**Лабораторная работа № 6**

**Часть 1**

**Задание1:**

**№1**

*#include <iostream>*

*class Base1 {*

*public:*

*Base1() {*

*std::cout << "Конструктор Base1 без параметров" << std::endl;*

*}*

*Base1(int value) {*

*std::cout << "Конструктор Base1 с параметром int: " << value << std::endl;*

*}*

*};*

*class Base2 {*

*public:*

*Base2() {*

*std::cout << "Конструктор Base2 без параметров" << std::endl;*

*}*

*Base2(double value) {*

*std::cout << "Конструктор Base2 с параметром double: " << value << std::endl;*

*}*

*};*

*class Derived : public Base1, public Base2 {*

*public:*

*Derived() {*

*std::cout << "Конструктор Derived без параметров" << std::endl;*

*}*

*Derived(char value) {*

*std::cout << "Конструктор Derived с параметром char: " << value << std::endl;*

*}*

*};*

*int main() {*

*Derived derived;*

*Derived derived2('A');*

*return 0;*

*}*

В данном примере класс Derived наследуется публично от классов Base1 и Base2. Он переопределяет конструкторы Derived() и Derived(int x, int y). При создании объектов класса Derived будет вызываться соответствующий конструктор, и будут выведены необходимые сообщения.

**№2**

*#include <iostream>*

*class Base1 {*

*private:*

*int i;*

*public:*

*Base1() : i(0) {*

*std::cout << "Конструктор Base1 без параметров" << std::endl;*

*}*

*Base1(int x) : i(x) {*

*std::cout << "Конструктор Base1 с параметром" << std::endl;*

*}*

*void put(int x) {*

*i = x;*

*}*

*int get() {*

*return i;*

*}*

*};*

*class Base2 {*

*public:*

*Base2() {*

*std::cout << "Конструктор Base2 без параметров" << std::endl;*

*}*

*Base2(int y) {*

*std::cout << "Конструктор Base2 с параметром" << std::endl;*

*}*

*};*

*class Derived : public Base1, public Base2 {*

*public:*

*Derived() : Base1() {*

*std::cout << "Конструктор Derived без параметров" << std::endl;*

*}*

*Derived(int x, int y) : Base1(x), Base2(y) {*

*std::cout << "Конструктор Derived с параметрами" << std::endl;*

*}*

*};*

*int main() {*

*Derived d1(10, 20);*

*Derived d2;*

*d1.put(15);*

*std::cout << "Значение поля i в d1: " << d1.get() << std::endl;*

*return 0;*

*}*

В этом примере для класса Base1 существуют две интерфейсные функции void put(int x) и int get(). Первая функция позволяет изменить значение закрытого поля i, вторая функция позволяет прочитать значение закрытого поля i.

В main мы вызываем эти две функции для объекта d1 класса Derived, которые позволяют нам изменить и прочитать значение этого поля.

**№8**

***#include <iostream>***

***#include <cstring>***

***class Base1 {***

***private:***

***int i;***

***public:***

***Base1() {***

***i = 0;***

***std::cout << "Конструктор Base1 без параметров" << std::endl;***

***}***

***Base1(int value) {***

***i = value;***

***std::cout << "Конструктор Base1 с параметром" << std::endl;***

***}***

***void put(int value) {***

***i = value;***

***}***

***int get() const { // Mark the get() function as const***

***return i;***

***}***

***};***

***class Base2 {***

***private:***

***char name[20];***

***public:***

***Base2() {***

***strcpy(name, "Пусто");***

***std::cout << "Конструктор Base2 без параметров" << std::endl;***

***}***

***Base2(const char\* value) {***

***strcpy(name, value);***

***std::cout << "Конструктор Base2 с параметром" << std::endl;***

***}***

***void put(const char\* value) {***

***strcpy(name, value);***

***}***

***const char\* get() const {***

***return name;***

***}***

***};***

***class Derived : public Base1, public Base2 {***

***private:***

***char ch;***

***public:***

***Derived() {***

***ch = 'V';***

***std::cout << "Конструктор Derived без параметров" << std::endl;***

***}***

***Derived(char c, const char\* n, int value) : Base1(value), Base2(n) {***

***ch = c;***

***std::cout << "Конструктор Derived с параметрами" << std::endl;***

***}***

***void put(char c) {***

***ch = c;***

***}***

***char get() const { // Mark the get() function as const***

***return ch;***

***}***

***friend std::ostream& operator<<(std::ostream& os, const Derived& obj) {***

***os << "i: " << obj.Base1::get() << ", name: " << obj.Base2::get() << ", ch: " << obj.get();***

***return os;***

***}***

***};***

***int main() {***

***std::cout << "Переменная без инициализации:" << std::endl;***

***Derived obj1;***

***std::cout << obj1 << std::endl;***

***std::cout << "\nДругая переменная с явной инициализацией:" << std::endl;***

***Derived obj2('X', "Значение", 42);***

***std::cout << obj2 << std::endl;***

***std::cout << "\nИзменение порядка вызова конструкторов базовых классов в Derived:" << std::endl;***

***Derived obj3('Y', "Изменено", 100);***

***std::cout << obj3 << std::endl;***

***return 0;***

***}***

Порядок вызова конструкторов в Derived определен базовыми классами и зависит от порядка их наследования. Если базовой класс Base1 идет перед Base2 в определении класса Derived, то конструктор Base1 вызывается перед Base2, и наоборот. В случае obj3, конструктор Derived с параметрами вызывается с передачей параметров для инициализации Base1 и Base2, и затем инициализирует поле ch значением 'Y'. Результат выводится на экран.

**Задание 2:**

***#include <iostream>***

***using namespace std;***

***//----------------------------------------------------------------------***

***class DomesticAnimal {***

***protected:***

***int weight;***

***int price;***

***string color;***

***public:***

***DomesticAnimal() {}***

***DomesticAnimal(int w, int p, string c) : weight(w), price(p), color(c) {}***

***void print() {***

***cout << "This is a Domestic Animal" << endl;***

***cout << "Weight: " << weight << endl;***

***cout << "Price: " << price << endl;***

***cout << "Color: " << color << endl;***

***}***

***};***

***//----------------------------------------------------------------------***

***class Cow : public DomesticAnimal {***

***public:***

***Cow(int w, int p, string c) : DomesticAnimal(w, p, c) {}***

***void print() {***

***cout << "---- Start of Cow ----" << endl;***

***DomesticAnimal::print();***

***cout << "This is a Cow" << endl;***

***cout << "---- End of Cow ----" << endl;***

***}***

***};***

***//----------------------------------------------------------------------***

***class Buffalo : public DomesticAnimal {***

***public:***

***Buffalo(int w, int p, string c) : DomesticAnimal(w, p, c) {}***

***void print() {***

***cout << "---- Start of Buffalo ----" << endl;***

***DomesticAnimal::print();***

***cout << "This is a Buffalo" << endl;***

***cout << "---- End of Buffalo ----" << endl;***

***}***

***};***

***//----------------------------------------------------------------------***

***class Beefalo : public Cow, public Buffalo {***

***public:***

***Beefalo(int w, int p, string c) : Cow(w, p, c), Buffalo(w, p, c) {}***

***void print() {***

***cout << "---- Start of Beefalo ----" << endl;***

***Cow::print(); // Вызов метода print() из класса Cow***

***Buffalo::print(); // Вызов метода print() из класса Buffalo***

***cout << "This is a Beefalo" << endl;***

***cout << "---- End of Beefalo ----" << endl;***

***}***

***};***

***//----------------------------------------------------------------------***

***//----------------------------------------------------------------------***

***int main() {***

***Cow cow(500, 1000, "Brown");***

***Beefalo beefalo(400, 1200, "Black");***

***cow.print();***

***beefalo.print();***

***return 0;***

***}***

**Вариант 2:**

***#include <iostream>***

***using namespace std;***

***class DomesticAnimal {***

***protected:***

***double weight;***

***double price;***

***string color;***

***public:***

***DomesticAnimal() {***

***weight = 0.0;***

***price = 0.0;***

***color = "None";***

***cout << "Конструктор DomesticAnimal без параметров" << endl;***

***}***

***DomesticAnimal(double weight\_value, double price\_value, string color\_value) {***

***weight = weight\_value;***

***price = price\_value;***

***color = color\_value;***

***cout << "Конструктор DomesticAnimal с параметрами" << endl;***

***}***

***~DomesticAnimal() { cout << "Деструктор DomesticAnimal " << endl; }***

***void print() {***

***cout***

***<< "Class DomesticAnimal { weight: " << weight***

***<< ", price: " << price***

***<< ", color: "<< color << " }" <<***

***endl;***

***}***

***};***

***class Cow : public DomesticAnimal {***

***public:***

***Cow() {***

***cout << "Конструктор Cow без параметров" << endl;***

***}***

***Cow(double weight\_value, double price\_value, string color\_value) : DomesticAnimal(weight\_value, price\_value, color\_value) {***

***cout << "Конструктор Cow с параметрами" << endl;***

***}***

***~Cow() { cout << "Деструктор Cow " << endl; }***

***void print() {***

***cout << "Class Cow" << endl;***

***DomesticAnimal::print();***

***}***

***};***

***class Buffalo : public DomesticAnimal {***

***public:***

***Buffalo() {***

***cout << "Конструктор Buffalo без параметров" << endl;***

***}***

***Buffalo(double weight\_value, double price\_value, string color\_value) : DomesticAnimal(weight\_value, price\_value, color\_value) {***

***cout << "Конструктор Buffalo с параметрами" << endl;***

***}***

***~Buffalo() { cout << "Деструктор Buffalo " << endl; }***

***void print() {***

***cout << "Class Buffalo" << endl;***

***DomesticAnimal::print();***

***}***

***};***

***class Beefalo : public Cow, public Buffalo {***

***public:***

***Beefalo() : Cow(), Buffalo(){***

***cout << "Конструктор Beefalo по умолчанию" << endl;***

***}***

***Beefalo(double weight\_value, double price\_value, string color\_value) : Cow(weight\_value, price\_value, color\_value), Buffalo(weight\_value, price\_value, color\_value){***

***cout << "Конструктор Beefalo по умолчанию" << endl;***

***}***

***~Beefalo() { cout << "Деструктор Beefalo " << endl; }***

***void print() {***

***cout << "Class Beefalo" << endl;***

***Cow::print();***

***Buffalo::print();***

***}***

***};***

***int main() {***

***Cow cow(500, 1000, "Brown");***

***Beefalo beefalo(400, 1200, "Black");***

***cow.print();***

***beefalo.print();***

***return 0;***

***}***

**Часть 2**

**Задание 1:**

***#include <iostream>***

***#include <cmath>***

***using namespace std;***

***//----------------------------------------------------------------------***

***class Figure {***

***public:***

***virtual double area() const = 0;***

***virtual void show() const = 0;***

***virtual double\* getRadius() { return nullptr; }***

***virtual double\* getLength() { return nullptr; }***

***virtual double\* getWidth() { return nullptr; }***

***};***

***//----------------------------------------------------------------------***

***class Circle : public Figure {***

***private:***

***double radius;***

***public:***

***Circle(double r) {***

***if (r <= 0) {***

***cout << "Ошибка: Радиус должен быть больше нуля." << endl;***

***exit(1);***

***}***

***radius = r;***

***}***

***double area() const override {***

***return M\_PI \* radius \* radius;***

***}***

***void show() const override {***

***cout << "Круг" << endl;***

***cout << "Радиус: " << radius << endl;***

***cout << "Площадь: " << area() << endl;***

***}***

***double\* getRadius() override {***

***return &radius;***

***}***

***};***

***//----------------------------------------------------------------------***

***class Rectangle : public Figure {***

***private:***

***double length;***

***double width;***

***public:***

***Rectangle(double l, double w) : length(l), width(w) {***

***if (l <= 0 || w <= 0) {***

***cout << "Ошибка: Длина и ширина должны быть больше нуля." << endl;***

***exit(1);***

***}***

***}***

***double area() const override {***

***return length \* width;***

***}***

***void show() const override {***

***if (length == width) {***

***cout << "Квадрат" << endl;***

***} else {***

***cout << "Прямоугольник" << endl;***

***}***

***cout << "Длина: " << length << endl;***

***cout << "Ширина: " << width << endl;***

***cout << "Площадь: " << area() << endl;***

***}***

***double\* getLength() override {***

***return &length;***

***}***

***double\* getWidth() override {***

***return &width;***

***}***

***};***

***//----------------------------------------------------------------------***

***//----------------------------------------------------------------------***

***int main() {***

***double radius, length, width;***

***cout << "Введите радиус круга: ";***

***cin >> radius;***

***Circle circle(radius);***

***cout << "Введите длину и ширину прямоугольника: ";***

***cin >> length >> width;***

***Rectangle rectangle(length, width);***

***// Вывод информации о фигурах***

***circle.show();***

***cout << "Адрес радиуса: " << circle.getRadius() << endl;***

***cout << endl;***

***rectangle.show();***

***cout << "Адрес длины: " << rectangle.getLength() << endl;***

***cout << "Адрес ширины: " << rectangle.getWidth() << endl;***

***cout << endl;***

***// Создание массива указателей на базовый класс Figure***

***Figure\* figures[2];***

***figures[0] = &circle;***

***figures[1] = &rectangle;***

***// Вывод информации о фигурах через массив указателей***

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

***figures[i]->show();***

***double\* radiusPtr = figures[i]->getRadius();***

***if (radiusPtr) {***

***cout << "Адрес радиуса: " << radiusPtr << endl;***

***}***

***double\* lengthPtr = figures[i]->getLength();***

***if (lengthPtr) {***

***cout << "Адрес длины: " << lengthPtr << endl;***

***}***

***double\* widthPtr = figures[i]->getWidth();***

***if (widthPtr) {***

***cout << "Адрес ширины: " << widthPtr << endl;***

***}***

***cout << endl;***

***}***

***return 0;***

***}***

**Задание 2:**