ФИО: Ребдев Павел Александрович
Группа: 5130904/30008
Лабораторная работа: «Виртуальные методы»

Постановка задачи

Создать абстрактный класс Pet, классы Cat и Dog, наследующиеся от Pet и класс FamilyPets

Исходные тексты программы

Файлы с исходными текстами лабораторной работы (полагаем <ROOT> для папки в котором располагаются исходные тексты):

./<ROOT>/main.cpp

```
#include <iostream>
#include "pet.hpp"
#include "dog.hpp"
#include "cat.hpp"
#include "familyPets.hpp"

int main()
{
  rebdev::Dog dog;
  rebdev::Dog * pDog = &dog;
  rebdev::Cat cat;
  rebdev::Cat * pCat = &cat;
  rebdev::Pet * pPet = nullptr;
  std::cout << pDog->name() << " " << pDog->voice() << '\n';
  std::cout << pCat->name() << " " << pCat->voice() << '\n';

  pPet = &cat;
  std::cout << (*pPet).name() << " " << (*pPet).voice() << '\n';

  pPet = &dog;
  std::cout << (*pPet).name() << " " << (*pPet).voice() << '\n';

  rebdev::FamilyPets family;
  family + pCat;
  family + pDog;

  family.voice();
  family.name();
  return 0;
}
```

./<ROOT>/pet.hpp

```
#ifndef PET_HPP
#define PET_HPP
namespace rebdev
{
  class Pet
  {
    public:
      virtual char * voice() = 0;
      virtual char * name() = 0;
  };
}
```

```
#endif
```

./<ROOT>/cat.hpp

```
#ifndef CAT_HPP
#define CAT_HPP
#include "pet.hpp"
namespace rebdev
{
  class Cat: public Pet
  {
    public:
    Cat();
    Cat(char * voice, char * name);

    char * voice();
    char * name();

    private:
    char * name_;
    char * voice_;
  };
}
#endif
```

./<ROOT>/cat.cpp

```
#include "cat.hpp"

rebdev::Cat::Cat():
  voice_("Miau"),
  name_("Barsik")
{};
rebdev::Cat::Cat(char * voice, char * name):
  voice_(voice),
  name_(name)
{};


char * rebdev::Cat::voice()
{
  return voice_;
};
char * rebdev::Cat::name(){
  return name_;
};
```

./<ROOT>/dog.hpp

```
#ifndef DOG_HPP
#define DOG_HPP
#include "pet.hpp"
namespace rebdev
{
  class Dog: public Pet
  {
    public:
```

```
    Dog();
    Dog(char * voice, char * name);

    char * voice();
    char * name();

    private:
    char * name_;
    char * voice_;
  };
}
#endif
```

./<ROOT>/dog.cpp

```
#include "dog.hpp"

rebdev::Dog::Dog():
  voice_("Gav"),
  name_("Polkan")
{};
rebdev::Dog::Dog(char * voice, char * name):
  voice_(voice),
  name_(name)
{};


char * rebdev::Dog::voice()
{
  return voice_;
};
char * rebdev::Dog::name(){
  return name_;
};
```

./<ROOT>/familyPets.hpp

```
#ifndef FAMILYPETS_HPP
#define FAMILYPETS_HPP
#include "pet.hpp"
#include <cstddef>
namespace rebdev
{
  class FamilyPets
  {
    public:
      FamilyPets();
      void voice();
      void name();
      FamilyPets * operator+(Pet * newPet);
    private:
      size_t maxNumOfPets_;
      size_t numOfPets_;
      Pet * * pets_;
  };
}
```

```
#endif
```

./<ROOT>/familyPets.cpp

```
#include "familyPets.hpp"
#include <iostream>

rebdev::FamilyPets::FamilyPets():
  maxNumOfPets_(0),
  numOfPets_(0),
  pets_(nullptr)
{};

void rebdev::FamilyPets::voice()
{
  for (size_t i = 0; i < numOfPets_; ++i)
  {
    std::cout << (pets_[i]->voice()) << " ";
  }
  std::cout << '\n';
};
void rebdev::FamilyPets::name()
{
    for (size_t i = 0; i < numOfPets_; ++i)
  {
    std::cout << (pets_[i]->name()) << " ";
  }
  std::cout << '\n';
};
rebdev::FamilyPets * rebdev::FamilyPets::operator+(Pet * newPet)
{
  Pet * * newPets = new Pet *[numOfPets_ + 1];
  for (size_t i = 0; i < numOfPets_; ++i)
  {
    newPets[i] = pets_[i];
  }
  newPets[numOfPets_] = newPet;
  delete[] pets_;
  pets_ = newPets;
  newPets = nullptr;

  numOfPets_ += 1;
  return this;
};
```