Bài thực hành 8

Phiên bản mới (09/04/2012)

- Đổi tên hàm getData trong lớp Date thành inputData
- Bổ sung vào lớp Date 3 hàm truy cập dữ liệu getDay, getMonth, getYear lần lượt trả về giá trị của các biến thành viên day, month, year

Mục tiêu

- Hướng dẫn cách viết và sử dụng hàm kiến tạo
- Từ khóa const, Hàm inline, Thành viên static
- Thư viện <vector>

Câu 1. [DateClass.cpp]

Trong bài này ta sẽ dùng lại lớp Date trong Bài thực hành 7.

DateClass.cpp

```
// DateClass.cpp
// Description: Chuong trinh cai dat lop Date bieu dien ngay
// ------------------------------------------------------------------------
#include <iostream>
#include <string>
using namespace std;

// Khai bao va khoi tao hang toan cuc MONTHS
// de luu ten tieng Anh cua cac thang
const string MONTHS[12] = {
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
};

// ------------------------------------------------------------------------
// Dinh nghia lop Date
class Date{
public:
    int getDay()const{return day;}
    int getMonth()const{return month;}
    int getYear()const{return year;}

    void inputData();// doc du lieu tu istream vao cac bien thanh vien
    void printShortForm();// in dang dd/mm/yyyy
    void printLongForm();// in xau tieng Anh day du
private:
```

```
    int day;
    int month;
    int year;
};

// ------------------------------------------------------------------------
// Dinh nghia ham main
int main(){
    Date d1;
    cout << "Khi chua khoi tao, gia tri cua d1: ";
    d1.printShortForm();
    cout << endl;

    cout << "Hay cap nhat d1: " << endl;
    d1.inputData();
    cout << "Gia tri moi cua d1: ";
    d1.printShortForm();
    cout << endl << "Xau tieng Anh cua d1: ";
    d1.printLongForm();

    cin.get();
    return 0;
}

// ------------------------------------------------------------------------
// Dinh nghia cac ham thanh vien cua Date

// doc du lieu tu istream vao cac bien thanh vien
void Date::inputData(){
    cout << "Nhap ngay: ";
    cin >> day;
    cout << "Nhap thang: ";
    cin >> month;
    cout << "Nhap nam: ";
    cin >> year;
    cin.ignore();// loai bo dau ENTER trong istream
}

// in dang dd/mm/yyyy
void Date::printShortForm(){
    cout << (day < 10 ? "0" : "") << day << "/"
        << (month < 10 ? "0" : "") << month << "/"
        << year;
}

// in xau tieng Anh day du
void Date::printLongForm(){
    cout << MONTHS[month - 1] << " " << day << ", " << year;
}
```

a) Copy chương trình trên và chạy thử. Bạn có thể thấy rằng khi chưa gọi tới hàm thành viên inputData, các biến thành viên sẽ được khởi tạo bởi những giá trị có sẵn trong bộ nhớ. Hãy viết hàm kiến tạo mặc định (không có tham số) khởi tạo các biến thành viên là thông số của ngày hiện tại. Chạy lại chương trình trên và quan sát kết quả.

(Tham khảo cách lấy ngày tháng hiện tại ở đây http://stackoverflow.com/questions/997946/c-get-current-time-and-date)

b) Viết hàm kiến tạo nhận 3 tham số int là 3 giá trị khởi tạo cho 3 biến thành viên . Chú ý sử dụng vùng khởi tạo (Tham khảo dòng 45, 46 của Display 7.1 Giáo trình). Bổ sung lời gọi tới hàm này trong main và chạy thử.

c) Nếu hàm thành viên không biến đổi các biến thành viên, bạn nên chỉ định rõ ràng điều đó trong khai báo hàm bằng cách bổ sung từ khóa const. Hãy thêm từ khóa const ở những nơi có thể trong lớp Date. (Tham khảo Display 7.4 Giáo trình).

d) Hãy tách phần giao diện, phần cài đặt và hàm main của chương trình DateClass thành 3 tệp Date.h, Date.cpp, main.cpp đưa chúng vào project dateapp1.

Câu 2. [BlogPost.h, BlogPost.cpp]

a) Hãy thêm vào project dateapp1 nói trên định nghĩa lớp BlogPost để biểu diễn thông tin một bài trên blog.

Các biến thành viên private của lớp này bao gồm:

- id có kiểu int (lưu mã của bài post, mã này dùng để phân biệt các bài post)
- creationDate có kiểu Date (lưu ngày viết bài post)
- title có kiểu string (lưu tiêu đề bài post)
- body có kiểu string (lưu nội dung bài post)

b) Hãy sử dụng một biến thành viên static, đặt tên là lastIssuedId để theo dõi id của đối tượng BlogPost mới nhất. Bạn cần kết hợp lastIssuedId, id và hàm kiến tạo sao cho id của các đối tượng BlogPost sẽ được tăng tự động, tức là: nếu hàm main bắt đầu với các lệnh khai báo

```
BlogPost abp, anotherbp;
```

thì abp sẽ có id bằng 1, anotherbp sẽ có id bằng 2.

(Tham khảo cách dùng từ khóa static ở Display 7.4 Giáo trình)

c) Hãy viết các hàm kiến tạo sau đây cho BlogPost:

- Hàm kiến tạo mặc định: đặt creationDate bằng ngày hiện tại, title và body rỗng
- Hàm kiến tạo với 3 tham số kiểu Date, string, string lần lượt là giá trị khởi tạo cho creationDate, title và body. Chú ý sử dụng vùng khởi tạo.

d) Viết các hàm thành viên truy cập sau dạng inline

- Hàm getId trả về id của đối tượng BlogPost
- Hàm getCreationDate trả về creationDate của đối tượng BlogPost
- Hàm getTitle trả về title của đối tượng BlogPost
- Hàm getBody trả về body của đối tượng BlogPost

Bổ sung từ khóa const ở những chỗ có thể.

e) Viết hàm thành viên `void update(string newTitle, string newBody);` để cập nhật đối tượng BlogPost với tiêu đề và nội dung mới.

f) Trong main.cpp, viết hàm void testBlogPost(); gọi tới tất cả các hàm của BlogPost. Gọi testBlogPost trong hàm main().

Câu 3. [main.cpp]

a) Trong hàm main, hãy khai báo

```
vector<BlogPost> bpvector;
```

để lưu một danh sách các BlogPost.

b) Hãy bổ sung một đoạn chương trình trong hàm main() lặp lại việc nhập thông tin cho từng BlogPost và thêm nó vào bpvector. Ở mỗi lần lặp, hỏi xem người dùng có muốn tiếp tục hay không.

Khi kết thúc lặp, hãy in ra id, title và creationDate của tất cả các phần tử trong bpvector (mỗi BlogPost trên 1 dòng, chú ý gióng cột).

(Tham khảo Display 7.7 Giáo trình)

c) Hãy bổ sung một đoạn chương trình trong hàm main() tìm trong bpvector những BlogPost viết vào ngày 13 và xóa chúng khỏi bpvector.

(Tham khảo cách xóa một phần tử trong vector ở đây: http://cplusplus.com/reference/stl/vector/erase/)

In lại các phần tử trong bpvector sau khi thực hiện phép xóa.