



Bài 7: Làm việc với File



Nội dung

1. File và luồng
2. File truy nhập tuần tự
3. File truy nhập ngẫu nhiên
4. Review
5. Bài tập



Ôn lại

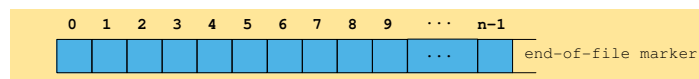
1. Khái niệm luồng (streams). Các lớp luồng.
2. Luồng xuất và các phương thức trên luồng xuất.
3. Luồng nhập và các phương thức trên luồng nhập.
4. Thư viện iomanip.

3



File và luồng(Streams)

- 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
- 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 luồng dữ liệu
 - tạo "đường liên lạc" từ đối tượng tới file

4



Các loại tệp tin (file)

- Tệp tin (file)
 - là một dãy các byte có giá trị từ 0 đến 255 ghi trên đĩa. Số byte của dãy chính là độ dài của một tệp
- Tệp tin văn bản (text file)
 - là tệp mà chứa các phần tử là các kí tự , không kể các kí tự điều khiển.
 - Số lượng tối đa: 255 ký tự / 1 dòng
 - Dấu kết thúc dòng là CR (mã 13) và LF (mã 10).
- Tệp tin nhị phân
 - là tệp mà các phần tử là các số nhị phân biểu diễn thông tin, chứa nhiều dữ liệu có mã điều khiển(các kí tự có mã ASCII từ 0 đến 31).
 - Khác với kiểu tệp tin văn bản, tệp tin nhị phân có kí hiệu kết thúc dòng là LF không có kí tự CR.


5



Truy cập file

- Đọc một tệp tin:
 - Mở tệp (file) với tên đã biết (tên = xâu kí tự)
 - Đọc nội dung của tệp
 - Đóng tệp lại khi đọc xong
- Ghi vào một tệp tin:
 - Tạo một tệp tin để ghi (gắn cho một cái tên)
 - Ghi nội dung lên tệp
 - Đóng tệp lại khi ghi xong

6




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



File truy nhập tuần tự (sequential-access file)

- 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 để ghi và di chuyển đến cuối file (thường dùng để nối dữ liệu vào file). Dữ liệu có thể được viết vào vị trí tùy ý trong file.
<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 đối với <code>ios::out</code>)
<code>ios::binary</code>	Mở file nhị phân (i.e., không phải file text) để đọc hoặc ghi.

- Theo mặc định, `ofstream` mở để ghi
- Ví dụ
 - `ofstream outClientFile("vidu.dat", ios::out);`
 - `ofstream outClientFile("vidu.dat");`

8




Ví dụ 1

- Ví dụ: viết chương trình yêu cầu người dùng nhập vào 5 số nguyên từ bàn phím. Sau đó, in tổng của chúng ra màn hình.

```
int tong = 0;
for (int i = 0; i < 5; i++)
{
    int x;
    cout<<"Nhap so thu : "<<i+1<<": ";    cin>>x;
    tong += x;
}
cout<<"Tong cua 5 so = "<<tong;
```

9



Ví dụ 1 - ghi file

- Lưu trữ thông tin nhập trên vào 1 file (ketqua.ktq) với mô tả như sau:
 - File đó có nhiều dòng
 - Mỗi dòng sẽ có 6 số, 2 số phân biệt nhau 1 dấu cách
 - 5 số đầu để ghi lại những số đã nhập tại lần đó
 - Số cuối cùng để ghi tổng của 5 số trên
- Biểu diễn sau cho ta biết cách tổ chức của một file (ketqua.ktq) như mô tả ở trên:

```
1 3 5 2 1 12
1 3 6 9 12 31
```

10

Ví dụ 1 - ghi file

```
#include <iostream.h>
#include <fstream.h>
int main(){
    char *filename = "ketqua.ktq";
    ofstream fout(filename);
    const int solan = 2;
    for (int i = 0; i < solan; i++)
    {
        int tong = 0;
        for (int j=0; j < 5; j++)
        {
            int x;
            cout<<"Nhap vao so thu "<< j+1 <<" : "; cin>>x;
            fout<< x <<" ";
            tong += x;
        }
        fout<<tong<<"\n";
    }
    fout.close();
    return 0;}
```

Tên file muốn ghi ra

Mở tệp tin để ghi

Nhập 2 lần 5 số nguyên

Ghi từng số vào file, phân biệt nhau dấu cách

Ghi tổng vào file rồi xuống dòng

Đóng file

11

Ví dụ 1 - Đọc file

- Đọc file ketqua.ktq trên và in ra màn hình dạng như sau:
 $1 + 3 + 5 + 2 + 1 = 12$
 $1 + 3 + 6 + 9 + 12 = 31$
- Đoạn chương trình sau sẽ minh họa điều này

Ví dụ 1 - Đọc file

```
#include <iostream.h>
#include <fstream.h>
int main(){
    char *filename = "ketqua.ktq";    ifstream fin(filename);
    int x, i = 0;
    while (fin>>x) ← "Mở file để đọc"
    {
        i++;    cout<<x; ← "Đọc cho đến khi hết file"
        if ((i % 5) == 0) ← "In giá trị vừa đọc ra màn hình"
            cout<<" "; ← "Đọc hết 5 số trên 1 dòng"
        else ← "Đọc đến tổng"
            if ((i % 6) == 0) {cout<<"\n";    i = 0;}
            else ← "Các số hạng"
                cout<<" + ";
    }
    fin.close();
    return 0;
}
```

13

Ví dụ 2

- Ví dụ: Viết chương trình quản lý tài khoản ngân hàng - Credit manager program
 - Nhập các thông tin về Số Tài khoản (account), Tên TK (name), Số dư tiền (balance). Ghi các dữ liệu trên vào file clients.dat

```
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
```

14

Ví dụ 2

```
#include <iostream>
#include <fstream>
using namespace std;
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? ";
}
```

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

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

15

Ví dụ 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
```

Đọc cho tới khi gặp EOF, nó trả về 0 và vòng lặp dừng.

Ghi dữ liệu ra file như ghi ra một luồng chuẩn

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

16

Ví dụ 3

■ Ví dụ 3

- Có file dữ liệu clients.dat (được tạo từ ví dụ 2)
- Đọc các thông tin từ file clients.dat. Hiển thị trên màn hình theo khuôn dạng

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

17

Ví dụ 3

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
void outputLine( int, const char * const, double );
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
```

18



Ví dụ 3

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

19



File truy nhập tuần tự (sequential-access file)

- 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**

20

Các hàm đặt lại vị trí của con trỏ

■ 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`

21

File truy nhập tuần tự (sequential-access file)

■ Các rắc rối khi cập nhật 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.

22

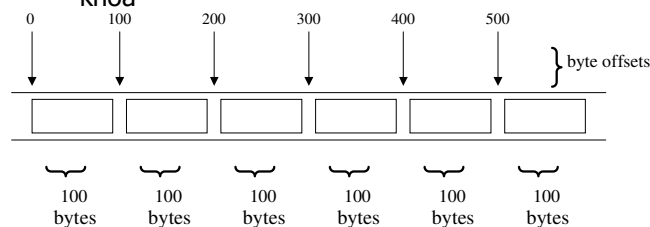
Random-Access Files (các file truy nhập ngẫu nhiên)

- 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
 - chèn 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

23

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



24



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 ***

25




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

26



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
 - `>>` nhận `char *`

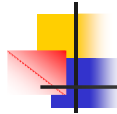
27



Review

1. File và luồng
2. File truy nhập tuần tự
3. File truy nhập ngẫu nhiên
4. Review
5. Bài tập

28



Bài tập

1. Thực hành các bài tập trong bài trên máy tính.
2. Viết 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