Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: ООП

ЛАБОРАТОРНАЯ РАБОТА № 5

«Простое наследование»

Выполнил:

студент гр. 1ПИб-01-21оп

Балакирев Яков Александрович

Проверил:

Доктор техничеких наук, Ершов Е.В.

Череповец, 2020 г

## Задания

1. Создайте производный класс для АТД, реализованного по заданию лабораторной работы 7, используя одиночное наследование.

2. Проверьте работоспособность АТД и производного класса на тестовом наборе данных.

## Контрольные вопросы

1. Дайте определение наследования.
2. Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?
3. Как выполняется конструктор при наследовании?
4. Дайте определение чисто виртуальной функции.
5. Какие операторы используются для разыменования указателя на член класса?

Код программы:

#include <iostream>

using namespace std;

// Базовый класс

template <class DB1, class DB2>

class RGD

{

protected:

DB1 spead, god, width;

char\* poezd;

public:

void vvod();

void vivod();

RGD();

RGD(DB2 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[10]{ "Swallow" };

}

template <class DB1, class DB2>

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

}

// Класс-наследник

template<class DB1, class DB2>

class Service : public RGD<DB1, DB2>

{

public:

void vvod();

void vivod();

Service();

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

cout << "Введите род службы: ";

cin >> service;

}

template<class DB1, class DB2>

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

{

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

cout << endl;

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\* n\_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

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

RGD\_1.vvod();

cout << endl << "Базовый класс:" << endl;

RGD\_1.vivod();

cout << endl;

// Конструктор с параметрами

cout << "Введите информацию о роде службы объекта:" << endl;

service\_1.vvod();

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

service\_1.vivod();

cout << endl;

// Конструктор по умолчанию

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

Service<int, char> service\_2;

service\_2.vivod();

cout << endl;

// Копирующий конструктор

cout << endl << "Копирующий конструктор:" << endl;

Service<int, char> service\_3(service\_1);

service\_3.vivod();

cout << endl;

// Оператор =

cout << endl << "Оператор =" << endl;

service\_2 = service\_1;

service\_2.vivod();

// Оператор ==

cout << endl << "Оператор ==" << endl;

if (service\_1 == service\_2) cout << "Объекты равны" << endl;

else cout << "Объекты не равны" << endl;

system("pause");

return 0;

}



**Контрольные вопросы**

1. Дайте определение наследования.

Наследование – это механизм получения нового класса на основе существующего класса. Существующий класс может быть дополнен или изменен для создания производного класса. Наследование осуществляется с помощью конструкции:

class имя\_класса:

public|protected|private)opt имя\_базового\_класса

{

объявления членов

};

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.

В C ++ есть несколько типов наследования:

· публичный (public)- публичные (public) и защищенные (protected) данные наследуются без изменения уровня доступа к ним;

· защищенный (protected) — все унаследованные данные становятся защищенными;

· приватный (private) — все унаследованные данные становятся приватными.

3. Как выполняется конструктор при наследовании?

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

1). Базовые классы инициализируются в порядке объявления.

2). Члены инициализируются в порядке объявления.

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;

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

Для разыменования указателя на член класса используются два оператора: .\* и →\*.

Сначала разыменовывается указатель для получения переменной-члена, а затем происходит доступ к члену объекта.