**Main.cpp**#include "human.h"

#include "car.h"

#include "animal.h"

using namespace age;

#include <iostream>

int main() {

Human\* human1 = new Human("Kirill", "Grechanov", humanAge::Kirill, 70.5, 173.3);

Human\* human2 = new Human("Kir", "Grech", humanAge::Kir, 80.5, 181.3);

--\*human1;

std::cout << \*human1;

++\*human2;

std::cout << \*human2;

if (\*human1 > \*human2) std::cout << "Рост " << human1->getFirstName() << " больше" << std::endl;

else if (\*human1 < \*human2) std::cout << "Рост " << human2->getFirstName() << " больше" << std::endl;

else std::cout << "Равны по росту";

std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" << std::endl;

Car\* car1 = new Car("Lada", "Grey", "Sedan", 2.1, carProbeg::Lada);

Car\* car2 = new Car("Volga", "White", "Hatchback", 1.2, carProbeg::Volga);

--\*car1;

std::cout << \*car1;

++\*car2;

std::cout << \*car2;

if (\*car1 > \*car2) std::cout << "Объем двигателя " << car1->getBrand() << " больше" << std::endl;

else if (\*car1 < \*car2) std::cout << "Объем двигателя " << car2->getBrand() << " больше" << std::endl;

else std::cout << "Равны по объему двигателя";

std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" << std::endl;

Animal\* animal1 = new Animal("Boris", "Grey", "Cat", 5.1, animalAge::Boris);

Animal\* animal2 = new Animal("Ginger", "White", "Dog", 6.9, animalAge::Ginger);

--\*animal1;

std::cout << \*animal1;

++\*animal2;

std::cout << \*animal2;

if (\*animal1 > \*animal2) std::cout << "Вес " << animal1->getName() << " больше" << std::endl;

else if (\*animal1 < \*animal2) std::cout << "Вес " << animal2->getName() << " больше" << std::endl;

else std::cout << "Равны по весу";

return 0;

}

**Age.h**

#pragma once

#include <string>

namespace age

{

enum humanAge

{

Kirill = 17,

Kir = 18,

};

enum carProbeg

{

Lada = 50000,

Volga = 70000,

};

enum animalAge

{

Boris = 3,

Ginger = 5

};

std::string correctAlpha(std::string str);

int correctCount (int count);

double correctDecimal (double decimal);

}

**Age.cpp**

#include "age.h"

#include <iostream>

std::string age::correctAlpha(std::string str)

{

setlocale(LC\_ALL, "");

for (int i = 0; i < str.length(); ++i)

{

if (!isalpha(str[i]) && !ispunct(str[i]))

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return "underfined";

}

}

return str;

}

int age::correctCount(int count)

{

setlocale(LC\_ALL, "");

if (count >= 0) return count;

else

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return 0;

}

}

double age::correctDecimal (double decimal)

{

setlocale(LC\_ALL, "");

if (decimal >= 0) return decimal;

else

{

std::cout << "Некорректный ввод данных в поле\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

return 0;

}

}

**Human.h**

#pragma once

#include <iostream>

#include <string>

#include "age.h"

class Human {public:

Human();

Human(std::string firstName, std::string lastName, int age, double weight, double height);

Human(std::string firstName, std::string lastName, int age, double weight);

Human(std::string firstName, std::string lastName, int age);

Human(std::string firstName, std::string lastName);

Human(std::string firstName);

~Human();

void setFirstName(std::string firstName);

std::string getFirstName();

void setLastName(std::string lastName);

std::string getLastName();

void setAge(int age);

int getAge();

void setWeight(double weight);

double getWeight();

void setHeight(double height);

double getHeight();

Human operator++();

Human operator--();

bool operator< (Human other);

bool operator> (Human other);

friend std::ostream& operator<< (std::ostream& output, Human human);

private:

std::string \_firstName;

std::string \_lastName;

int \_age;

double \_weight;

double \_height;

};

**Human.cpp**

#include "human.h"

#include <iostream>

Human::Human() {

this->\_firstName = "underfined";

this->\_lastName = "underfined";

this->\_age = 0;

this->\_weight = 0.0;

this->\_height = 0.0;

}

Human::Human(std::string firstName, std::string lastName, int age, double weight, double height) : \_firstName(age::correctAlpha(firstName)),

\_lastName(age::correctAlpha(lastName)), \_age(age::correctCount(age)), \_weight(age::correctDecimal(weight)), \_height(age::correctDecimal(height)) {}

Human::Human(std::string firstName, std::string lastName, int age, double weight) : Human(firstName, lastName, age, weight, 0) {}

Human::Human(std::string firstName, std::string lastName, int age) : Human(firstName, lastName, age, 0, 0) {}

Human::Human(std::string firstName, std::string lastName) : Human(firstName, lastName, 0, 0.0, 0.0) {}

Human::Human(std::string firstName) : Human(firstName, "underfined", 0, 0.0, 0.0) {}

Human::~Human() {}

void Human::setFirstName(std::string firstName)

{

this->\_firstName = age::correctAlpha(firstName);

}

std::string Human::getFirstName()

{

return \_firstName;

}

void Human::setLastName(std::string lastName)

{

this->\_lastName = age::correctAlpha(lastName);

}

std::string Human::getLastName()

{

return \_lastName;

}

void Human::setAge(int age)

{

this->\_age = age::correctCount(age);

}

int Human::getAge()

{

return \_age;

}

void Human::setWeight(double weight)

{

this->\_weight = age::correctDecimal(weight);

}

double Human::getWeight()

{

return \_weight;

}

void Human::setHeight(double height)

{

this->\_height = age::correctDecimal(height);

}

double Human::getHeight()

{

return \_height;

}

Human Human::operator++(){

\_height += 10;

return \*this;

}

Human Human::operator--(){

\_height -= 10;

return \*this;

}

bool Human::operator < (Human other){

return this->\_height < other.\_height;

}

bool Human::operator > (Human other){

return this->\_height > other.\_height;

}

std::ostream& operator<<(std::ostream& output, Human human){

output << "-----------------\n"

<< "Имя: " << human.\_firstName << " " << human.\_lastName << "\n"

<< "Возраст: " << human.\_age << " лет\n"

<< "Вес: " << human.\_weight << " кг\n"

<< "Рост: " << human.\_height << " см\n"

<< "-----------------\n";

return output;

}

**Car.h**

#pragma once

#include <iostream>

#include <string>

#include "age.h"

class Car {public:

Car();

Car(std::string brand, std::string color, std::string bodyType, double engineVolume, int probeg);

Car(std::string brand, std::string color, std::string bodyType, double engineVolume);

Car(std::string brand, std::string color, std::string bodyType);

Car(std::string brand, std::string color);

Car(std::string brand);

~Car();

void setBrand(std::string brand);

std::string getBrand();

void setColor(std::string color);

std::string getColor();

void setBodyType(std::string bodyType);

std::string getBodyType();

void setEngineVolume(double engineVolume);

double getEngineVolume();

void setProbeg(int probeg);

int getProbeg();

Car operator++();

Car operator--();

bool operator< (Car other);

bool operator> (Car other);

friend std::ostream& operator<< (std::ostream& output, Car car);

private:

std::string \_brand;

std::string \_color;

std::string \_bodyType;

double \_engineVolume;

int \_probeg;

};

**Car.cpp**

#include "car.h"

#include <iostream>

Car::Car() {

this->\_brand = "underfined";

this->\_color = "underfined";

this->\_bodyType = "underfined";

this->\_engineVolume = 0.0;

this->\_probeg = 0;

}

Car::Car(std::string brand, std::string color, std::string bodyType, double engineVolume, int probeg) : \_brand(age::correctAlpha(brand)),

\_color(age::correctAlpha(color)), \_bodyType(age::correctAlpha(bodyType)), \_engineVolume(age::correctDecimal(engineVolume)), \_probeg(age::correctCount(probeg)) {}

Car::Car(std::string brand, std::string color, std::string bodyType, double engineVolume) : Car(brand, color, bodyType, engineVolume, 0) {}

Car::Car(std::string brand, std::string color, std::string bodyType) : Car(brand, color, bodyType, 0.0, 0) {}

Car::Car(std::string brand, std::string color) : Car(brand, color, "underfined", 0.0, 0) {}

Car::Car(std::string brand) : Car(brand, "underfined", "underfined", 0.0, 0) {}

Car::~Car() {}

void Car::setBrand(std::string brand)

{

this->\_brand = age::correctAlpha(brand);

}

std::string Car::getBrand()

{

return \_brand;

}

void Car::setColor(std::string color)

{

this->\_color = age::correctAlpha(color);

}

std::string Car::getColor()

{

return \_color;

}

void Car::setBodyType(std::string bodyType)

{

this->\_bodyType = age::correctAlpha(bodyType);

}

std::string Car::getBodyType()

{

return \_bodyType;

}

void Car::setEngineVolume(double engineVolume)

{

this->\_engineVolume = age::correctDecimal(engineVolume);

}

double Car::getEngineVolume()

{

return \_engineVolume;

}

void Car::setProbeg(int probeg)

{

this->\_probeg = age::correctCount(probeg);

}

int Car::getProbeg()

{

return \_probeg;

}

Car Car::operator++(){

\_engineVolume += 0.5;

return \*this;

}

Car Car::operator--(){

\_engineVolume -= 0.5;

return \*this;

}

bool Car::operator < (Car other){

return this->\_engineVolume < other.\_engineVolume;

}

bool Car::operator > (Car other){

return this->\_engineVolume > other.\_engineVolume;

}

std::ostream& operator<<(std::ostream& output, Car car){

output << "-----------------\n"

<< "Бренд: " << car.\_brand << "\n"

<< "Цвет: " << car.\_color << "\n"

<< "Тип кузова: " << car.\_bodyType << "\n"

<< "Объем двигателя: " << car.\_engineVolume << " л\n"

<< "Пробег: " << car.\_probeg << " км\n"

<< "-----------------\n";

return output;

}

**Animal.h**

#pragma once

#include <iostream>

#include <string>

#include "age.h"

class Animal {public:

Animal();

Animal(std::string name, std::string species, std::string color, double weight, int age);

Animal(std::string name, std::string species, std::string color, double weight);

Animal(std::string name, std::string species, std::string color);

Animal(std::string name, std::string species);

Animal(std::string name);

~Animal();

void setName(std::string name);

std::string getName();

void setSpecies(std::string species);

std::string getSpecies();

void setColor(std::string color);

std::string getColor();

void setWeight(double weight);

double getWeight();

void setAge(int age);

int getAge();

Animal operator++();

Animal operator--();

bool operator< (Animal other);

bool operator> (Animal other);

friend std::ostream& operator<< (std::ostream& output, Animal animal);

private:

std::string \_name;

std::string \_species;

std::string \_color;

double \_weight;

int \_age;

};

**Animal.cpp**

#include "animal.h"

#include <iostream>

Animal::Animal() {

this->\_name = "underfined";

this->\_species = "underfined";

this->\_color = "underfined";

this->\_weight = 0.0;

this->\_age = 0;

}

Animal::Animal(std::string name, std::string species, std::string color, double weight, int age) : \_name(age::correctAlpha(name)),

\_species(age::correctAlpha(species)), \_color(age::correctAlpha(color)), \_weight(age::correctDecimal(weight)), \_age(age::correctCount(age)) {}

Animal::Animal(std::string name, std::string species, std::string color, double weight) : Animal(name, species, color, weight, 0) {}

Animal::Animal(std::string name, std::string species, std::string color) : Animal(name, species, color, 0.0, 0) {}

Animal::Animal(std::string name, std::string species) : Animal(name, species, "underfined", 0.0, 0) {}

Animal::Animal(std::string name) : Animal(name, "underfined", "underfined", 0.0, 0) {}

Animal::~Animal(){}

void Animal::setName(std::string name)

{

this->\_name = age::correctAlpha(name);

}

std::string Animal::getName()

{

return \_name;

}

void Animal::setSpecies(std::string species)

{

this->\_species = age::correctAlpha(species);

}

std::string Animal::getSpecies()

{

return \_species;

}

void Animal::setColor(std::string color)

{

this->\_color = age::correctAlpha(color);

}

std::string Animal::getColor()

{

return \_color;

}

void Animal::setWeight(double weight)

{

this->\_weight = age::correctDecimal(weight);

}

double Animal::getWeight()

{

return \_weight;

}

void Animal::setAge(int age)

{

this->\_age = age::correctCount(age);

}

int Animal::getAge()

{

return \_age;

}

Animal Animal::operator++(){

\_weight += 0.5;

return \*this;

}

Animal Animal::operator--(){

\_weight -= 0.5;

return \*this;

}

bool Animal::operator < (Animal other){

return this->\_weight < other.\_weight;

}

bool Animal::operator > (Animal other){

return this->\_weight > other.\_weight;

}

std::ostream& operator<<(std::ostream& output, Animal animal){

output << "-----------------\n"

<< "Имя: " << animal.\_name << "\n"

<< "Вид: " << animal.\_species << "\n"

<< "Окрас: " << animal.\_color << "\n"

<< "Вес: " << animal.\_weight << " кг\n"

<< "Возраст: " << animal.\_age << " лет\n"

<< "-----------------\n";

return output;

}