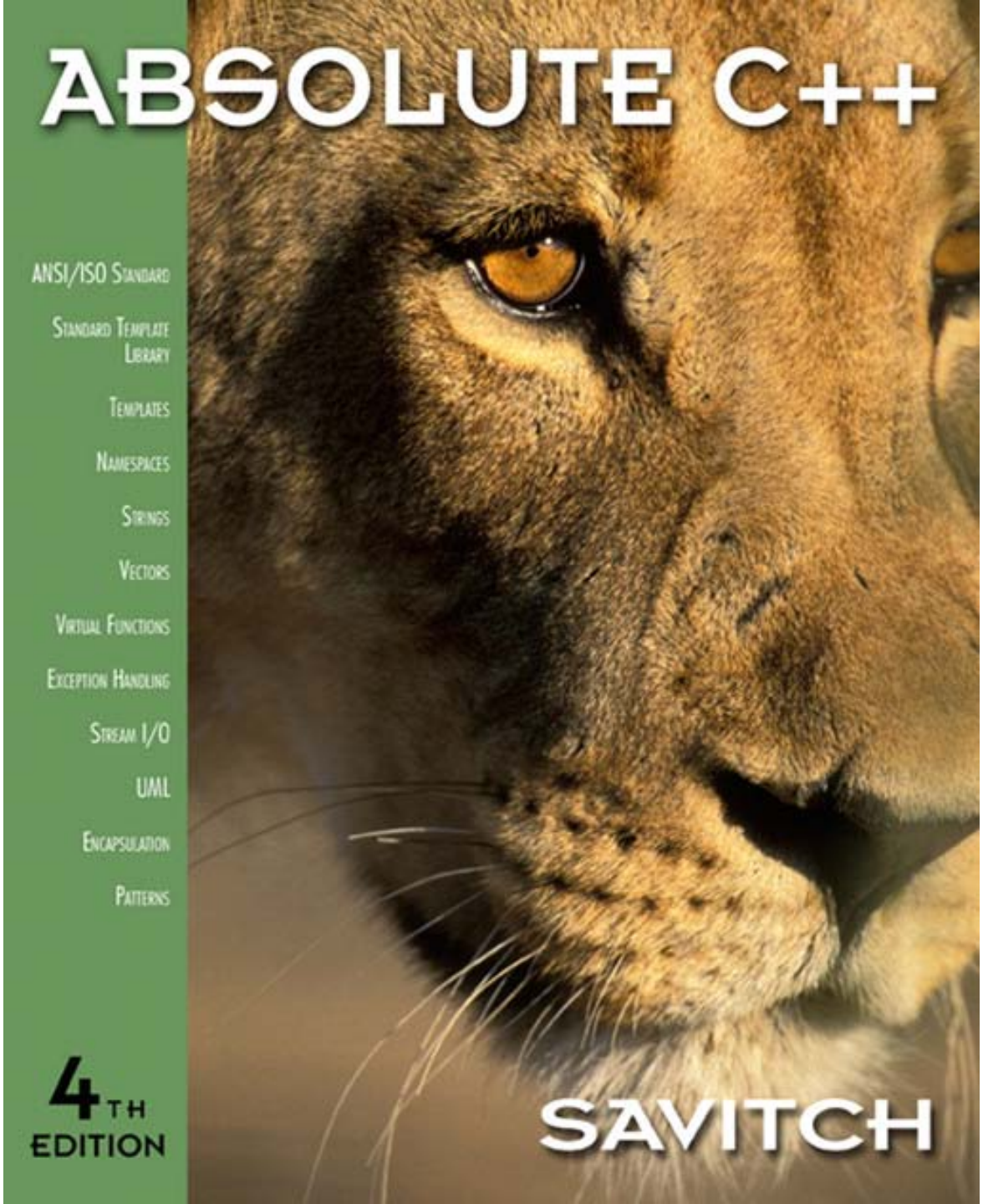
Bài 12: Đọc/ghi trên luồng và tệp

Giảng viên: Hoàng Thị Điệp
Khoa Công nghệ Thông tin – ĐH Công Nghệ

Thuật ngữ

- stream: dòng tin / dòng / luồng
- input stream: luồng nhập
- output stream: luồng xuất
- standard input stream: luồng nhập chuẩn == bàn phím
- standard output stream: luồng xuất chuẩn == màn hình
- file: tệp
- text file: tệp văn bản
- binary file: tệp nhị phân



Chapter 12

Streams and File I/O

Copyright © 2010 Pearson Addison-Wesley.
All rights reserved

PEARSON Addison Wesley

Mục tiêu bài học

- Đọc/ghi trên luồng
  - Đọc/ghi trên tệp
  - Đọc/ghi kí tự
- Công cụ đọc/ghi trên luồng
  - Tên tệp là input
  - Định dạng kết quả xuất, thiết đặt cờ
- Phân cấp luồng
  - Sơ lược về khái niệm thừa kế
- Phương thức truy cập ngẫu nhiên trên tệp

Giới thiệu

- Luồng
  - Các đối tượng đặc biệt
  - Chuyển phát input và output của chương trình
- Đọc/ghi trên tệp
  - Sử dụng khái niệm thừa kế
    - Giới thiệu ở chương 14 giáo trình
  - Đọc/ghi trên tệp rất hữu ích
    - Được giới thiệu ở bài này

Luồng

- Một dòng chảy kí tự
- Luồng nhập
  - Các kí tự chảy vào chương trình
    - Có thể xuất phát từ bàn phím
    - Có thể xuất phát từ tệp
- Luồng xuất
  - Các kí tự chảy từ chương trình ra
    - Có thể hướng tới màn hình
    - Có thể hướng tới tệp

Sử dụng luồng

- Ta đã sử dụng luồng ở các bài trước
  - cin
    - Đối tượng luồng nhập kết nối với bàn phím
  - cout
    - Đối tượng luồng xuất kết nối với màn hình
- Có thể định nghĩa các luồng khác
  - Hướng tới hoặc xuất phát từ tệp
  - Dùng tương tự như cin, cout

Sử dụng luồng giống cách dùng cin, cout

- Xét ví dụ:
  - Một chương trình định nghĩa đối tượng inStream xuất phát từ tệp nào đó:
    int theNumber;
    inStream >> theNumber;
    - Đọc giá trị từ luồng, gán cho biến theNumber
  - Chương trình này cũng định nghĩa đối tượng outStream hướng tới tệp nào đó
    outStream << "theNumber is " << theNumber;
    - Ghi giá trị vào luồng, từ đó sẽ đi vào tệp

Tệp

- Ta sẽ bàn về các thao tác trên tệp văn bản
- Đọc từ tệp
  - Khi chương trình lấy input
- Viết vào tệp
  - Khi chương trình truyền output ra
- Bắt đầu từ đầu tệp tới cuối tệp
  - C++ có những phương thức đọc/ghi khác
  - Nhưng ở đây ta chỉ bàn về những phương thức đơn giản trên tệp văn bản

Kết nối với tệp

- Trước tiên phải kết nối tệp và đối tượng luồng
- Để đọc:
  - Tệp → đối tượng ifstream
- Để ghi:
  - Tệp → đối tượng ofstream
- Lớp ifstream và lớp ofstream
  - Định nghĩa trong thư viện <fstream>
  - Đặt tên trong không gian tên std

Thư viện đọc/ghi tệp

- Để cho phép cả đọc tệp và ghi tệp trong chương trình:

```
#include <fstream>
using namespace std;
```

hoặc

```
#include <fstream>
using std::ifstream;
using std::ofstream;
```

Khai báo luồng

- Phải khai báo luồng như ta làm với tất cả các biến class khác:

```
ifstream inStream;
ofstream outStream;
```

- Sau đó phải kết nối nó với tệp:

```
inStream.open("infile.txt");
```

  - Gọi là mở tệp
  - Dùng hàm thành viên open
  - Có thể dùng đường dẫn đầy đủ

Sử dụng luồng

- Sau khi khai báo ta có thể sử dụng nó

```
int oneNumber, anotherNumber;
inStream >> oneNumber >> anotherNumber;
```

- Tương tự với luồng xuất:

```
ofstream outStream;
outStream.open("outfile.txt");
outStream       << "oneNumber = " << oneNumber
                << " anotherNumber = "
                << anotherNumber;
```

  - Truyền các mẩu dữ liệu tới tệp output

Tên tệp

- Chương trình và tệp
- Tệp có 2 tên trong chương trình của ta
  - Tên tệp ngoài
    - Còn gọi là tên tệp vật lý
    - Ví dụ "infile.txt"
    - Đôi khi được gọi là tên tệp thực sự
    - Chỉ dùng 1 lần duy nhất trong chương trình (để mở tệp)
  - Tên luồng
    - Còn gọi là tên tệp logic
    - Chương trình dùng tên này cho tất cả các hoạt động trên tệp

Đóng tệp

- Nên đóng tệp
  - khi chương trình hoàn thành đọc dữ liệu từ tệp hoặc ghi dữ liệu ra tệp
  - Lệnh đóng tệp sẽ ngắt kết nối giữa luồng và tệp
  - Đóng tệp cho ví dụ trước:

    ```
    inStream.close();
    outStream.close();
    ```

    - Không đối số
- Tệp tự động đóng khi chương trình kết thúc

flush cho tệp

- Dữ liệu xuất thường được "buffered"
  - Lưu lại tạm thời trước khi ghi vào tệp
  - Ghi theo nhóm
- Đôi khi cần ép ghi:
  outStream.flush();
  - Hàm thành viên flush có thể áp dụng cho tất cả các luồng xuất
  - Dữ liệu xuất bị buffered sẽ được ghi thực sự
- Lệnh đóng tệp sẽ tự động gọi tới flush()

Ví dụ tệp:
Display 12.1 Đọc/ghi đơn giản trên tệp (1/2)

Display 12.1 Simple File Input/Output

```
1  //Reads three numbers from the file infile.txt, sums the numbers,
2  //and writes the sum to the file outfile.txt.
3  #include <fstream>
4  using std::ifstream;
5  using std::ofstream;
6  using std::endl;

7  int main( )
8  {
9      ifstream inStream;
10     ofstream outStream;

11     inStream.open("infile.txt");
12     outStream.open("outfile.txt");

13     int first, second, third;
14     inStream >> first >> second >> third;
15     outStream << "The sum of the first 3\n"
16               << "numbers in infile.txt\n"
17               << "is " << (first + second + third)
18               << endl;
```

A better version of this program is given in Display 12.3.

Ví dụ tệp:
Display 12.1 Đọc/ghi đơn giản trên tệp (2/2)

```
19        inStream.close( );
20        outStream.close( );

21        return 0;
22    }
```

SAMPLE DIALOGUE

There is no output to the screen
and no input from the keyboard.

infile.txt
(Not changed by program)

```
1
2
3
4
```

outfile.txt
(After program is run)

```
The sum of the first 3
numbers in infile.txt
is 6
```

Nối vào một tệp

- Thao tác mở tệp chuẩn bắt đầu với tệp rỗng
  - Nếu tệp có dữ liệu trước khi mở thì toàn bộ dữ liệu sẽ bị xóa
- Mở để ghi nối:

```
ofstream outStream;
outStream.open("important.txt", ios::app);
```

  - Nếu tệp không tồn tại → tạo tệp
  - Nếu tệp tồn tại → ghi nối vào cuối tệp
  - Đối số thứ 2 là hằng định nghĩa sẵn cho lớp ios
    - Trong thư viện <iostream>, không gian tên std

Cú pháp khác để mở tệp

- Có thể chỉ định tên tệp khi khai báo
  - Truyền đối số (là tên tệp) cho hàm kiến tạo
- ifstream inStream;
  inStream.open("infile.txt");

  tương đương với:

  ifstream inStream("infile.txt");

Kiểm tra mở tệp thành công

- Thao tác mở tệp có thể thất bại
  - nếu tệp mở để đọc không tồn tại, hoặc
  - không có quyền ghi vào tệp xuất
  - Kết quả không lường trước được
- Hàm thành viên fail()
  - Gọi tới fail() để kiểm tra xem thao tác trên luồng có thành công hay không

```
inStream.open("stuff.txt");
if (inStream.fail())
{
  cout << "File open failed.\n";
  exit(1);
}
```

Đọc ghi kí tự với tệp

- Tất cả các thao tác đọc/ghi kí tự trên cin và cout đều áp dụng được cho tệp!
- Các hàm thành viên hoạt động tương tự:
  - get, getline
  - put, putback,
  - peek, ignore

Kiểm tra cuối tệp (1/2)

- Thường thao tác với tệp bằng cách lặp để xử lý từ đầu đến cuối
  - Điều kiện dừng lặp: đã đến cuối tệp chưa?
- Có 2 cách kiểm tra cuối tệp
  1. Dùng hàm thành viên eof()

```
inStream.get(next);
while (!inStream.eof())
{
    cout << next;
    inStream.get(next);
}
```

     - Đọc từng kí tự tới khi đến cuối tệp
     - Hàm thành viên eof() trả về giá trị bool

Kiểm tra cuối tệp (2/2)

2. Dùng chính lệnh đọc

– Thao tác đọc trả về giá trị bool
(inStream >> next)
  - Biểu thức trả về true nếu đọc thành công
  - Trả về false nếu lệnh đọc vượt khỏi điểm cuối tệp

– Ví dụ:

```
double next, sum = 0;
while (inStream >> next)
    sum = sum + next;
cout << "the sum is " << sum << endl;
```

Công cụ: Tên tệp

- Thao tác mở luồng

```
void open(const char * filename, ios_base:openmode mode = ios_base::in);
```

  - Đối số của hàm open() có kiểu xâu C
  - Có thể là giá trị hằng (như ta làm trong các ví dụ trước) hoặc biến

```
char fileName[16];
ifstream inStream;
cout << "Enter file name: ";
cin >> fileName;
inStream.open(fileName);
```

  - Linh hoạt hơn

Định dạng dữ liệu xuất bằng hàm thành viên của luồng

- "Công thức màu nhiệm" trong chương 1:

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
```

- Định dạng in số tới 2 chữ số sau dấu phẩy (12.52)
- Có thể dùng cho bất cứ luồng xuất nào
  - Luồng tệp cũng có các hàm thành viên giống hệt đối tượng cout

Hàm thành viên của luồng xuất

- Xét:

```
outStream.setf(ios::fixed);
outStream.setf(ios::showpoint);
outStream.precision(2);
```

- Hàm thành viên precision(x)
  - Viết số thập phân tới x chữ số sau dấu phẩy
- Hàm thành viên setf()
  - Cho phép thiết đặt một số cờ trên dữ liệu xuất

Hàm thành viên của luồng xuất

- Xét:

  outStream.width(5);

- Hàm thành viên width(x)
  - Đặt độ rộng cột là x cho giá trị xuất
  - Chỉ có tác dụng trên giá trị xuất ngay sau đó
  - Nếu muốn tất cả các giá trị xuất đều được in với độ rộng cột là x thì phải gọi tới width cho mỗi giá trị
    - Thường thì dữ liệu xuất có độ rộng cột khác nhau

Cờ

- Hàm thành viên setf()
  - Thiết lập giá trị logic trên các cờ xuất
- Tất cả các luồng xuất đều có thành viên setf()
- Cờ là hằng trong lớp ios
  - Trong thư viện <iostream>, không gian tên std

Ví dụ setf()

- Hằng cờ thông dụng:
  - outStream.setf(ios::fixed);
    - Dùng kí hiệu dấu phẩy tĩnh (thập phân)
  - outStream.setf(ios::showPoint)
    - Sử dụng dấu chấm thập phân
  - outStream.setf(ios::right);
    - Căn lề phải
- Đặt nhiều cờ trong một lời gọi setf:

```
outStream.setf(ios::fixed | ios::showpoint |
                                    ios::right);
```

Manipulator

- Manipulator được định nghĩa là
  "một hàm được gọi khác cách truyền thống"
  - Có thể có đối số
  - Lời gọi đặt sau toán tử <<
  - Làm việc như hàm thành viên
- setw() và setprecision() nằm trong thư viện <iomanip>, không gian tên std.

Ví dụ manipulator: setw()

- Đoạn mã dùng setw():

```
cout          << "Start" << setw(4) << 10
              << setw(4) << 20 << setw(6) << 30;
```

  - Cho kết quả là:

```
Start  10  20    30
```

- Lưu ý: setw() chỉ tác dụng trên giá trị xuất ngay sau nó

Manipulator setprecision()

- Đoạn mã dùng setprecision() :

```
cout.setf(ios::fixed | ios::showpoint);
cout    << "$" << setprecision(2) << 10.3 << "  "
        << "$" << 20.5 << endl;
```

- Cho kết quả là:

$10.30 $20.50

Lưu lại thiết đặt cờ

- Giá trị thiết lập cho các cờ định dạng sẽ không đổi nếu bạn không chỉ định tường minh
- Các cờ precision và setf có thể được lưu lại để khôi phục sau đó
  - Hàm precision() trả về thiết đặt hiện thời nếu lời gọi không có đối số
  - Hàm thành viên flags() làm công việc tương tự

Ví dụ: lưu lại thiết đặt cờ

- ```
  void outputStuff(ofstream& outStream)
  {
      int precisionSetting = outStream.precision();
      long flagSettings = outStream.flags();
      outStream.setf(ios::fixed | ios::showpoint);
      outStream.precision(2);
      outStream.precision(precisionSetting);
      outStream.flags(flagSettings);
  }
  ```
- Hàm ví dụ trên lưu lại rồi khôi phục thiết đặt cờ định dạng
  - Để gọi tới nó: `outputStuff(myStream);`
  ```

Khôi phục thiết đặt setf mặc định

- Ta cũng có thể khôi phục các thiết đặt mặc định:
  ```
  cout.setf(0, ios::floatfield);
  ```
- Không nhất thiết phải là định dạng “mới nhất”!
- Các giá trị mặc định này phụ thuộc vào cài đặt
- Lệnh này không khôi phục precision mặc định

Phân cấp luồng

- Mối quan hệ giữa các lớp
  - “Dẫn xuất từ"
    - Một lớp phát triển từ một lớp khác
    - Các đặc tính riêng được bổ sung
  - Ví dụ: Lớp biểu diễn tam giác dẫn xuất từ lớp biểu diễn đa giác
  - Ví dụ: Lớp biểu diễn luồng nhập từ tệp dẫn xuất từ lớp biểu diễn luồng nhập
    - Sau đó nó bổ sung hàm thành viên open và close
  - Tức là ifstream dẫn xuất từ istream

Ví dụ thực tế của thừa kế lớp

- Lớp biểu diễn xe máy tay ga dẫn xuất từ lớp biểu diễn xe máy
  - Mỗi xe máy tay ga là một xe máy
  - Xe máy tay ga “bổ sung các đặc tính” cho xe máy

Quan hệ thừa kế giữa các lớp biểu diễn luồng

- Nếu D dẫn xuất từ B →
  - Tất cả các đối tượng kiểu D cũng có kiểu B
  - Ví dụ: một xe máy tay ga là một xe máy
- Luồng:
  - Một đối tượng ifstream cũng là một đối tượng istream
  - Nên dùng istream để định kiểu tham số
    - Sẽ chấp nhận nhiều khả năng đối số hơn.

Ví dụ: Quan hệ thừa kế giữa các lớp biểu diễn luồng

```
void twoSumVersion1(ifstream& sourceFile)//ifstream with an 'f'
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

and

```
void twoSumVersion2(istream& sourceFile)//istream without an 'f'
{
    int n1, n2;
    sourceFile >> n1 >> n2;
    cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

Ví dụ: Quan hệ thừa kế giữa các lớp biểu diễn luồng

- Xét 2 hàm ở trang trước:
- twoSumVersion1(fileIn); // Hợp lệ!
- twoSumVersion1(cin); // Không hợp lệ!
  - Vì cin không có kiểu ifstream!
- twoSumVersion2(fileIn); // Hợp lệ!
- twoSumVersion2(cin); // Hợp lệ!
  - Linh hoạt hơn
  - Tham số kiểu istream chấp nhận cả 2 đối tượng

Truy cập ngẫu nhiên vào tệp

- Truy cập tuần tự
  - là phương thức truy cập thông dụng nhất
- Truy cập ngẫu nhiên
  - Truy cập nhanh tới các bản ghi
  - Có thể trong cơ sở dữ liệu rất lớn
  - Truy cập ngẫu nhiên tới bất cứ vị trí nào trong tệp
  - Dùng đối tượng fstream
    - cả nhập và xuất

Công cụ truy cập ngẫu nhiên

- Thao tác mở giống như istream hay ostream
  - Thêm đối số thứ 2
  - fstream rwStream;
    rwStream.open("stuff.dat", ios::in | ios:: out);
    - Mở tệp có tên là stuff.dat để đọc và ghi
- Di chuyển trong tệp
  - rwStream.seekp(1000);
    - Đặt con trỏ put ở byte thứ 1000
  - rwStream.seekg(1000);
    - Đặt con trỏ get ở byte thứ 1000

Kích cỡ truy cập ngẫu nhiên

- Để có thể di chuyển → ta cần biết kích cỡ
  - toán tử sizeof() xác định số byte cần cho mỗi đối tượng:
    sizeof(s) // trong đó s là biến string s = "Hello"
    sizeof(10)
    sizeof(double)
    sizeof(myObject)
  - Đặt con trỏ put ở bản ghi đối tượng thứ 100:
    rwStream.seekp(100*sizeof(myObject) – 1);

Tóm tắt 1

- Luồng kết nối với tệp nhờ thao tác mở tệp
- Hàm thành viên fail() kiểm tra xem thao tác mở tệp thành công hay không
- Các hàm thành viên định dạng dữ liệu xuất
  - width, setf, precision
  - Cách dùng với tệp tương tự cách dùng với màn hình (cout)
- Kiểu luồng có thể là kiểu của tham số hình thức
  - Nhưng cần được truyền tham chiếu

Tóm tắt 2

- Tham số kiểu istream (không có chữ "f") chấp nhận đối số là cin hoặc đối tượng ifstream
- Tham số kiểu ostream (không có chữ "f") chấp nhận đối số là cout hoặc đối tượng ofstream
- Hàm thành viên eof
  - Dùng để kiểm tra xem đã đọc tới vị trí cuối tệp hay chưa

Chuẩn bị bài tới

- Đọc chương 14 giáo trình (Thừa kế), chương 18 giáo trình (Xử lý ngoại lệ)