day10_刘立京

继承简答题

1. 三种继承方式对于基类成员的访问权限是怎样的

> `public`：派生类可访问基类`public`、`protected`成员，并且不改变其在派生类的权限
> `protected`：派生类可访问基类`public`、`protected`成员，并且将两者置为`protected`成员，在派生类定义中可以访问，在对象中不可访问
> `private`：派生类可访问基类`public`、`protected`成员，并且将二者置为`private`成员，在派生类定义中可以访问，在对象中不可访问

继承方式	基类成员	在子类中访问权限	子类内部模块访问性	子类对象访问性
公有继承	公有成员 保护成员 私有成员	公有的 保护的 不可访问	可以访问 可以访问 不可访问	可以访问 不可访问 不可访问
私有继承	公有成员 保护成员 私有成员	私有的 私有的 不可访问	可以访问 可以访问 不可访问	不可访问 不可访问 不可访问
保护继承	公有成员 保护成员 私有成员	保护的 保护的 不可访问	可以访问 可以访问 不可访问	不可访问 不可访问 不可访问

2. 继承中有哪些内容是不能进行继承的

> 友元函数
> 构造函数、析构函数
> 重载的`new/delete`、`=`赋值成员函数

3. 多基派生会产生的问题有哪些？怎样解决

> 可能会出现钻石型继承，需要定义为虚继承处理
> 成员名二义性的问题，需要限定作用域

4. 派生类对象之间的复制控制规则是什么

> 派生类对象构造函数递归地向上调用基类构造函数，尤其在带参数构造函数中需要显式调用基类带参构造函数，然后执行自己的初始化过程
> 派生类对象析构函数递归地调用基类的析构函数，先进行派生类析构函数销毁，再进行基类析构函数销毁
> 如果没有定义复制控制函数会自动调用基类复制控制函数 如果显式的定义了复制控制函数，必须显式调用基类复制控制函数

代码实现

1. 编写一个圆类Circle，该类拥有

① 1个成员变量，存放圆的半径；
② 两个构造方法
Circle( ) // 将半径设为0
Circle(double r) //创建Circle对象时将半径初始化为r
③ 三个成员方法
double getArea( ) //获取圆的面积
double getPerimeter( ) //获取圆的周长
void show( ) //将圆的半径、周长、面积输出到屏幕

源代码如下：

```
#include <iostream>

#define PI 3.141592654
using std::endl;
using std::cin;
using std::cout;

class Circle
{
public:
Circle()
: _radius(0)
{}
Circle(double r)
: _radius(r)
{}

double getArea()
{
        return _radius * _radius * PI;
}
double getPerimeter()
{
        return 2 * PI * _radius;
}

void show()
{
        cout<<_radius<<' '<<getPerimeter()<<' '<<getArea();
}

private:
double _radius;
};

int main()
{
Circle x;
x = 4;
x.show();
```

```
cout<<endl;
return 0;
}
```

运行结果：

```
Jims-MacBook-Pro:day10 jimlau$ make
g++ -Wall src/Circle.cc -o Circle
Jims-MacBook-Pro:day10 jimlau$ ./Circle
4 25.1327 50.2655
```

2. 编写一个圆柱体类Cylinder，它继承于上面的Circle类，还拥有

① 1个成员变量，圆柱体的高； ② 构造方法
Cylinder (double r, double h) //创建Circle对象时将半径初始化为r
③ 成员方法
double getVolume( ) //获取圆柱体的体积
void showVolume( ) //将圆柱体的体积输出到屏幕
编写应用程序，创建类的对象，分别设置圆的半径、圆柱体的高，计算并分别显示圆半径、圆面积、圆周长，圆柱体的体积。

```
#include "../include/Circle.hh"

class Cylinder: protected Circle
{
public:
    Cylinder (double r, double  h) //创建Circle对象时将半径初始化为r
    : Circle(r), _height(h){}
    double getVolume() //获取圆柱体的体积
    {
        return getArea() * _height;
    }
    void  showVolume() //将圆柱体的体积输出到屏幕
    {
        Circle::show();
        cout<<' '<<getVolume();
    }
private:
    double _height;
};

int main()
{
    Cylinder cc(4.1, 3.7);
    cc.showVolume();
    cout<<endl;
    return 0;
}
```

运行结果：

```
Jims-MacBook-Pro:day10 jimlau$ make
g++ -Wall src/Cylinder.cc -o Cylinder
Jims-MacBook-Pro:day10 jimlau$ ./Cylinder
4.1 25.7611 52.8102 195.398
Jims-MacBook-Pro:day10 jimlau$
```

3. 构建一个类person，包含字符串成员name（姓名），整型数据成员age（年龄），成员函数 `display()`用来输出name和age。构造函数包含两个参数，用来对name和age初始化。

构建一个类employee由person派生，包含department（部门），实型数据成员salary（工资）,成员函数display（）用来输出职工姓名、年龄、部门、工资，其他成员根据需要自己设定。

主函数中定义3个employee类对象，内容自己设定，将其姓名、年龄、部门、工资输出， 并计算他们的平均工资。

```
#include <iostream>

using std::cout;
using std::cin;
using std::endl;
using std::string;

class person
{
public:
    person()
    : _name(), _age(-1){}
    person(string nm, int age)
    : _name(nm), _age(age){}
    virtual void display()
    {
        cout<<_name<<' '<<_age<<' ';
    }
private:
    string _name;
    int _age;
};

class employee: public person
{
public:
    employee()
    : person()
    , _department(), _salary(-1)
    {}
```

```
    employee(string nm, int age, string depart, double salary)
    : person(nm, age)
    , _department(depart)
    , _salary(salary)
    {}
    void display()
    {
        person::display();
        cout<<_department<<' '<<_salary<<endl;
    }
    double getSalary()
    {
        return _salary;
    }
private:
    string _department;
    double _salary;
};

int main()
{
    employee g, t, q;
    g = employee("guang", 12, "MTA", 2200.63);
    t = employee("tou", 21, "DNT", 16003.89);
    q = employee("qiang", 35, "DNT", 54223.49);
    person& m1 = g;
    m1.display();
    g.display();
    t.display();
    q.display();
    cout<<"avg salary = "<<(g.getSalary() + t.getSalary() + q.getSalary())
/ 3<<endl;
    return 0;
}
```

运行结果:

```
Jims-MacBook-Pro:day10 jimlau$ ./person
guang 12 MTA 2200.63
guang 12 MTA 2200.63
tou 21 DNT 16003.9
qiang 35 DNT 54223.5
avg salary = 24142.7
Jims-MacBook-Pro:day10 jimlau$
```

4. 魔兽世界之二:装备版

mooc魔兽世界之二作业

魔兽世界(装备)Ac代码如下：

```
#include <iostream>
#include <vector>
#include <map>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::vector;
using std::map;

#define whichArm(arm)(arm == 0?"sword":(arm == 1?"bomb":"arrow"))

enum WORR
{
    Dra=0,
    Nin=1,
    Ice=2,
    Lio=3,
    Wol=4
};

class worrior
{
protected:
    worrior(int no, int strength)
    : _wno(no), _strength(strength){}
    void getInfo(int &Strength, int &WNo)const
    {
        Strength = _strength;
        WNo = _wno;
    }
private:
    int _wno;
    int _strength;

};

class Dragen: public worrior
{
public:
    Dragen(int strength, int num, int total)
    : worrior(num, strength)
    , _morale(((double)total - strength) / strength)
    , _arm(num % 3)
    {}
    void getInfo(int &Strength, int &WNo, double &Morale, uint8_t &Arm)
const
    {
        worrior::getInfo(Strength, WNo);
        Morale = _morale;
```

```
        Arm = _arm;
    }
private:
    double _morale;
    uint8_t _arm;
};

class Ninja: public worrior
{
public:
    Ninja(int strength, int num)
    : worrior(num, strength)
    , _mainArm(num % 3)
    , _secArm((num + 1) % 3)
    {}
    void getInfo(int &Strength, int &WNo, uint8_t &MainArm, uint8_t
&SecArm) const
    {
        worrior::getInfo(Strength, WNo);
        MainArm = _mainArm;
        SecArm = _secArm;
    }

private:
    uint8_t _mainArm;
    uint8_t _secArm;
};

class Iceman: public worrior
{
public:
    Iceman(int strength, int num)
    : worrior(num, strength)
    , _arm(num % 3)
    {}
    void getInfo(int &Strength, int &WNo, uint8_t &Arm)const
    {
        worrior::getInfo(Strength, WNo);
        Arm = _arm;
    }
private:
    uint8_t _arm;
};

class Lion: public worrior
{
public:
    Lion(int strength, int num, int total)
    : worrior(num, strength)
    , _loyalty(total - strength)
    {}
    void getInfo(int &Strength, int &WNo, int &Loyalty)const
    {
        worrior::getInfo(Strength, WNo);
```

```
        Loyalty = _loyalty;
    }
private:
    int _loyalty;
};

class Wolf: public worrior
{
public:
    Wolf(int strength, int num)
    : worrior(num, strength)
    {}
    void getInfo(int &Strength, int &WNo)const
    {
        worrior::getInfo(Strength, WNo);
    }
};

class center
{
public:
    center()
    : _n(0)
    {
        cin>>_M;
        cin>>_dStrength>>_nStrength>>_iStrength>>_lStrength>>_wStrength;
    }
    center(const center& cent)
    : _M(cent._M)
    , _dStrength(cent._dStrength)
    , _nStrength(cent._nStrength)
    , _iStrength(cent._iStrength)
    , _lStrength(cent._lStrength)
    , _wStrength(cent._wStrength)
    , _n(cent._n)
    {}
    Dragen& makeDragen()
    {
        dStroop.push_back(Dragen(_dStrength, ++_n, _M));
        _M -= _dStrength;
        return dStroop.back();
    }
    Ninja& makeNinja(){nStroop.push_back(Ninja(_nStrength, ++_n)); _M -=
_nStrength; return nStroop.back();}
    Iceman& makeIceman(){iStroop.push_back(Iceman(_iStrength, ++_n)); _M -
= _iStrength; return iStroop.back();}
    Lion& makeLion(){lStroop.push_back(Lion(_lStrength, ++_n, _M)); _M -=
_lStrength; return lStroop.back();}
    Wolf& makeWolf(){wStroop.push_back(Wolf(_wStrength, ++_n)); _M -=
_wStrength; return wStroop.back();}

    int MakeinLog(const char* centerName, const WORR worrType, const int
time)
    {
```

```
        int num; int strength;
        switch (worrType)
        {
        case Dra:
        {
            if(_M < _dStrength)
                return -1;
            double morale; uint8_t arm;
            makeDragen().getInfo(strength, num, morale, arm);
            genLog(time, centerName, "dragon", num, strength,
dStroop.size());
            printf("It has a %s,and it's morale is %.2lf\n",
whichArm(arm), morale);
            break;
        }
        case Nin:
            if(_M < _nStrength)
                return -1;
            uint8_t mainArm, secArm;
            makeNinja().getInfo(strength, num, mainArm, secArm);
            genLog(time, centerName, "ninja", num, strength,
nStroop.size());
            printf("It has a %s and a %s\n", whichArm(mainArm),
whichArm(secArm));
            break;
        case Ice:
            if(_M < _iStrength)
                return -1;
            uint8_t arm;
            makeIceman().getInfo(strength, num, arm);
            genLog(time, centerName, "iceman", num, strength,
iStroop.size());
            printf("It has a %s\n", whichArm(arm));
            break;
        case Lio:
            if(_M < _lStrength)
                return -1;
            int loyalty;
            makeLion().getInfo(strength, num, loyalty);
            genLog(time, centerName, "lion", num, strength,
lStroop.size());
            printf("It's loyalty is %d\n", loyalty);
            break;
        case Wol:
            if(_M < _wStrength)
                return -1;
            makeWolf().getInfo(strength, num);
            genLog(time, centerName, "wolf", num, strength,
wStroop.size());
            break;
        default:
            break;
        }
        return 0;
```

```
    }

private:
    inline void genLog(int time, const char* centerName, const char*
WorriorName, int num, int strength, size_t size)
    {
        printf("%03d %s %s %d born with strength %d,%ld %s in %s
headquarter\n", time, centerName, WorriorName, num, strength, size,
WorriorName, centerName);
    }
    int _M;
    int _dStrength, _nStrength, _iStrength, _lStrength, _wStrength;
    int _n;
    vector<Dragen> dStroop;
    vector<Ninja> nStroop;
    vector<Iceman> iStroop;
    vector<Lion> lStroop;
    vector<Wolf> wStroop;
};

/* 控制center制造勇士，直到不够制造最小生命值的勇士 */
void rollAll(center red, center blue)
{
    map<int, WORR> redOrder, blueOrder;
    redOrder[0] = Ice; redOrder[1] = Lio; redOrder[2] = Wol; redOrder[3] =
Nin; redOrder[4] = Dra;
    blueOrder[0] = Lio; blueOrder[1] = Dra; blueOrder[2] = Nin;
blueOrder[3] = Ice; blueOrder[4] = Wol;

    int iRed = 0, iBlue = 0;
    int bOK = 0, rOK = 0;
    int t = 0;
    uint8_t pBBit = 0, pRBit = 0;
    while(1)
    {
        for (int cnt = 0, right = -1; rOK == 0 && right == -1; iRed =
(iRed + 1) % 5)
        {
            right = red.MakeinLog("red", redOrder[iRed], t);
            if(right) ++cnt;
            if(cnt == 5) rOK = -1;
        }
        if(rOK && pRBit == 0)
        {
            printf("%03d %s headquarter stops making warriors\n", t,
"red");
            pRBit = 1;
        }

        for (int cnt = 0, right = -1; bOK == 0 && right == -1; iBlue =
(iBlue + 1) % 5)
        {
            right = blue.MakeinLog("blue", blueOrder[iBlue], t);
            if(right) ++cnt;
```

```
            if(cnt == 5) bOK = -1;
        }
        if(bOK && pBBit == 0)
        {
            printf("%03d %s headquarter stops making warriors\n", t,
"blue");
            pBBit = 1;
        }

        if(bOK && rOK)
            break;
        ++t;
    }
}

int main()
{
    int n;
    cin>>n;
    for (int i = 0; i < n; ++i)
    {
        printf("Case:%d\n", i + 1);
        center red, blue(red);
        rollAll(red, blue);
    }
    return 0;
}
```