

《面向对象程序设计》

实验报告

|  |  |
| --- | --- |
| 班 级： | **计222** |
| 学 号： | **22013166** |
| 姓 名： | **杨建鑫** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年11月

**实验六、综合实验**

**【实验目的】**

1、掌握虚函数和抽象类

2、理解和掌握基类与公有派生类的赋值兼容

3、理解并掌握利用类模板和函数模板方法

**【实验内容】**

1、定义一个抽象类，采用虚函数，计算三角形、矩形、圆形、扇形的面积。

**Code:**

#include <iostream>

#include <cmath>

const double PI = 3.14159;

*// 定义抽象类Shape*

class Shape

{

public:

    virtual double area() = 0; *// 纯虚函数*

};

*// 定义Triangle类*

class Triangle : public Shape

{

public:

    Triangle(double *base*, double *height*) : base(*base*), height(*height*) {}

    double area() override

    {

        return 0.5 \* base \* height;

    }

private:

    double base;

    double height;

};

*// 定义Rectangle类*

class Rectangle : public Shape

{

public:

    Rectangle(double *width*, double *height*) : width(*width*), height(*height*) {}

    double area() override

    {

        return width \* height;

    }

private:

    double width;

    double height;

};

*// 定义Circle类*

class Circle : public Shape

{

public:

    Circle(double *radius*) : radius(*radius*) {}

    double area() override

    {

        return PI \* radius \* radius;

    }

private:

    double radius;

};

*// 定义Sector类*

class Sector : public Shape

{

public:

    Sector(double *radius*, double *angle*) : radius(*radius*), angle(*angle*) {}

    double area() override

    {

        return 0.5 \* radius \* radius \* angle;

    }

private:

    double radius;

    double angle;

};

int main()

{

    Triangle t(3.0, 4.0);

    Rectangle r(3.0, 4.0);

    Circle c(3.0);

    Sector s(3.0, PI / 6); *// 30度扇形*

    std::cout << "Triangle area: " << t.area() << std::endl;

    std::cout << "Rectangle area: " << r.area() << std::endl;

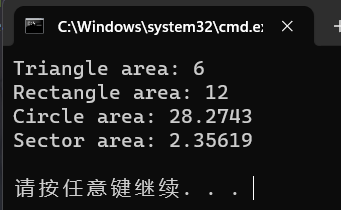
    std::cout << "Circle area: " << c.area() << std::endl;

    std::cout << "Sector area: " << s.area() << std::endl;

    return 0;

}

**Example：**



2、根据以下要求，完成编程。

1）定义一个类Animal,该类可以存放动物名字，并且有成员函数Identify()用于显示动物名字（输出: I am an animal：动物名字）

2）定义两个类Cat、Dog,都继承自Animal，并且都重载了Identify(),显示动物种类和名字（分别输出：I am a cat：动物名字； I am a dog：动物名字）

3)定义类Tiger,继承自类Cat，重载Identify(),显示输出：I am a tiger: 动物名字

4）定义一个动物园类Zoo，根据以下的提示编写动物类。注意指针指向虚函数的用法。

class Zoo

{

public:

Zoo(int max); //max为动物园最多能圈养动物数

~Zoo(){ delete residents; }

void Accept(Animal \*d); //接受动物

void ListAnimal(); //显示动物园所有的动物

private:

int max\_animals; //动物园能圈养的动物数

int num\_animals; //动物园动物园当前圈养的动物数

Animal \*\*residents; //指向动物园圈养的动物对象的指针数组

};

编程要求：创建三个对象Cat类对象 cat("汤姆猫")、Dog类对象dog("汪汪狗")、Tiger类对象tiger("贝乐虎"),先显示各自类的Identify()。再创建Zoo类对象shanghai, 将三只动物依次圈养到shanghai动物园,并依次输出当前圈养的动物信息。

Code：

#include <iostream>

#include <string>

class Animal

{

public:

    Animal(const std::string &*name*) : name(*name*) {}

    virtual void Identify()

    {

        std::cout << "I am an animal: " << name << std::endl;

    }

protected:

    std::string name;

};

class Cat : public Animal

{

public:

    Cat(const std::string &*name*) : Animal(*name*) {}

    void Identify() override

    {

        std::cout << "I am a cat: " << name << std::endl;

    }

};

class Dog : public Animal

{

public:

    Dog(const std::string &*name*) : Animal(*name*) {}

    void Identify() override

    {

        std::cout << "I am a dog: " << name << std::endl;

    }

};

class Tiger : public Cat

{

public:

    Tiger(const std::string &*name*) : Cat(*name*) {}

    void Identify() override

    {

        std::cout << "I am a tiger: " << name << std::endl;

    }

};

class Zoo

{

public:

    Zoo(int *max*) : max\_animals(*max*), num\_animals(0)

    {

        residents = new Animal \*[*max*];

    }

    ~Zoo() { delete[] residents; }

    void Accept(Animal \**d*)

    {

        if (num\_animals < max\_animals)

        {

            residents[num\_animals++] = *d*;

        }

    }

    void ListAnimals()

    {

        for (int i = 0; i < num\_animals; ++i)

        {

            residents[i]->Identify();

        }

    }

private:

    int max\_animals;

    int num\_animals;

    Animal \*\*residents;

};

int main()

{

    Cat cat("汤姆猫");

    Dog dog("汪汪狗");

    Tiger tiger("贝乐虎");

    cat.Identify();

    dog.Identify();

    tiger.Identify();

    Zoo shanghai(3);

    shanghai.Accept(&cat);

    shanghai.Accept(&dog);

    shanghai.Accept(&tiger);

    shanghai.ListAnimals();

    return 0;

}

Example：

文本

描述已自动生成

3、创建一个模板函数。该函数能对数组进行排序。采用冒泡法排序。要求分别对int数组和float数组进行排序。

Code：

#include <iostream>

template <typename T>

void BubbleSort(T *arr*[], int *n*)

{

    for (int i = 0; i < *n* - 1; ++i)

    {

        for (int j = 0; j < *n* - 1 - i; ++j)

        {

            if (*arr*[j] > *arr*[j + 1])

            {

                T temp = *arr*[j];

*arr*[j] = *arr*[j + 1];

*arr*[j + 1] = temp;

            }

        }

    }

}

template <typename T>

void PrintArray(T *arr*[], int *n*)

{

    for (int i = 0; i < *n*; ++i)

    {

        std::cout << *arr*[i] << " ";

    }

    std::cout << std::endl;

}

int main()

{

    int arr1[] = {4, 3, 2, 1};

    float arr2[] = {4.4, 3.3, 2.2, 1.1};

    BubbleSort(arr1, 4);

    BubbleSort(arr2, 4);

    PrintArray(arr1, 4);

    PrintArray(arr2, 4);

    return 0;

}

Example：

文本

描述已自动生成

4、创建一个模板类。该类可以装入一个数据，并能读出显示存入的数据。该数据可以是

int、 float、 char、 Student自定义的类)。类Student包含姓名和1个分数（提示：重载>>输出运算符）。

Code：

#include <iostream>

#include <string>

*// 自定义的Student类*

class Student

{

public:

    Student(const std::string &*name*, int *score*) : name(*name*), score(*score*) {}

    friend std::ostream &operator<<(std::ostream &*out*, const Student &*s*)

    {

*out* << "Name: " << *s*.name << ", Score: " << *s*.score;

        return *out*;

    }

private:

    std::string name;

    int score;

};

*// 模板类*

template <typename T>

class Box

{

public:

    Box(const T &*data*) : data(*data*) {}

    void Display()

    {

        std::cout << data << std::endl;

    }

private:

    T data;

};

int main()

{

    Box<int> box1(123);

    Box<float> box2(456.789f);

    Box<char> box3('A');

    Box<Student> box4(Student("Tom", 90));

    box1.Display();

    box2.Display();

    box3.Display();

    box4.Display();

    return 0;

}

Example：

文本

低可信度描述已自动生成