

**С. А. Бешенков,**

*Институт управления образованием Российской академии образования, г. Москва,*

**И. В. Акимова,**

*Пензенский государственный университет*

## ОСНОВЫ ЗАДАЧНОГО ПОДХОДА К ИЗУЧЕНИЮ ПРОГРАММИРОВАНИЯ

### *Аннотация*

В статье рассматривается актуальная для данного этапа развития методической науки проблема обучения программированию. Авторами показывается, что обучение программированию является неотъемлемым компонентом подготовки будущего учителя информатики. В качестве инструмента подготовки выбран так называемый «задачный» подход, суть которого заключается в опоре на классы задач, которые предлагаются для решения.

**Ключевые слова:** программирование, методика, задача.

Обучение программированию является одним из основных компонентов подготовки будущего учителя информатики. При этом возникают два естественных вопроса: чему учить и как учить. Ответ на второй вопрос, естественно, вытекает из ответа на первый. Поэтому основное внимание целесообразно уделить проблеме того, чему надо учить студентов — будущих учителей информатики в рамках традиционной для педагогических вузов дисциплины «Алгоритмизация и программирование».

Попробуем разобраться в этом вопросе.

Заметим сразу, что в последнее время стали появляться голоса, призывающие к увеличению в процессе преподавания информатики доли информационно-технологий за счет алгоритмизации и программирования. Мы не разделяем этого мнения. Сокращение времени на изучение программирования как в школе, так и в вузе, несомненно, обедняет информатику, ограничивает возможности в формировании особого типа мышления, практических навыков, которые находят свое применение и во многих других областях знаний.

Как нам представляется, программирование будет составлять обязательный компонент инфор-

матики при любом подходе к ее изучению. Однако цели изучения информатики могут меняться при переходе от одного этапа к другому.

На первом этапе (1985 год) программирование выступало как способ реализации алгоритмов, инструмент анализа и исследования программ. Необходимо было выбрать подходящий для этого язык. Как известно, основными языками были алгоритмический язык А. П. Ершова (реализованный в «Е-практикуме») и Basic, выбранный в качестве «реального» языка программирования.

По мере развития курса информатики точка зрения на цели, задачи и содержание раздела, связанного с программированием, в определенной мере менялась, менялись и методические подходы к его изучению.

Например, в работах Е. А. Ракитиной (см., в частности, [10]) предлагается изучение программирования вписать в контекст изучения моделирования. В работах Л. А. Кугеля предлагается делать акцент не на конкретном языке программирования, а на задачах, которые необходимо решить с его использованием [5].

Рассмотренный в данной статье подход развивает и углубляет названные выше (и другие) работы.

### **Контактная информация**

**Бешенков Сергей Александрович**, доктор пед. наук, профессор, гл. научный сотрудник Института управления образованием Российской академии образования, г. Москва; *адрес:* 105062, г. Москва, ул. Макаренко, д. 5/16, стр. 1Б; *телефон:* (495) 625-20-24; *e-mail:* srg57@mail.ru

**Акимова Ирина Викторовна**, канд. пед. наук, доцент, доцент кафедры «Алгебра и методика обучения математике и информатике» Педагогического института им. В. Г. Белинского Пензенского государственного университета; *адрес:* 440026, г. Пенза, ул. Красная, д. 40; *телефон:* (841-2) 54-88-13; *e-mail:* ulrih@list.ru

**S. A. Beshenkov,**

*Institute of Education Management of the Russian Academy of Education, Moscow,*

**I. V. Akimova,**

*Penza State University*

### **BASES OF TASK APPROACH TO PROGRAMMING STUDYING**

#### **Abstract**

The article is devoted to the problem of training in programming, urgent for this stage of development of methodical science. The authors show that training in programming is the integral component of training of future informatics teacher. As the instrument of training so-called "task" approach which essence consists in a support on classes of tasks which are offered for the decision is chosen.

**Keywords:** programming, methodics, task.

Основная мысль состоит в том, что язык программирования выступает лишь инструментом решения задачи. Действительно, задачи человек решает на протяжении всей своей жизни и деятельности, поэтому возникает иная постановка проблемы изучения программирования: на первое место выдвигать изучение не самого языка и его синтаксиса, а структуры, некоторых парадигм, принципов построения программ, ориентированных на задачи. Такого подхода предлагается придерживаться и в школе, и в вузе.

Безусловно, можно встретить ряд работ, открывающих данный подход (В. Е. Жужалов [2]), но они выполнены достаточно давно и требуют своего развития.

Исходя из всего вышесказанного, считаем, что первоочередным вопросом становится классификация задач, которые будут выступать базой для изучения языков программирования.

Анализируя основные образовательные программы педагогических специальностей вузов, мы можем отметить тенденцию изучения студентами не конкретных языков программирования, а, скорее, ведущих парадигм программирования. Это, на наш взгляд, объясняется тем, что в настоящее время наблюдается наличие большого количества различных языков, при этом продолжают появляться все новые или модернизируются уже существующие. В результате задача изучения конкретного языка может стать в итоге узконаправленной и не способствовать дальнейшему развитию и профессиональному росту выпускника. Но при этом изучение конкретной парадигмы может стать источником дальнейшего развития, самообразования.

Сам выбор парадигмы опирается в основном на класс задач, которые решает программист. Поэтому встает проблема классификации решаемых задач, выявления их свойств — признаков классификации,

которые затем позволили бы выбрать наиболее эффективную парадигму их решения.

*Предлагается рассматривать пространство задач с помощью введения своеобразной системы координат, в точках которой возникают классы задач, решаемых с помощью определенных парадигм* (рис. 1).

На наш взгляд, целесообразно выделить следующие оси этой системы координат.

**1. Местонахождение объекта задачи: пространство или время.**

Специфика реализации решения задачи зависит от особенностей проблемной области, от характера имеющихся знаний. Задачи, относящиеся к *пространственному* типу, нацелены на представление, описание данных; задачи, относящиеся к *временному* типу, нацелены на обработку данных.

В работе А. П. Ершова мы находим: «Во всяком алгоритмическом языке можно выделить две самостоятельные части. Первая из них предназначена для описания объектов перерабатываемой информации (исходных, промежуточных, окончательных результатов). Вторая из этих частей представляет собой набор средств для описания процессов переработки этих данных» [3]. Таким образом, намечается классификация использования языков программирования именно от решаемых задач. Далее автор отмечает: «Языки, ориентированные на обработку данных, характеризуются более развитым аппаратом, предназначенным для описания объектов перерабатываемой информации. В задачах, связанных с обработкой данных, последние, как правило, представляют собой совокупность объектов сложной структуры. Что касается средств, предназначенных в языке для описания процессов переработки данных — операторов языка, то выбор этих средств в значительной мере определяется ориентацией языка на класс

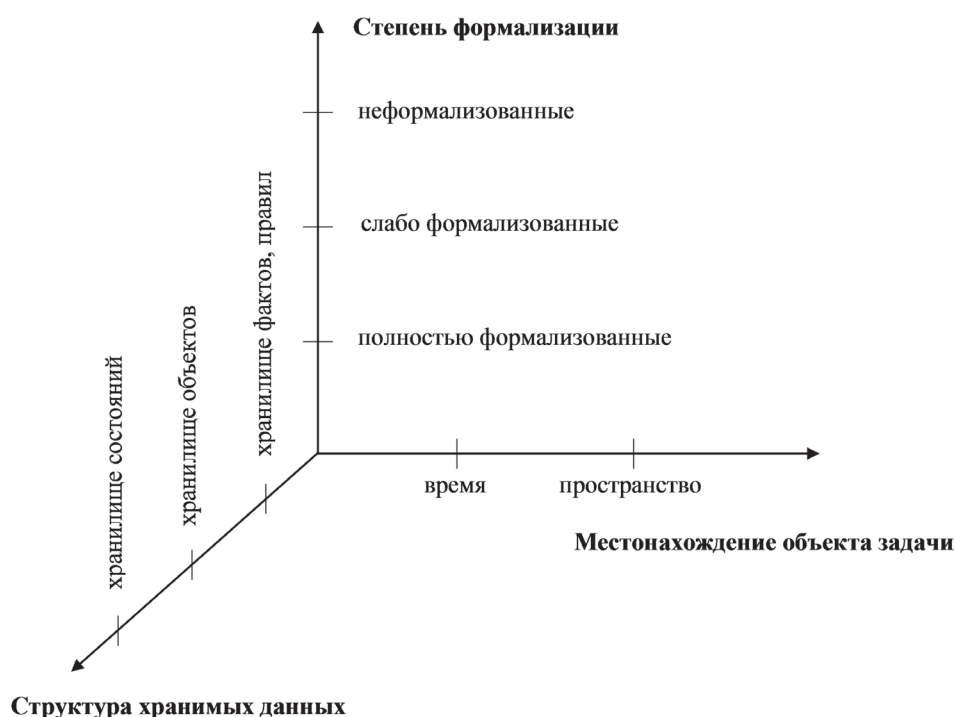


Рис. 1. «Задачная» система координат

задач и формой задания данных» [там же]. То есть подтверждается направление разделения языков и парадигм в соответствии с классами решаемых задач: задачи (и, соответственно, языки), ориентированные на представление, описание данных, и задачи, ориентированные на обработку, манипулирование данными. В заключение А. П. Ершов делает вывод: «В связи со сферой использования машин первого поколения в соответствующих алгоритмических языках получили широкое развитие средства описания собственно вычислительных процессов. По мере расширения области применения ЭВМ возникла необходимость в существенном расширении указанного аппарата в направлении развития средств, предназначенных для обработки объектов сложной структуры. Специфика названных задач и вызвала работы по созданию специальных языков для обработки данных. Одним из развитых в этом отношении и получившим признание среди данного класса проблемно ориентированных языков является Кобол» [там же]. Это подтверждает наше предположение о направленности классификации парадигм от задач, что различные языки программирования, несущие ту или иную парадигму, предназначены для определенного класса задач.

Позднее в своих трудах, относящихся к 1985 году, академик А. П. Ершов уже четко определял, что «при огромном разнообразии языков программирования они делятся на два класса: декларативные (аппликативные, функциональные) и императивные (процедурные, алгоритмические). Поговорим о различии между этими классами. В двух словах оно состоит в следующем: декларативная программа заявляет, *что* должно быть достигнуто в качестве цели, а императивная программа предписывает, *как* достичь поставленной цели» [4].

Такое деление парадигм, по словам А. П. Ершова, носит исторический характер. «Во второй половине 50-х годов появились достаточно полные императивные языки программирования, поддерживаемые производственными системами программирования. <...> В то же время по мере расширения области применения ЭВМ появилось много задач, как тогда говорили, невычислительного характера, для которых серьезную трудность представила исходная проблема строгого формулирования задачи. Язык математического анализа и алгебраических формул для этих задач недостаточен. Особенностью таких задач был их логический характер, а также необходимость работать со сложно организованной информацией» [4].

Далее в этой же работе А. П. Ершов продолжает: «Заклячая сравнение декларативных и императивных языков, мы видим, что свойство декларативности облегчает формулирование задачи, подлежащей решению на ЭВМ, а свойство императивности приближает запись алгоритма к тому, как он фактически исполняется на ЭВМ» [4].

Б. Хигман в своей работе «Сравнительное изучение языков программирования» также анализирует вариант классификации языков программирования, исходя из решения возникающих перед ним задач: «Другой причиной создания различных типов языков является различный подход к решению возникающих

задач» [11]. Автор отмечает, что на пути развития языков программирования «одни пытались идти по пути создания таких же методов описания, какие нам известны по естественным языкам, включая язык математики. Их усилия были направлены на создание семантики, полностью независимой от машины. Другие уделяли больше внимания общности описаний в рамках конкретной машины или группы машин. Некоторые взяли за основу тот безусловный факт, что вычисления, собственно, не являются типичной повседневной деятельностью и что, следовательно, лучшая грамматика языка вычислений, по-видимому, должна отличаться от грамматики естественных языков. Здесь следует остановиться на двух группах языков: на группе процедурных языков, в которых явно признается последовательный характер большинства вычислительных процессов, и группе функциональных языков, признающих только один императив “вычислить значение”, который может быть задан и неявно...» [там же].

Дж. Маккарти также неоднократно высказывал идею использования различных языков программирования для решения определенных задач [13].

Дж. Малпас также придерживается мнения, что «целью написания программы для ЭВМ является либо решение определенной задачи, либо представление системы, существующей в реальном мире, таким образом, чтобы программа могла использоваться для моделирования поведения этой системы. Целесообразно оценивать взгляды на мир, навязываемые различными языками программирования, в соответствии с тем, насколько хорошо они соответствуют указанным выше целям. <...> Нужно определить, какие языки и программные средства наилучшим образом ей удовлетворяют» [7].

Если провести анализ любой информационной системы, то выяснится, что общепринятым является разделение двух функций: определение данных и манипулирование данными. Конечно, в современных СУБД предполагается объединение этих двух функций в одном языке программирования, но такое историческое разделение дает нам основание обратиться к разделению функций задач.

Для задач, ориентированных на представление данных, классификацию можно продолжить следующим образом:

- логические модели;
- сетевые модели;
- продукционные модели;
- сценарии.

Для задач, ориентированных на обработку данных, классификация может быть продолжена следующим образом:

- описание отдельных действий и условий;
- описание функций и математических зависимостей;
- описание действий в виде правил: условие — изменение данных;
- описание действий в виде: шаблон — изменение данных.

## 2. Степень формализации задачи:

- полностью формализованная задача;
- слабо формализованная задача;
- неформализованная задача.

Степень формализации — это степень математического описания задачи, от которой во многом зависит эффективность работы всей системы, а также уровень автоматизации, определяемый степенью участия человека при принятии решения на основе получаемой информации. Чем точнее математическое описание задачи, тем выше возможности компьютерной обработки данных и тем меньше степень участия человека в процессе решения. Это и определяет степень автоматизации задачи.

Различают три типа задач, для которых создаются информационные системы:

- структурированные (формализуемые);
- неструктурированные (неформализуемые);
- частично структурированные.

*Структурированная (формализуемая) задача* — это задача, где известны все ее элементы и взаимосвязи между ними.

*Неструктурированная (неформализуемая) задача* — это задача, в которой невозможно выделить элементы и установить связи между ними.

В структурированной задаче удается выразить ее содержание в форме математической модели, имеющей точный алгоритм решения. Подобные задачи обычно приходится решать многократно, и они носят рутинный характер. Целью использования информационной системы для решения структурированных задач является полная автоматизация их решения, т. е. сведение роли человека к нулю.

Решение неструктурированных задач из-за невозможности создания математического описания и разработки алгоритма связано с большими трудностями. Возможность использования здесь информационной системы невелика. Решение в таких случаях принимается человеком из эвристических соображений на основе своего опыта и, возможно, косвенной информации из разных источников.

О большинстве задач можно сказать, что известна лишь часть их элементов и связей между ними. Такие задачи называются *частично структурированными*. В этих условиях можно создать информационную систему [6].

### 3. Структура хранимых данных.

Приведенная ниже классификация представлена в работе Н. Н. Непейвода и И. Н. Скопина [8]. В ней авторы подчеркивают, что система данных, обрабатываемая в задаче, имеет определяющее значение. Это связано с тем, что различные структуры, определяющие взгляд на данные в задаче, требуют различных языковых средств, которые должны быть представлены в соответствующем языке программирования. Речь идет и о самих структурах, и о средствах обработки.

В результате можно предложить классы задач, нацеленные на хранение следующих структур данных:

- хранилище фактов, правил;
- хранилище объектов (переменные, структуры данных, объекты);
- хранилище состояний.

Дж. Малпас определяет факт как утверждение о том, что соблюдается некоторое конкретное отношение. Факт записывается как имя, за которым следует список термов (они называются аргументами), заключенный в скобки. Правило — это факт,

истинностное значение которого зависит от значений истинности других фактов [7]. То есть факты и правила используются для фиксирования отношений между объектами при условии, что факт «представляет собой безусловно истинное утверждение» [12].

Правила содержат утверждения, истинность которых зависит от некоторых условий [1]. То есть правило — это заключение, для которого известно, что оно истинно, если одно или несколько других найденных заключений или фактов являются истинными.

Переменная — это именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменять [12]. Также можно определить переменную как подкласс идентификаторов, которым сопоставлено многократно используемое значение, ранее вычисленное в подходящем контексте. Подразумевается, что одна и та же переменная в разных контекстах может иметь разные значения.

Дж. Маккарти определял переменную как символ, который используется для представления аргумента в функции [13].

Возможна различная классификация переменных: статические и динамические, локальные и глобальные, простые и сложные (структурированные).

Структурные объекты (или просто структуры) — это объекты, которые состоят из нескольких компонент. Эти компоненты, в свою очередь, могут быть структурами. Использование структур обеспечивает конструктивность построений, гарантирует доступ к частям, из которых выстроено данное любой сложности. Структура может содержать компоненты разных типов.

Класс (объект) является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций для работы с ними [9]. Объекты объединяют внутри себя данные и действия (методы).

Хранилище состояний — одна из самых старых структур в программировании. Оно соответствует теоретическому понятию конечного автомата. На этот стиль программирования наталкивает само устройство существующих вычислительных машин, которые представляют собой гигантские конечные автоматы.

В значительной степени ориентированы на такой стиль машинные языки и, соответственно, языки ассемблера. Имеется много систем, в которых такие программы автоматически или полуавтоматически генерируются по графовому или табличному представлению, более органичному для такого стиля, например, в системе UML.

Суть программирования от состояний можно охарактеризовать следующим образом.

Определяются:

- множество так называемых *состояний*, которые может принимать конечный автомат;
- *переходы между состояниями*, которые осуществляются под внешним воздействием (например, под воздействием перерабатываемых данных).

Программа, написанная в таком стиле, является перечнем команд, фиксирующих переходы между состояниями. Если говорить в более «теоретических» терминах, то для каждой возможной пары <состояние, внешнее воздействие> указывается очередное состояние. Описание такой программы может быть осуществлено разными способами. Многие из них хорошо изучены теоретически и поддерживаются развитыми методиками программирования.

Современные методики программирования от состояний базируются на таблицах состояний, подобных таблице состояний конечного автомата. Эти таблицы часто также представлены в виде графов, что особенно удобно, когда не все возможные переходы между состояниями реализуемы [8].

Информационное пространство всех блоков и процедур при программировании от состояний в первом приближении одно и то же: состояния системы, моделируемой совокупностью программных действий. Но на самом деле многие блоки либо процедуры работают с подсистемами. Подсистемы, ввиду их автономности, могут иметь характеристики, прямо недоступные для общей системы, и в свою очередь могут иметь лишь ограниченный доступ к общему системному пространству данных. Более того, подсистемы могут общаться прямо, в обход иерархически вышестоящей системы.

Таким образом, структура информационного пространства при программировании от состояний в общих чертах соответствует той, которая навязывается современными системами с развитой модульностью.

Итак, в заданной системе координат каждая задача будет находиться на пересечении соответствующих координат и относиться к определенному типу, который будет задаваться тремя характеристиками (см. рис. 1). А для эффективного решения задач

каждого такого выделенного типа будет подходить определенная парадигма.

#### Список использованных источников

1. Братко И. Алгоритмы искусственного интеллекта на языке Prolog. 3-е изд. / пер. с англ. М.: Вильямс, 2001.
2. Жужжалов В. Е. Совершенствование содержания обучения программированию на основе интеграции парадигм программирования: дис. ... канд. пед. наук: 13.00.02. М., 2004.
3. Ершов А. П. Алгоритмические языки и программирование // История отечественной математики. Т. 4. Кн. 2. Киев, 1970.
4. Ершов А. П. Текст «Как учить программированию?». Подготовлен для выступления на семинаре журнала «Микропроцессорные средства и системы» // Архив академика А. П. Ершова. <http://ershov-arc.iis.nsk.su/archive/eaindex.asp?lang=1&did=40912>
5. Кугель Л. А. Обучение студентов алгоритмизации и программированию на основе структурно-алгоритмического подхода к постановке и реализации задач (на примере направления подготовки бакалавров «Прикладная информатика»): дис. ... канд. пед. наук: 13.00.02. М., 2014.
6. Макарова Н. В., Волков Б. В. Информатика: учебник для вузов. СПб.: Питер, 2011.
7. Маллас Дж. Реляционный язык Пролог и его применение / пер. с англ., под ред. В. Н. Соболева. М.: Наука. Гл. ред. физ.-мат. лит., 1990.
8. Непейвода Н. Н., Скопин И. Н. Основания программирования. Ижевск, М.: РХД, 2003.
9. Павловская Т. А. C/C++. Программирование на языке высокого уровня. СПб.: Питер, 2003.
10. Ракитина Е. А. Обучение программированию: моделирование и формализация // Информатика и образование. 2001. № 1.
11. Хигман Б. Сравнительное изучение языков программирования. М.: Мир, 1974.
12. Шрайнер П. А. Основы программирования на языке Пролог. М.: ИНТУИТ.ру, 2005.
13. McCarthy Jh. et al. LISP 1.5 Programmers Manual. MIT Press, 1962.

## НОВОСТИ

### Россияне смогут совершить виртуальные туры по самым лучшим городам страны

Общественная палата Российской Федерации и портал «Культура.РФ» разработали путеводитель по самым лучшим городам нашей страны: Москве, Санкт-Петербургу, Якутску, Волгограду, Чебоксарам, Новочеркаску, Ульяновску, Омску, Вологде и Екатеринбургу, сообщает пресс-служба Министерства культуры России.

Города победители были определены по результатам всероссийской акции #МарафонДобрыхДел.

«В первую очередь наша цель — еще раз привлечь внимание людей к российским городам. В виртуальных турах мы постарались показать большинство самых интересных достопримечательностей этих мест», — отметила инициатор акции #МарафонДобрыхДел, член ОП РФ Елена Сутормина.

Адрес виртуального путеводителя: [http://www.culture.ru/s/dobrye\\_goroda/](http://www.culture.ru/s/dobrye_goroda/)

(По материалам федерального портала «Российское образование»)

### В России усовершенствовали систему управления роботами на расстоянии

Группа российских специалистов из Владивостока и Москвы провела испытания усовершенствованной системы управления роботизированными механизмами на больших расстояниях. Система тестировалась на территории Дальневосточного федерального университета во Владивостоке, сообщает «Интерфакс». Специалисты предложили управлять роботами не с помощью ко-

манд в режиме реального времени, а с использованием мини-программ, которые аппарат с искусственным интеллектом самостоятельно реализует в автономном режиме. Подобная технология позволит решить проблему задержки сигналов при их передаче на большие расстояния и обеспечить более стабильную работу робототехники.

(По материалам международного компьютерного еженедельника «Computerworld Россия»)