【C++】Day72

Class	C++
Date	@March 13, 2022
Material	
Series Number	
Summary	

【Ch15】OOP

15.2.3 Conversions and Inheritance

Warning: Understanding conversions between base and derived class is essential to understanding how object-oriented programming works in C++.

We can bind a pointer or reference to a base-class type to an object of a type derived from the base class.

For example, we can use a `Quote&` to refer to a `Bulk_quote` object, and we can assign the address of a `Bulk_quote` object to a `Quote*`.

The fact that we can bind a reference to a base-class type to a derived object has a crucially important implication: When we use a reference (or pointer) to a base-class type, we don't know the actual type of the object to which the pointer or reference is bound.

That object can be an object of the base class or it can be an object of a derived class.

Note: The smart pointer classes support the derived-to-base conversion.

Static Type and Dynamic Type

When we use types related by inheritance, we often need to distinguish between the static type of a variable and the dynamic type of the object that expression represents.

The static type of an expression is always known at compile time-it is the type with which a variable is declared or that an expression yields.

The dynamic type is the type of the object in memory that the variable or expression represents. The dynamic type may not be known until run time.

For example, when `print_total` calls `net_price`:

```
double ret = item.net_price(n);
```

we know that the static type of item is `Quote&`. The dynamic type depends on the type of the argument to which item is bound.

The dynamic type of an expression that is neither a reference nor a pointer is always the same as that expression's static type.

For example, a variable of type `Quote` is always a `Quote` object.

Note: It is crucial to understand that the static type of a pointer or reference to a base class may differ from its dynamic type.

There is No Implicit Conversion from Base to Derived

Because the base class does not contain members of the derived class, it is impossible to cast an object of a base class to an object of a derived class.

What is sometimes a bit surprising is that we cannot convert from base to derived even when a base pointer of reference is bound to a derived object:

```
Bulk_quote bulk;
Quote *itemP = &bulk;
Bulk_quote *bulkP = itemP; // error: cannot convert base to derived
```

The compiler has no way to know (at compile time) that a specific conversion will be safe at run time.