Д.Е. Яблоков

ПРИМЕНЕНИЕ ОБОБЩЕННЫХ КОНЦЕПЦИЙ ИТЕРАТОРОВ И ФУНКЦИОНАЛЬНЫХ АДАПТЕРОВ ДЛЯ СОЗДАНИЯ АЛГОРИТМОВ С РАЗВИТОЙ СЕМАНТИКОЙ ОБРАБОТКИ ДАННЫХ

(Самарский университет)

Обобщенное программирование – это методология проектирования и реализации программ, суть которой состоит в разделении структур данных и алгоритмов через абстрактные требования. Спецификация абстрактных требований в обобщенном программировании подобна известному понятию абстрактного типа данных. Фактически абстрактный тип данных это и есть спецификация типа, состоящая из описания доступных операций и задающая их семантику с учетом предусловий и постусловий функционирования, удовлетворяющих зафиксированным в сигнатуре типа требованиям. В обобщенном программировании понятие абстрактного типа данных существенно расширяется. Вместо определения спецификации для отдельного типа производится описание семейства абстрактных типов, которое определяет их общий интерфейс и семантическое поведение. Набор требований, описывающих интерфейс и семантическое поведение семейства абстрактных типов данных называется концепцией. Классическим примером набора абстрактных требований к типу может служить упрощенная концепция целого числа, поддерживающего такие функциональные особенности как: операции унарной и бинарной арифметики, операции битовой логики, операции бинарного сдвига, операции отношений и операции присваивания. Есть много способов реализации компонента, удовлетворяющего требованиям концепции целого числа, но детали реализации при использовании этого компонента не важны до тех пор, пока она удовлетворяет заявленной спецификации.

Алгоритмы, разработанные в обобщенном стиле, могут быть применены к любым типам, удовлетворяющим требованиям концепции. Такая особенность, выражающаяся в использовании различных типов для одного и того же параметра функции, называется статическим полиморфизмом, когда все неоднозначности, связанные с использованием родственных элементов программы, разрешаются во время компиляции.

В объектно-ориентированном программировании требования к интерфейсу могут быть оформлены на уровне базовой абстракции, а полиморфизм реализуется с помощью виртуальных функций и наследования. Конкретные классы наследуют абстрактному базовому классу или интерфейсу и определяют реализацию и поведение этих функций. Для обобщения аргументы функций объявляются в терминах базового класса, а во время исполнения, в теле функции, вызовы соответствующих операций осуществляются уже для конкретного типа объекта. Таким образом любой экземпляр подтипа абстрактного базового

класса или интерфейса может быть использован в качестве параметра такой обобщенной функции.

В случае статического полиморфизма времени компиляции, который в полной мере поддерживается, например, в таком языке как C++, информация о типе параметра шаблона полностью известна алгоритму, который может быть размещен в другой единице трансляции относительно сигнатуры типа передаваемого аргумента. В такой ситуации семантика обработки данных при разработке алгоритма может быть достаточно развитой и в полной мере использовать все возможности, которые должны присутствовать в описании типа для соответствия предъявляемым к нему требованиям. В противовес этому, при реализации обобщенного алгоритма в чисто объектно-ориентированных языках, информация о типе передаваемого аргумента не является полной и соответствует либо перечню операций базовой абстракции, либо виду ограничения, вводимого на этапе компиляции для спецификации типов данных, которые могут быть использованы в качестве параметра шаблона. В итоге, в чисто объектно-ориентированных языках, не в полной мере поддерживающих средства обобщенного программирования, реализация алгоритмов с развитой семантикой обработки данных становится весьма затруднительной, а в некоторых случаях даже невозможной из-за отсутствия полной информации о свойствах типов передаваемых параметров.

В качестве решения, обеспечивающего надлежащий уровень гибкости при разработке обобщенных алгоритмов в таких языках как Java и C#, предлагается использование обобщенных концепций. Они объединяют в себе и объектно-ориентированный и обобщенный подходы и позволяют специфицировать требования к аргументам функций в виде базовых абстракций с дальнейшей возможностью их применения к множеству типов данных. Конкретная реализация обобщенной концепции передается в алгоритм в качестве дополнительного аргумента и задает соответствующую стратегию его работы.

Обобщённые концепции итераторов важны при проектировании библиотечных компонентов обработки и анализа данных, так как абстракции, построенные на базе этих понятий, могут являться интерфейсами между обобщёнными алгоритмами и структурами хранения данных. Они также имеют большое значение, поскольку являются обобщением указателей, т.е. объектов, которые указывают на другие объекты и могут использоваться как основа для создания компонентов, осуществляющих проход по диапазону. Если модель концепции итератора указывает на какую-либо позицию диапазона, то после применения операции продвижения (инкрементирование или декрементирование) она будет указывать позицию, являющуюся следующей или предыдущей, в зависимости от направления обхода. Нужно отметить, что взаимосвязь концепций итераторов со свойствами указателей не совсем однозначна. Например, указатели в языке C++ имеют очень развитую семантику и к ним применимы операции адресной арифметики, операция разыменования и т.п., но обобщённые алгоритмы на диапазонах в большинстве случаев используют лишь малое подмножество свойств указателей, другие же обобщённые алгоритмы используют иные под-

множества. Таким образом, существует несколько различных способов обобщения семантики указателей, при этом каждый из способов является отдельной обобщённой концепцией.

```
[Test()]
public void FindTest()
{
    int[] A = {4, 1, 0, 3, 2, 0, 6};
    var begin = IteratorFactory.Begin(A);
    var end = IteratorFactory.End(A);

    var result = Nonmutating.Find
    (
        begin,
        end,
        FunctionsFactory.Bind2nd
        (
            FunctionsFactory.EqualTo
            (
                ConceptsFactory.BinaryInteger()
            ), 0
        )
    );
    //First index of zero = 2
    Assert.AreEqual(begin.Difference(result), 2);
}
```

Рис. 1. Пример использования обобщенных концепций

Понятие адаптирующей концепции функционального объекта вводится для того, чтобы изменить поведение уже имеющейся обобщённой концепции нечто такого, что может вести себя как функция. Для этого необходимо создать адаптивный интерфейс, который, в определённом смысле, будет изменять поведение адаптирующей модели в терминах адаптируемой концепции. Важным аспектом применения адаптирующей модели является тот факт, что становится возможным сделать что-то вроде того, что было реализовано в качестве адаптируемой концепции, но с конкретным альтернативным поведением, например, изменяющим контекст отдельной операций. В общем случае это гарантирует, что обобщённые алгоритмы будут правильно и эффективно работать с адаптирующими моделями функторов, построенными на базе обобщённых концепций со строгими требованиями к семантике и формализованными интерфейсами.

Обобщенное программирование хорошо зарекомендовало себя при решении проблем повторного использования кода при реализации обобщенных алгоритмов. Обобщенные концепции итераторов обеспечивают интерфейс между структурами хранения данных и алгоритмами, предоставляя нужную гибкость при реализации без потери эффективности. Обобщенные концепции функциональных адаптеров могут быть полезны в различных контекстах. При разработке алгоритмов они, прежде всего, используются как параметры для указания

стратегии работы. Такой системный подход, используемый для построения абстракций и взаимозаменяемых компонентов, при проектировании алгоритма сохраняет семантику его работы и в обобщенном виде.

Литература

1. Devon, M. S. The Programming Paradigm Evolution /M.S. Devon // IEEE Computer – 2012. – № 6. – P. 93–95.

2. Яблоков, Д.Е. Использование обобщённых концепций в объектно-ориентированных языках программирования. МНТК «Перспективные информационные технологии»: Сб. науч. тр. / под ред. С.А. Прохорова. Самара: СГАУ – 2015. – С. 341–345.

3. Гамма, Э. Приемы объектно-ориентированного проектирования / Эрих Гамма – М.: Питер. – 2006. – 368 с.

4. Яблоков, Д.Е. Универсальная модель хранения данных как средство классификации при решении исследовательских задач / Д.Е. Яблоков // Известия СНЦРАН –2016 –18–№4(4) – С. 858–863.

5. Макконнелл, С. Совершенный код / Стив Макконнелл – М.: Питер, 2007. – 893 с.

6. Страуструп, Б. Язык программирования C++. Специальное издание / Б. Страуструп – М.: Издательство Бином – 2011. – 1136 с.

М.С. Якубов, Т.А. Хужакулов, М.М. Хусанов

РОЛЬ ЭКОЛОГИЧЕСКОЙ ОЦЕНКИ ПРИ ПОДГОТОВКЕ И РЕКОНСТРУКЦИИ ПРОЕКТОВ ВОДОХОЗЯЙСТВЕННОГО СЕКТОРА

(Ташкентский университет информационных технологии. Ташкент, Узбекистан)

Описаны основные виды экологической оценки и их роль в охране окружающей среды. It is described in the article the main kinds of ecological estimation and their role in environmental protection.

На сегодняшний день правильная, экологическая оценка способствует более быстрому информированию о той или иной опасности, которая может произойти или уже существует на обследуемой территории. Таким образом, само понятие экологической оценки является на сегодняшний день актуальным. В соответствии с мировыми стандартами и многочисленными конвенциями по охране окружающей среды экологическая оценка имеет общепризнанные во всем мире критерии [1].

Экологическая оценка — это процесс систематического анализа и оценки экологических последствий намечаемой деятельности, консультаций с заинтересованными сторонами, а также учет результатов этого анализа и консультаций в планировании, проектировании, утверждении и осуществлении данной деятельности.