

# KẾ THỪA (TT)

C++



Microsoft®

Visual Studio®

# PHẠM VI TRUY XUẤT

- ❖ KHI THIẾT LẬP QUAN HỆ KẾ THỪA, TA VẪN PHẢI QUAN TÂM ĐẾN **TÍNH ĐÓNG GÓI VÀ CHE DẤU THÔNG TIN**.
- ❖ ĐIỀU NÀY ẢNH HƯỞNG ĐẾN PHẠM VI TRUY XUẤT CỦA CÁC THÀNH PHẦN CỦA LỚP.
- ❖ HAI VẤN ĐỀ ĐƯỢC ĐẶT RA LÀ:
  - **TRUY XUẤT THEO CHIỀU DỌC**
  - **TRUY XUẤT THEO CHIỀU NGANG**

# PHẠM VI TRUY XUẤT

## ❖ TRUY XUẤT THEO CHIỀU DỌC:

- HÀM THÀNH PHẦN CỦA LỚP CON CÓ QUYỀN TRUY XUẤT CÁC THÀNH PHẦN CỦA LỚP CHA HAY KHÔNG?

## ❖ TRUY XUẤT THEO CHIỀU NGANG:

- CÁC THÀNH PHẦN CỦA LỚP CHA, SAU KHI KẾ THỪA XUỐNG LỚP CON, THÌ THẾ GIỚI BÊN NGOÀI CÓ QUYỀN TRUY XUẤT THÔNG QUA ĐỐI TƯỢNG CỦA LỚP CON HAY KHÔNG?

# TRUY XUẤT THEO CHIỀU DỌC

- ❖ LỚP CON CÓ QUYỀN TRUY XUẤT CÁC THÀNH PHẦN CỦA LỚP CHA HAY KHÔNG, HOÀN TOÀN DO LỚP CHA QUYẾT ĐỊNH. ĐIỀU ĐÓ ĐƯỢC XÁC ĐỊNH BẰNG THUỘC TÍNH KẾ THỪA.
- ❖ TRONG TRƯỜNG HỢP LỚP SINH VIÊN KẾ THỪA LỚP NGƯỜI, SINH VIÊN CÓ QUYỀN TRUY XUẤT HỌ TÊN CỦA CHÍNH MÌNH (ĐƯỢC KHAI BÁO Ở LỚP NGƯỜI) HAY KHÔNG?

# PHẠM VI TRUY XUẤT

## ❖ THUỘC TÍNH PUBLIC:

- THÀNH PHẦN NÀO CÓ THUỘC TÍNH PUBLIC THÌ CÓ THỂ TRUY XUẤT TỪ BẤT CỨ NƠI NÀO.

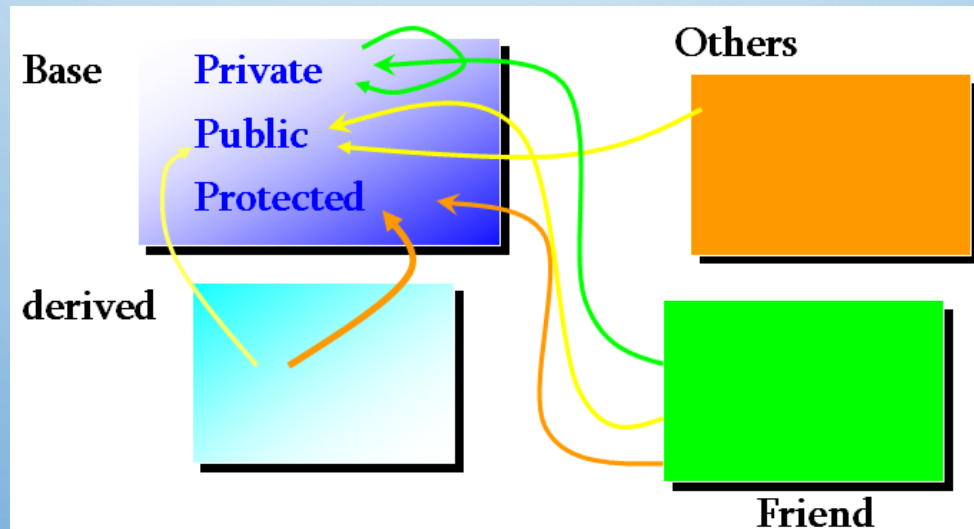
## ❖ THUỘC TÍNH PRIVATE: THÀNH PHẦN CÓ THUỘC TÍNH PRIVATE

- LÀ RIÊNG TƯ CỦA LỚP ĐÓ
- CHỈ CÓ HÀM THÀNH PHẦN CỦA LỚP VÀ NGOẠI LỆ CÁC HÀM BẠN ĐƯỢC PHÉP TRUY XUẤT.
- CÁC LỚP CON CŨNG KHÔNG CÓ QUYỀN TRUY XUẤT

# PHẠM VI TRUY XUẤT

## ❖ THUỘC TÍNH PROTECTED:

- CHO PHÉP QUI ĐỊNH MỘT VÀI THÀNH PHẦN NÀO ĐÓ CỦA LỚP LÀ **BẢO MẬT**, THEO NGHĨA THỂ GIỚI BÊN NGOÀI KHÔNG ĐƯỢC PHÉP TRUY XUẤT, NHƯNG TẤT CẢ CÁC LỚP CON, CHÁU... ĐỀU ĐƯỢC PHÉP TRUY XUẤT.



# VÍ DỤ THUỘC TÍNH PRIVATE

```
class Nguoi {  
    char *HoTen;  
    int NamSinh;  
public:  
    //...  
};  
class SinhVien : public Nguoi {  
    char *MaSo;  
public:  
    //...  
    void Xuat() const;  
};
```



# THUỘC TÍNH PRIVATE

- ❖ TRONG VÍ DỤ TRÊN, KHÔNG CÓ HÀM THÀNH PHẦN NÀO CỦA LỚP SINHVIEN CÓ THỂ TRUY XUẤT CÁC THÀNH PHẦN **HOTEN**, **NAMSINH** CỦA LỚP NGUOI.
- ❖ VÍ DỤ, ĐOẠN CHƯƠNG TRÌNH SAU ĐÂY SẼ GÂY RA LỖI:

```
VOID SINHVIEN::XUAT() CONST {  
    COUT << "SINH VIEN, MA SO: "<<MASO<<","HO  
    TEN:"<<HOTEN;  
}
```



# THUỘC TÍNH PRIVATE

❖TA CÓ THỂ KHẮC PHỤC LỖI TRÊN NHỜ KHAI BÁO LỚP SINHVIEN LÀ BẠN CỦA LỚP NGUOI NHƯ TRONG VÍ DỤ BAN ĐẦU:

```
class Nguoi {  
    friend class SinhVien;  
  
    char *HoTen;  
    int NamSinh;  
  
public:  
    //...  
};
```

# THUỘC TÍNH PRIVATE

- ❖ KHAI BÁO LỚP BẠN NHƯ TRÊN, LỚP SINHVIEN CÓ THỂ TRUY XUẤT CÁC THÀNH PHẦN PRIVATE CỦA LỚP NGUOI.
- ❖ CÁCH LÀM TRÊN CHỈ GIẢI QUYẾT ĐƯỢC NHU CẦU CỦA NGƯỜI SỬ DỤNG KHI MUỐN TẠO LỚP CON CÓ QUYỀN TRUY XUẤT CÁC THÀNH PHẦN DỮ LIỆU PRIVATE CỦA LỚP CHA.
- ❖ TUY NHIÊN, CẦN PHẢI SỬA LẠI LỚP CHA VÀ TẤT CẢ CÁC LỚP Ở CẤP CAO HƠN MỖI KHI CÓ MỘT LỚP CON MỚI.

# THUỘC TÍNH PROTECTED

❖ TRONG VÍ DỤ TRƯỚC, KHI CÀI ĐẶT LỚP NUSINH TA PHẢI THAY ĐỔI LỚP CHA SINHVIEN VÀ CẢ LỚP CƠ SỞ NGUOI Ở MỨC CAO HƠN.

# THUỘC TÍNH PROTECTED

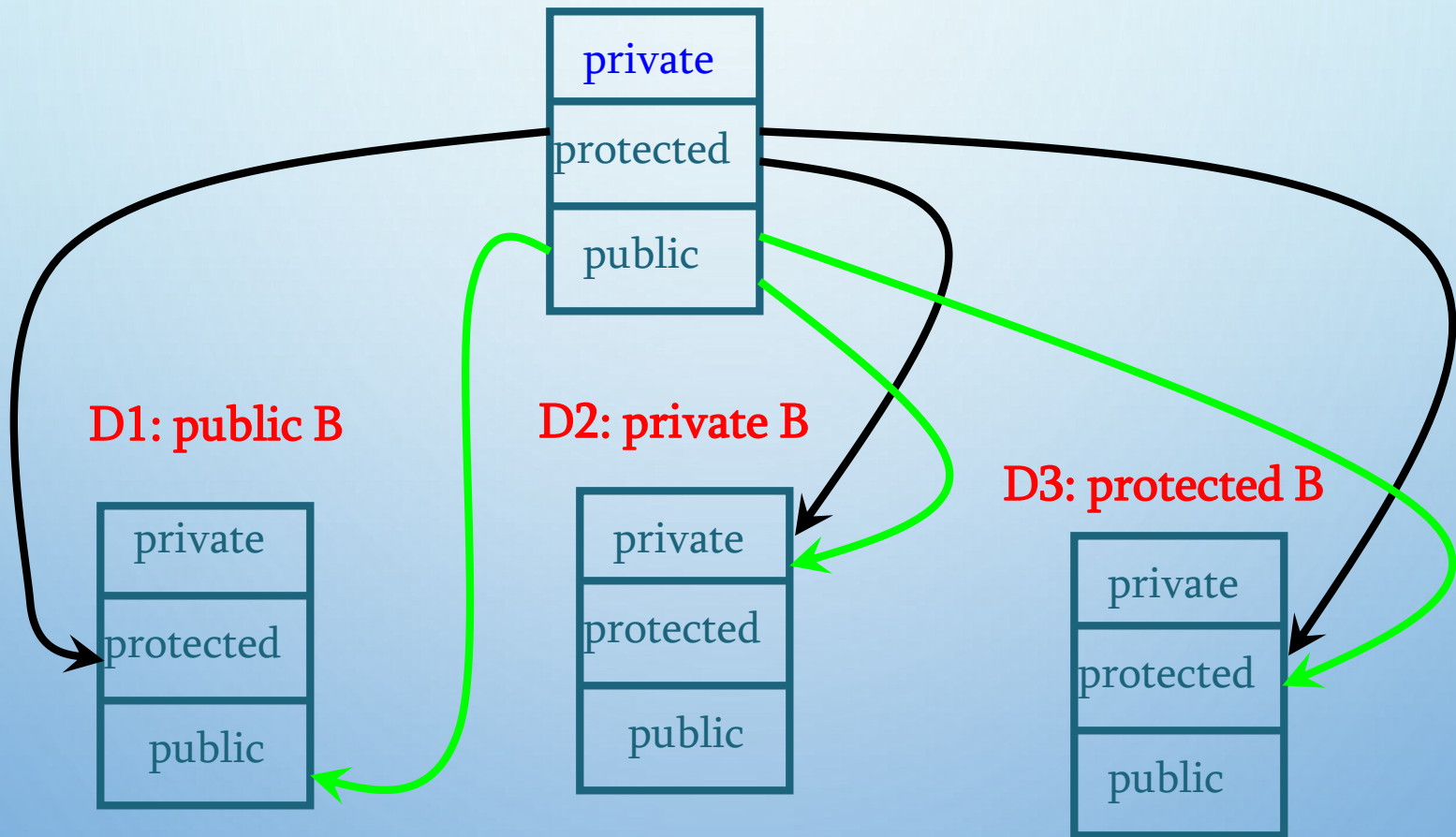
- ❖ LÀ CÁCH ĐỂ **TRÁNH PHẢI SỬA ĐỔI LỚP CƠ SỞ** KHI CÓ LỚP CON MỚI HÌNH THÀNH → ĐẢM BẢO TÍNH ĐÓNG GÓI.
- ❖ THÔNG THƯỜNG TA DÙNG THUỘC TÍNH **PROTECTED** CHO **THÀNH PHẦN DỮ LIỆU** VÀ **PUBLIC** CHO **THÀNH PHẦN PHƯƠNG THỨC**.
- ❖ TÓM TẮT, THÀNH PHẦN CÓ THUỘC TÍNH **PROTECTED** CHỈ CHO PHÉP NHỮNG LỚP CON KẾ THỪA ĐƯỢC PHÉP SỬ DỤNG.

# TRUY XUẤT THEO CHIỀU NGANG

❖ THÀNH PHẦN **PROTECTED VÀ PUBLIC** CỦA LỚP KHI ĐÃ KẾ THỪA XUỐNG LỚP CON THÌ THỂ GIỚI **BÊN NGOÀI CÓ QUYỀN TRUY XUẤT THÔNG QUA ĐỐI TƯỢNG THUỘC LỚP CON HAY KHÔNG?**

- ĐIỀU NÀY HOÀN TOÀN DO LỚP CON QUYẾT ĐỊNH BẰNG **PHẠM VI KẾ THỪA: KẾ THỪA PUBLIC, KẾ THỪA PROTECTED, KẾ THỪA PRIVATE**

# PHẠM VI TRUY XUẤT TRONG KẾ THỪA



# PHẠM VI TRUY XUẤT TRONG KẾ THỪA

Type of Inheritance

Access Control for Members

	private	Protected	public
private	?	?	?
protected	?	?	?
public	?	?	?



# PHẠM VI TRUY XUẤT TRONG KẾ THỪA

Base class member access specifier	Type of inheritance		
	public inheritance	protected inheritance	private inheritance
<b>Public</b>	<b>public</b> in derived class. Can be accessed directly by any non- <b>static</b> member functions, <b>friend</b> functions and non-member functions.	<b>protected</b> in derived class. Can be accessed directly by all non- <b>static</b> member functions and <b>friend</b> functions.	<b>private</b> in derived class. Can be accessed directly by all non- <b>static</b> member functions and <b>friend</b> functions.
<b>Protected</b>	<b>protected</b> in derived class. Can be accessed directly by all non- <b>static</b> member functions and <b>friend</b> functions.	<b>protected</b> in derived class. Can be accessed directly by all non- <b>static</b> member functions and <b>friend</b> functions.	<b>private</b> in derived class. Can be accessed directly by all non- <b>static</b> member functions and <b>friend</b> functions.
<b>Private</b>	Hidden in derived class. Can be accessed by non- <b>static</b> member functions and <b>friend</b> functions through <b>public</b> or <b>protected</b> member functions of the base class.	Hidden in derived class. Can be accessed by non- <b>static</b> member functions and <b>friend</b> functions through <b>public</b> or <b>protected</b> member functions of the base class.	Hidden in derived class. Can be accessed by non- <b>static</b> member functions and <b>friend</b> functions through <b>public</b> or <b>protected</b> member functions of the base class.

# PHƯƠNG THỨC THIẾT LẬP

- ❖ PHƯƠNG THỨC THIẾT LẬP CỦA LỚP CƠ SỞ **LUÔN LUÔN ĐƯỢC GỌI** MỖI KHI CÓ MỘT ĐỐI TƯỢNG CỦA LỚP DẪN XUẤT ĐƯỢC TẠO RA.
- ❖ NẾU MỌI PHƯƠNG THỨC THIẾT LẬP CỦA LỚP CƠ SỞ ĐỀU ĐÒI HỎI PHẢI CUNG CẤP THAM SỐ THÌ LỚP CON BẮT BUỘC PHẢI CÓ PHƯƠNG THỨC THIẾT LẬP ĐỂ CUNG CẤP CÁC THAM SỐ ĐÓ

# ĐỊNH NGHĨA CÁC THÀNH PHẦN RIÊNG

❖NGOÀI CÁC THÀNH PHẦN ĐƯỢC KẾ THỪA, LỚP DẪN XUẤT CÓ THỂ ĐỊNH NGHĨA THÊM


```
class HìnhTron : Diem {  
    double r;  
public:  
    HìnhTron( double tx, double ty, double rr) : Diem(tx, ty){  
        r = rr;  
    }  
    void Ve(int color) const;  
    void TinhTien( double dx, double dy) const;  
};  
HìnhTron t(200,200,50);
```

# ĐỊNH NGHĨA CÁC THÀNH PHẦN RIÊNG

❖ LỚP DẪN XUẤT CŨNG CÓ THỂ **OVERRIDE** CÁC PHƯƠNG THỨC ĐÃ ĐƯỢC ĐỊNH NGHĨA Ở TRONG LỚP CHA.

```
class A {  
    protected:  
        int x, y;  
    public:  
        void print () {  
            cout<<"From A"<<endl;  
        }  
};
```

```
class B : public A  
{  
    public:  
        void print () {  
            cout<<"From B"<<endl;  
        }  
};
```



# PHƯƠNG THỨC HỦY BỎ

- ❖ KHI MỘT ĐỐI TƯỢNG BỊ HỦY ĐI, PHƯƠNG THỨC HỦY BỎ CỦA NÓ SẼ ĐƯỢC GỌI. SAU ĐÓ, CÁC PHƯƠNG THỨC HỦY BỎ CỦA LỚP CƠ SỞ SẼ ĐƯỢC GỌI MỘT CÁCH TỰ ĐỘNG.
- ❖ VÌ VẬY, LỚP CON KHÔNG CẦN VÀ CŨNG KHÔNG ĐƯỢC THỰC HIỆN CÁC THAO TÁC DỌN DẸP CHO CÁC THÀNH PHẦN THUỘC LỚP CHA.

# CON TRỎ VÀ KẼ THÙA

## ❖ CON TRỎ TRONG KẼ THÙA HOẠT ĐỘNG THEO NGUYÊN TẮC SAU:

- CON TRỎ TRỎ ĐẾN ĐỐI TƯỢNG THUỘC LỚP CƠ SỞ THÌ CÓ THỂ TRỎ ĐẾN CÁC ĐỐI TƯỢNG THUỘC LỚP CON.
- NHƯNG CON TRỎ TRỎ ĐẾN ĐỐI TƯỢNG THUỘC LỚP CON THÌ KHÔNG THỂ TRỎ ĐẾN CÁC ĐỐI TƯỢNG THUỘC LỚP CƠ SỞ.
- CÓ THỂ ÉP KIỂU ĐỂ CON TRỎ TRỎ ĐẾN ĐỐI TƯỢNG THUỘC LỚP CON CÓ THỂ TRỎ ĐẾN ĐỐI TƯỢNG THUỘC LỚP CƠ SỞ. TUY NHIÊN THAO TÁC NÀY CÓ THỂ NGUY HIỂM.

# ĐA KẾ THỪA

❖ ĐA KẾ THỪA CHO PHÉP **MỘT LỚP CÓ THỂ LÀ DẪN XUẤT CỦA NHIỀU LỚP CƠ SỞ.**

```
CLASS A : PUBLIC B, PUBLIC C {
```

```
...
```

```
};
```

❖ CÁC ĐẶC ĐIỂM CỦA KẾ THỪA ĐƠN VẪN ĐÚNG CHO TRƯỜNG HỢP ĐA KẾ THỪA.



# ĐA KẾ THỪA

- ❖ LÀM THẾ NÀO BIỂU THỊ **TÍNH ĐỘC LẬP** CỦA **CÁC THÀNH PHẦN CÙNG TÊN** BÊN TRONG MỘT LỚP DẪN XUẤT?
- ❖ CÁC PHƯƠNG THỨC THIẾT LẬP VÀ HỦY BỎ ĐƯỢC GỌI NHƯ THẾ NÀO: THỨ TỰ, TRUYỀN THÔNG TIN, ...?
- ❖ LÀM THẾ NÀO GIẢI QUYẾT TÌNH TRẠNG **THỪA KẾ XUNG ĐỘT** TRONG ĐÓ, LỚP D DẪN XUẤT TỪ B VÀ C, VÀ CẢ HAI CÙNG LÀ DẪN XUẤT CỦA A