NGUYÊN LÝ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 4: Hàm tạo và các công cụ khác

Giảng viên: TS. Lý Anh Tuấn

Email: tuanla@tlu.edu.vn

Nội dung

1. Hàm tạo
   - Định nghĩa
   - Lời gọi
2. Các công cụ khác
   - Bổ từ const cho các tham số
   - Hàm nội tuyến
   - Dữ liệu thành viên tĩnh

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 dayValue, int monthValue);
                    //Hàm tạo khởi tạo day & month
    void input();
    void output();
    …
private:
    int day;
    int month;
}
```

Gọi hàm tạo

- Khai báo đối tượng:

```
DayOfYear date1(4, 7),
          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 day, month được khởi tạo:

date1.day → 4 date1.month → 7
date2.day → 5 date2.month → 5

Gọi hàm tạo

- Xét ví dụ:

```
DayOfYear date1, date2
date1.DayOfYear(4, 7);// 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 dayValue, int monthValue)
{
        day = dayValue;
        month = monthValue;
}
```

- Một cách định nghĩa khác

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

  - Dòng thứ 2 được gọi là “phần khởi tạo”
  - Phần thân để trống

Mục đích khác của hàm tạo

- Không chỉ khởi tạo dữ liệu
- Phần thân không cần để trống
  - Như trong phiên bản khởi tạo
- Dùng để xác thực dữ liệu!
  - Đảm bảo chỉ gán dữ liệu phù hợp cho các biến thành viên private

Nạp chồng hàm tạo

- Có thể nạp chồng hàm tạo giống như những hàm khác
- Nhắc lại: một tín hiệu hàm 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

```
1  #include <iostream>
2  #include <cstdlib> //de su dung exit
3  using namespace std;
4  class DayOfYear
5  {
6      public:
7          DayOfYear(int dayValue, int monthValue);
8          //Khoi tao ngay va thang bang cac doi so.
9          DayOfYear(int monthValue);
10         //Khoi tao ngay la ngay dau tien cua thang.
11         DayOfYear();
12         //Khoi tao ngay la ngay 1 Thang mot.
13         void input();
14         void output();
15         int getDay();
16         int getMonth();
17     private:
18         int day;
19         int month;
20         void testDate();
21 };
```

Hàm tạo mặc định

Ví dụ hàm tạo

```
22  int main()
23  {
24      DayOfYear date1(21, 2), date2(5), date3;
25      cout << "Cac ngay thang da khoi tao:\n";
26      date1.output(); cout << endl;
27      date2.output(); cout << endl;
28      date3.output(); cout << endl;
29      date1 = DayOfYear(31, 10);
30      cout << "date1 duoc thiet lap lai thanh:\n";
31      date1.output(); cout << endl;
32      return 0;
33  }
34
35  DayOfYear::DayOfYear(int dayValue, int monthValue)
36      : day(dayValue), month(monthValue)
37  {
38      testDate();
39  }
40  DayOfYear::DayOfYear(int monthValue) : day(1), month(monthValue)
41  {
42      testDate();
43  }
44  DayOfYear::DayOfYear() : day(1), month(1)
45  {/*Phan than ham de trong.*/}
```

Việc này gây ra một lời gọi đến hàm tạo mặc định. Lưu ý là không có cặp dấu ngoặc

Một lời gọi tường minh đến hàm tạo

Ví dụ hàm tạo

```
46  void DayOfYear::testDate()
47  {
48      if ((day < 1) || (day > 31))
49      {
50          cout << "Gia tri ngay khong hop le!\n";
51          exit(1);
52      }
53      if ((month < 1) || (month > 12))
54      {
55          cout << "Gia tri thang khong hop le!\n";
56          exit(1);
57      }
58  }
59  //Xem cac dinh nghia ham con lai trong cac vi du truoc
```

Kết quả thực hiện:

```
Cac ngay thang da khoi tao:
21 Thang Hai
1 Thang Nam
1 Thang Mot
date1 duoc thiet lap lai thanh:
31 Thang Muoi
```

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 date1;        // Đúng!
DayOfYear date();       // Sai!
```

Gọi hàm tạo tường minh

- Có thể gọi lại hàm tạo sau khi đối tượng được khai báo
- Việc này tạo ra một "đối tượng vô danh", nó sau đó được gán cho đối tượng hiện tại
- Ví dụ:
  DayOfYear holiday(4, 7);
  - Hàm tạo được gọi ở thời điểm khai báo đối tượng
  - Sau đó được goi tường minh để khởi tạo lại đối tượng:

    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?
  - Đúng: nếu không định nghĩa bất kỳ hàm tạo nào
  - Sai: 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 của 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 bao chứa

Ví dụ biến thành viên lớp

```
1  #include <iostream>
2  #include <cstdlib> //de su dung exit
3  using namespace std;
4  class DayOfYear
5  {
6      public:
7          DayOfYear(int dayValue, int monthValue);
8          //Khoi tao ngay va thang bang cac doi so.
9          DayOfYear(int monthValue);
10         //Khoi tao ngay la ngay dau tien cua thang.
11         DayOfYear();
12         //Khoi tao ngay la ngay 1 Thang Mot.
13         void input();
14         void output();
15         int getDay();
16         int getMonth();
17     private:
18         int day;
19         int month;
20         void testDate();
21 };
```

Ví dụ biến thành viên lớp

```
22  class Holiday
23  {
24      public:
25          Holiday();//Khoi tao toi 1 Thang Mot va khong ap dung Luat do xe
26          Holiday(int day, int month, bool theEnforcement);
27          void output();
28      private:
29          DayOfYear date;
30          bool parkingEnforcement;//true neu ap dung luat do xe
31  };
32  int main()
33  {
34      Holiday h(14, 2, true);
35      cout << "Kiem tra lop Holiday.\n";
36      h.output();
37      return 0;
38  }
39  Holiday::Holiday() : date(1, 1), parkingEnforcement(false)
40  {/*Phan than ham de trong*/}
41  Holiday::Holiday(int day, int month, bool theEnforcement)
42                   : date(day, month), parkingEnforcement(theEnforcement)
43  {/*Phan than ham de trong*/}
```

Biến thành viên của một kiểu lớp

Các lời gọi hàm tạo từ lớp DayOfYear

Ví dụ biến thành viên lớp

```
44  void Holiday::output()
45  {
46      date.output();
47      cout << endl;
48      if (parkingEnforcement)
49      cout << "Ap dung luat do xe.\n";
50      else
51      cout << "Khong ap dung luat do xe.\n";
52  }
53  DayOfYear::DayOfYear(int dayValue, int monthValue)
54      : day(dayValue), month(monthValue)
55  {
56      testDate();
57  }
58  //Xem cac dinh nghia ham con lai trong cac vi du truoc
```

Kết quả thực hiện:

```
Kiem tra lop Holiday.
14 Thang Hai
Ap dung luat do xe.
```

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

- Với kiểu dữ liệu lớn (chẳng hạn như 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 tham chiếu 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 nội 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 -> nội 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 hàm không truy cập tới dữ liệu của bất kỳ đối tượng nào
  - 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

```
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 << "Co bao nhieu nguoi trong nhom cua ban? ";
26          cin >> number;
27          cout << "Luot cua cac ban la: ";
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      cout << "Dung viec phuc vu.\n";
35      return 0;
36  }
```

Ví dụ thành viên tĩnh

```
37  Server::Server(char letterName) : name(letterName)
38  {/*Phan than ham de trong*/}
39  int Server::getTurn()
40  {
41      turn++;
42      return turn;
43  }
44  bool Server::stillOpen()
45  {
46      return nowOpen;
47  }
48  void Server::serveOne()
49  {
50      if (nowOpen && lastServed < turn)
51      {
52          lastServed++;
53          cout << "Quay " << name
54              << " dang phuc vu " << lastServed << endl;
55      }
56      if (lastServed >= turn) //Tat ca moi nguoi da duoc phuc vu
57          nowOpen = false;
58  }
```

Vì getTurn là tĩnh, chỉ các thành viên tĩnh mới có thể được tham chiếu ở đây

Ví dụ thành viên tĩnh

- Kết quả thực hiện:

```
Co bao nhieu nguoi trong nhom cua ban? 3
Luot cua cac ban la: 1 2 3
Quay A dang phuc vu 1
Quay B dang phuc vu 2
Co bao nhieu nguoi trong nhom cua ban? 2
Luot cua cac ban la: 4 5
Quay A dang phuc vu 3
Quay B dang phuc vu 4
Co bao nhieu nguoi trong nhom cua ban? 0
Luot cua cac ban la:
Quay A dang phuc vu 5
Dung viec phuc vu.
```

Tóm tắt

- Hàm tạo: tự động khởi tạo dữ liệu lớp
  - Được gọi khi khai 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 *nội 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