

## Лабораторная работа 10. Агрегация и композиция

### 1. Цель работы

Изучить отношения агрегации и композиции, получить навыки реализации агрегации и композиции на C++.

### 2. Краткие теоретические сведения

#### Типы отношений между классами

Языки программирования выработали несколько общих подходов к выражению отношений этих трех типов. В частности, большинство объектно-ориентированных языков непосредственно поддерживают разные комбинации следующих видов отношений:

- ассоциация;
- наследование;
- агрегация;
- использование;
- параметризация.

#### Выбор между агрегацией и наследованием

Среди видов отношений между классами наиболее часто встречаются *наследование* и *агрегация*.

Наследование применяется в тех случаях, когда один класс является уточненной, более специализированной формой другого. Чтобы определить, является ли понятие  $X$  уточненным вариантом понятия  $Y$ , можно составить предложение " $X$  является экземпляром  $Y$ ". Если утверждение звучит корректно (т.е. есть соответствие вашим представлениям о решаемой задаче), то можно заключить, что  $X$  и  $Y$  связаны отношением наследования.

Отношение агрегации ("*включать как часть*") имеет место, когда второе понятие является составной частью первого, но оба эти понятия не совпадают ни в каком смысле независимо от уровня общности абстракции. Например, автомобиль `Car` *имеет* двигатель `Engine`, хотя ясно, что это не тот случай, когда `Car` *является* экземпляром `Engine` или `Engine` *является* экземпляром `Car`. Класс `CCar` тем не менее *является* экземпляром класса автомобилей `CVehicle`, который в свою очередь *является* экземпляром класса транспортных средств `CMeansOfTransportation`. Аналогично проверке для наследования, чтобы проверить отношение агрегации, можно составить предложение " $X$  включает  $Y$  как часть" и оценить его корректность.

В большинстве случаев различие между агрегацией и наследованием очевидно.

#### Демонстрация агрегации и наследования

Чтобы проиллюстрировать агрегацию и наследование, рассмотрим построение абстрактного типа данных "Множество" (класс `CSet`) на основе класса "Связный список" (`CList`). Объекты класса `CList` содержат списки

целочисленных величин. Допустим, что имеется класс CList со следующим интерфейсом:

```
class CList {
public:
    CList();

    void AddToFront(int);
    int  FirstElement();
    int  Length();
    bool IsIncludes(int);
    int  Remove(int);
    ...
};
```

Класс "Список" позволяет добавлять новый узел в начало списка, получать значение первого узла, вычислять количество узлов, проверять, содержится ли значение в списке, и удалять узел с заданным значением из списка.

Предположим, что необходимо создать класс "Множество", который позволяет выполнять такие операции, как добавление элемента в множество, определение количества элементов, проверка принадлежности к множеству.

### **Использование агрегации**

Сначала рассмотрим, как построить класс "Множество" с помощью агрегации. Известно, что класс инкапсулирует внутри себя состояние и поведение. Когда для создания нового класса используется агрегация, то существующий класс включается в новый класс в виде переменной-члена (поля). Ниже показано, что внутри класса CSet заведена переменная-член data – объект класса CList.

```
class CSet {
public:
    CSet();

    void Add(int);
    int  Size();
    bool IsIncludes(int)

private:
    CList data;
};
```

Поскольку объект класса CList является переменной-членом класса "Множество", то этот объект надо инициализировать в конструкторе класса CSet. Если у конструктора объекта нет параметров, то он вызывается автоматически (как в данном случае), иначе его приходится вызывать явно.

Функции-члены класса CSet реализованы с использованием функций-членов класса CList. Например, функции-члены IsIncludes и Size для множества просто вызывают соответствующие функции-члены у списка:

```
int CSet::Size()
{
    return data.Length();
}

int CSet::IsIncludes( int newValue )
```

```

{
    return data.IsIncludes( newValue );
}

```

Функция-член `Add()` для добавления нового элемента в множество оказывается более сложной, т.к. сначала нужно убедиться, что в множестве данный элемент отсутствует (в множестве м.б. единственный элемент с заданным значением):

```

void CSet::Add( int newValue )
{
    if ( !IsIncludes( newValue ) )
        data.AddToFront( newValue );
}

```

Приведенный пример показывает, как агрегация помогает повторному использованию компонент в программах. Большая часть работы, связанной с хранением данных, в классе `CSet` прodelывается существовавшим ранее классом `CList`. Графическое изображение отношения агрегации показано на рис. 1.

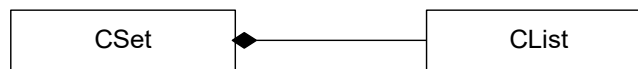


Рис. 1. Агрегация.

Следует понимать, что при создании нового класса с помощью агрегации классы `CList` и `CSet` будут абсолютно различны, и ни один из них не является уточнением другого.

### Использование наследования

При использовании наследования новый класс может быть объявлен как подкласс существующего класса. В этом случае все области данных и функции, связанные с исходным классом, автоматически переносятся в подкласс. Подкласс может определять дополнительные переменные и функции. Он переопределяет некоторые функции исходного класса, которые были объявлены в нем как виртуальные.

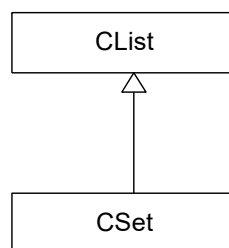


Рис. 2. Наследование.

Графическое изображение наследования приведено на рис. 2. Ниже показано, как можно применить наследование для создания класса `CSet` в форме подкласса класса `CList`. Подкласс является расширением существующего класса `CList`, поэтому все функции-члены списка оказываются применимы и к множеству.

```

class CSet : public CList {
public:

```

```

        CSet();
        void Add(int);
        int Size();
    };

```

Обратите внимание, что в подклассе не определено никаких новых переменных-членов. Вместо этого переменные-члены класса CList будут использоваться для хранения элементов множества.

Аналогично функции-члены родительского класса можно использовать в подклассе, поэтому не надо объявлять функцию-член IsIncludes – в классе CList есть функция-член с таким же именем и подходящим поведением. Функция-член для добавления в множество нового элемента выполняет вызовы двух функций-членов класса CList:

```

void CSet::Add( int newValue )
{
    if ( !IsIncludes( newValue ) )
        AddToFront( newValue );
}

```

Сравните этот вариант функции Add с вариантом из п. 3.1. Оба вида отношений – агрегация и наследование – являются мощными механизмами для многократного использования кода. В отличие от агрегации, наследование содержит неявное предположение, что подклассы на самом деле являются подтипами. Это значит, что объекты подкласса должны вести себя так же, как и объекты родительского класса.

### **Сравнение агрегации и наследования**

Оба вида отношений между классами – агрегацию и наследование – можно применить для реализации класса "Множество". На рассмотренном примере перечислим некоторые недостатки и преимущества двух подходов.

- Агрегация более проста. Ее преимущество заключается в том, что она ясно показывает, какие точно функции-члены будут содержаться в классе. Из описания класса CSet становится очевидно, что для объектов-множеств предусмотрены только операции добавления элемента, проверки на наличие элемента и определение количества элементов в множестве. Это справедливо независимо от того, какие функции-члены определены в классе CList.
- При наследовании функции-члены нового класса дополняют и, возможно, переопределяют набор функций-членов родительского класса. Таким образом, чтобы точно знать, какие функции-члены есть у подкласса, программист должен изучить описание родительского класса. Например, из описания класса CSet не видно, что у множеств можно выполнять проверку на наличие элемента в множестве (функция-член IsIncludes). Это можно понять только после изучения описания родительского класса CList. Т.о., у наследования есть неприятная особенность: чтобы полностью понять поведение и свойства подкласса, программист должен изучать описание одного или нескольких родительских классов.
- С другой стороны, указанную выше особенность наследования можно считать преимуществом: описание класса получается более кратким, чем в случае

агрегации. Применяя наследование, оказывается ненужным писать все функции для доступа к переменным-членам родительского класса. Наследование также часто обеспечивает большую функциональность. Например, применение наследования в нашем случае делает доступным для множеств не только проверку `IsIncludes`, но и функцию `Remove`.

- Наследование не запрещает пользователям манипулировать новыми классами через вызовы функций-членов родительского класса, даже если эти функции не вполне подходят под идеологию потомка. Например, при использовании наследования для класса `CSet` пользователи смогут добавлять элементы в множество с помощью унаследованной от класса `CList` функции `AddToFront`.
- При агрегации тот факт, что класс `CList` используется для хранения элементов множеств, – просто скрытая деталь реализации. Т.о., можно легко переписать класс `CSet` более эффективным способом (например, на основе массива) с минимальным воздействием на пользователей класса `CSet`. Если же пользователи рассчитывают на то, что класс `CSet` – это более специализированный вариант класса `CList`, то такие изменения было бы трудно реализовать.

**Вопрос:**

Как в конкретном случае выбрать один из двух механизмов реализации?

### 3. Задание (5 баллов)

Для задачи индивидуального варианта создать программу, реализующую отношение агрегации или композиции. Для всех закрытых полей определить метод установки значений (при недопустимых аргументах возвращать «false» и выдавать текст ошибки на экран) и метод чтения. Все классы следует наделить конструкторами и деструктором. Необходимо явно реализовать конструктор копирования и перегрузить оператор присваивания

### 4. Варианты заданий

1. Класс «КОМНАТА», содержит закрытые поля: сведения о метраже, высоте потолков и количестве окон и метод подсчета площади комнаты. Класс «КВАРТИРА», содержит закрытые поля: номер этажа, объекты класса «КОМНАТА», метод подсчета площади квартиры и метод вывода информации о комнатах квартиры.

2. Класс «СТУДЕНТ» содержит закрытые поля: номер студенческого билета, Фамилия, Имя, Отчество, дата рождения, массив из пяти оценок и метод подсчета среднего балла. Класс «СТУДЕНЧЕСКАЯ ГРУППА» содержит закрытые поля: название группы, курс, объекты класса «СТУДЕНТ», метод подсчета среднего балла для группы и метод вывода списка студентов, отсортированный по фамилиям в алфавитном порядке.

3. Класс «АВТОМОБИЛЬ» содержит закрытые поля: гос. номер, цвет, фамилия владельца и признак присутствия на стоянке и метод вывода сведений об автомобиле. Класс «АВТОСТОЯНКА», содержит закрытые поля: название автостоянки, объекты класса «АВТОМОБИЛЬ», методы поиска автомобиля по разным критериям и методы вывода списка присутствующих и отсутствующих на стоянке автомобилей.

4. Класс «ЗАПИСЬ» содержит закрытые поля: фамилия, имя, номер телефона, дата рождения. Класс «ЗАПИСНАЯ КНИЖКА», содержащий закрытые поля: фамилия и инициалы владельца, объекты класса «ЗАПИСЬ», метод поиска номера телефона и даты

рождения по фамилии и имени, а также метод получения списка людей, номер телефона которых начинается на три заданные цифры.

5. Класс «САМОЛЕТ» содержит закрытые поля: шестизначный номер рейса, название пункта назначения, время отправления и метод вывода сведений о самолете. Класс «АЭРОПОРТ» содержит закрытые поля: название аэропорта, объекты класса «САМОЛЕТ», метод поиска информации о самолетах, отправляющихся в течение часа после введенного с клавиатуры времени и метод вывода информации о самолетах, отправляющихся в заданный пункт назначения. Информация должна быть отсортирована по времени отправления.

6. Класс «КОМПЛЕКТУЮЩИЕ» содержит закрытые поля: название, цена, гарантийный срок и метод вывода сведений о комплектующем. Класс «КОМПЬЮТЕР» содержит закрытые поля: серийный номер, марка, объекты класса «КОМПЛЕКТУЮЩИЕ», метод замены комплектующих компьютера и метод вывода списка комплектующих компьютера.

7. Класс «ПЕСНЯ» содержит закрытые поля: номер песни, название песни, композитор, поэт, исполнитель и метод вывода сведений о песне. Класс «ДИСКИ» содержит закрытые поля: название диска, цена, объекты класса «ПЕСНЯ», метод поиска песни по названию, метод поиска песни по исполнителю и метод вывода списка песен диска, упорядоченный по названию песни.

8. Класс «ПОЕЗД» содержит закрытые поля: номер поезда, название пункта назначения, время отправления и метод вывода сведений о поезде. Класс «ВОКЗАЛ» содержит закрытые поля: название вокзала, объекты класса «ВОКЗАЛ», метод поиска информации о поездах, отправляющихся после введенного с клавиатуры времени и метод вывода информации о поездах, отправляющихся в заданный пункт назначения. Информация должна быть отсортирована по времени отправления.

9. Класс «ТОВАР» содержит закрытые поля: название товара, стоимость единицы товара в рублях, количество единиц товара и метод подсчета стоимости товара. Класс «СКЛАД» содержит закрытые поля: название склада и объекты класса «ТОВАР», метод подсчета стоимости всего товара и метод вывода списка товаров, отсортированный по названию товара.

10. Класс «ЛЕКАРСТВО» содержит закрытые поля: название лекарства, цена, показания к применению, противопоказания и метод вывода сведений о лекарстве. Класс «АПТЕКА» содержит закрытые поля: название аптеки, номер аптеки, объекты класса «ЛЕКАРСТВО», метод поступления новых лекарств в аптеку, метод поиска лекарства по названию и метод вывода списка лекарств аптеки.

11. Класс «АВТОБУС» содержит закрытые поля: номер автобуса, название пункта назначения, время отправления и метод вывода сведений об автобусе. Класс «АВТОВОКЗАЛ» содержит закрытые поля: название автовокзала, объекты класса «АВТОБУС», метод поиска информации об автобусах, отправляющихся после введенного с клавиатуры времени в заданный пункт назначения. Информация должна быть отсортирована по времени отправления.

12. Класс «ЖИВОТНОЕ» содержит закрытые поля: кличка животного, название вида, возраст, вес и метод вывода сведений о животном. Класс «ЗООПАРК» содержит закрытые поля: название зоопарка, объекты класса «ЖИВОТНОЕ», метод поступления новых животных в зоопарк, метод убийства животных из зоопарка и метод вывода списка животных зоопарка.

13. Класс «ВЕРТОЛЕТ» содержит закрытые поля: пятизначный идентификатор, название компании-владельца, время полета и метод вывода сведений о вертолете. Класс «АЭРОДРОМ» содержит закрытые поля: название аэродрома, объекты класса «ВЕРТОЛЕТ», метод поиска информации о вертолетах, вылетающих в течение часа после введенного с клавиатуры времени и метод вывода информации о вертолетах, зарезервированных для частных рейсов. Информация должна быть отсортирована по

времени отправления.

14. Класс «КОМПЛЕКТУЮЩИЕ» содержит закрытые поля: название, цена, гарантийный срок и метод вывода сведений о комплектующем. Класс «СТИРАЛЬНАЯ МАШИНА» содержит закрытые поля: серийный номер, марка, объекты класса «КОМПЛЕКТУЮЩИЕ», метод замены комплектующих стиральной машины и метод вывода списка комплектующих стиральной машины.

15. Класс «СТИХОТВОРЕНИЕ» содержит закрытые поля: название, поэт, тема (природа, любовная лирика и т.п.) и метод вывода сведений о песне. Класс «СБОРНИК» содержит закрытые поля: название сборника, цена, объекты класса «СТИХОТВОРЕНИЯ», метод поиска стихотворения по названию, метод поиска стихотворения по исполнителю и метод вывода списка стихотворений сборника, упорядоченный по названию.

16. Класс «ПОЕЗД» содержит закрытые поля: номер поезда, название пункта назначения, время отправления и метод вывода сведений о поезде. Класс «ВОКЗАЛ» содержит закрытые поля: название вокзала, объекты класса «ВОКЗАЛ», метод поиска информации о поездах, отправляющихся после введенного с клавиатуры времени и метод вывода информации о поездах, отправляющихся в заданный пункт назначения. Информация должна быть отсортирована по времени отправления.

17. Класс «ТОВАР» содержит закрытые поля: название товара, стоимость единицы товара в рублях, количество единиц товара и метод подсчета стоимости товара. Класс «МАГАЗИН» содержит закрытые поля: название торговой сети, номер магазина и объекты класса «ТОВАР», метод подсчета стоимости всего товара и метод вывода списка товаров, отсортированный по названию товара.

18. Класс «ЛЕКАРСТВО» содержит закрытые поля: название лекарства, цена, показания к применению, противопоказания и метод вывода сведений о лекарстве. Класс «АПТЕКА» содержит закрытые поля: название аптеки, номер аптеки, объекты класса «ЛЕКАРСТВО», метод поступления новых лекарств в аптеку, метод поиска лекарства по названию и метод вывода списка лекарств аптеки.

19. Класс «ТАКСИ» содержит закрытые поля: номер такси, ФИО водителя, время работы и метод вывода сведений об автобусе. Класс «АВТОБАЗА» содержит закрытые поля: название автобазы, объекты класса «ТАКСИ», метод поиска информации о такси на рейсе и на базе после введенного с клавиатуры времени работы.

20. Класс «ЗАПИСЬ» содержит закрытые поля: фамилия, имя, номер телефона, дата рождения. Класс «ЗАПИСНАЯ КНИЖКА», содержащий закрытые поля: фамилия и инициалы владельца, объекты класса «ЗАПИСЬ», метод поиска номера телефона и даты рождения по фамилии и имени, а также метод получения списка людей, номер телефона которых начинается на три заданные цифры.

## **5. Контрольные вопросы**

1. В чем разница между композицией и агрегацией?
2. В чем разница между наследованием и агрегацией?
3. В случае реализации отношений агрегации и композиции как может храниться информация о связанных объектах? В чем отличие?
4. Привести 3 примера отношения агрегации.
5. Привести 3 примера отношения композиции.