# Q&A List

2016011446 翁家翌

题目编号：2,6,10,14,18

The aim of the QA list is to ensure that the key knowledge in this course is delivered to you. Some questions are relatively easy to answer. You may just copy and paste from our slides/books. But some questions are challenging, which need more background knowledge than what we have learned in class. Please try to find out the answers by yourself through Baidu/Google or other reference books.

Assume the question No. is x and the student No. is y, then each student only needs to answer the following questions: {x | x % 4 == y % 4}. Please input answers after the assigned questions within the word file.

Besides answering the required questions, each student is encouraged to propose other questions and the corresponding answers. Throughout this semester, students with enough GOOD new questions along with GOOD answers will be granted up to 5 bonus credits.

## Lecture 1

* Command-line argument（命令行参数）
  1. Suppose that you want to write an *A+B* program, which get A and B from command-line argument, how to complete the task? Please give an example of the source code, as well as how to compile and run the program.
  2. Please list（列举）the potential advantages and disadvantages of obtaining arguments by file reading vs. by command-line argument.

文件：大数据io，便于保存数据

命令行：方便写脚本，直观，实时反馈

* 1. How to input a command-line argument with spaces in it, e.g., how to input ‘Hello World’ as one argument?
  2. What is the meaning of *argc* and *argv*? What are their data types? If command “*a.exe 1 2 3”* is executed in the terminal (终端), what are the values of *argc* and *argv*?
* Translation（翻译）from source code（源代码）to binary executable（二进制可执行文件）
  1. Please explain how interpreters（解释器）and compilers（编译器）work. What are the typical programming languages (典型编程语言) for each of the two translation methods? Please list the advantages and disadvantages of interpreters vs. compilers.
  2. GNU compiler g++ is universally used in both commercial and research projects. So you are supposed to know how to compile C++ source files into the binary file. State the basic usage (基本用法) to compile a source file. List useful compiling options (编译选项) as many as possible, and explain why and when we need these options.

Usage: g++ [options] file...

Options:

-pass-exit-codes Exit with highest error code from a phase

--help Display this information

--target-help Display target specific command line options

--help={common|optimizers|params|target|warnings|[^]{joined|separate|undocumented}}[,...]

Display specific types of command line options

(Use '-v --help' to display command line options of sub-processes)

--version Display compiler version information

-dumpspecs Display all of the built in spec strings

-dumpversion Display the version of the compiler

-dumpmachine Display the compiler's target processor

-print-search-dirs Display the directories in the compiler's search path

-print-libgcc-file-name Display the name of the compiler's companion library

-print-file-name=<lib> Display the full path to library <lib>

-print-prog-name=<prog> Display the full path to compiler component <prog>

-print-multiarch Display the target's normalized GNU triplet, used as

a component in the library path

-print-multi-directory Display the root directory for versions of libgcc

-print-multi-lib Display the mapping between command line options and

multiple library search directories

-print-multi-os-directory Display the relative path to OS libraries

-print-sysroot Display the target libraries directory

-print-sysroot-headers-suffix Display the sysroot suffix used to find headers

-Wa,<options> Pass comma-separated <options> on to the assembler

-Wp,<options> Pass comma-separated <options> on to the preprocessor

-Wl,<options> Pass comma-separated <options> on to the linker

-Xassembler <arg> Pass <arg> on to the assembler

-Xpreprocessor <arg> Pass <arg> on to the preprocessor

-Xlinker <arg> Pass <arg> on to the linker

-save-temps Do not delete intermediate files

-save-temps=<arg> Do not delete intermediate files

-no-canonical-prefixes Do not canonicalize paths when building relative

prefixes to other gcc components

-pipe Use pipes rather than intermediate files

-time Time the execution of each subprocess

-specs=<file> Override built-in specs with the contents of <file>

-std=<standard> Assume that the input sources are for <standard>

--sysroot=<directory> Use <directory> as the root directory for headers

and libraries

-B <directory> Add <directory> to the compiler's search paths

-v Display the programs invoked by the compiler

-### Like -v but options quoted and commands not executed

-E Preprocess only; do not compile, assemble or link

-S Compile only; do not assemble or link

-c Compile and assemble, but do not link

-o <file> Place the output into <file>

-pie Create a position independent executable

-shared Create a shared library

-x <language> Specify the language of the following input files

Permissible languages include: c c++ assembler none

'none' means revert to the default behavior of

guessing the language based on the file's extension

Options starting with -g, -f, -m, -O, -W, or --param are automatically passed on to the various sub-processes invoked by g++. In order to pass other options on to these processes the -W<letter> options must be used.

* 1. What is a library (库)? How to build a library using g++ compiler? How to use a library using g++ compiler? At what stage is a library loaded (载入)? Is it always during the linking stage (链接阶段)? Please explain the difference between static (静态) and dynamic (动态) libraries.
* Multi-file project
  1. Assume that there are *main.cpp*, *sum.cpp*, *product.cpp, and functions.h*, please write down the command to compile them into *main.exe* with g++.
  2. Please explain why multiple files are necessary for a project, especially the advantages of multi-file project compared with single-file project.
  3. What is separate compilation (分段编译)? Why is it better than direct compilation, i.e., to generate the executable directly from the source files? Please state the whole process of separate compilation.

g++ -c main.cpp -o main.o

g++ main.o -omain

直接编译的话，对于大工程来说太花时间，修改一个地方就要编译整个项目，完全没必要

* 1. What should you do to use a global variable (全局变量) or function (函数) in several source files (源文件)? Please give as many ways as you can.
  2. Is it allowed to redefine a variable or function with the keyword **extern**? For example, “extern int a = 2;”. Why?
  3. Please explain the usage of header files (头文件). Is it allowed to define the function body (函数体) in the header file? Do we need to compile header files using g++? Why?
  4. Please explain the differences between “” and <> when including a header file in the source file.

<>：环境变量的各个目录

“”：以上+当前目录

* Make and makefile
  1. Please explain the dependency rules of *make*.
  2. How to enable separate compilation (分段编译) of source files using *make and makefile*? Assume that we have *main.cpp, sum.cpp, product.cpp, and functions.h,* please write the makefile to generate *main.exe* byseparate compilation.
  3. On linux, how to generate a binary file with a makefile? How to use the library file to generate an executable file with makefile? Please give a simple example on Linux.
  4. Please explain more options for make, which are the command-line arguments of *make*.

用法：make [选项] [目标] ...

选项：

-b, -m 忽略兼容性。

-B, --always-make 无条件 make 所有目标。

-C DIRECTORY, --directory=DIRECTORY

在执行前先切换到 DIRECTORY 目录。

-d 打印大量调试信息。

--debug[=FLAGS] 打印各种调试信息。

-e, --environment-overrides

环境变量覆盖 makefile 中的变量。

--eval=STRING Evaluate STRING as a makefile statement.

-f FILE, --file=FILE, --makefile=FILE

从 FILE 中读入 makefile。

-h, --help 打印该消息并退出。

-i, --ignore-errors Ignore errors from recipes.

-I DIRECTORY, --include-dir=DIRECTORY

在 DIRECTORY 中搜索被包含的 makefile。

-j [N], --jobs[=N] 同时允许 N 个任务；无参数表明允许无限个任务。

-k, --keep-going 当某些目标无法创建时仍然继续。

-l [N], --load-average[=N], --max-load[=N]

在系统负载高于 N 时不启动多任务。

-L, --check-symlink-times 使用软链接及软链接目标中修改时间较晚的一个。

-n, --just-print, --dry-run, --recon

Don't actually run any recipe; just print them.

-o FILE, --old-file=FILE, --assume-old=FILE

将 FILE 当做很旧，不必重新生成。

-O[TYPE], --output-sync[=TYPE]

Synchronize output of parallel jobs by TYPE.

-p, --print-data-base 打印 make 的内部数据库。

-q, --question Run no recipe; exit status says if up to date.

-r, --no-builtin-rules 禁用内置隐含规则。

-R, --no-builtin-variables 禁用内置变量设置。

-s, --silent, --quiet Don't echo recipes.

-S, --no-keep-going, --stop

关闭 -k。

-t, --touch touch 目标而不是重新创建它们。

--trace Print tracing information.

-v, --version 打印 make 的版本号并退出。

-w, --print-directory 打印当前目录。

--no-print-directory 关闭 -w，即使 -w 默认开启。

-W FILE, --what-if=FILE, --new-file=FILE, --assume-new=FILE

将 FILE 当做最新。

--warn-undefined-variables 当引用未定义变量的时候发出警告。

* 1. Please list useful advanced grammars (高级语法) of makefile as many as you can. It is suggested to explain the usage of these grammars. For example, it is convenient to automatically detect all the source files in a folder (自动检测文件夹下的所有文件) and automatically compile them all. How does *makefile* support this?

## Lecture 2

* Abstraction

1. Generally speaking, what are the two major parts of a class? Please define a class that describes a *computer*, and then give an example of how to instantiate an object and access its members.
2. Where should we place the class definition, in the header file or the source file? Please try to explain why header files are necessary.

头文件。头文件包含所有声明，方便include和最终link。为了隐藏实现而需要头文件，而只有声明没有实现也会加快编译速度

1. Please try to explain why the designers want to organize data and functions into classes? What are the advantages and disadvantages?

* Access control

1. Please try to use access control （访问控制）to enable information hiding of your class definition for *computer* in Problem No.1.
2. Please try to explain the benefits of access control.
3. Please tell the difference between keywords ***public***, ***private*** and ***protected***. We know that access control works for objects of the class. Does access control work in member functions? I.e., is it possible that one member function cannot access another member?

public：大家都能访问

private：只在这个class内部才能访问

protected：在这个class内部、以及使用public或protected继承它的class能访问

1. Please try to explain why it is suggested to define a member function（成员函数） outside class definition.
2. Please explain the differences between keyword ***struct*** and ***class***.

* Friend

1. What should you do to permit a non-member function to access private or protected members? What about permitting another class to access them? Please give an example.

* Inline

1. What are the advantages of inline functions? When and where should we use keyword ***inline***?

编译的时候内联到原函数中，从而不产生额外的调用函数开销。

语句少、未调用其它函数的函数

1. Please try to explain the implementation of inline functions (内联函数的实现) in compiler, and explain the reason for the advantages of inline functions.
2. What will happen if we abuse (滥用) keyword ***inline***?

* Header guarding

1. Please give an example of header guarding using preprocessor directives (预编译命令) “#ifndef … #define …#endif”, and try to explain the possible compiling errors without them.

* This pointer

1. What does ***this*** mean? How to avoid the name conflicts (名字冲突) between member and non-member variables with ***this***? How to return current objects in member functions?

this指包含该函数/变量的类的变量

假设class A内部有一个int a; A(int a){this->a=a;}是不会有歧义的。

return \*this;

* Memory allocation

1. What are the differences between ***new/delete*** and ***malloc/free***?
2. Why does C++ bring in ***new/delete*** to replace ***malloc/free*?**

* Incomplete class

1. Please list the situations where we need to use incomplete type, i.e., forward declaration (前向声明).

## Lecture 3

* Scope

1. What is the scopes of a, b, c and d?

int a;

int f ()

{

static int c;

int b;

{int c; }

}

int main ()

{

f ();

}

2. What is the scopes of objects of O?

struct O

{

int i;

};

O f (O o)

{

return o;

}

const O& g (O o)

{

return std::move(o);

}

O h (const O& o)

{

return o;

}

作用域为整个函数

* Constructor and destructor

3. Is it necessary to declare destructors（析构函数） as public members? What about constructors（构造函数）?

Yes. 如果不是public，那么就无法构造/析构，也就无法使用该类

4. When does copy constructors and move constructors get called?

5. What are the benefits of constructors and destructors? Try to declare a class with constructors and destructor.

6. Is it possible to call constructor and destructor explicitly? It is necessary?

构造可以，析构不需要。因为构造的方式可以不一样，无法只通过默认构造方式进行构造

7. Specify which constructor will get called for each statement.

constructor ();

constructor (int);

constructor a;

constructor b(5);

constructor \*c = new constructor;

constructor \*d = new constructor (5);

constructor e[5] = {};

constructor f[5] = {(1), (2)};

8. How to define an object on stack（栈）? What about heap? What will happen if you define oversize object exceed your system’s limit?

9. Do we need to free memory of stack or heap? How to verify that?

10. What is the difference between the stack and the heap(堆)?

栈由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。

堆一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收。

11. It is necessary to apply destructor? Explain the reason.

12. Is the code below work fine? Explain the reason. How to fix the problem?

struct O

{

int i;

O (int i) {}

};

int main ()

{

O\* o = new O[5];

}

不行，没有无参数的构造函数，应该把第四行改成O(int i=0){}

## Lecture 4

* Function Overloading(函数重载)

1. It is necessary to overloaded functions? Please explain why it necessary and specify the situation that need to overloaded functions. What kinds of functions can be overloaded?

2. How the compilers distinguish two functions with the same name?

依照参数匹配

* Default Argument

3. Where to put the default arguments （默认参数）on? declarations or definitions?

4. Which arguments of a function can be defaulted with default values? Explain the reason.

5. Try to add an argument to an existing function with modifying the code as less as possible

6. Is the code below work fine?

struct O

{

int i;

O (int i = 0) {}

O (int i, int j = 0) {}

};

int main ()

{

O o(6);

}

No. o(6)会产生歧义

* Macro（宏）

7. Give the details of *header guarding method* and explain why it is necessary.

8. Explain why we should avoid using macros to define numbers? Please give the alternative grammars to replace using macros.

9. In the following example, explain the reason to remove commented line for avoiding compile error and write down the output of the code below.

enum smallenum: std::int16\_t {A, B, C };

enum color {red, yellow, green = 20, blue};

enum class altitude: char{ high='h',low='l',};

enum {d,e,f = e + 2};

int f1 (int a) {}

int f2 (char a) {}

int main ()

{

std::cout << sizeof(smallenum) << " " << A << " " << C << std::endl;

std::cout << sizeof(red) << std::endl;

std::cout << sizeof(d) << " " << d << std::endl;

f1 (d);

f1 (color::red);

//f2 (altitude::high);

f1 (A);

}

* Constants

10. Please explain why we can define a constant variable in the header file without causing compiling errors? What should be done for you to use a constant variable defined in another source file?

因为include一个文件就相当于把这个文件直接粘进来，变成单文件，所以没事

在.h中extern const blabla = …;然后在cpp中const blabla = …;

11. Please explain the difference between the following statements.



12. How to initialize a constant member variable(常量成员变量)?

13. How to convert a constant variable into a non-constant one?

14. Where should the keyword “const” be placed to declare or define a constant member function(常量成员函数)? Please give an example.

class A{

int a,b;

int calc()const{return a+b;}

};

15. Please state the restrictions of constant member functions. Can we have any other alternative (i.e., access the members in other ways)?

## Lecture 5

* Inline function(内联函数)

1. Why do we need C++ grammars such as function-like macros and inline functions? Try to describe a typical situation where the two grammars are used.

2. What is problem with the following code?



会变成 while((s=getchar())>=’A’&&(s=getchar())<=’Z’)，从而变成跳着读字符

3. Please give an example showing why brackets “()” are important in avoiding errors in function-like macros（宏函数）.

4. Please explain the main advantages of inline functions over function-like macros.

5. Is the keyword inline below necessary? Please explain this grammar.

class A {

int x\_;

public:

inline int setX(int x) { x\_ = x; }

inline int getX() { return x\_; }

};

6. Where is keyword ***inline*** supposed to be placed? on declarations or definitions? Please try to explain why.

定义中。inline 函数对编译器而言必须是可见的，以便它能够在调用点内展开该函数。

7. Can we add the keyword ***inline*** for every function? Please state the rule for setting inline functions (e.g., from google coding style).

8. Find the relation or similarity between ***inline*** and the C++ ***class*** grammar?

* Namespace

9. Explain why we need namespace and try to describe some situations where namespaces are needed.

10. Please enumerate all the ways of creating a namespace and referring to the names in the namespace using some examples.

namespace afo{…}

namespace afo = std;

using namespace afo;

11. What is the scope (作用域) of “using namespace”? Is it legal to include “using namespace” in header files? Please explain the reason.

* Static

12. Please state in detail the lifetime (生命期) of static variables. How is *static* related to C++ *class*?

13. Where are static variables stored in the memory?

14. Can we access to a static variable (静态变量) defined in another source file directly and how? Please give an example.

.h: exetern static int a;

.cpp: static int a=0;

15. Please state the concept of internal linkage (内部链接), and list all the different types of variables with the internal linkage.

16. Can we define two static variables with the same name in two different source files?

17. What are the differences between static and non-static member functions?

18. Please list all the approaches to access the static member functions (静态成员函数) and static member variables（静态成员变量）, and give examples.

class A{static int a;static int geta(){return a;}};

A::a=0; cout<<A::geta();

19. Please give an example of object counting (对象计数). Please state the necessity of object counting and try to describe a typical situation of using object counting.

* Singleton

20. Please give a complete example of the singleton design pattern.

21. Please explain the necessity of singleton and try to describe some typical situations of using singletons.

## Lecture 6

* Reference

1. Please explain the basic concept of *reference*（引用）.

2. Is it legal to declare a reference variable without defining it? Is it legal to change the value of a *reference variable*（改变引用变量的值）? Please explain in detail with examples.

No. Yes. int a=1;int&b=a; b=2;printf(“%d\n”,a);//2

3. What are the advantages of reference compared with pointer?

4. It is possible to return a reference of a local variable of a function? Why? What are the advantages of returning references rather than values in functions?

5. How to apply references as member variables? Please give a complete example.

6. It is possible to use reference variables as elements in std containers such as std::vector? And how about pointers?

No. Yes.

7. Please list some examples of returning multiple values simultaneously from functions.

8. What are the differences between passing references (引用传参) and passing values (值传参) in function arguments?

9. Please explain the *least privilege principle* preferably with an example.

10. What is the difference between rvalue references and lvalue references? Please state the basic concept and usage of rvalue references.

左值：可寻址变量

右值：可读变量

class obj{obj(obj&& other){cout<<"create obj"<<endl;}};

* Copy constructor

11. Please give an example of copy constructor.

12. How is reference associated with the C++ class grammar?

13. Please list the cases where copy constructors are called. How to avoid the copy constructor in the listed cases?

14. What’s the function of compiler-generated copy constructor? What is the fatal disadvantage of depending on the compiler-generated copy constructor? Please give an example of the fatal error.

默认浅复制。class A{int \*m;}a; 如果A b(a);的话，在析构的时候m会被删两次，产生段错误

15. Why should we avoid copy constructors? How to ensure that copy constructor not be called?

16. Please state the differences between move constructors and copy constructors.

## Lecture 7

* Composition (组合)

1. Please explain the concept of *composition* and try to describe a car using *composition* with examples. The example is required to include the declaration and definition of constructors and destructor.

2. Please list the different ways (i.e., public, protected, private) of *composition* and discuss their safety.

（1）public：不安全，能够被所有成员访问

（2）protected：较安全，能够被派生类的成员访问，不能够被其它成员访问

（3）private：安全，只能被class内部的成员访问

（4）friend：有风险，将class内部的私有成员暴露给该函数

3. Please explain the differences between *aggregation* (聚集) and *borrow* (借用) in composition with examples.

4. Please explain the basic concept and usage of *aggregate* initialization.

5. Please state the order of constructing and destructing member objects.

6. Please show the usage of constructor initializer list (构造函数初始化列表) in *composition* with examples.

class Point\_3{public:double x,y,z;

Point\_3(double \_x=0,double \_y=0,double \_z=0):x(\_x),y(\_y),z(\_z){}};

7. We recommend that class members should be initialized by the constructor initializer list (构造函数初始化列表) in the same order as their declaration in the class? Why? And in which order are the members actually initialize (constructors initializer list or their declaration)?

* Inheritance (继承)

8. Please explain the concept of *inheritance* with examples. It is required that the example include the declaration and definition of constructors and destructor.

9. Please explain the differences between public, protected and private inheritance for different access control (public, protected and private) in base class, preferably with a form (表格).

10. Please state the order of constructing and destructing the base parts and derived parts for a derived object.

构造：对于每个类，先构造基类，然后按照声明的顺序构造每个成员，最后该类构造完毕。

析构：按照构造该类的倒序析构所有成员。

11. Which functions cannot be inherited from the *base class* (基类)? And please try to explain the reason.

12. Please explain the differences between public inheritance and private inheritance. What are the typical examples using the different inheritance methods?

13. Please give an example showing how to re-declare a private member function inherited from base class to be public.

14. Please explain the basic concept of incremental development (增量式).

增量式开发，又叫增量开发，是软件工程当中一种常用的软件开发过程思想。其中增量是指在软件开发过程中，先开发主要功能模块，再开发次要功能模块，逐步完善，最终开发出符合需求的软件产品。

15. Please explain the concept, usage and rules of *multiple inheritance*（多重继承）.

* Code reuse

16. Please list the common ways of code reuse, and try to compare between the different ways.

## Lecture 8

* Name hiding (名字隐藏)

1. Please give an example illustrating the effect of name hiding.

2. How to overload the member functions of base class without name hiding? Give an example for each method.

class A{void try(){}};

class B:public A{public:using A::try;};

3. How to expose some specific member functions of base class? Give an example for each method.

* Composition vs. inheritance

4. Please think of an example using public composition. Then please try to modify the example and use private composition instead.

5. Please think of an example using public inheritance. Then please try to modify the example and use private inheritance instead. If private inheritance is not feasible for the example, please explain the reason.

6. Please explain the connotation of all kinds of composition and inheritance (public, protected, priviate).

继承：基类接口能够直接被外部调用，不需要重载

组合：调用借口时需要访问该类成员的名称

|  |  |  |  |
| --- | --- | --- | --- |
|  | 原来是public | 原来是protected | 原来是private |
| public继承/组合： | 转换为public | 转换为protected | 转换为private |
| protected继承/组合： | 转换为protected | 转换为protected | 转换为private |
| private继承/组合： | 转换为private | 转换为private | 转换为private |

7. Please explain the differences between private inheritance and public inheritance.

8. Please give an example illustrating when private inheritance can be used.

* Upcasting (向上转型)

9. What is the meaning of upcasting? It is true that upcasting works fine in *protected inheritance* (保护继承) and *private inheritance* (私有继承)? Please try to explain why it works or not.

10. Please explain the reason why it is necessary to define a copy-constructor by upcasting. Please give an example of the copy-constructor with upcasting.

如果用指针声明一个类变量的时候，直接转换会出问题。

class B;

class A{public:A(const B&tmp){...}};

class B:public A{...}

11. What is the meaning of *object slicing* (对象切割)? Do you think object slicing is good? Why?

12. What are the differences between *object slicing* and *upcasting*?

## Lecture 9

* Virtual function (虚函数)

1. Please give an example to explain *polymorphism* (多态性).

2. Is it necessary to redefine a virtual function in *derived class* (派生类)? And is the keyword *virtual* necessary in redefining the virtual function in *derived class*?

第一问：看情况。如果是纯虚函数则必须重新定义

第二问：不必要。

3. What is the meaning of keyword *final*? What problem will be caused without it? Please give an example showing a typical situation that keyword *final* is necessary.

4. What is the meaning of keyword *override*? What problem will be caused without it? Please give an example of a typical situation that keyword *override* is necessary.

* Late binding (后期绑定)

5. Please explain the difference between *early binding* and *late binding*.

6. What kind of functions are bond with early binding? And what functions are bond with late binding?

前期绑定：在基类中实现，对于每个派生类而言所需要的功能都一样

后期绑定：基类中用虚函数声明，在每个派生类中分别实现。

7. Does late binding work for objects? How about pointers and references?

* VTABLE (虚函数表) and VPTR (虚函数指针)

8. What are VTABLE and VPTR? Where are they stored in?

9. How do VTABLE and VPTR enable polymorphism?

* Pure virtual function (纯虚函数) and abstract class (抽象类)

10. Why *pure virtual functions* (纯虚函数) are necessary in some cases? Please give an example of pure virtual function.

当定义一个抽象类的时候需要，比如定义一个数学上的群，只有运算关系的限制。

virtual void try(int)=0;

11. What is an abstract class?

12. Please explain the purpose and advantages of abstract class.

13. Can *pure virtual function* have function body? What is the purpose to provide a function body for a pure virtual function?

14. Please give an example where a derived class of an abstract class is still an abstract class.

class A{public:virtual void try()=0;};

class B:public A{};

B仍为抽象类

## Lecture 10

* Singleton

1. What is the basic function (功能) of singleton?

2. Please try to implement a singleton example where the instance is dynamically created (动态创建) at its first visit.

class A{public:

static int cnt;

void first\_init(){…}

A(){if(cnt==0)cnt++,first\_init();}

};

A::cnt=0;

3. Please list some typical situations where singleton is useful or necessary.

4. Please plot a typical UML class diagram of singleton pattern.

* Adapter

5. What is the basic function (功能) of adapter?

6. Please give an example using adapter design pattern.

class MyString{

std::string a;

public:

MyString(const std::string&tmp=””):a(tmp){}

char operator[](int t){return a[t]>=’A’&&a[t]<=’Z’?a[t]-’A’+’a’:a[t];}

};

7. Please list some typical situations where adapter is useful or necessary.

8. Please provide an example to illustrate advantages of the OOP principle “Program to an interface, not an implementation”.

9. Please provide an example to illustrate advantages of the open-closed principle.

10. Please try to explain the relation between “Program to an interface” and the open-closed principle.

都是把需求放在首位，根据需求来设计接口，根据接口来设计代码，接口可拓展。

11. Please plot a typical UML class diagram with “object adapter” design pattern.

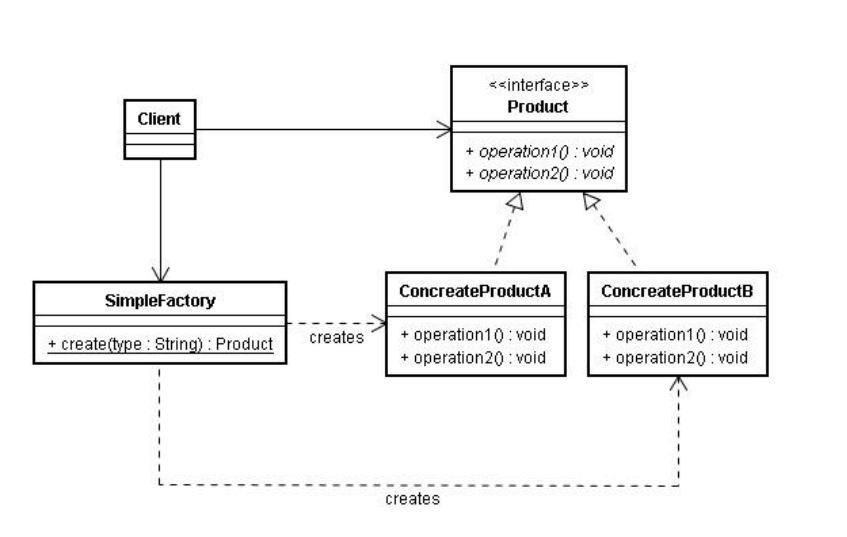
* Simple Factory

11. What is the basic function (功能) of simple factory?

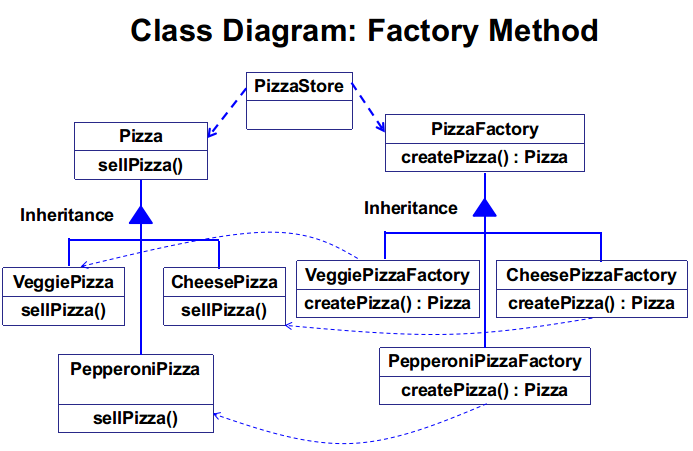
12. Please give an example using simple factory.

13. Please list some typical situations where simple factory is useful or necessary.

14. Please plot a typical UML class diagram with simple factory pattern.



## Lecture 11

* Factory method

1. What is the basic function (功能) of factory method?

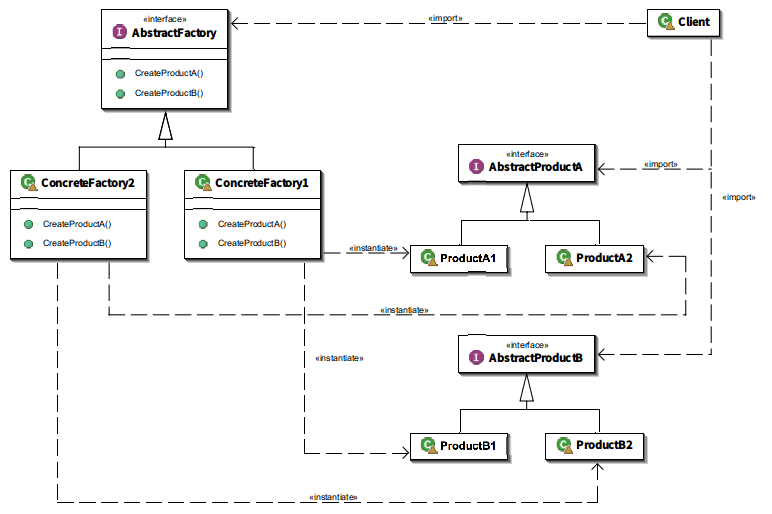
2. Please plot a typical UML class diagram with factory method.

见←

3. Please list some typical situations where factory method is useful or necessary.

4. Please give an example using factory method design pattern.

5. Please state the single responsibility principle.

6. Please plot a typical UML class diagram with abstract factory pattern for a general situation.

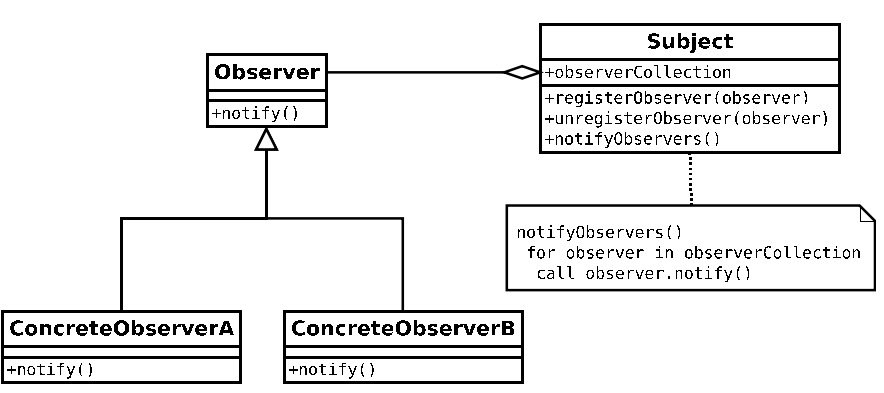
见→

7. Please state the main idea of *abstract factory*. What is its advantage compared with factory method?

8. What is the main idea of *parameterized factory method*? What is the advantages of parameterized factory method compared with factory pattern and simple factory pattern?

9. What is the key difference among simple factory, factory method, parameterized factory method, and abstract factory?

* Observer

10. Plot a typical UML class diagram with observer pattern for a general situation.

见←

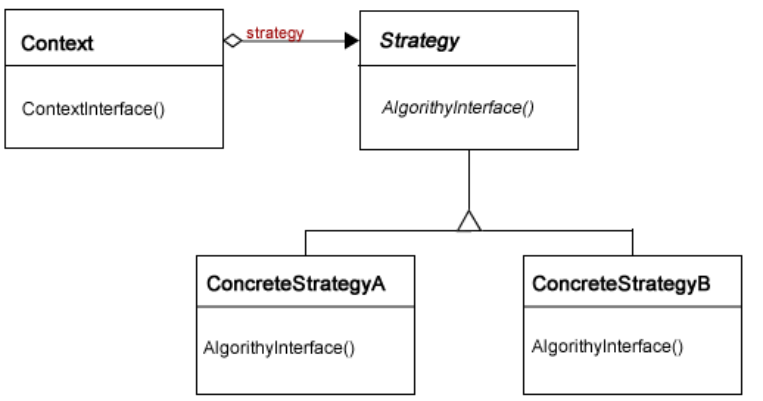
11. Please give an example using observer pattern?

12. List some typical situations where observer pattern is useful or necessary.

13. What is the basic function (功能) of observer pattern?

## Lecture 12

* Strategy

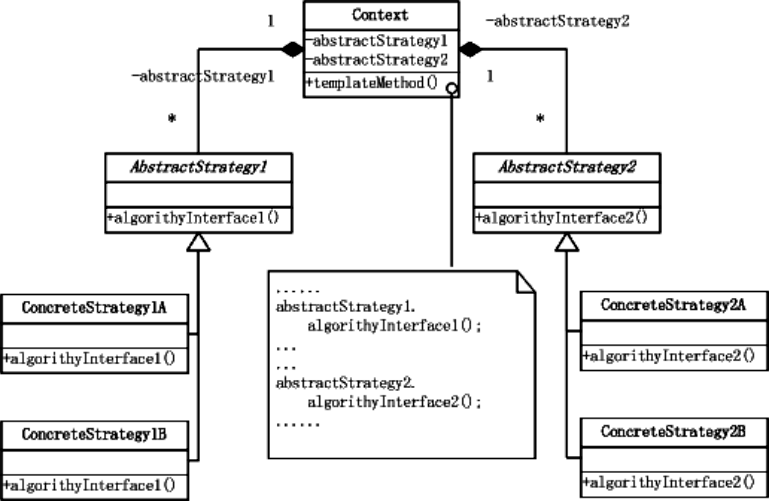
1. What is the basic function (功能) of strategy pattern?

2. Plot a typical UML class diagram with strategy pattern for a general situation

见→

3. List some typical situations where strategy pattern is useful or necessary.

4. Please give an example using strategy pattern.

* Template method

5. What is the basic function (功能) of template pattern?

6. Plot a typical UML class diagram with template pattern for a general situation.

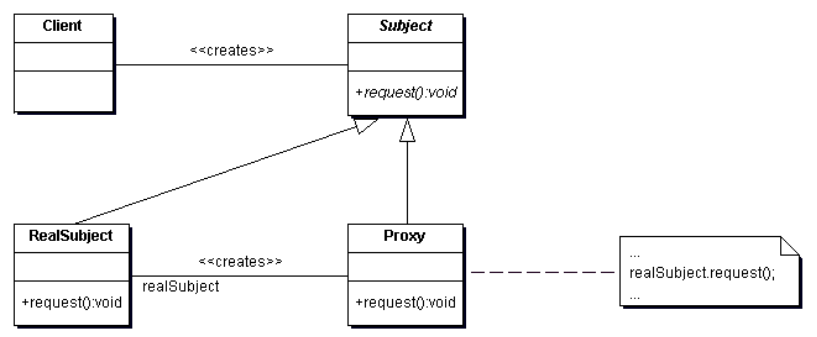
见←

7. List some typical situations where template pattern is useful or necessary.

8. Please give an example using template method design pattern.

* Proxy

9. What is the basic function (功能) of proxy pattern?

10. Plot a typical UML class diagram with proxy pattern for a general situation

见→

11. List some typical situations where proxy pattern is useful or necessary.

12. Please give an example using proxy pattern.

13. What is the main difference between proxy and adapter?

* Command

14. What is the basic function (功能) of command pattern?

Encapsulates a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations (支持撤消操作)Using Command Pattern, we encapsulate method invocation (方法调用), so that the invoker (调用者) does not need to know implementation details

15. How to implement Undo and Redo in command pattern?

## Lecture 13

* Operator overloading

1. Please state the essence (本质) of operator overloading.

2. Please state the restrictions on operator overloading.

Cannot change the usage

cout << a << endl; -> a << cout;

Cannot define new operators by yourself

Example: using “\*\*” for exponentiation is illegal

Cannot change precedence rules

“\*” > “+”

“+” > “||”

Cannot change the number of operands

Unary : “++”, “--"

Binary: “+”, “-”, “\*”, “/”

Operators that cannot be overloaded

“.”, “.\*”, “::”, “?:”

3. Assume @ is an operator overloaded as member function, please write down the declaration of the functions corresponding to the following statements.

a) x @ y;

b) @x;

c) y@;

4. Assume @ is an operator overloaded as friend function, please write down the declarations of functions corresponding to the following statements.

a) x @ y;

b) @x;

c) y@;

5. When are you supposed to overload operator as member functions? And when are you supposed to overload operator as friend functions? Please try to explain your answer in detail.

6. Why should stream operators “<<” and “>>” be overloaded as friend functions?

**Assume a class called Vector3 is declared as follows.**

class Vector3

{

public:

Vector3 (int x = 0, int y = 0, int z = 0);

~Vector3 ();

private:

int \_x, \_y, \_z;

};

否则输入输出流无法传递到下一个要被输出的变量的类中

7. Please write down the declaration of the function that is called for the following statement. (Note that vec3 is a well-defined object.)

Vector3 vec3\_ = vec3;

8. Please write down a declaration of stream operator “<<” for Vector3, making the following statements legal.

* 1. std::cout << vec3 << std::endl;
  2. std::cout << std::left << std::setw (5) << vec3 << std::endl;

9. Why is bit-copy dangerous, especially when there are pointer members in the class? Please give an example where bit-copy is adopted by the compiler, which potentially causes bugs.

10. Please write down a declaration of assignment operator “=” for Vector3 (see the definition above), making the following statements legal.

Vector3 vec, vec\_;

vec = vec\_ = vec3;

Vector3&operator=(const Vector3&a)const{return a;}

11. Please explain why self-assignment checking is important in overloading assignment operator.

12. Please state the difference between operator postfix ++ (--) and prefix ++ (--). Please write down the function body of operator postfix ++ and prefix ++ for the following class.

class Integer

{

public:

Integer (int x = 0): \_x (x) {}

private:

int \_x;

};

* Function object (函数对象)

13. What is a function object? What is the difference between function objects and normal functions?

14. What is the difference between function objects and normal objects?

函数对象：是一个方法，没有数据成员

普通对象：有数据成员，有内置函数

15. Please state the advantages of function object compared with function pointer (函数指针).

16. Please complete the missing code in the following program such that we can obtain the sum of the integers in *arr*.

#include <iostream>

#include <array>

#include <algorithm>

#include <random>

const size\_t size = 1 << 5;

std::ostream& operator << (std::ostream& os, const Sum& sum)

{

os << "sum:" << sum.\_x << std::endl;

return os;

}

int main ()

{

std::random\_device rd;

std::mt19937 mt (rd ());

std::uniform\_int\_distribution<> dis (0, size);

std::array<int, size> arr;

for (auto& element: arr)

element = dis (mt);

Sum s (0);

s = std::for\_each (std::begin (arr), std::end (arr), s);

std::cout << s;

return 0;

}

## Lecture 14

* Template

1. Please state the original intention (意图) of bringing in template mechanism.

2. Please write down the general form of function template definition.

template<class T>T min(T&a,T&b){return a<b?a:b;}

3. Please state all the data types that can be template arguments.

4. What is template specialization (模板特化)? Please write down the general form of it.

5. Please implement (实现) the following code such that we can get Fibonacci sequence value of 100 within O(1) runtime.

int main ()

{

Fib<100> f;

cout << f.value << endl;

return 0;

}

6. What is implicit call instantiation of template? When does it work?

bool x = min(3, 5);

在编译器能够自动推导出参数类别的时候

7. Can a template parameter appear twice or more in the argument list? Can a template argument never appear in the template function/class body?

8. Please write down the general form of class template.

9. Why are the definitions of template functions and template classes supposed to be placed in header files?

10. What should you do to use default value for template parameters? What should you do to use constant values in template? What should you do to inherit from a template class? Please illustrate them with certain examples.

template <class T=int>T min(T&a,T&b){return a<b?a:b;}

template <class T, double limit=0>T Relu(T a){return a>limit?a:limit;}

class bitGauss{

std::vector<std::bitset<500> >mat;

public:

...

};

11. Please plot the UML class diagram for the iterator design pattern.

12. Plese implement a general “count\_if” algorithm using template along with the iterator design pattern.

13. Please give an example to illustrate how the function object (函数对象) is used in STL.