Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: ООП

ЛАБОРАТОРНАЯ РАБОТА № 5

«Простое наследование»

Выполнил:

студент гр. 1ПИб-01-21оп

Коровин Даниил Андреевич

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 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);

}

// Класс-наследник

template<class T1, class T2>

class Material : public Oven<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();

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

cin >> material;

}

template<class T1, class T2>

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

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

cout << endl;

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

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

Oven1.input();

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

Oven1.print();

cout << endl;

// Конструктор с параметрами

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

Material1.input();

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

Material1.print();

cout << endl;

// Конструктор по умолчанию

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

Material<int, char> Material2;

Material2.print();

cout << endl;

// Копирующий конструктор

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

Material<int, char> Material3(Material1);

Material3.print();

cout << endl;

// Оператор =

cout << endl << "Оператор =" << endl;

Material2 = Material1;

Material2.print();

// Оператор ==

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

if (Material1 == Material2) cout << "Объекты равны" << endl;

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

// Оператор +

cout << endl << "Оператор +" << endl;

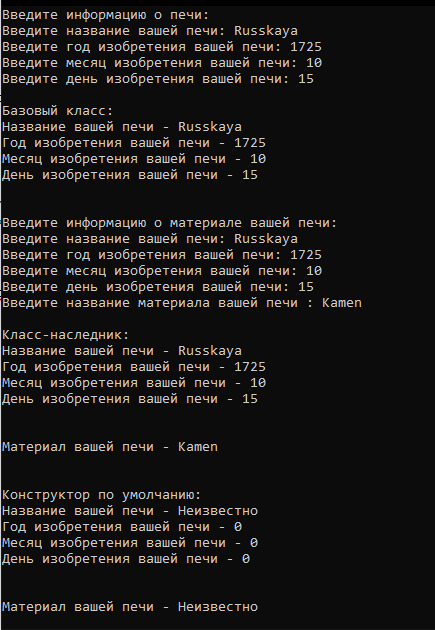
Oven1 = Material1 + Material2;

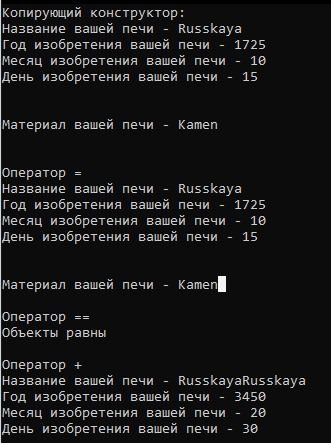
Oven1.print();

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 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:

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.