Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: ООП

ЛАБОРАТОРНАЯ РАБОТА № 6

«Множественное наследование»

Выполнил:

студент гр. 1ПИб-01-21оп

Балакирев Яков Александрович

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 DB1, class DB2>

class RGD

{

protected:

DB1 spead, god, width;

char\* poezd;

public:

void vvod();

void vivod();

RGD();

RGD(DB1 n\_spead, DB1 n\_god, DB1 n\_width, char\* n\_poezd);

RGD(const RGD<DB1, DB2>& temp);

~RGD<DB1, DB2>();

RGD<DB1, DB2> operator = (const RGD<DB1, DB2>& rgd);

bool operator == (const RGD<DB1, DB2>& rgd);

};

template <class DB1, class DB2>

void RGD<DB1, DB2>::vvod()

{

cout << "Введите название: ";

cin >> poezd;

cout << "Введите скорость: ";

cin >> spead;

cout << "Введите год выпуска: ";

cin >> god;

cout << "Введите ширину: ";

cin >> width;

}

template <class DB1, class DB2>

void RGD<DB1, DB2>::vivod()

{

cout << poezd << endl;

cout << "Скорость:" << spead << endl;

cout << "Год:" << god << endl;

cout << "Ширина:" << width << endl;

cout << endl;

}

template <class DB1, class DB2>

RGD<DB1, DB2>::RGD()

{

spead = (DB1)140;

god = (DB1)2011;

width = (DB1)3480;

poezd = new char[11]{ "Swallow" };

}

template <class DB1, class DB2>

RGD<DB1, DB2>::RGD(DB1 n\_spead, DB1 n\_god, DB1 n\_width, char\* n\_poezd)

{

spead = n\_spead;

god = n\_god;

width = n\_width;

poezd = new char[strlen(n\_poezd) + 1];

strcpy(poezd, n\_poezd);

}

template <class DB1, class DB2>

RGD<DB1, DB2>::RGD(const RGD<DB1, DB2>& temp)

{

spead = temp.spead;

god = temp.god;

width = temp.width;

poezd = new char[strlen(temp.poezd) + 1];

strcpy(poezd, temp.poezd);

}

template <class DB1, class DB2>

RGD<DB1, DB2>::~RGD()

{

delete[]poezd;

}

template <class DB1, class DB2>

RGD<DB1, DB2> RGD<DB1, DB2> :: operator = (const RGD& rgd)

{

if (this == &rgd)

return \*this;

spead = rgd.spead;

god = rgd.god;

width = rgd.width;

poezd = new char[strlen(rgd.poezd) + 1];

strcpy(poezd, rgd.poezd);

return \*this;

}

template<class DB1, class DB2>

bool RGD<DB1, DB2> :: operator == (const RGD<DB1, DB2>& rgd)

{

return (spead + god + width == rgd.spead + rgd.god + rgd.width);

}

// Второй базовый класс

template <class DB1, class DB2>

class Material

{

protected:

char\* material;

public:

Material<DB1, DB2>();

Material<DB1, DB2>(const char\* temp);

void vvod\_M();

void vivod\_M();

};

template<class DB1, class DB2>

Material<DB1, DB2>::Material()

{

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

}

template<class DB1, class DB2>

Material<DB1, DB2>::Material(const char\* temp)

{

material = temp;

}

template<class DB1, class DB2>

void Material<DB1, DB2>::vvod\_M()

{

cout << "Введите материал вагона:";

cin >> material;

}

template<class DB1, class DB2>

void Material<DB1, DB2>::vivod\_M()

{

cout << "Материал: " << material << endl;

}

// Класс-наследник

template<class DB1, class DB2>

class Service : public RGD<DB1, DB2>, public Material<DB1, DB2>

{

public:

void vvod();

void vivod();

Service();

Service(DB1 n\_spead, DB1 n\_god, DB1 n\_width, char\* n\_poezd, char\* service);

~Service();

Service<DB1, DB2>(const Service& temp);

Service& operator = (const Service<DB1, DB2>& rgd);

protected:

char\* service;

};

template<class DB1, class DB2>

void Service<DB1, DB2>::vvod()

{

RGD<DB1, DB2>::vvod();

Material<DB1, DB2>::vvod\_M();

cout << "Введите род службы: ";

cin >> service;

}

template<class DB1, class DB2>

void Service<DB1, DB2>::vivod()

{

RGD<DB1, DB2>::vivod();

cout << endl;

Material<DB1, DB2>::vivod\_M();

cout << "Род службы:" << service << endl;

}

template<class DB1, class DB2>

Service<DB1, DB2>::Service() : RGD<DB1, DB2>()

{

service = new char[11]{ "passenger" };

}

template<class DB1, class DB2>

Service<DB1, DB2>::Service(DB1 n\_spead, DB1 n\_god, DB1 n\_width, char\* n\_poezd, char\* service)

{

this->service = service;

}

template<class DB1, class DB2>

Service<DB1, DB2>::~Service()

{

RGD<DB1, DB2>::~RGD();

}

template<class DB1, class DB2>

Service<DB1, DB2>::Service<DB1, DB2>(const Service& temp)

{

service = temp.service;

this->spead = temp.spead;

this->god = temp.god;

this->width = temp.width;

this->poezd = new char[strlen(temp.poezd) + 1];

strcpy(this->poezd, temp.poezd);

}

template<class DB1, class DB2>

Service<DB1, DB2>& Service<DB1, DB2> :: operator = (const Service& rgd)

{

service = rgd.service;

this->spead = rgd.spead;

this->god = rgd.god;

this->width = rgd.width;

this->poezd = new char[strlen(rgd.poezd) + 1];

strcpy(this->poezd, rgd.poezd);

return (\*this);

}

int main()

{

setlocale(LC\_ALL, "Russian");

RGD<int, char> RGD\_1;

Service<int, char> Service\_1;

// Базовый класс 1

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

RGD\_1.vvod();

// Базовый класс 2

cout << "Введите материал вагона:" << endl;

Material<int, char> Material\_1;

Material\_1.vvod\_M();

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

cout << "Базовый класс 1:" << endl;

RGD\_1.vivod();

cout << endl;

cout << "Базовый класс 2:" << endl;

Material\_1.vivod\_M();

cout << endl;

// Производный класс с множественным наследованием

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

cout << "Введите информацию о объете, материале и роде службы:" << endl;

Service\_1.vvod();

cout << endl;

cout << "Класс-наследник:" << endl;

Service\_1.vivod();

cout << endl;

// Объект без параметров

cout << "Объект по умолчанию:" << endl;

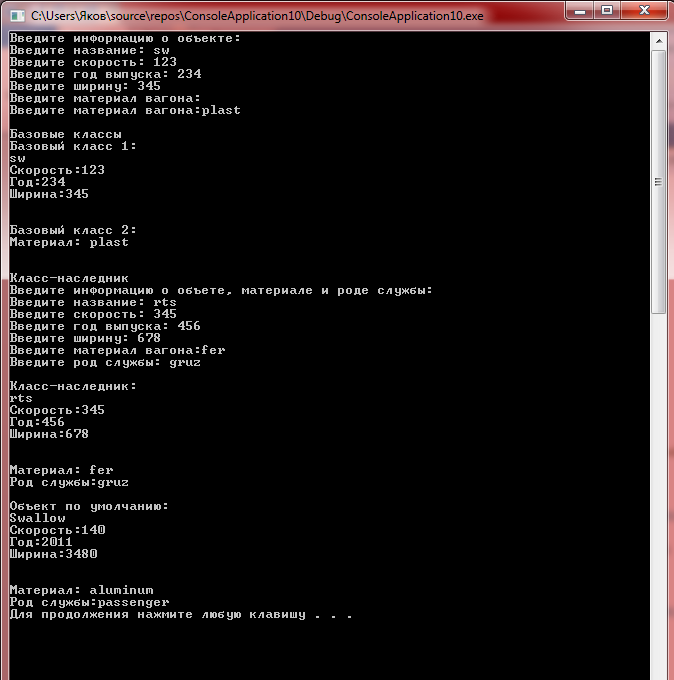
Service<int, char> Service\_2;

Service\_2.vivod();

system("pause");

return 0;

}



**Контрольные** **вопросы**

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

Множественное наследование позволяет одному дочернему классу иметь несколько родителей. Оно даёт возможность классу перенимать функциональность у множества других классов. Применяется в случае необходимости соединения функционала двух классов.

2.Опишите синтаксис заголовка производного класса при множественном наследовании.

Class ClassName : public/protected/private Parent1, public/protected/private Parent2, … public/protected/private ParentN

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 (вызывается void конструктор) перед любыми невиртуальными базовыми классами.

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

Произойдёт многократное копирование полей данных weight, price, color из предков класса Beefalo.