

Discussion 1: A brief review of C++

Jan.7 2005

Caijie Zhang

<http://www.cs.ucsb.edu/~caijie>

1. Basic class syntax

A *class* in C++ consists of its members. These members can be either data or functions (sometimes are called methods). Each *instance* of a class is an *object*. Each object contains the data components specified in the class.

(1) Visibility of class members (p13)

Public: A member that is public may be accessed by methods in any class

Private: A member that is private may only be accessed by methods in its class.

Protected: A member that is protected may only be accessed by methods in its class or its subclass.

Tip: In a class, all members are private by default, so the initial public is not optional.

(2) Constructors (p13)

A constructor is a method that describes how an instance of the class is constructed. Its name is the same as that of the class and it doesn't have a return value.

Default constructor: If no constructor is explicitly defined, one that initialized the data members using language defaults is automatically generated.

Tip: There may be several different constructors in the same class with different parameters

(3) Constant Member Functions (p15)

A member function that examines but does not change the state of its object is an *accessor*.

For example: `int read() const;`

(4) Parameter Passing (p21)

Call by value, Call by reference and Call by constant reference

Tip: If the formal parameter should be able to change the value of the actual argument, you must use call by reference. Otherwise, the value of the actual argument cannot be changed by the formal parameter. If the type is a primitive type, use call by value. Otherwise, the type is a class type and would generally be passed using call by constant reference.

For example: `double avg(int n, bool & errorFlag, const String & name);`

2. Separation of Interface and Implementation

(1) Preprocessor Commands (p15)

The interface is typically placed in a file that ends with .h. Source code that requires knowledge of the interface must *#include* the interface file.

Tip: To avoid read an interface file more than once: #ifndef _a_h_ #define _a_h_ #endif

(2) Scoping Operator (p17)

The syntax is `ClassName::member`. The `::` is called the *scoping operator*

(3) Signatures must match exactly (p17)

The signature of an implemented member function must match exactly the signature listed in the class interface.

3. Templates

(1) Function Templates (p30)

A *function template* is not an actual function, but instead is a pattern for what could become a function.

The line containing the template declaration indicates that *Comparable* is the template argument: it can be replaced by any type to generate a function.

(2) Class Templates (p32)

A *class template* works much like a function template. It is not an actual class, but instead is a pattern for what could become a class.

The line containing the template declaration indicates that *Object* is the template argument: it can be replaced by any type to generate a class.