**Лабораторная работа 5**

**Одиночное наследование**

**Цель работы:** изучить способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса; приобрести практические навыки наследования.

**Задание:** создайте производный класс для АТД, реализованного по заданию лабораторной работы 4, используя одиночное наследование. Проверьте работоспособность АТД на тестовом наборе данных.

**Код программы:**

#include <iostream>

#include <string>

#include <clocale>

using namespace std;

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

/////////////////////////////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;

}

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

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

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

class Meow2 : public Meow1

{

protected:

int size;

public:

Meow2();

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

Meow2(const Meow2& a);

~Meow2();

Meow2 operator =(const Meow2 &a);

void Print() const;

};

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

Meow2::Meow2() :Meow1()

{

size = 0;

}

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

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

{

size = s;

}

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

Meow2::Meow2(const Meow2& a)

{

if (&a != this) {

Meow1::Meow1(a);

size = a.size;

}

}

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

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

{

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

else

{

this->Copy(a);

return \*this;

}

}

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

Meow2 :: ~Meow2()

{

Meow1::~Meow1();

}

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

void Meow2::Print() const

{

Meow1::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);

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

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 mur7 = mur6.sum(mur5);

mur6.Copy(mur7);

mur6.Print();

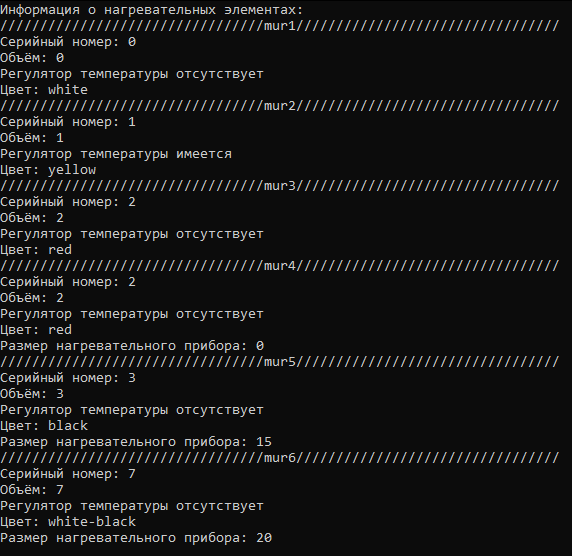
cout << endl;

system("pause");

return 0;

}

**Результаты:**



**Контрольные вопросы:**

**1. Дайте определение наследования.**

Наследование (inheritance) – это механизм получения нового класса на основе существующего класса. Существующий класс может быть дополнен или изменен для создания производного класса.

**2. Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?**

Модификаторы прав доступа к членам класса (public, protected и private) могут применяться в объявлении класса в любом порядке и сколько угодно раз. Открытый член доступен во всей области видимости, где виден класс. Закрытый член доступен другим функциям-членам своего класса. Защищенный член доступен не только другим функциям-членам своего класса, но и функциям-членам класса, унаследованного непосредственно от данного класса. Если все члены открыты, то ключевое слово class можно заменить ключевым словом struct.

**3. Как выполняется конструктор при наследовании?**

При наследовании и инициализации членов класса конструкторы выполняются в следующем порядке:

а) Базовые классы инициализируются в порядке объявления.

б) Члены инициализируются в порядке объявления.

Виртуальные базовые классы создаются до того, как создан любой из производных классов, и до того, как созданы невиртуальные базовые классы. Порядок их создания – «из глубины, слева направо». Деструкторы вызываются в обратном выполнению конструкторов порядке.

**4. Дайте определение чисто виртуальной функции.**

Чисто виртуальная функция – это виртуальная функция-член, тело которой не определено. Она объявляется внутри класса:

virtual прототип\_функции = 0;

Класс, который имеет хотя бы одну виртуальную функцию, называется абстрактным базовым классом. Нельзя объявлять переменные абстрактного базового класса, но можно объявлять и полиморфно использовать указатели на такой базовый класс. Чисто виртуальный деструктор должен иметь определение.

**5. Какие операторы используются для разыменования указателя на член класса?**

Указатель на член класса имеет тип T::\*, где T – имя класса. Для разыменования указателя на член класса используются два оператора: .\* и →\*.

Рассмотрим, например, выражение z.\*fet. Здесь сначала разыменовывается указатель для получения переменной-члена, а затем происходит доступ к члену объекта z.