**Лабораторная работа 6**

**Множественное наследование**

**Цель работы:** изучить принципы и механизмы множественного наследования, правила доступа к базовым классам; приобрести практические навыки работы с базовыми и производными классами при множественном наследовании.

**Задание:**

1. Используя предыдущую программу, создайте новый производный класс с применением множественного наследования.

2. Проверьте работоспособность АТД и производных классов на тестовом наборе данных.

**Код программы:**

#include <iostream>

#include <string>

#include <clocale>

using namespace std;

////////////////////////////////////////////////////////////////////////

//////////////////////////ПЕРВЫЙ БАЗОВЫЙ КЛАСС//////////////////////////

////////////////////////////////////////////////////////////////////////

class Meow1

{

protected:

int number;

int volume;

char\* color;

bool tempreg;

public:

Meow1();

Meow1(int a, int b, char\* c, bool d);

Meow1(const Meow1& a);

~Meow1();

Meow1 operator =(const Meow1 &a);

void Print() const;

Meow1 sum(const Meow1& a);

bool Cmp(const Meow1& a);

void Copy(const Meow1& a);

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow1::Meow1()

{

number = 0;

volume = 0;

tempreg = false;

color = new char[1];

strcpy(color, "white");

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow1::Meow1(int a, int b, char\* c, bool d)

{

number = a;

volume = b;

tempreg = d;

color = new char[strlen(c) + 1];

strcpy(color, c);

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow1::Meow1(const Meow1& a)

{

if (&a != this)

{

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

}

}

/////////////////////////////ДЕСТРУКТОР/////////////////////////////

Meow1 :: ~Meow1()

{

delete[] color;

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow1 Meow1 :: operator=(const Meow1& a)

{

if (this == &a) return \*this;

else

{

this->Copy(a);

return \*this;

}

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow1 :: Print() const {

cout << "Серийный номер: " << number << endl;

cout << "Объём: " << volume << endl;

cout << "Регулятор температуры ";

if (tempreg)

{

cout << "имеется";

}

else

{

cout << "отсутствует";

}

cout << endl;

cout << "Цвет: " << color << endl;

};

/////////////////////////////ФУНКЦИЯ СЛОЖЕНИЯ/////////////////////////////

Meow1 Meow1 :: sum(const Meow1& a) {

Meow1 f;

char\* tmp = new char[strlen(color) + 1];

strcpy(tmp, color);

f.number = number + a.number;

f.volume = volume + a.volume;

f.tempreg = false;

strcat(tmp, "-");

strcat(tmp, a.color);

f.color = new char[strlen(tmp) + 1];

strcpy(f.color, tmp);

return f;

};

/////////////////////////////ФУНКЦИЯ СРАВНЕНИЯ/////////////////////////////

bool Meow1::Cmp(const Meow1& a) {

return volume == a.volume && number == a.number && tempreg == a.tempreg && color == a.color;

};

/////////////////////////////ФУНКЦИЯ КОПИРОВАНИЯ/////////////////////////////

void Meow1 :: Copy(const Meow1& a) {

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

}

////////////////////////////////////////////////////////////////////////

//////////////////////////ВТОРОЙ БАЗОВЫЙ КЛАСС//////////////////////////

////////////////////////////////////////////////////////////////////////

class Meow3

{

protected:

int temper;

double length;

public:

Meow3();

Meow3(int a, double b);

Meow3(const Meow3& a);

Meow3 operator =(const Meow3 &a);

void Print() const;

void Switch();

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow3::Meow3()

{

temper = 0;

length = 15;

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow3::Meow3(int a, double b)

{

temper = a;

length = b;

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow3::Meow3(const Meow3& a)

{

if (&a != this)

{

temper = a.temper;

length = a.length;

}

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow3 Meow3 :: operator=(const Meow3& a)

{

temper = a.temper;

length = a.length;

return \*this;

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow3::Print() const {

cout << "Температура: " << temper << endl;

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

};

/////////////////////////////ФУНКЦИЯ ВЫКЛЮЧЕНИЯ/////////////////////////////

void Meow3 :: Switch(){

temper = 0;

}

///////////////////////////////////////////////////////////////////////

///////////////////////////ПРОИЗВОДНЫЙ КЛАСС///////////////////////////

///////////////////////////////////////////////////////////////////////

class Meow2 : public Meow1, public Meow3

{

protected:

int size;

public:

Meow2();

Meow2(int a, int b, char\* c, bool d, int s, int t, double l);

Meow2(const Meow2& a);

~Meow2();

Meow2 operator =(const Meow2 &a);

void Print() const;

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow2::Meow2() :Meow1(), Meow3()

{

size = 0;

temper = 0;

length = 0;

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow2::Meow2(int a, int b, char\* c, bool d, int s, int t, double l) :Meow1(a, b, c, d), Meow3(t, l)

{

size = s;

temper = t;

length = l;

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow2::Meow2(const Meow2& a)

{

if (&a != this) {

Meow1::Meow1(a);

Meow3::Meow3(a);

size = a.size;

temper = a.temper;

length = a.length;

}

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow2 Meow2 :: operator=(const Meow2& a)

{

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

size = a.size;

temper = a.temper;

length = a.length;

return \*this;

}

/////////////////////////////ДЕКСТРУКТОР/////////////////////////////

Meow2 :: ~Meow2()

{

Meow1::~Meow1();

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow2::Print() const

{

Meow1::Print();

Meow3::Print();

cout << "Размер нагревательного прибора: " << size << endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

Meow1 mur1, mur2(1, 1, (char\*)"yellow", true), mur3(2, 2, (char\*)"red", false);

Meow3 mur7, mur8(40, 40);

Meow2 mur4, mur5(3, 3, (char\*)"black", false, 15, 40, 50), mur6(4, 4, (char\*)"white", true, 20, 30, 20);

Meow2 mur9;

cout << "Информация о нагревательных элементах: " << endl;

cout << "/////////////////////////////////mur1/////////////////////////////////" << endl;

mur1.Print();

cout << "/////////////////////////////////mur2/////////////////////////////////" << endl;

mur2.Print();

cout << "/////////////////////////////////mur3/////////////////////////////////" << endl;

mur3.Print();

cout << "/////////////////////////////////mur4/////////////////////////////////" << endl;

mur4.Copy(mur3);

mur4.Print();

cout << "/////////////////////////////////mur5/////////////////////////////////" << endl;

mur5.Print();

cout << "/////////////////////////////////mur6/////////////////////////////////" << endl;

Meow1 mur6\_1 = mur6.sum(mur5);

mur6.Copy(mur6\_1);

mur6.Print();

cout << "/////////////////////////////////mur7/////////////////////////////////" << endl;

mur7.Print();

cout << "//////////////////////////mur8 до выключения//////////////////////////" << endl;

mur8.Print();

mur8.Switch();

cout << "////////////////////////mur8 после выключения/////////////////////////" << endl;

mur8.Print();

cout << "/////////////////////////////////mur9/////////////////////////////////" << endl;

mur9.Print();

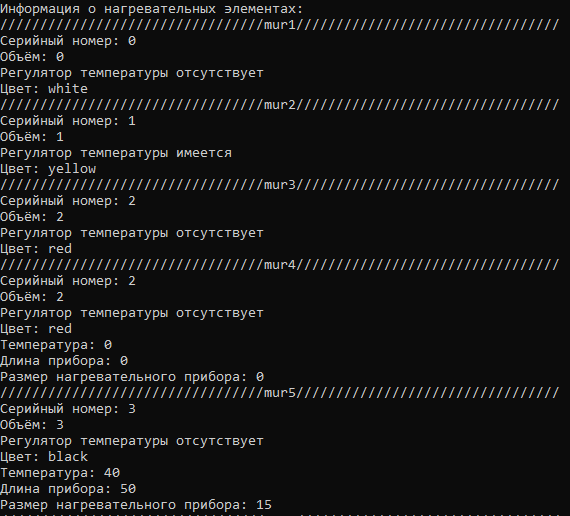
cout << endl;

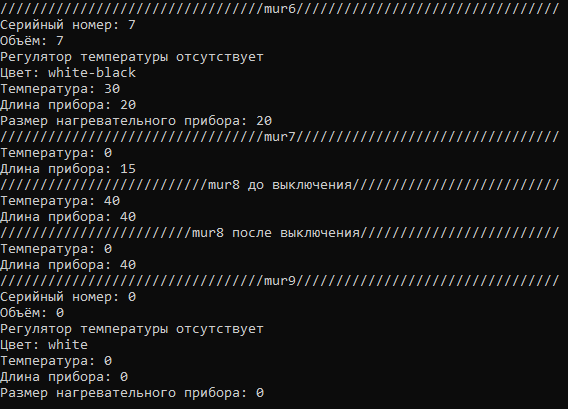
system("pause");

return 0;

}

**Результаты:**





**Контрольные вопросы:**

**1. С какой целью и в каких случаях используется множественное наследование?**

Множественное наследование (multiple inheritance) делает возможным получение производного класса от нескольких базовых классов. Синтаксис заголовка класса расширяется, чтобы можно было использовать список базовых классов с атрибутами доступа.

**2. Опишите синтаксис заголовка производного класса при множественном наследовании.**

class Meow2 : public Meow1, public Meow3 {}

**3. Дайте определение ориентированного ациклического графа.**

Ориентированный ациклический граф – это граф, узлы которого являются классами, а ориентированные ребра направлены от производных классов к базовым.

**4. Где инициируются виртуальные базовые классы?**

Виртуальные базовые классы инициализируются (вызывается void-конструктор) перед любыми невиртуальными базовыми классами и в том порядке, в котором они появляются в прямом ациклическом графе наследования при просмотре его снизу-вверх и слева направо.

Если виртуальный базовый класс имеет хотя бы один конструктор, то он должен иметь void-конструктор.

**5. Что будет, если из объявлений классов Cow и Buffalo убрать ключевое слово virtual (см. последний пример в теоретических положениях)?**

Ключевое слово virtual в классе Cow и классе Buffalo предотвращает многократное копирование полей данных weight, price, color из предков класса Beefalo.