**Лабораторная работа № 6**

**Часть 1**

**Задание1:**

**№1**

*#include <iostream>*

*class Base1 {*

*public:*

*Base1() {*

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

*}*

*Base1(int x) {*

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

*}*

*};*

*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() {*

*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;*

*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.

**№3**

**№4**

**№5**

**№6**

**№7**

**№8**

**Задание 2:**

**№1**

**№2**

**№3**

**№4**

**№5**

**№6**

**Часть 2**

**Задание 1:**

**Задание 2:**