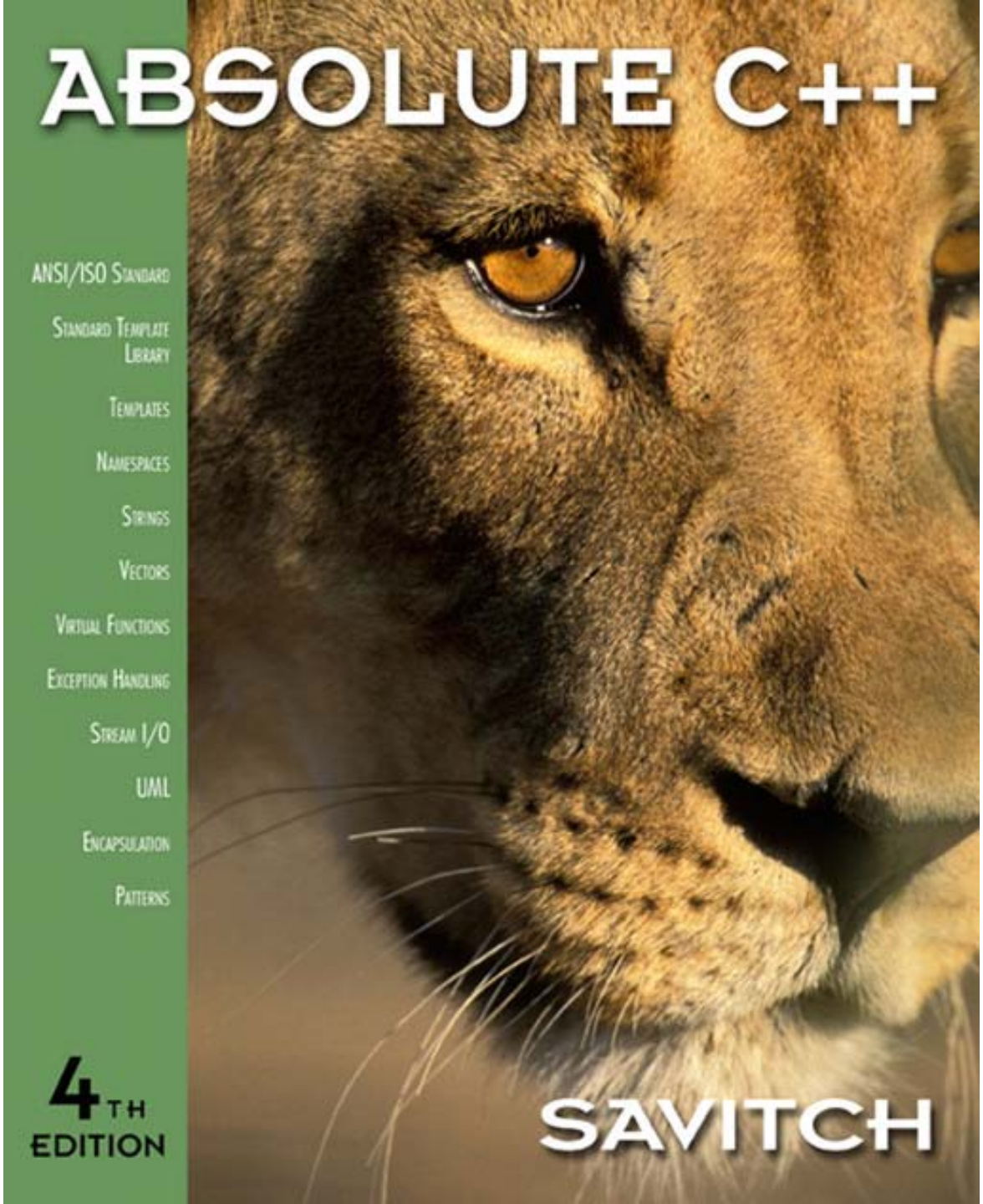
Bài 6: struct và class

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



Chapter 6

Structures and Classes

PEARSON Addison Wesley

Mục tiêu bài học

- struct
  - Kiểu định nghĩa bằng struct
  - struct làm đối số của hàm
  - Khởi tạo struct
- class
  - Định nghĩa, hàm thành viên
  - Thành viên public và private
  - Hàm truy cập và hàm biến đổi dữ liệu
  - So sánh struct và class

struct

- Là kiểu dữ liệu nhóm thứ 2 trong cua học
- Nhắc lại:
  - Mảng: tập hợp các giá trị cùng kiểu
  - struct: tập hợp các giá trị khác kiểu
- Có thể được xử lý như một thực thể, giống mảng
- Khác biệt quan trọng: Phải định nghĩa struct
  - Trước khi khai báo bất cứ biến nào

Kiểu định nghĩa bằng struct

- Thường được định nghĩa toàn cục
- Bộ nhớ không được cấp phát khi bạn định nghĩa struct
  - Chỉ là "chỗ đặt trước" để biết struct của ta sẽ như thế nào
- Định nghĩa:

```
struct CDAccountV1   ←        tên của "kiểu" mới
                              định nghĩa bằng struct
{
        double balance;  ←    tên thành viên
        double interestRate;
        int term;
};
```

Khai báo biến struct

- Khi đã định nghĩa struct, bạn có thể khai báo các biến thuộc kiểu mới này:
  CDAccountV1 account;
  - Giống như khai báo các kiểu đơn
  - Biến account có kiểu CDAccountV1
  - Nó chứa các giá trị thành viên
    - Mỗi thành viên là một phần của struct

Truy cập thành viên struct

- Dùng toán tử dấu chấm để truy cập thành viên
  - account.balance
  - account.interestRate
  - account.term
- Được gọi là “các biến thành viên”
  - Là các phần của biến struct
  - Các struct khác nhau có thể có biến thành viên trùng tên
    - Không xung đột

Ví dụ struct:
Display 6.1 Một định nghĩa struct (1/3)

Display 6.1 A Structure Definition

```
1  //Program to demonstrate the CDAccountV1 structure type.
2  #include <iostream>
3  using namespace std;

4  //Structure for a bank certificate of deposit:
5  struct CDAccountV1
6  {
7      double balance;
8      double interestRate;
9      int term;//months until maturity
10 };

11 void getData(CDAccountV1& theAccount);
12 //Postcondition: theAccount.balance, theAccount.interestRate, and
13 //theAccount.term have been given values that the user entered at the keyboar
```

An improved version of this structure will be given later in this chapter.

Ví dụ struct:
Display 6.1 Một định nghĩa struct (2/3)

```
14  int main( )
15  {
16      CDAccountV1 account;
17      getData(account);

18      double rateFraction, interest;
19      rateFraction = account.interestRate/100.0;
20      interest = account.balance*(rateFraction*(account.term/12.0));
21      account.balance = account.balance + interest;

22      cout.setf(ios::fixed);
23      cout.setf(ios::showpoint);
24      cout.precision(2);
25      cout << "When your CD matures in "
26           << account.term << " months,\n"
27           << "it will have a balance of $"
28           << account.balance << endl;

29      return 0;
30  }
```

(continued)

Ví dụ struct:
Display 6.1 Một định nghĩa struct (3/3)

Display 6.1 A Structure Definition

```
31  //Uses iostream:
32  void getData(CDAccountV1& theAccount)
33  {
34      cout << "Enter account balance: $";
35      cin >> theAccount.balance;
36      cout << "Enter account interest rate: ";
37      cin >> theAccount.interestRate;
38      cout << "Enter the number of months until maturity: ";
39      cin >> theAccount.term;
40  }
```

SAMPLE DIALOGUE

Enter account balance: **$100.00**
Enter account interest rate: **10.0**
Enter the number of months until maturity: **6**
When your CD matures in 6 months,
it will have a balance of $105.00

Lỗi khi dùng struct

- Dấu chấm phẩy sau định nghĩa struct
  - PHẢI có ; :

```
struct WeatherData
{
  double temperature;
  double windVelocity;
};
```

← dấu chấm phẩy là BẮT BUỘC!

  - Bắt buộc vì bạn "có thể" khai báo biến struct ở vị trí này

Phép gán cho biến struct

- Cho trước struct có tên là CropYield
- Khai báo 2 biến struct:
  CropYield apples, oranges;
  - Cả 2 đều là biến có kiểu CropYield định nghĩa bởi struct
  - Phép gán đơn giản sau đây là hợp lệ:
    apples = oranges;
    - Thực hiện sao chép từng biến thành viên từ oranges vào apples

struct làm đối số của hàm

- Được truyền vào như các kiểu đơn
  - Truyền giá trị
  - Truyền tham chiếu
  - Hay phối hợp
- Biến struct cũng có thể là giá trị trả về của hàm
  - Kiểu trả về là kiểu struct
  - Cậu lệnh return trong định nghĩa hàm sẽ gửi biến struct về nơi gọi hàm

Khởi tạo struct

- Có thể khởi tạo lúc khai báo
  - Ví dụ:

```
struct Date
{
        int month;
        int day;
        int year;
};
Date dueDate = {12, 31, 2003};
```

  - Lệnh khai báo này cung cấp dữ liệu ban đầu cho cả 3 biến thành viên

class

- Tương tự như struct
  - Có các biến thành viên
  - Có thêm hàm thành viên?
- Liên hệ với lập trình hướng đối tượng
  - Tập trung vào các đối tượng
    - Đối tượng: chứa dữ liệu và các phép toán
    - Trong C++, các biến của kiểu định nghĩa bởi class là các đối tượng

Định nghĩa class

- Định nghĩa tương tự như struct
- Ví dụ:

```
class DayOfYear                    ← tên của kiểu mới
{
public:
        void output();             ← hàm thành viên!
        int month;
        int day;
};
```

- Chú ý là ví dụ chỉ đưa ra các nguyên mẫu hàm
  - Định nghĩa của các hàm này nằm đâu đó trong chương trình

Khai báo đối tượng

- Khai báo giống như tất cả các biến
  - Kiểu có sẵn, kiểu định nghĩa bằng struct
- Ví dụ:

  DayOfYear today, birthday;
  - Khai báo 2 đối tượng có kiểu DayOfYear
- Đối tượng bao gồm:
  - Dữ liệu
    - Các thành viên month, day
  - Các phép toán (hàm thành viên)
    - output()

Truy cập thành viên của class

- Ta truy cập tới các thành viên của class giống như làm với struct
- Ví dụ:

  today.month
  today.day

  – Và để truy cập tới hàm thành viên:
  today.output(); ← gọi hàm thành viên

Hàm thành viên của class

- Ta phải định nghĩa hay "cài đặt" các hàm thành viên của class
- Giống các định nghĩa hàm khác
  - Có thể đặt sau định nghĩa main()
  - Phải chỉ định class:
    void DayOfYear::output()
    {…}
    - :: là toán tử chỉ định phạm vi
    - Hướng dẫn trình biên dịch xem thành viên này tới từ class nào
    - Định danh đứng trước dấu :: được gọi là từ định kiểu (type qualifier)

Định nghĩa hàm thành viên của class

- Chú ý định nghĩa hàm thành viên output() (ở ví dụ phía sau)
- Tham chiếu tới dữ liệu thành viên của class
  - Không cần dùng từ định kiểu
- Hàm được dùng cho tất cả các đối tượng của class
  - Khi được gọi, nó sẽ tham chiếu tới dữ liệu của "đối tượng đó"
  - Ví dụ:
    today.output();
    - Hiển thị dữ liệu của đối tượng "today"

Ví dụ class hoàn chỉnh:
Display 6.3 class với 1 hàm thành viên (1/4)

Display 6.3 Class with a Member Function

```
1  //Program to demonstrate a very simple example of a class.
2  //A better version of the class DayOfYear will be given in Display 6.4.
3  #include <iostream>
4  using namespace std;

5  class DayOfYear
6  {
7  public:
8      void output( );
9      int month;
10     int day;
11 };

12 int main( )
13 {
14     DayOfYear today, birthday;
15     cout << "Enter today's date:\n";
16     cout << "Enter month as a number: ";
17     cin >> today.month;
18     cout << "Enter the day of the month: ";
19     cin >> today.day;
20     cout << "Enter your birthday:\n";
21     cout << "Enter month as a number: ";
22     cin >> birthday.month;
23     cout << "Enter the day of the month: ";
24     cin >> birthday.day;
```

Normally, member variables are `private` *and not* `public`*, as in this example. This is discussed a bit later in this chapter.*

Member function declaration

(continued)

Ví dụ class hoàn chỉnh: **Display 6.3** class với 1 hàm thành viên (2/4)

Display 6.3 Class with a Member Function

```
25      cout << "Today's date is ";
26      today.output( );
27      cout << endl;
28      cout << "Your birthday is ";
29      birthday.output( );
30      cout << endl;

31      if (today.month == birthday.month && today.day == birthday.day)
32          cout << "Happy Birthday!\n";
33      else
34          cout << "Happy Unbirthday!\n";
35      return 0;
36  }
37  //Uses iostream:
38  void DayOfYear::output( )
39  {
40      switch (month)
41      {
42          case 1:
43              cout << "January "; break;
44          case 2:
45              cout << "February "; break;
46          case 3:
47              cout << "March "; break;
48          case 4:
49              cout << "April "; break;
```

Calls to the member function **output**

Member function definition

Ví dụ class hoàn chỉnh:
Display 6.3 class với 1 hàm thành viên (3/4)

```
50          case 5:
51              cout << "May "; break;
52          case 6:
53              cout << "June "; break;
54          case 7:
55              cout << "July "; break;
56          case 8:
57              cout << "August "; break;
58          case 9:
59              cout << "September "; break;
60          case 10:
61              cout << "October "; break;
62          case 11:
63              cout << "November "; break;
64          case 12:
65              cout << "December "; break;
66          default:
67              cout << "Error in DayOfYear::output. Contact software vendor.";
68      }
69
70      cout << day;
71  }
```

Ví dụ class hoàn chỉnh:
Display 6.3 class với 1 hàm thành viên (4/4)

Display 6.3 Class with a Member Function

SAMPLE DIALOGUE

```
Enter today's date:
Enter month as a number: 10
Enter the day of the month: 15
Enter your birthday:
Enter month as a number: 2
Enter the day of the month: 21
Today's date is October 15
Your birthday is February 21
Happy Unbirthday!
```

Toán tử dấu chấm và toán tử phân tích phạm vi

- Dùng để xác định xem dữ liệu/hàm là thành viên của cái gì
- Toán tử dấu chấm (.)
  - Chỉ định thành viên của một đối tượng cụ thể
- Toán tử phân tích phạm vi (::)
  - Chỉ định xem định nghĩa hàm tới từ class nào

class

- Kiểu định nghĩa bởi class là một kiểu hoàn thiện
  - Giống int, double...
- Có thể có các biến thuộc kiểu định nghĩa bởi class
  - Ta gọi chúng là đối tượng
- Có thể có tham số thuộc kiểu class
  - Truyền giá trị
  - Truyền tham chiếu
- Có thể sử dụng kiểu class như những kiểu khác

Tính đóng gói

- Bất cứ kiểu dữ liệu nào cũng bao gồm
  - Dữ liệu (miền dữ liệu)
  - Các phép toán (có thể thực hiện trên dữ liệu)
- Ví dụ:

  kiểu dữ liệu int có:
  Dữ liệu: +-32,767
  Các phép toán: +,-,*,/,%,các phép logic.
- Kiểu class cũng như vậy
  - Nhưng ta cần chỉ định dữ liệu và các phép toán được phép thực hiện trên dữ liệu!

Kiểu dữ liệu trừu tượng

- “Trừu tượng"
  - Lập trình viên không cần biết chi tiết cài đặt
- Viết tắt là "ADT"
  - Abstract Data Type
  - Tập hợp các giá trị dữ liệu cùng với tập các phép toán định nghĩa cho các giá trị đó
- ADT thường độc lập với ngôn ngữ
  - Ta cài đặt ADT trong C++ bằng class
    - class của C++ định nghĩa ADT
  - Các ngôn ngữ khác cũng cài đặt ADT

Bàn thêm về tính đóng gói

- Đóng gói
  - có nghĩa là “thu dữ liệu về một mối”
- Khai báo một class → Sinh 1 đối tượng
- Đối tượng là “bao đóng” của
  - Các giá trị dữ liệu
  - Các phép toán trên dữ liệu (các hàm thành viên)

Nguyên lý lập trình hướng đối tượng

- Che giấu thông tin
  - "Người dùng" một class không biết chi tiết các bước trong mỗi phép toán
- Trừu tượng hóa dữ liệu
  - "Người dùng" một ADT/class không biết chi tiết các bước xử lý dữ liệu bên trong ADT/class
- Đóng gói
  - Thu dữ liệu và phép toán về một mối, nhưng giấu đi các "chi tiết"

Thành viên public và private

- Dữ liệu trong class thường được chỉ định là private trong định nghĩa!
  - Nguyên lý được khuyến khích trong LTHĐT
  - Che giấu dữ liệu khỏi người dùng
  - Chỉ cho phép xử lý thông qua các phép toán
    - tức các hàm thành viên
- Thành viên public (thường là hàm thành viên) có thể truy cập bởi người dùng

public và private: Ví dụ 1

- Chỉnh sửa ví dụ trước:

```
class DayOfYear
{
public:
        void input();
        void output();
private:
        int month;
        int day;
};
```

- Dữ liệu hiện giờ là private
- Các đối tượng không thể truy cập trực tiếp vào dữ liệu

public và private: Ví dụ 2

- Dùng tiếp ví dụ trước
- Khai báo đối tượng:
  DayOfYear today;
- Đối tượng today chỉ có thể truy cập các thành viên public
  - cin >> today.month; // KHÔNG ĐƯỢC PHÉP!
  - cout << today.day; // KHÔNG ĐƯỢC PHÉP!
  - Bạn phải gọi tới các phép toán public:
    - today.input();
    - today.output();

Phong cách public và private

- Có thể trộn public & private
- Thường thì public được đặt trước tiên
  - Giúp lập trình viên dễ quan sát được phần nào của class mình có thể sử dụng
  - Dữ liệu private bị "giấu đi" nên nó không liên quan tới người dùng
- Bên ngoài định nghĩa class, ta không thể thay đổi (hay truy cập) dữ liệu private

Hàm truy cập (accessor) và hàm biến đổi (mutator)

- Đối tượng cần làm gì đó với dữ liệu của nó
- Gọi tới hàm thành viên truy cập (accessor)
  - Cho phép đối tượng đọc dữ liệu
  - Còn gọi là hàm thành viên get
  - Chỉ đơn thuần truy xuất dữ liệu thành viên
- Hàm thành viên biến đổi (mutator)
  - Cho phép đối tượng biến đổi dữ liệu
  - Được sử dụng tùy theo ứng dụng

Tách giao diện và cài đặt

- Người dùng một class không cần hiểu chi tiết từng bước cài đặt class
  - Nguyên lý LTHĐT → Tính đóng gói
- Người dùng chỉ cần biết "các quy tắc"
  - Được gọi là "giao diện" của class
    - Trong C++ → các hàm thành viên public và chú thích đi kèm
- Cài đặt của class được ẩn đi
  - Định nghĩa hàm thành viên nằm đâu đó
  - Người dùng không cần thấy chúng

So sánh struct và class

- Kiểu định nghĩa bởi struct
  - Thường thì tất cả các thành viên đều là public
  - Không có hàm thành viên?
- Kiểu định nghĩa bởi class
  - Thường thì tất cả các thành viên đều là private
  - Những hàm thành viên giao diện được để public
- Nói chính xác thì chúng giống hệt nhau
  - Theo trực giác thì chúng có cơ chế khác nhau

Tư duy hướng đối tượng

- Trung tâm của lập trình
  - Trước đây → thuật toán là trung tâm
  - LTHĐT → dữ liệu là trung tâm
- Thuật toán vẫn tồn tại
  - Đơn giản là họ tập trung vào dữ liệu
  - Được "sinh ra" để nhắm vào dữ liệu
- Thiết kế các giải pháp phần mềm
  - Định nghĩa nhiều đối tượng và cách chúng tương tác với nhau

Tóm tắt 1

- struct là một tập các kiểu khác nhau
- class được dùng để kết hợp dữ liệu và hàm thành một đơn vị → đối tượng
- Các biến và hàm thành viên
  - Có thể là public → truy cập được từ ngoài class
  - Có thể là private → chỉ truy cập được trong định nghĩa hàm thành viên
- Kiểu định nghĩa bởi class và struct có thể là kiểu của tham số hình thức của hàm

Tóm tắt 2

- Định nghĩa class trong C++
  - Nên tách thành 2 phần chính
    - Giao diện: những gì người sử dụng cần biết
    - Cài đặt: chi tiết hoạt động của class

Chuẩn bị bài tới

- Đọc chương 7 giáo trình: Hàm kiến tạo và các công cụ khác