***Lập trình hướng đối tượng.***

1. ***Lập trình hướng đối tượng là gì?***

* Là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải và xây dựng chương trình
* Là phương pháp lập trình dựa trên lớp (class) và đối tượng (object).

1. ***Lớp đối tượng là gì?***

* Lớp là một tả trừ tượng của những đối tượng cùng bản chất.
* Ngược lại, một đối tượng là thể hiện cụ thể của những mô tả trừu tượng đó.
* Lớp đối tượng là một tập các đối tượng mà chúng có đặc tính tương tự nhau. Một lớp đối tượng gồm 2 đặc trưng:
* Thuộc tính: là một thuộc tính của đối tượng ,có giá trị nhất định cho mỗi đối tượng tại mỗi thời điểm trong hệ thống.
* Thao tác: là hành vi là đối tượng này tương tác với đối tượng khác hoặc với chính nó.

1. ***Đối tượng là gì?***

* Trong thực tế, đối tượng là một thực thể: người, sự vật hoặc bảng số liệu,...
* Đối tượng có 2 đặc trưng cơ bản:
* Thuộc tính
* Thao tác

1. ***Các đặc điểm của lập trình hướng đối tượng?***

* Tính đóng gói: Nhóm những gì có liên quan với nhau lại vào thành một, để khi dùng đến ta chỉ cần dùng một tên để gọi nó.

VD:

class PS{

private:

int iTuSo, iMauSo;

public:

int get\_Tu\_So(){

return iTuSo;}

void set\_Tu\_So(int tu){

iTuSo= tu;}

};

* Tính kế thừa:
* Là cơ chế cho phép lớp D có thuộc tính và thao tác của lớp C cứ như thể thuộc tính và thao tác này được định nghĩa trong lớp D.
* Là cơ chế cho phép cài đặt nhiều quan hệ giữa các đối tượng.

VD:

Con kế thừa từ Cha Tam giác cân kế thừa từ Tam giác

* Tính đa hình:
* Là cơ chế cho phép tên thuộc tính hoặc thao tác được định nghĩa tại nhiều lớp, có thể có nhiều cài đặt khác nhau tại mỗi lớp trong các lớp.
* Tính trừu tượng hóa:
* Cái nhìn khái quát hóa về một tập có các đối tượng có chung đặc điểm quan trọng và bỏ những chi tiết dư thừa.

VD:

Mèo kế thừa Động vật Chó kế thừa Động vật

1. ***Phạm vi truy xuất***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bên trong lớp | Bên ngoài lớp | Lớp bạn | Lớp con |
| private | 🞮 |  | 🞮 |  |
| Protected | 🞮 |  | 🞮 | 🞮 |
| public | 🞮 | 🞮 | 🞮 | 🞮 |

* Các thành phần được khai báo trong phạm vi private: chỉ có thể truy xuất bên trong lớp và lớp bạn.
* Các thành phần được khai báo trong phạm vi protected : chỉ có thể truy xuất bên trong lớp, lớp bạn và các lớp con kế thừa nó.
* Các thành phần được khai báo trong phạm vi public : có thể truy xuất bất kì đâu trong chương trình.

1. ***Contructor***

* Phương thức thiết lập là một loại phương thức đặc biệt được dùng để khai báo thể hiện của lớp.
* Bất kì đối tượng nào khi khai báo đều sẽ sư dụng phương thức thiết lập để khai báo các giá trị thành phần của đối tượng đó.
* Hàm thiết lặp sử dụng phương thức mà tên phương thức đó trùng với tên lớp và không có kiểu giá trị trả về( kể cả hàm void).
* Là thuộc tính của public.

1. ***Destructor***

* Phương thức hủy là một loại phương thức đặc biệt được dùng để dọn dẹp dữ liệu trước khi thu hồi bộ nhớ.
* Bất kì đối tượng nào khi sử dụng xong đều sẽ sử dụng phương thức hủy để xóa các giá trị thành phần của đối tượng đó.
* Hàm hủy sử dụng phương thức mà tên phương thức đó là “~” với tên lớp và không có kiểu giá trị trả về( kể cả hàm void).
* Là thuộc tính của public.

1. ***Phân loại Constructor***

|  |  |  |
| --- | --- | --- |
| Constructor mặc định | Constructor có tham số | Constructor sao chép |
| Là phương thức thiết lập khi sử dụng sẽ không có đối số nào được cung cấp. Và khi ta không tạo constructor cho lớp, máy tính sẽ tự gọi phương thức thiết lập mặc đình do chính nó tạo ra. | Là phương thức thiết lập có đối số đầu vào là tham số có kiểu dưc liệu khác với chính lớp đó.  Có thể có nhiều đối số và kiểu dữ liệu là khác nhau. | Là phương thức thiết lập có đối số l=đầu vào là tham chiếu đến dối tượng thuộc về chính lớp đó.  Tạo ra đối tượng mới giống một phần của đối tượng cũ. Không giống với phép gán thông thường. |
| PS::PS(){  iTuSo=0;  iMauSo=1;  } | PS::PS(int a, int b){  iTuSo=a;  iMauSo=b;  }  PS::PS(int a){  iTuSo=a;  iMauSo=1;  } | PS::PS(const &PS a){  iTuSo=a.iTuSo;  iMauSo=a.iMauSo;  } |

Thường trong khi viết chuong trình ta thường hay viết như sau để đầy đủ hơn:

PS::PS(int a=0, int b=1){

iTuSo=a;

iMauSo=b;

}

1. ***Phân biệt Constructor và Destructor***

|  |  |
| --- | --- |
| Constructor | Destructor |
| Dùng để khởi tạo đối tượng của lớp | Dùng để dọn dẹp đối tượng trước khi thu hồi bộ nhớ |
| Không có kiểu trả về | Không có kiểu trả về |
| Là thuộc tính của public | Là thuộc tính của public |
| Có thể có hoặc không có tham số | Không có tham số |
| Có 3 kiểu constructor | Có duy nhất 1 destructor |
| “Tên class” | “~” + “tên class” |

1. ***Hàm bạn và lớp bạn***

* Hàm bạn: là không thuộc lớp nhưng vẫn có thể truy vấn dẫn xuất private và protected của lớp mà nó là bạn.

VD:

class A{

private:

int val;

public:

friend int get\_Val(A);

};

int get\_Val(A a) {

return a. val;

}

* Lớp bạn: là lớp mà ta có thể truy vấn dẫn xuất private và protected của lớp mà nó là bạn.

VD:

class A{

private:

int val;

public:

friend class B;

};

class B{

public:

int get\_val(A a){

return a.val;}

};

1. ***Override và Overload***

* Override: là tính năng cho phép lớp con cung cấp triển khai cụ thể hơn cho phương thức đã được cung cấp ở lớp cha .

VD:

class A{

private:

int val;

public:

virtual void Nhap() = 0;

};

class B: public class A{

public:

void Nhap() {

cin>>val;}

};

* Overload có hai dạng :
* Overload function: là ta dùng một tên gọi để gọi tên các hàm có cùng chức năng( mục đích) . Nhưng khác nhau bởi kiểu dữ liệu vầ số lượng đối số đầu vào.

VD:

class A{

private:

int val;

public:

void Xuat(int a){

cout<< a;}

void Xuat(int a, double b){

cout<< a<<b;}

};

* Overload operator: là ta định nghĩa các phép toán tử trên kiểu tự định nghĩa.

VD:

int main(){

PS a,b;

cout<<a+b; // (a.cong(b)).xuat();

return 0;}

1. ***Kế thừa.***

* Kế thừa là một đặc điểm của ngôn ngữ dùng để biểu diễn mối quan hệ đặc biệt hóa và tổng quát hóa giữa các lớp .Các lớp trừu tượng hóa và đucợ tổ chức dưới dạng là một sơ đồ phân cấp lớp.
* Lợi ích:
* Cho phép xây dựng lớp mới từ lớp cũ.
* Cho phép các lớp chia sẻ mã chương trình chung nhờ đó có thể dễ dàng sửa chữa và nâng cấp hệ thống .Ngoài ra trong c++, kế thừa còn được định nghĩa là sự tương thích nên ta có cơ chế chuyển kiểu tự động.

VD:

Con kế thừa từ Cha Tam giác cân kế thừa từ Tam giác

1. ***Phạm vi truy xuất trong kế thừa***

|  |  |  |  |
| --- | --- | --- | --- |
|  | private | protected | public |
| private |  |  |  |
| protected | private | protected | protected |
| public | private | protected | public |

* Nếu phương thức kế thừa là Public:
* Các thành phần có phạm vi truy xuất là Public của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Public.
* Các thành phần có phạm vi truy xuất là Protected của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Protected.
* Các thành phần có phạm vi truy xuất là Private của lớp cha thì không truyền xuống lớp con (lớp con không có quyền kế thừa lại các thành phần có phạm vi truy xuất là Private của lớp cha).
* Nếu phương thức kế thừa là Protected:
* Các thành phần có phạm vi truy xuất là Public của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Protected.
* Các thành phần có phạm vi truy xuất là Protected của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Protected.
* Các thành phần có phạm vi truy xuất là Private của lớp cha thì không truyền xuống lớp con (lớp con không có quyền kế thừa lại các thành phần có phạm vi truy xuất là Private của lớp cha).
* Nếu phương thức kế thừa là Private:
* Các thành phần có phạm vi truy xuất là Public của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Private.
* Các thành phần có phạm vi truy xuất là Protected của lớp cha khi xuống lớp con cũng có phạm vi truy xuất là Private.
* Các thành phần có phạm vi truy xuất là Private của lớp cha thì không truyền xuống lớp con (lớp con không có quyền kế thừa lại các thành phần có phạm vi truy xuất là Private của lớp cha).

1. ***Đa hình***

* Là hiện tượng của các đối tượng trong các lớp khác nhau nhưng có khả năng hiểu cùng một thông điệp bằng nhiều cách khác nhau.

VD:

Chuyển động (đa hình)

* Người: đi
* Chim: bay
* Cá : bơi

1. ***Lớp cơ sở trừu tượng và phương thức thuần ảo***

* Lớp cơ sở trừu tượng:
* Là lớp cơ sở không có đối tượng nào thuộc về
* Có ít nhất một hàm thuẩn ảo được khai báo trong nó.
* Tập hợp nhiều hàm thuẩn ảo trong lớp trừu tượng được gọi là interface.
* Phương thức thuần ảo:
* Là phương thức ảo, không có thâ nhàm( không có nội dung) và kiểu giá trị trả là 0.

VD:

class A{

private:

int val;

public:

virtual void Nhap() = 0;

};

1. ***Truy vấn dẫn xuất và cập nhật***

|  |  |
| --- | --- |
| Get – truy vấn dẫn xuất | Set – cập nhật |
| Trả về giá trị của thuộc tính thuộc đối tượng trong lớp | Gán giá trị của thuộc tính thuộc đối tượng trong lớp |
| int get(){  return iTuSo;} | void set(int tu){  iTuSo = tu;} |