МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«РОССИЙСКИЙ БИОТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
(РОСБИОТЕХ)»
Институт промышленной инженерии, информационных технологий и
мехатроники
Кафедра «Информатика и вычислительная техника пищевых производств»
Направление: 09.03.01 Информатика и вычислительная техника

ЛАБОРАТОРНАЯ РАБОТА № 5-6

на тему:

« НАСЛЕДОВАНИЕ КЛАССОВ В C++ »

Вариант № 14

Выполнил: Студент 1 курса,
гр. 24о-090301/ИИ-1
ФИО Капралов Олег
Вадимович
Проверил: Ящун Т.В.

Москва, 2025

Цель работы

Ознакомиться с основными механизмами наследования классов и интерфейсов изучить понятия виртуальных функций и абстрактных классов.

Задание

Система управления рецептами и ингредиентами в ресторане

Разработать консольное приложение на C++, моделирующее учет рецептов и ингредиентов с поддержкой

полиморфизма, валидации данных и операций расчета себестоимости.

1. Абстрактный базовый класс Ingredient

Секции: private, protected, public.

Поля: id (уникальный идентификатор), name (название), pricePerUnit (цена за единицу в рублях).

Геттеры/сеттеры для всех полей.

Конструкторы: по умолчанию, копирования, преобразования (создание по id).

Виртуальный деструктор.

Чисто виртуальные getCalories() – возвращает калорийность на единицу, getType() – возвращает тип ингредиента (например, "базовый").

Переопределяемые методы: validate() – проверка pricePerUnit > 0 .

Перегрузка операторов: operator== – сравнение по id .

Переопределяемые методы для ввода и вывода информации (объявленные в секции protection).

Дружественные операторы вывода информации об объекте и ввода данных объекта с консоли, содержащие вызов переопределяемых методов ввода и вывод информации.

2. Класс Vegetable (наследник Ingredient)

Секции: private, protected, public.

Поля: isOrganic (органический продукт), vitamins (список витаминов, например, "A, C").

Конструкторы: по умолчанию, копирования, преобразования (создание по id и name).

Переопределение методов: ввода и вывода информации об объекте, getCalories() – возвращает 30

калорий для органических, 25 для остальных, getType() – возвращает "Овощ", validate() – проверка, что name не пустая строка.

Дополнительный метод: checkSeason() – возвращает сезонность (лето/зима).

Перегрузка операторов: operator+ – объединение витаминов из двух овощей.

Программный код методов, содержащих более 1-го оператора, должен быть вынесен за пределы описания класса или в отдельный модуль – в объявлении класса должно быть только описание прототипов

таких методов.

3. Класс Meat (наследник Ingredient, final)

Секции: private, protected, public.

Поля: fatPercentage (процент жирности), origin (происхождение: "говядина", "курица").

Переопределение методов: ввода и вывода информации об объекте, getCalories() – формула:

fatPercentage * 10 + 100, getType() – возвращает "Мясо", validate() – проверка

fatPercentage ∈ [1, 50].

Дополнительный метод: recommendCookingTime() – расчет времени приготовления на основе

Словестное описание программы:

Эта программа — цифровая кухонная книга для ресторана. Она помогает шеф-повару вести учет продуктов и рецептов.

Что она умеет:

Хранить данные о продуктах: овощах (морковь, лук) и мясе (говядина, курица).

Запоминать их цену, калорийность и особенности (например, жирность мяса или витамины в овощах).

Проверять, правильно ли заполнены данные (цена не минусовая, жирность в разумных пределах).

Собирать рецепты из ингредиентов и считать их себестоимость.

Подсказывать полезную информацию: время готовки мяса, сезонность овощей, калорийность блюд.

Скриншоты:

1. Абстрактный базовый класс Ingredient

```
class Ingredient {
protected:
    string id;
    string name;
    double pricePerUnit;

    virtual ostream& print(ostream& os) const { /*...*/ }
    virtual istream& input(istream& is) { /*...*/ }

public:
    Ingredient() : id(""), name(""), pricePerUnit(0) {}
    Ingredient(const string& id, const string& name, double price)
        : id(id), name(name), pricePerUnit(price) {}
    Ingredient(const Ingredient& other)
        : id(other.id), name(other.name), pricePerUnit(other.pricePerUnit) {}

    virtual ~Ingredient() {}

    string getId() const { return id; }
    string getName() const { return name; }
    double getPricePerUnit() const { return pricePerUnit; }

    void setId(const string& id) { this->id = id; }
    void setName(const string& name) { this->name = name; }
    void setPricePerUnit(double price) { this->pricePerUnit = price; }

    virtual double getCalories() const = 0;
    virtual string getType() const = 0;

    virtual bool validate() const {
        return pricePerUnit > 0 && !id.empty() && !name.empty();
    }

    bool operator==(const Ingredient& other) const {
        return id == other.id;
    }

    friend ostream& operator<<(ostream& os, const Ingredient& ing) {
        return ing.print(os);
    }

    friend istream& operator>>(istream& is, Ingredient& ing) {
        return ing.input(is);
    }
};
```

2. Класс Vegetable (наследник Ingredient)

```
class Vegetable : public Ingredient {
protected:
    bool isOrganic;
    string vitamins;

    ostream& print(ostream& os) const override { /*...*/ }
    istream& input(istream& is) override { /*...*/ }

public:
    Vegetable() : Ingredient(), isOrganic(false), vitamins("") {}
    Vegetable(const string& id, const string& name, double price, bool organic, const string& vit)
        : Ingredient(id, name, price), isOrganic(organic), vitamins(vit) {}
    Vegetable(const Vegetable& other)
        : Ingredient(other), isOrganic(other.isOrganic), vitamins(other.vitamins) {}

    ~Vegetable() override {}

    double getCalories() const override {
        return isOrganic ? 30 : 25;
    }

    string getType() const override {
        return "Овощ";
    }

    bool validate() const override {
        return Ingredient::validate() && !vitamins.empty();
    }

    string checkSeason() const {
        if (name.find("картофель") != string::npos || name.find("морковь") != string::npos) {
            return "зима";
        }
        return "лето";
    }

    Vegetable operator+(const Vegetable& other) const {
        string newVitamins = vitamins;
        if (!newVitamins.empty() && !other.vitamins.empty()) {
            newVitamins += ", ";
        }
        newVitamins += other.vitamins;
        return Vegetable(id + "_" + other.id, name + " и " + other.name,
                         (pricePerUnit + other.pricePerUnit) / 2,
                         isOrganic && other.isOrganic, newVitamins);
    }
};
```

3. Класс Meat (наследник Ingredient, final)

```
class Meat final : public Ingredient {
private:
    double fatPercentage;
    string origin;

    ostream& print(ostream& os) const override { /*...*/ }
    istream& input(istream& is) override { /*...*/ }

public:
    Meat() : Ingredient(), fatPercentage(0), origin("") {}
    Meat(const string& id, const string& name, double price, double fat, const string& orig)
        : Ingredient(id, name, price), fatPercentage(fat), origin(orig) {}
    Meat(const Meat& other)
        : Ingredient(other), fatPercentage(other.fatPercentage), origin(other.origin) {}

    ~Meat() override {}

    double getCalories() const override {
        return fatPercentage * 10 + 100;
    }

    string getType() const override {
        return "Мясо";
    }

    bool validate() const override {
        return Ingredient::validate() && fatPercentage >= 1 && fatPercentage <= 50 && !origin.empty();
    }

    int recommendCookingTime() const {
        if (origin == "говядина") {
            return fatPercentage < 10 ? 30 : 45;
        } else if (origin == "курица") {
            return 25;
        }
        return 30;
    }

    Meat& operator+=(double increase) {
        fatPercentage += increase;
        if (fatPercentage > 50) fatPercentage = 50;
        return *this;
    }
};
```

4. Класс CompositeIngredient (наследник Ingredient и NutritionCalculator)

```
class CompositeIngredient : public Ingredient, public NutritionCalculator {
private:
    vector<shared_ptr<Ingredient>> components;
    bool isReadyToServe;

    ostream& print(ostream& os) const override { /*...*/ }
    istream& input(istream& is) override { /*...*/ }

public:
    CompositeIngredient() : Ingredient(), isReadyToServe(false) {}
    CompositeIngredient(const string& id, const string& name, double price, bool ready)
        : Ingredient(id, name, price), isReadyToServe(ready) {}

    ~CompositeIngredient() override {}

    double getCalories() const override {
        return calculateTotalCalories();
    }

    string getType() const override {
        return "Составной ингредиент";
    }

    bool validate() const override {
        if (components.size() > 10) return false;
        return Ingredient::validate();
    }

    void addComponent(shared_ptr<Ingredient> ingredient) { /*...*/ }
    void addDecoration(const string& decoration) { /*...*/ }

    shared_ptr<Ingredient> operator[](size_t index) {
        if (index < components.size()) {
            return components[index];
        }
        return nullptr;
    }

    double calculateTotalCalories() const { /*...*/ }
};
```

6. Демонстрация динамического полиморфизма

```
void demonstratePolymorphism() {
    if (ingredients.empty()) {
        cout << "Нет ингредиентов для демонстрации.\n";
        return;
    }

    cout << "=== Демонстрация полиморфизма ===\n";
    for (const auto& ing : ingredients) {
        cout << *ing << "\n";
        cout << "Тип: " << ing->getType() << "\n";
        cout << "Калории: " << ing->getCalories() << "\n";

        if (auto veg = dynamic_cast<Vegetable*>(ing.get())) {
            cout << "Сезонность: " << veg->checkSeason() << "\n";
        } else if (auto meat = dynamic_cast<Meat*>(ing.get())) {
            cout << "Рекомендуемое время приготовления: " << meat->recommendCookingTime() << " мин\n";
        } else if (auto comp = dynamic_cast<CompositeIngredient*>(ing.get())) {
            cout << "Общее количество калорий: " << comp->calculateTotalCalories() << "\n";
        }
        cout << "----------------------------\n";
    }
}
```

Код на bash для теста:

```
#!/bin/bash

find_binary() {
    local binary_name="restaurant_management"
    for path in "./" "/usr/bin/" "/usr/local/bin/"; do
        if [[ -x "${path}${binary_name}" ]]; then
            echo "${path}${binary_name}"
            return 0
        fi
    done
    echo "Ошибка: бинарник $binary_name не найден" >&2
    exit 1
}

run_test() {
    local test_name="$1"
    local input_commands="$2"
    local expected_output="$3"
    local binary_path="$4"

    echo -e "
Тест: $test_name"

    expect << EOF
        set timeout 15
        spawn "$binary_path"
        sleep 2

        expect {
            "Выберите действие:" { }
            timeout {
                puts "Ошибка: меню не появилось"
                exit 1
            }
        }
        sleep 1

        send "$input_commands
"
        sleep 2

        expect {
            -re "$expected_output" {
                puts "Успех"
                exit 0
            }
            timeout {
                puts "Ошибка: не получен ожидаемый вывод"
                exit 1
            }
            eof {
                puts "Ошибка: программа завершилась неожиданно"
                exit 1
            }
        }
EOF
}

main() {
    local binary_path=$(find_binary)
    echo "Исполняемый файл: $binary_path"
    echo "Запуск тестов..."

    run_test "Добавить овощ"              "2
V1
Морковь
50
да
A,B
1
0
"               "Овощ.*Морковь.*Цена: 50 руб.*Органический: Да.*Витамины: A,B"              "$binary_path"

    run_test "Добавить мясо"              "3
M1
Говядина
300
говядина
15
1
0
"               "Мясо.*Говядина.*Цена: 300 руб.*Происхождение: говядина.*Жирность: 15%"              "$binary_path"

    run_test "Создать составной ингредиент"              "4
C1
Салат
0
да
1
2
0
1
0
"               "Составной ингредиент.*Салат.*Компоненты:.*Морковь.*Говядина"              "$binary_path"

    run_test "Показать полиморфизм"              "5
0
"               "Тип: Овощ.*Тип: Мясо.*Тип: Составной ингредиент"              "$binary_path"

    run_test "Объединить овощи"              "2
V2
Картофель
40
нет
C,D
6
1
2
0
"               "Овощи успешно объединены!.*Морковь и Картофель.*Витамины: A,B, C,D"              "$binary_path"

    echo -e "
Все тесты выполнены"
}

main "$@"
```

```
Исполняемый файл: ./restaurant_management
Запуск тестов...

Тест: Добавить овощ
spawn ./restaurant_management

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: 2
V1
Морковь
50
да
A,B
1
0


=== Добавить овощ ===
Введите ID ингредиента: Введите название ингредиента: Введите цену за единицу (руб.): Органический продукт? (да/нет): Введите список витаминов (через запятую): Овощ успешно добавлен!

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие:
=== Показать все ингредиенты ===
=== Список всех ингредиентов ===
1. Овощ [ID: V1, Название: Морковь, Цена: 50 руб., Органический: Да, Витамины: A,B]
   Тип: Овощ, Калории: 30

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: Работа завершена. До свидания!
Успех

Тест: Добавить мясо
spawn ./restaurant_management

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: 3
M1
Говядина
300
говядина
15
1
0


=== Добавить мясо ===
Введите ID ингредиента: Введите название ингредиента: Введите цену за единицу (руб.): Введите происхождение (говядина/курица и т.д.): Введите процент жирности (1-50): Мясо успешно добавлено!

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие:
=== Показать все ингредиенты ===
=== Список всех ингредиентов ===
1. Мясо [ID: M1, Название: Говядина, Цена: 300 руб., Происхождение: говядина, Жирность: 15%]
   Тип: Мясо, Калории: 250

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: Работа завершена. До свидания!
Успех

Тест: Создать составной ингредиент
spawn ./restaurant_management
```

```
=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: 4
C1
Салат
0
да
1
2
0
1
0


=== Создать составной ингредиент ===
Введите ID ингредиента: Введите название ингредиента: Введите цену за единицу (руб.): Введите цену за единицу (руб.): Введите цену за единицу (руб.): Готов к подаче? (да/нет): Список ингредиентов пуст
.
Выберите ингредиенты для добавления (через пробел, 0 для завершения):
Составной ингредиент успешно создан!

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие:
=== Показать все ингредиенты ===
=== Список всех ингредиентов ===
1. Составной ингредиент [ID: C1, Название: Салат, Цена: 1 руб., Готов к подаче: Нет]
Компоненты:

   Тип: Составной ингредиент, Калории: 0

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: Работа завершена. До свидания!
Ошибка: программа завершилась неожиданно
```

Тест: Показать полиморфизм

```
=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие: 2
V2
Картофель
40
нет
C,D
6
1
2
0


=== Добавить овощ ===
Введите ID ингредиента: Введите название ингредиента: Введите цену за единицу (руб.): Органический продукт? (да/нет): Введите список витаминов (через запятую): Овощ успешно добавлен!

=== Меню управления ингредиентами ===
1. Показать все ингредиенты
2. Добавить овощ
3. Добавить мясо
4. Создать составной ингредиент
5. Демонстрация полиморфизма
6. Объединить овощи
0. Выход
Выберите действие:
=== Объединить овощи ===
=== Список всех ингредиентов ===
1. Овощ [ID: V2, Название: Картофель, Цена: 40 руб., Органический: Нет, Витамины: C,D]
   Тип: Овощ, Калории: 25
Выберите два овоща для объединения (введите номера через пробел):
```