



NGÔN NGỮ LẬP TRÌNH

Bài 4: Cấu trúc và lớp

Giảng viên: Lý Anh Tuấn

Email: tuarla@tlu.edu.vn

Nội dung

1. Cấu trúc

- Các kiểu cấu trúc
- Cấu trúc là đối số hàm
- Khởi tạo cấu trúc

2. Lớp

- Định nghĩa, hàm thành viên
- Các thành phần public và private
- Hàm truy cập và hàm biến đổi

3. Hàm tạo

- Định nghĩa
- Lời gọi

4. Các công cụ khác

- Bỏ từ const cho các tham số
- Hàm trực tuyến
- Dữ liệu thành viên tĩnh

Cấu trúc

- Là kiểu dữ liệu gộp giống như mảng
- Tuy nhiên:
 - Mảng là tập các giá trị có cùng kiểu
 - Cấu trúc là tập các giá trị có kiểu khác nhau
- Định nghĩa cấu trúc:
 - Trước khi khai báo biến
 - Phạm vi toàn cục
 - Không cấp phát bộ nhớ

Cấu trúc

- Ví dụ:

```
struct CDAccountVI ← tên của kiểu cấu trúc mới
{
    double balance;      ← tên thành viên
    double interestRate;
    int term;
};
```

- Khai báo biến cho kiểu mới này

```
CDAccountVI account;
```

- Giống như khai báo các kiểu đơn giản
- Biến account có kiểu là CDAccountVI
- Nó bao chứa các giá trị thành viên

Truy cập các thành viên cấu trúc

- Sử dụng toán tử . để truy cập tới các thành viên
 - `account.balance`
 - `account.interestRate`
 - `account.term`
- Các biến thành viên
 - Là thành phần của biến cấu trúc
 - Các cấu trúc khác nhau có thể có các biến thành viên cùng tên

Ví dụ về cấu trúc

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 về cấu trúc

```
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 về cấu trúc

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 thường gặp với cấu trúc

- Quên dấu chấm phẩy sau định nghĩa cấu trúc
struct WeatherData
{
 double temperature;
 double windVelocity;
}; ← Cần có dấu chấm phẩy!
- Bạn có thể khai báo các biến cấu trúc ở vị trí này

Phép gán cấu trúc

- Cho trước một cấu trúc tên là CropYield
- Khai báo hai biến cấu trúc:

CropYield apples, oranges;

- Cả hai biến là kiểu cấu trúc CropYield
- Cho phép thực hiện phép gán đơn giản:
apples = oranges;
- Việc này sao chép mỗi biến thành viên của apples thành biến thành viên của oranges

Cấu trúc là đối số hàm

- Có thể được truyền giống như các kiểu dữ liệu đơn giản
 - Truyền giá trị
 - Truyền tham chiếu
 - Hoặc kết hợp
- Cũng có thể được trả về bởi hàm
 - Kiểu trả về là kiểu cấu trúc
 - Lệnh trả về trong định nghĩa hàm gửi biến cấu trúc trở về cho lời gọi

Khởi tạo cấu trúc

- Có thể khởi tạo lúc khai báo

- Ví dụ:

```
struct Date
```

```
{
```

```
    int month;
```

```
    int day;
```

```
    int year;
```

```
};
```

```
Date dueDate = {12, 31, 2003};
```

- Khai báo cung cấp dữ liệu khởi tạo cho cả ba biến thành viên

Lớp

- Tương tự như cấu trúc, lớp bao gồm:
 - Các dữ liệu thành viên (giống cấu trúc)
 - Có thêm các hàm thành viên
- Tích hợp cho lập trình hướng đối tượng
 - Tập trung vào các đối tượng
 - Đối tượng: Bao gồm dữ liệu và các thao tác
 - Trong C++, các biến kiểu lớp là các đối tượng

Định nghĩa lớp

- Được định nghĩa tương tự cấu trúc
- Ví dụ:

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

- Lưu ý: chỉ có nguyên mẫu của hàm thành viên, thi hành hàm nằm ở nơi khác

Khai báo các đối tượng

- Được khai báo tự động các biến
 - Kiểu định nghĩa trước, kiểu cấu trúc
- Ví dụ: `DayOfYear today, birthday;`
 - Khai báo hai đối tượng của kiểu lớp `DayOfYear`
- Các đối tượng bao gồm
 - Dữ liệu: Các thành viên `month`, `day`
 - Các thao tác (các hàm thành viên): `output()`

Truy cập thành viên lớp

- Các thành viên được truy cập tương tự như cấu trúc
- Ví dụ:
 - `today.month`
 - `today.day`
- Để truy cập đến hàm thành viên:
 - `today.output();` ← Gọi hàm thành viên

Hàm thành viên lớp

- Phải định nghĩa hoặc thi hành các hàm thành viên lớp
- Giống với các định nghĩa hàm khác
 - Có thể đặt sau định nghĩa main()
 - Phải chỉ rõ thuộc lớp nào
- VD:

```
void DayOfYear::output()  
{...}
```

- :: là toán tử phân giải phạm vi
- Nói cho trình biên dịch thành viên là từ lớp nào
- Tên lớp trước :: được gọi là định kiểu

Định nghĩa hàm thành viên lớp

- Xem định nghĩa của hàm thành viên `output()` (trong ví dụ tiếp theo)
- Tham chiếu đến dữ liệu của lớp
- Hàm được sử dụng cho tất cả các đối tượng của lớp
 - Tham chiếu đến dữ liệu của đối tượng đó khi được gọi
 - VD: `today.output();` //hiển thị dữ liệu của đối tượng `today`

Ví dụ lớp hoàn chỉnh

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 lớp hoàn chỉnh

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 lớp hoàn chỉnh

```
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 lớp hoàn chỉnh

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!

Vai trò của lớp

- Lớp là một kiểu đầy đủ: giống như các kiểu dữ liệu int, double, vân vân
- Biến của một kiểu lớp: được gọi là đối tượng
- Tham số của một kiểu lớp: có thể truyền giá trị, truyền tham biến
- Có thể sử dụng kiểu lớp giống như bất kỳ kiểu dữ liệu nào khác

Đóng gói

- Kiểu dữ liệu bất kỳ bao gồm
 - Dữ liệu (phạm vi của dữ liệu)
 - Các thao tác (có thể được thi hành trên dữ liệu)
- Ví dụ: kiểu dữ liệu int có

Dữ liệu: +-32,767

Các thao tác: +, -, *, /, %, so sánh, vân vân
- Với lớp: chúng ta chỉ ra dữ liệu, còn các thao tác phụ thuộc vào dữ liệu
- Đối tượng là đóng gói của giá trị giữ liệu và các thao tác trên dữ liệu

Nguyên tắc lập trình hướng đối tượng

- Ẩn thông tin: người dùng lớp không được biết chi tiết về các thao tác làm việc như thế nào
- Trừu tượng dữ liệu: người dùng không được biết chi tiết về dữ liệu được vận hành như thế nào trong ATD/lớp
- Đóng gói: Buộc dữ liệu và các thao tác với nhau, nhưng ẩn đi các chi tiết

Thành viên public và private

- Dữ liệu trong lớp thường được khai báo private trong định nghĩa
 - Duy trì các quy tắc của OOP
 - Ẩn dữ liệu với người dùng
 - Chỉ cho phép vận hành bởi các thao tác (hàm thành viên)
- Các mục public (thường là các hàm thành viên) cho phép người dùng truy cập

Ví dụ public và private

- Sửa ví dụ trước:

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

- Dữ liệu bây giờ là private
- Các đối tượng không thể truy cập trực tiếp
- public thường được đặt trước private

Ví dụ public và private

- Sử dụng ví dụ trước
- Khai báo đối tượng:
DayOfYear today;
- Đối tượng today chỉ có thể truy cập các hàm thành viên
 - cin >> today.month; // Không được phép!
 - cout << today.day; // Không được phép!
- Thay vào đó phải gọi các thao tác public
 - today.input();
 - today.output();

Hàm truy cập và hàm biến đổi

- Đối tượng cần làm việc với dữ liệu của nó
- Hàm thành viên truy cập
 - Cho phép đối tượng đọc dữ liệu
 - Còn được gọi “hàm thành viên get”
 - Truy hồi dữ liệu thành viên
- Hàm thành viên biến đổi
 - Cho phép đối tượng thay đổi dữ liệu
 - Thao tác dựa vào ứng dụng

Tách biệt giao diện và sự thực thi

- Người dùng lớp không cần xem chi tiết về việc lớp được thực thi như thế nào
- Người dùng chỉ cần các luật
 - Được gọi là giao diện của lớp
 - Trong C++, bao gồm các hàm thành viên và các chú thích liên quan
- Sự thi hành của lớp được ẩn đi:
 - Các định nghĩa hàm thành viên nằm ở chỗ khác
 - Người dùng không cần thấy chúng

Hàm tạo

- Khởi tạo các đối tượng
 - Khởi tạo một vài hoặc tất cả các biến thành viên
 - Cũng cho phép thực hiện các hành động khác
- Một kiểu hàm thành viên đặc biệt
 - Được gọi tự động khi khai báo đối tượng
- Là một công cụ hữu ích
 - Là nguyên tắc cơ bản của lập trình hướng đối tượng

Định nghĩa hàm tạo

- Giống như các hàm thành viên khác ngoại trừ:
 - Phải có cùng tên với tên lớp
 - Không trả về giá trị, thậm chí là void
- VD: Định nghĩa lớp với hàm tạo

```
class DayOfYear
{
public:
    DayOfYear(int monthValue, int dayValue);
        //Hàm tạo khởi tạo month & day
    void input();
    void output();
    ...
private:
    int month;
    int day;
}
```


Gọi hàm tạo

- Khai báo đối tượng:

DayOfYear date1 (7, 4),
date2(5, 5);

- Các đối tượng được tạo theo cách:
 - Hàm tạo được gọi
 - Các giá trị trong ngoặc được truyền như là các đối số cho hàm tạo
 - Các biến thành viên month, day được khởi tạo:
date1.month → 7 date1.day → 4
date2.month → 5 date2.day → 5

Gọi hàm tạo

- Xét ví dụ:

DayOfYear date1, date2

date1.DayOfYear(7, 4); // Không hợp lệ!

date2.DayOfYear(5, 5); // Không hợp lệ!

- → Không thể gọi hàm tạo giống như các hàm thành viên khác

Định nghĩa hàm tạo

- Giống như các hàm thành viên khác:

```
DayOfYear::DayOfYear(int monthValue, int dayValue)
{
    month = monthValue;
    day = dayValue;
}
```

- Một cách định nghĩa khác

```
DayOfYear::DayOfYear(                int monthValue,
                                     int dayValue)
    : month(monthValue), day(dayValue) ←
{...}
```

- Dòng thứ 3 được gọi là “phần khởi tạo”
- Thân để trống

Nạp chồng hàm tạo

- Có thể nạp chồng hàm tạo giống như những hàm khác
- Nhắc lại: một ký hiệu bao gồm
 - Tên hàm
 - Danh sách tham số
- Cung cấp các hàm tạo với tất cả các danh sách tham số có thể có

Ví dụ hàm tạo

Display 7.1 Class with Constructors

```
1  #include <iostream>
2  #include <cstdlib> //for exit
3  using namespace std;

4  class DayOfYear
5  {
6  public:
7      DayOfYear(int monthValue, int dayValue);
8      //Initializes the month and day to arguments.

9      DayOfYear(int monthValue);
10     //Initializes the date to the first of the given month.

11     DayOfYear( ); ←————— default constructor
12     //Initializes the date to January 1.

13     void input();
14     void output();
15     int getMonthNumber();
16     //Returns 1 for January, 2 for February, etc.
```

This definition of DayOfYear is an improved version of the class DayOfYear given in Display 6.4.


Ví dụ hàm tạo

```
17     int getDay( );
18 private:
19     int month;
20     int day;
21     void testDate( );
22 };
```


```
23 int main()
24 {
25     DayOfYear date1(2, 21), date2(5), date3;
26     cout << "Initialized dates:\n";
27     date1.output( ); cout << endl;
28     date2.output( ); cout << endl;
29     date3.output( ); cout << endl;
30
31     date1 = DayOfYear(10, 31);
32     cout << "date1 reset to the following:\n";
33     date1.output( ); cout << endl;
34     return 0;
35 }
```

```
36 DayOfYear::DayOfYear(int monthValue, int dayValue)
37     : month(monthValue), day(dayValue)
38 {
39     testDate( );
40 }
```

This causes a call to the default constructor. Notice that there are no parentheses.



*an explicit call to the constructor
DayOfYear::DayOfYear*



Ví dụ hàm tạo

Display 7.1 Class with Constructors

```
41 DayOfYear::DayOfYear(int monthValue) : month(monthValue), day(1)
42 {
43     testDate( );
44 }

45 DayOfYear::DayOfYear( ) : month(1), day(1)
46 { /*Body intentionally empty.*/}

47 //uses iostream and cstdlib:
48 void DayOfYear::testDate( )
49 {
50     if ((month < 1) || (month > 12))
51     {
52         cout << "Illegal month value!\n";
53         exit(1);
54     }
55     if ((day < 1) || (day > 31))
56     {
57         cout << "Illegal day value!\n";
58         exit(1);
59     }
60 }
```

<Definitions of the other member functions are the same as in Display 6.4.>

SAMPLE DIALOGUE

Initialized dates:
February 21
May 1
January 1
date1 reset to the following:
October 31

Hàm tạo không đối số

- Tránh nhầm lẫn với hàm chuẩn không đối số
- Gọi hàm chuẩn không đối số :
`callMyFunction();`
- Khai báo đối tượng không có các khởi tạo:
`DayOfYear dateI; // Đúng!`
`DayOfYear date(); // Sai!`

Gọi hàm tạo trường minh

- Có thể gọi lại hàm tạo sau khi đối tượng được khai báo
- Lời gọi như vậy tạo ra một “đối tượng vô danh”, sau đó được gán cho đối tượng hiện tại
- Ví dụ:

`DayOfYear holiday(7, 4);`

- Hàm tạo được gọi ở thời điểm khai báo đối tượng
- Sau đó được khởi tạo lại:
`holiday = DayOfYear(5, 5);`

Hàm tạo mặc định

- Được định nghĩa là hàm tạo không đối
- Nên định nghĩa nó trong mọi trường hợp
- Được khởi tạo tự động nếu không định nghĩa hàm tạo nào
- Không được khởi tạo tự động nếu đã định nghĩa ít nhất một hàm tạo
- Nếu không có hàm tạo mặc định
 - Không thể khai báo: `MyClass myObject;`

Biến thành viên kiểu lớp

- Biến thành viên lớp có thể là một đối tượng có kiểu là một lớp khác
- Có một ký pháp đặc biệt:
 - Cho phép gọi hàm tạo của đối tượng thành viên
 - Bên trong hàm tạo của lớp phía ngoài

Ví dụ biến thành viên lớp

Display 7.3 A Class Member Variable

```
1  #include <iostream>
2  #include<cstdlib>
3  using namespace std;

4  class DayOfYear
5  {
6  public:
7      DayOfYear(int monthValue, int dayValue);
8      DayOfYear(int monthValue);
9      DayOfYear( );
10     void input( );
11     void output( );
12     int getMonthNumber( );
13     int getDay( );
14 private:
15     int month;
16     int day;
17     void testDate( );
18 };
```

The class DayOfYear is the same as in Display 7.1, but we have repeated all the details you need for this discussion.

Ví dụ biến thành viên lớp

```
19 class Holiday
20 {
21 public:
22     Holiday( );//Initializes to January 1 with no parking enforcement
23     Holiday(int month, int day, bool theEnforcement);
24     void output( );
25 private:
26     DayOfYear date;
27     bool parkingEnforcement;//true if enforced
28 };

29 int main( )
30 {
31     Holiday h(2, 14, true);
32     cout << "Testing the class Holiday.\n";
33     h.output( );
34
35     return 0;
36 }

37 Holiday::Holiday( ) : date(1, 1), parkingEnforcement(false)
38 { /*Intentionally empty*/}

39 Holiday::Holiday(int month, int day, bool theEnforcement)
40     : date(month, day), parkingEnforcement(theEnforcement)
41 { /*Intentionally empty*/}
```

member variable of a class type

Invocations of constructors from the class DayOfYear.

(continued)

Ví dụ biến thành viên lớp

Display 7.3 A Class Member Variable

```
42 void Holiday::output( )
43 {
44     date.output( );
45     cout << endl;
46     if (parkingEnforcement)
47         cout << "Parking laws will be enforced.\n";
48     else
49         cout << "Parking laws will not be enforced.\n";
50 }

51 DayOfYear::DayOfYear(int monthValue, int dayValue)
52                     : month(monthValue), day(dayValue)
53 {
54     testDate( );
55 }
```

Ví dụ biến thành viên lớp

```
56 //uses iostream and cstdlib:
57 void DayOfYear::testDate( )
58 {
59     if ((month < 1) || (month > 12))
60     {
61         cout << "Illegal month value!\n";
62         exit(1);
63     }
64     if ((day < 1) || (day > 31))
65     {
66         cout << "Illegal day value!\n";
67         exit(1);
68     }
69 }
70
71 //Uses iostream:
72 void DayOfYear::output( )
73 {
74     switch (month)
75     {
76     case 1:
77         cout << "January "; break;
78     case 2:
79         cout << "February "; break;
80     case 3:
81         cout << "March "; break;
82         .
83         .
84         .
```

The omitted lines are in Display 6.3, but they are obvious enough that you should not have to look there.

Ví dụ biến thành viên lớp

Display 7.3 A Class Member Variable

```
82         case 11:
83             cout << "November "; break;
84         case 12:
85             cout << "December "; break;
86         default:
87             cout << "Error in DayOfYear::output. Contact software vendor.";
88     }

89     cout << day;
90 }
```

SAMPLE DIALOGUE

Testing the class Holiday.
February 14
Parking laws will be enforced.

Các phương pháp truyền tham số

- Hiệu quả của việc truyền tham số
 - Truyền giá trị
 - Truyền tham biến
 - Không khác biệt với các kiểu đơn giản
 - Với kiểu lớp -> lợi ích rõ rệt
- Nên sử dụng truyền tham biến
 - Cho dữ liệu “lớn”, chẳng hạn như kiểu lớp

BỔ TỪ const cho các tham số

- Những kiểu dữ liệu lớn (chẳng hạn các lớp)
 - Nên sử dụng phương pháp truyền tham biến
 - Thậm chí hàm không thực hiện sửa đổi gì
- Bảo vệ đối số
 - Sử dụng tham số hằng còn được gọi là tham số truyền tham biến hằng
 - Đặt từ khóa const trước kiểu
 - Làm cho tham số chỉ đọc
 - Mọi nỗ lực sửa đổi sẽ dẫn đến lỗi biên dịch
- Áp dụng cho các tham số hàm thành viên lớp

Hàm trực tuyến

- Với hàm không phải là hàm thành viên:
 - Sử dụng từ khóa *inline* trong khai báo hàm và đầu mục hàm
- Với hàm thành viên lớp
 - Đặt thi hành của hàm trong định nghĩa lớp -> trực tuyến tự động
- Chỉ sử dụng cho những hàm rất ngắn
- Mã lệnh thực sự được chèn vào nơi gọi
 - Loại bỏ phụ phí
 - Hiệu quả hơn, nhưng chỉ sử dụng với hàm ngắn

Thành viên tĩnh

- Biến thành viên tĩnh
 - Tất cả đối tượng của lớp chia sẻ một bản sao
 - Một đối tượng thay đổi nó → tất cả đều thấy sự thay đổi
- Sử dụng cho việc “giám sát”
 - Một hàm thành viên có được gọi thường xuyên không
 - Có bao nhiêu đối tượng tồn tại ở một thời điểm cho trước
- Đặt từ khóa static trước kiểu

Hàm tĩnh

- Hàm thành viên có thể là tĩnh
 - Nếu không truy cập tới dữ liệu đối tượng cần thiết
 - Và vẫn là thành viên của lớp
 - Làm cho nó trở thành một hàm tĩnh
- Có thể được gọi bên ngoài lớp
 - Từ các đối tượng không lớp
VD: `Server::getTurn();`
 - Và bởi các đối tượng lớp
VD: `myObject.getTurn();`
- Chỉ có thể sử dụng dữ liệu tĩnh, hàm tĩnh

Ví dụ thành viên tĩnh

Display 7.6 Static Members

```
1  #include <iostream>
2  using namespace std;

3  class Server
4  {
5  public:
6      Server(char letterName);
7      static int getTurn( );
8      void serveOne( );
9      static bool stillOpen( );
10 private:
11     static int turn;
12     static int lastServed;
13     static bool nowOpen;
14     char name;
15 };

16 int Server::turn = 0;
17 int Server::lastServed = 0;
18 bool Server::nowOpen = true;
```

Ví dụ thành viên tĩnh

```
19  int main( )
20  {
21      Server s1('A'), s2('B');
22      int number, count;
23      do
24      {
25          cout << "How many in your group? ";
26          cin >> number;
27          cout << "Your turns are: ";
28          for (count = 0; count < number; count++)
29              cout << Server::getTurn( ) << ' ';
30          cout << endl;
31          s1.serveOne( );
32          s2.serveOne( );
33      } while (Server::stillOpen( ));
34
35      cout << "Now closing service.\n";
36
37      return 0;
38  }
```

Ví dụ thành viên tĩnh

Display 7.6 Static Members

```
39 Server::Server(char letterName) : name(letterName)
40 { /*Intentionally empty*/}

41 int Server::getTurn( )
42 {
43     turn++;
44     return turn;
45 }
46 bool Server::stillOpen( )
47 {
48     return nowOpen;
49 }

50 void Server::serveOne( )
51 {
52     if (nowOpen && lastServed < turn)
53     {
54         lastServed++;
55         cout << "Server " << name
56             << " now serving " << lastServed << endl;
57     }
```

← Since `getTurn` is static, only static members can be referenced in here.

Ví dụ thành viên tĩnh

```
58     if (lastServed >= turn) //Everyone served
59         nowOpen = false;
60 }
```

SAMPLE DIALOGUE

How many in your group? **3**

Your turns are: 1 2 3

Server A now serving 1

Server B now serving 2

How many in your group? **2**

Your turns are: 4 5

Server A now serving 3

Server B now serving 4

How many in your group? **0**

Your turns are:

Server A now serving 5

Now closing service.

Tóm tắt

- Cấu trúc là một tập các kiểu khác nhau
- Lớp được sử dụng để kết hợp dữ liệu và hàm vào trong một đơn vị duy nhất -> đối tượng
- Biến thành viên và hàm thành viên
 - Có thể là public → được truy cập bên ngoài lớp
 - Có thể là private → chỉ được truy cập bên trong một định nghĩa hàm
- Kiểu lớp và kiểu cấu trúc có thể làm tham số cho hàm
- Định nghĩa lớp C++ nên tách biệt hai phần chính
 - Giao diện: cái người dùng cần
 - Sự thi hành: chi tiết về việc lớp thực thi

Tóm tắt

- Hàm tạo: tự động khởi tạo dữ liệu lớp
 - Được gọi khi khi báo đối tượng
 - Hàm tạo có cùng tên với lớp
- Hàm tạo mặc định không có tham số
 - Nên được định nghĩa trong mọi trường hợp
- Biến thành viên lớp có thể là đối tượng của một lớp khác
- Có thể *trực tuyến* các định nghĩa hàm rất ngắn -> thi hành hiệu quả hơn
- Các biến thành viên tĩnh được chia sẻ bởi các đối tượng thuộc cùng một lớp