NO.

Date . .

C++

一、常量：程序运行过程中值不变.

整型、实型、字符型、字符串、布尔型.

变量：auto 在栈：暂时性存储.(栈存方式)

static：在内存中以固定地址存放、整个程序运行期间有效.

extern：在所有函数和程序段中都可引用.

register：通用寄存器.

符号常量：const 数据类型说明符 常量名=常量值.

[强制多态]低级：char, short, int, unsigned, long, unsigned long, float, double

二、函数.

1. 参数传递

{值调用.(单向传递)

引用调用(声明引用时，同时对它初始化使之指向已存在的对象).

2. 内联函数.

编译时嵌入调用处，节省了参数传递、控制转移.

3. 默认形参

在定义时可预先声明默认的形参值、从右向左声明.

不同的调用域内允许说明不同的默认形参.

NO.

Date . .

4. 函数重载：

{形参类型不同
{形参个数不同.

编译器实现第法画成.

三. 类与对象

抽象（数据抽象）（行为抽象）.

封装：成为可复用模块.

继承.

多态.

[对象是类的实例化]

1. 访问控制：

基类的函数	基类	继承类
public	public protected private	public protected 不可直接访问
protected	public protected private	protected protected 不可直接访问
private	public protected private	private private 不可直接访问.

2. 类的成员函数：

N0.

Date · ·

④ 声明.

返回值类型 类名::函数成员名(参数表). {---}

② 带默认形参值的成员函数,(与前面所述相同).

③ 内联成员函数.: 在编译时被插入到每一个调用它的地方.

{ 隐式声明: 放在类体内.
显式声明: 加 inline 写在类体外.

④ 对象: 类名 对象名.

访问对象的公有成员: 对象名.公有成员函数名(参数表).

3.

① 构造函数:

1) 对数据成员的初始化. 在对象创建时自动调用

2) 若没有构造函数,编译器会自动生成 默认形式的构造函数(不做任何事情).

3) 可以是内联函数,可带参数表,可带默认的形参值,可重载.

4) 对象所占的内存只含数据成员,不含函数成员代码的副本.

② 拷贝构造函数:

把每个数据成员的值都复制到新对象中. (同类的对象).

{ 1) 当用类的一个对象去初始化该类的另一个对象时.
2) 调用形参是类的对象的函数,进行形实结合时.
3) 函数 返回值是类的对象的函数执行完,返回给调用者时.

N0.

Date . .

(调用拷贝构造函数将对象的数据成员值拷贝到临时无名对象中，临时对象的值再通过"="赋给调用者，然后临时无名对象自动消失，无需析构函数).

(若没定义类的拷贝构造函数，则自动生成默认的拷贝构造函数，完成复制数据成员的功能).

③析构函数.

在对象生存期结束时刻自动被调用，内存被释放.

可以是虚函数.

不显式声明则系统生成一个默认的空析构函数.

4. 类的组合. 即类的对象内嵌到另一个类中作为数据成员.

①构造函数: 类名::类名(形参表): 内嵌对象1(形参表), 内嵌对象2(形参)...

{类的初始化} ——转为初始化列表.

1) 对基本类型的数据成员也可通过初始化列表进行初始化，比赋值语句的效率高.

2) 调用构造函数的顺序:

1. 按内嵌对象的次序调用内嵌对象的constructor
↓
本类的constructor

②析构函数:

本类的destructor
↓
内嵌的destructor(反顺序).

NO.

Date . .

③拷贝构造函数：

形式：C::C(C &c1):b(c1.b)

C类包含B类的对象b作为成员

c1为C类的一个对象.

注：循环依赖时要前向引用声明.

四. C++程序结构

1. 作用域：标识符的有效范围

- 函数原型作用域（声明时的小圆括号内）
- 块作用域（花括号）.
  - 类作用域（如果类X的成员函数中没有声明同名的局部作用域标识符，那么在该函数内可访问数据成员）.
- 文件作用域.（始于声明点（外部），止于文件尾）. → 全局变量.

2. 可见性：标识符是否可以被引用.

先声明后引用.

同名隐藏规则（内层优先）.

3. 生存期.

- 静态生存期：static 对象类型 对象名（地址固定）.
- 动态生存期：（局部变量）（暂时性存储）

亚龙纸制品

N0.

Date

三. 类与对象(补)

5. 类的静态成员：static声明成员，有静态生存期，通过 类名::标识符 访问（通过非闭联函数访问）

类的非静态成员：在每个对象中都有一个copy

类的静态函数成员：访问该类的静态数据和函数成员.

保护对象之间共享的数据，不依赖于任何对象.

6. 类的友元.

友元函数：友元是一般函数或类的成员函数.

友元类：友元是一个类，其中所有成员函数都是友元函数 } 访问私有和保护成员

友元关系 { 不能传递 / 不是双向 / 不能继承.

7.

① 常引用：所引用的对象不能被更新，作形参可以减少不必要的复制.

const 类型说明符 &引用名 = 所引用的对象

② 常对象：必须初始化，不能被更新（整个对象的生存期）.

类名 const 对象名；

~~对象名.成员名 (实参表)~~ ~~通过对象名访问其公有数据~~

③ 常成员函数：返回类型 函数名(形参) const;

不能更新对象的数据成员.
不能调用类中的非常成员函数.
常对象只能调用常成员函数,也是常对象唯一的对外接口.
常成员函数是同名非常成员函数的重载.

⑩常数据成员: const (常量 or 常引用)
{在任何函数不能对其赋值.
构造函数只能通过初始化列表("冒号"加"括号")初始化之

8. 讨论
对象数组: 类名 数组名[下标表达式]
访问: 数组名[下标].成员名.

对象指针: 用指向对象地址的变量(只有数据成员,没有函数成员时)
类名 * 对象指针名
访问: 对象指针名 -> 成员名

this 指针: 指向当前被成员函数操作的对象

五. 动态内存分配
{new 类型名(初值列表) :返回指针
delete 指针名.
当然,类型名包括类的名(类类型).此时指针就是对象指针.

NO.

Date . .

多维数组(动态): new 类型名 [下标1][下标2]...

n个

类型名 (*P) [下标2]...

n-1个.

用来指向此类型的多维数组.

delete []p.

[关于析构函数]: 对象

浅拷贝: 两数组共用一内存区. 导致delete时在同一内存上释放了两次

深拷贝: 创建一块新内存区, 两次释放不同内存区, 要自己写!

六. 继承与派生.

1. 声明: class 派生类名: 继承方式 基类名1, 继承方式 基类名2 ....

{ ... };

{ 多继承
  单继承.

2. 规则 { 继承了除构造和析构函数之外的所有成员(包括 copy constructor)
派生类的新成员可同名隐藏规则(内层优先).
可加入新的构造和新析构函数.

NO.

Date

类型兼容规则：

凡是基类能解决的问题，公有派生类都可解决

- 派生类的对象 可赋值给基类对象
- 派生类的对象 可初始化 基类的引用. → 在构造函数(派生类中)起作用.
- 派生类的对象地址 可赋给基类的指针.

派生类的构造和析构函数.

派生类的默认构造函数只对派生类新增数据成员初始化.

若欲对从基类继承下的成员也初始化，则需再编写构造函数.

析构函数也如此.

① 构造顺序：调用基类的构造函数. (左→右)

↓

调用内嵌对象的构造函数 (左→右). 只是基类的数据成员

↓

派生类新成员的构造.

派生类名::派生类名(参数总表):基类名1(参数表1)...

基类名n(参数表n), 内嵌对象名1(内嵌对象参数表1), ...,

内嵌对象名m(内嵌对象参数表m).

{ 派生类新增成员的初始化语句 }

② 拷贝构造函数：C::C(C &c1):B(c1)

基类的引用　派生类的引用.　(类型兼容性规则).

NO.

Date

③ 析构函数

顺序：

派生类

↓

内嵌对象成员

↓

基类

七. 多态性

静态联编 { 强制多态：类型转换（四类）
重载多态：函数重载、运算符重载
参数多态：模板 } 编译时

动态联编：包含多态：虚函数 —— 运行时

1. 运算符重载：返回类型 operator 运算符 (形参)
"函数名"

作为 { 成员函数：少一形参，另一个由 this 指针给出。
友员函数：多形参数同，在类名声明友元函数。 } 两种方式

2. 虚函数

不是非静态成员函数

virtual 函数类型 函数名(形参)
{ ... }

NO.

Date . .

1) 派生类的虚函数覆盖了基类的虚函数，隐藏基类中同名函数的所有其他重载形式.

2) 虚函数是动态联编的. 但默认形参值是静态联编的.

3) 虚析构函数 virtual ~类名().

派生类虚函数 (体现基类的行为)

派生类非虚函数 (不希望被派生类改变功能)

通过使用基类型指针可访问到该指针正在指向的派生类的同名函数.

3. ① 函数模板:

```
template <typename T>
类型名 函数名(参数表)
{...}
```

T 表示一种抽象的类型. 在调用时要具体化.

② 类模板:

```
template <typename T>
class 类名
{...};
```

1) 在类外定义其成员函数:

```
template <typename T>
类型名 类名<T>::函数名(参数表)
{...}
```

NO.

Date

2) 模板类的成员函数必须是函数模板。

3) 建立对象的声明： 类模板 <模板参数表(具体)> 对象名1, …… ;

八. 流类库与输入/输出

ifstream　istringstream

↘ ↙

istream

键盘、屏幕、打印机、通信端口、磁盘文件 ……	读：提取 →	内存
	← 写：插入	

ostream

↙ ↘

ofstream　ostringstream

类模板： basic_xxxx. ∈ <iosfwd>

1. <istream> 输出流： ostream(基)
<fstream>
↓
ofstream

- cout、cerr、clog.
- open、close
- put write
- seekp tellp.
- fail、eof、clear、bad…

} 对象(成员函数)

① cout << … << …

fixed

scientific

~~showpos~~

left

right dec. oct. hex.

→ ios_base

<iomanip> fill. setw. width. setiosflags. resetiosflags.

② open("filename", iosmode).

↓

∈ ios_base.

app. ate. in out trunc binary ☆

③ put('A').

④ close() 是成员函数，需用对象去访问.

⑤. write((char*)&x, int size).
（&x：地址；int size：字节数.）

⑥ seekp(int).

⑦. eof. (返回非0.) 未遇文件尾返回0.
(内存已到尾).

......

---

2. <istream> 输入流: istream (基) → cin.

<fstream>

istream ↓ ifstream. { open. close. get getline. read. good. seekg. tellg. } 对象(成员函数)

① cin >> ...... >>.

{ ios_base::in (默认)
ios_base::binary

② get 也读空白字符.

③ getline(地址, 读的数目, 终止符)
└ 读完会删.

④ close()

⑤ read((char*)地址, 字节数).

⑥ seekg.(int).

(文件的终止符)

⑦ good() (返回非0) 未读到文件尾返回0.

N0.

Date . .

九、异常处理：

try {……throw 表达式……

……throw 表达式……}

catch (异常类型声明) {处理办法}

catch (异常类型声明) {处理办法}

⋮

catch (……) {处理所有异常}

一旦 catch 捕获到一个异常则处理，处后跳出 try 块执行 next 语

在函数的声明中可列出可能抛掷的所有异常类型

void fun() Throw (A, B, C, D)

若不指，则可抛出任何类型异常

void fun()

若如此 void fun() throw(); 则不抛任何类型