# 计算机学院 高级程序语言设计 课程实验报告

| 实验题目：类的继承 | | 学号：202300130150 |
|---|---|---|
| 日期：2024.4.9 | 班级： 4 | 姓名： 王成意 |

实验目的：
练习类的继承

实验步骤与内容：
1. 通过下列描述进行类的设计
   描述略
设计内容如下：

```
class vehicle

{

private:

    const float speed;

    float location;



public:

    vehicle(float speed, float location = 0) :

speed(speed)

    {

        this->location = location;

    }

    void drive(float duration)

    {

        this->location += duration * this->speed;
```

```
        printf("Location now:%f\n", this->location);
    }
};

class truck : public vehicle
{
private:
    const float load_capacity;
    float load_weight;

public:
    truck(float load_capacity, float speed, float
load_weight = 0) : load_capacity(load_capacity),
vehicle(speed)
    {
        this->load_weight = load_weight;
    }
    bool load(float w)
    {
        if (this->load_weight + w <
this->load_capacity)
        {
```

```
            this->load_weight += w;
            return true;
        }
        else
        {
            printf("Operation load %f:beyond
capacity!\n", w);
            return false;
        }
    }
    bool unload(float w)
    {
        if (this->load_weight - w > 0)
        {
            this->load_weight -= w;
            return true;
        }
        else
        {
            printf("Operation unload %f:%f >=
load_weight!", w, w);
            return false;
```

```
        }
    }
    float getLoad_weight()
    {
        return this->load_weight;
    }
};

class refrigeratorCar : public vehicle
{
private:
    float temperature;
    const float room_temperature;

public:
    refrigeratorCar(float startem, float speed) :
room_temperature(startem), vehicle(speed) {}
    void reset()
    {
        this->temperature = this->room_temperature;
    }
    void setTemperature(float t)
```

```
    {
        this->temperature = t;
    }
    float getTemperature()
    {
        return this->temperature;
    }
};

class refrigeratorTruck : public refrigeratorCar,
public truck
{
    private:
        ;
    public:
        refrigeratorTruck(float speed,float
startem,float
capacity):truck(capacity,speed),refrigeratorCar(starte
m,speed){}
        bool load(float w,float t)
        {
            if(!truck::load(w))
```

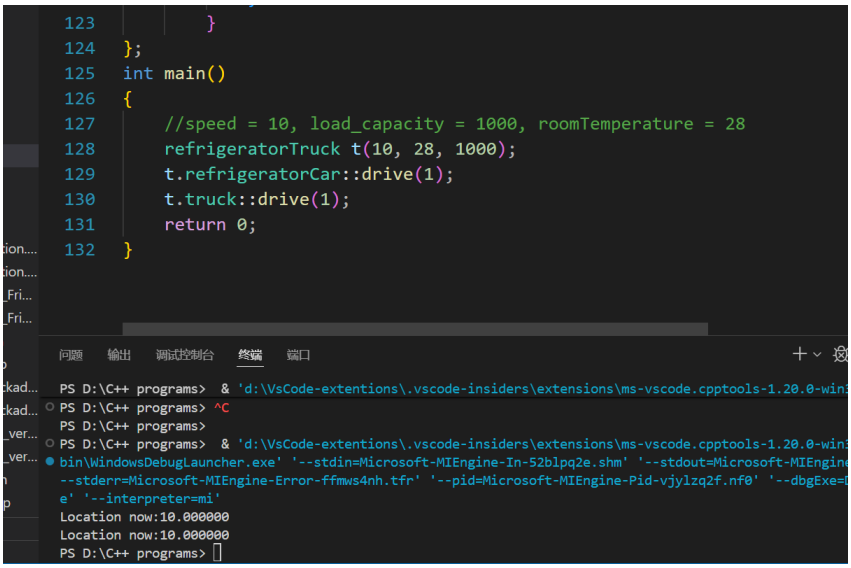
```
        {
            return false;
        }
        else
        {
            if (t < this->getTemperature())
            {
                this->setTemperature(t);
            }
            return true;
        }

    }
    bool unload(float w)
    {
        if(!truck::unload(w))
        {
            return false;
        }
        else
        {
            if(this->getLoad_weight()==0)
```

```
                {
                    this->reset();
                }
                return true;
            }
        }
};
```

2. 首先不利用虚基类设计上述类，在 main 函数里执行以下指令，截图并回答下列问题。

（1）通过调用 t.RefrigeratorCar::drive 以及 t.Truck::drive 两个函数，分析这两个函数打印的从哪里 drive 到哪里的信息，分析他们之间的关系。对同一辆车来说，这样合理吗？



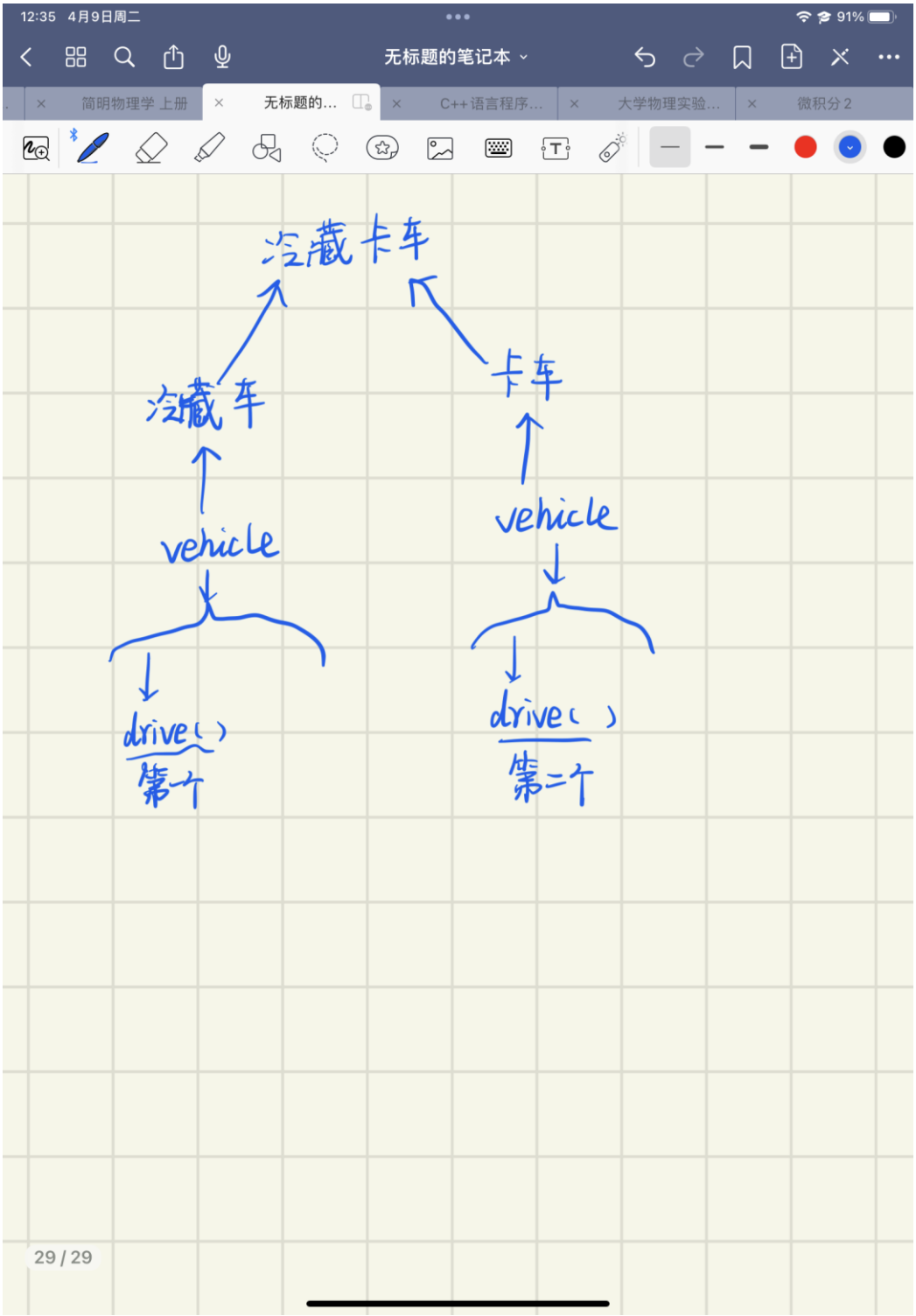

```
t.refrigeratorCar::drive(1):
```

这是从类：refrigeratorCar 继承的 vehicle 类里面的 drive 打印出来的。

```
t.truck::drive(2):
```

这是从类：truck 继承的 vehicle 里面的 drive 打印出来的。

他们的关系如图：




对同一辆车来说，这样显然不合理。

（2）可以直接调用 t.drive(1)吗？为什么？

不能，因为 RefrigeratorTruck 继承的两个类都有 vehicle 会造成二义性。事实上，如果调用会出现以下错误：

```
.exe" -std=c++17
D:\C++ programs\1.cpp: In function 'int main()':
D:\C++ programs\1.cpp:129:4: error: request for member 'drive' is ambiguous
  t.drive(1);
    ^~~~~
D:\C++ programs\1.cpp:15:7: note: candidates are: 'void vehicle::drive(float)'
  void drive(float duration)
       ^~~~~
D:\C++ programs\1.cpp:15:7: note:                 'void vehicle::drive(float)'
```

在不用虚基类的时候，可以像下面这样对他父类的父类 Vehicle 进行构造吗？会报什么错误？

尝试：将 RefrigeratorTruck 的构造函数更改为：

```
refrigeratorTruck(float speed, float startem, float
capacity) : vehicle(speed), truck(capacity, speed),
refrigeratorCar(startem, speed)
    {
        cout << "RefrigeratorTruck is constructed" <<
endl;
    }
```

运行结果：

```
正在启动生成...
cmd /c chcp 65001>nul && D:\mingw64\bin\g++.exe -fdiagnostics-color=always -g "D:\C++ programs\1.cpp" -o "D:\C++ program
.exe" -std=c++17
D:\C++ programs\1.cpp: In constructor 'refrigeratorTruck::refrigeratorTruck(float, float, float)':
D:\C++ programs\1.cpp:93:66: error: type 'vehicle' is not a direct base of 'refrigeratorTruck'
  refrigeratorTruck(float speed, float startem, float capacity) : vehicle(speed), truck(capacity, speed), refrigeratorCa
tartem, speed)
                                                                 ^~~~~~~
```

type 'vehicle' is not a direct base of 'refrigeratorTruck'。RefrigeratorTruck 不直接继承 vehicle 类。

3. 之后利用虚基类设计上述类，在 main 函数里执行以下指令，截图并回答下列问题。

更改部分：

```
class truck : virtual public vehicle
{
private:
    const float load_capacity;
    float load_weight;

public:
```

```
class refrigeratorCar : virtual public vehicle
{
private:
    float temperature;
    const float room_temperature;

public:
```

以及：

```
refrigeratorTruck(float speed, float startem, float

capacity) : truck(capacity, speed),

refrigeratorCar(startem, speed), vehicle(speed)
```

（太长了截图截不过来）

再次运行：

```
132 };
133 int main()
134 {
135     // speed = 10, load_capacity = 1000, roomTemperature = 28
136     refrigeratorTruck t(10, 28, 1000);
137     t.refrigeratorCar::drive(1);
138     t.truck::drive(1);
139     t.drive(1);
140     return 0;
141 }
```

```
问题  输出  调试控制台  终端  端口
xp41sjmj.5a2' '--stdout=Microsoft-MIEngine-Out-k4lz2r0e.wj1' '--stderr=Microsoft-MIEngine-Error
xe' '--interpreter=mi'
vehicle is constructed
RefrigeratorCar is constructed
truck is constructed
RefrigeratorTruck is constructed
Location now:10.000000
Location now:20.000000
Location now:30.000000
```

（1） 通过调用 t.RefrigeratorCar::drive 以及 t.Truck::drive 两个函数，分析这两个函数打印的从哪里 drive 到哪里的信息，分析他们之间的关

系。对同一辆车来说，现在合理吗？
t.RefrigeratorCar::drive 打印的是虚基类：vehicle 的信息，而 t.Truck::drive 打印的也是虚基类：vehicle 的信息，他们是等价的。对同一辆车来说，现在合理了。

4. 在载具类的 drive 函数中除了打印从哪里 drive 到哪里的信息之外，再通过 cout<<&location<<endl;打印一下 location 变量的地址，分析用不用虚基类两种情况下 t.RefrigeratorCar::drive(1);以及 t.Truck::drive(1);两次调用打印的 location 变量的地址有何变化。用虚基类之后这两个地址有什么关系？t.drive(1);这次调用打印的结果与前两次巧用打印的变量地址有什么关系？试着分许他们的关系和虚基类在这里所起的作用。

使用虚基类：

```
Location now:10.000000
0x62fe14
Location now:20.000000
0x62fe14
Location now:30.000000
0x62fe14
```

不使用虚基类（也不执行 t.drive(1)）：

```
Location now:10.000000
0x62fe04
Location now:10.000000
0x62fe14
PS D:\C++ programs>
```

由此可见，使用虚基类之后这两个地址相同。
t.drive(1)打印变量的地址与前两次也相同。这意味着虚基类在这里起的作用是将不同类继承的同一虚基类共享数据。

5. 在这个例子中，父类的数据成员应该被设置为什么类型的？私有、保护、还是公有的？说出你的理由。

我认为应该设置成私有的，这样可以保证数据相互独立不会彼此牵制，同时只需新增 get 和 set 函数接口对内部数据进行操作即可，这样也保证了程序的可读性。

6. 在各个类的构造函数中打印信息通报是哪个类被构造了，在析构函数中打印信息通报是哪个类被析构了。分析利用虚基类和不利用虚基类两种情况下构造函数和析构函数的调用顺序。
使用虚基类：

```
vehicle is constructed
RefrigeratorCar is constructed
truck is constructed
RefrigeratorTruck is constructed
Location now:10.000000
0x62fe04
Location now:20.000000
0x62fe04
Location now:30.000000
0x62fe04
refrigeratorTruck is destructed.
truck is destructed.
refrigeratorCar is destructed.
vehicle is destructed.
```

参数表中的顺序：

```
refrigeratorTruck(float speed, float startem, float

capacity) : truck(capacity, speed),

refrigeratorCar(startem, speed), vehicle(speed)
```

更改参数表顺序：

```
vehicle(speed) ,truck(capacity, speed),

refrigeratorCar(startem, speed)
```

结果不变：

```
 xe' '--interpreter=mi'
 vehicle is constructed
 RefrigeratorCar is constructed
 truck is constructed
 RefrigeratorTruck is constructed
 Location now:10.000000
 0x62fe04
 Location now:20.000000
 0x62fe04
 Location now:30.000000
 0x62fe04
 refrigeratorTruck is destructed.
 truck is destructed.
 refrigeratorCar is destructed.
 vehicle is destructed.
PS D:\C++ programs>
```

可见参数表对其无影响。

更改继承顺序：

```
public truck, public refrigeratorCar
```

发现：

```
vehicle is constructed
truck is constructed
RefrigeratorCar is constructed
RefrigeratorTruck is constructed
Location now:10.000000
0x62fe04
Location now:20.000000
0x62fe04
Location now:30.000000
0x62fe04
refrigeratorTruck is destructed.
refrigeratorCar is destructed.
truck is destructed.
vehicle is destructed.
```

说明对于有虚基类时的构造函数，先构造虚基类，再按照类中继承顺序依次构造。

无虚基类：

```
vehicle is constructed
truck is constructed
vehicle is constructed
RefrigeratorCar is constructed
RefrigeratorTruck is constructed
Location now:10.000000
0x62fe04
Location now:10.000000
0x62fdf4
refrigeratorTruck is destructed.
refrigeratorCar is destructed.
vehicle is destructed.
truck is destructed.
vehicle is destructed.
```

更改参数表为：

```
refrigeratorTruck(float speed, float startem, float

capacity) : refrigeratorCar(startem, speed),

truck(capacity, speed)
```

结果：

```
vehicle is constructed
truck is constructed
vehicle is constructed
RefrigeratorCar is constructed
RefrigeratorTruck is constructed
Location now:10.000000
0x62fe04
Location now:10.000000
0x62fdf4
refrigeratorTruck is destructed.
refrigeratorCar is destructed.
vehicle is destructed.
truck is destructed.
vehicle is destructed.
PS D:\C++ programs>
```

与原来相同。

可见参数表对其无影响。

更改继承顺序：

```
vehicle is constructed
RefrigeratorCar is constructed
vehicle is constructed
truck is constructed
RefrigeratorTruck is constructed
Location now:10.000000
0x62fdf4
Location now:10.000000
0x62fe04
refrigeratorTruck is destructed.
truck is destructed.
vehicle is destructed.
refrigeratorCar is destructed.
vehicle is destructed.
PS D:\C++ programs>
```

有影响。

可见无虚基类时，构造函数按照继承顺序执行。当被继承的类有继承类时，仍然调用被继承类的继承类的构造函数。

对于析构：观察以上图可知，析构函数执行顺序和构造函数相反。

结论分析与体会：

这是一次非常好的实验，使我对于类的继承有了新的理解，一些课程上没弄明白的问题也都在实验中迎刃而解。