

D1 文档

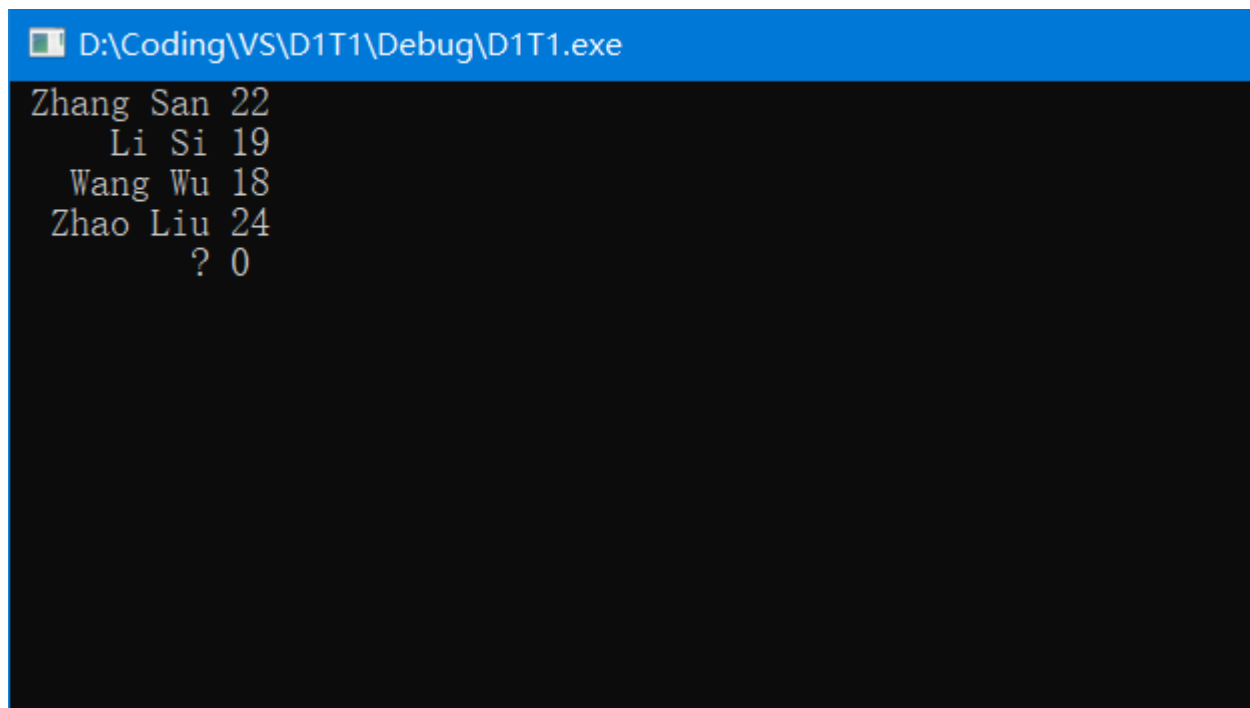
一.第一题&第二题

1.问题描述:

- (1) 为足球队员对象定义一个类使得可以自动输出队员信息
- (2) 完成查询队员年龄的功能

2.运行示例:

(1)



```
D:\Coding\VS\D1T1\Debug\D1T1.exe
Zhang San 22
Li Si 19
Wang Wu 18
Zhao Liu 24
? 0
```

(2)

D:\Coding\VS\D1T2\Debug\D1T2.exe

```
Zhang San 22
    Li Si 19
    Wang Wu 18
    Zhao Liu 24
        ? 0

22
19
18
24
0
```

3.代码分析

本题的main()函数已经给定，因此没有编写相应的测试函数。主要考查了类的构造函数和运算符“<<”和“[]”的重载。

在第一题中，由于主函数的newCommers数组只显示构造了四个对象，因此考虑两种实现方法，一是自行编写默认构造函数，而是考虑在构造函数中对参数赋予初始值，我选择了后一种方法。

对于“<<”运算符的重载，由于操作符的第一个运算量不是当前类，所以不能将它作为本类的成员函数重载，考虑在类外作为友元函数重载。

对于“[]”，考虑作为类内部的成员函数重载。

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

class Member
{
    friend ostream & operator<<(ostream &out, Member &obj);
private:
    string name;
    int age;
public:
    Member(string n = "?", int a = 0)
    {
        name = n;
        age = a;
    }
    string getName() { return name; }
    int getAge() { return age; }
```

```

};

ostream & operator<<(ostream &out, Member &obj)
{
    out << setw(10) << obj.name << " " << obj.age;
    return out;
}

//-----此部分为T1.1内容-----//

class MemberList
{
private:
    Member * member;
    int num;
public:
    MemberList(Member* m, int n)
    {
        member = m;
        num = n;
    }

    int& operator[](string n)
    {
        int age = 0;
        for (int i = 0; i < num; i++)
        {
            if (n == member[i].getName())
            {
                age = member[i].getAge();
                break;
            }
        }
        return age;
    }
};

//-----此部分为T1.2内容-----//

int main()
{
    Member newCommers[5] = { Member("Zhang San", 22),
        Member("Li Si", 19),
        Member("Wang Wu", 18),
        Member("Zhao Liu", 24) };
    for (int i = 0; i < 5; i++)
    {
        cout << newCommers[i] << endl;
    }
    string name[5] = { "Zhang San", "Li Si", "Wang Wu", "Zhao Liu", "Pin Yin" };
    MemberList list(newCommers, 5);
    for (int i = 0; i < 5; i++)
    {
        cout << list[name[i]] << endl;
    }
    while (1);
}

```

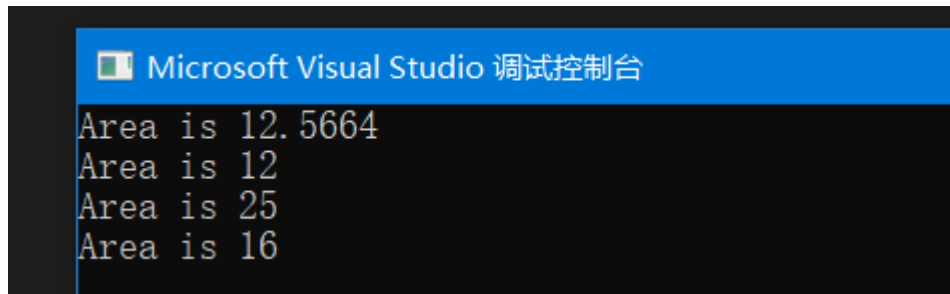
```
    return 0;
}
```

二.第二题

1.问题描述:

在不改变main()函数的条件下补充代码，计算圆形、长方形、正方形的面积。

2.运行示例:



```
Microsoft Visual Studio 调试控制台
Area is 12.5664
Area is 12
Area is 25
Area is 16
```

3.代码分析:

根据main函数分析可得，通过shape指针访问了不同形状各自的getarea() 函数，是典型的多态行为，而shape类本身并不实例化对象，因此创建抽象类shape()类，派生出各形状类，并编写各自的getarea()函数。

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

class Shape
{
public:
    virtual double getarea() = 0;           //纯虚函数
};

class Circle : public Shape
{
private:
    double r;
public:
    Circle(double r) :r(r) {}
    double getarea() { return 3.1416*r*r; }
};

class Rectangle : public Shape
{
private:
    double w, l;
public:
    Rectangle(double w, double l) :w(w), l(l) {}
    double getarea() { return w * l; }
};

class Square : public Shape
```

```

{
private:
    double a;
public:
    Square(double a) :a(a) {}
    double getarea() { return a * a; }
};

int main(int argc, char **argv) {
    Shape * shapes[4];
    Circle circle(2.0);
    Rectangle rectangle(3.0, 4.0);
    Square square1(5.0);
    Square square2(4.0);
    shapes[0] = &circle;
    shapes[1] = &rectangle;
    shapes[2] = &square1;
    shapes[3] = &square2;
    for (int k = 0; k<4; k++) {
        cout << "Area is " << shapes[k]->getarea() << endl;
    }
    return 0;
}

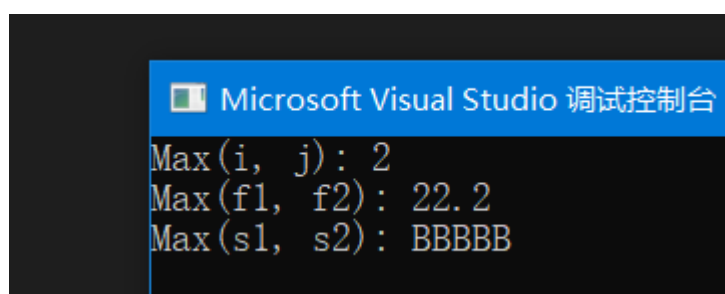
```

三.第三题

1.问题描述:

添加一个模板函数Max，使得程序能够返回正确的结果。要求不改变main()函数。

2.运行示例:



```

Microsoft Visual Studio 调试控制台
Max(i, j): 2
Max(f1, f2): 22.2
Max(s1, s2): BBBBB

```

3.代码分析:

```

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

template<typename T>                                //创建模板函数
T Max(const T& left, const T& right)
{
    return left > right ? left : right;
}

```

```
}

int main()
{
    int i = 1;
    int j = 2;
    cout << "Max(i, j): " << Max(i, j) << endl;
    double f1 = 11.1;
    double f2 = 22.2;
    cout << "Max(f1, f2): " << Max(f1, f2) << endl;
    string s1 = "AAAAA";
    string s2 = "BBBBB";
    cout << "Max(s1, s2): " << Max(s1, s2) << endl;
    return 0;
}
```

四.总结

主要复习了面向对象设计的一些基本概念，如类，继承，重载，多态等，这些内容之前已经学过，本次作业对以上概念进行了一些复习，总体来说较为简单。