

# 实验六 类

## 一、 问题描述

### 1. 实验目的：

掌握类的若干基本概念和特性，并能够应用于程序编写。

掌握验证性实验的基本方法和过程(认知、实验、总结)。

### 2. 实验内容：

分别编写一段测试代码来回答任务书中的相关问题（每一个问题，用一个工程文件，同时需要记录相应的调试过程），具体问题请参考“实验任务说明06.doc”；

调试的过程；（动态调试的相关截图，比如 设置断点、查看当前变量值等）；

编译出来的可执行程序单独放在一个目录下（bin/exe/debug目录下，同时附上输入数据说明和输出结果）

## 二、 实验过程

### 1. 名词解释

#### 1、对象和类

在面向对象程序设计中,对象是描述其属性的数据以及对这些数据施加的一组操作封装在一起构成的统一体。类是 C++ 的核心特性，类用于指定对象的形式，它包含了数据表示法和用于处理数据的方法。类提供了对象的蓝图，所以基本上，对象是根据类来创建的。声明类的对象，就像声明基本类型的变量一样。

#### 2、封装

“封装”就是将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体（即类）；封装的目的是增强安全性和简化编程，使用者不必了解具体的实现细节，而只是要通过外部接口，一特定的访问权限来使用类的成员。

#### 3、类的前向声明

前向引用声明，是在引用未定义的类之前，将该类的名字告诉编译器，使编译器知道那是一个类名。这样，当程序中使用这个类名时，编译器就不会认为是错误，

而类的完整定义可以在程序的其他地方。

## 2. 填空题

- 1、C++中通常使用关键字 class 定义类类型。
- 2、在类体内定义成员函数是 方法 函数。
- 3、在类体外定义成员函数时，函数名前要加上 类名和 作用域运算符（：:）。
- 4、用class定义的类中的数据成员和成员函数默认访问属性是 private。
- 5、类中定义的数据成员 能 在类体内进行初始化，应通过类的 构造函数 进行初始化。
- 6、类中的访问权限修饰符可以以任意顺序出现多次。（正确、错误）（tips：在一个类中，`public`、`protected`、`private` 可以出现多次，每个限定符的有效范围到出现另一个限定符或类结束为止。但是为了使程序清晰，应该养成这样的习惯，使每一种成员访问限定符在类定义体中只出现一次。）

## 3. 简答题

### 3.1 说明一个类的公有成员、保护成员和私有成员的区别。

答：公有成员：关键字`public`将类成员声明为公有成员，可以被类对象及其所有成员访问。私有成员：关键字`private`将类成员声明为私有成员，不能被类对象直接访问。保护成员：关键字`protected`修饰的成员声明为保护成员，不能被类对象直接访问。其访问权限与私有成员近似，所不同的是其可对于基类的派生类是可见的，而私有成员则不可见

### 3.2 什么是数据隐藏？引入私有成员的原因？

答：数据隐藏是封装的目的，为对对象的内部数据表现形式和实现细节进行隐藏。要想访问封装过的对象中的数据，只有使用已定义的操作这一种办法。引入私有成员的原因是为了封装。暴露出来的接口应该是稳定的，利用类中的公共接口函数，我们就可以访问调用类的私有成员。好处是可以提高代码的安全性，防止造成错误的输入与输出。因为在类的公共接口函数中，我们可以对输入的具体值进行限定，那么这样就不会造成数据的错误。

### 3.3 为什么在一个类中要有公有成员函数？

答：只要这个成员变量具备无条件的可读可写权限时，就可将其设为`public`。从封装的角度，“尽量”不暴露内部数据结构，但需要明确的是，封装的本质是封装实现细节，并不是隐藏一切。一个成员变量如果具备无条件的读写权限，说明其为外部开放数据，`get`、`set`的好处是为了在编程过程中提醒调用者正在对该变量做读写的操作，也可对输入输出数据做参数检查，防止无效数据和非法访问。如果`get`、`set`仅用于简单传递数据，说明这个成员变量具备无条件的读写权限，从代码简洁方面考虑，并且完全可以略去`get`、`set`这种“墨守陈规”的操作了。

## 四、验证题

4.1 设计一段代码，验证私有成员的性质：在类定义外访问（如main函数中）私有成员是受限的。

### （1）设计代码

```
#include <iostream>

class Player {
private:
    int x, y;
    int speed;
public:
    int money;
};

int main() {
    Player player;
    player.x = 5; //不可以修改私有变量
    player.m
    (字段) int Player::x
    联机搜索
    成员 "Player::x" (已声明 所在行数:5) 不可访问
    联机搜索

#include <iostream>

class Player {
private:
    int x, y;
    int speed;
public:
    int money;
};

int main() {
    Player player;
    //player.x = 5; //不可以修改私有变量
    player.money = 10;
}
```

### （2）结果说明

设计了一个Player类，其中x, y和speed是其私有成员，money是其公有成员，在类定义外访问（main函数中）私有成员x是受限的，而访问公有成员money是允许的。

## 五、程序设计题

## 5.1 设计Date类，该类采用三个整数存储日期：year、month和day。写一个完整的程序。

### （1）设计思路

#### ① 设计类

设计Date类，该类采用三个整数存储日期：year、month和day。成员函数setdate()函数带有默认参数2010，12，25。除此之外，类的成员函数还有用于读取数据的cindata、用于打印不同数据格式的cout1、cout2、cout3，用于将月份由数字转换为英文名称的getmonth。

```
class Date {
private:
    int month;
    int day;
    int year;
public:
    void cindata();
    void cout1();
    void cout2();
    void cout3();
    string getmonth(int month);

    void setdata() {
        year = 2010;
        month = 12;
        day = 25;
    }
};
```

```

void Date::cindata() {
    cout << "请输入日期（如2022 3 27）:";
    cin >> year >> month >> day;
    if (month > 12 || month < 1)    throw "无效月份";
    if (day > 31 || day < 1)    throw "无效日期";
}

void Date::cout1() {
    cout << endl << month << "-" << day << "-" << year % 100 << endl;
}

void Date::cout2() {
    cout << endl << getmonth(month) << " " << day << "," << year << endl;
}

void Date::cout3() {
    cout << endl << day << " " << getmonth(month) << " " << year << endl;
}

string Date::getmonth(int month) {
    switch (month) {
        case 1: return "January";
        case 2: return "February";
        case 3: return "March";
        case 4: return "April";
        case 5: return "May";
        case 6: return "June";
        case 7: return "July";
        case 8: return "August";
        case 9: return "September";
        case 10: return "October";
        case 11: return "November";
        case 12: return "December";
    }
}

```

## ② 设计main函数

```

int main()
{
    Date date;
    int choice;
    while (true) {
        while (true) {
            try {
                date.cindata();
                break;
            }
            catch (const char* msg) {
                cerr << msg << endl; //输出throw的错误信息
            }
        }
        while (true) {
            cout << "\n1. 输出12-25-11格式" << endl;
            cout << "2. 输出December 25, 2011格式" << endl;
            cout << "3. 输出25 December 2011格式" << endl;
            cout << "4. 重新输入" << endl;
            cout << "选择: ";
            cin >> choice;
            if (choice == 4) {
                cout << endl;
                break;
            }
            switch (choice) {
                case 1: date.cout1(); break;
                case 2: date.cout2(); break;
                case 3: date.cout3(); break;
                default: break;
            }
        }
    }
}

```

## (2) 实验结果:

程序首先读取用户输入的日期，随后支持用户在选择输出模式后进行日期的输出。提供有以下三种方式：(1)12-25-10; (2)December 25, 2010; (3)25 December 2010。对于日期day成员，不能接受大于31或小于1的值，否则就会抛出无效日期的错误；对于月month，不能接受大于12或小于1的值，否则就会抛出无效月份的错误。

```
请输入日期（如2022 3 27）:2022 3 77  
无效日期  
请输入日期（如2022 3 27）:2022 3 27
```

```
1. 输出12-25-11格式  
2. 输出December 25, 2011格式  
3. 输出25 December 2011格式  
4. 重新输入  
选择: 1
```

```
3-27-22
```

```
1. 输出12-25-11格式  
2. 输出December 25, 2011格式  
3. 输出25 December 2011格式  
4. 重新输入  
选择: 2
```

```
March 27, 2022
```

```
1. 输出12-25-11格式  
2. 输出December 25, 2011格式  
3. 输出25 December 2011格式  
4. 重新输入  
选择: 3
```

```
27 March 2022
```

5.2 在5.1的基础上，采用多文件编程的形式。注意：.h文件中的头文件卫士

（1）Date.h头文件：

```

#pragma once
#include <string>
#include <iostream>
using namespace std;

class Date {
private:
    int month;
    int day;
    int year;
public:
    void cindata();
    void cout1();
    void cout2();
    void cout3();
    std::string getmonth(int month);

    void setdata() {
        year = 2010;
        month = 12;
        day = 25;
    }
};

```

(2) Date.cpp文件:

---

```

#include "Date.h"

void Date::cindata() {
    cout << "请输入日期 (如2022 3 27) :";
    cin >> year >> month >> day;
    if (month > 12 || month < 1)    throw "无效月份";
    if (day > 31 || day < 1)       throw "无效日期";
}

void Date::cout1() {
    cout << endl << month << "-" << day << "-" << year % 100 << endl;
}

void Date::cout2() {
    cout << endl << getmonth(month) << " " << day << ", " << year << endl;
}

void Date::cout3() {
    cout << endl << day << " " << getmonth(month) << " " << year << endl;
}

```



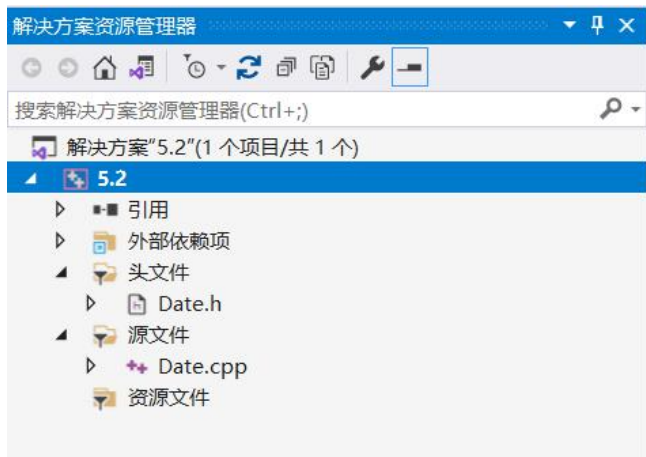
```
string Date::getmonth(int month) {  
    switch (month) {  
        case 1: return "January";  
        case 2: return "February";  
        case 3: return "March";  
        case 4: return "April";  
        case 5: return "May";  
        case 6: return "June";  
        case 7: return "July";  
        case 8: return "August";  
        case 9: return "September";  
        case 10: return "October";  
        case 11: return "November";  
        case 12: return "December";  
    }  
}
```

```

int main()
{
    Date date;
    int choice;
    while (true) {
        while (true) {
            try {
                date.cindata();
                break;
            }
            catch (const char* msg) {
                cerr << msg << endl; // 输出throw的错误信息
            }
        }
        while (true) {
            cout << "\n1. 输出12-25-11格式" << endl;
            cout << "2. 输出December 25, 2011格式" << endl;
            cout << "3. 输出25 December 2011格式" << endl;
            cout << "4. 重新输入" << endl;
            cout << "选择: ";
            cin >> choice;
            if (choice == 4) {
                cout << endl;
                break;
            }
            switch (choice) {
                case 1: date.cout1(); break;
                case 2: date.cout2(); break;
                case 3: date.cout3(); break;
                default: break;
            }
        }
    }
}

```

### (3) 项目目录



#### (4) 实验结果

和实验5.1相同，得到相同结果。

```
D:\XPfile\学习资料\年级分类\大二下\c++\myexp\第五周\src\5.2\Debug\5.2.exe
请输入日期（如2022 3 27）:2022 3 27

1. 输出12-25-11格式
2. 输出December 25, 2011格式
3. 输出25 December 2011格式
4. 重新输入
选择: 1

3-27-22

1. 输出12-25-11格式
2. 输出December 25, 2011格式
3. 输出25 December 2011格式
4. 重新输入
选择: 2

March 27, 2022

1. 输出12-25-11格式
2. 输出December 25, 2011格式
3. 输出25 December 2011格式
4. 重新输入
选择: 3

27 March 2022
```

5.3 采要计算3个长方体的体积，请编写一个程序。

#### (1) 设计思路：

设计一个名为Cuboid的类，数据成员有代表长、宽、高、体积的length、width、height、volume。成员函数setdata可以实现实现由键盘输入长宽高。对于输入的

信息，不能接受小于等于0的值，如果小于等于0将会抛出错误。成员函数volumevalue可以计算长方体的体积。成员函数printvalue输出长宽高后，输出体积。

## (2) Cuboid.h头文件

```
#pragma once
#include<iostream>
using namespace std;
class Cuboid
{
private:
    float length;
    float width;
    float height;
    float volume;
public:
    void setdata();
    void volumevalue();
    void printvalue();
};
```

## (3) Cuboid.cpp文件

```
#include "Cuboid.h"

inline void Cuboid::setdata() {
    cout << "长方体的长、宽、高分别为: ";
    cin >> length >> width >> height;
    if (length<=0 || width <= 0 || height <= 0) {
        throw "无效输入";
    }
}

inline void Cuboid::volumevalue() {
    volume = length * width * height;
}

inline void Cuboid::printvalue() {
    cout << endl<<"长方体的长、宽、高为: ";
    cout << length << ' ' << width << ' ' << height << endl ;
    cout << "长方体的体积为: ";
    cout << volume << endl << endl;
```

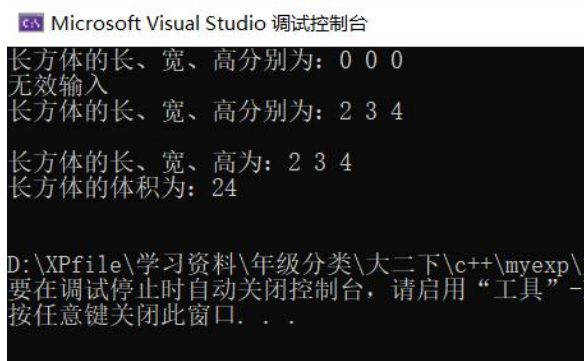
```

int main()
{
    Cuboid cub;
    while (true) {
        try {
            cub.setdata();
            break;
        }
        catch (const char*msg) {
            cerr << msg << endl;
        }
    }
    cub.volumevalue();
    cub.printvalue();
    return 0;
}

```

#### (4) 实验结果:

程序成功运行，实现由键盘输入长宽高。对于输入的信息，不能接受小于等于0的值。如果小于等于0则抛出无效输入的错误。对输入的长宽高计算长方体的体积。输出长宽高后，输出体积。



Microsoft Visual Studio 调试控制台

```

长方体的长、宽、高分别为: 0 0 0
无效输入
长方体的长、宽、高分别为: 2 3 4

长方体的长、宽、高为: 2 3 4
长方体的体积为: 24

D:\XPfile\学习资料\年级分类\大二下\c++\myexp\1
要在调试停止时自动关闭控制台，请启用“工具”->
按任意键关闭此窗口. . .

```

## 六、程序填空题，通过注释方式描述程序执行过程

### (1) 程序填空

```
#include <iostream>
```

```
#include <cstring>
```

```
class Student {
```

```
public:
```

```
void Register(char *n,char s='M',int a=19,float sc=90);
```

```

        void Showme();
private:
        char name[20];
        char sex;
        int age;
        float score;
};

void Student:: Register(char *n,char s,int a,float sc) {
        strcpy(name,n);
        sex=s;
        age=a;
        socre=sc;
}

inline void Student:: showme() { //描述为内联
        cout<<name <<'\'<<sex <<'\'<<age <<'\'<<score<<endl;
}

int main() {
        Student stu1,stu2;
        Stu1.Register("Tom");
        Stu2.Register("Lucy",'F',18,98);
        cout<<"输出"<<endl;
        Stu1.showme();
        Stu2.showme();
        return 0;
}

```

(2) 通过注释的方法描述程序执行过程：

```

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstring>
using namespace std;

class Student {
public:
    void Register(const char* n, char s = 'M', int a = 19, float sc = 90);
    void showme();
private:
    char name[20];
    char sex;
    int age;
    float score;
};

void Student::Register(const char* n, char s, int a, float sc) {
    strcpy(name, n);
    sex = s;
    age = a;
    score = sc;
}

inline void Student::showme() { //内联
    cout << name << '\t' << sex << '\t' << age << '\t' << score << endl;
}

int main() {
    Student Stu1, Stu2;
    //注册名为Tom的同学, sex、age、score采用default值M、19、90
    Stu1.Register("Tom");
    //注册名为Lucy, sex为F, age为18, score为98的同学
    Stu2.Register("Lucy", 'F', 18, 98);
    cout << "输出" << endl;
    Stu1.showme(); //输出TOM的信息
    Stu2.showme(); //输出LUCY的信息
    return 0;
}

```

### (3) 实验结果:



Microsoft Visual Studio 调试控制台

```

输出
Tom      M      19      90
Lucy     F      18      98

```

D:\XPfile\学习资料\年级分类\大  
要在调试停止时自动关闭控制台，  
按任意键关闭此窗口。 . . .

## 七、关于this指针的认识

(1) 在C++中，this指针的作用是什么？

答：this指针存在于类的成员函数中，是指向当前对象实例的指针。this指向被调用函数类实例的地址。一个对象的this指针并不是对象本身的一部分。this指针的作用域是在类内部，当在类的非静态成员函数中访问类的非静态成员的时候，编译器会自动将对象本身的地址作为一个隐含参数传递给函数。换句话说就是，即使你没有用this指针，编译器在编译的时候也会自动加上this的，它是一个隐含形参，对各成员的访问均通过this进行。

(2) 程序运行过程与结果分析：显示地“得瑟”应用this指针。

定义了一个名为A的类，包含私有变量x，构造函数用于给x赋值。this指针是指向当前对象实例的指针，因此控制台打印出来的this实际上是当前对象实例a、b、c的地址，而使用this->x可以打印当前实例的x成员。

```
#include <iostream>
using namespace std;
class A {
public:
    A(int x1) { x = x1; }    //构造函数，有木有？
    void disp() {
        cout << "\n this = " << this << "when x =" << this->x;
    }
private:
    int x;
};
int main() {
    A a(1), b(2), c(3);    //对象初始化
    a.disp();
    b.disp();
    c.disp();
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
this = 010FFC2Cwhen x =1
this = 010FFC20when x =2
this = 010FFC14when x =3
```



同样的，也可以隐式地使用this。

```
#include <iostream>
using namespace std;
class A {
public:
    A(int x1) { x = x1; }    //构造函数，有木有？
    void disp() {
        cout << "this = " << this << " when x = " << x << endl;
    }
private:
    int x;
};
int main() {
    A a(1), b(2), c(3);    //对象初始化
    a.disp();
    b.disp();
    c.disp();
    return 0;
}
```

### 三、 附录

源程序文件项目清单： 4.1 5.1 5.2 5.3 6.1 7.1