Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: ООП

ЛАБОРАТОРНАЯ РАБОТА № 6

«Множественное наследование»

Выполнил:

студент гр. 1ПИб-01-21оп

Коровин Даниил Андреевич

Проверил:

Доктор техничеких наук, Ершов Е.В.

Череповец, 2020 г

# Задания

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

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

# Контрольные вопросы

1. С какой целью и в каких случаях используется множественное наследование?
2. Опишите синтаксис заголовка производного класса при множественном наследовании.
3. Дайте определение ориентированного ациклического графа.
4. Где инициируются виртуальные базовые классы?
5. Что будет, если из объявлений классов Cow и Buffalo убрать ключевое слово virtual (см. последний пример в теоретических положениях)?

Код программы:

#include <iostream>

using namespace std;

// Первый базовый класс

template <class T1, class T2>

class Oven {

protected:

T1 year, month, day;

char\* name;

public:

void input();

void print();

Oven();

Oven(T2 date\_year, T1 date\_month, T1 date\_day, char\* n);

Oven(const Oven<T1, T2>& a);

~Oven<T1, T2>();

Oven<T1, T2> operator = (const Oven<T1, T2>& a);

Oven<T1, T2> operator + (Oven<T1, T2>& a);

bool operator == (const Oven<T1, T2>& a);

};

template <class T1, class T2>

void Oven<T1, T2>::input() {

cout << "Введите название вашей печи: ";

cin >> name;

cout << "Введите год изобретения вашей печи: ";

cin >> year;

cout << "Введите месяц изобретения вашей печи: ";

cin >> month;

cout << "Введите день изобретения вашей печи: ";

cin >> day;

}

template <class T1, class T2>

void Oven<T1, T2>::print() {

cout << "Название вашей печи - " << name << endl;

cout << "Год изобретения вашей печи - " << year << endl;

cout << "Месяц изобретения вашей печи - " << month << endl;

cout << "День изобретения вашей печи - " << day << endl;

cout << endl;

}

template <class T1, class T2>

Oven<T1, T2>::Oven() {

year = (T2)0;

month = (T1)0;

day = (T1)0;

name = new char[11]{ "Неизвестно" };

}

template <class T1, class T2>

Oven<T1, T2>::Oven(T2 date\_year, T1 date\_month, T1 date\_day, char\* n) {

year = date\_year;

month = date\_month;

day = date\_day;

name = new char[strlen(n) + 1];

strcpy(name, n);

}

template <class T1, class T2>

Oven<T1, T2>::Oven(const Oven<T1, T2>& a) {

year = a.year;

month = a.month;

day = a.day;

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

strcpy(name, a.name);

}

template <class T1, class T2>

Oven<T1, T2>::~Oven() {

delete[]name;

}

template <class T1, class T2>

Oven<T1, T2> Oven<T1, T2> :: operator = (const Oven& a) {

if (this == &a)

return \*this;

year = a.year;

month = a.month;

day = a.day;

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

strcpy(name, a.name);

return \*this;

}

template<class T1, class T2>

Oven<T1, T2> Oven<T1, T2> :: operator + (Oven& a)

{

Oven<T1, T2> t;

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

strcpy(t.name, name);

for (int i = strlen(name); i < strlen(name) + strlen(a.name); i++) {

t.name[i] = a.name[i - strlen(name)];

}

t.name[strlen(name) + strlen(a.name)] = '\0';

t.year = year + a.year;

t.month = month + a.month;

t.day = day + a.day;

return (t);

}

template<class T1, class T2>

bool Oven<T1, T2> :: operator == (const Oven<T1, T2>& a) {

return (year + month + day == a.year + a.month + a.day);

}

// Второй базовый класс

template <class T1, class T2>

class TypeOven {

protected:

char\* type;

public:

TypeOven<T1, T2>();

TypeOven<T1, T2>(const char\* type);

void inputType();

void printType();

};

template<class T1, class T2>

TypeOven<T1, T2>::TypeOven() {

type = new char[11]{ "Неизвестно" };

}

template<class T1, class T2>

TypeOven<T1, T2>::TypeOven(const char\* typed) {

type = typed;

}

template<class T1, class T2>

void TypeOven<T1, T2>::inputType() {

cout << "Введите тип вашей печи:";

cin >> type;

}

template<class T1, class T2>

void TypeOven<T1, T2>::printType() {

cout << "Тип вашей печи: " << type << endl;

}

// Класс-наследник

template<class T1, class T2>

class Material : public Oven<T1, T2>, public TypeOven<T1, T2> {

public:

void input();

void print();

Material();

Material(T2 date\_year, T1 date\_month, T1 date\_day, char\* n, char\* material);

~Material();

Material<T1, T2>(const Material& a);

Material& operator = (const Material<T1, T2>& a);

protected:

char\* material;

};

template<class T1, class T2>

void Material<T1, T2>::input() {

Oven<T1, T2>::input();

TypeOven<T1, T2>::inputType();

cout << "Введите название материала вашей печи : ";

cin >> material;

}

template<class T1, class T2>

void Material<T1, T2>::print() {

Oven<T1, T2>::print();

cout << endl;

TypeOven<T1, T2>::printType();

cout << "Материал вашей печи - " << material << endl;

}

template<class T1, class T2>

Material<T1, T2>::Material() : Oven<T1, T2>() {

material = new char[11]{ "Неизвестно" };

}

template<class T1, class T2>

Material<T1, T2>::Material(T2 date\_year, T1 date\_month, T1 date\_day, char\* n, char\* material) {

this->material = material;

}

template<class T1, class T2>

Material<T1, T2>::~Material() {

Oven<T1, T2>::~Oven();

}

template<class T1, class T2>

Material<T1, T2>::Material<T1, T2>(const Material& a) {

material = a.material;

this->year = a.year;

this->month = a.month;

this->day = a.day;

this->name = new char[strlen(a.name) + 1];

strcpy(this->name, a.name);

}

template<class T1, class T2>

Material<T1, T2>& Material<T1, T2> :: operator = (const Material& a) {

material = a.material;

this->year = a.year;

this->month = a.month;

this->day = a.day;

this->name = new char[strlen(a.name) + 1];

strcpy(this->name, a.name);

return (\*this);

}

int main()

{

setlocale(LC\_ALL, "Russian");

Oven<int, char> Oven1;

Material<int, char> Material1;

// Базовый класс 1

cout << "Введите информацию о печи:" << endl;

Oven1.input();

// Базовый класс 2

cout << "Введите информацию о вашей печи:" << endl;

TypeOven<int, char> Type1;

Type1.inputType();

cout << endl << "Базовые классы" << endl;

cout << "Базовый класс 1:" << endl;

Oven1.print();

cout << endl;

cout << "Базовый класс 2:" << endl;

Type1.printType();

cout << endl;

// Производный класс с множественным наследованием

cout << endl << "Класс-наследник" << endl;

cout << "Введите информацию о печи, ее материале и типе:" << endl;

Material1.input();

cout << endl;

cout << "Класс-наследник:" << endl;

Material1.print();

cout << endl;

// Объект без параметров

cout << "Объект без параметров:" << endl;

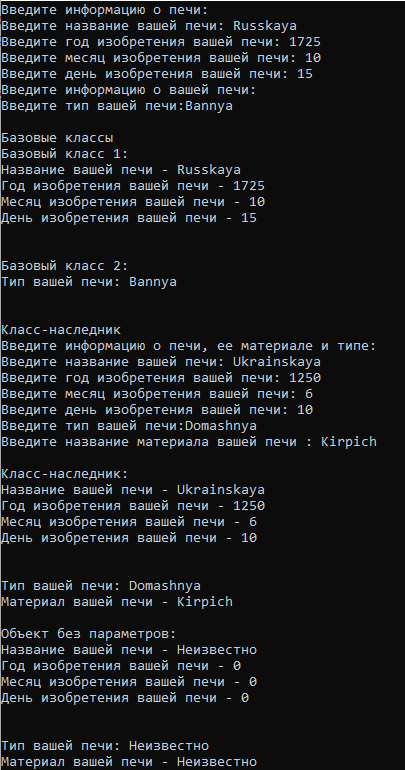
Material<int, char> Material2;

Material2.print();

system("pause");

return 0;

}



**Контрольные** **вопросы**

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

Множественное наследование делает возможным получение производного класса от нескольких базовых классов и позволяет классу перенимать их функциональность.

2.Опишите синтаксис заголовка производного класса при множественном наследовании.

Синтаксис заголовка класса расширяется, чтобы можно было использовать список базовых классов с атрибутами доступа.

class производный: public базовый\_1, private базовый\_2 {…

3.Дайте определение ориентированного ациклического графа.

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 перед любыми невиртуальными базовыми классами.

5.Что будет, если из объявлений классов Cow и Buffalo убрать ключевое слово virtual?

Ключевое слово virtual в классе Cow и классе Buffalo предотвращает многократное копирование полей данных weight, price, color из предков класса Beefalo. Так же нельзя будет использовать виртуальные функции.