

COMPUTER SCIENCE PROGRAMMING C++

Lesson 20: OOP 2

Nguyễn Văn Hiếu



FullHouse

NỘI DUNG CHÍNH

- Ôn luyện kiến thức cũ
- Access modifier: private, protected, public và các tác dụng của từng keyword
- Tính trừu tượng, tính kế thừa trong C++



Nội dung chính

01

ACCESS MODIFIERS

private, protected, public

02

ABSTRACT

Tính trừu tượng của OOP trong C++

03

INHERITANCE

Tính kế thừa của OOP trong C++

04

LEARNING SKILLS

Viết được một chương trình cơ bản ứng dụng OOP



01

INTRODUCTION

Access modifier ?

Access modifier là từ khóa phạm vi truy cập để giới hạn khả năng truy cập data của một số thành phần



AccessModifier

Có 3 loại access modifier tồn tại trong C++:

- Private: riêng tư
- Protected: bảo vệ
- Public: công cộng



Tác dụng của AM

Một số ứng dụng thường thấy:

- Giới hạn truy cập tới các thành viên lớp được xác định bởi các khu vực public, private và protected được gán nhãn bên trong thân lớp.
- Bảo vệ dữ liệu nhạy cảm, quan trọng trong hệ thống chương trình.



Chương trình demo

Class : lớp

```
#include <iostream> // input - output - stream
using namespace std;
class Person
{
private:
    // Attributes
    string name;
    int age;

public:
    Person() {} // Default constructor
    Person(string name, int age) // Custom constructor
    {
        this->name = name;
        this->age = age;
    }
    ~Person() {} // Destructor
    // Methods
    voidNhapThongTin(){}
    voidInThongTin(){}
};
```

Attributes : thuộc tính

Constructor : hàm tạo

Methods : phương thức

Destructor : hàm hủy

Object : đối tượng

```
int main()
{
    Person p = Person("nam", 20);
    p.
    InThongTin() inline void Person::InThongTin()
    NhapThongTin()
    ~Person()
}
```



Chương trình demo

```
#include <iostream> // input - output - stream
using namespace std;
class Person
{
private:
    // Attributes
    string name;
    int age;
public:
    Person() {} // Default constructor
    Person(string name, int age) // Custom constructor
    {
        this->name = name;
        this->age = age;
    }
    ~Person() {} // Destructor
    // Methods
    void NhapThongTin(){}
    void InThongTin(){}
};

int main()
{
    Person p = Person("nam",20);
    p.
    re InThongTin inline void Person::InThongTin()
    NhapThongTin
    ~Person
4 UTF-8 C
```

private : riêng tư

public : công cộng



Tác dụng



PRIVATE

Đảm bảo dữ liệu chỉ được sử dụng trong nội bộ chính nó
Không leak ra ngoài



PROTECTED

Đảm bảo dữ liệu được sử dụng trong nội bộ và có thể kế thừa cho lớp con
Không leak ra ngoài



PUBLIC

Dữ liệu có thể được sử dụng, truy xuất ở mọi nơi



Trăm hay không bằng tay quen




FullHouse



02

Tính trừu tượng

Tính trừu tượng trong OOP C++



Abstract in OOP

Trừu tượng hóa (abstraction) liên quan tới việc chỉ cung cấp thông tin cần thiết tới bên ngoài và ẩn chi tiết cơ sở của chúng.

Ví dụ: để biểu diễn thông tin cần thiết trong chương trình mà không hiển thị chi tiết về chúng



Tác dụng của abstract

Phần nội vi hay bên trong lớp được bảo vệ tránh khỏi các lỗi do người dùng vô ý, mà có thể gây hư hỏng trạng thái của dữ liệu.

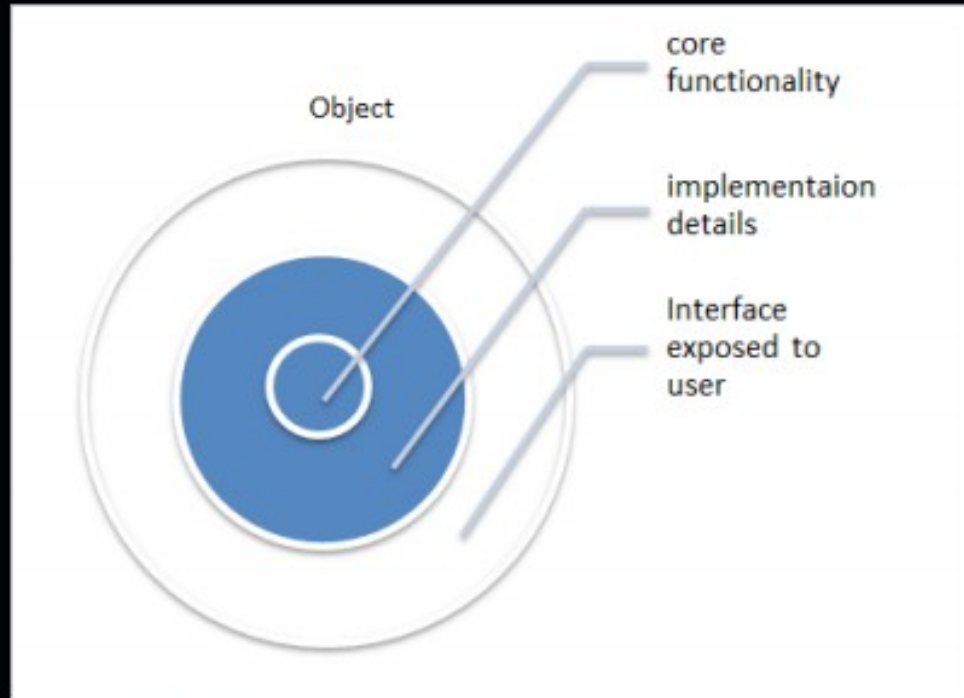
Triển khai lớp có thể tiến hành qua thời gian để đáp ứng yêu cầu thay đổi hoặc bug các báo cáo mà không yêu cầu thay đổi trong code của người dùng.



abstract

Lớp là sự trừu tượng hóa của đối tượng
Lớp được coi là bản thiết kế để tạo ra một đối tượng

Ẩn hết tất cả sự
phức tạp trong
nội bộ



Chỉ cho kết quả cần
thiết mà không quan
tâm sự hoạt động của
chương trình






03

Tính kế thừa

Tính kế thừa trong OOP C++



Inheritance in OOP

Kế thừa là sự liên quan giữa hai class với nhau, trong đó có class cơ sở (Base Class) và class con (Derived Class).

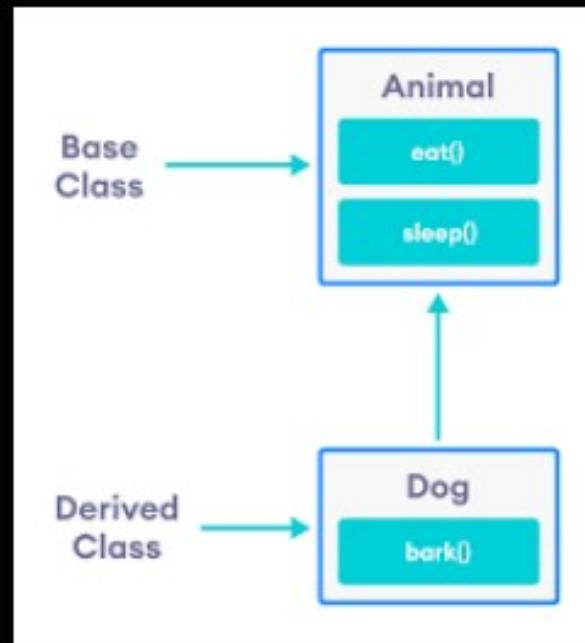
Khi kế thừa class con được hưởng tất cả các phương thức và thuộc tính của class cơ sở.



inheritance

Tư tưởng của kế thừa trong C++ là có thể tạo ra một class mới được xây dựng trên các lớp đang tồn tại.

Khi kế thừa từ một lớp đang tồn tại, có thể sử dụng lại các phương thức và thuộc tính của lớp cơ sở.



Đồng thời có thể khai báo thêm các phương thức và thuộc tính khác.



Lưu ý

Lớp con chỉ được truy cập các thành viên public và protected của class cơ sở.
Nó không được phép truy cập đến thành viên private của class cơ sở.



Một số bài tập ứng dụng





Thank

Cảm ơn tất cả các bạn đã theo dõi

