C++ Training Course

More about OOP

Lesson Objectives

- Understand about class declaration and definition
- Understand about hidden “this” pointer
- Understand about static member variable and function
- Understand about const class objects and member functions
- Understand about friend functions and classes

Section 1

Class Declaration and Definition

Class Declaration and Definition. Agenda

khai báo và định nghĩa lớp. Chương trình

- Declaration and implementation
- Why need separate declaration and implementation
- What should define in header file vs cpp file, and what inside class definition vs outside

tách biệt khai báo triển khai

Khai báo và triển khai
Tại sao cần phải tách biệt khai báo và triển khai
Cần định nghĩa gì trong file header và file cpp, và những gì trong định nghĩa lớp so với bên ngoài

Class Declaration and Definition

- Declaration of class
  - ✓ Declare class, declare member variables, prototypes of member functions Khai báo lớp, khai báo biến thành viên, nguyên mẫu các hàm thành viên
  - ✓ Put inside class definition (header file) Đặt trong định nghĩa lớp (tệp header)
- Implementation of class
  - ✓ Implement for member functions that declared in class (function body) Triển khai hàm thành viên: Triển khai cho các hàm thành viên được khai báo trong lớp (thân hàm)
  - ✓ Put outside class definition (cpp file) Đặt bên ngoài định nghĩa lớp (tệp cpp)

Class Declaration and Definition

Tại sao cần phải tách biệt khai báo và triển khai

- Why need separate declaration and implementation
  - ✓ As classes get longer and more complicated, having all the member function definitions inside the class can make the class harder to manage and work with Khi các lớp trở nên dài và phức tạp hơn, việc đặt tất cả các định nghĩa hàm thành viên bên trong lớp có thể làm cho lớp khó quản lý và làm việc.
  - ✓ Using an already-written class only requires understanding its public interface Sử dụng một lớp đã được viết chỉ đòi hỏi hiểu rõ giao diện công cộng của nó.
  - ✓ If change anything about the code in header file, it recompiles every file that includes that header. If change the code in cpp file, only that cpp file is recompiled Nếu thay đổi bất kỳ điều gì trong tệp header, sẽ cần phải biên dịch lại mọi tệp bao gồm tệp header đó. Nếu thay đổi mã trong tệp cpp, chỉ tệp cpp đó mới được biên dịch lại.
  - ✓ Very common for libraries Rất phổ biến cho các thư viện
    - Faster when build Nhanh hơn khi xây dựng
    - Share precompiled library for many apps Chia sẻ thư viện được biên dịch trước cho nhiều ứng dụng
    - Intellectual property reasons Lý do về sở hữu trí tuệ

Class Declaration and Definition

Định nghĩa trong tệp header và tệp cpp, cũng như bên trong và bên ngoài định nghĩa lớp

- What should define in header file vs cpp file, and what inside class definition vs outside
  - ✓ Class used in only one file, non-reusable: define directly in cpp file that it's used in
    không thể tái sử dụng
    Nếu lớp chỉ được sử dụng trong một tệp, không thể tái sử dụng: định nghĩa trực tiếp trong tệp cpp mà nó được sử dụng.
  - ✓ Class used in multiple files, reusable: define in header file that has same name as the class
    Nếu lớp được sử dụng trong nhiều tệp, có thể tái sử dụng: định nghĩa trong tệp header có cùng tên với tên của lớp.
  - ✓ Trivial member functions: define inside the class
    Các hàm thành viên đơn giản: định nghĩa bên trong lớp.
  - ✓ Non-trivial member functions: define in cpp file that has same name as the class
    Các hàm thành viên phức tạp: định nghĩa trong tệp cpp có cùng tên với tên của lớp.

Section 2

Hidden “this” pointer

this là một con trỏ ẩn đặc biệt được C++ tự động khởi tạo mỗi khi một phương thức của class được gọi.

Nó tồn tại trong mọi phương thức không tĩnh (non-static) của class và trỏ đến địa chỉ của đối tượng hiện tại mà phương thức đang thực thi từ đó.

this là hằng con trỏ, nghĩa là không thể gán lại giá trị cho nó để trỏ đến một đối tượng khác.

Hidden “this” pointer. Agenda

- Hidden “this” pointer mechanic
- Explicitly referencing “this”
- Chaining member functions

Hidden “this” pointer mechanic

```
class Simple
{
private:
   int m_id = 0;
public:
   void setID(int id) { m_id = id; }
};

int main()
{
   Simple simple;
   simple.setID(2);
   return 0;
}
```

Hidden “this” pointer mechanic

- When call simple.setID(2), compiler actually calls setID(&simple, 2) Khi gọi simple.setID(2), trình biên dịch thực ra gọi setID(&simple, 2).
- Inside setID(), the “this” pointer holds the address of object simple Bên trong hàm setID(), con trỏ "this" chứa địa chỉ của đối tượng simple
- Any member variables inside setID() are prefixed with “this->”. So when we say m_id = id, compiler is actually executing this->m_id = id, which in this case updates simple.m_id to id

tiền tố

thực hiện

Bất kỳ biến thành viên nào bên trong setID() đều có tiền tố là "this". Vì vậy khi chúng ta nói m_id = id, trình biên dịch thực ra thực hiện this->m_id = id, điều này trong trường hợp này cập nhật simple.m_id thành id.

Explicitly referencing “this”

- If member function has parameter with same name as member variable, you can disambiguate them by using this pointer

```
class Something
{
private:
   int data;
public:
   Something(int data)
   {
      this->data = data; // this->data is the member, data is the local parameter
   }
};
```

Khi một hàm thành viên có tham số có cùng tên với biến thành viên, bạn có thể làm rõ chúng bằng cách sử dụng con trỏ "this".

- **Recommend**: should use “m_” prefix for member variables, shouldn’t use explicitly referencing this for member variables and functions

Khuyên khích: nên sử dụng tiền tố "m_" cho các biến thành viên, không nên sử dụng việc tham chiếu rõ ràng đến "this" cho biến thành viên và các hàm thành viên.

Chaining member functions

- Can make function chainable by make each function return *this
- Most often used when overloading operators

```
class Calc
{
private:
   int m_value;
public:
   Calc() { m_value = 0; }
   Calc& add(int value) { m_value += value; return *this; }
   Calc& sub(int value) { m_value -= value; return *this; }
};

// use chaining member functions
Calc calc;
calc.add(5).sub(3);
```

Thường được sử dụng nhiều khi nạp chồng các toán tử.

chainebble: trả về giá trị của chính đối tượng đó sau khi thay đổi thông qua con trỏ this

Section 3

Static member variable and function

Static member variable and function. Agenda

- Static member variable vs member variable
- Static member function vs member function

Nói đến static function thì chắc chắn function đó là 1 phương thức của 1 class nào đó, không có hàm static thông thường

vị trí bộ nhớ của một biến static không thay đổi giữa các phân đoạn BSS (Block Started by Symbol) và Initialized Data Segment dựa trên trạng thái gán giá trị sau khi khai báo. Sự phân biệt giữa việc nằm trong phân đoạn BSS hay Initialized Data Segment được xác định vào thời điểm biên dịch, không phải khi chạy chương trình.

Nếu một biến static được khai báo mà không được khởi tạo rõ ràng, nó sẽ được đặt trong phân đoạn BSS. Phân đoạn này dành cho các biến static và toàn cục không được khởi tạo hoặc được khởi tạo giá trị là 0.
Nếu một biến static được khai báo và có khởi tạo rõ ràng là một giá trị khác không, nó sẽ được đặt trong Initialized Data Segment.
Sau khi được cấp phát, biến sẽ duy trì trong phân đoạn được chỉ định suốt thời gian thực thi chương trình. Gán một giá trị cho biến static sau này trong chương trình không làm thay đổi vị trí bộ nhớ của nó; nó chỉ thay đổi giá trị được lưu tại địa chỉ đó.

```
class MyClass {
private:
    static int staticVar; // Biến static
    int normalVar;        // Biến thành viên thông thường
public:
    MyClass(int normalValue) : normalVar(normalValue) {} // Constructor
    static void staticFunction(int value) {}
    void normalFunction() { }
};
int MyClass::staticVar = 0; // Khởi tạo biến static
int main() {
    MyClass::staticFunction(5);
    MyClass obj(3);
    obj.normalFunction();
    MyClass::staticFunction(10);
    return 0;
}
```

Static member variable vs member variable

Static member variable	Member variable
Belong to class Thuộc về lớp	Belong to a particular object Thuộc về một đối tượng cụ thể
Shared for all objects => same value for all objects Được chia sẻ cho tất cả các đối tượng => cùng một giá trị cho tất cả các đối tượng	Each of object obtains a copy of member variable => different value for each of object Mỗi đối tượng có một bản sao của biến thành viên => giá trị khác nhau cho mỗi đối tượng
Exist even if no objects of the class have been instantiated. Created when the program starts, and destroyed when the program ends	Created when the object is instantiated and destroyed when the object is destroyed
Tồn tại ngay cả khi không có đối tượng của lớp được khởi tạo. Được tạo ra khi chương trình bắt đầu.	Được tạo ra khi đối tượng được khởi tạo và bị hủy khi đối tượng bị hủy

Static member variable vs member function

Static member function	Member function
Belong to class, not attached to any particular object Thuộc về lớp, không liên kết với bất kỳ đối tượng cụ thể nào.	Attached to any particular object Được liên kết với một đối tượng cụ thể.
Can be called even if no objects of the class have been instantiated Có thể gọi ngay cả khi không có đối tượng của lớp được khởi tạo.	Only can be called with a particular object which is instantiated Chỉ có thể được gọi với một đối tượng cụ thểđã được khởi tạo.
Only can access to other static member variables and functions Chỉ có thể truy cập vào các biến và hàm thành viên static khác.	Can access to other static and normal member variables and functions Có thể truy cập các biến thành viên tĩnh và bình thường khác và các hàm thành viên.
No this pointer Không có con trỏ this.	Can use this pointer Có thể sử dụng con trỏ this.

Section 4

Const class objects and member functions

Const class objects and member functions

- Const class objects
- Const member functions

Const class objects

- After a const class object has been initialized, any attempt to modify the member variables of the object is disallowed Sau khi một đối tượng lớp const đã được khởi tạo, mọi cố gắng thay đổi các biến thành viên của đối tượng sẽ không được phép:
  - ✓ Cannot change member variable directly Không thể thay đổi trực tiếp các biến thành viên.
  - ✓ Cannot call member function to change member variable Không thể gọi các hàm thành viên không const để thay đổi các biến thành viên.
- Only call const member function

```
class Something
{
public:
    int m_value = 0;
    void setValue(int value) { m_value = value; }
};

const Something something; // calls default constructor
something.m_value = 5;       // compiler error: violates const
something.setValue(5);       // compiler error: violates const
```

Const member functions

- Const member function is a member function that guarantees it will not modify the object or call any non-const member functions

```
class Something
{
private:
    int m_value = 0;

public:
    // note addition of const keyword after parameter list, but before function body
    int getValue() const
    {
        return m_value;
    }
};
```

Hàm thành viên const là một hàm thành viên trong đó đảm bảo rằng nó sẽ không thay đổi đối tượng hoặc gọi bất kỳ hàm thành viên không const nào.

phân biệt const ở 2 vị trí

từ khóa const trong C++

từ khóa const được sử dụng để chỉ định rằng một biến, tham số, phương thức hoặc đối tượng không thể thay đổi sau khi được khởi tạo.

1. Biến hằng : const int x = 10; // Biến x là hằng và không thể thay đổi giá trị
2. Con trỏ và const :
   a, con trỏ tới hằng: const int *ptr1 = &x; // Giá trị mà ptr1 trỏ tới không thể thay đổi thông qua ptr1
   b, hằng con trỏ: int *const ptr2 = &x; // ptr2 là hằng con trỏ và không thể thay đổi địa chỉ
   c, const int *const ptr3 = &x; // ptr3 là hằng con trỏ tới hằng
3. Hàm const :
   a, hàm trả về hằng: const int getValue() {};
   b, tham số hằng: Tham số của hàm là hằng và không thể thay đổi bên trong hàm. void printValue(const int x);
   c, Hàm thành viên hằng: phương thức của một lớp không thể thay đổi bất kỳ đặc tính dữ liệu nào của đối tượng.

```
int getValue() const {
    return value; // Hàm này không thể thay đổi giá trị của value - do value là đặc tính của class
}
```

Section 5

Friend functions and classes

đối trọng của từ khóa const là từ khóa mutable: các đặc tính cảu class được gắn từ khóa này thì kể cả là trong trường hợp 3.c hàm thành viên hằng thì vẫn có thể thay đổi giá trị:

```
class MyClass {
private:
    int regularData;   // Thành viên dữ liệu bình thường
    mutable int mutableData;  // Thành viên dữ liệu mutable

public:
    MyClass(int reg, int mut) : regularData(reg), mutableData(mut) {}

    // Hàm thành viên const, nhưng có thể thay đổi thành viên dữ liệu mutable
    void display() const {
        // regularData = 10;  // Lỗi, không thể thay đổi thành viên dữ liệu bình thường trong hàm const
        mutableData = 20;  // Được phép, có thể thay đổi thành viên dữ liệu mutable trong hàm const
        std::cout << "Regular Data: " << regularData << std::endl;
        std::cout << "Mutable Data: " << mutableData << std::endl;
    }
};
```

Friend functions and classes. Agenda

- Friend functions
- Friend classes
- When should use

Friend functions

- Friend function is function can access to private member of other class

Hàm bạn là hàm có thể truy cập vào các thành viên private của một lớp khác.

```
class Accumulator
{
private:
   int m_value;
public:
   // Make the reset() function a friend of this class
   friend void reset(Accumulator &accumulator);
};

// reset() is now a friend of the Accumulator class
void reset(Accumulator &accumulator)
{
   // And can access the private data of Accumulator objects
   accumulator.m_value = 0;
}
```

Friend classes

- Friend class is class can access to private member of other class

```
class Storage
{
private:
    int m_nValue;
public:
    // Make the Display class a friend of Storage
    friend class Display;
};

class Display
{
public:
    void displayItem(const Storage &storage)
    {
        std::cout << storage.m_nValue << '\n';
    }
};
```

Friend class là một lớp có thể truy cập vào các thành viên private của một lớp khác.

When should use

- When two or more classes need to work very closely together (this class need access to private member of other class) but don’t want public member functions to access to private member

  Khi hai hoặc nhiều lớp cần làm việc rất gần nhau (lớp này cần truy cập vào thành viên private của lớp khác) nhưng không muốn các hàm thành viên public truy cập vào thành viên private.

- Private class data pattern

  Mẫu dữ liệu lớp private.

References

- http://www.learncpp.com/cpp-tutorial/89-class-code-and-header-files/
- http://www.learncpp.com/cpp-tutorial/8-8-the-hidden-this-pointer/
- http://www.learncpp.com/cpp-tutorial/811-static-member-variables/
- http://www.learncpp.com/cpp-tutorial/812-static-member-functions/
- https://www.learncpp.com/cpp-tutorial/810-const-class-objects-and-member-functions/
- http://www.learncpp.com/cpp-tutorial/813-friend-functions-and-classes/

Lesson Summary

- Class Declaration and Definition
- The hidden “this” pointer
- Static member variable and function
- Const class objects and member functions
- Friend functions and classes

Thank you