

Ngôn ngữ lập trình C++

Chương 7 – Ra vào dữ liệu

Chương 7 : Ra vào dữ liệu

Đề mục

- 7.1 Giới thiệu
- 7.2 Dòng – Stream
 - 7.2.2 Các file header thư viện iostream
 - 7.2.3 Các đối tượng và các lớp I/O
- 7.3 Xuất theo dòng
 - 7.3.1 Xuất các biến kiểu char*.
- 7.4 Nhập theo dòng
 - 7.4.1 Các thành viên get và getline
 - 7.4.2 Các thành viên peek, putback, và ignore
- 7.5 I/O không định dạng sử dụng read, write, và gcount
- 7.6 Giới thiệu về các stream manipulator
- 7.7 Các trạng thái lỗi của dòng
- 7.8 Đồng bộ một dòng ra và một dòng vào

Chương 7 : Ra vào dữ liệu

Đề mục (tiếp)

- 7.9 File và dòng (stream)
- 7.10 File truy nhập tuần tự
- 7.11 Các hàm định vị cho file truy nhập tuần tự
- 7.12 Các rắc rối khi cập nhật file truy nhập tuần tự
- 7.13 File truy nhập ngẫu nhiên
 - 7.13.1 Dữ liệu thô và dữ liệu định dạng
 - 7.13.2 Ghi file truy nhập ngẫu nhiên
 - 7.13.3 Ghi dữ liệu vào vị trí tùy ý trong file truy nhập ngẫu nhiên
 - 7.13.4 Đọc tuần tự dữ liệu từ file truy nhập ngẫu nhiên
- 7.14 Ví dụ: Chương trình quản lý giao dịch

7.1 Giới thiệu

- C++ I/O
 - Hướng đối tượng
 - sử dụng tham chiếu, chồng hàm, chồng toán tử
 - An toàn về các kiểu dữ liệu
 - nhạy cảm với kiểu dữ liệu
 - báo lỗi nếu kiểu không khớp
 - có thể dùng cho cả kiểu người dùng tự định nghĩa và các kiểu chuẩn
 - làm cho C++ có khả năng mở rộng

7.2 Dòng - Stream

- Stream – dòng:
 - chuỗi byte, kết thúc bởi ký hiệu *end_of_file*
 - Input: từ bàn phím, đĩa... vào bộ nhớ
 - Output: từ bộ nhớ ra màn hình, máy in...
 - file cũng được coi là một dòng
- Các dòng cổ điển
 - vào/ra **char** (1 byte)
 - các ký tự giới hạn bảng mã ASCII
- Các thư viện dòng chuẩn
 - Một số ngôn ngữ cần các bảng chữ cái đặc biệt
 - Unicode
 - kiểu ký tự **wchar_t**
 - Có thể thực hiện I/O với các ký tự Unicode

7.2.2 Các file header thư viện iostream

- thư viện **iostream**
 - có các header file với hàng trăm chức năng vào/ra
 - **<iostream.h>**
 - vào chuẩn – Standard input (**cin**)
 - ra chuẩn – Standard output (**cout**)
 - dòng báo lỗi không có bộ nhớ đệm – Unbuffered error (**cerr**)
 - dòng báo lỗi có dùng bộ nhớ đệm – Buffered error (**clog**)
 - **<iomanip.h>**
 - các stream manipulator (có tham số) để định dạng I/O
 - **<fstream.h>**
 - các thao tác xử lý file

7.2.3 Các đối tượng và các lớp I/O

- **<< và >>**
 - các toán tử chèn và tách dòng
- **cin**
 - đối tượng **istream**
 - nối với input chuẩn (thường là bàn phím)
 - **cin >> grade;**
 - trình biên dịch tự xác định kiểu của **grade**
 - gọi toán tử thích hợp (đã được định nghĩa chồng)
 - không cần thông tin thêm về kiểu dữ liệu

7.2.3 Các đối tượng và các lớp I/O

- **cout**

- đối tượng **ostream**
- nối với output chuẩn (thường là màn hình)
- **cout << grade;**
 - cũng như với **cin**, không cần thêm thông tin về kiểu

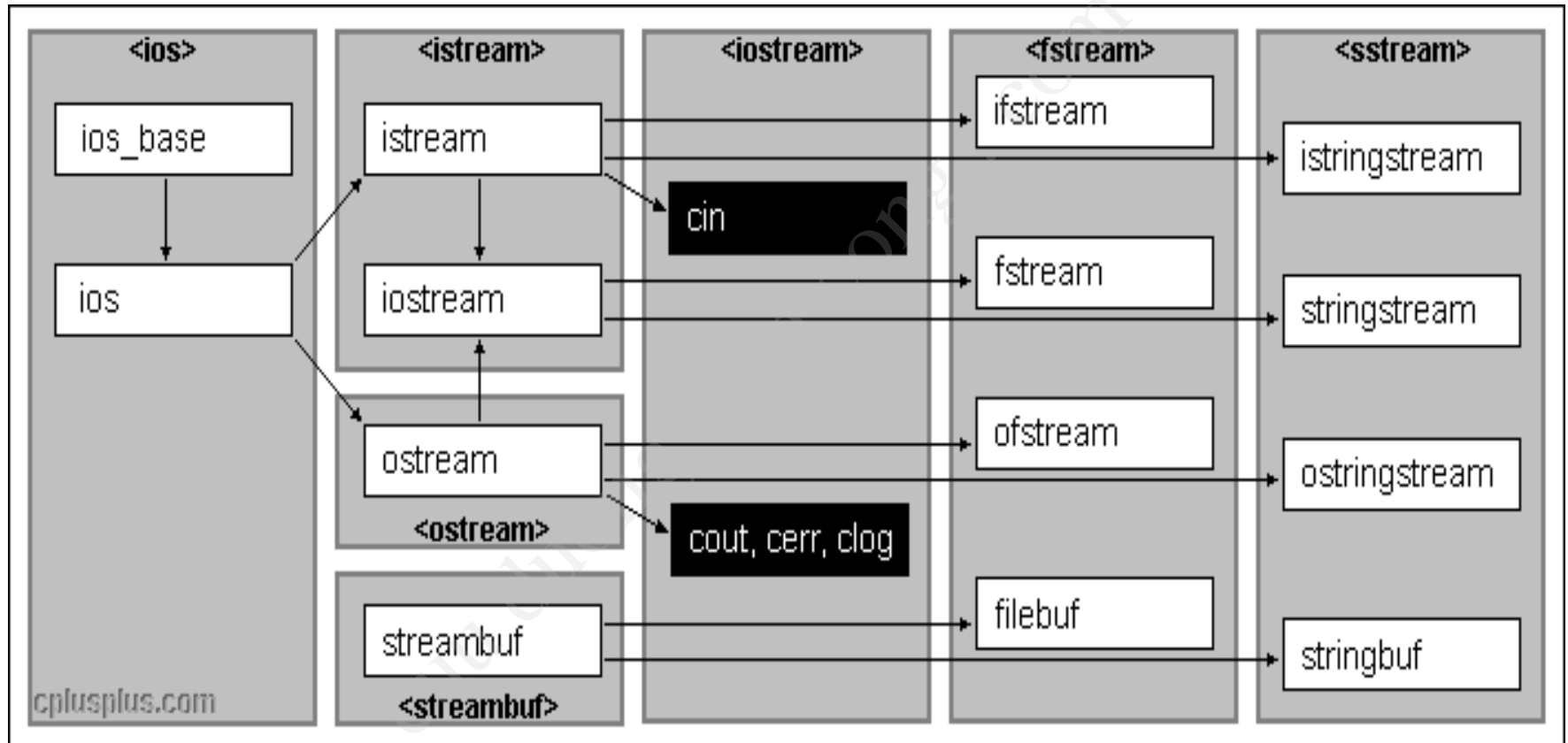
- **cerr, clog**

- các đối tượng **ostream**
- nối với thiết bị báo lỗi chuẩn
- **cerr** xuất ngay lập tức
- **clog** sử dụng bộ nhớ đệm trước khi xuất
 - xuất khi bộ nhớ đệm đầy hoặc khi được xả (flushed)
 - ưu điểm hiệu năng (giải thích tại môn Hệ điều hành)

7.2.3 Các đối tượng và các lớp I/O

- C++ xử lý file tương tự
 - Các kiểu đối tượng dành cho xuất nhập **char**
 - **ifstream** (file input)
 - **ofstream** (file output)
 - **fstream** (file I/O)

7.2.3 Các đối tượng và các lớp I/O



7.3 Xuất theo dòng

- Output
 - sử dụng **ostream**
 - định dạng và không định dạng dữ liệu xuất
 - dành cho các kiểu dữ liệu chuẩn (<<)
 - các ký tự (hàm **put**)
 - các kiểu số nguyên (thập phân, bát phân, cơ số 16)
 - các số chấm động
 - quy định độ chính xác, vị trí dấu chấm, ký hiệu khoa học
 - dữ liệu được căn lề, chèn ký tự trống
 - điều khiển chữ hoa/chữ thường

7.3.1 Xuất các biến kiểu char *

- C++ tự động xác định kiểu dữ liệu
 - in giá trị của một **char ***
 - địa chỉ bộ nhớ của ký tự đầu tiên
- Rắc rối
 - toán tử << được định nghĩa chồng để in xâu kết thúc bằng null
 - cách giải quyết: đổi thành **void ***
 - sử dụng khi in giá trị của một con trỏ
 - in dưới dạng một số cơ số 16

```

1 // Fig. 12.3: fig12_03.cpp
2 // Printing the address stored in a char * variable.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 int main()
9 {
10     char *word = "test";
11
12     // display value of char *, then display value of char *
13     // static_cast to void *
14     cout << "Value of word is: " << word << endl
15         << "Value of static_cast< void * >( word ) is: "
16         << static_cast< void * >( word ) << endl;
17
18     return 0;
19
20 } // end main

```

Để in giá trị của con trỏ, ta phải đổi sang kiểu **void ***. Nếu không, chương trình sẽ in chuỗi ký tự.

fig12_03.cpp
(1 of 1)

fig12_03.cpp
output (1 of 1)

```

Value of word is: test
Value of static_cast< void *>( word ) is: 0046C070

```

7.3.2 Xuất ký tự bằng hàm thành viên put

- hàm **put**
 - in các ký tự
 - **cout.put('A');**
 - Có thể gọi liên
 - **cout.put('A').put('\\n');**
 - Toán tử dấu chấm (.) được tính từ trái sang phải
 - Có thể sử dụng giá trị bằng số (mã ASCII)
 - **cout.put(65);**
 - in ký tự 'A'

7.4 Nhập theo dòng

- dữ liệu vào có định dạng và không định dạng
 - **istream**
- toán tử **>>**
 - Thường bỏ qua các ký tự trắng (blank, tab, newline)
 - Có thể thay đổi
 - Trả về **0** khi gặp EOF
 - nếu không, trả về tham chiếu tới **istream**
 - **cin >> grade**
 - các bit trạng thái được bật nếu xảy ra lỗi
 - chi tiết sẽ nói đến sau

7.4.1 Các hàm thành viên get và getline

- hàm **get**
 - **cin.get()**
 - trả về một ký tự từ dòng (kể cả ký tự trắng)
 - trả về **EOF** nếu gặp end-of-file
- End-of-file
 - đánh dấu kết thúc dữ liệu vào
 - *ctrl-z* tại DOS/Windows
 - *ctrl-d* tại UNIX và Mac
 - **cin.eof()**
 - trả về **1 (true)** nếu đã gặp EOF


```
1 // Fig. 12.4: fig12_04.cpp
2 // Using member functions get, put and eof.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 int main()
10 {
11     int character; // use int, because char cannot represent EOF
12
13     // prompt user to enter line of text
14     cout << "Before input, cin.eof() is " << cin.eof() << endl
15          << "Enter a sentence followed by end-of-file:" << endl;
16
17     // use get to read each character; use put to display it
18     while ( ( character = cin.get() ) != EOF )
19         cout.put( character );
20
21     // display end-of-file character
22     cout << "\nEOF in this system is: " << character << endl;
23     cout << "After input, cin.eof() is " << cin.eof() << endl;
24
25     return 0;
```

Hàm **get** (không có đối số) trả về đúng một ký tự nhập vào, trừ khi gặp **EOF**.

26

27 } // end main

Before input, cin.eof() is 0

Enter a sentence followed by end-of-file:

Testing the get and put member functions

Testing the get and put member functions

^Z

EOF in this system is: -1

After input cin.eof() is 1

fig12_04.cpp

(2 of 2)

fig12_04.cpp

output (1 of 1)

7.4.1 Các hàm thành viên `get` và `getline`

- **`get (charRef)`**
 - đối số là tham chiếu ký tự
 - đọc một ký tự, lưu vào **`charRef`**
 - trả về tham chiếu tới **`istream`**
 - nếu hết file, trả về **`-1`**
- **`get (charArray, size, delimiter)`**
 - đọc cho đến khi được **`size-1`** ký tự, hoặc đến khi gặp ký tự phân cách
 - phân cách mặc định **`'\n'`**
 - ký tự phân cách được để lại dòng nhập
 - có thể loại bỏ bằng **`cin.get()`** hoặc **`cin.ignore()`**
 - tự động thêm null vào cuối để kết thúc mảng

fig12_05.cpp
(1 of 2)

```
1 // Fig. 12.5: fig12_05.cpp
2 // Contrasting input of a string via cin and cin.get.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 int main()
10 {
11     // create two char arrays, each with 80 elements
12     const int SIZE = 80;
13     char buffer1[ SIZE ];
14     char buffer2[ SIZE ];
15
16     // use cin to input characters into buffer1
17     cout << "Enter a sentence:" << endl;
18     cin >> buffer1;
19
20     // display buffer1 contents
21     cout << "\nThe string read with cin was:" << endl
22          << buffer1 << endl << endl;
23
24     // use cin.get to input characters into buffer2
25     cin.get( buffer2, SIZE );
```

cin sẽ chỉ đọc cho đến ký tự trắng đầu tiên.

Không chỉ ra ký tự phân cách, do đó sẽ sử dụng phân cách mặc định (\n).

```
26
27 // display buffer2 contents
28 cout << "The string read with cin.get was:" << endl
29     << buffer2 << endl;
30
31 return 0;
32
33 } // end main
```

fig12_05.cpp
(2 of 2)

fig12_05.cpp
output (1 of 1)

Enter a sentence:
Contrasting string input with cin and cin.get

The string read with cin was:
Contrasting

The string read with cin.get was:
string input with cin and cin.get

7.4.1 Các hàm thành viên `get` và `getline`

- **`getline(array, size, delimiter)`**
 - như phiên bản 3 tham số của **`get`**
 - đọc **`size-1`** ký tự, hoặc cho đến khi thấy ký tự phân cách
 - mặc định `\n`
 - loại bỏ ký tự phân cách khỏi dòng vào
 - đặt ký tự null vào cuối mảng

```

1  // Fig. 12.6: fig12_06.cpp
2  // Inputting characters using cin member function getline.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  int main()
10 {
11     const int SIZE = 80;
12     char buffer[ SIZE ]; // create array of 80 characters
13
14     // input characters in buffer via cin function getline
15     cout << "Enter a sentence:" << endl;
16     cin.getline( buffer, SIZE );
17
18     // display buffer contents
19     cout << "\nThe sentence entered is:" << endl << buffer << endl;
20
21     return 0;
22
23 } // end main

```

Enter a sentence:
Using the getline member function

The sentence entered is:
Using the getline member function

p

7.4.2 Các hàm thành viên **peek, putback và ignore** của istream

- **ignore ()**
 - lấy các ký tự khỏi dòng (mặc định là 1 ký tự)
 - dừng khi gặp ký tự phân cách
 - phân cách mặc định là **EOF**
- **putback ()**
 - đẩy ký tự vừa đọc được bằng **get ()** trở lại dòng
- **peek ()**
 - trả về ký tự tiếp theo trong dòng nhưng không lấy ra khỏi dòng

7.5 I/O không định dạng sử dụng **read**, **write** và **gcount**

- I/O không định dạng
 - **read** (hàm thành viên của **istream**)
 - đọc các byte thô vào mảng char
 - nếu đọc được không đủ số ký tự, đặt **failbit**
 - **gcount()** trả về số ký tự đã đọc được tại lần gọi gần nhất
 - **write** (hàm thành viên của **ostream**)
 - xuất các byte từ mảng char
 - dừng khi gặp ký tự null
- ```
char buffer[] = "HAPPY BIRTHDAY";
cout.write(buffer, 10);
```
- xuất 10 char đầu tiên

```

1 // Fig. 12.7: fig12_07.cpp
2 // Unformatted I/O using read, gcount and write.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 int main()
10 {
11 const int SIZE = 80;
12 char buffer[SIZE]; // create array of 80 characters
13
14 // use function read to input characters
15 cout << "Enter a sentence:" << endl;
16 cin.read(buffer, 20);
17
18 // use functions write and gcount to display buffer characters
19 cout << endl << "The sentence entered was:" << endl;
20 cout.write(buffer, cin.gcount());
21 cout << endl;
22
23 return 0;
24
25 } // end main

```

đọc 20 ký tự từ dòng vào.  
Hiển thị số ký tự đọc được,  
sử dụng **write** và **gcount**.

Enter a sentence:  
Using the read, write, and gcount member functions  
The sentence entered was:  
Using the read, writ

## 7.6 Giới thiệu về các Stream Manipulator

- Stream manipulator thực hiện việc định dạng
  - đổi hệ cơ số (hex, oct, dec, setbase)
  - độ rộng (ký tự) in ra dành cho dữ liệu xuất (setw)
  - đặt số chữ số sau dấu phẩy (setprecision)
  - in/không in phần sau dấu phẩy của số nguyên (showpoint/noshowpoint)
  - căn trái/phải/giữa (left/right/internal)
  - ký tự chèn vào các vị trí còn trống (setfill)
  - định dạng khoa học/dấu chấm động (scientific/fixed)
  - in các giá trị bool dạng chữ(true,false)/số (boolalpha/noboolalpha)
  - ...

## 7.7 Các trạng thái lỗi của dòng

- Kiểm tra trạng thái dòng bằng qua các bit trạng thái
  - **eofbit** được bật khi gặp EOF
    - hàm **eof** trả về **true** nếu **eofbit** được bật
    - **cin.eof()**
  - **failbit** được bật khi dòng xảy ra lỗi
    - dữ liệu không mất, lỗi có thể khôi phục được
    - hàm **fail** trả về **true** nếu bit được bật
  - **badbit** được bật khi mất dữ liệu
    - thường là không khôi phục được
    - hàm **bad**
  - **goodbit** bật khi **badbit**, **failbit** và **eofbit** tắt
    - hàm **good**

## 7.7 Các trạng thái lỗi của dòng

- Các hàm thành viên
  - **rdstate()**
    - trả về trạng thái lỗi của dòng
    - có thể dùng để kiểm tra **goodbit**, **badbit**, v.v...
    - sử dụng **good()**, **bad()** thì hơn
  - **clear()**
    - đổi số mặc định là **goodbit**
    - đặt dòng trở về trạng thái tốt để có thể tiếp tục I/O
    - có thể truyền các giá trị khác
      - **cin.clear( ios::failbit )**
      - bật **failbit**


fig12\_22.cpp  
(1 of 2)

```
1 // Fig. 12.22: fig12_22.cpp
2 // Testing error states.
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7 using std::cin;
8
9 int main()
10 {
11 int integerValue;
12
13 // display results of cin functions
14 cout << "Before a bad input operation:"
15 << "\ncin.rdstate(): " << cin.rdstate()
16 << "\n cin.eof(): " << cin.eof()
17 << "\n cin.fail(): " << cin.fail()
18 << "\n cin.bad(): " << cin.bad()
19 << "\n cin.good(): " << cin.good()
20 << "\n\nExpects an integer, but enter a character: ";
21
22 cin >> integerValue; // enter character value
23 cout << endl;
24
```

in trạng thái ban đầu, sử dụng  
các hàm thành viên

fig12\_22.cpp  
(2 of 2)

```
25 // display results of cin functions after bad input
26 cout << "After a bad input operation:"
27 << "\ncin.rdstate(): " << cin.rdstate()
28 << "\n cin.eof(): " << cin.eof()
29 << "\n cin.fail(): " << cin.fail()
30 << "\n cin.bad(): " << cin.bad()
31 << "\n cin.good(): " << cin.good() << endl << endl;
32
33 cin.clear(); // clear stream
34
35 // display results of cin functions after clearing cin
36 cout << "After cin.clear()"
37 << "\ncin.fail(): " << cin.fail()
38 << "\ncin.good(): " << cin.good() << endl;
39
40 return 0;
41
42 } // end main
```



Before a bad input operation:

```
cin.rdstate(): 0
 cin.eof(): 0
cin.fail(): 0
 cin.bad(): 0
cin.good(): 1
```

Expects an integer, but enter a character: A

After a bad input operation:

```
cin.rdstate(): 2
 cin.eof(): 0
cin.fail(): 1
 cin.bad(): 0
cin.good(): 0
```

After cin.clear()

```
cin.fail(): 0
cin.good(): 1
```



## 7.8 Đồng bộ một dòng ra và một dòng vào

- Rắc rối với output có bộ nhớ đệm
  - chương trình tương tác (hỏi người sử dụng, người sử dụng trả lời)
  - lời yêu cầu cần hiện ra trước khi nhập
    - output trong bộ nhớ đệm chỉ hiện ra khi bộ nhớ đệm đầy hoặc được xả (flushed)
- hàm thành viên **tie**
  - đồng bộ hóa các dòng
  - Output hiện ra trước các input tiếp theo
  - được thực hiện tự động với **cin** và **cout**, nhưng có thể viết
    - **cin.tie( &cout )**
  - cần thực hiện tường minh đối với các cặp I/O khác
  - để bỏ đồng bộ
    - **istream.tie( 0 )**

## 7.9 File và dòng

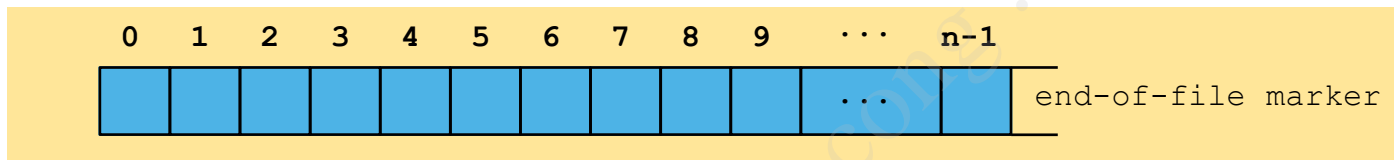
- Lưu trữ dữ liệu
  - Mảng, biến là dạng lưu trữ tạm thời
  - File là dạng lưu trữ bền vững
    - đĩa từ - magnetic disk, đĩa quang - optical disk, băng từ - tape
- trong chương này
  - tạo, cập nhật, xử lý file
  - truy nhập tuần tự (sequential access) và truy nhập ngẫu nhiên (random access)
  - xử lý có định dạng và xử lý thô (formatted and raw processing)

## 7.9 File và dòng

- Từ nhỏ nhất tới lớn nhất
  - Bit (binary digit)
  - Byte: 8 bits
    - Có thể lưu trữ 1 ký tự (**char**)
    - còn dùng để lưu Unicode dành cho bộ ký tự lớn hơn (**wchar\_t**)
  - Trường - Field: nhóm ký tự có nghĩa
    - tên
  - Bản ghi - Record: nhóm các trường có liên quan
    - **struct** hoặc **class** trong C++
    - trong hệ thống trả lương (payroll system): tên, mã, địa chỉ, lương
    - mỗi trường liên quan đến cùng một nhân viên.
    - Khóa của bản ghi - Record key: trường dùng để xác định duy nhất bản ghi
  - File: nhóm các bản ghi có liên quan
    - danh sách lương cho cả công ty
  - Cơ sở dữ liệu - Database: nhóm các file có liên quan
    - danh sách lương, các tài khoản, ...

## 7.9 File và dòng

- C++ coi file là một chuỗi byte - stream
  - Kết thúc bằng ký hiệu *end-of-file*



- Khi file mở
  - một đối tượng được tạo và kết nối với một dòng
  - tạo "đường liên lạc" từ đối tượng tới file
  - **cin**, **cout**, v.v... được tạo khi **<iostream>** được include
    - liên lạc giữa chương trình và file/thiết bị

## 7.10 File truy nhập tuần tự (sequential-access file)

- C++ không quy định cấu trúc file
  - Khái niệm "bản ghi" phải được cài đặt bởi lập trình viên
- Mở file
  - tạo đối tượng từ các lớp
    - **ifstream** (input only - chỉ đọc)
    - **ofstream** (output only - chỉ ghi)
    - **fstream** (I/O – file vừa đọc vừa ghi)
  - Constructor lấy tên file và kiểu mở file

```
ofstream outClientFile("filename", fileOpenMode);
```
  - Hoặc, tạo object trước rồi gắn với một file sau

```
ofstream outClientFile;
outClientFile.open("filename", fileOpenMode);
```

## 7.10 File truy nhập tuần tự

- Các kiểu mở file - File-open modes

| Mode                     | Description                                                                    |
|--------------------------|--------------------------------------------------------------------------------|
| <code>ios::app</code>    | Viết tiếp output vào cuối file.                                                |
| <code>ios::ate</code>    | Mở một file đã tồn tại ở cuối file. Nếu file không tồn tại, nó sẽ được tạo ra. |
| <code>ios::in</code>     | Mở file để đọc.                                                                |
| <code>ios::out</code>    | Mở file để ghi.                                                                |
| <code>ios::trunc</code>  | Loại bỏ nội dung file nếu nó tồn tại (mặc định là <code>ios::out</code> )      |
| <code>ios::binary</code> | Mở file nhị phân (i.e., không phải file text) để đọc/ghi.                      |

– theo mặc định, **ofstream** mở để ghi

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

## 7.10 File truy nhập tuần tự

- Các phép toán
  - Overloaded **operator!**
    - **!outClientFile** hoặc **!inClientfile**
    - Trả về nonzero (true) nếu **badbit** hoặc **failbit** bật
      - mở file không tồn tại để đọc, không có quyền mở
  - Overloaded **operator void\***
    - chuyển đổi đối tượng dòng thành con trỏ
    - 0 khi **failbit** hoặc **badbit** được bật, nếu không: nonzero
      - **failbit** bật khi gặp EOF
    - **while ( inClientFile >> myVariable )**
      - lặp cho đến khi gặp EOF
  - Ghi/đọc file (như **cout**, **cin**)
    - **outClientFile << myVariable**
    - **inClientFile >> myVariable**
  - Đóng file
    - **outClientFile.close()**
    - đóng tự động khi destructor được gọi

**fig14\_04.cpp**  
(1 of 2)

```
1 // Fig. 14.4: fig14_04.cpp
2 // Create a sequential file.
3 #include <iostream>
```

```
.....
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10
11 #include <fstream>
12
13 using std::ofstream;
```

Lưu ý các header file cần cho file I/O.

```
15 #include <cstdlib> // exit prototype
```

```
16
17 int main()
18 {
```

**ofstream** object được tạo và dùng để mở file **clients.dat**. Nếu file chưa tồn tại, nó sẽ được tạo.

```
19 // ofstream constructor opens file
20 ofstream outClientFile("clients.dat", ios::out);
```

```
21
22 // exit program if unable to create file
```

**!** operator dùng để kiểm tra xem có xảy ra lỗi khi mở file không.

```
23 if (!outClientFile) { // overloaded ! operator
24 cerr << "File could not be opened" << endl;
25 exit(1);
26
```

```
27 } // end if
```



fig14\_04.cpp  
(2 of 2)

```
28
29 cout << "Enter the account, name, and balance." << endl
30 << "Enter end-of-file to end input.\n? ";
31
32 int account;
33 char name[30];
34 double balance;
35
36 // read account, name and balance from cin, then place in file
37 while (cin >> account >> name >> balance) {
38 outClientFile << account << ' ' << name << ' ' << balance
39 << endl;
40 cout << "? ";
41
42 } // end while
43
44 return 0; // ofstream destructor closes file
45
46 }
```

**cin** được ngắt đôi thành 1 pointer.  
Khi gặp EOF, nó trả về 0 và vòng lặp dừng.

Ghi dữ liệu ra file như ghi ra một dòng chuẩn.

File đóng khi destructor của object được gọi.  
Có thể đóng một cách tường minh bằng cách gọi **close()**.

```
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
```

**fig14\_07.cpp**  
(1 of 3)

```
1 // Fig. 14.7: fig14_07.cpp
2 // Reading and printing a sequential file.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10 using std::left;
11 using std::right;
12 using std::fixed;
13 using std::showpoint;
14
15 #include <fstream>
16
17 using std::ifstream;
18
19 #include <iomanip>
20
21 using std::setw;
22 using std::setprecision;
23
24 #include <cstdlib> // exit prototype
25
26 void outputLine(int, const char * const, double);
27
```

```

28 int main()
29 {
30 // ifstream constructor opens the file
31 ifstream inClientFile("clients.dat", ios::in);
32
33 // exit program if ifstream could not open file
34 if (!inClientFile) {
35 cerr << "File could not be opened" << endl;
36 exit(1);
37
38 } // end if
39
40 int account;
41 char name[30];
42 double balance;
43
44 cout << left << setw(10) << "Account" << setw(13)
45 << "Name" << "Balance" << endl << fixed << showpoint;
46
47 // display each record in file
48 while (inClientFile >> account >> name >> balance)
49 outputLine(account, name, balance);
50
51 return 0; // ifstream destructor closes the file
52
53 } // end main

```

mở file để đọc và kiểm tra.

fig14\_07.cpp  
(2 of 3)

Đọc từ file đến khi gặp  
EOF.

fig14\_07.cpp  
(3 of 3)

fig14\_07.cpp  
output (1 of 1)

```
54
55 // display single record from file
56 void outputLine(int account, const char * const name,
57 double balance)
58 {
59 cout << left << setw(10) << account << setw(13) << name
60 << setw(7) << setprecision(2) << right << balance
61 << endl;
62
63 } // 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  |

## 7.11 Các hàm định vị cho file tuần tự

- con trỏ vị trí ghi số thứ tự của byte tiếp theo để đọc/ghi
- các hàm đặt lại vị trí của con trỏ:
  - **seekg** (đặt vị trí đọc cho lớp **istream**)
  - **seekp** (đặt vị trí ghi cho **ostream**)
  - **seekg** và **seekp** lấy các đối số là *offset* và *mốc*
    - Offset: số byte tương đối kể từ mốc
    - Mốc (**ios::beg** mặc định)
      - **ios::beg** - đầu file
      - **ios::cur** - vị trí hiện tại
      - **ios::end** - cuối file
- các hàm lấy vị trí hiện tại của con trỏ:
  - **tellg** và **tellp**

## 7.11 Các hàm định vị cho file tuần tự

- Ví dụ
  - **fileObject.seekg(0)**
    - đến đầu file (vị trí 0), mặc định đối số thứ hai là **ios::beg**
  - **fileObject.seekg(n)**
    - đến byte thứ n kể từ đầu file
  - **fileObject.seekg(n, ios::cur)**
    - tiến n byte
  - **fileObject.seekg(y, ios::end)**
    - lùi y byte kể từ cuối file
  - **fileObject.seekg(0, ios::cur)**
    - đến cuối file
  - **seekp** tương tự
  - **location = fileObject.tellg()**
    - lấy vị trí đọc hiện tại của **fileObject**

## 7.11 Các hàm định vị cho file tuần tự

- Ví dụ:
  - chương trình quản lý tài khoản ngân hàng - Credit manager program
  - dữ liệu: file clients.dat
  - các chức năng:
    1. in danh sách các tài khoản rỗng (account with zero balance)
    2. in danh sách các tài khoản âm (account with credit)
    3. in danh sách các tài khoản dương (account with debit)
  - hoạt động của chương trình
    1. menu cho phép người dùng chọn một chức năng hoặc chọn dừng chương trình
    2. thực hiện chức năng đã chọn và in kết quả
    3. quay lại menu

```
1 // Fig. 14.8: fig14_08.cpp
2 // Credit-inquiry program.
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::ios;
8 using std::cerr;
9 using std::endl;
10 using std::fixed;
11 using std::showpoint;
12 using std::left;
13 using std::right;
14
15 #include <fstream>
16
17 using std::ifstream;
18
19 #include <iomanip>
20
21 using std::setw;
22 using std::setprecision;
23
24 #include <cstdlib>
25
```

**fig14\_08.cpp**  
(1 of 6)



```

26 enum RequestType { ZERO_BALANCE = 1, CREDIT_BALANCE,
27 DEBIT_BALANCE, END };
28 int getRequest();
29 bool shouldDisplay(int, double);
30 void outputLine(int, const char * const, double);
31
32 int main()
33 {
34 // ifstream constructor opens the file
35 ifstream inClientFile("clients.dat", ios::in);
36
37 // exit program if ifstream could not open file
38 if (!inClientFile) {
39 cerr << "File could not be opened" << endl;
40 exit(1);
41
42 } // end if
43
44 int request;
45 int account;
46 char name[30];
47 double balance;

```

**fig14\_08.cpp**  
(2 of 6)

fig14\_08.cpp  
(3 of 6)

```
49 // get user's request (e.g., zero, credit or debit balance)
50 request = getRequest();
51
52 // process user's request
53 while (request != END) {
54
55 switch (request) {
56
57 case ZERO_BALANCE:
58 cout << "\nAccounts with zero balances:\n";
59 break;
60
61 case CREDIT_BALANCE:
62 cout << "\nAccounts with credit balances:\n";
63 break;
64
65 case DEBIT_BALANCE:
66 cout << "\nAccounts with debit balances:\n";
67 break;
68
69 } // end switch
70
```

```
71 // read account, name and balance from file
72 inClientFile >> account >> name >> balance;
73
74 // display file contents (until eof)
75 while (!inClientFile.eof()) {
76
77 // display record
78 if (shouldDisplay(request, balance))
79 outputLine(account, name, balance);
80
81 // read account, name and balance from file
82 inClientFile >> account >> name >> balance;
83
84 } // end inner while
85
86 inClientFile.clear(); // reset eof for next input
87 inClientFile.seekg(0); // move to beginning of file
88 request = getRequest(); // get additional request from user
89
90 } // end outer while
91
92 cout << "End of run." << endl;
93
94 return 0; // ifstream destructor closes the file
95
96 } // end main
```

Dùng **clear** để bỏ cờ **eof**. Dùng **seekg** để đặt con trỏ định vị file về đầu file.

fig14\_08.cpp  
(5 of 6)

```
97
98 // obtain request from user
99 int getRequest()
100 {
101 int request;
102
103 // display request options
104 cout << "\nEnter request" << endl
105 << " 1 - List accounts with zero balances" << endl
106 << " 2 - List accounts with credit balances" << endl
107 << " 3 - List accounts with debit balances" << endl
108 << " 4 - End of run" << fixed << showpoint;
109
110 // input user request
111 do {
112 cout << "\n? ";
113 cin >> request;
114
115 } while (request < ZERO_BALANCE && request > END);
116
117 return request;
118
119 } // end function getRequest
120
```

fig14\_08.cpp  
(6 of 6)

```
121 // determine whether to display given record
122 bool shouldDisplay(int type, double balance)
123 {
124 // determine whether to display credit balances
125 if (type == CREDIT_BALANCE && balance < 0)
126 return true;
127
128 // determine whether to display debit balances
129 if (type == DEBIT_BALANCE && balance > 0)
130 return true;
131
132 // determine whether to display zero balances
133 if (type == ZERO_BALANCE && balance == 0)
134 return true;
135
136 return false;
137
138 } // end function shouldDisplay
139
140 // display single record from file
141 void outputLine(int account, const char * const name,
142 double balance)
143 {
144 cout << left << setw(10) << account << setw(13) << name
145 << setw(7) << setprecision(2) << right << balance
146 << endl;
147
148 } // end function outputLine
```

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 |
|-----|-------|--------|

fig14\_08.cpp  
output (1 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

? 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.

**fig14\_08.cpp**  
**output (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

? 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.

## 7.12 Các rắc rối khi cập nhật file truy nhập tuần tự

- cập nhật các file truy nhập tuần tự
  - Rủi ro: ghi đè các dữ liệu khác
  - Ví dụ: đổi tên từ "White" thành "Worthington"

- Dữ liệu cũ

300 White 0.00 400 Jones 32.87

- Chèn dữ liệu mới

300 Worthington 0.00



300 White 0.00 400 Jones 32.87



300 Worthington 0.00ones 32.87

Dữ liệu bị ghi đè

- Định dạng bị rối loạn
- Vấn đề có thể tránh được, nhưng biện pháp không hay.

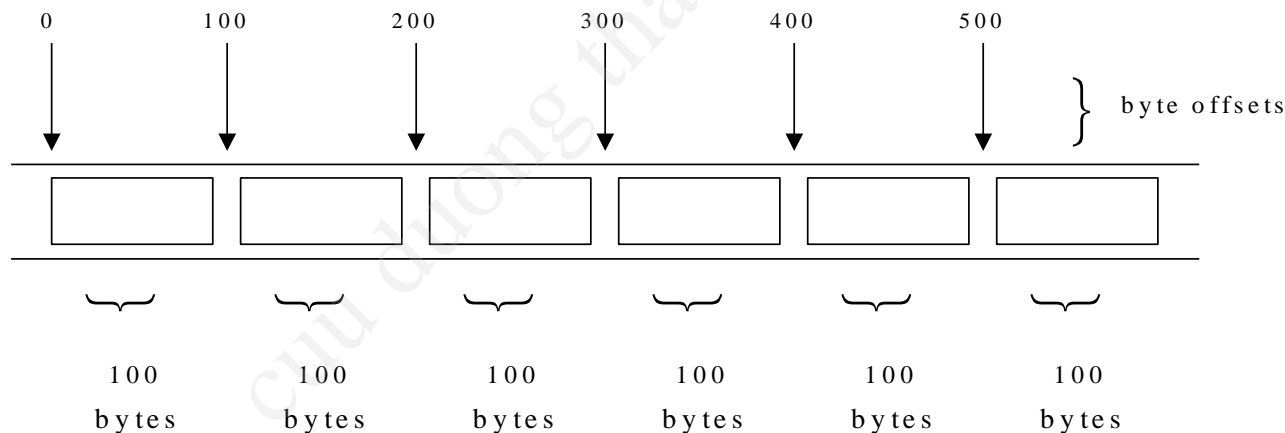


## 7.13 Random-Access Files (các file truy nhập ngẫu nhiên)

- Truy nhập tức thời - Instant access
  - muốn định vị bản ghi một cách nhanh chóng
    - các hệ thống đặt vé máy bay (airline reservations), máy rút tiền tự động (ATM)
  - các file tuần tự phải duyệt qua từng bản ghi một
- Giải pháp: các file truy nhập ngẫu nhiên
  - khả năng truy nhập tức thời
  - chen bản ghi mà không phá các dữ liệu khác
  - cập nhật/xóa một phần tử dữ liệu mà không làm thay đổi các dữ liệu khác

## 7.13 File truy nhập ngẫu nhiên (random-access file)

- C++ không quy định quy cách file
  - lập trình viên phải tự tạo quy cách cho các file truy nhập ngẫu nhiên
  - cách đơn giản nhất: các bản ghi độ dài cố định
    - tính toán được vị trí trong file từ kích thước bản ghi và khóa



## 7.13.1 Dữ liệu thô và dữ liệu định dạng

- Ví dụ: **"1234567"** (**char \***) và **1234567** (**int**)
  - định dạng: **char \*** cần 8 byte (1 byte cho mỗi ký tự + null)
  - thô: **int** lấy một số cố định byte (có thể là 4)
    - 123 có cùng kích thước theo byte với 1234567
- các phép toán **<<** và **>>** dành cho dữ liệu định dạng
  - **outFile << number**
    - ghi **number** (**int**) dưới dạng **char \***
    - số lượng byte không cố định
- hàm **write()** và **read()** dành cho dữ liệu thô
  - **outFile.write( const char \*, size );**
    - ghi ra các byte dạng thô
    - lấy tham số là con trỏ tới địa chỉ bộ nhớ, số byte cần ghi
      - sao chép dữ liệu trực tiếp từ bộ nhớ sang file
      - Không đổi thành **char \***

## 7.13.2 Ghi file truy nhập ngẫu nhiên

- Ví dụ hàm **write()**

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

- **&number** là **int \***

- đổi thành **const char \*** bằng **reinterpret\_cast**

- **sizeof(number)**

- kích thước của **number** (một số **int**) tính theo byte

- tương tự đối với hàm **read** (more later)

- Chú ý:

- chỉ dùng **write/read** giữa các máy tương thích

- mở file kiểu **ios::binary** để đọc/ghi thô

- thường dùng để ghi toàn bộ một **struct** hoặc một đối tượng ra file

## 7.13.2 Ghi file truy nhập ngẫu nhiên

- Bài toán
  - chương trình quản lý tài khoản
  - Lưu trữ tối đa 100 bản ghi kích thước cố định
  - Bản ghi
    - Mã tài khoản - Account number (khóa)
    - Họ và tên - First and last name
    - Số tiền hiện có trong tài khoản - Balance
  - Các thao tác:
    - cập nhật, tạo mới, xóa, liệt kê tất cả các tài khoản ra một file
- Tiếp theo: chương trình tạo file chứa 100 bản ghi rỗng

Class **ClientData** lưu  
thông tin về từng người.  
100 đối tượng **ClientData**  
rõng sẽ được ghi ra 1 file.

```
1 // Fig. 14.10: clientData.h
2 // Class ClientData definition used in Fig. 14.12–Fig. 14.15.
3 #ifndef CLIENTDATA_H
4 #define CLIENTDATA_H
5
6 #include <iostream>
7
8 using std::string;
9
10 class ClientData {
11
12 public:
13
14 // default ClientData constructor
15 ClientData(int = 0, string = "", string = "", double = 0.0);
16
17 // accessor functions for accountNumber
18 void setAccountNumber(int);
19 int getAccountNumber() const;
20
21 // accessor functions for lastName
22 void setLastName(string);
23 string getLastName() const;
24 }
```

**clientData.h**  
(2 of 2)

```
25 // accessor functions for firstName
26 void setFirstName(string);
27 string getFirstName() const;
28
29 // accessor functions for balance
30 void setBalance(double);
31 double getBalance() const;
32
33 private:
34 int accountNumber;
35 char lastName[15];
36 char firstName[10];
37 double balance;
38
39 }; // end class ClientData
40
41 #endif
```

Đặt giới hạn kích thước tên và họ.  
**accountNumber** (một số **int**)  
và **balance** (**double**) đã có kích  
thước cố định.

## ClientData.cpp (1 of 4)

```
1 // Fig. 14.11: ClientData.cpp
2 // Class ClientData stores customer's credit information.
3 #include <iostream>
4
5 using std::string;
6
7 #include <cstring>
8 #include "clientData.h"
9
10 // default ClientData constructor
11 ClientData::ClientData(int accountNumberValue,
12 string lastNameValue, string firstNameValue,
13 double balanceValue)
14 {
15 setAccountNumber(accountNumberValue);
16 setLastName(lastNameValue);
17 setFirstName(firstNameValue);
18 setBalance(balanceValue);
19
20 } // end ClientData constructor
21
22 // get account-number value
23 int ClientData::getAccountNumber() const
24 {
25 return accountNumber;
26
27 } // end function getAccountNumber
```



## ClientData.cpp (2 of 4)

```
28
29 // set account-number value
30 void ClientData::setAccountNumber(int accountNumberValue)
31 {
32 accountNumber = accountNumberValue;
33
34 } // end function setAccountNumber
35
36 // get last-name value
37 string ClientData::getLastName() const
38 {
39 return lastName;
40
41 } // end function getLastName
42
43 // set last-name value
44 void ClientData::setLastName(string lastNameString)
45 {
46 // copy at most 15 characters from string to lastName
47 const char *lastNameValue = lastNameString.data();
48 int length = strlen(lastNameValue);
49 length = (length < 15 ? length : 14);
50 strncpy(lastName, lastNameValue, length);
51
52 // append null character to lastName
53 lastName[length] = '\\0';
```

**ClientData.cpp**  
(3 of 4)

```
54
55 } // end function setLastName
56
57 // get first-name value
58 string ClientData::getFirstName() const
59 {
60 return firstName;
61
62 } // end function getFirstName
63
64 // set first-name value
65 void ClientData::setFirstName(string firstNameString)
66 {
67 // copy at most 10 characters from string to firstName
68 const char *firstNameValue = firstNameString.data();
69 int length = strlen(firstNameValue);
70 length = (length < 10 ? length : 9);
71 strncpy(firstName, firstNameValue, length);
72
73 // append new-line character to firstName
74 firstName[length] = '\\0';
75
76 } // end function setFirstName
77
```

```
78 // get balance value
79 double ClientData::getBalance() const
80 {
81 return balance;
82
83 } // end function getBalance
84
85 // set balance value
86 void ClientData::setBalance(double balanceValue)
87 {
88 balance = balanceValue;
89
90 } // end function setBalance
```

**ClientData.cpp**  
**(4 of 4)**

fig14\_12.cpp  
(1 of 2)

```
1 // Fig. 14.12: fig14_12.cpp
2 // Creating a randomly accessed file.
3 #include <iostream>
4
5 using std::cerr;
6 using std::endl;
7 using std::ios;
8
9 #include <fstream>
10
11 using std::ofstream;
12
13 #include <cstdlib>
14 #include "clientData.h" // ClientData class definition
15
16 int main()
17 {
18 ofstream outCredit("credit.dat", ios::binary);
19
20 // exit program if ofstream could not open file
21 if (!outCredit) {
22 cerr << "File could not be opened." << endl;
23 exit(1);
24 }
25 }
```

Mở 1 file để ghi thô,  
sử dụng một đối tượng **ofstream**  
và **ios::binary**.

```
26
27 // create ClientData with no information
28 ClientData blankClient;
29
30 // output 100 blank records to file
31 for (int i = 0; i < 100; i++)
32 outCredit.write(
33 reinterpret_cast< const char * >(&blankClient),
34 sizeof(ClientData));
35
36 return 0;
37
38 } // end main
```

Tạo một đối tượng rỗng.  
Dùng **write** để ghi dữ liệu  
thô ra 1 file (truyền tham số là  
địa chỉ đối tượng và kích  
thước đối tượng).

## 7.13.3 Ghi dữ liệu vào vị trí tùy ý trong file truy nhập ngẫu nhiên

- Dùng **seekp** để ghi vào vị trí chính xác trong file
  - Bản ghi đầu tiên bắt đầu từ đâu?
    - Byte 0
  - Bản ghi thứ hai?
    - Byte 0 + sizeof(object)
  - Bản ghi bất kỳ?
    - (Recordnum - 1) \* sizeof(object)

**fig14\_13.cpp**  
(1 of 3)

```
1 // Fig. 14.13: fig14_13.cpp
2 // Writing to a random access file.
3 #include <iostream>
.....
19 #include <cstdlib>
20 #include "clientData.h" // ClientData class definition
21
22 int main()
23 {
24 int accountNumber;
25 char lastName[15];
26 char firstName[10];
27 double balance;
28
29 ofstream outCredit("credit.dat", ios::binary);
30
31 // exit program if ofstream cannot open file
32 if (!outCredit) {
33 cerr << "File could not be opened." << endl;
34 exit(1);
35
36 } // end if
```

Mở file để ghi thô (binary writing).

```

38 cout << "Enter account number "
39 << "(1 to 100, 0 to end input)\n? ";
40
41 // require user to specify account number
42 ClientData client;
43 cin >> accountNumber;
44 client.setAccountNumber(accountNumber);
45
46 // user enters information, which is copied into file
47 while (client.getAccountNumber() > 0 &&
48 client.getAccountNumber() <= 100) {
49
50 // user enters last name, first name and balance
51 cout << "Enter lastname, firstname, balance\n? ";
52 cin >> setw(15) >> lastName;
53 cin >> setw(10) >> firstName;
54 cin >> balance;
55
56 // set record lastName, firstName and balance values
57 client.setLastName(lastName);
58 client.setFirstName(firstName);
59 client.setBalance(balance);

```

Nhập account number, ghi vào đối tượng. Nó chưa được viết ra file.

fig14\_13.cpp  
(2 of 3)



Đặt **outCredit** vào vị trí thích hợp trong file (dựa vào account number).

fig14\_13.cpp  
(3 of 3)

```
60
61 // seek position in file of user-specified record
62 outCredit.seekp((client.getAccountNumber() - 1) *
63 sizeof(ClientData));
```

Ghi đối tượng **ClientData** vào file tại vị trí đó.

```
64
65 // write user-specified information in file
66 outCredit.write(
67 reinterpret_cast< const char * >(&client),
68 sizeof(ClientData));
69
70 // enable user to specify another account number
71 cout << "Enter account number\n? ";
72 cin >> accountNumber;
73 client.setAccountNumber(accountNumber);
```

```
74
75 } // end while
```

```
76
77 return 0;
```

```
78
79 } // end main
```

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

Lưu ý các account có thể  
được tạo theo thứ tự tùy ý.

**fig14\_13.cpp**  
**output (1 of 1)**

## 7.13.4 Đọc tuần tự dữ liệu từ file truy nhập ngẫu nhiên

- **read** - tương tự **write**

- Đọc các byte thô từ file vào bộ nhớ
- `inFile.read( reinterpret_cast<char *>( &number ), sizeof( int ) );`
  - **&number**: địa chỉ để lưu dữ liệu
  - **sizeof(int)**: số byte cần đọc
- Không dùng **inFile >> number** cho dữ liệu thô - nhị phân
  - **>> nhận char \***

- Chương trình tiếp theo

- lấy dữ liệu từ một file random-access
- duyệt tuần tự qua từng bản ghi
  - If no data (accountNumber == 0) then skip

fig14\_14.cpp  
(1 of 2)

```
1 // Fig. 14.14: fig14_14.cpp
2 // Reading a random access file.
...
25 #include "clientData.h" // ClientData class definition
26
27 void outputLine(ostream&, const ClientData &);
28
29 int main()
30 {
31 ifstream inCredit("credit.dat", ios::in);
32
33 // exit program if ifstream cannot open file
34 if (!inCredit) {
35 cerr << "File could not be opened." << endl;
36 exit(1);
37 } // end if
38
39
40 cout << left << setw(10) << "Account" << setw(16)
41 << "Last Name" << setw(11) << "First Name" << left
42 << setw(10) << right << "Balance" << endl;
43
44 ClientData client; // create record
45
46 // read first record from file
47 inCredit.read(reinterpret_cast< char * >(&client),
48 sizeof(ClientData));
```

Đọc sizeof(ClientData) byte và ghi vào đối tượng client. Đây có thể là một bản ghi rỗng.

```

50 // read all records from file
51 while (inCredit && !inCredit.eof()) {
52
53 // display record
54 if (client.getAccountNumber() != 0)
55 outputLine(cout, client);
56
57 // read next from file
58 inCredit.read(reinterpret_cast< char * >(&client),
59 sizeof(ClientData));
60
61 } // end while
62
63 return 0;
64
65 } // end main
66
67 // display single record
68 void outputLine(ostream &output, const ClientData &record)
69 {
70 output << left << setw(10) << record.getAccountNumber()
71 << setw(16) << record.getLastName().data()
72 << setw(11) << record.getFirstName().data()
73 << setw(10) << setprecision(2) << right << fixed
74 << showpoint << record.getBalance() << endl;
75
76 } // end outputLine

```

Vòng lặp dừng khi có lỗi đọc  
(**inCredit == 0**) hoặc gặp EOF  
(**inCredit.eof() == 1**)

Output non-empty accounts.  
Lưu ý **outputLine** lấy 1  
tham số kiểu **ostream**. Ta  
có thể dễ dàng output ra một  
file khác (mở bằng một  
**ofstream** object, là dẫn  
xuất của **ostream**).

| 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   |

**fig14\_14.cpp**  
**output (1 of 1)**

## 7.14 Ví dụ: Chương trình xử lý giao dịch

- Bài toán:
  - chương trình quản lý các tài khoản ngân hàng, cho phép truy nhập trực tiếp từng tài khoản
  - dữ liệu: file truy nhập ngẫu nhiên **credit.dat**
- Các chức năng cho người dùng (các lựa chọn cho menu)
  - Lựa chọn 1: ghi các account ra file **print.txt**

| 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   |

- Lựa chọn 2: cập nhật bản ghi

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

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

## 7.14 Ví dụ: Chương trình xử lý giao dịch

- Các chức năng (tiếp)
  - Lựa chọn 3: thêm bản ghi

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

- Lựa chọn 4: xóa bản ghi

```
Enter account to delete (1 - 100): 29
Account #29 deleted.
```

- Mở file vừa đọc vừa ghi
  - Dùng **fstream** object
  - nhiều file-open mode cùng lúc

```
fstream inOutCredit("credit.dat", ios::in | ios::out);
```



```
1 // Fig. 14.15: fig14_15.cpp
2 // This program reads a random access file sequentially, updates
3 // data previously written to the file, creates data to be placed
4 // in the file, and deletes data previously in the file.
5 #include <iostream>
6
7 using std::cout;
8 ...
15 using std::showpoint;
16
17 #include <fstream>
18
19 using std::ofstream;
20 using std::ostream;
21 using std::fstream;
22
23 #include <iomanip>
24
25 using std::setw;
26 using std::setprecision;
27
28 #include <cstdlib> // exit prototype
29 #include "clientData.h" // ClientData class definition
```

fig14\_15.cpp  
(2 of 14)

```
30
31 int enterChoice();
32 void printRecord(fstream&);
33 void updateRecord(fstream&);
34 void newRecord(fstream&);
35 void deleteRecord(fstream&);
36 void outputLine(ostream&, const ClientData &);
37 int getAccount(const char * const);
38
39 enum Choices { PRINT = 1, UPDATE, NEW, DELETE, END };
40
41 int main()
42 {
43 // open file for reading and writing
44 fstream inOutCredit("credit.dat", ios::in | ios::out);
45
46 // exit program if fstream cannot open file
47 if (!inOutCredit) {
48 cerr << "File could not be opened." << endl;
49 exit (1);
50
51 } // end if
52
```

Mở file để đọc và ghi (cần  
fstream object).

Hiện menu và trả về lựa chọn người dùng.

```
53 int choice;
54
55 // enable user to specify action
56 while ((choice = enterChoice()) != END) {
57
58 switch (choice) {
59
60 // create text file from record file
61 case PRINT:
62 printRecord(inOutCredit);
63 break;
64
65 // update record
66 case UPDATE:
67 updateRecord(inOutCredit);
68 break;
69
70 // create record
71 case NEW:
72 newRecord(inOutCredit);
73 break;
74
75 // delete existing record
76 case DELETE:
77 deleteRecord(inOutCredit);
78 break;
79
```

```

80 // display error if user does not select valid choice
81 default:
82 cerr << "Incorrect choice" << endl;
83 break;
84
85 } // end switch
86
87 inOutCredit.clear(); // reset end-of-file indicator
88
89 } // end while
90
91 return 0;
92
93 } // end main
94
95 // enable user to input menu choice
96 int enterChoice()
97 {
98 // display available options
99 cout << "\nEnter your choice" << endl
100 << "1 - store a formatted text file of accounts" << endl
101 << " called \"print.txt\" for printing" << endl
102 << "2 - update an account" << endl
103 << "3 - add a new account" << endl
104 << "4 - delete an account" << endl
105 << "5 - end program\n? ";

```

fig14\_15.cpp  
(5 of 14)

fig14\_15.cpp  
(6 of 14)

```
106
107 int menuChoice;
108 cin >> menuChoice; // receive choice from user
109
110 return menuChoice;
111
112 } // end function enterChoice
113
114 // create formatted text file for printing
115 void printRecord(fstream &readFromFile)
116 {
117 // create text file
118 ofstream outPrintFile("print.txt", ios::out);
119
120 // exit program if ofstream cannot create file
121 if (!outPrintFile) {
122 cerr << "File could not be created." << endl;
123 exit(1);
124 } // end if
125
126
127 outPrintFile << left << setw(10) << "Account" << setw(16)
128 << "Last Name" << setw(11) << "First Name" << right
129 << setw(10) << "Balance" << endl;
130
```

In ra **print.txt**. Trước tiên, in header của bảng.

fig14\_15.cpp

Đến đầu file, đọc dữ liệu về tài khoản, và in bản ghi nếu nó không rỗng.

Lưu ý **outputLine** lấy đối số là đối tượng **ostream** object (lớp cơ sở của **ofstream**). Nó có thể ghi ra file (như trong trường hợp này) hoặc **cout**.

```
131 // set file-position pointer to beginning of record file
132 readFromFile.seekg(0);
133
134 // read first record from record file
135 ClientData client;
136 readFromFile.read(reinterpret_cast< char * >(&client),
137 sizeof(ClientData));
138
139 // copy all records from record file into text file
140 while (!readFromFile.eof()) {
141
142 // write single record to text file
143 if (client.getAccountNumber() != 0)
144 outputLine(outPrintFile, client);
145
146 // read next record from record file
147 readFromFile.read(reinterpret_cast< char * >(&client),
148 sizeof(ClientData));
149
150 } // end while
151
152 } // end function printRecord
153
```

```
154 // update balance in record
155 void updateRecord(fstream &updateFile)
156 {
157 // obtain number of account to update
158 int accountNumber = getAccount("Enter account to update");
159
160 // move file-position pointer to correct record in file
161 updateFile.seekg(
162 (accountNumber - 1) * sizeof(ClientData));
163
164 // read first record from file
165 ClientData client;
166 updateFile.read(reinterpret_cast< char * >(&client),
167 sizeof(ClientData));
168
169 // update record
170 if (client.getAccountNumber() != 0) {
171 outputLine(cout, client);
172
173 // request user to specify transaction
174 cout << "\nEnter charge (+) or payment (-): ";
175 double transaction; // charge or payment
176 cin >> transaction;
```

Đây là **fstream** (I/O) vì ta phải đọc balance cũ, cập nhật nó, và ghi balance mới.

fig14\_15.cpp  
(9 of 14)

```
177
178 // update record balance
179 double oldBalance = client.getBalance();
180 client.setBalance(oldBalance + transaction);
181 outputLine(cout, client);
182
183 // move file-position pointer to correct record in file
184 updateFile.seekp(
185 (accountNumber - 1) * sizeof(ClientData));
186
187 // write updated record over old record in file
188 updateFile.write(
189 reinterpret_cast< const char * >(&client),
190 sizeof(ClientData));
191
192 } // end if
193
194 // display error if account does not exist
195 else
196 cerr << "Account #" << accountNumber
197 << " has no information." << endl;
198
199 } // end function updateRecord
200
```



```
201 // create and insert record
202 void newRecord(fstream &insertInFile)
203 {
204 // obtain number of account to create
205 int accountNumber = getAccount("Enter new account number");
206
207 // move file-position pointer to correct record in file
208 insertInFile.seekg(
209 (accountNumber - 1) * sizeof(ClientData));
210
211 // read record from file
212 ClientData client;
213 insertInFile.read(reinterpret_cast< char * >(&client),
214 sizeof(ClientData));
215
216 // create record, if record does not previously exist
217 if (client.getAccountNumber() == 0) {
218
219 char lastName[15];
220 char firstName[10];
221 double balance;
```

Đây là **fstream** vì ta đọc thử để xem đã có sẵn một bản ghi rồi hay chưa, nếu chưa, ta ghi một bản ghi mới.

```
222
223 // user enters last name, first name and balance
224 cout << "Enter lastname, firstname, balance\n? ";
225 cin >> setw(15) >> lastName;
226 cin >> setw(10) >> firstName;
227 cin >> balance;
228
229 // use values to populate account values
230 client.setLastName(lastName);
231 client.setFirstName(firstName);
232 client.setBalance(balance);
233 client.setAccountNumber(accountNumber);
234
235 // move file-position pointer to correct record in file
236 insertInFile.seekp((accountNumber - 1) *
237 sizeof(ClientData));
238
239 // insert record in file
240 insertInFile.write(
241 reinterpret_cast< const char * >(&client),
242 sizeof(ClientData));
243
244 } // end if
245
```

fig14\_15.cpp  
(12 of 14)

```
246 // display error if account previously exists
247 else
248 cerr << "Account #" << accountNumber
249 << " already contains information." << endl;
250
251 } // end function newRecord
252
253 // delete an existing record
254 void deleteRecord(fstream &deleteFromFile)
255 {
256 // obtain number of account to delete
257 int accountNumber = getAccount("Enter account to delete");
258
259 // move file-position pointer to correct record in file
260 deleteFromFile.seekg(
261 (accountNumber - 1) * sizeof(ClientData));
262
263 // read record from file
264 ClientData client;
265 deleteFromFile.read(reinterpret_cast< char * >(&client),
266 sizeof(ClientData));
267
```

là **fstream** vì ta đọc để kiểm tra xem account có tồn tại không. Nếu có, ta ghi dữ liệu rỗng (xóa nó). Nếu không, không cần xóa.

fig14\_15.cpp  
(13 of 14)

```
268 // delete record, if record exists in file
269 if (client.getAccountNumber() != 0) {
270 ClientData blankClient;
271
272 // move file-position pointer to correct record in file
273 deleteFromFile.seekp((accountNumber - 1) *
274 sizeof(ClientData));
275
276 // replace existing record with blank record
277 deleteFromFile.write(
278 reinterpret_cast< const char * >(&blankClient),
279 sizeof(ClientData));
280
281 cout << "Account #" << accountNumber << " deleted.\n";
282
283 } // end if
284
285 // display error if record does not exist
286 else
287 cerr << "Account #" << accountNumber << " is empty.\n";
288
289 } // end deleteRecord
290
```

```
291 // display single record
292 void outputLine(ostream &output, const ClientData &record)
293 {
294 output << left << setw(10) << record.getAccountNumber()
295 << setw(16) << record.getLastName().data()
296 << setw(11) << record.getFirstName().data()
297 << setw(10) << setprecision(2) << right << fixed
298 << showpoint << record.getBalance() << endl;
299
300 } // end function outputLine
301
302 // obtain account-number value from user
303 int getAccount(const char * const prompt)
304 {
305 int accountNumber;
306
307 // obtain account-number value
308 do {
309 cout << prompt << " (1 - 100): ";
310 cin >> accountNumber;
311
312 } while (accountNumber < 1 || accountNumber > 100);
313
314 return accountNumber;
315
316 } // end function getAccount
```

**outputLine** rất mềm dẻo, và có thể ghi ra **ostream** object bất kỳ (chẳng hạn 1 file hoặc **cout**).