Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт педагогики психологии и социологии
Кафедра информационных технологий обучения
и непрерывного образования

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Смолянинова О.Г.
« _____» _____ 20 ___ г.

БАКАЛАВРСКАЯ РАБОТА

44.03.01 «Педагогическое образование»
44.03.01.09. «Информатика и информационные технологии в образовании»

Серия уровневых заданий по программированию для обучающихся 6-х классов на основе модели индивидуального прогресса

Руководитель _____ канд. пед. наук, доцент Баженова К.А.
Выпускник _____ Морозова К.В.

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В федеральном государственном образовательном стандарте третьего поколения в пункте «Общие положения», обозначено, что достижения личностных, предметных и метапредметных результатов освоения основной образовательной программы основного общего образования, необходимы для продолжения образования, и являются предметом итоговой оценки освоения обучающимися основной образовательной программы [28]. Таким образом, обозначена необходимость выстраивать учебный процесс, так, чтобы учащиеся могли усваивать специфические предметные действия, личностные и метапредметные в рамках предмета «Информатика».

В примерной основной образовательной программе основного общего образования в пункте «1.2.3.12. Информатика» отмечены планируемые предметные результаты основного образования [24]. Данным фактом отмечена необходимость образовательному учреждению достигать этих результатов, а также оценивать динамику индивидуальных достижений учащихся по информатике.

Как отметил И.Д. Фрумин, в современном образовании есть два разных подхода к оценке качества образования в децентрализованных образовательных системах – это оценка как проверка и оценка как механизм диалога и саморазвития. Оценка как проверка (оценка-контроль), строится на стремлении выявить недостатки школы. В этом случае реализуется не идея ответственности, а идея подконтрольности. В случае оценки как механизма диалога и саморазвития (оценка-поддержка) ответственность возникает не из подконтрольности, а из открытости и прозрачности. У этих подходов разные цели, и у второго подхода цель именно оценка динамики индивидуальных достижений обучающихся [30, с.18-20]. Следовательно, достоинство второго подхода в том, что процедура оценки результатов образования является в то же время процессом развития учителя и самого школьного института, так как оценивается динамика индивидуальных достижений учащихся, что

соответствует федеральному государственному образовательному стандарту третьего поколения.

В настоящее время разработана модель индивидуального прогресса учащегося (Б.Д. Эльконин, Б.И. Хасан, А.М. Аронов, О.В. Знаменская, и др.); разработан и применяется в ряде российский школ диагностический инструментарий «Дельта» для измерения уровня индивидуального прогресса мышления и понимания (О.В. Знаменская, О.И. Свиридова, Л.А. Рябинина и др); разработаны и реализуются программы повышения квалификации для педагогов по освоению идеи применения модели индивидуального прогресса (О.И. Дятлова, О.В. Знаменская, О.И. Свиридова, Л.А. Рябинина и др); подходы к обеспечению условий для пробно-поисковых действий учащихся (Л.С. Выготский, Л.М. Долгова, П.Г. Нежнов, К.Н. Поливанова и др).

Модель индивидуального прогресса является условием для достижения личностных, предметных и метапредметных результатов. В методике «Дельта» разработаны уровневые задания для основной школы по следующим предметам: математика, русский язык, и частично, физика и биология. Разработчиками уровневых заданий алгоритмической линии по математике являются А.М. Аронов, О.В. Знаменская, К.А. Баженова, В.Г. Ликонцева, О.А. Францен и др. Однако, уровневых задач по информатике нет.

Базовым понятием в информатике является понятие «программирование». Понятия «программирование» и «алгоритм» в контексте деятельности программиста (computer science) изучали Н. Вирт, А.П. Ершов, С. Паперт, А.Д. Перлис и др.; понятие «алгоритм» как одно из важных элементов математики и информатики изучали А.К. Звонкин, Ю.А. Макаренков, А.А. Столяр и др.; в учебниках по информатике по основной школе программирование изучается в таких средах как Visual Basic, КУМИР (Л.Л. Босова, Н.Д. Угринович, И.Г. Семакин и др.).

Целью исследования является разработка серии уровневых заданий по программированию для учащихся 6 классов на основе модели индивидуального прогресса.

Объект исследования: модель индивидуального прогресса освоения понятия «программирование» учащимися 6 классов.

Предмет исследования: серия уровневых заданий по программированию для учащихся 6 классов на основе модели индивидуального прогресса.

В основу исследования была положена следующая **гипотеза**. Уровневыми задачами на освоение понятия программирование можно считать, если:

– Задания первого уровня предполагают выполнение заданного алгоритма самим учащимся. В данном задании «алгоритм» понимается как инструкция к действию для самого учащегося, учащийся становится на позицию исполнителя.

– Задания второго уровня направлены на выделение общего принципа работы с понятием «программа». При решении задач становится явным отличие алгоритма от программы.

– Задания третьего уровня ориентированы на поиск оптимальных решений. Чтобы решить такое задание нужно уметь редактировать программу в связи с изменением материала. Умение использовать простые коды для написания сложных.

В соответствии с поставленной целью и выдвинутой гипотезой были определены следующие **задачи**:

1. Провести анализ понятий «программирование», «алгоритм» для того, чтобы выделить уровни освоения понятия «программирования».
2. Выделить специфику серии заданий алгоритмической линии по математике, составленных на основе модели индивидуального прогресса.
3. Разработать серию уровневых заданий по информатике на основе модели индивидуального прогресса.

4. Составить методические комментарии к заданиям.

Структура выпускной квалификационной работы состоит из введения, двух глав, заключения, списка используемых источников, приложений.

1. МОДЕЛЬ ИНДИВИДУАЛЬНОГО ПРОГРЕССА НА МАТЕРИАЛЕ УЧЕБНОГО ПРЕДМЕТА ИНФОРМАТИКА

1.1. Алгоритмическая культура в современном образовании

В этом параграфе на основе анализа исследований авторов вводится образовательное значение понимания сущности алгоритмической культуры. Это позволяет зафиксировать характеристики алгоритмической культуры, её место в программировании.

Обучение в узком смысле может пониматься как традиционное обучение, которое связано с передачей знаний, умений и навыков. Знания фигурируют в виде фактов, понятий, законов, теории. Умение – это действие, направляемое с четко осознаваемой целью, и достигается в традиционном обучении упражнениями. Навык уже автоматизированное средство и достигается посредством выполнения многократных похожих упражнений [15, с.45].

Обучение в широком смысле может пониматься как «процесс освоения нового» [15, с.45]. Здесь речь идет не только о знаниях, умениях, навыках, но и о новом опыте, о прохождении новых событий, и, что нас интересует – об освоении новых компетентностей [15, с.45].

При введении понятия освоения значительно изменяются позиции учащегося и учителя, они становятся «равномощными» (равнозначными). Тем не менее, учитель в отличие от учащегося, уже имеет в наличии средства освоения. В таком случае, учащийся осваивает новое, в то время как учитель помогает это новое ему осваивать [15, с.45].

В связи с принципами деятельностного подхода в педагогике, в основе которой лежит культурно-историческая теория Л.С. Выготского, содержание образования в каждом учебном предмете должно строиться как система культурных средств действия [2, с.35]. В традиционной дидактике основной тип действия учащегося – это тренировочное действие, направленное на формирование навыка выполнения задания по образцу или алгоритму. В деятельностной дидактике существует понятие пробных действий.

Организация пространства для пробных действий, способствует активности и становлению самостоятельности самого учащегося. Такой процесс обучения способствует повышению уровня индивидуальных образовательных результатов за счет того, что опирается на личные интересы и инициативы учащего [10, с.191]. Для того чтобы что-то понять учащемуся необходимо совершить ряд действий с предметом, чтобы выявить содержание будущего понятия, а также, чтобы определить первичное содержание этого понятия [8, с.101]. Таким образом, стержень содержания школьного образования является процесс присвоения культурных средств (способов) мышления и действия. Такой подход позволит перейти в массовой школе на индивидуальный прогресс учащегося и его оценку [3, с.35]. В таком случае, реализовывается сознательность учащихся, так как они получают знания не в готовом виде, а выясняют условия их происхождения [3, с.35-36]. Получается, для того, чтобы определить содержание в школьном предмете, следует выяснить какие основные понятия и деятельности должны быть сформированы в определенном предмете (предметно-деятельностные линии). Таким образом, если рассматривать преподавание информатики в деятельностном контексте, то следует определить базовую деятельность, которую предстоит освоить обучающимся.

В примерной основной образовательной программе основного общего образования (далее — ПООП ООО) в пункте «1.2.3.12. Информатика» отмечены следующие основные пункты планируемых предметных результатов основного образования: информация и способы её представления, основы алгоритмической культуры, использование программных систем и сервисов, работа в информационном пространстве [23]. Таким образом, основные пункты планируемых предметных результатов можно представить, как следующие предметно-деятельностные линии учебного предмета «информатика»:

– линия представления информации, где основное понятие «информация»;

– алгоритмическая линия, где основные понятия «алгоритм» и «программирование»;

– линия базовых навыков работы с компьютером (в примерной основной образовательной программе представлено как использование программных средств и сервисов), где основные понятия «программные средства» и «сервисы»;

– линия деятельности в информационном пространстве, где основные понятия «информационное пространство», «информационная этика» и «ИКТ».

В алгоритмическую культуру, которая прописана в основной образовательной программе основного общего образования, входят следующие предметные результаты: сформировать понятие об алгоритме и навыки составления и анализа алгоритмов; сформировать понятие об исполнителе и практические навыки работы с исполнителем; сформировать понятие о программе и практические навыки написания программ; развитие алгоритмического мышления [25, с.76].

Э.А. Нигматулина, Т.А. Степанова утверждают, что под алгоритмической культурой, как правило, «принято понимать совокупность специфических представлений, умений и навыков, связанных с понятием алгоритма, формами и способами его записи», а также «основой компьютерной грамотности, которые сегодня являются частью общей культуры каждого человека» [20, с.82].

Программирование развивает операционный стиль мышления, куда входят следующие умения и навыки: умение планировать структуру действий, необходимых для достижения цели, при помощи фиксированного набора средств; умение строить информационные модели для описания объектов и систем [12, с.34]. Также программирование развивает алгоритмическое мышление, которое шире, чем операционное мышление. Алгоритмическое мышление предполагает понимание сути базовых алгоритмических структур (линейный алгоритм, ветвление, цикл), также

умение их использовать при составлении простых алгоритмов, и строить сложные алгоритмы на основе простых [25, с.21].

Стась А.Н. и Долганова Н.Ф. приводят перечень компонентов алгоритмического стиля мышления:

– способность к формированию предметных суждений;

– способность к формированию репродуктивных и продуктивных навыков;

– способность к анализу задачи, ее декомпозиции на уровне процессов;

– способность к формализации задачи;

– понимание и способность к реализации элементарных алгоритмических операций [25, с.22-27].

Таким образом, возникает предположение о том, что обучение программированию способствует развитию алгоритмической культуры учащегося. Далее, в параграфе 1.3. будет рассмотрена деятельность программирование.

1.2. Уровневая модель освоения предметного способа действий

В этом параграфе описана модель индивидуального прогресса. В качестве примера приводится описание уровней освоения предметного действия на материале математики.

В своих трудах Л.С. Выготский пишет: «Педагогика должна ориентироваться не на вчерашний, а на завтрашний день детского развития». [5, с.251]. Так, в требованиях ФГОС, где на сегодняшний день обучение должно ориентироваться не на трансляцию знаний и навыков по образцу, а на достижения учащихся, которые, выходят за границы традиционной школьной успешности, где оценивается багаж знаний (основанных на концепции Коменского – «учить всех всему»). Такие образовательные достижения связывают с понятием «компетенция» и во ФГОС они представлены как личностные, предметные и метапредметные результаты.

Учителю следует ориентироваться на эти результаты. Здесь будет уместен вопрос «как формируется то или иное умение и переходит в компетентность?» Учителю важно понимать, как его действия, совместная работа с учеником влияют на индивидуальный прогресс учащегося. Важно, что речь идет не о внешних признаках изменений учащегося, то есть не о количестве пройденных тем, или даже не о сданных экзаменах, а о качественных изменениях в развитии мышления и действия учащегося, которые формируются в определенные компетенции.

Модель индивидуального прогресса основывается на культурно-исторической концепции Л.С. Выготского. Согласно этой концепции, учащийся развивается в той мере, в какой присваивает культурные орудия, которые являются способами действия и средствами мышления. Таким образом, центральный элемент модели индивидуального прогресса является опосредствование, то есть присвоение культурных орудий, и учебное содержание рассматривается как система культурных орудий [18, с.18]. Л.С. Выготский описывал три стадии присвоения культурных орудий следующим образом: изначально средства появляются вне человека, затем эти средства человек оборачивает на себя, и, наконец, пропадает необходимость во внешнем средстве — средство становится личным орудием человека, компетенцией [4, с.27]. Таким образом, на основе культурно-исторической концепции Л.С. Выготского выстроена трехуровневая модель индивидуального прогресса.

Авторы уровневой модели индивидуального прогресса (Б.Д. Эльконин, А.М. Аронов, О.В. Знаменская, Л.А. Рябинина, О.И. Свиридова, Б.И. Хасан и др.), определяют индивидуальный прогресс как «степень, в которой ребенок овладевает культурными способами действия», а компетенция понимается как «средство, прошедшее все стадии освоения» [18, с. 11].

В образовательном цикле выделяются три фазы становления компетентности: первые две связаны с присвоением общего способа действия, а третья — это этап, на котором сам способ функционализируется

[18, с.27]. Таким образом, чтобы выделить уровни освоения понятия в предметной области, следует, определить какую компетенцию понятие формирует.

На основе модели индивидуального прогресса конструируются специальные задачи, которые называются уровневыми задачами. Они позволяют определять уровень освоения предметного действия учащегося. И дают учителю возможность управлять формированием предметного действия. Такие задачи позволяют определить, какое предметное средство еще осваивает или уже освоил учащийся – правило, общий способ или же ключевую идею. [18, с.43]. Следует отметить, что применяя уровневые задачи в образовательном процессе, есть возможность оценить достижения учащегося в данный момент и зону его ближайшего развития, в связи с этим – осмысленно поставить новую педагогическую задачу. Уровневые задания, построенные на модели индивидуального прогресса, могут являться условием для освоения предметного действия.

Индивидуальный прогресс позволяет увидеть «комплексную положительную динамику личных ресурсов, которая включает в себя линейные и уровневые приращения способностей мышления и понимания», где уровневое приращение – выход учащегося на следующий уровень мышления и понимания, а линейное приращение – увеличение степени свободы действия в рамках какого-то одного уровня [18, с.44].

Модель индивидуального прогресса состоит из следующих уровней освоения учебного действия. Первый уровень – освоение общего смысла и формы способа действия, это означает, что учащийся воспроизвел способ действия по образцу [17, с.15]. Таким образом, выполняя задания первого уровня, учащийся ориентируется на известные ему алгоритмы, шаблоны, правила.

Второй уровень – выполнение задания на основе выделенного существенного отношения (общего принципа). На данном уровне уже недостаточно демонстрации образца. Для успешного решения задания

необходимо уметь удерживать существенные отношения. Учащийся при выполнении действий ориентируется на общие принципы и понятия изучаемого предмета. Это означает, что учащийся способен анализировать материал, обнаруживать закономерности и существенные характеристики в изучаемом предмете, причем он способен преодолевать «зашумления» [17, с.18]. Это означает, что ученик второго уровня не просто усвоил материал соответствующего учебного предмета, но и понимает, как он организован, то есть он освоил общий принцип действия.

Третий уровень – включение обобщенного способа действия в состав личных ресурсов учащегося, то есть возникает способность увидеть и осуществить действие, в котором способ, заимствованный в известном предмете, уместен и выступает как преобразующий материал для другого предмета. Учащийся, освоивший третий уровень, может в новых ситуациях, отличных от ситуации формирования, принимать и отвергать, корректировать и преобразовывать само существенное основание способа действия. Кроме того, учащийся овладел способами и знаниями так, что может применять их для решения разнообразных, в том числе жизненных задач. Это требует умственных действий рефлексии, синтеза и обобщения высокого уровня [17, с.18]. Следует отметить, что учащийся третьего уровня освоил действия рефлексии, синтеза и обобщения, что способствует включению общего способа действия в состав личных ресурсов учащегося, который он может преобразовывать и применять.

В диагностическом комплекте методики «Дельта» выделено три основные для школы предметно-деятельностные линии учебного предмета «математика»: моделирование, следование инструкции, формулирование утверждений. Нас интересует именно линия «следование инструкции», так как в этой линии основное понятие – алгоритм. В «Теории алгоритмов» понятие алгоритма трактуется как «общий, единообразный, точно определенный способ решения задачи из данной проблемы» [15]

Понятие алгоритм в математике – одно из фундаментальных основ. В математике важно умение формулировать и применять алгоритмы для развития математических умений и умений формулировать правила, выполнять их.

Алгоритмическая линия предметной области «математика» представлена в серии уровневых заданий «следование инструкции». Серия «следование инструкции» связана с пониманием и реализацией алгоритмов, на которые опирается система математических процедур. Уровни применительно к серии «следование инструкции»:

– На первом уровне задания на опознание текста как инструкции к действию без устных пояснений;

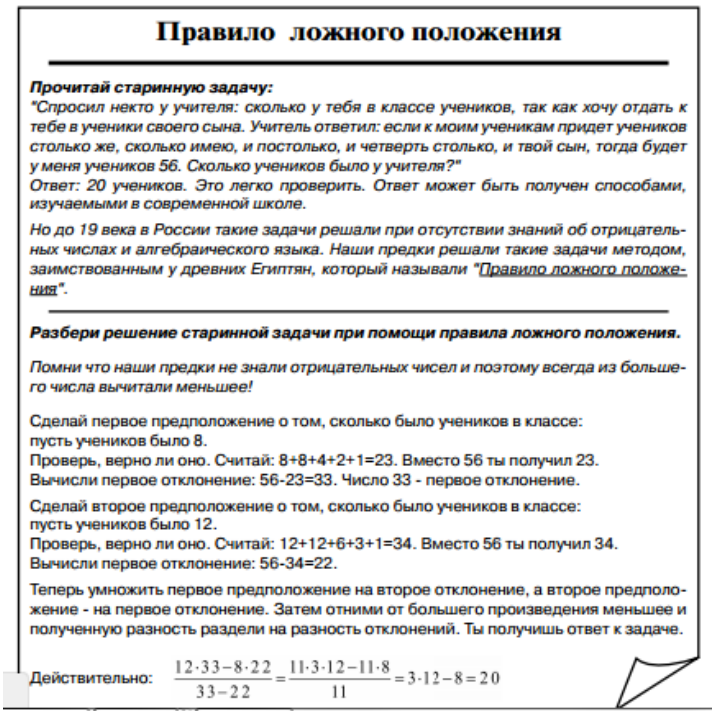
– На втором уровне задания на выделение существенного и адекватного применения для решения задачи алгоритма, данного в виде образца применения;

– На третьем уровне задание на редактирование алгоритма в соответствии с изменением типа материала [17]. Следовательно, серия «следование инструкции» предметной области «математика» может служить условием не только для оценивания, но и становления компетенции умения применять алгоритмы для решения математических задач.

Чтобы приступить к определению уровней освоения понятия «программирование», необходимо проанализировать специфику серии заданий алгоритмической линии по математике, составленных на основе модели индивидуального прогресса.

Рассмотрим серию уровневых заданий следования инструкции под названием «Правило ложного положения» из учебного издания «Мониторинг индивидуального прогресса учебных действий школьников» [17]. В отличие от других линий освоения предметного действия в математике (моделирование и формулирование утверждений) учащемуся предполагается ознакомиться с образцом (см. рис.1).

Почему именно здесь дается образец? Так как линия называется алгоритмической, она говорит сама за себя. В первую очередь, учащемуся нужно научиться действовать согласно инструкции. В примере «Правило ложного положения» представлен алгоритм. При решении задачи задан образец рассуждения для решения задачи – это последовательность действий для достижения определенного результата, в частности, определить «сколько учеников было у учителя».



Правило ложного положения

Прочитай старинную задачу:
"Спросил некто у учителя: сколько у тебя в классе учеников, так как хочу отдать к тебе в ученики своего сына. Учитель ответил: если к моим ученикам придет учеников столько же, сколько имею, и постолько, и четверть столько, и твой сын, тогда будет у меня учеников 56. Сколько учеников было у учителя?"
Ответ: 20 учеников. Это легко проверить. Ответ может быть получен способами, изучаемыми в современной школе.

Но до 19 века в России такие задачи решали при отсутствии знаний об отрицательных числах и алгебраического языка. Наши предки решали такие задачи методом, заимствованным у древних Египтян, который называли "Правило ложного положения".

Разбери решение старинной задачи при помощи правила ложного положения.

Помни что наши предки не знали отрицательных чисел и поэтому всегда из большего числа вычитали меньшее!

Сделай первое предположение о том, сколько было учеников в классе:
пусть учеников было 8.
Проверь, верно ли оно. Считай: 8+8+4+2+1=23. Вместо 56 ты получил 23.
Вычисли первое отклонение: 56-23=33. Число 33 - первое отклонение.

Сделай второе предположение о том, сколько было учеников в классе:
пусть учеников было 12.
Проверь, верно ли оно. Считай: 12+12+6+3+1=34. Вместо 56 ты получил 34.
Вычисли первое отклонение: 56-34=22.

Теперь умножить первое предположение на второе отклонение, а второе предположение - на первое отклонение. Затем отними от большего произведения меньшее и полученную разность раздели на разность отклонений. Ты получишь ответ к задаче.

Действительно: $\frac{12\cdot 33-8\cdot 22}{33-22}=\frac{11\cdot 3\cdot 12-11\cdot 8}{11}=3\cdot 12-8=20$

Рисунок 1 - Правило ложного положения [17]

В данном образце предлагается ознакомиться с образцом – прочитать задачу и разобрать её посредством решения с помощью правила ложного положения. Учащемуся следует прочитать и понять его, чтобы приступить к решению самих уровневых задач. Образец написан в виде пошаговой инструкции к действию, как ответ на вопрос: «что делать, чтобы добиться определенного результата?». Образец должен быть написан четко и понятно, чтобы на вопросы учащегося ответ уже содержался в самом образце, так как он дается для создания ситуации понимания.

Учащийся знает некоторые алгоритмы вычисления таких задач, однако в образце предлагается решить другим способом: правилом ложного положения.

Учащийся выделяет поверхностно признаки алгоритма решения задачи посредством правила ложного положения: первое предположение число наугад, это число при необходимости изменяется по ходу задачи (постольку, четверть столько); вычисления отклонений; формула для получения ответа.

После образца дано двухуровневое задание первого и второго уровня:

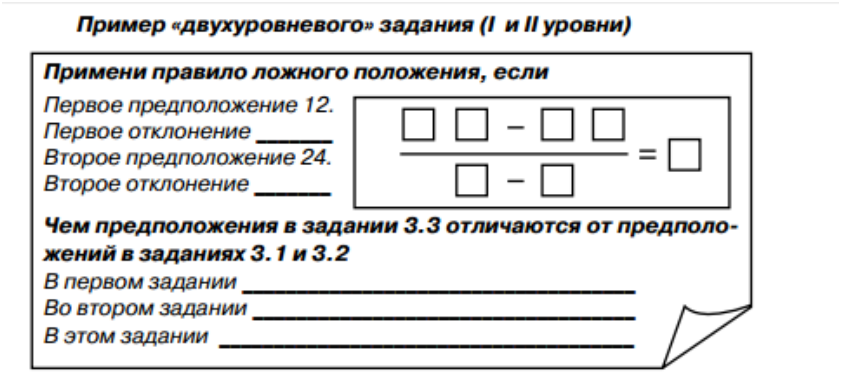


Рисунок 2 - Пример двухуровневого задания (1 и 2 уровни) [17]

Это задание ученик может решить, только если освоит образец. На этапе такого освоения предметного средства — правило.

Задание является двухуровневым потому, что в первом случае (первый уровень) – это этап освоения образца и здесь учащимся формально применяется инструкция (алгоритм, шаблон, правило) и не обнаруживается изменение материала. Задание специально сконструировано таким образом, чтобы учащийся задумался: «а ко всем ли таким случаям может применяться правило ложного положения?» Считается, что учащийся выполнил задание на первом уровне, если формально верно заполнил схему и не заметил арифметической ошибки. Учащийся выполнил задание на втором уровне, если он заметил арифметическую ошибку и отказался от поиска неверного ответа из-за нее.

В задании третьего уровня нужно изменить правило ложного положения для применения в новых условиях:

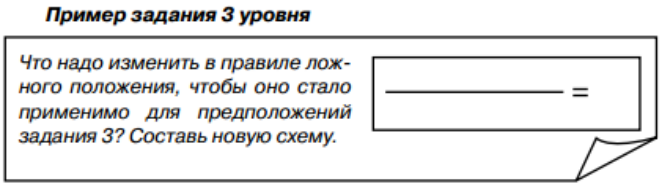


Рисунок 3 - Пример задания 3 уровня, [17]

Этап освоения действия на этом уровне – ключевая идея. Для того, чтобы решить задание такого уровня, учащемуся надо понять то, что не изменяется, фиксировано в правиле ложного положения, и то, что можно изменить в нем. То, что никак нельзя изменить – и есть ключевая идея. Учащийся, определив эту идею, понимает, как математическое правило организовано, за что отвечает каждое действие в правиле ложного положения. В таком случае учащийся сможет решить задание на третьем уровне.

Все операции в алгоритме входят в школьную программу и соответственно предполагаются уже известными для учащихся. Успешное выполнение следующих действий является критерием понимания алгоритма с точки зрения математики:

– реконструкция правила по данному частному случаю его применения (освоение нового алгоритма);

– применение алгоритма для решения аналогичной задачи, используя предложенный пример как образец;

– обнаружение ограничений (неприменимости) алгоритма, связанных с особенностями предметного материала и, наконец, перестройка алгоритма так, чтобы он охватывал обнаруженные новые случаи.

1.3. Уровни освоения понятия «программирование»

Задачей этого параграфа является ввести понятие программирование и уровни его освоения. С. Пейперт, Н.Н. Непейвода и др. разделяют деятельность программирования и обучение программированию. Обучение программированию как культурной деятельности основано на реконструкции деятельности программированию.

Понятие «программирование» – базовое понятие информатики (computer science). Программирование — это деятельность, а информатика — это наука про свойства объектов, с которыми имеет дело эта деятельность. Информатика появилась благодаря следующим аспектам исторического развития программирования: появление ЭВМ, программирование ЭВМ и языки программирования, компьютерные сети, развитие программного обеспечения [30, с. 748].

Чтобы сконструировать уровневые задачи освоения понятия «программирование» на основе модели индивидуального прогресса, необходимо выяснить, в чем состоит содержание понятия программирования. В литературе понятие «программирование» фигурирует в двух ракурсах. В узком смысле, под программированием понимается разработка программного обеспечения, составление программного кода для ЭВМ, которая в качестве результата выдаёт набор данных – решение задачи [24, с. 64].

Например, Непейвода Н.Н. утверждает, что цель работы программиста – «создание программы, описывающей некоторый процесс» [19]. Под этим процессом автор имеет в виду следующие структуры: декомпозицию программы; структурирование процесса её выполнения, где появляются процессы, вызовы процедур и т.д.; структурирование данных, перерабатываемых в ходе компиляции программы. Все эти виды структур взаимосвязаны и при программировании планируются совместно. В таком случае, под программой автор понимает структуру данных для компилятора. [19, с. 245]

В широком смысле, программирование – деятельность, направленная не только на создание программного кода, но и на формализацию «предопределенного состояния по реакции на событие, реализуемого средствами математики или естественных наук» [24, с. 64]. Таким образом, программирование включает два этапа. Первый – перевод реальной ситуации

на формальный язык математики. Второй – составление программного кода – кодирование.

Однако, первый этап сложный, поскольку реальная ситуация или процесс вначале описывается как алгоритм действий, понятный для человека. Чтобы суметь написать алгоритм для ЭВМ в виде программы нужно понимать, о чем пишешь. Алгоритм изначально делается для человека, а потом уже для ЭВМ. Только после этого осуществляется формализация на язык математики и язык программирования. Поэтому этап формализации мы выделяем одним из ключевых моментов. Каждый раз программист сталкивается с новой задачей. Программирование может быть разным и задачи, решаемые через программирование могут быть совершенно разными от «исправить ошибки в чужом коде» до «написать сегодня до вечера сайт» (с некоторыми программистскими задачами можно ознакомиться, например, на сайте фриланса, или на форумах по программированию. С более серьезными программистскими задачами, над которыми должны работать группы программистов, можно ознакомиться в научных изданиях, - например, о робототехнике, ИИ). Приведем пример, расчета взрыва бомбы. Физики знали, что происходит с ураном в лаборатории, но никто не представлял, что будет происходить в бомбе. Поэтому заранее нужно было всё точно сосчитать, и без ошибок, чтобы не взорвалось раньше времени. Процесс формализации здесь очень трудоемок: техническая модель, описывающая механизм бомбы; физическая модель, описывающая воздействия окружающей среды; физическая модель, описывающая сам процесс цепной реакции и его запуск. Всё это математически формализуется. После чего пишется код программы [28, с. 47-86]. Бывают случаи, что на поиски алгоритма решения некоторых задач требуются несколько лет, это очень громоздкие задачи, как с полетом ракеты. Но когда алгоритм уже создан и не имеет ошибок, то есть уже является полностью готовой программа, он уже не требует каких-либо рассуждений и сводится только к строгому выполнению команд алгоритма.

Ряд авторов выделяют этапы процесса разработки программы (Данова Н.С. Пономарев О.П., Хоор, Рогов): формализация и создание технического задания на исходную задачу; разработка алгоритма решения задачи; написание программы – кодирование; отладка и тестирование. Рассмотрим их более подробно.

Этап формализации. Разработка любого программного обеспечения начинается с анализа требований к будущему программному продукту. [24, с.28] На данном этапе определяется, решаема ли задача. Если необходимо, то задачу перевести в формальную модель. Как правило, на язык математики, – задача принимает вид формулы, и дальше следует этап построения алгоритма для ЭВМ [32, с.15]. Таким образом, вначале человек описывает процесс в виде элементов и формы явлений. Например, экономика или биология требует достаточно много времени, так как, чтобы разработать определенную программу в этих областям, надо понимать ту часть, те механизмы, над которыми работаешь, для которых создаешь программу. И формализовать, описать механизм математическими методами для биологии или экономики другие формы найти достаточно проблематично, так как в основном в программировании используются именно математические методы.

Этап построения алгоритма (алгоритмизация). Этот этап заключается в построении решения задачи в форме алгоритма [32, с.16].

Алгоритм строится для решения задачи, которая сформулирована, прошла этап формализации. Формулировка задачи описывает, каким требованиям должно удовлетворять решение задачи, а алгоритм, решающий эту задачу, находит объект, этим требованиям удовлетворяющий. Алгоритм считают правильным, если он выдает результат, удовлетворяющий требованиям задачи [16, с.14]. На этом этапе важно знать о том, что есть алгоритм и какими алгоритм обладает свойствами, раскрывающими его значение.

Этап кодирования. Запись программы на каком-либо языке программирования. На данном этапе происходит «процесс написания

программного кода с целью реализации определенного алгоритма на определенном языке программирования для функционирования в компьютерных устройствах с целью получения определенного результата» [25]. При этом, программой является последовательность инструкций, предназначенная для исполнения вычислительной машиной, на определенном языке программирования с целью получения определенного результата [25, с. 65]. Чтобы алгоритм можно было считать программой он должен быть однозначно понятным для человека; удовлетворять точно определенным правилам построения – синтаксису языка программирования; также понятен для исполнителя (абстрактного или конкретного) [20, с.6].

Фаулер М. утверждает, что хороший код, должен ясно сообщать на какой результат он направлен, что он делает. Если код большой и сложный, то лучше его разделить на мелкие составные части, это облегчит понимание кода и нахождение в нем ошибок [27, с.27]. Чтобы минимизировать ошибки в коде, желательно еще на этапе написания алгоритма, делать его структурированным. Фаулер М. пишет, что «только хорошие программисты пишут код, понятный людям». С ним нельзя не согласиться, в хаосном коде даже сам разработчик будет путаться, из-за того, что нет явной цели — в таком случае, код будет написан зря и проще его полностью стереть и начать заново, чем долго искать причину [27, с.32]

Этап отладки (компиляция). На этапе отладки осуществляется поиск ошибок и исправление обнаруженных. Чтобы исправить ошибку необходимо определить причину её появления – фрагмент или оператор, которые содержат ошибку. Не всегда причины появления ошибки очевидны, - такие можно найти по косвенным признакам посредством тщательного анализа кода программы. Рогов разделяет ошибки по характеру возникновения: синтаксические, алгоритмические, структурные и концептуальные [25, с. 101-102]. После этапа отладки, если имеются ошибки в алгоритме то следует вернуться к этапу алгоритмизации, а может даже и к формализации, возможно, не совсем правильно поставлена задача. Если ошибки в самом

синтаксисе кода, то следует вернуться к кодированию и устранять ошибки в самом коде.

Этап тестирования. На этом этапе проверяются характеристики поведения или свойства программы. Тестирование должно заканчиваться некоторыми ожидаемыми результатами, которые занесены в отчет о тестировании. В отчете о тестировании представляется анализ результатов тестирования программы, также сопоставление с требованиями и оценка программы [25, с. 99-100].

Теперь обратимся к обучению программированию. Э.А. Нигматулина считает, что «традиционно обучение программированию осуществляется на учебных примерах и задачах, линейно – от простого к сложному». Такой подход способствует тому, чтобы учащийся научился понимать возможности алгоритмических приемов и особенности кодирования, также осваивать новые приемы, которые позволяют развить алгоритмическую культуру и «привить навыки алгоритмической деятельности» [21, с.82]. Такой подход – от простого к сложному обосновывает уровневую модель индивидуального прогресса, которая используется в схеме оценивания дельта.

Так, в учебнике по информатике для 6 класса Босовой Л.Л. обучение программированию начинается с жизненных и математических примеров задач. После чего определяется алгоритм как описание последовательности шагов, приводящей к результату. Автором отмечается, что перед составлением алгоритма должны быть четко определены начальные условия и ожидаемый результат выполнения алгоритма. Исполнителем алгоритма может быть человек или всевозможные технические устройства, способные выполнять определенный набор команд (компьютер, станок и др.) Далее автор пишет про формы записи алгоритмов, где определяется понятие программы как алгоритма, записанного на понятном исполнителю языке. Для написания программы необходимо знать систему команд исполнителя. После автор переходит к типам алгоритмов: линейный, с ветвлениями, с повторениями (циклический). После чего следует тема про управление

исполнителем чертёжником, где автор отмечает про то, что данный исполнитель выполняет только правильные команды. Выделяются синтаксические ошибки и логические. Если допущена синтаксическая ошибка, то причина в коде, в программе и чертёжник сразу же сообщает об ошибке. Если допущена логическая ошибка, то есть все команды записаны верно, но их выполнение не будет приводить к поставленной цели — значит, причина в алгоритме и нужно вернуться к этапу разработки алгоритма, чтобы исправить ошибку [1, с. 100-132].

Н. Вирт в книге «Алгоритмы и структуры данных» отмечает, что научить программированию невозможно просто дав какой-либо список инструкций учащемуся. Как минимум, чтобы научить программированию, необходимы образцовые примеры и прилежание самого учащегося. Эти самые образцы нужны для того, чтобы учащийся научился видеть в программе процессы ее разработки: написание алгоритма, формализация ситуации на математический язык и после на язык программирования [6, с.13].

Н.Н. Непейвода утверждает «практическая цель обучающегося — научиться даже без предварительного изучения языка видеть в нем общие конструкции» и «понимать программы, написанные на нем». Также автор на первом этапе обучения предлагает примеры программ на разных языках программирования, для того, чтобы учащийся научился видеть общие конструкции [20, с.2-3].

С. Паперт обозначает программирование как общение с машиной на языке понятном ей, и пользующемуся ею человеку [23, с. 14]. Также, он выделяет два результата обучения программированию детей: учащиеся могут научиться пользоваться компьютером профессионально и обучение пользоваться компьютером может изменить способ изучения других предметов у учащегося [23, с.17]. Для изучения программирования необходимы соответствующие условия: более частый и свободный доступ к компьютеру, чем предполагается в школе.

Теория поэтапного формирования (П.Я. Гальперин) умственных действий и понятий в узком, собственном смысле слова представляет собой детально разработанную систему положений о механизмах и условиях сложных многоплановых изменений, связанных с образованием у человека новых образов, действий и понятий. Речь идет о понимании пути освоения человеком общечеловеческого опыта. Так, путь освоения программирования лежит через деятельность программирования, через его этапы.

Освоение общего способа не может быть его сообщением — трансляцией учителя о нем. Оно должно быть выстроено именно как решение учебных задач, таким образом, чтобы учащийся начинал с предметно-практического действия. Реальное предметное действие в дальнейшем переходит в понятие. Так, учащийся осознанно подходит к определенному понятию, благодаря специально организованной работе школьников, где учащиеся осуществляют такие действия как поиск и проба средств решения задачи [11, с. 36].

Подход П.Я. Гальперина заключается в том, что содержание деятельности человека формируется в индивидуальном опыте, а процесс этого формирования совершается по этапам. На каждом этапе перед учеником появляется задание, и представленная ориентировочная основа действия. В результате успешного выполнения и подкреплений, ориентировочная основа действия превращается в динамический стереотип, а каждый из ее ориентиров — сначала в раздражитель отдельной операции, а затем в элемент динамического стереотипа. Таким образом, внешняя организация задачи и процесса ее решения превращается в некую систему условных связей, составляющих частей новых знаний и умений [6, с. 136].

На материале программирования — раздражителем являются подзадачи, которые ведут к основной задаче. Посредствам основной задачи осваивается динамический стереотип, то есть деятельность. Следовательно, именно от постановки задач зависит освоение учеником этапов

программирования – формализация, алгоритмизация, кодирование. После чего, это осваивается и становится компетенцией учащегося.

Как мы видели выше, ключевыми этапами является формализация, алгоритмизация и кодирование.

Можно выделить следующие уровни освоения понятия «программирования», на основе модели индивидуального прогресса и анализа специализированной литературы по основам программирования. Задания первого уровня предполагают выполнение алгоритма, они создают опору на поверхностное осознание понятия «алгоритм». В данном задании «алгоритм» понимается как инструкция к действию для самого учащегося, учащийся становится на позицию исполнителя. Учащийся на первом уровне учится работать с образцами кодов, желательно, чтобы образцы были написаны на разных языках программирования. Либо с образцами алгоритмов.

Задания второго уровня направлены на понимание понятия «программа». Алгоритм, записанный на понятном компьютеру языке программирования – есть программа. Учащийся при выполнении задания первого уровня был в позиции исполнителя–человека, при выполнении задания второго уровня учащийся встает в позицию исполнителя–машины. Здесь будет уместно процитировать Алана Перлиса: «Чтобы понять программу, необходимо отождествить себя и с машиной, и с программой». Задания второго уровня ориентированы на выделение общего способа действия в задаче – способы построения алгоритма и программы.

Задачи первого и второго уровня можно сконструировать и иным способом. В первом уровне задачи могут быть сделаны как будто для исполнителя-компьютера, а для второго уровня для исполнителя–человека. Задания третьего уровня ориентированы на поиск оптимальных решений и модификацию алгоритма. Чтобы решить такое задание нужно уметь редактировать алгоритмы в связи с изменением материала.

Реконструкция понятия программирования позволяет сделать вывод о том, что программирование способствует развитию алгоритмической культуры у учащихся, куда входят алгоритмическое и операционное мышление. Понятие программирования позиционируется как деятельность программирования и обучение программированию. В деятельности программирования ключевым моментами мы выделили формализацию, алгоритмизацию и кодирование (отладка-компиляция), так как это основополагающие моменты в процессе программирования, без них программы быть не может. Рассматривая обучение программированию важно понимать, что можно учащиеся могут научиться программировать, научиться управлять компьютером профессионально, а может и компьютер программировать обучаемого, то есть повлиять на его способ изучения других предметов.

2. РАЗРАБОТКА СЕРИИ УРОВНЕВЫХ ЗАДАНИЙ АЛГОРИТМИЧЕСКОЙ ЛИНИИ «ИСПОЛНИТЕЛЬ – ЧЕРТЁЖНИК»

2.1. Структура и содержание серии заданий «Исполнитель–Чертёжник»

Среда КУМИР разработана на основе методики Ершова, где главная мысль заключалась в том, что входной язык программирования (для освоения понимания понятия программирования) совершенно независим от конкретных машин, не имеющий никаких привязок к той или иной машине, язык в виде коротких общих указаний, удобном для формулирования задач вычислительной математики [13, с.11-13]

Помимо этого, в среде КУМИР исполнитель чертёжник является средством, позволяющим интегрировать школьные предметы «Информатику» и «Математику». Она помогает осваивать такие понятия как координатная плоскость и система координат, симметрия. В приложении Б мы видим, что средой исполнителя чертежника является координатная плоскость, а в командах для этого исполнителя содержится система координат. Например, команда «сместиться в точку (1,0)» – описание того, где будет находиться точка и часть алгоритма, то есть, что сделать, чтобы оказаться в определенной точке (в результате написания буквы). В уровневых заданиях алгоритмической линии «Исполнитель–Чертёжник» мы не затрагиваем понятие симметрии. Возможно, при конструировании буквы «А», нужно соблюдать симметрию, но акцент на этом не делаем.

В среде Лого, в частности, исполнителе черепашке школьный предмет «Математика» расширяется, здесь уже появляются такие понятия, как угол, градус и др. Исполнитель–Черепашка может работать со всеми основными алгоритмическими структурами: линейная, разветвляющаяся и циклическая, а исполнитель–чертежник в КУМИРе только с линейной. Таким образом, КУМИР помогает понять, что такое управляемая машина (в роли управляемой машины исполнители), а Лого – перейти от управления к программированию (расширенный функционал команд).

Серия уровневых заданий алгоритмической линии «Исполнитель-Чёртежник» ориентирована на освоение предметного действия понятия «программирование».

В уровневых заданиях выделяются существенным условия, при которых чертёжник будет писать буквы на координатной плоскости. Данный момент необходим для понимания того, в каких рамках работает исполнитель и вообще для понимания того, что у машинного исполнителя есть границы его действия (СКИ) и есть условия определенной программной среды, которую нужно прочувствовать, чтобы понимать, как задавать команды, как работать с определенным исполнителем в определенной среде. Условия, при которых «Исполнитель-Чертёжник» будет писать буквы (и не только буквы) на координатной плоскости:

– линии прямые;

– нет наклонов и закруглений — линии рисуются по координатной сетке;

– более одного раза по одной и той же линии проходить нельзя;

– в начале рисования перо по умолчанию находится в точке (0;0);

– рисование происходит на координатной плоскости.

Серию уровневых заданий можно перестроить под среду исполнителя черепашки. В таком случае, нужно учитывать некое различие СКИ (система команд исполнителя) чертежника от черепашки. У черепашки в отличие от чертежника имеется направление точки (черепашка, как человек, стоит в каком-либо направлении), более того, черепашка может поворачиваться под определенно заданным углом, и черепашка более ориентирована на программирование. Чтобы умело ориентироваться в среде черепашки, учащиеся уже должны пройти по школьной программе математическое понятие «градус» и уметь оперировать с ним, либо про «градус» должно быть четко и понятно прописано в образце, который предоставляется перед уровневыми заданиями.

Характеристика уровневых заданий.

Задание 1 уровня. Алгоритм представляет собой последовательность команд, определяющих действия исполнителя. Указание к действию для формального исполнителя в виде команд (задания см. в приложении А).

Здесь важно понимать о том, что есть формальные и неформальные исполнители. Формальными исполнителями могут быть компьютер, робот, прочие технические устройства, в редких случаях человек (например, выполнение алгоритма умножения столбиком). Такие исполнители производят алгоритм без всяких элементов творчества с его стороны, и абсолютно не обдумывая его, то есть формально. Программа — это алгоритм, представленный на языке исполнителя. Например, робот может выполнять работу токаря, если он умеет выполнять все операции токаря (включать станок, закреплять резец, перемещать резец, замерять изделие). От исполнителя не требуется понимания сущности алгоритма, он должен лишь точно выполнять команды, не нарушая их последовательности. Таким образом, алгоритм и программа не отличаются по содержанию, но могут отличаться по форме.

Задание 1.1. ориентировано на выполнение действий по алгоритму.

Задание 1.2. ориентировано на реконструкцию алгоритма, образ которого задан. Не нужно выходить за рамки образца, только переходная позиция, где по какому-либо образу пишется алгоритм.

2 уровень

Задания о различии границ возможностей между человеком и компьютером/чертёжником и любым другим формальным исполнителем – а значит, о различии алгоритмов. Соответственно, алгоритм для человека может иметь практически любой вид, так как человек может при желании найти ответ, даже дополнить пробелы в алгоритме. А для формального исполнителя алгоритм должен быть точный и соблюден синтаксис языка программирования – алгоритм в виде программы – в любом случае, алгоритм должен иметь результат. Так как,– не могу точно сформулировать – если бы вместо буквы была абракадабра, то алгоритм бы не имел смысла, он бы

ничего не дал, это была бы просто каша из несвязных команд. Историю происхождения алгоритма говорит о том, что он был впервые применен для высчитывания столбиком – результат то, что получится при вычислении.

Чтобы написать программу «Буква Г» нужно:

- Представить образ буквы (по элементам и одновременно как систему элементов);
- Определить точку начала рисования буквы;
- Определить траекторию движения рисования буквы;
- Написать алгоритм «Буква Г»;
- Написать алгоритм «Буква Г» для формального исполнителя, то есть программу.

В задании 2.1. учащемуся предлагается координатная плоскость как ресурс для представления образа результата. Учащийся, который освоил образец и выходит на общий способ действия, начнет рисовать на плоскости ту часть алгоритма, которая уже есть; потом он мысленно или карандашом на координатной плоскости будет выискивать образ буквы; и после этого он запишет алгоритм до конца. Учащийся, который не смог освоить общий способ действия, либо допишет алгоритм не правильно, либо не сможет решить задание вообще.

В задании 2.2. учащемуся нужно представить образ написания буквы чертёжником, иначе он не сможет написать алгоритм. Учащийся, который увидит будущий образ слова, сможет написать алгоритм. Соответственно, учащийся, который не увидит, столкнётся с затруднениями.

Учащийся оперирует знаниями о системе команд исполнителя, выбирает то, что может исполнитель выполнять.

Задание 2.1. ориентировано на выделение общего способа действия написания буквы. Решено, если учащийся смог представить образ буквы на координатной плоскости и дописать недостающие команды в алгоритме. Учащийся первого уровня, это решить не сможет, так как он не может отойти от образца

Задание 2.2. ориентировано на представление образа слова, чтобы написать к нему алгоритм.

Чтобы решить задание третьего уровня нужно присвоить общий способ действия, который определился на втором уровне. На этом уровне совершается процесс кодирования. В том числе, освоенность понятия «программирования» на третьем уровне означает свободу выбора программного кода для решения задачи, а также высокую степень при формализации задачи.

2.2. Характеристика выполнения серии заданий учащимися 6 класса

Уровневые задания решаются учащимися самостоятельно. Крафт утверждает, что самостоятельная деятельность есть обязательное условие саморазвития личности [17]. Рассмотрим, что умеет делать ученик, освоивший предметное действие на материале информатики (раздел «Алгоритмизация и программирование»).

Учащийся принимает позицию формального исполнителя и выполняет алгоритм формально в указанной последовательности, то есть абстрагируется от содержания алгоритма, механически исполняя каждую команду алгоритма. Он может следовать алгоритму, вовсе не вникая в смысл своих действий до тех пор, пока он не выполнит алгоритм до конца и не увидит полностью образ результата своих действий.

Это является важным процессом для понимания того, что исполнитель-компьютер алгоритм выполняет формально и никак иначе. Учащийся, выполняя алгоритм пошагово, может видеть каждую деталь в нем: за что отвечает каждая команда (например, выполнив команду, которая призывает к действию провести линию к определенной точке – ученик увидит прямую черту), и наконец, к каким результатам приводит вся система команд, то есть алгоритм.

Учащийся выделяет формальные свойства алгоритма: дискретность (ученик видит, что алгоритм идет не сплошным текстом, а разбит на

отдельные команды), детерминированность (ученик замечает, что алгоритм имеет строго определенную последовательность команд и каждая команда в алгоритме должна быть однозначно и точно определена), результативность (ученик видит, что алгоритм обязательно приводит к какому-либо результату), также алгоритм должен быть выполнимый и понятный, иначе он не имеет смысла. Учащийся понимает, что алгоритм это последовательность действий, обязательно приводящая к результату.

В заданиях первого уровня проверяется овладение алгоритмом действия. На данном этапе учащемуся, чтобы выполнить задания, необходимо отойти от формального выполнения алгоритма и начать рассуждать, проникать глубже в его содержание. Ученику необходимо представить образ будущего результата выполнения алгоритма. Учащийся начинает замечать, что главное свойство алгоритма – результативность, так как если алгоритм не приводит к результату, то он не имеет смысла. Поэтому, обучающийся может сам выбрать последовательность действий при создании элементов объекта

В заданиях второго уровня учащийся определяет разницу между алгоритмом и программой. Понимает, что программа пишется для формального исполнителя, алгоритм — для не формального (человека). Определяет разницу между формальными и не формальными исполнителями (см.параграф 2.1.).

Наконец, учащийся понимает, что есть такие этапы как формализация (в серии уровневых заданий алгоритмической линии «Исполнитель-Чертёжник» этого этапа нет, так как задания уже формализованы), алгоритмизация (написание алгоритма), кодирование (написание программы).

Комментарии к заданиям:

1. Задание считается выполненным, если правильно выполняется учащимся исходный алгоритм. Первому уровню соответствует такое решение, когда учащийся может воспроизводить алгоритм в соответствие с

образцом. Посредником при решении задач является образец, правило, шаблон или алгоритм, в частном случае именно алгоритм.

Выполнение этого задания свидетельствует о том, что учащийся освоил образец. Если учащийся не выполнил это задание, это значит, что основные предметные образцы учащимся осваиваются не успешно.

Задание 1.1. (Приложение А) ориентировано на следование образцу.

Задание 1.2. (Приложение А) Учащийся, который выполнил задание, умеет по заданному образу, воспроизвести алгоритм.

Двухуровневые задания.

Задание 1.2.1. (Приложение А) ориентировано на выделение элементов буквы. Задание будет выполнено на первом уровне если учащийся изобразит элементы буквы Ю нераздельно (возможно и раздельно для этого уровня) и напишет алгоритм не для одного элемента буквы, а для буквы Ю полностью. Задание сделано на втором уровне, если учащийся изобразит элементы буквы раздельно и алгоритм одного из элементов буквы.

Задание 1.2.2. (Приложение А) Задание сделано на первом уровне, если учащийся напишет алгоритм формально, перепишет алгоритм для исполнителя-человека, не осознавая того, что алгоритм надо написать для исполнителя-чертёжника.

Чтобы решить задание второго уровня нужно выделить общий способ действия написания букв исполнителем чертёжником, чтобы понять, как работает этот исполнитель, как он выполняет команды, отображая их на координатной плоскости.

Задание второго уровня считается выполненным, если правильно и до конца выполнен алгоритм по заданной инструкции в новых условиях, выходящих за рамку применимости инструкции, и обнаружено существенное отличие условий, не позволяющее непосредственно применить освоенную инструкцию ранее. А также если преодолены «зашумления» в задании.

Учащийся, который смог выполнить задание второго уровня, умеет выделять существенное и обнаруживать, и избавляться от «зашумлений».

Выполнение данного задания свидетельствует о том, что учащийся смог образ результата буквы представить в виде системы, одновременно разбивая букву на части, для того, чтобы передать этот образ в виде алгоритма для чертежника.

Задачи могут быть с «зашумлениями». Так как учащийся второго уровня может выделить существенные характеристики, даже когда имеются «зашумления», учащийся первого уровня «зашумления» преодолеть не сможет.

Для того, чтобы написать алгоритм для чертёжника, нужно понять общий способ действия написания печатных букв, представленных в алфавите:

– четко представлять образ букв;

– определить положение в пространстве будущей буквы (точка);

– определить в каком направлении начать писать букву;

– определить, написание буквы как системы и по частям (методом проб).

Если учащийся не смог выполнить задание данного уровня это означает, что ученик умеет только воспроизводить образцы. Ему нужно больше поразмышлять над общим способом действия, так как он не освоил усваивает те понятия и способы действия, на которых эти образцы, которые он умеет воспроизводить (если решает задачи первого уровня) основаны (недостаточное количество решенных задач второго уровня).

Конкретно в уровневых заданиях алгоритмической линии «Исполнитель–Чертёжник» понятие алгоритма осваивается как общий способ действия – алгоритм для человека и алгоритм для ЭВМ.

Выполнение задания третьего уровня свидетельствует о том, что учащийся способен не только видеть границы применения нового алгоритма, заданного ранее образцом применения, но и трансформировать этот алгоритм в соответствии с изменениями в материале.

Учащийся способен образ результата буквы (разбитый оптимально на элементы) сконструировать в алгоритм, чтобы чертежник смог написать букву оптимально. Разбитый оптимально образ результата – это значит, подобранный образ для алгоритма так, например, чтобы чертёжник как можно меньше выполнял команд. Помимо этого, учащийся умеет видеть общий способ действия, и на основе его способен разрабатывать общую часть кода в алгоритме для отдельно взятых элементов, удерживая букву как систему.

На данном этапе выполнения заданий, учащемуся можно предложить, - самому проверить правильно ли он написал код программы для исполнителя-чертежника, посредством программы «Чертежник».

В таких задачах может требоваться видоизменить способ в соответствии с новыми условиями.

Если учащийся не смог выполнить задание третьего уровня, это указывает на то, что предметные средства не стали его личным ресурсом, и он еще нуждается в какой-либо опоре. Понятие программирование не становится компетентным в его употреблении. Возможно, учащийся от невнимательности допустил ошибку в алгоритме, в таком случае рекомендуется вернуться к алгоритму.

Наличие у учащегося первого уровня означает, что он нуждается в образцах и постоянной помощи учителя, предмет он осваивает формально. Наличие второго и третьего уровней означает, что учащийся отошел от формального освоения предмета и способен к самостоятельному мышлению; он мысленно относится к предмету.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвящена разработке на основе модели индивидуального прогресса серии уровневых заданий алгоритмической линии «Исполнитель – Чертёжник», ориентированной на освоение понятия «программирование» учащимися 6 классов.

В процессе работы нами были разделены: деятельность программирования и обучение программированию. На основе анализа деятельности программирования выделены обязательные предметные действия, которые надлежит освоить ученику, чтобы быть компетентным в этой деятельности. Так, в понятии программирования мы выделили три ключевых этапа: формализация, алгоритмизация (процесс написания алгоритма) и кодирование (написание кода). Ряд авторов отмечают, что существенным этапом при освоении программирования является умение перевести действия с языка, понятного человеку, на язык, воспринимаемый машиной. Без понимания разницы между формальным исполнителем и человеком, этапы программирования невозможны. Например, чтобы робот умел брать предметы, нужно узнать, как это человек берет предмет рукой — как протягивается рука и как пальцы пытаются охватить предмет и понимать, что робот умеет исполнять только команды и если что-то упустить в командах, сам робот не предусматривает этого. Так, если пропустить команды для большого пальца, то он не сможет охватить предмет, а только сделает попытку.

Сделан анализ специфики серии заданий алгоритмической линии по математике, составленных на основе модели индивидуального прогресса. В диагностическом комплекте методики «Дельта» алгоритмическая линия представлена в серии уровневых заданий «следование инструкции». Серия «следование инструкции» связана с пониманием и реализацией алгоритмов, на которые опирается система математических процедур. Уровни применительно к серии «следование инструкции»:

- первый уровень ориентирован на опознание текста как инструкции к действию;
- второй уровень ориентирован на выделение существенного и адекватное применение для решения задачи алгоритма, данного в виде образца применения;
- третий уровень ориентирован на редактирование алгоритма в соответствии с изменением типа материала.

Реконструированы понятия «алгоритм» и «программирование». Программирование – процесс написания программы на языке программирования. Частью программирования является написание алгоритма – конечной последовательности действий, описывающей процесс преобразования объекта из начального в конечное, записанной с помощью точных и понятных исполнителю команд.

На основе обозначенного понимания разработаны уровневые задания по информатике, в основу которых положена уровневая модель индивидуального прогресса.

Материалы и выводы могут быть использованы в учебном процессе в образовательных учреждениях основного образования на уроках по информатике для основной школы как условие для достижения определенных предметных и метапредметных достижений.

Нашу гипотезу о характеристике уровневых заданий, направленных на освоение понятия «программирования» можно считать подтверждённой.

В работе были решены задачи, поставленные во введении, таким образом, была достигнута цель и решены задачи написания выпускной квалификационной работы, а именно были разработана серии уровневых заданий. Однако, она требует апробации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Босова, Л.Л. Информатика : учебник для 6 класса / Л.Л. Босова, А.Ю. Босова. – Москва : Бином. Лаборатория знаний, 2013. – 213 с.
2. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт – М. : ДМК Пресс, 2010. – 272 с.
3. Воронцов, А.Б. Современная дидактика – основа для проектирования практики школьного образования? / А.Б. Воронцов // Материалы научно-методической конференции / Современная дидактика и качество образования. – Красноярск, 2009. В 2 ч. Ч.1. С. 29-36.
4. Выготский, Л.С. Собрание сочинений : В 6-ти т. Т. 1. Вопросы теории и истории психологии / Л.С. Выготский – М. : Педагогика, 1982. – 488 с.
5. Выготский, Л.С. Собрание сочинений : В 6-ти т. Т. 2. Вопросы теории и истории психологии / Л.С. Выготский – М. : Педагогика, 1982. – 488 с.
6. Гальперин, П.Я. Введение в психологию : Учебное пособие для вузов. — 2-е изд. — М. : «Книжный дом «Университет», 2000. – 336 с.
7. Горленко, Н.М. Принципы обучения как руководство к действиям субъектов учебного процесса : проблемы определения / Н.М. Горленко // Материалы научно-методической конференции / Современная дидактика и качество образования. – Красноярск, 2009. В 2 ч. Ч.1. С. 99-105.
8. Д. Спольски. Джоэл о программировании / Спольски Д. – СПб : Символ-плюс, 2006. – 352 с.
9. Данова, Н.С. Роль программирования в школьном курсе информатики / Н.С. Данова, О. П. Пономарев // Психолого-педагогический журнал Гаудеамус. – Тамбов, 2015. №7. С. 191-196.
10. Долгова, Л.М. Пробные действия учащихся в пространстве инновационной школы / Л.М. Долгова // Вестник Томского государственного университета / Народное образование. Педагогика. – Томск, 2007. № 303. С. 191-193.

11. Ершов, А.П. Алгоритмический язык в школьном курсе основ информатики и вычислительной техники / А.П. Ершов. – Новосибирск : НГУ, 1985. – 26 с.

12. Ершов, А.П. Теоретическое программирование / Ю.Л. Ершов. – Новосибирск : НГУ, 1982. – 516 с.

13. Знаменская, О.В. Оценка-поддержка индивидуального прогресса учеников : методика «Дельта» : методическое пособие / О.В. Знаменская, Л.А. Рябинина, О.И. Свиридова. – Красноярск : Сибирский федеральный университет, 2014. – 110 с.

14. Ковалева, Т.М. Обучение как освоение нового / Т.М. Ковалева // Материалы научно-методической конференции / Современная дидактика и качество образования. – Красноярск, 2009. В 2 ч. Ч.1. С. 44-49.

15. Кормен, Т. Алгоритмы. Построение и анализ / Т. Кормен, Ч. Лейзер : изд. 3-е. – Вильямс : 2009. – С. 14.

16. Крафт, Н.Н. Самостоятельная работа как средство саморазвития студентов / Н.Н. Крафт // Вестник Адыгейского государственного университета / Народное образование. Педагогика. – 2006, №4.

17. Мониторинг индивидуального прогресса учебных действий школьников/ под ред. П.Г. Нежнова, Б.И. Хасана, Б.Д. Эльконина. – Красноярск : Печатный центр КПД, 2006. – 132 с.

18. Мониторинг индивидуального прогресса учебных действий школьников / О.В. Знаменская, О.С. Островерх, Л.А. Рябинина, Б.И. Хасан. – Красноярск, 2009. – 118 с.

19. Непейвода, Н.Н. Основы программирования [Электронный ресурс] / Н.Н. Непейвода, И.Н. Скопин. – Режим доступа : http://www.twirpx.com/file/199296/.

20. Нигматулина, Э.А. Условия формирования алгоритмической культуры студентов на основе информационного подхода / Э.А. Нигматулина // Вестник Красноярского государственного педагогического университета им. В.П. Астафьева. – 2011. – №1. – С. 82-86.

21. Основное общее образование : федеральный государственный образовательный стандарт [Электронный ресурс] : Справочная правовая система «КонсультантПлюс». – Режим доступа : http://www.consultant.ru.

22. Пейперт, С. Переворот в сознании : дети, компьютеры и плодотворные идеи / С. Пейперт – Москва : Педагогика, 1989. – 224 с.

23. Примерная основная образовательная программа образовательного учреждения. Основная школа / [сост. Е. С. Савинов]. — М. : Просвещение, 2011.

24. Рогов, А.Ю. Технологии программирования : учебное пособие / А.Ю. Рогов, О.В. Простиненко. – СПб : СПбГТИ(ТУ), 2010. – 112 с.

25. Стась А. Н. Развитие алгоритмического мышления в процессе обучения будущих учителей информатики / А. Н. Стась, Н.Ф. Долганова // Вестник Томского государственного педагогического университета. – 2012. - №7. С. 241-244.

26. Фаулер, М. Рефакторинг. Улучшение существующего кода / М. Фаулер – СПб : Символик-Плюс, 2003. – 432 с.

27. Федеральный государственный образовательный стандарт основного общего образования [Электронный ресурс] : приказ Минобрнауки России от 17 декабря 2010 г. № 1897 // Справочно-правовая система «КонсультантПлюс». Режим доступа : http://www.consultant.ru.

28. Фейнман, Р. Вы, конечно, шутите, Мистер Фейнман! / Р. Фейнман – М. : Астрель, 2012. – 188 с.

29. Фрумин И.Д. Две идеологии в управлении образованием : между контролем и поддержкой (на примере вопроса об оценке качества образования) // Политика, основанная на знании : опыт Англии и Шотландии / под ред. И.А. Вальдмана. – М. : Университетская книга, 2006.

30. Хасанова, С.Л. Курс «История информатики» в системе образования / С.Л. Хасанова, Е.А. Рассказова // Фундаментальные исследования / Народное образование. Педагогика. – 2014, №9-4. С. 747-751.

ПРИЛОЖЕНИЕ А
Серия уровневых заданий
алгоритмической линии «Исполнитель-Чертёжник»

Алгоритм создается для того, чтобы кто-то его исполнял. Исполнителем, может быть сам разработчик, другой человек, или компьютер.

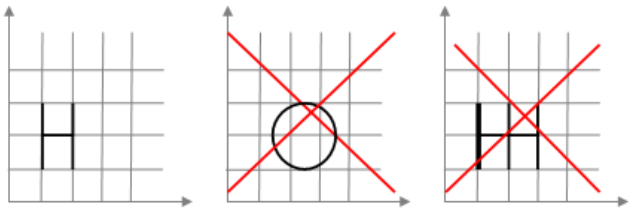
Предположим, что одна компания сделала тебе заказ: научить исполнителя чертежника рисовать буквы. Написать букву самому – действие, которое человек совершает автоматически, так как он уже умеет читать, писать. Объяснить принцип написания букв исполнителю-человеку задача не из легких, а уж тем более научить исполнителя-компьютера.

Открой программу КУМИР, перейди к исполнителю чертёжнику (с его командами можешь ознакомиться в меню «Инфо» => «Алгоритмы»).

Компьютер отличается от человека. У компьютера свой язык и свои функции и нужно пользоваться ими при написании алгоритма. Так как компьютер действует пошагово, то есть выполняет каждое действие алгоритма, тебе нужно разобрать свое действие на кусочки и превратить в команды для исполнителя.

Условия, при которых исполнитель-чертежник будет писать буквы на координатной плоскости:

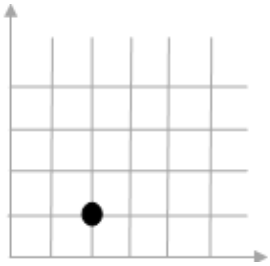
- линии прямые;
- нет наклонов и закруглений — линии рисуются по координатной сетке;
- более одного раза по одной и той же линии проходить нельзя;
- в начале рисования перо по умолчанию находится в точке (0;0);
- изначально перо поднято;
- рисование происходит на координатной плоскости;



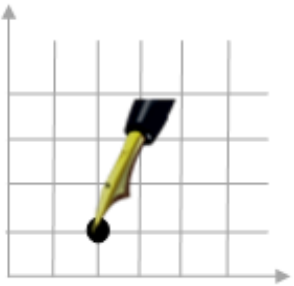
Разбери пошаговое написание буквы чертежником:

Программа «Буква Г»

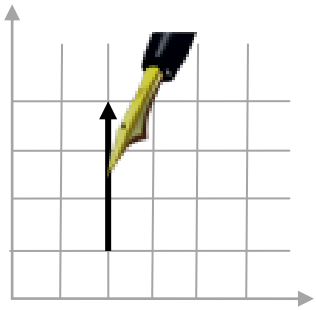
Сместиться в точку (2;1)



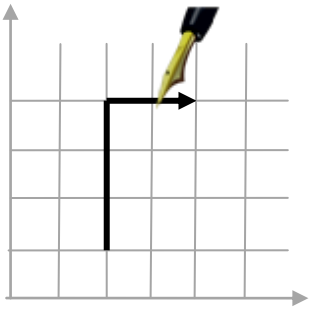
Опустить перо



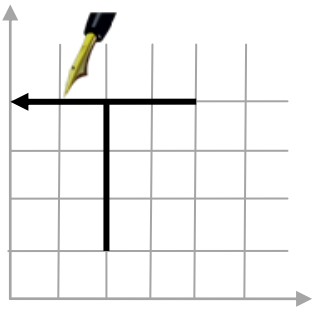
Сместиться в точку (2;4)



Сместиться в точку (4;4)



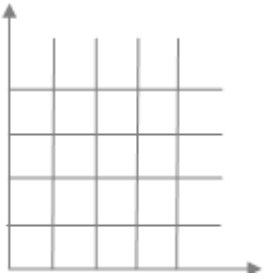
Алгоритм «Буква Г» может стать элементом для алгоритма «Буква Т».
В данном случае остается только добавить несколько команд к уже готовому алгоритму:
Поднять перо
Сместиться в точку (2;4)
Опустить перо
Сместиться в точку (0;4)



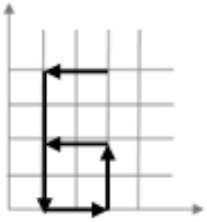
Первый уровень

Задание 1.1. Выполни алгоритм написания буквы, будучи чертежником:

Сместиться в точку (3;1)
Опустить перо
Сместиться в точку (3;4)
Сместиться в точку (5;4)
Сместиться в точку (5;1)



Задание 1.2. Напиши программу «Буква Б», которая изображена на координатной плоскости:



Двухуровневые задания (первый и второй уровни)

Задание 1.2.1. Нарисуй элементы буквы Ю. какие буквы можно рисовать из этих элементов? Напиши алгоритм для чертёжника где часть алгоритма будет про один из элементов буквы Ю. Притом эта часть буквы Ю должна применяться как минимум к двум другим буквам (а другая часть другие буквы….):

Задание 1.2.2. Написан алгоритм для исполнителя-человека, напиши его для чертёжника. Отличаются ли алгоритм для человека и для исполнителя? Чем?

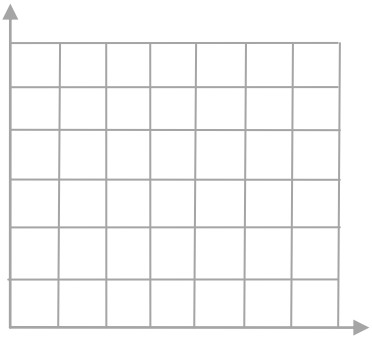
Алгоритм:
Поставь ручку на точку (1;0)
Перемести ручку, не отрывая от листа в точку (1;3)
Подними ручку
Поставь ручку в точку (2;3)
Перемести ручку, не отрывая от листа в точку (2;2)
Перемести ручку, не отрывая от листа в точку (1;2)
Подними ручку
Поставь ручку в точку (1;1)
Перемести ручку, не отрывая от листа в точку (2;1)
Перемести ручку, не отрывая от листа в точку (2;0)

Второй уровень

Задание 2.1. Узнай, про какую букву написан алгоритм и допиши пропуски в нем

Сместиться в точку (1,1)
Опустить перо
Сместиться в точку (1,5)
Поднять перо
Сместиться в точку (1, 3)
Опустить перо
Сместиться в точку (6,3)
_____
Сместиться в точку (4,1)
_____
Сместиться в точку (4,5)

_____
_____
_____
_____



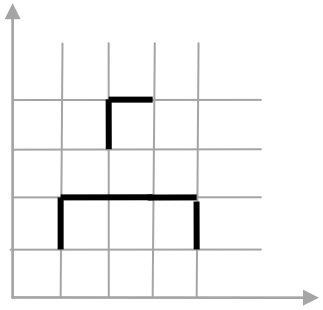
Какая буква у тебя получилась? _____

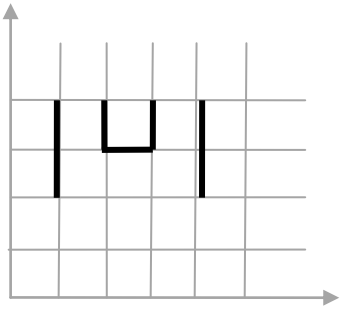
Сможет ли чертёжник выполнить алгоритм с пропусками, чтобы получилась та буква, которая вышла у тебя? Обосную.

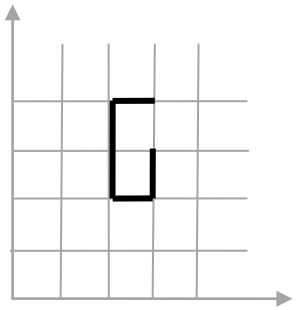
_____
_____
_____
_____

Для чего этот алгоритм?

Задание 2.2. Какое слово изображено на координатных плоскостях? Дорисуй элементы букв и напиши для них алгоритм:







_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

С каких элементов ты начал писать алгоритм? Что будет если писать с других?

Использовал ли ты алгоритмы букв, которые были в прошлых заданиях?

_____
_____
_____
_____
_____

Задание 3 уровня

А знаешь ли ты, что программисты не всегда пишут алгоритм «с нуля»? Программист пишет много программ, и его база алгоритмов постоянно пополняется. Если есть база алгоритмов, то в ней найдутся и отдельные элементы для других программ. Значит, в написании алгоритма «с нуля» нет необходимости.

Задание 3.1. Нужно научить чертежника рисовать буквы Р, А, Ф. Что общего между ними? (У них есть общие элементы?) Напиши алгоритм для чертежника так, чтобы, его можно было использовать для всех этих букв. Добавь к готовому алгоритму дополнительные шаги (элементы) для каждых букв по отдельности.

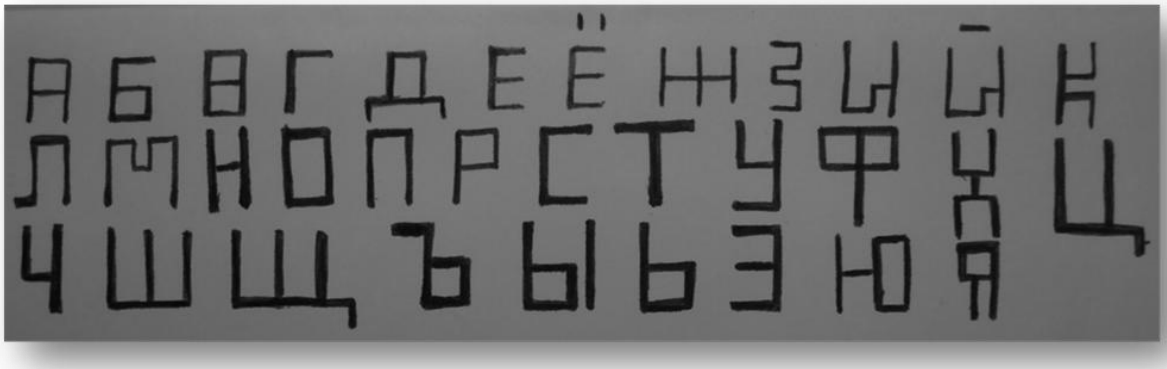
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Задание 3.2. Напиши программу, которая рисовала бы буквы с помощью фрезеровочного аппарата. Составь базу элементарных элементов для написания букв.

Задание 3.3. Код из среды программирования Лого переделай под среду программирования КУМИР. Изначально черепашка смотрит вверх. Обозначения кодов в Лого: PD (перо опустить), RIGHT (поворот направо), LEFT (поворот налево), FORWARD <число> (движение вперед на указанное число шагов).

```
ТО Буква
PD
RIGHT
FORWARD <1>
LEFT
FORWARD <3>
RIGHT
FORWARD <1>
RIGHT
FORWARD <3>
END
```

Алфавит букв для исполнителя-чертежника:


А Б В Г Д Е Ё Ж З И Й К
Л М Н О П Р С Т У Ф Х Ц
Ч Ш Щ Ъ Ы Ь Э Ю Я

ПРИЛОЖЕНИЕ Б

Как выглядит код для буквы «К» в среде «КУМИР - 1.9.0», «Чертежник»