УДК 372.800.4 https://doi.org/10.24158/spp.2020.1.11

Локалов Владимир Анатольевич

кандидат педагогических наук, доцент, старший преподаватель факультета программной инженерии и компьютерной техники Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (Университета ИТМО)

Миронов Андрей Сергеевич

тьютор факультета программной инженерии и компьютерной техники Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (Университета ИТМО)

РАЗВИТИЕ АБСТРАКТНОГО МЫШЛЕНИЯ ШКОЛЬНИКОВ В ПРОЦЕССЕ ОБУЧЕНИЯ ОСНОВАМ ПРОГРАММИРОВАНИЯ

Аннотация:
В статье рассматривается проблема развития абстрактного мышления школьников при их обучении на курсе «Основы программирования» в системе дополнительного образования детей. Актуальность проблемы связана с необходимостью формирования у учащихся профессионального подхода к разработке программ. Описан метод оценки уровня развития абстрактного мышления, суть которого состоит в выявлении взаимосвязи между типовыми ошибками, допущенными учащимися при написании программ, и стадиями развития их понятийного мышления, по Л.С. Выготскому (синкреты, комплексы, настоящие понятия). Возможности указанного метода демонстрируются на примерах, связанных с развитием понятия «абстрактный тип данных», правильное формирование которого является необходимым условием системного мышления программиста. Даются методические рекомендации по развитию абстрактного мышления, которые были разработаны в процессе преподавания курса «Основы программирования» в Детско-юношеском компьютерном центре Университета ИТМО.

Ключевые слова:
методика преподавания информатики, обучение программированию, мотивация обучения, развитие абстрактного мышления.

Lokalov Vladimir Anatolyevich

PhD in Education Science, Associate Professor, Senior Lecturer, School of Software Engineering and Computer Systems, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics

Mironov Andrei Sergeyevich

Tutor, School of Software Engineering and Computer Systems, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics

DEVELOPMENT OF SCHOOLCHILDREN'S ABSTRACT THINKING IN TEACHING BASIC PROGRAMMING

Summary:
The study considers the problem of schoolchildren's abstract thinking development in teaching the Basic Programming course within the system of children's additional education. The relevance of the problem is related to the need for generation of students' professional approach to program design. The study presents a method for assessing the level of abstract thinking development which allows to identify the relationship between typical students' mistakes in writing programs and the stages of their conceptual thinking development according to Lev Vygotsky's theory (syncretes, complexes and real concepts). The potential of the given method is shown through the examples related to the development of "abstract data type" concept, the correct formation of which is a necessary condition for a programmer's systems thinking. The paper provides methodological recommendations on abstract thinking development, which were elaborated during teaching the Basic Programming course at the Children and Youth Computer Center of the Saint Petersburg National Research University of Information Technologies, Mechanics and Optics.

Keywords:
methods of teaching computer science, teaching programming, learning motivation, abstract thinking development.

Одной из важнейших задач обучения программированию в системе дополнительного образования школьников является постепенное формирование у учащихся профессионального подхода к разработке программ. Формирование этого подхода невозможно без развития абстрактного мышления, которое позволяет не только грамотно поставить задачу, выбрать методологию программирования, но и сосредоточить внимание программиста прежде всего на формальном решении задачи [1], которая в контексте выбранной методологии выражается через определенную систему абстракций (типа данных, управления и модульности) и модели вычислительного процесса [2].

Несмотря на важность развития абстрактного мышления в процессе обучения программированию, эта проблема лишь отчасти затрагивается в ряде публикаций, посвященных методике изучения программирования в рамках курса школьной информатики. Например, в работе Д.М. Гребневой [3] упоминается принцип возрастания степени абстрактности в контексте так

называемого семиотического подхода к обучению программированию. Согласно этому принципу, решая задачи по программированию, учащиеся должны постепенно переходить к более абстрактным знакам, знаковым системам и формальным описаниям. Однако, раскрывая данный принцип, автор мало уделяет внимания психолого-педагогическому обоснованию этапов развития абстрактного мышления и не делает каких-либо конкретных выводов, касающихся реальной практики организации и проведения занятий по программированию.

В работе И.Р. Дединского [4] мы уже можем видеть ряд более или менее конкретных методических рекомендаций, направленных на развитие формального стиля мышления программиста. В частности, предлагается фокусировать внимание учащегося не на процессе кодирования, а на процессе решения задачи. Таким образом, знание синтаксиса и семантики языка программирования оказывается не основной, а вспомогательной задачей обучения. Основной мерой усвоения знаний объявляется «когнитивно-технологическая единица», появляющаяся в результате использования механизмов рефлексии и критического мышления на различных этапах решения задач. На преподавателя же возлагается обязанность «следить за качеством кода обучаемых и стилем их мышления, поскольку искусство программирования – это искусство мышления» [5].

Безусловно, принципиально важно понимать, что существует связь между качеством разрабатываемых программ и качеством мыслительной деятельности, но не менее важно также понимать, что критическое мышление и рефлексия являются необходимыми, но отнюдь не достаточными условиями формирования грамотного подхода к решению задач. В самом деле, первые проявления критического мышления и рефлексии можно обнаружить уже в дошкольном возрасте [6]. Заметим, что для младшего школьного возраста уже имеется достаточно много методических разработок и программных средств (LOGO, Scretch и др.), которые предназначены для развития алгоритмических способностей интеллекта. Эти разработки можно было бы использовать и для развития критического мышления и рефлексии, однако следует подчеркнуть, что они созданы для школьников 7–11 лет, которые находятся еще на стадии конкретных интеллектуальных операций и мыслят, преимущественно опираясь на наглядно-образное содержание. Несмотря на интерес и желание детей этого возраста программировать, едва ли можно надеяться на то, что наглядные алгоритмы будут способствовать формированию грамотного подхода к решению задач. Этот подход может появиться только тогда, когда учащийся научится свободно рассматривать задачу на разных уровнях обобщения и детализации, а также системно мыслить и обобщать [7].

Таким образом, можно констатировать, что на сегодняшний день для работ, в которых так или иначе затрагиваются проблемы развития абстрактного мышления, характерно следующее:

– доказывается и обосновывается необходимость развития абстрактного мышления, но отсутствует теоретическая модель, в соответствии с которой можно было бы это развитие организовать;

– нет сколько-нибудь опробованных способов оценки уровня развития абстрактного мышления, которые можно было бы использовать непосредственно в практике преподавания основ программирования;

– практические рекомендации, полученные эмпирически, не дают ответа на вопрос, как развивать абстрактное мышление у тех учащихся, которые не обладают определенным уровнем рефлексии и критического мышления.

Для того чтобы обеспечить возможность выработки общих подходов к развитию абстрактного мышления, в настоящей работе мы предлагаем использовать теоретическую модель Л.С. Выготского, описывающую развитие понятий. Выбор данной модели можно обосновать экспериментальным подтверждением ее достоверности, а также тем, что она является одним из компонентов концепции зоны ближайшего развития (ЗБР), которая в настоящее время широко используется в педагогической практике.

Л.С. Выготский представлял понятие как единство его знаковой и смысловой стороны, а под развитием понятия подразумевал изменение значения понятия в процессе его использования [8]. Абстрактные понятия возникают на определенном этапе такого развития и являются результатом длинной цепочки преобразования соотношения «знак – смысл».

Принципиально важно, что, по теории Л.С. Выготского, понятия могут развиваться а) спонтанно («житейские понятия»), б) в результате обучения («научные понятия»).

В практике преподавания основ программирования мы сталкиваемся с тем, что спонтанные механизмы играют большую роль в образовании ошибок. Учащиеся в момент решения учебной задачи пытаются воспользоваться накопленным опытом непосредственно, не подвергая его необходимому критическому переосмыслению. В этом случае можно наблюдать первые две стадии развития понятийного мышления, о которых писал Л.С. Выготский, а именно стадии а) синкрет, б) комплексов.

Стадия синкрет

Синкретическая стадия развития понятия характеризуется субъективным установлением связей между знаком и значением понятий. Субъективность в данном случае будет заключаться в том, что учащийся ориентируется прежде всего на свой опыт и, не подвергая его никакой критической переработке, пытается использовать для решения задач. В этом случае преподавателю принципиально важно разобраться в том, какой именно «непригодный» опыт учащийся пытается применять.

Приведем пример, когда понятие «переменная» находится на стадии синкрет. Это случается при решении задач, в которых принципиально важно использовать переменную сначала одного типа, а затем другого. В такой ситуации бывает, что учащиеся не замечают, что есть какие-то разные типы, и используют переменные только одного типа. Например, в задаче, где требуется построить круговую диаграмму успеваемости, сначала надо посчитать проценты троек, четверок и пятерок (промежуточные результаты), которые должны храниться в переменных типа Real, а затем рассчитать углы круговой диаграммы, которые должны быть целыми числами (так требует библиотечная процедура рисования сектора). Часть учащихся использует при решении такой задачи только целочисленные переменные, что приводит к существенной ошибке результата, а часть пользуется только переменными Real, что приводит к синтаксической ошибке в программе при использовании графической функции.

В другой задаче, где требуется получить пару случайных буквенных символов (прописной и строчной), учащиеся используют только переменные типа Char, пытаясь присваивать целое значение генератора случайных чисел символьной переменной, тогда как для решения данной задачи требуется сначала сохранить результат генератора в целочисленную переменную (Integer), а уже затем преобразовать сохраненное значение в прописной и строчной символы.

Можно предположить, что далеко не последнюю роль в появлении ошибок в описанных выше примерах сыграл тот факт, что понятие «переменная» уже находится в сфере практического опыта учащихся (оно используется на уроках алгебры), а такой признак данного понятия, как «тип переменной», в их опыте отсутствует. Отсутствие типа у алгебраических переменных автоматически приводит к тому, что считается, что переменные программирования также не имеют типа. Привычные операции с алгебраическими переменными и выражениями один к одному переносятся на, казалось бы, такие же операции в языке программирования. Поэтому в программах школьников зачастую можно увидеть выражения, существующие сами по себе. Вместо

$$SUM := S1 + S2 + S3;$$

мы видим просто:

$$S1 + S2 + S3;$$

Типичной ошибкой является присваивание одному выражению или функции другого выражения, например в переменную записывается случайное число с помощью выражения

$$Random(6) + 1 := A;$$

тогда как правильной записью было бы

$$A := Random(6) + 1;$$

Свойство мышления образовывать синкреты и применять их для решения задач называется синкретизмом мышления. Этот тип мышления опирается на использование так называемых схем мышления [9], неких высказываний, идей, которые были зафиксированы в сознании без какой-либо существенной мыслительной переработки и приняты за истину. Такие схемы часто возникают под влиянием высказываний авторитетных личностей (родители, учителя и т. п.) или берутся из какого-нибудь источника информации. Типичный вопрос учащихся, связанный с наличием у них синкретизма, следующий: «Сколько переменных *надо* объявлять в программе?» В данном случае после того, как учащийся услышал от учителя, что надо объявить какое-то количество переменных в разрабатываемой программе, он представляет себе, что это число переменных является атрибутом этой программы и что надо заранее знать это «правильное» число.

Стадия комплексов

Как было показано выше, понятия, находящиеся в своем развитии на синкретической стадии, приводят к серьезным, иногда – фатальным ошибкам. Дальнейшее развитие понятий может и должно опираться на критическое переосмысление первичного опыта. Это переосмысление связано прежде всего с выделением в процессе практической деятельности одного или нескольких объективных признаков, на основании которых образуется понятие (в отличие от субъективных признаков на синкретической стадии). Выделенные признаки могут относиться, а могут и не относиться к существенным характеристикам понятия. Важно, что понятийный комплекс характеризуется тем, что строится на признаках, которых всегда недостаточно для образования полноценного (настоящего) понятия, что приводит к появлению весьма характерных ошибок.

Например, используя переменные в своих первых программах, учащиеся постепенно начинают понимать, что им можно присваивать значения, однако такая существенная характеристика, как ограниченность размера памяти, отводимого для хранения значения переменной, часто остается без внимания. Это приводит к ряду ошибок, подчас плохо уловимых, не контролируемых и не понимаемых учащимися. Во-первых, результат сложения двух больших положительных чисел типа Integer может быть отрицательным за счет того, что этот результат просто не помещается в переменную соответствующего типа. Во-вторых, операции с вещественным типом Real могут приводить к ошибкам вычислений, особенно если промежуточные операции содержат функции округления. Например, в программе одного из учащихся, в которой требовалось вращать графический объект, наблюдался любопытный эффект. Из-за многократного некорректного округления при пересчете координат точек при повороте на большие углы графическое изображение вращающегося объекта уменьшалось.

Одним из важнейших и существенных признаков понятия «переменная», который часто «теряется», когда это понятие находится еще на стадии комплекса, является «множество допустимых операций *для данного типа* переменной». Эта потеря может привести к тому, что учащиеся могут считать, что операции, которые они умеют производить с переменными числового типа, автоматически можно производить и с переменными других типов. Например, зная, что строки можно складывать (конкатенация), школьники тут же пытаются их вычитать и даже делить. Из поля зрения школьников часто выпадает и такой признак понятия «выражение», как его «тип», который определяется как тип результата вычислений, заданный выражением. Это существенный признак, поскольку он необходим для корректной записи результата выражения в переменную. В понятийном комплексе «выражение» этот признак подменяется другим, несущественным признаком, который определяется по типу переменных, в него входящих. Так, в операторе

$$C := A \,/\, B,$$

где A, B и C имеют тип Integer;

тип выражения $A \,/\, B$ (понятийного комплекса) считается целым, тогда как на самом деле этот тип является вещественным, и попытка присвоения его результата целой переменной приводит к ошибке.

При изучении структурированных переменных и написании соответствующих программ можно наблюдать такую форму комплекса, как комплект, для которого характерно объединение значений понятия по признаку дополнения. В этой ситуации разные, но объединенные в одном наборе объекты именуются одинаково. Нередко целое называется именем части. Так, некоторые школьники называют системный блок «процессором», а школьники, изучающие программирование, не сразу понимают разницу между элементом массива и массивом в целом. В этом случае они испытывают затруднения и с пониманием того, что такое индекс и как его использовать для обращения к элементу массива. Само понятие «индекс» может восприниматься как порядковый номер в натуральном ряду, поэтому даже индексы, начинающиеся с нуля, вызывают определенное непонимание.

Приведенные выше характерные ошибки, естественно, не являются фатальными, они тем или иным образом преодолеваются, и понятия «переменная» или «выражение» постепенно переходят от стадии комплексов к настоящим понятиям. Факторами, способствующими развитию понятий, являются психофизиологическое созревание (возраст), мотивация, а также грамотная методика преподавания основ программирования.

Развитие абстрактных понятий

Формирование настоящего понятия прежде всего означает завершение осознания всех его существенных объективных признаков. Этот процесс тесно связан с соответствующей практикой применения еще до конца не осознанного понятия для решения практических задач. Инструментальная функциональность такого понятия, успешность его применения говорит о том, что понятие уже существует, но, в силу своей неосознанности и отсутствия включенности в понятийную систему предметной области, оно является лишь «потенциальным» понятием [10]. Как выделение существенных признаков, так и «потенциальность» понятия (его функциональность) являются необходимыми условиями для формирования настоящих понятий.

Как известно, систему понятий можно представить в виде многоуровневой пирамиды, на каждом уровне которой располагаются понятия. В основании такой пирамиды находятся конкретные понятия, носящие характер имен собственных. Чем выше уровень, тем выше степень абстракции. Применительно к нашему случаю снизу вверх будут располагаться следующие понятия:

– переменная «a» (если переменная «a» определена как переменная типа Integer);
– переменная типа Integer;
– переменная (в контексте конкретного языка программирования).

Учащийся начинает знакомиться с понятием «переменная» именно в контексте конкретного языка программирования и конкретной задачи, где переменная имеет имя, а затем в процессе накопления практического опыта происходит постепенное «прорастание» этого понятия вверх, к вершине пирамиды. Естественно предположить, что в приведенном выше примере можно было бы добавить понятие с более высоким уровнем абстракции: переменную, рассмотренную уже безотносительно ко всякому языку программирования. Формирование такого понятия является одной из задач профессионально-ориентированного курса «Основы программирования».

Итак, очевидно, что после «входа» в пирамиду понятий с уровня абстракции, который определяется функциональностью понятия, дальнейшее развитие понятия идет в двух направлениях: вниз (в сферу конкретного практического опыта) и вверх (в направлении обобщения и формирования более абстрактных понятий).

Отметим важность выбора базового языка программирования, определяющего начальный уровень абстракции, от которого развитие понятий идет вверх и вниз. Проблемы могут возникнуть, если базовым является нетипизированный язык, который не предусматривает явного объявления типа переменной. Неявное задание типа намного усложняет понимание учащимися этого существенного признака переменной, что не может не отразиться на процессе образования настоящего понятия.

Движение вверх по пирамиде понятий, появление обобщающих понятий связано не только с формированием абстрактного мышления – этот процесс также необходимым и для развития системного мышления [11]. Системность мышления программиста позволяет сделать процесс решения задач максимально эффективным, а также справиться с многообразием языков программирования и сложностями языковых реализаций и конструкций.

Основой системного мышления является операция обобщения, суть которой состоит в выделении общих признаков, обеспечивающих одинаковую функциональность обобщаемых понятий. В процессе обобщения происходит абстрагирование, т. е. отбрасывание ряда признаков. В результате обобщения появляется новое, более абстрактное понятие, находящееся на более высоком уровне в пирамиде понятий.

Процесс обобщения понятия «переменная» инициируется появлением нового класса задач, отличающихся структурной и функциональной сложностью, для решения которых необходимо абстрагироваться от ряда признаков, характеризующих тип переменной (размер ячейки памяти, множество допустимых значений и др.).

Понятие «тип переменной» обобщается до понятия «абстрактный тип данных», которое полностью определяется исключительно через множество допустимых операций над данными. Новый уровень абстракции понятия «переменная» дает возможность проектировать объектно-ориентированные системы, решать задачи с использованием разнообразных статических и динамических структур данных (стеки, очереди, множества, деревья и др.) на определенном этапе, не ориентируясь на конкретный язык программирования и не заботясь об их реализации.

Удаленность уровня абстрактного типа данных в пирамиде понятий от уровня конкретной реализации не должна приводить к разрыву смысловых связей между этими уровнями, в противном случае могут возникнуть проблемы с разработкой программного кода, адекватного найденному абстрактному решению.

Похожие проблемы могут возникнуть при попытке обобщить неразвитые понятия, находящиеся на стадии комплексов. Это обобщение может привести лишь к образованию нового комплекса, но отнюдь не понятия, находящегося на более высоком уровне абстракции.

Методические рекомендации по развитию абстрактного мышления

Проанализировав закономерности развития абстрактных понятий, связанных с программированием, нельзя не заметить, что необходимым условием этого процесса является постоянное накопление учащимися практического опыта решения последовательности задач, в процессе которого они периодически сталкиваются с проблемами и для преодоления которых вынуждены использовать новое понятие или обобщать ряд понятий, сформированных ранее. При этом методика должна быть направлена на сокращение стадий синкрет и комплексов за счет максимально быстрого прорастания научных понятий (которые при самостоятельном изучении могут остановиться в развитии на уровне вербальных схем) в сферу практического опыта.

Расширение практического опыта, дающее эффект интеллектуального развития, в свою очередь должно быть связано с мотивацией овладения новыми видами и способами деятельности за счет совершенствования инструментальных средств.

Упомянутые выше принципы развития абстрактного мышления, на первый взгляд, кажутся очевидными, однако на пути его реализации по ряду причин возникает множество проблем.

Первые и существенные сложности встречаются уже на ранней стадии, когда перед учащимися не могут быть поставлены достаточно сложные задачи, которые, во-первых, вызывали

бы у них сильный интерес, а во-вторых, требовали бы предварительного этапа проектирования, на котором решались бы не в терминах языка программирования, а с помощью отвлеченных понятий. Чтобы предотвратить появление привычки у учащихся мыслить в терминах языка программирования и использовать лобовые решения (типа копирования строк кода вместо организации модулей), преподаватель должен попытаться найти соответствующие методические приемы. Например, для активизации внутренней мотивации на решение простых задач он может предложить учащимся сюжеты задач, стилистически отвечающие их чувству юмора. К этим задачам относится задача моделирования электронного родителя (с помощью оператора выбора), который будет спрашивать о школьных отметках и по-разному реагировать на то, что сообщил ему ребенок. Подобные задачи стимулируют поиск простого алгоритмического решения на естественном языке, которое на основе использования средств того же естественного языка может быть обобщено на целый класс подобных задач.

На протяжении всего курса «Основы программирования» преподаватель должен знакомить учащихся с новыми понятиями, уровень абстракции которых постоянно возрастает по мере изучения курса. Для того чтобы методика введения новых понятий способствовала развитию абстрактного мышления учащихся, целесообразно при ее разработке учитывать важнейший психологический закон, определяющий условия возникновения интеллектуальных реакций, который обычно называется «закон запруды». Суть его состоит в том, что условия для интеллектуального развития возникают тогда, когда на пути привычного психического процесса возникает препятствие, что приводит к росту психической энергии, которая может быть перенаправлена на интеллектуальное решение возникшей проблемы.

На учебных занятиях вообще и по программированию в частности ситуация препятствия возникает, когда учащийся не может с помощью привычных способов справиться с решением поставленной перед ним учебной задачи и вынужден для ее решения искать какие-то новые подходы, методы, инструменты. Подобную ситуацию можно и нужно использовать, когда необходимо научить учащихся пользоваться новыми (для учащихся, конечно) интеллектуальными инструментами для решения задач – понятиями, связанными с концепциями данных, управления, модульности. Для этого прежде всего преподавателю следует выбрать и поставить перед учащимися такую задачу, чтобы они не смогли ее решить с помощью инструментальных средств и подходов, имеющихся у них на данный момент.

Например, если требуется познакомить учащихся с понятием «массив», то нужно предложить им такую задачу, чтобы ее невозможно или чрезвычайно сложно было бы решить с помощью переменных простых типов. К таким задачам, в частности, относится задача нахождения среднего арифметического случайного ряда чисел и вывода на экран отклонения для каждого значения. Для наглядности можно посчитать отклонение роста каждого учащегося группы от среднего роста группы. Обсуждение решения задачи целесообразно начать с попытки решить эту задачу известным способом, т. е. с помощью набора разноименных переменных. Сложность и громоздкость такого решения позволяет преподавателю обосновать необходимость введения особого типа данных – массива, как бы состоящего из перенумерованного набора однотипных переменных, а также показать, что использование массива позволяет решать целый класс подобных задач независимо от объема исходных данных (подсчет среднего роста и отклонений для учащихся школы, района и т. д.).

Описанный подход к введению нового понятия дает сразу несколько преимуществ с точки зрения как обучения программированию в целом, так и развития абстрактного мышления в частности. Во-первых, он позволяет учащимся понять, что изучение нового инструмента всегда связано с необходимостью появления у него новой функциональности, которая дает возможность преодолеть возникшую трудность. Это в конечном счете приводит к пониманию связи инструментальных возможностей с функциональностью разрабатываемой программы. Во-вторых, успешное решение сложной задачи повышает мотивацию их обучения. И, наконец, в-третьих, новое понятие дает возможность «достраивать» пирамиду понятий, постепенно продвигаясь в сторону формирования абстрактных понятий. Так, переменная типа массив, обычно первый структурированный тип данных, с которым сталкиваются учащиеся в процессе обучения основам программирования, является важным шагом на пути формирования абстракции данных.

Предложенные выше методические рекомендации по развитию абстрактного мышления были разработаны и опробованы при проведении занятий на курсе «Основы программирования» Детско-юношеского компьютерного центра Университета ИТМО [12]. Практика показала, что в процессе освоения курса при переходе ко все более и более сложным задачам, принципиально требующим высокого уровня абстрактного мышления для их формализации, учащиеся могут испытывать серьезные затруднения. Для их успешного преодоления требуется такая организация

учебного процесса, при которой каждое занятие способствовало бы постепенному формированию осознанного понимания учащимися того, что повышение уровня абстракции понятий, используемых в программировании, является не только способом упрощения решения задач, но и единственной возможностью профессионального совершенствования программиста.

Ссылки:

1. Kramer J. Is Abstraction the Key to Computing? // Communications of the ACM. 2007. April. Vol. 50, no. 4. P. 36–42. https://doi.org/10.1145/1232743.1232745.
2. Одинцов И.О. Профессиональное программирование. Системный подход. 2-е изд., перераб. и доп. СПб., 2004. 624 с.
3. Гребнева Д.М. Семиотический подход к обучению программированию в школе [Электронный ресурс] // Научное обозрение. Педагогические науки. 2016. № 3. С. 13–27. URL: https://science-pedagogy.ru/ru/article/view?id=1495 (дата обращения: 29.11.2019).
4. Дединский И.Р. Аналитический подход к довузовскому преподаванию программирования [Электронный ресурс] // Сайт методики довузовского обучения программированию и проектной деятельности в информатике. 2011. URL: http://ded32.net.ru/news/2011-04-03-58 (дата обращения: 29.11.2019).
5. Там же.
6. Чуковский К.И. Собрание сочинений [Электронный ресурс] : в 15 т. Т. 2 / сост., коммент. Е. Чуковской. 2-е изд., электрон., испр. М., 2012. 640 с. URL: http://www.chukfamily.ru/wp-content/uploads/2017/03/Chukovskiy_K._Sobraniye_sochineniy_Tom_2.pdf (дата обращения: 29.11.2019).
7. Одинцов И.О. Указ. соч.
8. Выготский Л.С. Лекции по психологии. Мышление и речь. М., 2019. 432 с.
9. Пиаже Ж. Речь и мышление ребенка. М., 2008. 848 с.
10. Выготский Л.С. Указ. соч.
11. Там же.
12. Lokalov V.A. Children's Computer Club as an Example of Non-Formal Educational System in the Field of Informatics [Электронный ресурс] // Educational Alternatives. Journal of International Scientific Publications. 2014. Vol. 12. P. 27–37. URL: https://www.scientific-publications.net/en/article/1000475/ (дата обращения: 29.11.2019).

References:

Chukovskiy, KI 2012, *Collected Works*, in 15 vols, vol. 2, 2nd ed., Moscow, 640 p., viewed 29 November 2019, <http://www.chukfamily.ru/wp-content/uploads/2017/03/Chukovskiy_K._Sobraniye_sochineniy_Tom_2.pdf>, (in Russian).

Dedinsky, IR 2011, 'Analytical Approach to Pre-University Teaching Programming', *Website of Methodology of Pre-University Teaching Programming and Project Activities in Computer Science*', viewed 29 November 2019, <http://ded32.net.ru/news/2011-04-03-58>, (in Russian).

Grebneva, DM 2016, 'Semiotic Approach to Teaching Programming at High School', *Nauchnoe obozrenie. Pedagogicheskiye nauki*, no. 3, pp. 13-27, viewed 29 November 2019, <https://science-pedagogy.ru/ru/article/view?id=1495>, (in Russian).

Kramer, J 2007, 'Is Abstraction the Key to Computing?', *Communications of the ACM*, April, vol. 50, no. 4, pp. 36-42, https://doi.org/10.1145/1232743.1232745.

Lokalov, VA 2014, 'Children's Computer Club as an Example of Non-Formal Educational System in the Field of Informatics', *Educational Alternatives. Journal of International Scientific Publications*, vol. 12, pp. 27-37, viewed 29 November 2019, <https://www.scientific-publications.net/en/article/1000475>.

Odintsov, IO 2004, *Professional Programming. A Systemic Approach*, 2nd ed., St. Peterburg, 624 p., (in Russian).

Piaget, J 2008, *The Language and Thought of the Child*, Moscow, 848 p., (in Russian).

Vygotsky, LS 2019, *Lectures on Psychology. Thinking and Speech*, Moscow, 432 p., (in Russian).

Редактор: Хорева Людмила Николаевна
Переводчик: Герасимова Валентина Евгеньевна