1. #include <sstream>
2. #include <stdlib.h>
3. using namespace std;
5. /\*
6. \* Classe Zoo
7. \*/
8. void Zoo::adicionaAnimal(Animal \*al) {
9. animais.push\_back(al);
10. }
11. int Zoo::numAnimais() const {
12. return animais.size();
13. }
15. int Zoo::numVeterinarios() const {
16. return veterinarios.size();
17. }
19. string Zoo::getInformacao() const {
20. typename vector<Animal\*>::const\_iterator it= animais.begin();
21. string information;
22. while(it!=animais.end()){
23. information += (\*it)->getInformacao()+"\n";
24. it++;
25. }
26. return information;
27. }
29. bool Zoo::animalJovem(string nomeA) {
30. typename vector<Animal\*>::const\_iterator it= animais.begin();
31. while(it!=animais.end()){
32. if(nomeA==(\*it)->getNome() && (\*it)->eJovem()==true)
33. return true;
34. it++;
35. }
36. return false;
37. }

void Zoo::alocaVeterinarios(istream &isV) {

    string nome,cod;

    while(!isV.eof()){

        getline(isV,nome);

        getline(isV,cod);

        Veterinario \*v= new Veterinario(nome,stol(cod));

        veterinarios.push\_back(v);

    }

    int vet=0;

    for(unsigned int x=0;x<animais.size();x++){

        animais[x]->setVeterinario(veterinarios[vet]);

        vet++;

        vet=vet%veterinarios.size();

    }

}

bool Zoo::removeVeterinario(string nomeV){

    bool found\_vet=false;

    int pos;

    for (size\_t i = 0; i < veterinarios.size(); i++) {

        if (veterinarios[i]->getNome() == nomeV) {

            veterinarios.erase(veterinarios.begin() + i);

            pos = i;

            found\_vet=true;

            break;

        }

    }

    if(found\_vet==true){

        if(pos==veterinarios.size())

            pos=0;

        for (size\_t i = 0; i < animais.size(); i++) {

            if (animais[i]->getVet()->getNome() == nomeV)

                animais[i]->setVeterinario(veterinarios[pos]);

        }

    }

    else

        return false;

}

bool Zoo::operator<(Zoo &zoo2) const {

    int sum\_z1, sum\_z2;

    typename vector<Animal\*>::const\_iterator it= animais.begin();

    typename vector<Animal\*>::const\_iterator it\_2= zoo2.animais.begin();

    while(it!=animais.end()){

        sum\_z1+=(\*it)->getIdade();

        it++;

    }

    while(it\_2!=zoo2.animais.end()){

        sum\_z2+=(\*it\_2)->getIdade();

        it\_2++;

    }

    if(sum\_z1<sum\_z2)

        return true;

    else

        return false;

}

#include "veterinario.h"

#include <sstream>

using namespace std;

Veterinario::Veterinario(string nome, int cod) {

    this->nome=nome;

    this->codOrdem=cod;

}

string Veterinario::getNome() const {

    return nome;

}

long Veterinario::getCod() const {

    return codOrdem;

}

#include "animal.h"

#include <sstream>

#include <vector>

using namespace std;

/\*

 \* Classe Animal

 \*/

int Animal::maisJovem=999;

string Animal::getNome() const {

    return nome;

}

Animal::Animal(string nome, int idade) {

    this->nome=nome;

    this->idade = idade;

    this->vet=NULL;

    if(Animal::maisJovem>idade)

        Animal::maisJovem=idade;

}

int Animal::getMaisJovem() {

    return Animal::maisJovem;

}

int Animal::getIdade() const {

    return idade;

}

void Animal::setVeterinario(Veterinario \*v) {

    this->vet = v;

}

Veterinario \* Animal::getVet() {

    return this->vet;

}

Cao::Cao(string nome, int idade, string raca):Animal(nome,idade) {

    this->raca=raca;

}

bool Cao::eJovem() const {

    if(this->idade<5)

        return true;

    else

        return false;

}

string Cao::getInformacao(){

    string info;

    if(vet==NULL)

        info=this->nome+", "+to\_string(this->idade)+", "+this->raca;

    else

        info=this->nome+", "+to\_string(this->idade)+", "+this->vet->getNome()+", "+to\_string(this->vet->getCod())+", "+this->raca;

    return info;

}

Voador::Voador(int vmax, int amax) {

    this->velocidade\_max = vmax;

    this->altura\_max = amax;

}

int Voador::getVelocidade() const {

    return this->velocidade\_max;

}

int Voador::getAltura() const {

    return this->altura\_max;

}

Morcego::Morcego(string nome, int idade, int vmax, int amax):Animal(nome,idade),Voador(vmax,amax) {

}

bool Morcego::eJovem() const {

    if(this->idade<4)

        return true;

    else

        return false;

}

string Morcego::getInformacao() {

    string info;

    if(vet==NULL)

        info=this->nome+", "+to\_string(this->idade)+", "+to\_string(this->getVelocidade())+", "+to\_string(this->getAltura());

    else

        info=this->nome+", "+to\_string(this->idade)+", "+this->vet->getNome()+", "+to\_string(this->vet->getCod())+", "+to\_string(this->getVelocidade())+", "+to\_string(this->getAltura());

    return info;

}