**Main.cpp**

#include "human.h"

#include "adult.h"

#include "child.h"

#include "car.h"

#include "motorcar.h"

#include "truck.h"

#include "animal.h"

#include "pet.h"

#include "wild.h"

using namespace age;

#include <iostream>

int main() {

Adult\* human1 = new Adult("Kirill", "Grechanov", 18, 70.5, 173.3, "teacher");

Child human2("Kir", "Grech", 17, 80.5, 181.3, 4);

Human\* pHuman1 = human1;

Human\* pHuman2 = &human2;

pHuman1->display();

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

pHuman2->display();

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

Motorcar\* car1 = new Motorcar("Lada", "Grey", "Sedan", 2.1, 50000, 5);

Truck car2("Volga", "White", "Hatchback", 1.2, 60000, 6);

Motorcar\* pCar1 = car1;

Truck\* pCar2 = &car2;

pCar1->display();

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

pCar2->display();

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

Pet\* animal1 = new Pet("Boris", "Grey", "Cat", 5.1, 4, "fish");

Wild animal2("Ginger", "White", "Dog", 6.9, 6, "forest");

Animal\*\* arr = new Animal\*[2]{animal1, &animal2};

for (int i = 0; i < 2; ++i)

{

arr[i]->display();

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

}

delete human1;

delete car1;

delete animal1;

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);

virtual ~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();

virtual void display();

private:

std::string \_firstName;

std::string \_lastName;

int \_age;

double \_weight;

double \_height;

};

**Human.cpp**

#include "human.h"

#include <iostream>

Human::Human() {

this->\_firstName = "undefined";

this->\_lastName = "undefined";

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, "undefined", 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;

}

void Human::display() {

std::cout << "Имя: " << \_firstName << " " << \_lastName << "\n"

<< "Возраст: " << \_age << "\n"

<< "Вес: " << \_weight << " kg\n"

<< "Рост: " << \_height << " cm\n";

}

**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);

virtual ~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();

virtual void display();

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 = "undefined";

this->\_color = "undefined";

this->\_bodyType = "undefined";

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, "undefined", 0.0, 0) {}

Car::Car(std::string brand) : Car(brand, "underfined", "undefined", 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;

}

void Car::display() {

std::cout << "Бренд: " << \_brand << "\n"

<< "Цвет: " << \_color << "\n"

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

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

<< "Пробег: " << \_probeg << " км\n";

}

**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);

virtual ~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();

virtual void display();

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 = "undefined";

this->\_species = "undefined";

this->\_color = "undefined";

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, "undefined", 0.0, 0) {}

Animal::Animal(std::string name) : Animal(name, "undefined", "undefined", 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;

}

void Animal::display() {

std::cout << "Имя: " << \_name << "\n"

<< "Вид: " << \_species << "\n"

<< "Окрас: " << \_color << "\n"

<< "Вес: " << \_weight << " кг\n"

<< "Возраст: " << \_age << " лет\n";

}

**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 "undefined";

}

}

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;

}

}

**Adult.h**

#pragma once

#include "human.h"

class Adult : public Human

{public:

Adult();

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

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

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

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

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

Adult(std::string firstName);

~Adult();

void setProfession(std::string profession);

std::string getProfession();

void display();

private:

std::string \_profession;

};

**Adult.cpp**

#include "adult.h"

Adult::Adult()

{

setFirstName("undefined");

setLastName("undefined");

setAge(0);

setWeight(0);

setHeight(0);

this->\_profession = "undefined";

}

Adult::Adult(std::string firstName, std::string lastName, int age, double weight, double height, std::string profession) :

Human(firstName, lastName, age, weight, height), \_profession(age::correctAlpha(profession)) {}

Adult::Adult(std::string firstName, std::string lastName, int age, double weight, double height) : Human(firstName, lastName, age, weight, height), \_profession("undefined") {}

Adult::Adult(std::string firstName, std::string lastName, int age, double weight) : Human(firstName, lastName, age, weight, 0), \_profession("undefined") {}

Adult::Adult(std::string firstName, std::string lastName, int age) : Human(firstName, lastName, age, 0, 0), \_profession("undefined") {}

Adult::Adult(std::string firstName, std::string lastName) : Human(firstName, lastName, 0, 0, 0), \_profession("undefined") {}

Adult::Adult(std::string firstName) : Human(firstName, "underfined", 0, 0, 0), \_profession("undefined") {}

Adult::~Adult() {}

void Adult::setProfession(std::string profession)

{

this->\_profession = age::correctAlpha(profession);

}

std::string Adult::getProfession()

{

return \_profession;

}

void Adult::display()

{

setlocale(LC\_ALL, "");

Human::display();

std::cout << "Профессия: " << \_profession << std::endl;

}

**Child.h**

#pragma once

#include "human.h"

class Child : public Human

{public:

Child();

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

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

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

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

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

Child(std::string firstName);

~Child();

void setGrade(int grade);

int getGrade();

void display();

private:

int \_grade;

};

**Child.cpp**

#include "child.h"

Child::Child()

{

setFirstName("undefined");

setLastName("undefined");

setAge(0);

setWeight(0);

setHeight(0);

this->\_grade = 0;

}

Child::Child(std::string firstName, std::string lastName, int age, double weight, double height, int grade) :

Human(firstName, lastName, age, weight, height), \_grade(age::correctCount(grade)) {}

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

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

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

Child::Child(std::string firstName, std::string lastName) : Human(firstName, lastName, 0, 0, 0), \_grade(0) {}

Child::Child(std::string firstName) : Human(firstName, "undefined", 0, 0, 0), \_grade(0) {}

Child::~Child() {}

void Child::setGrade(int grade)

{

this->\_grade = age::correctCount(grade);

}

int Child::getGrade()

{

return \_grade;

}

void Child::display()

{

setlocale(LC\_ALL, "");

Human::display();

std::cout << "Класс: " << \_grade << std::endl;

}

**Motorcar.h**

#pragma once

#include "car.h"

class Motorcar : public Car

{public:

Motorcar();

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

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

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

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

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

Motorcar(std::string brand);

~Motorcar();

void setSeats(int seats);

int getSeats();

void display();

private:

int \_seats;

};

**Motorcar.cpp**

#include "motorcar.h"

Motorcar::Motorcar()

{

setBrand("undefined");

setColor("undefined");

setBodyType("undefined");

setEngineVolume(0);

setProbeg(0);

this->\_seats = 0;

}

Motorcar::Motorcar(std::string brand, std::string color, std::string bodeType, double engineVolume, int probeg, int seats) :

Car(brand, color, bodeType, engineVolume, probeg), \_seats(age::correctCount(seats)) {}

Motorcar::Motorcar(std::string brand, std::string color, std::string bodeType, double engineVolume, int probeg) : Car(brand, color, bodeType, engineVolume, probeg), \_seats(0) {}

Motorcar::Motorcar(std::string brand, std::string color, std::string bodeType, double engineVolume) : Car(brand, color, bodeType, engineVolume, 0), \_seats(0) {}

Motorcar::Motorcar(std::string brand, std::string color, std::string bodeType) : Car(brand, color, bodeType, 0, 0), \_seats(0) {}

Motorcar::Motorcar(std::string brand, std::string color) : Car(brand, color, "undefined", 0, 0), \_seats(0) {}

Motorcar::Motorcar(std::string brand) : Car(brand, "undefined", "undefined", 0, 0), \_seats(0) {}

Motorcar::~Motorcar() {}

void Motorcar::setSeats(int seats)

{

this->\_seats = age::correctCount(seats);

}

int Motorcar::getSeats()

{

return \_seats;

}

void Motorcar::display()

{

setlocale(LC\_ALL, "");

Car::display();

std::cout << "Кол-во сидений: " << \_seats << std::endl;

}

**Truck.h**

#pragma once

#include "car.h"

class Truck : public Car

{public:

Truck();

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

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

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

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

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

Truck(std::string brand);

~Truck();

void setWheels(int wheels);

int getWheels();

void display();

private:

int \_wheels;

};

**Truck.cpp**

#include "truck.h"

Truck::Truck()

{

setBrand("undefined");

setColor("undefined");

setBodyType("undefined");

setEngineVolume(0);

setProbeg(0);

this->\_wheels = 0;

}

Truck::Truck(std::string brand, std::string color, std::string bodeType, double engineVolume, int probeg, int wheels) :

Car(brand, color, bodeType, engineVolume, probeg), \_wheels(age::correctCount(wheels)) {}

Truck::Truck(std::string brand, std::string color, std::string bodeType, double engineVolume, int probeg) : Car(brand, color, bodeType, engineVolume, probeg), \_wheels(0) {}

Truck::Truck(std::string brand, std::string color, std::string bodeType, double engineVolume) : Car(brand, color, bodeType, engineVolume, 0), \_wheels(0) {}

Truck::Truck(std::string brand, std::string color, std::string bodeType) : Car(brand, color, bodeType, 0, 0), \_wheels(0) {}

Truck::Truck(std::string brand, std::string color) : Car(brand, color, "undefined", 0, 0), \_wheels(0) {}

Truck::Truck(std::string brand) : Car(brand, "undefined", "undefined", 0, 0), \_wheels(0) {}

Truck::~Truck() {}

void Truck::setWheels(int wheels)

{

this->\_wheels = age::correctCount(wheels);

}

int Truck::getWheels()

{

return \_wheels;

}

void Truck::display()

{

setlocale(LC\_ALL, "");

Car::display();

std::cout << "Кол-во колес: " << \_wheels << std::endl;

}

**Pet.h**

#pragma once

#include "animal.h"

class Pet : public Animal

{public:

Pet();

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

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

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

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

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

Pet(std::string name);

~Pet();

void setFood(std::string food);

std::string getFood();

void display();

private:

std::string \_food;

};

**Pet.cpp**

#include "pet.h"

Pet::Pet()

{

setName("undefined");

setSpecies("undefined");

setColor("undefined");

setWeight(0);

setAge(0);

this->\_food = "undefined";

}

Pet::Pet(std::string name, std::string species, std::string color, double weight, int age, std::string food) :

Animal(name, species, color, weight, age), \_food(age::correctAlpha(food)) {}

Pet::Pet(std::string name, std::string species, std::string color, double weight, int age) : Animal(name, species, color, weight, age), \_food("undefined") {}

Pet::Pet(std::string name, std::string species, std::string color, double weight) : Animal(name, species, color, weight, 0), \_food("undefined") {}

Pet::Pet(std::string name, std::string species, std::string color) : Animal(name, species, color, 0, 0), \_food("undefined") {}

Pet::Pet(std::string name, std::string species) : Animal(name, species, "underfined", 0, 0), \_food("undefined") {}

Pet::Pet(std::string name) : Animal(name, "undefined", "undefined", 0, 0), \_food("undefined") {}

Pet::~Pet() {}

void Pet::setFood(std::string food)

{

this->\_food = age::correctAlpha(food);

}

std::string Pet::getFood()

{

return \_food;

}

void Pet::display()

{

setlocale(LC\_ALL, "");

Animal::display();

std::cout << "Любимая еда: " << \_food << std::endl;

}

**Wild.h**

#pragma once

#include "animal.h"

class Wild : public Animal

{public:

Wild();

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

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

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

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

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

Wild(std::string name);

~Wild();

void setHabitat(std::string habitat);

std::string getHabitat();

void display();

private:

std::string \_habitat;

};

**Wild.cpp**

#include "wild.h"

Wild::Wild()

{

setName("undefined");

setSpecies("undefined");

setColor("undefined");

setWeight(0);

setAge(0);

this->\_habitat = "undefined";

}

Wild::Wild(std::string name, std::string species, std::string color, double weight, int age, std::string habitat) :

Animal(name, species, color, weight, age), \_habitat(age::correctAlpha(habitat)) {}

Wild::Wild(std::string name, std::string species, std::string color, double weight, int age) : Animal(name, species, color, weight, age), \_habitat("undefined") {}

Wild::Wild(std::string name, std::string species, std::string color, double weight) : Animal(name, species, color, weight, 0), \_habitat("undefined") {}

Wild::Wild(std::string name, std::string species, std::string color) : Animal(name, species, color, 0, 0), \_habitat("undefined") {}

Wild::Wild(std::string name, std::string species) : Animal(name, species, "underfined", 0, 0), \_habitat("undefined") {}

Wild::Wild(std::string name) : Animal(name, "undefined", "undefined", 0, 0), \_habitat("undefined") {}

Wild::~Wild() {}

void Wild::setHabitat (std::string habitat)

{

this->\_habitat = age::correctAlpha(habitat);

}

std::string Wild::getHabitat()

{

return \_habitat;

}

void Wild::display()

{

setlocale(LC\_ALL, "");

Animal::display();

std::cout << "Место обитания: " << \_habitat << std::endl;

}