# 基类与派生类对象之间的赋值兼容关系

小组成员：
邵东林 :202307885
谭棵 :202306630
苏怡力 :202305962

# 目录

Contents

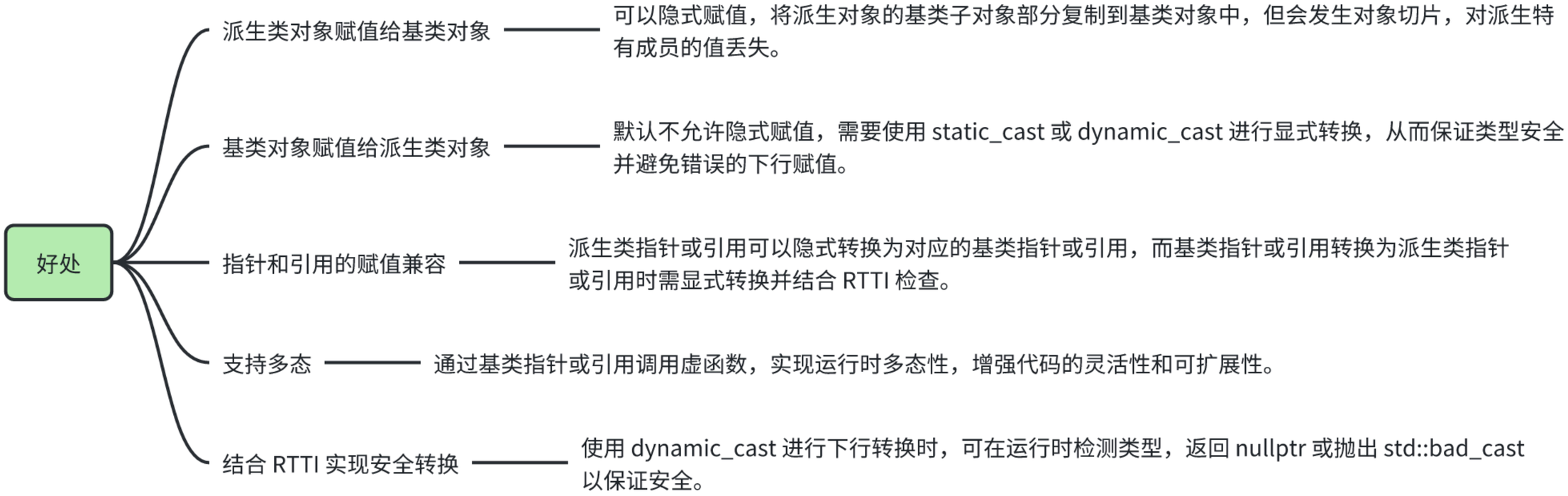
# 思想与好处

Background significance of the topic

01.

# 思想与好处

RTTI 是 “Run-Time Type Information”（运行时类型信息）的缩写。



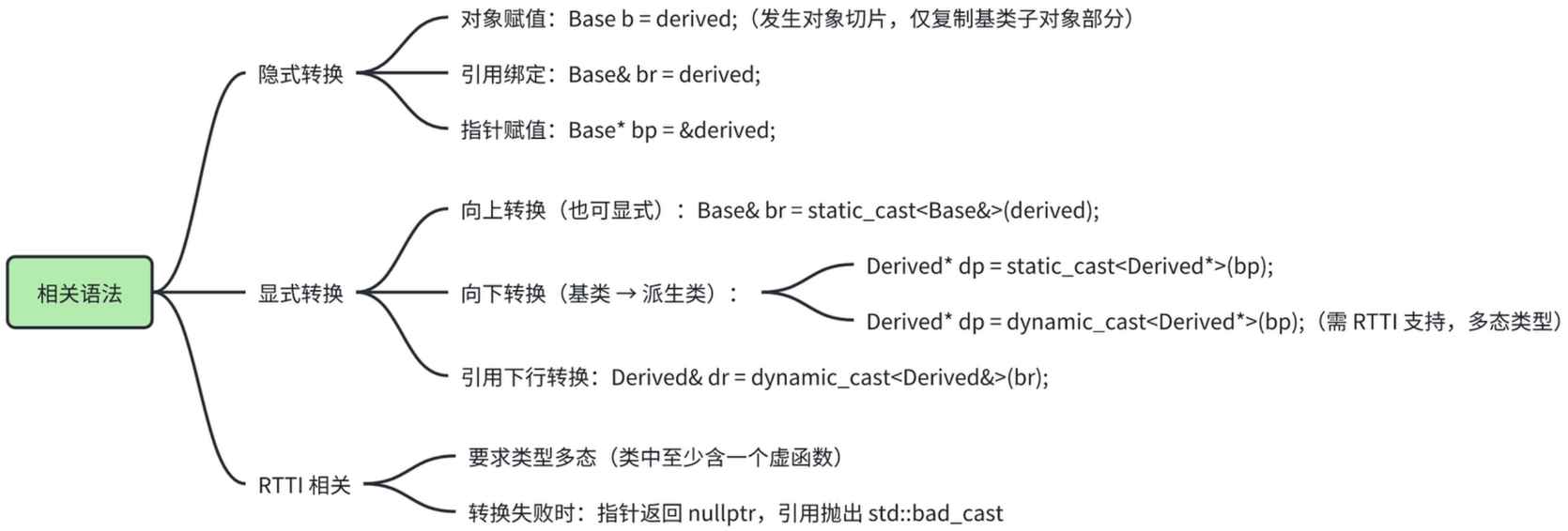

**所谓赋值兼容规则，即在任何需要基类对象的地方都可以用该基类的公有派生类的对象来代替**

# 相关语法

Relevant Syntax

02.

# 1.总览




根据赋值兼容规则，在基类Base的对象可以使用的任何地方，都可以用派生类Derived的对象来代替，但只能使用从基类继承来的成员。具体表现在以下几个方面。

1）派生类对象可以向基类对象赋值，即用派生类对象中从基类继承下来的数据成员，逐个赋值给基类对象的数据成员。

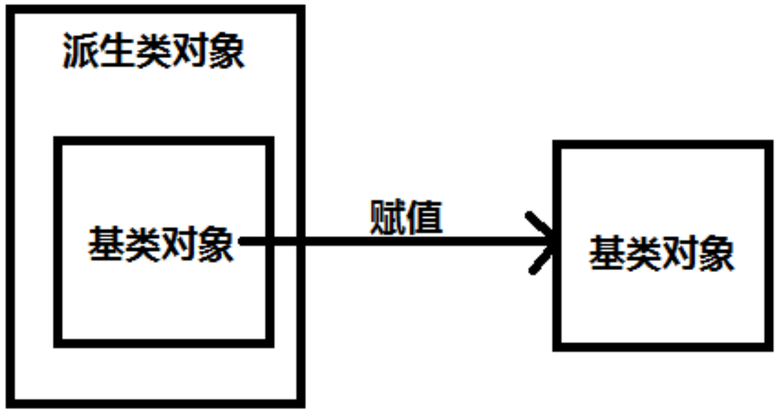
2）派生类对象可以初始化基类对象的引用。

3）派生类对象的地址可以赋给指向基类对象的指针。

4）如果函数的形参是基类对象或基类对象的引用，在调用函数时可以用派生类对象作为实参。

# 1）派生类对象可以向基类对象赋值

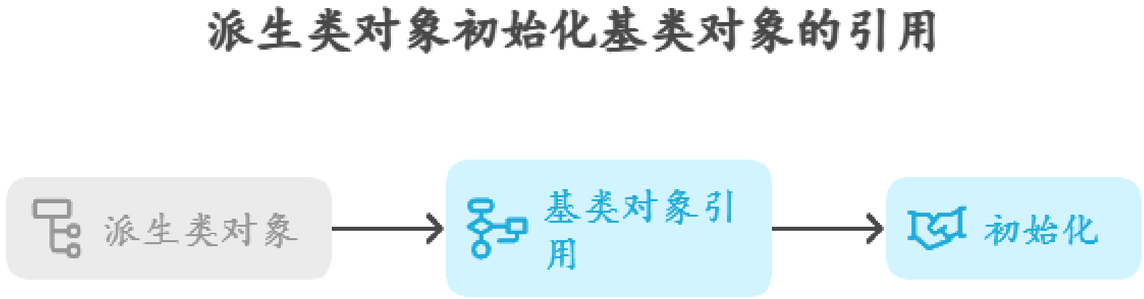
```
Base b;          //定义基类Base的对象b
Derived d;       //定义基类Base的公有派生类对象d
b=d;             //用派生类对象d对基类对象b赋值
```




**解释：**

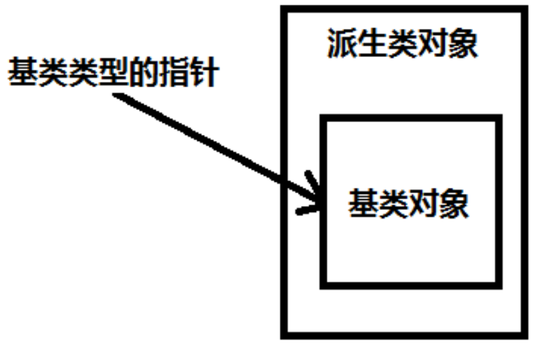
**这样赋值的效果是，对象b中所有数据成员都将具有对象d中对应数据成员的值**

# 2）派生类对象可以初始化基类对象的引用




```
Base b;                 //定义基类Base的对象b
Derived d;              //定义基类Base的公有派生类对象d
Base &br=d;             //定义基类Base的对象引用br，
                        //并用派生类对象d对其初始化
```

# 3）派生类对象的地址可以赋给指向基类对象的指针

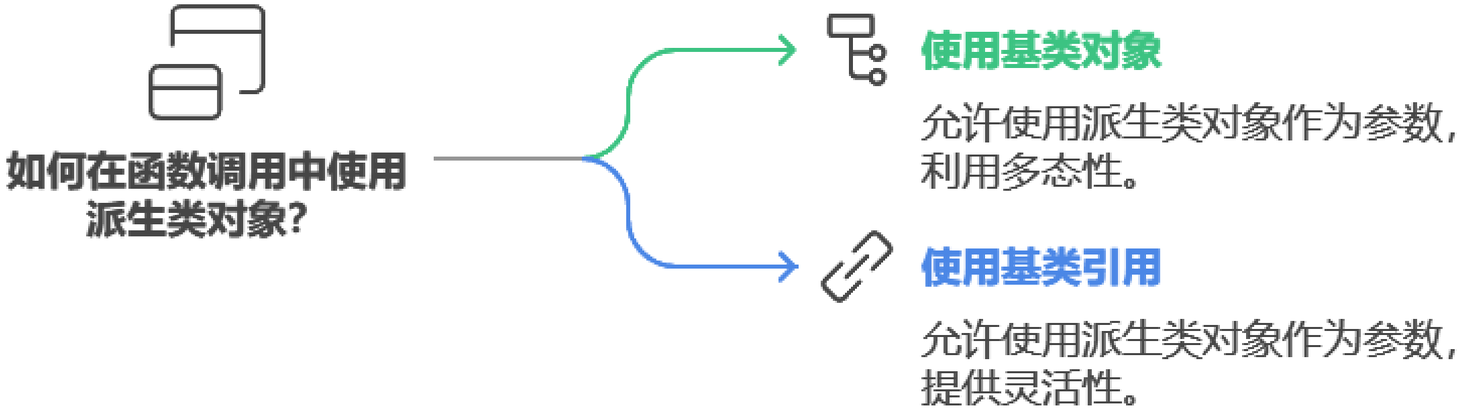



```
Derived d;          //定义基类Base的公有派生类对象d
Base *bp=&d;        //把派生类对象的地址&d赋值给指向基类指针bp
```

**解释：**

**这种形式的转换，是在实际应用程序中最常见到**

# 4）如果函数的形参是基类对象或基类对象的引用，在调用函数时可以用派生类对象作为实参






```
class Base{                          //声明基类Base
public c;
 int i;
....
};
class Derived:public Base{           //声明基类Base的公有派生类Derived
....
};
void fun(Base &bb)                   //普通函数，形参为基类Base对象的引用
{  cout<<bb.i<<endl;                 //输出该引用所代表的对象的数据成员i
}
fun(d4);
```

**解释：**

**在调用函数fun时可以用派生类Derived的对象d4作为实参**

# 示例分析

Example Analysis

03.

# 示例分析

```
#include <iostream>
#include <string>
using namespace std;

class Computer
{ // 父类
public:
    Computer(string cpu, string gpu, double ram, double storage) : cpu(cpu), gpu(gpu), ram(ram), storage(storage) {}
    void show()
    {
        cout << "Computer: cpu=" << cpu << ",gpu=" << gpu << ",ram=" << ram << ",storage=" << storage << endl;
    }

protected:  // 让子类可以访问
    string cpu;
    string gpu;
    double ram;
    double storage;
};

class Laptop : public Computer
{ // 子类 - 笔记本
public:
    Laptop(string cpu, string gpu, double ram, double storage) : Computer(cpu, gpu, ram, storage) {}
    void show()
    {
        cout << "Laptop: cpu=" << cpu << ",gpu=" << gpu << ",ram=" << ram << ",storage=" << storage << endl;
    }

};
int main()
{
    Computer computer("Intel", "NVIDIA RTX 3060Ti", 16, 512);
    cout << "基类对象: " << endl;
    computer.show();
    Laptop laptop("Intel", "NVIDIA RTX 3060", 16, 512);
    cout << "将派生类对象赋值给基类对象: " << endl;
    computer = laptop; // 将派生类对象赋值给基类对象
    computer.show();
    return 0;
}
```

**说明:**

1. 声明为指向基类对象的指针可以指向它的公有派生的对象，但不允许指向它的私有派生的对象。
2. 允许将一个声明为指向基类的指针指向其公有派生类的对象，但是不能将一个声明为指向派生类对象的指针指向其基类的一个对象

```
基类对象:
Computer: cpu=Intel,gpu=NVIDIA RTX 3060Ti,ram=16,storage=512
将派生类对象赋值给基类对象:
Computer: cpu=Intel,gpu=NVIDIA RTX 3060,ram=16,storage=512
```

# 知识点总结

Key Points summary

04.

# 知识点总结

- 基类与派生类对象之间的赋值兼容规则是指在需要基类对象的任何地方，都可以使用公有派生类的对象来代替
- 派生类对象可以向基类对象赋值，即用派生类对象中从基类继承下来的数据成员，逐个赋值给基类对象的数据成员。
- 派生类对象可以初始化基类对象的引用。
- 派生类对象的地址可以赋给指向基类对象的指针。
- 如果函数的形参是基类对象或基类对象的引用，在调用函数时可以用派生类对象作为实参。
- 声明为指向基类对象的指针可以指向它的公有派生的对象，但不允许指向它的私有派生的对象。
- 允许将一个声明为指向基类的指针指向其公有派生类的对象，但是不能将一个声明为指向派生类对象的指针指向其基类的一个对象

THANK YOU FOR WATCHING

# 恳请大家批评指正

小组成员：邵东林 谭棵 苏怡力