DOI 10.23859/1994-0637-2017-5-80-22
УДК 372.8:004

Касторнов Анатолий Федосеевич
Кандидат педагогических наук, профессор, Череповецкий государственный университет (Череповец, Россия)
E-mail: a_kastornov@mail.ru

Касторнова Василина Анатольевна
Кандидат педагогических наук, доцент, Институт управления образованием Российской академии образования (Москва, Россия)
E-mail: kastornova_vasya@mail.ru

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ КАК ПЕРСПЕКТИВНОЕ НАПРАВЛЕНИЕ РАЗВИТИЯ ШКОЛЬНОГО КУРСА ИНФОРМАТИКИ

Kastornov Anatoly Fedoseevich
PhD in Pedagogical Sciences, Professor, Cherepovets State University (Cherepovets, Russia)
E-mail: a_kastornov@mail.ru

KastornovaVasilina Anatol'evna
PhD in Pedagogical Sciences, Associate Professor, Institute of Education Management of the Russian Academy of Education, (Moscow, Russia)
E-mail: kastornova_vasya@mail.ru

OBJECT-ORIENTED PROGRAMMING AS A PROMISING DIRECTION FOR THE DEVLOPMENT OF SCOOL INFORMATICS COURS

Аннотация. Авторы рассматривают вопросы сравнительного анализа двух современных парадигм программирования, показывая преимущества объектно-ориентированного подхода при разработке программ по сравнению с процедурным подходом, так как он наиболее естественен для восприятия окружающей человека действительности, что имеет преимущества в формировании алгоритмической культуры школьников.

Ключевые слова: парадигмы программирования, объектно-ориентированный подход, языки программирования, объекты и классы, алгоритмическая культура, школьная информатика

Abstract. The author considers the comparative analysis of two modern programming paradigms, shows the advantages of object-oriented approach to programming compared to procedural, as it is most natural to the human perception of reality, which has advantages in the formation of algorithmic culture of schoolchildren.

Keywords: programming paradigm, object-oriented, programming languages, objects and classes, algorithmic culture, school informatics

Введение

Программирование как творческий процесс при решении сложных задач предполагает их разделение на все меньшие и меньшие части (подзадачи). Разделение сложной задачи на подзадачи принято называть ее декомпозицией. Существуют на данный момент две ее формы: алгоритмическая и объектно-ориентированная. При алгоритмической декомпозиции идет вычленение отдельных структур (подзадач, модулей) по технологии «сверху-вниз», а затем из них «собирается» алгоритм (программа) решения всей задачи.

При объектно-ориентированном подходе также идет выделение в ней ее составляющих элементов, но не в виде подзадач, а в форме различных абстракций (объектов, классов) данной предметной области. В этой декомпозиции задача представля-

ется совокупностью автономных действующих лиц, которые взаимодействуют друг с другом и обеспечивают поведение будущего программного продукта на более высоком уровне. При этом каждый элемент этой декомпозиции (объект) обладает своим собственным поведением и моделирует решаемую задачу. Ввиду того, что декомпозиция основана на объектах, ее вполне естественно следует называть объектно-ориентированной декомпозицией.

Однозначного ответа на вопрос какая декомпозиция сложной задачи более эффективна не существует. Каждой из них присущи положительные и отрицательные моменты. Но сконструировать сложную систему одновременно двумя этими способами не представляется возможным, так как они противоречат друг другу, хотя природа каждого объекта зиждется на алгоритмической основе. Следует начать разработку программы либо по алгоритмам, либо по объектам, а затем оценить полученные при этом результаты.

Опыт показывает, что полезнее всего начинать работу с объектной декомпозиции, так как именно она поможет лучше справиться с приданием организованности сложным программным системам. Специалисты в области программирования считают, что объектная декомпозиция имеет ряд преимуществ по сравнению с алгоритмической. Как правило, объектная декомпозиция меньше алгоритмической, так как в ней повторно используются общие механизмы, что приводит к существенной экономии изобразительных средств. Кроме того, объектная декомпозиция помогает лучше разобраться в сложной программной системе, разделяя ее на более простые структуры не в виде подпрограмм (процедур), а в виде объектов и классов.

Основная часть

В 60–70-е годы XX столетия сложность компьютерных программ стала расти и возникла необходимость разработки методов, помогающих справиться с этой проблемой. Наибольшую популярность имеет структурное программирование по методу «сверху вниз». Этот метод используется в традиционных языках программирования высокого уровня типа Fortran, Cobol, Algol, Pascal и др. Здесь основной базовой структурой является подпрограмма (процедура), вот почему эти языки получили название «процедурные». Программа, написанная на процедурном языке, имеет структурный вид: во время своей работы она организует процесс так, что одни подпрограммы вызывают другие подпрограммы, а при рекурсии даже самих себя. Структурное программирование и получило свое название за счет того, что ему соответствует алгоритмическая декомпозиция, при которой происходит разбиение большой задачи на подзадачи [2].

В настоящее время методы программирования принято разделить на две основные группы:

- метод структурного программирования сверху вниз;
- объектно-ориентированное программирование (далее – ООП).

Несмотря на то, что в первом методе речь идет о процедуре, а во втором об объекте, но в каждом из них присутствует алгоритмическая декомпозиция. Однако структурный подход не имеет возможности выделения абстракций и не способен обеспечить ограничение доступа к данным. В основе объектно-ориентированного программирования лежит представление о том, что программный продукт представляет собой совокупность взаимодействующих друг с другом объектов, каждый из которых представляет собой экземпляр определенного класса, а сами классы имеют иерархическую структуру. Такие языки программирования как Smalltalk, Object Pascal, C++, Clos и Ada относятся к объектно-ориентированным языкам.

Основополагающим в ООП является организация иерархий классов и объектов. Эта структура важна тем, что в ней работает принцип взаимодействия объектов друг с другом. Объектно-ориентированное программирование принципиально отличается

от традиционных подходов структурного программирования тем, что здесь поиному происходит процесс декомпозиции [1] .

Стиль программирования, проявляющийся в выборе программистом соответствующего языка, принято называть парадигмой программирования. Стиль программирования определяется способом построения программ, основанным на определенных принципах программирования. Можно выделить три стиля (парадигмы) программирования, которые перечислены в таблице ниже.

Таблица

Стили программирования

Стили программирования	Абстракции
1. Процедурный	Алгоритмы
2. Объектный	Классы и объекты
3. Логический	Предикаты (правила и факты)

Ни один из этих стилей нельзя признать наилучшим. Все зависит от специфики решаемой задачи. Например, при проектировании баз знаний более подходит логический, так как он основан на правилах; а для вычислительных задач более подходит процедурный. Среди всех стилей объектно-ориентированный подход программирования не только применим для широкого круга задач, но и является по своей сути тем фундаментом, на котором основываются другие парадигмы.

Для каждой методологии программирования характерны свои особенности, которые проявляются в большей степени в способе восприятия решаемой задачи. В основе объектно-ориентированной программы лежит понятие объектной модели, имеющей четыре главных конструктивных элемента: абстрагирование, инкапсуляция, модульность, иерархия. Без них разрабатываемая программа не будет являться объектно-ориентированной.

В ООП любая программа представляет собой совокупность объектов, каждый из которых является экземпляром определенного класса, а классы обладают свойством наследования (иерархия). Из этого следует, что:

1) в ООП базовые элементы есть объекты, а не алгоритмы;

2) каждый объект представляет собой экземпляр какого-либо определенного класса;

3) классы образуют иерархическую структуру.

При соблюдении всех трех указанных требований компьютерную программу можно считать объектно-ориентированной. Нарушение принципа иерархичности приводит к тому, что такое программирование нельзя отнести к ООП, это, скорее всего, программирование на основе абстрактных типов данных.

Следует обратить внимание на то, что не все языки программирования являются объектно-ориентированными в строгом смысле этого слова. Так, например, на языке Pascal можно, используя тип Record, создавать объекты и классы, но от этого данный язык не становится объектно-ориентированным, здесь наблюдаются только «отголоски» принципа объектности. Принято считать, что язык программирования является объектно-ориентированным только тогда, когда выполняются определенные условия.

Основным признаком этого отличия является понятие объекта как некоторой абстракции данных, а также наличие в программе интерфейса в виде именованных операций и собственных данных с ограничением доступа к ним. При этом:

- объекты принадлежат соответствующим типам (классам)
- классы наследуют свойства супер типов (суперклассов).

С учетом этих требований языки Smalltalk, Object Pascal, C++ и Clos можно отнести к группе объектно-ориентированных языков, а вот Ada является объектным языком в строгом смысле этого слова. Объекты и классы фигурируют в обеих группах языков, поэтому все они используют объектно-ориентированный стиль программирования.

Значение абстрактных типов данных в разрешении проблемы сложности систем сыграло определенную положительную роль, но не решило всех проблем. Процедуры также поддерживают технологию абстрагирования, но они способны описывать только абстрактные действия и не годятся для описания абстрактных объектов. А при решении сложной задачи успех во многом зависит от возможности оперирования сложными объектами. В процедурных языках эта проблема решается двумя способами. Во-первых, разрабатываются методы так называемых потоков данных, вносящих упорядоченность в работе с данными. Во-вторых, осуществляется типизация данных, которая нашла свое отражение, например, в языке Pascal.

Переход на объектно-ориентированное программирование начался с языка Simula и получил развитие в последующих языках программирования высокого уровня Smalltalk, Object Pascal, C++, Clos, Ada и Eiffel. Эти языки принято называть объектными или объектно-ориентированными. В этих языках структура программ имеет форму графа, а не дерева, присущего алгоритмическим языкам.

Понятие «объект» возникло более 30 лет назад при разработке компьютеров, преследующих цель отхода от фон-неймановской архитектуры, чтобы устранить барьер между высоким уровнем программного продукта и низким уровнем ЭВМ. Введение понятия объекта и соответствующего ему стиля программирования создает благоприятные возможности для более качественных средств, обеспечивающих лучшее выявление ошибок, большую эффективность написания программ, сокращение программного кода, улучшение компиляции, экономию памяти.

Объектный подход присущ как объектным, так и объектно-ориентированным языкам программирования. Этот стиль программирования начался с языков Simula и Smalltalk, а затем их идеи были внедрены в другие языки высокого уровня, которые получили название объектно-ориентированных языков. Примером этого явления стали модификации языка C в виде C++ и Objective C. Языки Object Pascal, Eiffel и Ada возникли из языка Pascal. На основе языка Дшыз были созданы Flavors, Loops и Clos (Common LISP Object System), которые уже обладают возможностями языков Simula и Smalltalk.

Категория объекта использовалась человеком с давних времен. Еще античные греки высказали идею о том, что мир можно трактовать в понятиях как объектов, так и событий. А в XVII веке Р. Декарт отмечал, что человечество рассматривает мир с объектно-ориентированной точки зрения. Для человека объектом является:

- осязаемый и (или) видимый предмет;
- нечто, воспринимаемое мышлением;
- нечто, на что направлена мысль или действие.

Объект представляет собой модель окружающей действительности и, значит, существует во времени и пространстве. Термин «объект» в программировании стали использовать впервые в языке Simulate, где он использовался для моделирования реальности.

Объектом можно назвать конкретно опознаваемый предмет, явление или сущность (реальную или абстрактную), которая проявляет себя в некоторой предметной области своим функциональным назначением. Можно определить объект как некую субстанцию, имеющую четко очерченные границы. В ООП понятия класса и объекта неразрывно связаны друг с другом, так что невозможно говорить об объекте без его привязки к классу. Однако они существенно отличаются по своей сути. Если объект обозначает конкретную сущность, определенную во времени и пространстве, то

класс определяет лишь абстракцию существенного в объекте. Другими словами, классом является набор объектов, обладающих общей структурой и одинаковым поведением.

Итак, каждый конкретный объект является экземпляром, то есть элементом класса. Следует отметить, что, проводя аналогию с математикой, классы, используемые в языках программирования, необходимы, но не достаточны для декомпозиции сложных задач. По своей сути класс осуществляет связь между абстракцией и всеми ее «подопечными» (экземплярами класса, то есть объектами). Класс взаимодействует со своими «клиентами» с помощью соответствующего интерфейса.

Выводы

Программирование как вид человеческой деятельности, возникшей в связи с появлением ЭВМ в середине XX века, нашло свое отражение и в школьном курсе информатики и вычислительной техники. Первоначально именно оно составляло содержание этой дисциплины. Сначала это было программирование в содержательных обозначениях, а потом был осуществлен переход на процедурные языки высокого уровня, ориентированный на алгоритмическую парадигму программирования. Однако, исходя из вышеизложенного, следует заключить, что объектно-ориентированная парадигма программирования наиболее естественна для восприятия окружающей человека действительности, поэтому ее использование в курсе школьной информатики и ИКТ может дать положительный эффект в деле формирования алгоритмической культуры учащихся, что, по мысли основоположника этого предмета в школе А.П. Ершова, является основной составляющей дисциплины «Информатика» [3].

Литература

1. Гради Буч и др. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008. 720 с.

2. Касторнова В.А. Структуры данных и алгоритмы их обработки на языке программирования Паскаль. СПб.: БХВ-Петербург, 2016. 304 с.

3. Касторнов А.Ф., Касторнова В.А. Языки программирования и их роль в становлении предметной области «Информатика» // Педагогическая информатика. 2016. № 1. С. 59–68.

References

1. Gradi Buch. *Ob"ektno-orientirovannyi analiz i proektirovanie s primerami prilozhenii* [Object-Oriented Analysis and Design with Application]. Moscow, 2008. 720 p.

2. Kastornova V.A. *Struktury dannyh i algoritmy ih obrabotki na iazyke programmirovaniia Paskal'* [Data structures and algorithms for their processing in the Pascal programming language]. St. Petersburg, 2016. 304 p.

3. Kastornov A.F., Kastornova V.A. Iazyki programmirovaniia i ih rol' v stanovlenii predmetnoi oblasti «Informatika» [Programming languages and their role in the development of the subject area «Informatics»]. *Pedagogicheskaia informatika* [Pedagogical Informatics], 2016, no. 1, pp. 59–68.

Касторнов А.Ф., Касторнова В.А. Объектно-ориентированное программирование как перспективное направление развития школьного курса информатики // Вестник Череповецкого государственного университета. 2017. №5(80). С. 177–181.

For citation: Kastornov A.F., Kastornova V.A. Object-oriented programming as a promising direction for the devlopment of scool informatics cours. *Bulletin of the Cherepovets State University*, 2017, no. 5 (80), pp. 177–181.