УДК 378.147, 316.325

Е. В. Маликов

ЯЗЫКИ ПРОГРАММИРОВАНИЯ И УПРАВЛЕНЧЕСКИЕ НАВЫКИ

Поднимаются вопросы об общих управленческих задачах и целях изучения языков программирования при подготовке менеджеров-управленцев самого широкого профиля и разных степеней ответственности. Предлагается в процессе такой подготовки различать три уровня управленческих компетенций и три подхода к изучению языков программирования, а само понятие «программа» рассматривать в трех аспектах: программа как абстрактный конечный автомат; программа как синтаксический конструкт; программа как система – модель мира. В рамках заявленной задачи для обучения предлагаются средства программирования от примитивных до высокоуровневых; выстраивается последовательность освоения основ программирования; проводится сравнение ряда языков высокого уровня с точки зрения поставленной задачи; анализируется имеющийся опыт внедрения данных идей в учебный процесс.

Ключевые слова: *менеджмент, языки программирования, социальные компетенции.*

Невзирая на то что информационные технологии проникли во все сферы обыденной жизни, вопрос «Нужно ли учить менеджера программированию?» далек от окончательного решения, причем преобладает точка зрения, согласно которой умение писать программы хотя бы на некоторых из существующих языков программирования считается для менеджера избыточным.

С данной точкой зрения никак нельзя согласиться: как языки ассемблера, так и языки высокого уровня нужны не одним узким специалистам, поскольку способствуют формированию управленческих навыков, если задачу такового формирования решать серьезно. Конечно, мы отдаем себе отчет во второстепенности преподавания языков высокого уровня (ЯВУ) будущим менеджерам, но, если в программе есть дисциплины, связанные с программированием, почему не извлечь из их изучения максимальный эффект?

Строго говоря, задача преподавания информационно-коммуникационных технологий неспециалистам не имеет общего решения – здесь сколько преподавателей, столько и подходов, что, на взгляд авторов, неплохо, поскольку способствует поддержанию знаменитой «цветущей сложности» здорового организма [1]. При этом наш личный подход может быть обозначен как «общегуманитарный» [2], в его рамках и будет рассмотрено преподавание программирования для неспециалистов, избегая принципиально слова «алгоритмизация», поскольку для управленцев более привычными будут являться такие термины, как «приказ», «команда», «система». Покажем, что в этих терминах программирование излагается ничуть не хуже, чем в терминах «алгоритм», «оператор» и им подобных.

Заметим, что управленцем является каждый из нас: даже рядовой, выполняя приказ, согласно уставу должен проявлять разумную инициативу. То есть уже на уровне низшего управленческого звена возникает деление на «приказ» и «команду», что находит почти буквальный аналог в программировании, когда мы начинаем рассматривать «задание» и «программу», «команду» и их различие. Еще больше дает человеку знание программирования на более высоких ступенях управленческих иерархий. Но, поскольку на каждом уровне предусмотрены свои компетенции, предлагаем ввести в качестве рабочего деление управления на три категории:

1) управление собой;

2) управление людьми;

3) управление процессами, а подход к программированию сосредоточить на понятии «программа», рассматриваемом в трех аспектах:

1) программа как абстрактный конечный автомат;

2) программа как синтаксический конструкт;

3) программа как система – модель мира.

Итак, с чего начнем?

Сегодня принято начинать с ответов на те вызовы, которые ставит перед нами время и на которые мы вынуждены отвечать здесь и сейчас. Такой подход имеет право на существование, но он лишен стратегической перспективы.

Беда в том, что мы учим ответам на вызовы нынешнего дня и почти ничего не можем сказать о будущем. Мы не только не знаем ответов на завтрашние вопросы, мы не знаем самих вопросов. Мы догадываемся об одном: нужно уметь находить ответы на универсальные вызовы и иметь в виду, что управление и управленцы нужны будут всегда.

Некоторые преподаватели Евразийского открытого института (г. Москва) столкнулись с «тотальностью» охвата населения программированием относительно недавно, когда при вузе был основан колледж, который хотя и рассматривается как кадровый резерв нашего аудиторного будущего, вовсе не является учебным заведением для особо одаренных детей. Нет, его контингент обычен, и именно эта его обыкновенность делает работу с ним репрезентативной.

Что входит в общие компетенции выпускника по программе СПО, например, по направлению «Прикладная информатика» (по отраслям)? Это выборочно:

– понимание сущности и социальной значимости своей будущей профессии;

– организация собственной деятельности и умение выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество;

– способность принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность;

– умение осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач;

– умение работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями;

– умение брать на себя ответственность за работу членов команды (подчиненных), результат выполнения заданий.

При этом если мы имеем дело, к примеру, с техником-программистом, то он должен обладать следующими профессиональными компетенциями (выборочно):

– обеспечивать содержание проектных операций;

– определять сроки и стоимость проектных операций;

– определять качество проектных операций;

– определять ресурсы проектных операций;

– определять риски проектных операций, которые не являются, строго говоря, независимыми, а всего лишь подкрепляют перечисленные выше общие компетенции.

Как мы можем видеть, и общие, и профессиональные компетенции включают управленческие навыки, которые должны быть привиты обучающемуся путем любого тренинга, включая написание программ. И если о работе в коллективе и ответственности за подчиненных на данном этапе в рамках изучения программирования говорить сложно, то все остальное может решаться с помощью недорогих технических средств обучения и достаточно скромных интеллектуальных усилий.

Вспомним аналогию с армией. Получая приказ (задание), его исполнитель самостоятельно детализирует задачу в терминах команд – элементарных действий. В нашем случае приказом становится задание написать программу, а инструментом для его исполнения – какой-то (достаточно) элементарный язык программирования. Чуть ли не общим местом является предлагать в качестве такого языка «Бейсик», что, по мнению авторов, является ошибкой (как и весь «Бейсик» в целом): здесь уместнее всего воспользоваться языком ассемблера. Почему плох «Бейсик» против ассемблера?

В то время как язык ассемблера представляет полный набор инструментов для работы с конкретным устройством, язык «Бейсик», типологически являясь алгоритмическим языком, похож на случайную выборку из инструментария, применяющегося в некоторой достаточно широкой предметной области. Наличие всего двух типов данных (максимально расширенных таксонов «числа» и «строки»), необязательность деклараций, включение механизма умолчаний не добавляют «Бейсику» привлекательности с точки зрения дисциплины программирования. Объяснить это новичку нет возможности, а значит, в представлении начинающего с языком высокого уровня свяжется то, что явно не является удачей в данной сфере, и у такого учащегося сформируется неправильное представление о программировании вообще, а вместе с тем, как мы увидим ниже, и о мире, в котором он живет.

Язык ассемблера полностью определяет мир (архитектура + система команд), к которому он применим. На этапе первоначального знакомства с программированием с его помощью хорошо описывать программу как конечный абстрактный автомат – парадоксально, но именно эта часть программирования является наиболее понятной для начинающих. Лучше всего для этой цели подходят программируемые калькуляторы типа «Электроника МК-52» с обратной польской записью команд. Программирование на МК-52 хорошо тем, что «разводит по углам» выражение и его порождение в виде последовательности команд стандартного формата «регистр – регистр», «регистр – память» и «память – регистр». Указанные калькуляторы дешевы по сравнению с ПК и довольно просты в обращении. При этом «Электроника МК-52» имеет ряд видимых преимуществ перед ПК: на калькуляторе обучающийся, во-первых, сразу вводит программу «с пульта», а во-вторых, довольно скоро упирается в ограниченность памяти и системы команд калькулятора, что заставляет программиста разрешать задачу в разном наборе и разной последовательности элементарных команд. Здесь и проявляется инициатива, о которой говорят уставы и которая заявлена большинством компетенций выпускника.

Автору данного текста довелось более 10 лет обучать программированию на калькуляторе студентов-физиков и студентов-математиков Томского государственного педагогического университета (ТГПУ), а также два последних года – студентов-менеджеров и учащихся колледжа при Евразийском открытом институте (ЕАОИ). В результате он может с уверенностью сказать, что ни рисование

двоичных деревьев, ни написание программы на бланке с распределением памяти по стеку и последующее нажимание на кнопки не вызывает никаких затруднений даже у новичков, вовсе не ориентированных на программирование. Даже отсутствие реального «железа» (а последние два года нами использовался эмулятор) не уменьшает интереса к программированию на калькуляторе. Как учащиеся колледжа, так и студенты-первокурсники воспринимают такое программирование едва не в качестве компьютерной игры, чуть ли не «стрелялки», где каждое нажатие на кнопку влечет видимый результат. Данным «аттрактором» не стоит пренебрегать, поскольку «тактильное» восприятие информации умалилось только в последнее, «гутенберговское» время [3], но о его важности, к примеру в компьютерных играх, следуя Маклюэну, всерьез говорят в среде академических философов [4]. А что будет завтра с «галактикой Гутенберга», сделавшей ставку почти исключительно на зрительное восприятие информации, нам вообще не дано знать.

Итогом обучения программированию на МК (подход к программе как к автомату) является умение организовать собственную деятельность, используя имеющиеся в наличии ресурсы, тогда как остальные социальные навыки (из перечисленных) на данном этапе одним программированием не привьешь. Учащийся (потенциально) становится способен описывать вселенную в рамках картезианской картины мира (а не только смутно догадываться о познаваемости вселенной), воспринимать мир и его познание как конечный автомат (что тоже является позитивистским предположением и восходит к философии Декарта), а также приобретает навык управления собой. Но как управлять окружающими?

В пределах заявленной темы первый вузовский семестр хорошо рассматривать как подготовительный к освоению программирования и языков, его обеспечивающих, и отводить его под усиленное изучение того, чего лишь касаются учащиеся колледжа. На фоне углубленного рассмотрения логического устройства ЭВМ и автоматического исполнения программ понимание студентами того, что есть «программа» на языке ассемблера, становится при работе на МК автоматическим. Очень важным является еще одно приложение МК-практики: когда в истории развития ЭВМ говорится о том, что «первые ОС вводились прямо с пульта», апелляция к МК бывает более чем кстати. Таким образом, программирование на калькуляторе становится дешевым и действенным средством, обеспечивающим историческую связь этапов развития электронно-вычислительной техники, и материалом, «цементирующим» разрозненные части computer science. К сказанному следует добавить, что язык ассемблера предназначен для компьютерного процессора, устройства, об интеллекте которого речь вообще не идет, поэтому освоение автокода (особенно для МК) не является непосильным даже для относительно слабого учащегося.

Первый семестр, таким образом, должен быть посвящен тому, чтобы выработать у студента понимание «программы» и «машины» как синонимов автомата. Начиная со второго семестра упор должен быть сделан на ЯВУ, поскольку перед студентами становятся более серьезные задачи, чем управлять собой, ведь самообладанию мы предполагаем выучиваться в течение первого семестра. Во втором семестре на программу стоит взглянуть «синтаксически».

Какой язык программирования лучше всего взять здесь в качестве базового?

Предположим, что мы готовим бакалавров, которые должны уметь среди прочего в рамках нашего курса:

– осуществлять профессиональную деятельность и уметь решать задачи, соответствующие его квалификации;

– обладать специальной подготовкой в предметной области и знаниями перспективных информационных технологий проектирования, создания, анализа и сопровождения профессионально ориентированных информационных систем;

– знать задачи предметной области и методы их решения; технологии адаптации профессионально ориентированных информационных систем; перспективы развития информационных технологий и информационных систем в предметной области, их взаимосвязь со смежными областями;

– уметь формулировать и решать задачи проектирования профессионально ориентированных информационных систем для предметной области с использованием различных методов и решений; ставить и решать задачи, связанные с организацией диалога между человеком и информационной системой;

– владеть методами системного анализа в предметной области;

– иметь опыт работы с основными объектами, явлениями и процессами, связанными с информационными системами в предметной области, и использования методов их научного исследования; разработки проектных решений и их реализации в заданной инструментальной среде.

Для прививания соответствующих навыков, в принципе, может подойти любой ЯВУ, включая даже языки создания сайтов, поскольку об алгоритмизации можно говорить на любом языке программирования. Однако штука в том, что мы договорились не употреблять слово «алгоритмизация», да и о программе как об абстрактном конечном

автомате мы будем вспоминать теперь лишь от случая к случаю.

После «отмены „Бейсика“» нам хотелось бы исключить язык С и все его разновидности из процесса «обучения первоначальных». Дело не в том, что данный язык плох (а он хорош с известными оговорками), а в том, что он опасен. Из-за относительной «низкоуровневости» он может быть рекомендован только опытным программистам. Несмотря на то что в С есть типизация данных, из-за особенностей, связанных с нетривиальными правилами преобразования типов, которые не всегда очевидны и небезупречно логичны, и с неявным выполнением нескольких операций в одной, он может быть допущен к использованию лишь в качестве инструмента. Язык этот в некотором смысле близок автокоду, и его место скорее «среди железа», чем среди ЯВУ. К тому же не следует забывать, что серьезные задачи решаются отнюдь не на ПК, а языки программирования, которые для этого используются, отнюдь не всегда С. О последнем обязательно должно быть сказано в курсе «Языки программирования», там же должно быть определено его место, там же должно быть объявлено о его ограниченной опасности.

Исключив временно С, а ранее насовсем «Бейсик», мы будем вынуждены остановиться на языке «Паскаль», который признаем наилучшим среди учебных. Во-первых, он стандартизован ISO вместе с описанной в нем «реализацией уровня ноль». Во-вторых, благодаря данной реализации он способен «развернуться» практически на любой машине. В-третьих, в «Паскале» имеет место строгая типизация данных. Наконец, в нем отсутствует механизм каких-либо умолчаний и допустимы только очевиднейшие преобразования типов, к тому же строго заданные правилами «совместности по присваиванию». А в случае если есть какое-то недопонимание в преобразовании, на помощь спешит диагностическое сообщение компилятора об ошибке: type mismatch – несоответствие типов.

Посмотрим, как помогает «Паскаль» следующему уровню менеджмента – управлению людьми и как его пользователь может определять свое место в мире, приобретая управленческие навыки.

Что предполагает успешное управление людьми? Умение в готовую схему вставить наиболее подходящие звенья. Менеджер среднего звена, как правило, не связан с разработкой процессов, ему нужно работать с людьми, умея увидеть каждого на своем месте. В конечном счете эффективность работы такого менеджера определяется тем, как он может достичь наилучшего результата в заданных условиях. И как бы мы ни критиковали инструментальный разум и ни принижали его свойства [5], действия этого разума бывают необходимы хотя бы там, где нужно классифицирование (а в языке «Паскаль», например, оно является важнейшим инструментом описания действительности).

Язык «Паскаль» предлагает нам программу в качестве не столько автомата, сколько системы, сколько требуемого задачей описания модели целого «класса миров». Точно так же, как формула $c^2 = a^2 + b^2$ описывает «мир» не одного прямоугольного треугольника, а их бесконечного числа (или чуть более чем всех – пример из Яаакко Хинтикки [6]), так и программа на языке «Паскаль» устанавливает отношения между переменными. Мир, описываемый такой программой, достаточно богат, поскольку количество таксонов, на которые может быть разбито все многообразие объектов этого мира (используем слово «объект» в неспециальном значении), теоретически бесконечно. Мы можем выделять простые типы, характеризующиеся одним атрибутом, мы можем создавать сложные записные типы (структуры); единственное, что мы не можем – устанавливать новые связи (отношения, операции) как внутри таксонов, так и между ними. В этом смысле мы довольствуемся ролью классического «управляющего в лавке», который вынужден подчиняться не им придуманным законам, но должен в их рамках добиваться наибольшей выгоды. Его profit в том, чтобы система работала с минимальными затратами и максимальной эффективностью. С этой целью объекты мира расставляются так, чтобы их взаимодействие (в рамках задания) было если не наиболее разумным и очевидным (а очевидность не есть грех), то наиболее выгодным. Где под выгодой понимается то трудноуловимое мастерство в построении красивых моделей (систем) и создании их живописных картин (программ), о котором говорится в до сих пор не устаревших классических книгах [7–9], посвященных к тому же программированию на алголоподобных (т. е. близких к «Паскалю») языках.

Мы видим, что качества «Паскаля» способствуют воспитанию у менеджера навыков, отмеченных, например, в сухом перечне компетенций специалиста, который продолжает свое образование уже за пределами «общих областей». Так, например, курс «Информационные технологии в управлении персоналом» прямо подразумевает, что в результате его изучения у студента появятся:

– готовность к кооперации с коллегами, к работе на общий результат, навык организации и координации взаимодействия между людьми, контроля и оценки эффективности деятельности других;

– осознание основ современной философии и концепций управления персоналом, сущности и задач, закономерностей, принципов и методов управления персоналом, умение применять теоретические положения в управленческой деятельности по

отношению к персоналу;

– знание основ разработки и внедрения требований к должностям, *критериев подбора и расстановки персонала* (курсив наш. – *Е. М.*) и умение применять их на практике.

Конечно, о приобретении готовности к кооперации с коллегами в рамках изучения языка «Паскаль» речь еще не идет, хотя успешное написание программных комплексов коллективами авторов имеет место. Хотя это возможно и на «Фортране», тонкость, однако, в том, что в данных языках кооперация есть «побочная линия» использования, а не магистраль. Поэтому хотелось бы найти язык, созданный для решения совместных задач par excellence, для чего в обзорном курсе «Языки программирования» в качестве базового для практических нужд можно и до́лжно взять тот, который будет в наибольшей степени содействовать решению нашей задачи воспитания менеджера на данном этапе.

Итак, язык «Паскаль» при должном к себе отношении способен готовить не столько программиста (программисту в конечном итоге все равно, на чем работать, – выбор языка часто определяет заказчик), сколько человека, разбирающегося в построении моделей в рамках заданной картины мира и в построении, что более важно, управляемых систем, реализованных тем более прямо, чем более «паскалевски» мыслит программист.

Однако кроме отмеченных качеств язык «Паскаль» имеет еще одно, переоценить которое невозможно. Изучая программирование на «Паскале» с точки зрения синтаксиса языка и рассматривая программу как синтаксический конструкт, программист волей-неволей коснется грамматики. И здесь логичность и стройность определения языка обязательно будут им связываться (по аналогии) с разумностью и стройностью самого мироздания, ведь правила «Паскаля» суть законы всех миров, которые могут быть созданы в его терминах. А поскольку мы на Паскале моделируем реальные ситуации без насилия над ними, то следует думать, что реальность соприродна языку, причем красота и ясность описания языка сродни тем краеугольным камням, на которых держится мир, разумный и познаваемый, который не есть плод тщетных усилий неумелого демиурга, а совершенный продукт абсолютного строителя.

Однако как быть дальше, когда язык «Паскаль» становится базой плюсквамперфекта, а задачи усложняются?

Студенту, ориентированному на управление процессами, следует взглянуть на программу как на систему и обратиться к более серьезному средству программирования. На первый взгляд, лучше, чем язык «Алгол-68», здесь что-то придумать сложно: описание этого языка еще более полное и строгое, чем описание «Паскаля»; имея все привлекательные черты «Паскаля», «Алгол-68» лишен его ограничений и недостатков. В нем не только есть возможность создавать бесконечное число таксонов, но и существует средство устанавливать между ними отношения, описывая операции как внутри таксонов, так и между ними. Таким образом, мы в языке «Алгол-68» видим мощное и универсальное средство воспитания того класса менеджеров, представители которого способны моделировать процессы, привлекая к их созданию исключительно языковые средства, оставаясь на позиции строгой дисциплинированности.

Однако при всех привлекательных чертах «Алгол-68» не может быть рекомендован к изучению по ряду внешних причин. Первая, и самая главная, – он не стандартизован ISO, поэтому трудно говорить о том, где кончается собственно «Алгол-68», а где начинаются его «диалекты», определенные позднейшими добавками, связанными, например, с подключающими предложениями, модулями, модалами или средствами раздельной компиляции. Вторая причина: язык «Алгол-68» чрезвычайно труден для изучения. Третья: он хорош для теоретических изысканий (был оценен в свое время специалистами чрезвычайно высоко), но область его применения мала. Факультативное изучение этого языка менеджерами стоит всячески поощрять, но делать такое изучение обязательным – непозволительная роскошь.

Как быть?

Сведения о кое-каких иных языках можно почерпнуть в литературе. Из книг, например, мы можем узнать [10] о существовании языка, обучать которому опасаются только из страха перед его сложностью. Но ведь мы только что отвергли «Алгол-68» вовсе не из-за трудностей, связанных с его освоением.

Заменить «Алгол-68» можно не менее мощным языком «Ада». Данный язык в отличие от условно живого «Алгола-68» безусловно жив – он создан по заказу Пентагона и является рабочим языком американской армии; он стандартизован ISO, при этом стандартизации подлежит каждая разработка, предназначенная для конкретного «железа» и конкретного софта; подобно «реализации уровня ноль» языка «Паскаль» в нем есть маленькая «Ада» (*Small Ada*), которая несет основные черты большого языка. Если к этому добавить, что в 1995 г. был принят новый стандарт (известный как «Ада-95»), в который были введены средства объектного программирования, а в 2007 г. опубликованы изменения, усилившие объектно ориентированную составляющую языка, то к «Аде» стоит присмотреться.

Поскольку «Ада» является алголоподобным языком – в его основу положен «Паскаль», изучение «Ады» после «Паскаля» является естественным. Автор данных строк имеет опыт преподавания «Ады» студентам-старшекурсникам физико-математического факультета ТГПУ в 1999–2000 гг. в рамках обзорного курса «Языки программирования». Результат утешает: студенты легко воспринимают сложности «Ады», способны писать на нем (взаимодействующие) программы.

«Ада» может сыграть неоценимую роль в воспитании управленца. Пусть «Ада» не предполагает определения новых отношений между таксонами программы, как делает это «Алгол-68», она обеспечивает реальное взаимодействие между реальными людьми (и сопряжение бóльших единиц программирования, нежели переменные) в процессе совместной разработки комплексов программ. К этому «Ада» приспособлена изначально. Родовые модули, механизмы раздельной компиляции, пакеты и их подключение – все это позволяет коллективу разработчиков (каковым может быть группа студентов) выстраивать своих членов в иерархическую структуру, причем эта структура (и связанная с нею задача управления) будет описываться языком «Ада».

Завтрашний день принесет свои заботы, но среди них останутся универсальные – задачи управления, и мы обязаны способствовать тому, чтобы управленцы всех уровней выходили максимально снаряженными для любой возможной исторической битвы. Точно так же мы обязаны использовать для воспитания управленцев все доступные средства, выбирая из них наилучшие, предлагая каждому менеджеру оптимальный для его позиции язык программирования.

Список литературы

1. Леонтьев К. Н. Средний европеец как идеал и орудие всемирного разрушения // Избранное. М.: Рарогъ, Московский рабочий, 1993. С. 119–168.
2. Маликов Е. В. Информационно-коммуникационные технологии и социальные компетенции // Вестн. Томского гос. пед. ун-та (TSPU Bulletin). 2013. Вып. 11 (139). С. 144–149.
3. Маклюэн М. Понимание медиа: внешние расширения человека. М.; Жуковский: Канон-пресс-Ц; Кучково поле, 2003. 464 с.
4. Кузнецов М. М. Опыт коммуникации в информационную эпоху. Исследовательские стратегии Т. В. Адорно и М. Маклюэна. М.: ИФ РАН, 2011. 143 с.
5. Хоркхаймер М. Затмение разума. К критике инструментального разума. М.: Канон+ РОИИ «Реабилитация», 2011. 224 с.
6. Хинтикка Я. Логико-эпистемологические исследования. М.: Прогресс, 1980. 448 с.
7. Дал У., Дейкстра Э., Хоор К. Структурное программирование. М.: Мир, 1975. 248 с.
8. Дейкстра Э. Дисциплина программирования. М.: Мир, 1978. 275 с.
9. Турский В. Методология программирования. М.: Мир, 1981. 264 с.
10. Кауфман В. Ш. Языки программирования. Концепции и принципы. М.: Радио и связь, 1993. 423 с.

Маликов Е. В., доцент.
Евразийский открытый институт.
Ул. Нижинская, 7, стр. 1, Москва, Россия, 119501.
E-mail: eugene@malikow.ru

Материал поступил в редакцию 21.05.2014.

E. V. Malikov

PROGRAMMING LANGUAGES AND MANAGERIAL SKILLS

The paper deals with the problems of general managerial skills and the goals of the study of programming languages in the training of managers. In the course of such training the author proposes to distinguish between three levels of management competencies and three approaches to the study of programming languages. Therefore the notion of "program" is seen in three aspects: a program as an abstract finite-state automaton; program as a syntactic construct; program as a system – a model of the world. According to the aims of the course software from primitive level to high level is offered; the steps of learning are established; a number of high-level languages are compared; the existing experience is analyzed.

As part of the stated objective for training the author offers means of programming from primitive to high-level; builds a sequence of mastering the basics of programming; compares a number of high-level languages in terms of the task; analyses the existing experience of implementing these ideas in the learning process.

Key words: *management, programming languages, social competence.*

References

1. Leont'ev K. N. *Sredniy evropeets kak ideal I orudie vsemirnogo razrusheniya* [The average European as the ideal and the tool of the world destruction]. Izbrannoye [Selected]. Moscow, Rarog Publ.; Moskovskiy rabochiy Publ., 1993. Pp. 119–168 (in Russian).
2. Malikov E. V. Informatsionno-kommunikatsionnye tekhnologii i sotsial'nye kompetentsii [Information and communication technologies and social competence]. *Vestnik Tomskogo gosudarstvennogo pedagogicheskogo universiteta – TSPU Bulletin*, 2013, vol. 11 (139), pp. 144–149 (in Russian).
3. McLuhan M. *Ponimaniye Media: Vneshniye rasshireniya cheloveka* [Understanding Media: The Extensions of Man]. Moscow, KANON-press-Ts Publ.; Zhukovskiy, Kuchkovo pole Publ., 2003. 464 p. (in Russian).
4. Kuznetsov M. M. *Opyt kommunikatsii v informatsionnuyu epokhu. Issledovatel'skiye strategii T. V. Adorno i M. Maklyuena* [The experience of communication in the information age. Research strategy T. Adorno and M. McLuhan]. Moscow, IFRAN Publ., 2011. 143 p. (in Russian).
5. Horkheimer M. *Zatmeniye razuma. K kritike instrumental'nogo razuma* [Eclipse of Reason]. Moscow, Kanon+ ROII Reabilitatsiya Publ., 2011. 224 p. (in Russian).
6. Hintikka J. *Logiko-epistomologicheskiye issledovaniya* [Logical and epistemological studies]. Moscow, Progress Publ., 1980. 448 p. (in Russian).
7. Dahl O.-J., Dijkstra E. W., Hoare C. A. R. *Strukturnoye programmirovaniye* [Structured programming]. Moscow, Mir Publ., 1975. 248 p. (in Russian).
8. Deykstra E. W. *Distsiplina programmirovaniya* [Discipline of programming]. Moscow, Mir Publ., 1978. 275 p. (in Russian).
9. Turski W. M. *Metodologiya programmirovaniya* [Computer programming methodology]. Moscow, Mir Publ., 1981. 264 p. (in Russian).
10. Kaufman V. Sh. *Yazyki programmirovaniya. Kontseptsii I printsipy* [Programming languages. Concepts and principles]. Moscow, Radio i svyaz' Publ., 1993. 423 p. (in Russian).

Euroasian Open Institute.
Ul. Nezhinskaya, 7, Building 1, Moscow, Russia, 119501.
E-mail: eugene@malikow.ru