C++重点、易错知识点整理

第一章

1、泛型程序设计是指向程序中数据类型中加入类型参数的一种能力，也称为参数化的类型或参数多态性。

2、c++程序开发通常要经过 5 个阶段，包括编辑、预处理、编译、连接、运行与调试。

3、编译过程分为词法分析、语法分析、代码生成这 3 个步骤。

4、使用名字空间 std 的方法有 3 种：

1、利用 using namespace 使用名字空间；使用方法如下：

```
using namespace std;
```

2、用域分辨符::为对象分别指定名字空间；例如：

```
std::cout<<"Hello!"<<std::endl;
```

3、用 using 与域分辨符指定名字空间；例如：

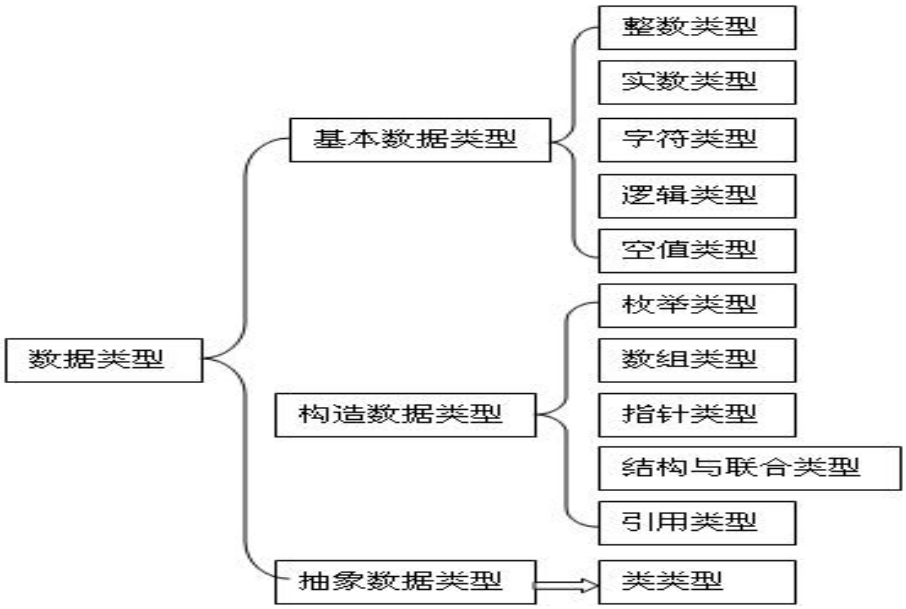
```
using std::cout;
```

5、c++中常用操作符：

操纵符	作用	说明
oct	数据以八进制形式输出	作用范围为后续输出的整数对象，小数不起作用
dec	数据以十进制形式输出(默认)	
hex	数据以十六进制形式输出	
endl	换行并刷新输出流	
setw(n)	设置输出宽度	需包含头文件 iomanip，仅对后一个对象起作用
setprecision(n)	设置输出小数位数（默认为 6）	需包含头文件 iomanip，作用范围为后续对象

第二章

1、c++的数据类型：



2、在定义变量的同时赋初值还有另外一种方法，就是在变量后面将初值放在括号中，格式如下：数据类型 变量名 1（初值 1）,变量名 2（初值 2）,…,变量名 n（初值 n）;

3、常变量定意格式：const 数据类型 符号常量名 = 数值;

或数据类型 const 符号常量名 = 数值;

※在定义常变量时，一定要赋初值，且在程序中间不能更新其值。

4、常量和非左值表达式是没有内存地址的。

5、在逻辑表达式求值中注意短路求值。

6、运算符优先级的规律：

（1）运算符的优先级按单目、双目、三目、赋值依次降低；

（2）算术、移位、关系、按位、逻辑运算的优先级依次降低。

7、标准 c++提供了新式的强制类型转换运算，格式如下：

（1）static_cast<类型说明符>(表达式);
（2）reinterpret_cast<类型说明符>(表达式);
（3）const_cast<类型说明符>(表达式);
（4）dynamic_cast<类型说明符>(表达式);

※static_cast 用于一般表达式的类型转换；

※reinterpret_cast 用于非标准的指针数据类型转换，如将 void*转换成 char*；

※const_cast 将 const 表达式转换成非常量类型，常用于将限制 const 成员函数的 const 定义解除；

※dynamic_cast 用于进行对象指针的类型转换。

第三章

第四章

1、内联函数的定义必须出现在对该函数的调用之前。

2、递归函数不能定义为内联函数。

3、说明一个内联函数只是请求而不是命令编译器对它进行扩展。

带有默认形参值的函数：

1、若函数具有多个形参，则默认形参值必须自右向左连续的定义，并且在一个默认形参值的右边不能有未指定默认值的参数。

2、在调用一个函数时，若果省去了某个实参，则直到最右端的实参都要省去。

3、默认形参值的说明必须出现在函数调用之前。若函数原型中已给出了形参的默认值，则在函数定义中不得重复制定，即使所指定的默认值完全相同也不行。

4、在同一个作用域内，一旦定义了默认形参值，就不能在定义它。

5、如果几个函数说明出现在不同的作用域内，则允许对它们提供不同的默认形参值。

6、在函数的原型给出了形参的默认值时，形参名可以省略。

第五章

1、相同类型的指针类型才可以想减；两个指针是不可以相加的。

2、一个 void 类型的地址赋值给非 void 类型的指针变量，要使用类型强制转换。

3、要初始化多重指针，要从第一层开始，逐步向高层进行。

4、要将字符串 q 复制给 p，除了使用与复制字符数组相同的三种方法外，还可用程序段：

```
while(*q)
    *p++=*q++;
    *p='\0';
```

5、**new 的语法形式**：指针变量=new 类型名(初值列表;)

6、**new 创建数组的语法格式**：指针变量=new 类型名[下表表达式];

※下表表达式与数组初始化时的常量表达式不同，可以是变量表达式。

7、**使用 new 建立多重数组语法格式**：指针变量=new 类型名[下表表达式 1][下标表达式 2][…];

※再用 new 建立多维数组时，只有下标表达式 1 可以是任意正整数的表达式，而其它下标表达式必须是值为正整数的常量表达式。

8、**delete 语法格式**：delete 指针变量名;

9、**delete 删除动态数组格式**：delete [ ]指针变量名;

※[]中不需要说明对象的个数。

※对于一个已分配内存的指针，只能用 delete 释放一次。

10、**指针常量定义格式**：数据类型 *const 指针名=变量名;

11、**常量指针定义格式**：

```
const 数据类型 *指针名=变量名;
或
数据类型 const *指针名=变量名;
```

12、指针常量：指向的对象不能更改，对象的值能改。

13、常量指针：指向的对象能更改，对象的值不能更改。

14、在定义结构体类型时，只要结构体名不同，那么都是不同的类型，即使它们的成员完全相同。

第六章

1、根据变量**定义的位置**，变量分为**全局变量**与**局部变量**。

2、根据**变量的存储方法**，变量分为**静态变量**和**动态变量**，具体分为 4 种：自动型（auto）、寄存器型（register）、外部型（extern）和静态型（static）。

※使用 extern 只是将一个已存在的变量、函数声明为 extern 变量函数，而不是定义变量，所以不能赋初值。

※当一个变量被声明为全局静态变量时，不能再被声明为 extern 变量。

3、**c++程序的内存通常被分为 4 个区：**

（1）**全局数据区**（全局变量、静态变量、字符串变量和常变量）

（2）**代码区**（所有的函数和代码）

（3）**栈区**（为运行函数而分配的函数参数、局部变量和返回地址）

（4）**堆区**（动态分配内存）

4、宏定义可以嵌套定义、被重复定义，但不能递归定义。

5、定义名字空间的格式如下：namespace 名称 {成员};

第七章

1. 在类体内不允许对成员函数进行初始化。
2. 定义类时，类的数据成员不占内存空间；但是，建立类的对象时，**只为**每个对象分配用于保存**数据成员**的内存，不为函数成员分配内存。
3. 在定义类时，不能定义该类的变量，只能定义该类类型的指针成员和该类类型的引用成员。
4. 浅拷贝和深拷贝的区别：当类的数据成员是指针类型时，深拷贝能为新的对象分配内存空间（分配内存空间一般由 new 运算符实现；拷贝构造函数就是深拷贝），而浅拷贝不能。
5. 对象指针只能访问该类的公有数据成员和函数成员。
6. 除非是作为函数参数与函数返回值，对象引用在定义时必须要初始化。
7. 非静态函数有一个指向当前对象的 this 指针，而静态函数没有。静态函数不属于任何类。
8. 在组合类的构造函数中，初始化列表既不能决定是否调用成员对象的构造函数，也不能决定调用构造函数的顺序，成员对象调用顺序由成员对象定义的顺序决定。
9. 静态变量和静态对象都只被构造一次。
10. 类的静态数据成员必须进行初始化，且其初始化语句既不属于任何类，也不属于包括主函数在内的任何函数。
11. 静态变量的初值缺省时为 0,；动态变量的缺省初值不确定。
12. 静态成员函数可以直接访问类中说明的静态成员，但不能直接访问类中的非静态成员。
13. 静态数据成员不是对象成员，在引用时不需要用对象名。（？为什么？）
14. 不允许常对象调用任何类的成员函数，而且常对象一旦被定义，在其生存期内不允许改变。
15. 只有类的常成员函数才能访问该类的常对象；const 对象不能访问非常成员函数。
16. 常成员函数必须进行初始化，且初始化只能通过构造函数的初始化列表进行。
17. **常成员函数的定义格式**：返回类型　成员函数名（参数表）const;
18. 常成员函数不能更新类的数据成员。
19. 保护成员具有双重角色，对派生类的成员函数而言，它是共有成员，但对所在类之外定义的其它函数而言则是私有成员。
20. 使用 this 指针返回对象则不需要调用构造函数，但会调用拷贝构造函数。
21. 先建立全局变量，在建立局部变量。

构造函数的调用条件：

1. 建立对象时；
2. 建立动态对象（如 p=new char）时；

※定义对象指针、对象引用时均没有建立对象，所以此时不调用构造函数。

※数组元素为 n，定义数组时调用构造函数的次数为 n。

※一个对象可以建立多个构造函数，但是只有一个析构函数。

※构造函数没有返回值，也不能用 void 修饰。

※构造函数被声明定义为公有函数。

拷贝构造函数的调用条件：

1. 当用类的一个对象去初始化该类的另一个对象时；
2. 如果函数的形参是类的对象，调用函数时，将对象作为函数实参传递给函数的形参；
3. 如果函数的返回值是类的对象，函数执行完成，将返回值返回。

4. 使用 this 指针返回对象。

※拷贝构造函数只能有一个参数，并且是对某个对象的引用。

※建立对象时，构造函数和拷贝构造函数有且仅有一个被调用。

※再重新定义拷贝构造函数后，默认拷贝构造函数与默认构造函数就不存在了；但是，再重新定义构造函数后，默认构造函数就不存在了，但默认拷贝构造函数还存在。

析构函数的调用条件：析构函数在程序结束时由系统自动调用。

组合对象的构造函数的调用顺序：在定义一个组合类的对象时，不仅他自身的构造函数将被调用，而且还将调用其成员函数的构造函数，调用**先后顺序**为：

1. 成员对象按照在其组合类的声明中出现的次序，依次调用各自的构造函数，而不是按初始化列表中的顺序。
2. 组合类对象调用组合类构造函数。
3. 调用析构函数，析构函数的调用顺序与构造函数正好相反。

构造、拷贝以及析构函数的调用顺序：对象被析构的顺序与其创建的顺序相反。

第八章

1、如果几类定义了带形参表的构造函数时，派生类就应当定义构造函数。如果积累没有定义构造函数，派生类也可以不定义构造函数。

2、派生类的构造函数和析构函数都不能被继承。

3、多继承时，对同一个基类，不允许直接继承两次。

4、**最远派生类**（c++将建立对象时所使用的派生类称为最远派生类）的构造函数要调用该公共基类的构造函数，而且只能被调用一次。

5、**公共虚基类子对象只初始化一次。**

6、c++规定，在初始化列表中同时出现对虚基类和非虚基类构造函数的调用，虚基类的构造函数优于非虚基类的构造函数的执行。

类型兼容的三种情况：

1、派生类对象可以赋值给基类对象；

2、派生类对象可以初始化积累的引用；

3、派生类的地址可以赋值给指向基类的指针。

单继承或多继承时，派生类构造函数的调用顺序：

1、调用基类的构造函数；

2、调用内嵌成员对象的构造函数调用顺序按照它们在类中定义的顺序；

3、派生类自己的构造函数。

单继承时，派生类的析构构造函数的调用顺序：

1、调用派生类析构函数；

2、然后调用派生类成员对象析构函数；

3、最后调用基类析构函数。

第九章

1、c++中的运算符除了“①.②.* ③::④?:⑤ siziof”之外，其余全部可以被重载。

2、对于运算符重载为类的友元函数，VC++6.0 不允许在声明重载运算符之前使用 using namespace std，多以分别列出对 cout 和 endl 的使用：using std::cout;using std::endl;。

3、**重载为类的成员函数语法形式为**：

返回类型 类名::operator 运算符（形参表）
{函数体;
}

4、**重载为类的有原函数的定义格式：**

Friend 函数类型 operator 运算符（形参表）
{函数体;
}

5、无法声明一个抽象类的对象。

抽象类：

1、 带有纯虚构函数的类被称为抽象类。纯虚构函数的定义形式：
virtual 函数类型 函数名（参数表）=0;
2、 抽象类不能用作参数类型、函数返回值或强制类型转换；
3、 可以定义一个抽象类的指针和引用。通过抽象类的指针和引用，可以指向并访问各派生类成员，这种访问具有多态特征。

成员函数运算符与友元函数运算符的特点：

1、 一般情况下，单目运算符最好重载为类的成员函数；双目运算符最好重载为类的友元函数。
2、 一些双目运算符不能重载为类的友元函数：=、()、[]、->。
3、 类型转换函数只能定义为一个类的成员函数而不能定义为类的友元函数。
4、 如果一个运算符的操作需要对象的状态，选择重载为成员函数较好。
5、 若运算符所需的操作数（尤其是第一个操作数）希望有隐式转换，则只能选用友元函数。
6、 当运算符函数是一个成员函数时，最左边的操作数（或者只是最左边的操作数）必须是运算符类的一个类对象（后者是对该类对象的引用）。如果左边的操作数必须是一个不同的对象，或者是一个基本数据类型的对象，该运算符函数必须作为一个友元函数来实现。
7、 当需要重在运算符的运算具有可交换性时，选择重载为友元函数。

运算符被重载的规则：

1、 重载后运算符的优先级与结合性不会改变。
2、 不能改变原运算符操作数的个数。
3、 不能冲在 c++中没有的运算符。
4、 不能改变运算符的原有语义。

类的成员函数的运算符重载规则：

① 双目运算符重载为类的成员函数时，函数只显式说明一个参数，该形参是运算符的右操作数。

② 前置单目运算符重载为类的成员函数时，不需要显式说明参数，即函数没有形参。

③ 后置单目运算符重载为类的成员函数时，为了与前置单目运算符区别，函数要带有一个整型形参。

类的友元函数的运算符重载规则：

①双目运算符 B 重载后，表达式 oprd1 B oprd2 等同于 operator B(oprd1,oprd2)；
②前置单目运算符 B 重载后，表达式 B oprd 等同于 operator B(oprd) ；
③后置单目运算符 ++和--重载后，表达式 oprd B 等同于 operator B(oprd,int)

虚函数：

1、 虚函数不能使静态函数，也不能是友元函数。
2、 内联函数是不能在运行中动态确定其位置的，即使虚函数在类的内部定义，编译时，仍

将其看做非内联的。

3、 只有类的成员函数才能说明为类的虚函数；虚函数仅适合于有继承关系的类对象。

4、 构造函数不能是虚函数，析构函数可以是虚函数，而且通常声明为虚函数。

※动态联编只能通过指针或引用标志对象来操作函数。如果采用一般类型的标志对象来操作虚函数，则将采用静态联编方式调用虚函数。

第十章

类模板定义语法：

```
template<模板参数表>
class 类名
{成员名；
};
```

模板类的成员函数在类外定义的语法格式：

```
template<模板参数表>
类型 类名<模板参数名表>::函数名(参数表)
{函数体;
};
```

类模板实例化、建立对象的语法形式： 类模板名<模板参数值表>对象 1,对象 2,…,对象 n;

※类模板的类型参数也可以采用默认值。带默认模板参数值的类模板的默认值给出顺序为从右向左，实参值结合顺序为从左向右。

函数模板的定义形式：

```
template<class 类型名 1,class 类型名 2,…>
返回类型 函数名(形参表)
{函数体;
}
```

函数显示实例化的格式： 函数名<具体函数名 1, 具体函数名 2,…,常量表达式>(实参表)

函数重载

类型兼容