小孩子养宠物（示例）

vcg

2021 年 6 月 25 日

1 小孩子养宠物

一个多态的例子。小孩子养了很多种宠物，每种宠物有不同的行为。用多态来描述。CKid 从 CPerson 中派生，两者有特别与一般的联系。CPetBase 是宠物的基类，借助多态机制，可以把小孩子养的宠物们放在 CPetBase* 类型的 vector 容器中。

main.cpp

```cpp
#include <iostream>
#include <string>
#include "kid.h"
#include "dog.h"
#include "cat.h"
#include "horse.h"

using namespace std;

int main()
{
    CKid kid("Du", 19);

    //以下模拟养了一些宠物
    int nPets = 10;
    for (int i = 0; i < nPets; i++)
    {
        int type = rand() % 3;
        CPetBase *pet;
        if (type == 0)
        {
            pet = new CCat();
        }
        else if (type == 1)
        {
            pet = new CDog();
        }
        else
            pet = new CHorse();
        pet->SetID(i);
        kid.KeepPet(pet);
    }

    //看看小孩的宠物们有啥本领
    for (int i = 0; i < kid.PetsNumber(); i++)
    {
        CPetBase *pet = kid.Pet(i);
        pet->SpecialAbility();
    }

    return 0;
}
```

Person.h

```

#pragma once
```

```
#include <string>

class CPerson
{
public:
    CPerson(const std::string &nname="NO", int aage=0) { name = nname, age = aage; }
private:
    std::string      name;
    int              age;
};
```

Kid.h

```
#pragma once
#include <vector>
#include "Person.h"
#include "PetBase.h"

class CKid : public CPerson
{
public:
    CKid(const std::string &nname = "NO", int aage = 0):CPerson(nname, aage) {};

    ~CKid()
    {
        for (unsigned int i = 0; i < pets.size(); i++)delete pets[i];
    }
    void KeepPet(CPetBase * pet)
    {
        pets.push_back(pet);
    }
    int PetsNumber() { return pets.size(); }
    CPetBase *Pet(int n) { return pets[n]; }
private:
    std::vector<CPetBase *> pets;
};
```

PetBase.h

```
#pragma once
#include <string>

class CPetBase
{
public:
    CPetBase(std::string name="Pet") { this->name = name; }
    virtual ~CPetBase() {};

```

```
10      void SetID(int id) { this->id = id; }
11      int ID(int id)const { return id; }
12      virtual void SpecialAbility() = 0;  //显示信息模拟
13  protected:
14      std::string name;
15      int      id;
16  };
```

Cat.h

```
1   #pragma once
2   #include <iostream>
3   #include "PetBase.h"
4
5   class CCat : public CPetBase
6   {
7   public:
8       CCat(std::string name = "Pet") { this->name = name; }
9
10      virtual void SpecialAbility() { std::cout << "喵 ~\n"; };
11  };
```

Dog.h

```
1   #pragma once
2   #include <iostream>
3   #include "PetBase.h"
4
5   class CDog : public CPetBase
6   {
7   public:
8       CDog(std::string name = "Pet") { this->name = name; }
9
10      virtual void SpecialAbility() { std::cout << "汪 ~\n"; };
11  };
```

Horse.h

```
1   #pragma once
2   #include <iostream>
3   #include "PetBase.h"
4
5   class CHorse : public CPetBase
6   {
7   public:
8       CHorse(std::string name = "Horse") { this->name = name; }
9
10      virtual void SpecialAbility() { std::cout << "多罗 ~\n"; };
```

```
11 };
```