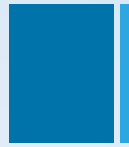




C++

第五讲 类和对象（一）

C++备课组 丁盟



自我介绍

丁盟

qq : 2622885094





上一讲教学目标

- 掌握C++中引用的使用
- 掌握C++中引用作为函数参数的使用方法
- 了解C++中的强制类型转换



本讲教学目标

- 理解C++中类的概念
- 掌握C++中类的定义与访问

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

实例化对象

4

类和对象的简单应用举例

类与对象概述

- ❖ **对象**：现实世界中某个具体的**实体**在计算机逻辑中的映射和体现。
- ❖ 现实中的**对象**都具有属性和行为。

类与对象概述

❖ **类**：是一种抽象的**数据类型**，是同种对象的集合与抽象，是具有共同行为和属性的**若干对象**的统一描述体。

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

实例化对象

4

类和对象的简单应用举例

类的声明



抽象



武将类

行为
改名 升级

属性
姓名 等级
经验 体力
精力 统率
速度 攻击
防御

映射



```
struct Generals
{
    void setName();
    void upLevel();
    string name;
    int level;
    int exp;
    int physical;
    int energy;
    int command;
    int velocity;
    int attack;
    int defensive;
};
```

类的声明

❖一般格式：

```
class <类名> //类名
{
    public:
        <成员函数或成员变量的声明>
    protected:
        <成员函数或成员变量的声明>
    private: //默认属性！！
        <成员函数或成员变量的声明>
};
```

类体

类的声明

```
class Generals
{
public:
    void setName();
    void upLevel();
private:
    string m_strName;
    int    m_iLevel;
    int    m_iExp;
    int    m_iPhysical;
    int    m_iEnergy;
    int    m_iCommand;
    int    m_iVelocity;
    int    m_iAttack;
    int    m_iDefensive;
};
```

// 类名

// 访问属性

// 数据成员

// 类体

类的声明

[注意]

- 类名，**class**是声明类的关键字，<类名>是标示符，通常以**首字母大写**开头,以与对象,函数,变量区别。
- 类体可以包含**数据成员**和**成员函数**两部分，数据成员和成员函数声明必须以**分号**结束！
- 数据成员的命名,习惯以**m_**开头后加表示类型的小写字母，最后是描述变量的标识符。
`int m_intYear or int m_iYear;`
- 声明一个类就相当于声明了一个类型和一个作用域。

类的声明

- **访问权限修饰符**，包括公有的(**public**)，私有的(**private**)和保护(**protected**)三类。

访问权限修饰符	类内	类外	
	被本类成员函数访问	被本类的对象引用	被非成员函数访问
public	√	√	√
private	√	×	×
protected	√	×	×

类的声明

- 通常数据成员设为私有，以实现信息隐蔽。
- 属性的顺序任意，通常public放最上面，private放最下面,并非一定要含有这三种访问属性。
- 每个部分的有效范围到出现另一个限定符（或类体结束）为止。
- 类的声明是不占据内存单元的，对象才占内存。

类的声明 - 练习

- 请声明一个类来存储时间。私有成员变量包括：
m_iHour , m_iMin , m_iSec。共有成员函数包括：
`void set(int aHour , int aMin , int aSec)`
`void display(void)`。
- 请声明一个类来存储空间中的点。私有成员包括：
m_dX , m_dY。共有成员函数包括：
`void set(double aX, double aY)`
`double getX(void)`
`double getY(void)`

类的声明 - 练习

```
class Time {  
public:  
    void set(int aHour, int aMin, int aSec);  
    void display();  
private:  
    int m_iHour;  
    int m_iMinute;  
    int m_iSec;  
};  
  
void Time::set(int aHour, int aMin, int aSec) {  
    m_iHour    = aHour;  
    m_iMinute  = aMin;  
    m_iSec     = aSec;  
}  
  
void Time::display() {  
    cout << m_iHour << ":" << m_iMinute << ":" << m_iSec << endl;  
}
```


类的声明 - 练习

```
class Point {
public:
    void set(double aX, double aY )
    {
        m_dX = aX;
        m_dY = aY;
    }
    double getX()
    {
        return m_dX;
    }
    double getY()
    {
        return m_dY;
    }
private: // 私有成员只能被本类的成员函数调用
    double m_dX;
    double m_dY;
};
```

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

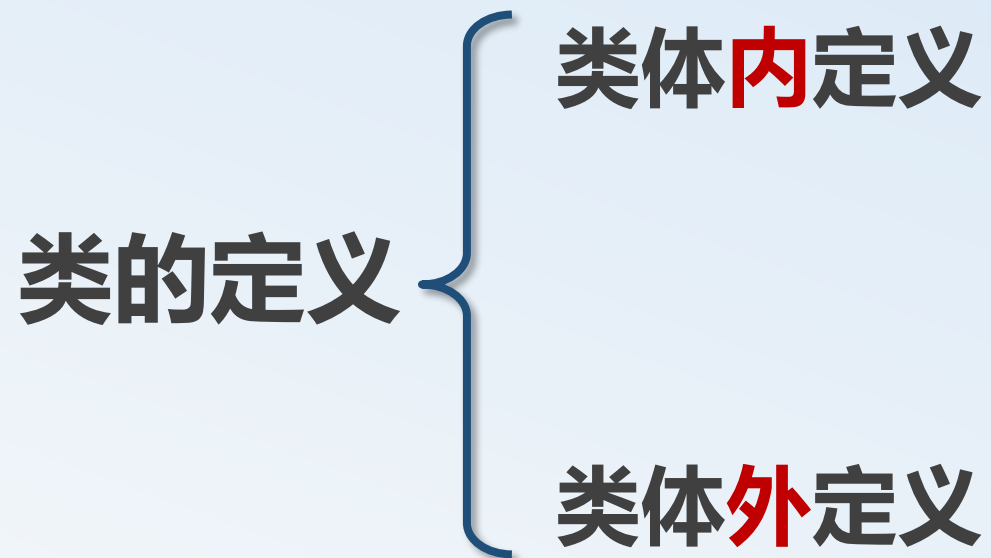
实例化对象

4

类和对象的简单应用举例

类的定义

❖ 类的定义：给出**成员函数**具体的功能实现。



类的定义

➤ 类体内定义：

```
class Point{  
public:  
    void setX( int aX ) { m_iX = aX; }  
    void setY( int aY ) { m_iY = aY; }  
    void move( int aX, int aY ){  
        m_iX += aX;  
        m_iY += aY;  
    }  
    void display(){  
        cout << "X:" << m_iX << endl;  
        cout << "Y:" << m_iY << endl;  
    }  
private:  
    int m_iX;  
    int m_iY;  
};
```

类的定义

➤ 类体内定义：

```
<返回值>  <类名>::<函数名> (<参数表>)  
{  
    函数体;  
}
```

类的定义

类体外定义

```
class Time {  
public:  
    void set(int aHour, int aMin, int aSec);  
    void display();  
private:  
    int m_iHour;  
    int m_iMinute;  
    int m_iSec;  
};  
  
void Time::set(int aHour, int aMin, int aSec) {  
    m_iHour    = aHour;  
    m_iMinute  = aMin;  
    m_iSec     = aSec;  
}  
  
void Time::display() {  
    cout << m_iHour << ":" << m_iMinute << ":" << m_iSec << endl;  
}
```

类的定义

[注意]

- 成员函数类体外定义时，必须在函数名前面加 **"类名::"**。
- 类体外定义成员函数时，**类体内必须声明该成员函数**！
- **不允许**在类的声明部分对成员变量进行初始化。

```
class Point
```

```
{
```

```
private:
```

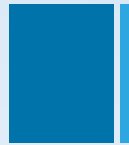
```
    int m_iX = 0;
```

```
    int m_iY = 0;
```



```
};
```

- 将类的声明与定义分开，类的声明放到头文件，定义放到源文件中。



类的定义

- 类体内定义成员函数系统自动设为inline
- 类体外定义成员函数默认不是内联函数

类的定义

```
class Student
{
public:
    void display() // inline void display()
    {
        ... ..
    }
private:
    int      m_iNum;
    string   m_strName;
    char     m_cSex;
};
```

类的定义

```
class Student
{
public:
    inline void display()const;
private:
    int      m_iNum;
    string   m_strName;
    char     m_cSex;
};
void Student::display()const
{
    ... ..
}
```

成员函数在类体外定义，并不默认为inline函数，若想成员函数为inline应显式声明

类的定义

```
class Student
{
public:
    inline void display()const;
private:
    int      m_iNum;
    string m_strName;
    char      m_cSex;
};
inline void Student::display()const
{
    ... ..
}
```

inline的位置：
声明时必须有
定义时可省略。

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

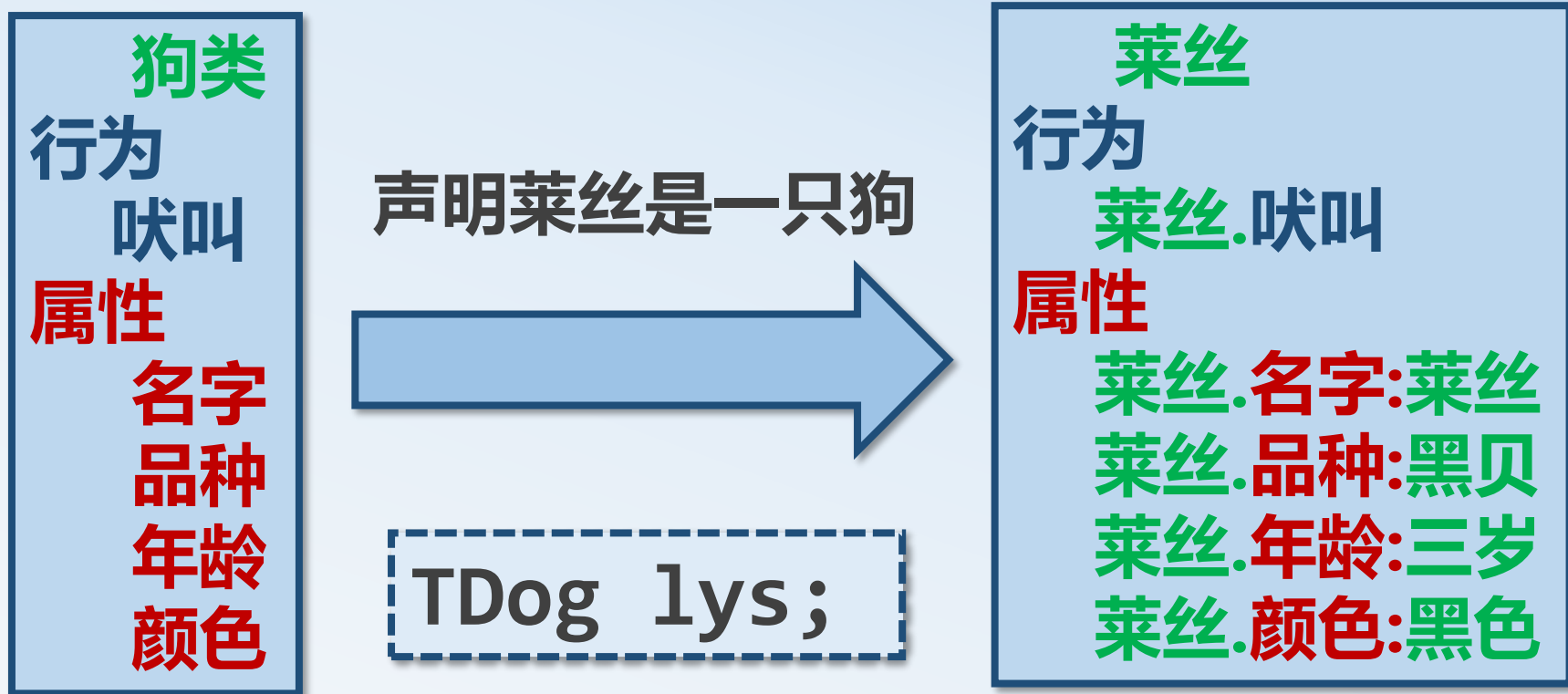
实例化对象

4

类和对象的简单应用举例

对象的声明与成员访问

❖ 概念：声明对象就是创建对象的过程



对象的声明与成员访问

❖一般格式：**<类名> <对象名表>;**

对象名表

一个对象

```
TDate yesterday;
```

多个对象

```
TDate today, tomorrow;
```

对象数组

```
TDate date[3];
```

对象指针

```
TDate *p = &today;
```

对象引用

```
TDate &rdate = today;
```

对象的声明与成员访问

❖ 回忆结构体成员的访问

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

struct Student
{
    string name;
};

Student stu, *p = &stu;
Student &rstu = stu;
```

```
int main(void)
{
    stu.name = "Jack";
    cout << stu.name << endl;
    (*p).name = "Mike";
    cout << (*p).name << endl;
    p->name = "Mike";
    cout << p->name << endl;
    rstu.name = "Jack";
    cout << rstu.name << endl;
    return 0;
}
```

对象的声明与成员访问

❖ 成员的访问方式：

- 通过对象名访问
- 通过对象数组元素访问
- 通过对象指针访问
- 通过对象引用访问

对象的声明与成员访问

➤方法一：

通过对象名访问

对象名 . 成员名

```
int main(void)
{
    Point pt;
    pt.set( 100, 100 );
    pt.display();
    return 0;
}
```

```
class Point{
public:
    void display(){
        cout<<"X: "<<m_iX<<endl;
        cout<<"Y: "<<m_iY<<endl;
    }
    void set( int aX, int aY){
        m_iX = aX;
        m_iY = aY;
    }
private:
    int m_iX;
    int m_iY;
};
```

对象的声明与成员访问

➤方法二：

通过对象数组元素访问

对象数组元素 . 成员名

```
class Point
{
    ... ..
};

int main(void)
{
    Point arr[2];
    arr[0].set( 100, 100 );
    arr[0].display();
    arr[1].set( 200, 200 );
    arr[1].display();
    return 0;
}
```

对象的声明与成员访问

➤方法三：

通过对象指针访问

对象指针 -> 成员名

(*对象指针) . 成员名

```
class Point
{
    ... ..
};
int main(void)
{
    Point pt;
    Point *p = &pt;
    p->set(10, 10);
    p->display();
    (*p).set(20, 20);
    (*p).display();
    return 0;
}
```

对象的声明与成员访问

- 方法四：
通过对象引用访问

对象引用 . 成员名

```
class Point
{
    ... ..
};

int main(void)
{
    Point pt;
    Point &rp = pt;
    rp.set(10,10);
    rp.display();

    return 0;
}
```

对象的声明与成员访问

❖成员访问总结：

对象名 . 数据成员名;
对象名 . 成员函数名([参数表]);

对象数组元素 . 数据成员名;
对象数组元素 . 成员函数名([参数表]);

(*p) . 数据成员名; // p->数据成员名;
(*p) . 成员函数名([参数表]); // p->成员函数名([参数表]);

对象引用 . 数据成员名;
对象引用 . 成员函数名([参数表]);

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

实例化对象

4

类和对象的简单应用举例

实例化对象

```
class Student
{
public:
    void init(string aNum="000",
              string aName="NULL",
              bool aSex=true);
    void display()const;

private:
    string m_iNum;
    string m_strName;
    bool m_bSex;
};
```

```
void Student::init(string aNum,
                   string aName,
                   bool aSex){
    m_iNum = aNum;
    m_strName = aName;
    m_bSex = aSex;
}

void Student::display()const{
    cout << "num:" << m_iNum
          << endl;
    cout << "name:" << m_strName
          << endl;
    cout << "sex:" << boolalpha
          << m_bSex << endl;
}
```

■ 实例化对象



实例化对象 - 栈区

- ❖ **概念**：局部对象和形参表中的对象
- ❖ **创建**：在函数体内或者形参表中定义对象
- ❖ **释放**：函数执行完时由系统释放
- ❖ **访问**：对象名，指向对象的指针或引用

```
int main(void){  
    Student localStu;  
    Student *p = &localStu;  
    Student &r = localStu;  
    localStu.display();  
    p->display();    // (*p).display();  
    r.display();  
    return 0;  
}
```



实例化对象



实例化对象 - 堆区

❖概念：用**new**创建的对象

❖创建：

new 类名;

new 类名[无符号整型表达式];

可为无符号
整型变量

❖释放：

delete 指针名

delete []指针名

❖访问：对象名，指向对象的指针或引用

实例化对象 - 堆区

➤ 指针访问：

```
int main(void)
{
    Student *pStu = NULL;
    pStu = new Student;
    pStu->init("001", "a", 0);
    pStu->display();
    delete pStu;
    pStu = NULL;

    return 0;
}
```

➤ 引用访问：

```
int main(void)
{
    Student &r = *new Student;
    r.init("002", "b", 1);
    r.display();
    delete &r;

    return 0;
}
```

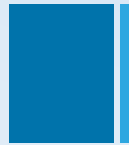
实例化对象 - 堆区

```
int main(void){
    const int KSize = 3;
    Student *pStuArr = NULL;
    pStuArr = new Student[KSize];
    string num,name;
    bool sex;
    for( Student *p=pStuArr; p!=pStuArr+KSize; p++){
        cout << "学号,姓名, 性别 : ";
        cin >> num >> name >> sex;
        p->init( num, name, sex );
    }
    for( Student *p=pStuArr; p!=pStuArr+KSize; p++){
        p->display();
    }
    delete []pStuArr;
    pStuArr = NULL;
    return 0;
}
```

实例化对象 - 堆区

[注意]

- delete只能回收由new返回的指针指向的内存空间。
- 不用的堆对象一定要delete,一个指针只能delete一次。
- 释放堆对象数组时，无论几维的堆对象均为**delete []p;**



实例化对象



实例化对象 - 全局区

- ❖ **概念**：全局对象，局部静态对象
- ❖ **创建**：定义全局对象，局部静态对象
- ❖ **释放**：整个程序执行完成后，由系统自动释放
- ❖ **访问**：对象名，指向对象的指针或引用

实例化对象 - 全局区

```
class Student
{
public:
    void set( const string aName )
    {
        m_strName = aName;
    }
    void display() const
    {
        cout<<m_strName<<endl;
    }
private:
    string m_strName;
};
```

实例化对象 - 全局区

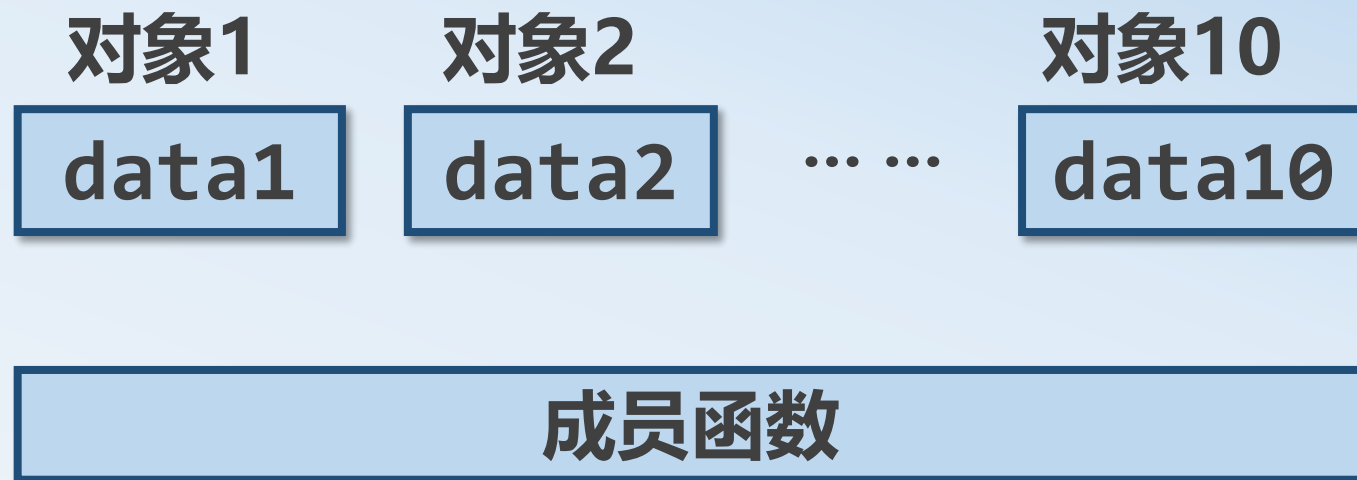
```
Student s_stu2, *p = & s_stu2, &r = s_stu2;  
int main(void) {  
    static Student s_stu1;  
    s_stu1.set ("Lily" );  
    s_stu1.display();  
    s_stu2.set ("Morrison");  
    s_stu2.display();  
    (*p).setName("Jack");  
    p->display();  
    r.set ("Jimi");  
    r.display();  
    return 0;  
}
```

■ 实例化对象 – 对象的存储

- ❖ 无论对象存储在什么区域，存储的规则是相同的。
- ❖ **思考：**对象的存储方式？

实例化对象 - 对象的存储

❖ 对象的存储方式：



每个类的对象分别存储本对象的所有成员变量。
一个类的所有对象共用成员函数。

实例化对象 - 对象的存储

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

class Time
{
public:
    void set();
private:
    int m_iHour;
    int m_iMinute;
    int m_iSec;
};
```

```
void Time::set()
{
    cin >> m_iHour
        >> m_iMinute
        >> m_iSec;
}

int main(void)
{
    cout << sizeof(Time)
        << endl;
    return 0;
}
```

实例化对象 – 对象的存储

[注意]

- 类所占的内存由成员变量的内存 “和” 加间隙组成（规则与结构体所占内存相同）。
- 无论成员函数在类体内还是在类体外定义，**成员函数都不占用对象的存储区。**
- 说成员函数是对象的，只是从逻辑角度上说的，而非物理角度而言。
- 对象的存储类别和作用域与内置类型变量相同。

1

类与对象概述

2

类

类的声明

类的定义

3

对象

对象的声明与成员访问

实例化对象

4

类和对象的简单应用举例

类和对象的简单应用举例

❖ 通过一个例子说明怎样用类设计程序

```
class Time
{
public:
    int m_iHour;
    int m_iMinute;
    int m_iSec;
};

int main()
{
    Time current;
    cout << "input hour : ";
    cin >> current.m_iHour;
```

无法实现数据隐藏，
只有属性没有行为

```
    cout << "input minute : ";
    cin >> current.m_iMinute;
    cout << "input sec : ";
    cin >> current.m_iSec;
    cout << current.m_iHour
        << ":"
        << current.m_iMinute
        << ":"
        << current.m_iSec
        << endl;
    return 0;
}
```


类和对象的简单应用举例

```
class Time {  
public:  
    void input();  
    void display();  
private:  
    int m_iHour;  
    int m_iMinute;  
    int m_iSec;  
};  
void Time::input() {  
    cin >> m_iHour;  
    cin >> m_iMinute;  
    cin >> m_iSec;  
}
```

```
void Time::display() {  
    cout << m_iHour << ":"  
        << m_iMinute << ":"  
        << m_iSec << endl;  
}  
int main(void) {  
    Time cur;  
    cur.input();  
    cur.display();  
    Time pas;  
    pas.input();  
    pas.display();  
    return 0;  
}
```

类和对象的简单应用举例

[注意]

- **数据成员**常常设为**private**，以实现信息隐蔽。
- 声明类时，通常将类的声明与实现分离，类的声明在头文件中，类的实现在一个同名源文件中，主函数定义在一个源文件中。



本讲教学目标

- 理解C++中类的概念
- 掌握C++中类的定义与访问



THANKS

