

C++程序设计

主讲：王老师



尚德机构

学习是一种信仰

教材介绍

C++程序设计

(2008年版)

编著：刘振安

机械工业出版社



考试题型

单选题 $1\text{分} \times 20\text{题} = 20\text{分}$

填空题 $1\text{分} \times 20\text{题} = 20\text{分}$

程序改错题 $4\text{分} \times 5\text{题} = 20\text{分}$

完成程序题 $4\text{分} \times 5\text{题} = 20\text{分}$

程序分析题 $5\text{分} \times 2\text{题} = 10\text{分}$

程序设计题 $10\text{分} \times 1\text{题} = 10\text{分}$



第二章 从结构到类的演变





本章主要内容



- 结构的演化
- 从结构演变一个简单的类
- 面向过程与面向对象
- C++面向对象程序设计的特点
- 使用类和对象
- string对象数组与泛型算法



§2.1 结构的演化

一、结构发生质的演变

1、函数与数据共存

C + + 中首先允许结构中可以定义函数，这些函数称为**成员函数**，形式如下：

```
struct 结构名{
```

```
    数据成员
```

```
    成员函数
```

```
};
```

可以像C语言中结构变量使用结构成员的方法那样，通过C + + 的结构对象使用成员函数，形式：**结构对象.成员函数**



§2.1 结构的演化

2、封装性

如果在定义**结构**时，将数据成员使用private关键字定义，则产生封装性，**没有使用private定义的成员函数，默认为public。**

要注意，**私有的数据成员，必须通过公有的成员函数才能使用，而不能不通过公有的成员函数直接来使用，否则就会出错，这就称为数据的封装性。**



§2.1 结构的演化

二、使用构造函数初始化结构的对象

函数名与结构同名，称为构造函数，专门用于初始化结构的对象。

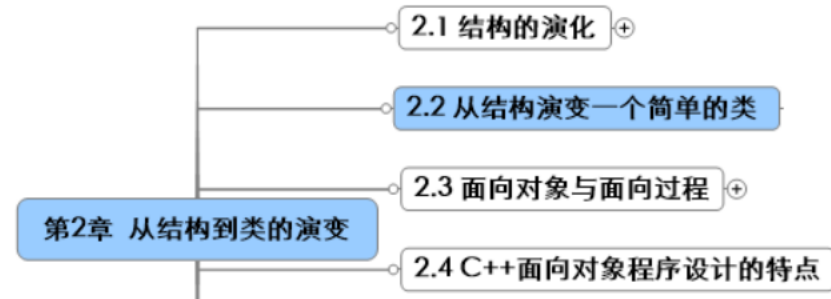
构造函数使用的一般形式为：

构造函数名 对象名（初始化参数）；

程序在运行时，会自动完成初始化任务。



§2.2 从结构演变一个简单的类



1、用关键字**class**代替struct，就是一个标准的类。

2、类的示意图

上例中的Point类可以看作直角坐标系中的点类，其结构示意图如右：

第一个方框中是类名；第二个方框中是坐标点的数据，称为属性（或称数据成员）；第三个方框中表示类所提供的具体操作方法，实际上是如何使用数据x和y，以实现预定功能的函数，这里称为成员函数。

类名Point
具有的属性x和y
提供的操作 构造函数Point Setxy用来给对象赋值 Display用来输出x和y



```
#include <iostream>
using namespace std;

class Point
{    //定义类Point
private:
    double x,y; //类Point的数据成员
public:
    Point( ){ }; //类Point的无参数构造函数
    Point(double a,double b) {x=a;y=b;} //具有两个参数的构造函数
    void Setxy(double a,double b) {x=a;y=b;} //成员函数，用于重新设置数据成员
    void Display( ){cout<<x<<"\t"<<y<<endl;} //成员函数，按指定格式输出数据成员
};

void main( )
{
    Point a; //定义类Point的对象a
    Point b(18.5,10.6); //定义类Point的对象b并初始化
    a.Setxy(10.6,18.5); //为对象a的数据成员赋值
    a.Display(); //显示对象a的数据成员
    b.Display(); //显示对象b的数据成员
}
```

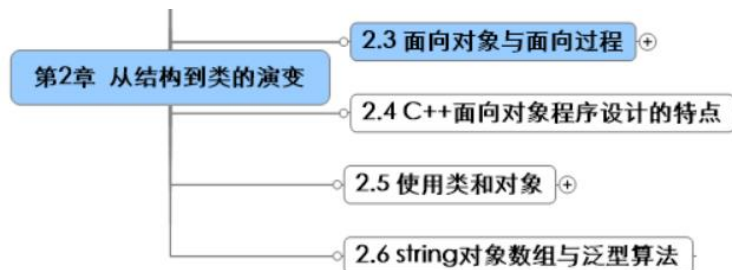
§2.3 面向过程与面向对象

1、面向过程的方法

所谓面向过程，就是不必了解计算机的内部逻辑，而把精力集中在对如何求解问题的算法逻辑和过程的描述上，通过编写程序把解决问题的步骤告诉计算机。

C语言就是面向过程的结构化程序设计语言，其程序设计特点就是通过函数设计，实现程序功能的模块化、结构化。但实际工作中，尽管结构化程序设计中的分而治之的想法非常好，但在结构化程序设计语言和结构化程序设计方法下却难以贯彻到底，特别是在软件规模在三四万行以上时，开发和维护就十分困难。

2、面向对象的方法



- 为了解决面向过程的方法存在的问题，人们提出了面向对象的方法。**所谓对象，就是现实世界中客观存在的事物。**相对于过程，对象是稳定的，复杂的对象可由简单的对象组成，对象各自完成特定的功能。
- 在面向对象程序设计中，可以将一组密切相关的函数统一封装在一个对象中，从而合理而又有效的避免全局变量的使用，更彻底的实现了结构化程序设计的思想。
- **结构化程序设计使用的是功能抽象，而面向对象程序设计不仅能进行功能抽象，还能进行数据抽象，**“对象”实际上是功能抽象和数据抽象的统一。
- 面向对象的程序设计方法不是以函数过程和数据结构为中心，而是以**对象**代表来求解问题的中心环节，追求的是现实问题空间与软件系统解空间的近似和直接模拟，从而使人们对复杂系统的认识过程与系统的程序设计实现过程尽可能的一致。



软件开发过程及发展趋势

软件开发开发者将被开发的整个业务范围称为**问题域**，软件开发是对给定问题求解的过程，分为两项主要内容：**认识**和**描述**。“认识”就是在所要处理的问题域范围内，通过人的思维，对该问题域客观存在的事务以及对所要解决的问题产生正确的认识和理解。“描述”就是用一种语言把人们对问题域中事务的认识、对问题及解决方法的认识描述出来，最终的描述必须使用一种能够被机器读懂的语言，即编程语言。

人们习惯使用的自然语言和计算机能够理解并执行的编程语言之间存在很大的差距，称为“语言的鸿沟”。由于人的认识差距，问题域和自然语言之间也有缝隙，机器语言和自然语言之间鸿沟最宽，程序设计语言的发展趋势则是为了鸿沟变窄。



§2.4 C++ 面向对象程序设计的特点

2.4.1 对象 C++可使用**对象名、属性和操作**三要素描述

2.4.2 抽象和类 将一组对象的共同特征进一步**抽象**出来，形成类

2.4.3 封装：将对象的属性和操作结合成一个独立的单位

2.4.4 继承：一个类可以获得另一个类的特性的机制

2.4.5 多态性：相同名称的函数，完全不同的行为

面向对象的程序设计具有**抽象、封装、继承和多态性**等关键要素。

1、对象

C++ 中的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，C++ 中使用**对象名、属性和操作**三要素来描述对象，如右所示：对象名用来标识一个具体对象；用数据来表示对象的属性，一个属性就是描述对象静态特征的一个数据项；操作是描述对象动态特征（行为）的一个函数序列（使用函数实现操作），也称方法或服务。数据称为数据成员，函数称为成员函数，**一个对象由一组属性和对这组属性进行操作的成员函数构成。**

对象名
属性1
属性2
.....
属性n
操作1
操作2
.....
操作n





对象举例

2.4.1 对象 C++可使用**对象名、属性和操作**三要素描述

2.4.2 抽象和类 将一组对象的共同特征进一步**抽象**出来，形成类

2.4 C++面向对象程序设计的特点

2.4.3 **封装**：将对象的属性和操作结合成一个独立的单位

2.4.4 **继承**：一个类可以获得另一个类的特性的机制

2.4.5 **多态性**：相同名称的函数，完全不同的行为

例：用简单对象表示平面上的A (3.5, 6.4) 和B (8.5, 8.9) 两个坐标点。

可用如下的对象结构图表示A和B的对象结构：

A
x (3.5)
y (6.4)
Display () ;
Setxy () ;
Move () ;

B
x (8.5)
y (8.9)
Display () ;
Setxy () ;
Move () ;



抽象和类

2.4 C++面向对象程序设计的特点

2.4.1 对象 C++可使用**对象名**、**属性**和**操作**三要素描述

2.4.2 抽象和类 将一组对象的共同特征进一步**抽象**出来，形成类

2.4.3 **封装**：将对象的属性和操作结合成一个独立的单位

2.4.4 **继承**：一个类可以获得另一个类的特性的机制

2.4.5 **多态性**：相同名称的函数，完全不同的行为

抽象是一种从一般的观点看待事物的方法，即集中于事物的本质特征，而不是具体细节或具体实现。面向对象的方法把程序看作是由一组抽象的对象组成的，如果把一组对象的共同特征进一步抽象出来，就形成了“类”的概念。

对于一个具体的类，它有许多具体的个体，这些个体叫做“**对象**”，同一类的不同对象具有相同的行为方式。**一个对象是由一些属性和操作构成的**，对象的属性和操作描述了对对象的内部细节，**类是具有相同的属性和操作的一组对象的集合**，它为属于该类的全部对象提供了统一的抽象描述，其内部包括**属性和操作**两个主要部分。



抽象和类

2.4 C++面向对象程序设计的特点

2.4.1 对象 C++可使用**对象名**、**属性**和**操作**三要素描述

2.4.2 抽象和类 将一组对象的共同特征进一步**抽象**出来，形成类

2.4.3 **封装**：将对象的属性和操作结合成一个独立的单位

2.4.4 **继承**：一个类可以获得另一个类的特性的机制

2.4.5 **多态性**：相同名称的函数，完全不同的行为

- 类的作用是定义对象，类和对象的关系如同一个模具和其铸造出来的铸造件的关系，对象之间就像是同一模具铸出的零件，模样相同，铸造材料可能不同。
- 类给出了属于该类的全部对象的抽象定义，而对象则是符合这种定义的实体。
- 所谓一个类的所有对象具有相同属性和操作，是指它们的定义形式（即属性的个数、名称、数据类型）相同，而不是说每个对象的属性值都相同。



封装

为了保护类的安全，即限制使用类的属性和操作，需要将类封装起来。封装就像用同一个模具铸造零件，各零件使用的材料（数据成员）和铸造工艺（成员函数）均不同，每一种材料都有其对应的铸造工艺，材料与铸造工艺是成套使用（封装）的，虽然铸出零件的模样一样，但如果用错了铸造工艺，就可能出废品。

所谓“封装”，就是把对象的属性和操作结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。按照面向对象的封装原则，一个对象的属性和操作是紧密结合的，对象的属性只能由这个对象的操作来存取。



封装

对象的操作分为内部操作和外部操作，前者只供对象内部的其他操作使用，不对外提供；后者对外提供一个信息接口，通过这个接口接受对象外部的消息并为之提供操作（服务）。对象内部数据结构的这种不可访问性称为信息（数据）隐藏。数据封装给数据提供了与外界联讯的标准接口，只有通过这些接口，使用规范的方式，才能访问这些数据，同时程序员也只需要和接口打交道，而不必了解数据的具体细节。



封装

封装要求一个对象应具备明确的功能，并具有接口以便和其他对象相互作用，同时，对象的内部实现（代码和数据）是受保护的，外界不能访问它们，这样封装一方面使得程序员在设计程序时能专注于自己的对象，同时也切断了不同模块之间数据的非法使用，减少了出错。

在类中，封装是通过存取权限实现的，例如每个类的属性和操作分为私有和公有两种类型，对象的外部职能访问对象的公有部分，而不能直接访问对象的私有部分。



当类中有数据成员的访问权限为私有时，不允许对对象进行初始化。

```
class A {  
    float x,y;  
public:  
    void Setxy( float a, float b ){ x=a; y=b; }  
    void Print(void) { cout<<x<<'\t'<<y<<endl; }  
} a1,a2;
```

定义全局对象

```
void main(void)  
{  
    A a3,a4;
```

定义局部对象

```
}
```

对象的使用：

一个对象的成员就是该对象的类所定义的成员，
有成员数据和成员函数，引用时同结构体变量类似，用“.”运算符。


```

class A {
    float x,y;
public:
    float m,n;
    void Setxy( float a, float b ){ x=a; y=b; }
    void Print(void) { cout<<x<<'\\t'<<y<<endl; }
};

void main(void)
{
    A a1,a2; //定义对象
    a1.m=10; a1.n=20; //为公有成员数据赋值
    a1.Setxy(2.0 , 5.0); //为私有成员数据赋值
    a1.Print();
}

```

输出： 2 5

a1	
x	2.0
y	5.0
m	10
n	20
	Setxy()
	Print()

a2	
x	
y	
m	
n	
	Setxy()
	Print()

用成员选择运算符“.”只能访问对象的公有成员，而不能访问对象的私有成员或保护成员。若要访问对象的私有的数据成员，只能通过对象的公有成员函数来获取。



```
class A {  
    float x,y;  
public:  
    float m,n;  
    void Setxy( float a, float b ){ x=a; y=b; }  
    void Print(void) { cout<<x<<'\\t'<<y<<endl;}  
};
```

必须通过类内公有函数访问私有数据成员

```
void main(void)  
{  
    A a1,a2;  
    a1.m=10; a1.n=20; //为公有成员数据赋值  
    a1.x=2; a1.y=5;  
    a1.Setxy(2.0,5.0);  
    a1.Print();  
}
```

非法，私有成员不能在类外访问

同类型的对象之间可以整体赋值，这种赋值与对象的成员的访问权限无关。

```
class A {  
    float x,y;  
public:  
    float m,n;  
    void Setxy( float a, float b ){ x=a; y=b; }  
    void Print(void) { cout<<x<<'\t'<<y<<endl; }  
};  
  
void main(void)  
{  
    A a1,a2;  
    a1.m=10; a1.n=20;    //为公有成员数据赋值  
    a1.Setxy(2.0,5.0);  
    a2=a1;                  
    a1.Print(); a2.Print();  
}
```

类作用域、类类型的作用域和对象的作用域

类体的区域称为类作用域。类的成员函数与成员数据，其作用域都是属于类的作用域，仅在该类的范围内有效，故不能在主函数中直接通过函数名和成员名来调用函数。

```
class A {  
    float x,y;  
public:  
    float m,n;  
    void Setxy( float a, float b ){ x=a; y=b; }  
    void Print(void) { cout<<x<<'\t'<<y<<endl; }  
};
```

```
void main(void)
```

```
{ A a1,a2;  
  a1.m=20; a1.n=10;  
  a1.Setxy(2.0, 5.0);  
  a1.Print();  
}
```

用对象名调用

```
void main(void)
```

```
{ A a1,a2;  
  m=20; n=10;  
  Setxy(2.0, 5.0);  
  Print();  
}
```

不能直接调用

继承

2.4 C++面向对象程序设计的特点

2.4.1 对象 C++可使用**对象名、属性和操作**三要素描述

2.4.2 抽象和类 将一组对象的共同特征进一步**抽象**出来，形成类

2.4.3 **封装**：将对象的属性和操作结合成一个独立的单位

2.4.4 **继承**：一个类可以获得另一个类的特性的机制

2.4.5 **多态性**：相同名称的函数，完全不同的行为

继承是一个类可以获得另一个类的特性的机制，支持**层次概念**，通过继承，低层的类只需定义特定于它的特征，而共享高层的类中的特征。继承具有重要的实际意义，它简化了人们对事物的认识和描述。

继承就像铸造中母模与子模的关系。



多态性

不同的对象可以调用相同名称的函数，但可导致完全不同的行为的现象称为多态性。

利用多态性，程序中只需进行一般形式的函数调用，函数的实现细节留给接受函数调用的对象，这大大提高了解决人们复杂问题的能力。

多态性就像是铸造时不同的零件、不同材料所铸的同一款零件虽然可以使用相同的铸造工艺，但铸出的零件用途、使用寿命和使用方法都不一样。



```
#include <iostream>
using namespace std;
```

```
int max(int,int); //2个整形参数的函数原型
int max(int,int,int); //3个整形参数的函数原型
```

```
void main( )
{
    cout<<max(35,25)<<endl<<max(25,39,35)<<endl;
}
```

```
int max(int m1,int m2)
{
    return(m1>m2)?m1:m2;
}
```

```
int max(int m1,int m2,int m3)
{
    int t=max(m1,m2);
    return max(t,m3);
}
```

执行结果：
35
39

§2.5 使用类和对象

1、使用string对象

实际上string类很复杂，如右的string类的简化图中只给出了下例中涉及的属性和部分操作。

由图，类string的属性是一个字符串str，同名函数string是构造函数，用来初始化字符串，另外三个成员函数用来对属性str进行操作，其中find成员函数用来在str字符串中检索需要的子串；size成员函数计算并输出str存储的单词长度；substr成员函数用来返回str字符串中的子串。

string

str

string

find

size

substr



使用string对象

string

str

string

find

size

substr

在程序中可以使用string类定义存储字符串的对象，这些对象属于string类，因此还要使用`#include <string>`来包含这个类的头文件。

因为string需要的是字符串，所以string类的对象不能用单引号括起来的单个字符常量初始化，即：

```
string str = ' A' ; × //同理string str(' A' ); ×
```

但可以使用双引号括起来的单个字符常量初始化，即：

```
string str = "A" ; √ //同理string str( "A" ); √
```

如果string的对象str的内容为“ab”，则str[0]=' a' ， str[1]=' b' 。



使用string对象

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例：

2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

string

str

string

find

size

substr

例：使用string对象及初始化

程序中使用了两种方法给string类的两个对象初始化，也可将它们定义在一行中：`string str1("We are here!");` `string str2="Where are you?";`
string类还提供将两个字符串连接起来组成一个新字符串的能力，用 `" + "` 号将后面的字符串连接到前一个字符串的后面，也可以与单个字符常量连接，如：

`str1 = str1 + " " + str1;` 将原来的两个str1的内容用空格连接起来。



使用string类的典型成员函数

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例:

2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

string对象是通过调用成员函数实现操作，从而提供对象的行为或消息传递的，对象调用成员函数的语法为：

对象名称.成员函数（参数（可供选择的消息内容））

常用的string类成员函数：

```
substr:    string newstr = str1.substr(3,3);  
find:      int i = str1.find( "are" ,0);  
getline:   getline (cin, InputLine, ' \n' ) ;  
swap:      str1.swap(str2); = str2.swap(str1);  
begin和end: copy(str1.begin( ),str1.end( ),str2.begin( ));
```



使用string类的典型成员函数

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例:

2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

①成员函数**substr**用来返回给定字符串的子串，格式为：

对象名称.substr（要截取子串的起始位置，截取的长度）；

如：string str1("We are here!"); 语句中要从对象str1中截取字符串are，可用下面的语句实现：

```
string newstr = str1.substr(3,3);
```

此时newstr的内容为are，括号中的第一个3是因为C++规定字符串的计数从0开始，所以a处于位置3；第二个3是因为are是3个字符，所以截取子串的长度为3。

注意给出的要截取子串的起始位置必须位于字符串中，否则出错。如果截取长度大于字符串长度，则是可以的，程序将自动截取到字符串末尾，如语句string strnew = newstr.substr(2,8);和语句string strnew = newstr.substr(2,1);等效，都是截取字符e。





使用string类的典型成员函数

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例:

2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

②成员函数find用来在主串中检索所需字符串，格式为：

对象名称.find（要查找的字符串，开始查找的位置）；

函数返回查找到的字符串在主串中的位置，如：

```
int i = str1.find( "are" ,0);
```

表示从str1字符串的位置0开始查找are出现的位置，结果为3。

如果不给位置参数，默认位置参数为0。



使用string类的典型成员函数

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例:

2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

③string类还提供一个辅助功能，以便使用getline从流cin中读出输入的一行给string类的对象，如：

```
string InputLine;  
getline (cin, InputLine, ' \n' ) ;  
cout<<" your input:" <<InputLine<<endl;
```

如果输入 "I am here! " ，则得到如下结果：
your input:I am here!



```
#include <iostream>
#include <string>    //在程序中包含string类的头文件
using namespace std;

void main( )
{
    string str1("We are here!"); //用构造函数string定义对象str1并赋值
    string str2="Where are you?"; //用构造函数string定义对象str2并赋值
    cout<<str1[0]<<str1[11]<<","<<str1<<endl;
    cout<<str2[0]<<str2[13]<<","<<str2<<endl;
    cout<<"please input a word:";
    cin>>str1;    //输入good
    cout<<"lenght of the "<<str1<<" is "<<str1.size( )<<". "<<endl;
}
```

W!,We are here!
W?,Where are you?
please input a word:good
lenght of the good is 4.

3、使用complex对象

C++ 标准库提供complex类定义复数对象，在程序中包含这个类的头文件为：

```
#include <complex>
```

复数 (complex number) 类需要两个初始值：实部和虚部，complex是一个模板类，利用构造函数初始化的格式为：

```
complex <数据类型> 对象名 (实部值, 虚部值);
```

```
如: complex <int> num1(2,3); //定义复数2 + 3i
```

```
complex <float> num2(2.5,3.6); //定义复数2.5 + 3.6i
```

complex的real和imag成员函数用来输出对象的实部和虚部的值，如：

```
cout<<num2.real()<<" , " <<num2.imag()<<endl;
```



```
#include<iostream>
#include<complex>
#include<string>
using namespace std;
```

```
void main( ){
    complex <int> num1(2,3);
    complex <float> num2(3.5,4.5);
    string str1("real is ");
    string str2="image is ";
    cout<<str1<<num1.real()<<','<<str2<<num1.imag()<<endl;
    cout<<str1<<num2.real()<<','<<str2<<num2.imag()<<endl;
}
```

real is 2,image is 3
real is 3.5,image is 4.5

4、使用对象小结

2.5 使用类和对象

2.5.1 使用string对象

2.5.2 使用string类的典型成员函数实例：

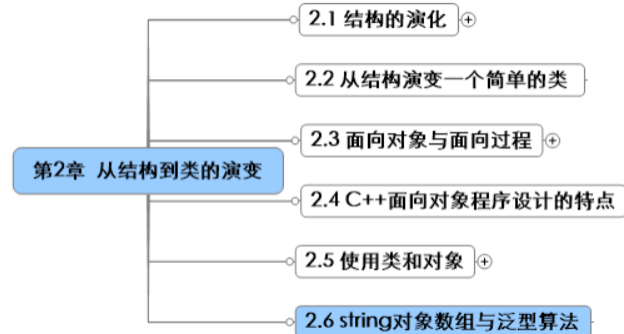
2.5.3 使用complex对象 real、imag

2.5.4 使用对象小结

标准库提供的类都是经过抽象，代表了一类对象，例如string类描述的是字符串特性，具有一个用来描述对象静态性质的字符串，字符串的值可以区分不同的对象；通过一系列的操作方法对这个字符串进行操作，用函数实现这些方法，又称这些函数为成员函数；数据成员（属性）和成员函数（方法）代表了字符串一类事物的特征。



String类的使用方法一般如下:



```
#include <string>
```

```
string str1; //定义这个类的一个对象str1
```

```
str1 = "this is a string" ; //给str1赋值
```

```
string str2 ( "this is a str2" ) ; //定义str2并赋值
```

```
cout<<str1<<str2.size( );
```

可以先定义对象，然后给它赋值（如上式str1），也可以在定义对象的同时初始化对象（如上式str2），如果要使用对象的成员函数，则用“.”运算符（如最后一句）。

同理，复数类可以抽象为具有实部和虚部的数据成员，以及能对复数进行基本操作的成员函数，与string不同的是，定义复数类时与数据成员的类型无关，当定义复数类的对象时才指定实部和虚部的数据类型。

注意，类是抽象出一类物质的共同特征，模板则是归纳出不同类型事物的共同操作。



§2.6 string对象数组与泛型算法

第2章 从结构到类的演变

2.1 结构的演化 ⊕

2.2 从结构演变一个简单的类

2.3 面向对象与面向过程 ⊕

2.4 C++面向对象程序设计的特点

2.5 使用类和对象 ⊕

2.6 string对象数组与泛型算法

1、要注意不要将该节介绍的find函数与string本身的find函数混淆。

另外，string类还有一个swap成员函数，用来交换两个对象的属性。假设有两个string类的对象str1和str2，下面两种调用方式是等效的：

```
str1.swap(str2);
```

```
str2.swap(str1);
```

例：演示string对象

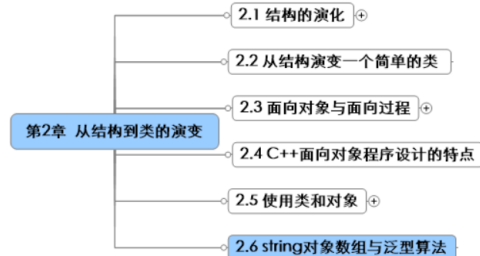



```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
void main( )
{
    string str1="we are here!",str2=str1;
    reverse(&str1[0],&str1[0]+12);
    copy(&str1[0],&str1[0]+12,&str2[0]);
    cout<<str1<<endl<<str2<<endl;
    reverse_copy(&str2[0],&str2[0]+12,ostream_iterator<char>(cout));
}
```

!ereh era ew
!ereh era ew
we are here!



2、string类的成员函数begin和end



二者都用来指示元素位置，前者指示第一个元素，后者指示最后一个元素后面的字符串结束位置。如果begin不等于end，算法首先作用于begin所指元素，并将begin前进一个位置，然后作用于当前的begin所指元素，如此继续前进，直到begin等于end为止，所以它们是元素存在的半开区间 [begin,end)。

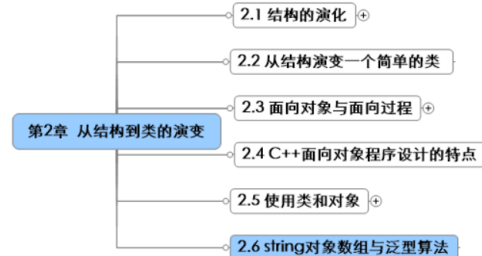
在实例中，例如str2值为“wrrheeeeea!”，则reverse(str2.begin()+2, str2.begin()+8); 执行后str2值为“wreeeehra!”，这是因为begin()+2 ~ begin+8是一个半开区间，区间包含的元素不是7个，而是上面阴影中包含的6个，即处于begin()+8位置的元素a并不被包含在这个半开区间中。同理，上例中的&str1[0] ~ &str1[12]也是这样的一个半开区间，包含的元素是12个而不是13个。

有了这两个成员函数，就可以用它们表示元素存储区间，使用string定义的字符串中不用字符‘\0’为结束符，而使用char定义的字符串则自动在尾部加入‘0’作为结束符。

例：演示string对象使用成员函数表示存储空间



演示string对象数组



3、虽然可以声明string类的数组，但只能对数组分量使用这些操作，不能对整个数组使用这些操作，**swap成员函数**可以用来交换两个数组分量。

例：演示string对象数组



```
#include <iostream>
#include <string>
#include <algorithm>
#include <functional>
using namespace std;

void main( )
{
    string str1="wearehere!",str2(str1);
    reverse(str1.begin( ),str1.end( )); //str1逆向
    cout<<str1<<endl;    //输出str1="!ereheraew"
    copy(str1.begin( ),str1.end( ),str2.begin( ));
    sort (str1.begin( ),str1.end( )); //按默认升幂排序str1
    cout<<str1<<endl;    //输出str1 = "!aeeeeerrw"
    cout<<str2<<endl;    //输出str2 = "!ereheraew"
    reverse_copy(str1.begin( ),str1.end( ),str2.begin( ));
    cout<<str2<<endl;    //输出str2 = "wrrheeeea!"
```

```
reverse(str2.begin( )+2,str2.begin( )+8); //此时str2 = "wreeeehra!"  
copy(str2.begin( )+2,str2.begin( )+8,ostream_iterator<char>(cout));  
//输出"eeeehr"  
sort(str1.begin( ),str1.end( ),greater<char>( )); //str1降幂排列  
cout<<str1<<endl;          //输出str1 = "wrrheeeea! "
```

```
str1.swap(str2);          //互换内容  
cout<<str1<<" "<<str2<<endl;  
//输出wreeeehra!(str1) wrrheeeea!(str2)
```

```
cout<<(*find(str1.begin( ),str1.end( ),'e')== 'e')<<" "  
      <<(*find(str1.begin( ),str1.end( ),'e')== 'o')<<endl;  
}          //输出1 0 , 注意上面的find不是成员函数find
```

```
!ereheraew  
!aeeeeherw  
!ereheraew  
wrrheeeea!  
eeeehrwrrheeeea!  
wreeeehra! wrrheeeea!  
1 0
```

```

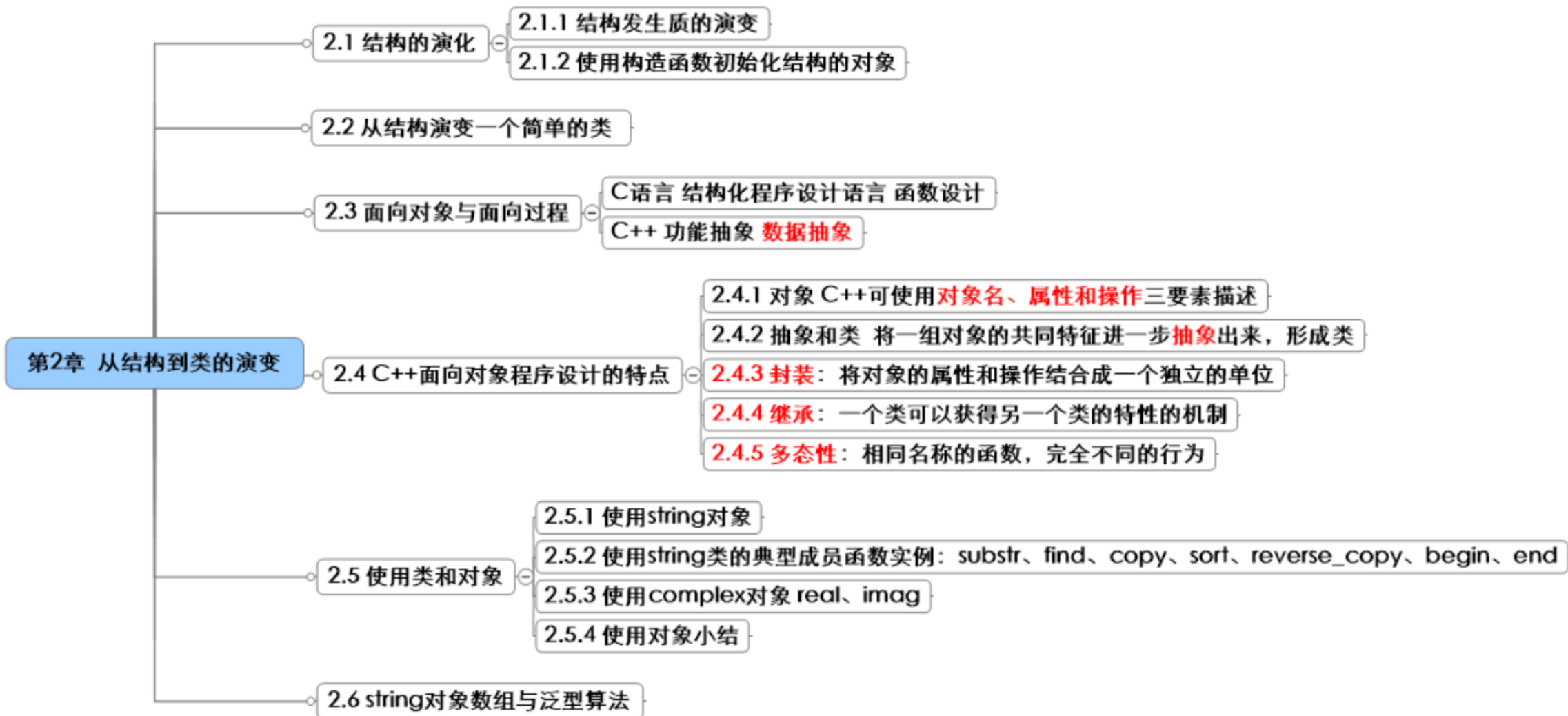
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
void main( )
{
    string str[ ]={"we are here!","where are you?","welcome"};
    for(int i = 0;i<3;i++){
        copy(str[i].begin( ),str[i].end( ),ostream_iterator<char>(cout));
        cout<<endl;
    } //for循环, 换行分别输出we are here! Where are you? Welcome!
    str[0].swap(str[2]); //互换, str[0] = "Welcome!" str[2] = "we are here!"
    str[0].swap(str[1]); //互换, str[0] = "Where are you?" str[1] = "Welcome!"
    for(i=0;i<3;i++)
        cout<<str[i]<<endl; //for循环, 换行分别输出Where are you?
}

```

we are here!
where are you?
welcome
where are you?
welcome
we are here!



本章总结



2018年4月 部分真题讲解



1. 下列关于 C++ 标识符的命名不合法的是

A. Pad

B. name_1

C. A#bc

D. _a12

2. 若有以下类型标识符定义: `int x = 2; char w = 'a'; float y = 23.45f; double z = 45.6789;` 则表达式 `w * x + y - z` 的结果类型是

A. float

B. char

C. int

D. double

3. 局部变量可以隐藏全局变量,那么在有同名全局变量和局部变量的情形时,可以用下列哪一项提供对全局变量的访问

A. 作用域运算符

B. 指针运算符

C. 提取运算符

D. 插入运算符

4. 下列关于 delete 运算符的描述中,错误的是

A. 它必须用于 new 返回的指针

B. 对一个指针可以使用多次该运算符

C. 它也适用于空指针

D. 指针名前只用一对方括号,不管所删除数组的维数

5. 在 C++ 中,类与类之间的继承关系具有

- A. 自反性 B. 对称性 C. 传递性 D. 反对称性

6. 对类中声明的变量,下列描述中正确的是

- A. 属于全局变量
B. 属于该类,某些情况下也可被该类不同实例所共享
C. 只属于该类
D. 任何情况下都可被该类所有实例共享

7. 在类定义的外部,可以被访问的成员有

A. 所有类成员

B. private 或 protected 的类成员

C. public 的类成员

D. public 或 private 的类成员

8. 已知: `int m = 10;` 下列表示引用的方法中,正确的是

A. `int &z;`

B. `int &t = 1;`

C. `int &x = m;`

D. `float &f = &m;`

9. 对于 `int *pa[5];` 的描述中,正确的是

A. pa 是一个指向数组的指针,所指向的数组是 5 个 int 型元素

B. pa 是一个指向某数组中第 5 个元素的指针,该元素是 int 型变量

C. pa[5] 表示数组的第 5 个元素的值,是 int 型的值

D. pa 是一个具有 5 个元素的指针数组,每个元素是一个 int 型指针

14. 用 new 运算符创建一维数组的正确形式是

A. `int * p = new a[10];`

B. `float * p = new float[10];`

C. `int * p = new float[10];`

D. `int * p = new int[5] = {1,2,3,4,5,6};`

16. 如果有 `int x, * p; float y, * q;` 则下面操作中, 正确的是

A. `p = x`

B. `p = q`

C. `p = &x`

D. `p = &y`

19. 考虑函数原型 `void test(int a, int b = 7, char ch = '*')`, 下面的函数调用中, 属于不合法调用的是

- A. `test(5)` B. `test(5, 8)` C. `test(6, '#')` D. `test(0, 0, '*')`

20. 使用 `setw()` 时需要包含头文件

- A. `iostream.h` B. `fstream.h` C. `iomanip.h` D. `stdlib.h`

21. 面向对象程序设计不仅能进行功能抽象,而且能进行_____抽象。
22. C++ 提供的预处理语句有 3 种,文件包含、条件编译和_____。
23. 在类体外面定义成员函数时,必须用关键字_____重写类模板声明。
24. 输入流 `istream` 用来处理标准输入的一个对象的是_____。
25. 编译时的多态性通过_____函数实现。
26. 假定 $x = 5, y = 6$, 则表达式 $x++ * ++y$ 的值为_____。
27. 在 C++ 程序中,对刚创建的对象进行初始化的工作由构造函数来完成;而对象被删除前的一些清理工作则是由_____函数来完成的。

28. 在用 class 定义一个类时,数据成员和成员函数的默认访问权限是_____。
29. 不同对象可以调用相同名称的函数,但执行完全不同行为的现象称为_____。
30. 用 new 申请某一个类的动态对象数组时,在该类中必须能够匹配到没有形参或_____的构造函数,否则应用程序会产生一个编译错误。
31. 在 C++ 中,变量的三个基本要素是指:变量名、变量类型和_____。
32. 若有定义 `int a = 3;` 则执行完语句 `a++ = a-- = a * a;` 之后, a 的值为_____。
34. 假定 x 是一个逻辑量,则 `x&&0` 的值为_____。
40. 在函数体之前加_____关键字可以防止覆盖函数改变数据成员的值。

程序改错题:

```
#include <iostream>
using namespace std;
void setzero(int &a) {
    a = 0;
}
void main() {
    int x = 7; y = 9;
    setzero(x);
    setzero(y);
    cout << x << y << endl;
}
```

```
#include <iostream>
using namespace std;
void main() {
    int a = 11, b = 12;
    const int *p = a;
    cout << a << b << *p << endl;
}
```


程序改错题:

```
#include <iostream>
using namespace std;
class f{
    int a,b;
public:
    void set( int aa,int bb) {
        a = aa;b = bb;
    }
    void show() {
        cout << a << " " << b << endl;
    }
};
void main() {
    f. set( 1 ,2) ;f. show() ;
}
```

程序改错题：

```
#include <iostream>
using namespace std;
void main() {
    int a[6] = {1,2,3,4,5,6,7};
    int i;
    for(i=0;i<7;i++)
        cout << a[i] << endl;
}
```

完成程序题:

46. 完成程序,使其输出结果为 28

```
using namespace std;
int a[8] = {1,2,3,4,5,6,7};
void fun(int b[],int n);
void main() {
    int m = 8;
    fun(a,m);
    cout << a[7] << endl;
}
void fun(int b[],int n) {
    int i;
    for(_____)
        b[7] += b[i];
}
```

完成程序题:

```
#include <iostream>
```

```
using namespace std;
```

```
class base {
```

```
    _____;
```

```
public:
```

```
    base(int x,int y) { a = x;b = y; }
```

```
    _____(base &p) {
```

```
        cout << p. a << " , " << p. b << endl; }
```

```
};
```

```
void main() {
```

```
    base b(78,87);
```

```
    b.show(b);
```

```
}
```

程序分析题:

```
#include <iostream.h>
using namespace std;
void main( ) {
    int a[3][4] = { {1,2,7,8} , {5,6,11,15} , {9,20,3,7} } ;
    int m = a[0][0];
    int i,j;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
            if( a[i][j] > m) m = a[i][j];
    cout << "m is:" << endl;
    cout << m << endl;
}
```


程序设计题:

声明一个 circle 类,有数据成员 Radius(半径, float 型),成员函数 GetArea() 计算圆的面积。在 main 函数中声明一个 circle 类的对象 c1,其半径为 5.6。调用 GetArea() 函数计算 c1 的面积,并显示该计算结果(`cout << "圆的面积:" << c1.GetArea << endl;`)。

```
#include <iostream.h>

class circle{
public:
    float GetArea( );
    circle( float r );
protected:
    float Radius;
};

circle::circle( float r ) {
    Radius = r;
}

float circle::GetArea( ) {
    return Radius * Radius * 3.14;
}

void main( ) {
    circle c1(5.6);
    cout << " 圆的面积:" << c1.GetArea() << endl;
}
```



祝大家顺利通过考试!

