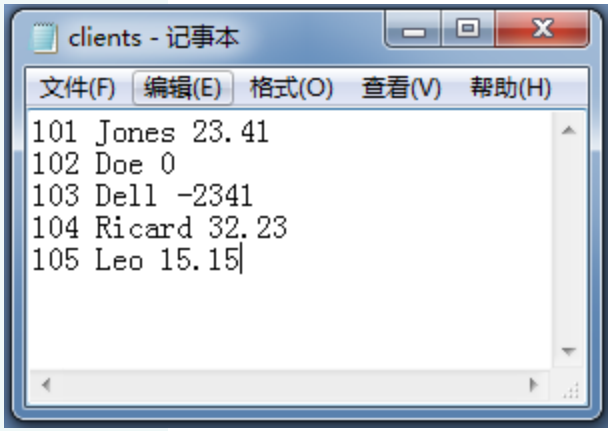
```
int getRequest()
{
   int request; // request from user

   // display request options
   cout << "\nEnter request" << endl
      << " 1 - List accounts with zero balances" << endl
      << " 2 - List accounts with credit balances" << endl
      << " 3 - List accounts with debit balances" << endl
      << " 4 - End of run" << fixed << showpoint;

   do // input user request
   {
      cout << "\n? ";
      cin >> request;
   } while ( request < ZERO_BALANCE && reque

   return request;
} // end function getRequest
```

```
// determine whether to display given record
bool shouldDisplay( int type, double balance )
{
   // determine whether to display zero balances
   if ( type == ZERO_BALANCE && balance == 0 )
      return true;

   // determine whether to display credit balances
   if ( type == CREDIT_BALANCE && balance < 0 )
      return true;

   // determine whether to display debit balances
   if ( type == DEBIT_BALANCE && balance > 0 )
      return true;
```

```
void outputLine( int account, const string &name, double balance )
{
   cout << left << setw( 10 ) << account << setw( 13 ) << name
      << setw( 7 ) << setprecision( 2 ) << right << balance << endl;
} // end function outputLine
```

dDisplay

```
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

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

Accounts with credit balances:
400       Stone         -42.16

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

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

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



Topics

- 14.1 Introduction
- 14.2 The Data Hierarchy
- 14.3 Files and Streams
- 14.4 Creating a Sequential File
- 14.5 Reading Data from a Sequential File
- **14.6 Random-Access Files**

14.6 Random-Access Files

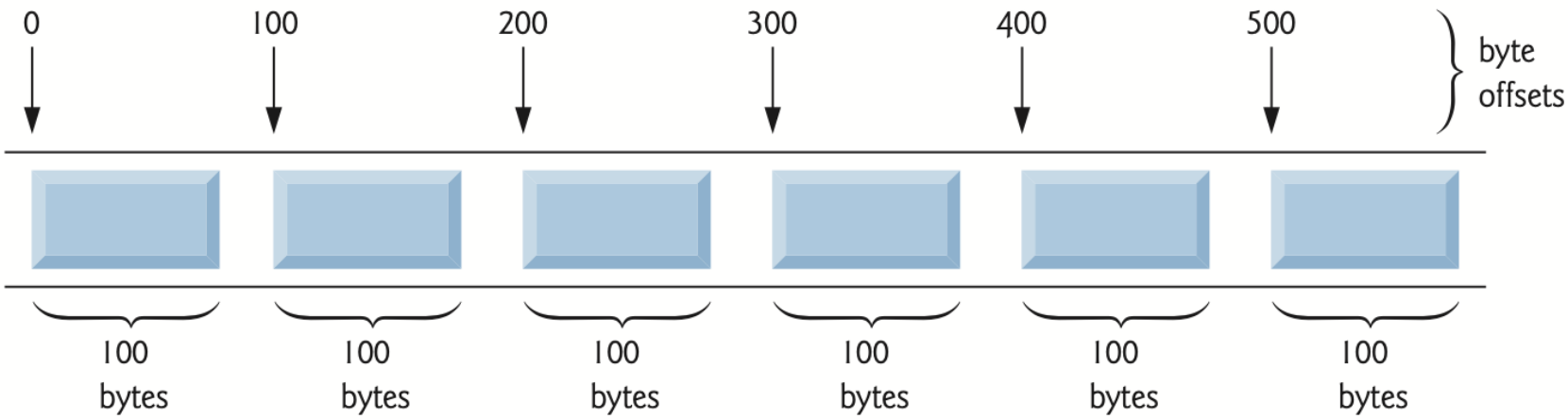


Fig. 14.8 | C++ view of a random-access file.

```
outFile << number;
```

```
outFile.write( reinterpret_cast< const char * >( &number ),
   sizeof( number ) );
```

```
class ClientData
{
public:
   // default ClientData constructor
   ClientData( int = 0, const std::string & = "",
      const std::string & = "", double = 0.0 );

   // accessor functions for accountNumber
   void setAccountNumber( int );
   int getAccountNumber() const;

   // accessor functions for lastName
   void setLastName( const std::string & );
   std::string getLastName() const;

   // accessor functions for firstName
   void setFirstName( const std::string & );
   std::string getFirstName() const;

   // accessor functions for balance
   void setBalance( double );
   double getBalance() const;
private:
   int accountNumber;
   char lastName[ 15 ];
   char firstName[ 10 ];
   double balance;
}; // end class ClientData
```

```
fstream outCredit( "credit.dat", ios::in | ios::out | ios::binary );
// require user to specify account number
ClientData client;
cin >> accountNumber;

// user enters information, which is copied into f
while ( accountNumber > 0 && accountNumber <= 100
{
   // user enters last name, first name and balanc
   cout << "Enter lastname, firstname, balance\n? 
   cin >> lastName;
   cin >> firstName;
   cin >> balance;

   // set record accountNumber, lastName, firstNar
   client.setAccountNumber( accountNumber );
   client.setLastName( lastName );
   client.setFirstName( firstName );
   client.setBalance( balance );

   // seek position in file of user-specified record
   outCredit.seekp( ( client.getAccountNumber() - 1 ) *
      sizeof( ClientData ) );

   // write user-specified information in file
   outCredit.write( reinterpret_cast< const char * >( &client ),
      sizeof( ClientData ) );
   // enter user to enter another account
   cout << "Enter account number\n? ";
   cin >> accountNumber;
} // end while
```

```
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
```

```
ifstream inCredit( "credit.dat", ios::in | ios::binary );
ClientData client; // create record

// read first record from file
inCredit.read( reinterpret_cast< char * >( &client ),
   sizeof( ClientData ) );

// read all records from file
while ( inCredit && !inCredit.eof() )
{
   // display record
   if ( client.getAccountNumber() != 0 )
      outputLine( cout, client );

   // read next from file
   inCredit.read( reinterpret_cast< char * >( &client ),
      sizeof( ClientData ) );
} // end while
void outputLine( ostream &output, const ClientData &record )
{
   output << left << set
      << setw( 16 ) << r
      << setw( 11 ) << r
      << setw( 10 ) << s
      << showpoint << re
} // end function outputLine
```

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

14.6 Random-Access Files

- Maintain a bank's account information (100 records)
  - Update existing accounts (update balance)
  - Add new accounts
  - Delete accounts
  - Store a formatted listing of all current accounts in a text file.

14.6 Random-Access Files

```
int enterChoice();
void createTextFile( fstream& );
void updateRecord( fstream& );
void newRecord( fstream& );
void deleteRecord( fstream& );
void outputLine( ostream&, const ClientData & );
int getAccount( const char * const );

enum Choices { PRINT = 1, UPDATE, NEW, DELETE, END };
```

14.6 Random-Access Files

```
fstream inOutCredit( "credit.dat", ios::in | ios::out | ios::binary );
int choice; // store user choice
// enable user to specify action
while ( ( choice = enterChoice() ) != END )
{
   switch ( choice )
   {
      case PRINT: // create text file from record file
         createTextFile( inOutCredit );
         break;
      case UPDATE: // update record
         updateRecord( inOutCredit );
         break;
      case NEW: // create record
         newRecord( inOutCredit );
         break;
      case DELETE: // delete existing record
         deleteRecord( inOutCredit );
         break;
      default: // display error if user does not select valid choice
         cerr << "Incorrect choice" << endl;
         break;
   } // end switch
   inOutCredit.clear(); // reset end-of-file indicator
} // end while
```

14.6 Rando

```
void outputLine( ostream &output, const ClientData &record )
{
   output << left << setw( 10 ) << record.getAccountNumber()
      << setw( 16 ) << record.getLastName()
      << setw( 11 ) << record.getFirstName()
      << setw( 10 ) << setprecision( 2 ) << right << fixed
      << showpoint << record.getBalance() << endl;
} // end function outputLine
```

```
void createTextFile( fstream &readFromFile )
{
   // create text file
   ofstream outPrintFile( "print.txt", ios::out );
   // output column heads
   outPrintFile << left << setw( 10 ) << "Account" << setw( 16 )
      << "Last Name" << setw( 11 ) << "First Name" << right
      << setw( 10 ) << "Balance" << endl;
   // set file-position pointer to beginning of readFromFile
   readFromFile.seekg( 0 );
   // read first record from record file
   ClientData client;
   readFromFile.read( reinterpret_cast< char * >( &client ),
      sizeof( ClientData ) );
   // copy all records from record file into text file
   while ( !readFromFile.eof() )
   {
      // write single record to text file
      if ( client.getAccountNumber() != 0 ) // skip empty records
         outputLine( outPrintFile, client );

      // read next record from record file
      readFromFile.read( reinterpret_cast< char * >( &client ),
         sizeof( ClientData ) );
   } // end while
} // end function createTextFile
```

s Files

```
void updateRecord( fstream &updateFile )
{
   // obtain number of account to update
   int accountNumber = getAccount( "Enter account to update" );
   // move file-position pointer to correct record in file
   updateFile.seekg( ( accountNumber - 1 ) * sizeof( ClientData ) );
   // read first record from file
   ClientData client;
   updateFile.read( reinterpret_cast< char * >( &client ),
      sizeof( ClientData ) );
   // update record
   if ( client.getAccountNumber() != 0 )
   {
      outputLine( cout, client ); // display the record
      // request user to specify transaction
      cout << "\nEnter charge (+) or payment (-): ";
      double transaction; // charge or payment
      cin >> transaction;
      // update record balance
      double oldBalance = client.getBalance();
      client.setBalance( oldBalance + transaction );
      outputLine( cout, client ); // display the record
      // move file-position pointer to correct record in file
      updateFile.seekp( ( accountNumber - 1 ) * sizeof( ClientData ) );
      // write updated record over old record in file
      updateFile.write( reinterpret_cast< const char * >( &client ),
         sizeof( ClientData ) );
   } // end if
   else // display error if account does not exist
      cerr << "Account #" << accountNumber
         << " has no information." << endl;
} // end function updateRecord
```

ss Files

```
void newRecord( fstream &insertInFile )
{
   // obtain number of account to create
   int accountNumber = getAccount( "Enter new account number" );
   // move file-position pointer to correct record in file
   insertInFile.seekg( ( accountNumber - 1 ) * sizeof( ClientData ) );
   // read record from file
   ClientData client;
   insertInFile.read( reinterpret_cast< char * >( &client ),
      sizeof( ClientData ) );
   // create record, if record does not previously exist
   if ( client.getAccountNumber() == 0 )
   {
      string lastName;
      string firstName;
      double balance;
      // user enters last name, first name and balance
      cout << "Enter lastname, firstname, balance\n? ";
      cin >> lastName;
      cin >> firstName;
      cin >> balance;
      // use values to populate account values
      client.setLastName( lastName );
      client.setFirstName( firstName );
      client.setBalance( balance );
      client.setAccountNumber( accountNumber );
      // move file-position pointer to correct record in file
      insertInFile.seekp( ( accountNumber - 1 ) * sizeof( ClientData ) );
      // insert record in file
      insertInFile.write( reinterpret_cast< const char * >( &client ),
         sizeof( ClientData ) );
   } // end if
   else // display error if account already exists
      cerr << "Account #" << accountNumber
         << " already contains information." << endl;
} // end function newRecord
```

```
void deleteRecord( fstream &deleteFromFile )
{
   // obtain number of account to delete
   int accountNumber = getAccount( "Enter account to delete" );
   // move file-position pointer to correct record in file
   deleteFromFile.seekg( ( accountNumber - 1 ) * sizeof( ClientData ) );
   // read record from file
   ClientData client;
   deleteFromFile.read( reinterpret_cast< char * >( &client ),
      sizeof( ClientData ) );
   // delete record, if record exists in file
   if ( client.getAccountNumber() != 0 )
   {
      ClientData blankClient; // create blank record
      // move file-position pointer to correct record in file
      deleteFromFile.seekp( ( accountNumber - 1 ) *
         sizeof( ClientData ) );
      // replace existing record with blank record
      deleteFromFile.write(
         reinterpret_cast< const char * >( &blankClient ),
         sizeof( ClientData ) );
      cout << "Account #" << accountNumber << " deleted.\n";
   } // end if
   else // display error if record does not exist
      cerr << "Account #" << accountNumber << " is empty.\n";
} // end deleteRecord
```

14.6 Random-Access Files

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

```
Enter account to update (1 - 100): 37
37          Barker          Doug              0.00

Enter charge (+) or payment (-): +87.99
37          Barker          Doug             87.99
```

```
Enter new account number (1 - 100): 22
Enter lastname, firstname, balance
? Johnston Sarah 247.45
```

Summary

- 三种文件流
  - • 文件输入流(ifstream)
  - • 文件输出流(ofstream)
  - • 文件输入/输出流(fstream)
- 文件处理步骤 (顺序/随机)
  - • 定义文件流对象
  - • 打开文件: open
  - • 读写文件
  - • 关闭文件: close

Homework

- 实验必选题目：

  14.10 (随机文件), 14.13 (顺序文件)
- 实验任选题目：

- 作业题目：