МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В.П. АСТАФЬЕВА»
(КГПУ им. В.П. Астафьева)

Институт математики физики и информатики

Кафедра информатики и информационных технологий в образовании

Мальцева Кристина Дмитриевна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Реализация технологии И.Р. Дединского при обучении программированию в основной школе

Направление подготовки: 44.03.05 Педагогическое образование

Профиль образовательной программы: Физика и информатика

ДОПУСКАЮ К ЗАЩИТЕ

Зав. кафедрой ИиИТО
д-р пед. наук., проф.
_____ Н.И. Пак
20.06.2019 г
(дата, подпись)

Руководитель
канд. пед. наук, доц. каф. ИиИТО
_____ Л.М. Ивкина

Дата защиты «22» июня 2019

Обучающийся
_____ К.Д. Мальцева

Оценка хорошо

Красноярск 2019

Оглавление

Введение

За последние десятки лет происходит стремительная информатизация всех сторон жизни общества и всех сфер производственной деятельности.

Информатика становится одной из фундаментальных областей научного знания, которая изучает информационные процессы, методы, а так же средства обработки информации. Эта дисциплина развивается стремительно быстро, она тесно связана с использованием информационных технологий, а область ее применения в жизни с каждым днем расширяется все больше.

Выпускникам школ необходимо обладать достаточными знаниями и навыками для эффективного использования современных информационных технологий в своей дальнейшей деятельности.

Главный вопрос в системе обучения информатики остаетсяв том, необходимо ли выпускнику уметь решать возникающие задачи с помощью программирования, или достаточным является освоение пользовательских технологий и навыки поиска готовых решений.

Когда в школьной программе только появился курс информатики в основе обучения было изучение программирования. Это было связано с дефицитом компьютеров.

По мере развития системы программирования, а также с появлением и широким распространением персональных компьютеров стало возможным решение практико-ориентированных задач без программирования.

Изучение информатики, так же как и других предметов в школьной программе необходимо для дальнейшей профессиональной деятельности учеников, для этого им необходимы лишь пользовательские навыки. Подход «обучение через программирование» устарел и перестал учитывать интересы учащихся.

Главным аргументом против обучения программированию в школе в последнее время становится его сложность и узкая специализация.

В разделе «Алгоритмизация и программирование» школьного курса учащимся предлагается для решения множество различных задач. Эти задачи обычно математические, результатом которых являются цифры - решения квадратного уравнения на черном экране, что совершенно не наглядно. У школьников создается впечатление о программировании как о скучном, сложном и устаревшем занятии. Другое дело, если результатом программирования будут трехмерные графические игры с нелинейным сюжетом, в которые дети играют дома.

Интерес к изучению снижается, и достаточно сложно объяснить, что в основе игр лежит та же математика, те же алгоритмические конструкции. Таким образом, с точки зрения методики обучения информатике существует потребность в сквозной наглядной практической задаче, обеспечивающей связь всех необходимых для изучения в школе аспектов программирования, упрощающей понимание объектно-ориентированного программирования с помощью наглядности и поддерживающей на высоком уровне познавательную мотивацию.

Еще одной из важнейших целей обучения информатике в школе становится решение совершенно новой, появившейся в последние годы, воспитательной задачи: уменьшение непродуктивного самостоятельного использования компьютера, перенаправление внимания учащихся с задач, не связанных с обучением и воспитанием (многочасовые компьютерные игры, чаты, социальные сети и т.д.), на продуктивное и эффективное использование компьютера, мобильных компьютерных устройств, информационных и телекоммуникационных технологий.

Практика показывает, что существуют динамические компьютерные игры (то есть игры, насыщенные движением, действием, изменением объектов и их свойств с течением времени), которые школьники могутразрабатывать в процессе обучения. Существует активный интерес школьников к такой деятельности. Разработка динамических игр, несложная

с точки зрения программирования, может внести серьезный вклад в повышение мотивации к учению, преодоление когнитивных затруднений, интеллектуальное развитие школьников.

Существующие исследования по использованию компьютерных игр в обучении школьников затрагивают, в основном, потенциал компьютерных игр как средства обучения и мотивации, но не рассматривают эти игры как объект разработки, то есть не существует научно обоснованной методики обучения программированию, основанной на создании школьниками динамических компьютерных игр.

Таким образом, выбор темы «Разработка технологии И.Р. Дединского при обучении программирования в основной школе» является актуальной и основана на решении сквозной проектной задачи по созданию компьютерной игры.

Цель исследования – описать методические рекомендации обучения программированию с использованием технологии Дединского И.Р.

Объект исследования: процесс обучения программированию основной школы.

Предмет исследования: использование технологии Дединского И.Р. обучения программированию основной школы в курсе информатики.

Цель и предмет определили постановку и необходимость решения следующих задач исследования:

1. Провести обзор и анализ состояния обучения программированию в основной школе для выделения проблем;
2. Провести обзор подходов обучения программированию;
3. Описать особенности технологии Дединского И.Р.;
4. Разработать методическое планирование уроков и рекомендации обучения программированию с использованием технологии Дединского И.Р.;
5. Разработать подробные конспекты обучения программированию с использованием технологии Дединского И.Р.

Методы исследования:

- теоретические: системный анализ отечественной и зарубежной психолого-педагогической, научно-методической литературы по философии, педагогике, психологии; критический анализ существующих подходов к обучению информатике и программированию, а также использованию электронных ресурсов по рассматриваемой проблеме;

- эмпирические: обобщение опыта преподавания информатики; анализ содержания учебных программ, планов, пособий, диссертаций, материалов конференций по вопросам обучения программированию в школе.

Структура выпускной квалификационной работы состоит из введения, двух глав, заключения, библиографического списка и приложений.

Глава 1. Теоретические основыобучения программированию в основной школе.

1.1 Проблемы обучения программированию в основной школе.

Информатика – особый предмет в школьной программе, достаточно «молодой», не обремененный пока еще изобилием «официальных, стандартных» методов и методик преподавания. Это объясняется многими причинами: разнообразием имеющейся в школах вычислительной техники, разнообразием имеющегося программного обеспечения, разным количеством часов, выделенных на преподавание и т.д.

В государственном стандарте по информатике отмечается, что в результате изучения информатики и ИКТ на базовом уровне ученик в области программирования должен:

1. знать основные свойства алгоритмов, типы алгоритмических конструкций: следование, ветвление, цикл, понятие вспомогательного алгоритма;
2. уметь использовать алгоритмические конструкции, выполнять и строить простые алгоритмы, выполнять базовые операции над объектами: цепочками символов, числами, списками, деревьями;
3. использовать приобретенные знания и умения в практической деятельности и повседневной жизни при выполнении индивидуальных и коллективных проектов, в учебной деятельности, в дальнейшем освоении профессий.

Данные знания, умения и навыки формируются при изучении темы «Алгоритмизация и программирование». [36]

В чистом виде программирование интересует небольшую категорию людей. Теория алгоритмов или программирование - это чересчур специальные вещи на сегодняшний день, когда компьютеры продаются в супермаркетах рядом с телевизорами и DVD-проигрывателями. Сегодня простому пользователю программировать не нужно, хотя еще недавно такого

просто не могло быть. Поэтому достаточно часто слышишь вопрос: Зачем всех подряд учить программированию, если это реально нужно нескольким ученикам собравшимся в технический вуз причем на соответствующие специальности? Большинство людей, использующих компьютеры, не пишут своих собственных программ, и им практически вообще не требуется знать программирование. Если рассуждать, что пользователю достаточно знать только "три кнопки", и на информатике в школе нужно давать только пользовательский курс, то по аналогии можно утверждать, что на математике нужно учить пользоваться калькулятором, зачем школьникам эти логарифмы, производные, интегралы, если есть компьютер, калькулятор, да и таблицу умножения знать не нужно, главное уметь нажимать кнопки на элементарном уровне.

Изучать программирование нужно. Изучая программирование, ученики лучше понимают сущность работы компьютера, его возможности и ограничения. Программирование помогает школьникам развивать навыки мышления, а также привычку к аккуратной работе. Нет лучшего способа развить логику мышления, точность формулировок, аккуратность, чем программирование. Ряд школьных предметов вообще не связан с какой – либо стороной мышления, а настроен на усложнение знаний в конкретной области, на развитие кругозора учащихся. Информатика развивает специфический стиль мышления.

Программирование - это такая основополагающая вещь которую хоть в малой степени, но надо знать всем. Умение строить алгоритмы и программировать их на алгоритмических языках отлично развивает логическое мышление.

Если человек понимает как создаются программные средства и умеет хотя бы на элементарном уровне программировать, то он лучше будет понимать принцип работы любого прикладного ПО, и в случае ошибки или в случае нестандартной ситуации будет знать что делать и сможет справиться с

ней. При построении обучения учащихся теме «Алгоритмизация и программирование» каждый учитель информатики сталкивается с огромным количеством вопросов: как построить изложение материала, какие использовать методические разработки, в какой форме проводить занятия, какие составить практические задания, какой материал использовать учащимся при изучении и другие. Все эти вопросы возникают из-за отсутствия четко и в полном объеме изложенных учебно-методических материалов и учебников для изучения данной темы.

На изучение информатики в 7 - 9 классах отводится 138 часов: 35 часов в 7 классе (1 час в неделю), 35 часов в 8 классе (1 час в неделю), 68 часов в 9 классе (2 часа в неделю). В 9 классе 1 час в неделю отводится из школьного резерва. Из этого количества часов отводится 19 часов на изучение темы «Алгоритмы и исполнители», причем подразумевается изучение формальных исполнителей алгоритмов. [36] Среднее (полное) общее образование базового уровня включает в себя 34 часа в 10 классе, 34 часа в 11 классе. Программа рассчитана на 1 ч в неделю. В данное количество часов не входят часы на изучение темы «Алгоритмизация и программирование» [46]. Предполагается, что учитель будет использовать язык программирования во время решения задач при изучении других тем.

Таким образом, объём часов на изучение темы «Алгоритмизация и программирование» не дает возможности в полной мере изучить данную тему в школьном курсе. В этом и заключается несоответствие выделяемого количества часов на изучение данной темы с объемом рассматриваемого материала за данное количество часов, и в этом выражается несоответствие к требованиям выпускника по форме единого государственного экзамена.

И вот тут и проблема как заинтересовать учеников программированием, как научить понимать и решать задачи. Из опыта работы каждый из учителей информатики может сказать, что наибольший интерес у учащихся вызывает графика, как у 8, так и у 11 классов, при

работе с которой на экране виден красочный результат выполнения программы. Порой трудно объяснить учащимся, что все, что выполняет компьютер, - это программы. В связи с этим возникла идея давать большую часть материала, используя графику. Отсюда вытекает и изменение учебного плана занятий: на первое место можно сразу поставить изучение графических операторов, а затем уже с их помощью объяснять (по возможности) весь остальной материал. Ученику проще увидеть и сделать, чем пытаться понять, что так происходит на самом деле. Используя подобные программы, можно проиллюстрировать работу всех основных операторов, а если использовать элементы блок-схем, то и продемонстрировать работу алгоритмов без использования операторов.

Применяя такой опыт в работе, можно наблюдать у учащихся некий элемент соревновательности, желание сделать лучше и красивее своего товарища. Данный тип уроков приносит результаты как в младших, так и в старших классах.

1.2 Методические подходы к обучению программированию в основной школе.

В дидактике термин «подход» определяется как совокупность принципов, определяющих стратегию обучения, при этом каждый принцип регулирует разрешение конкретных противоречий, возникающих в процессе обучения, а их взаимодействие – разрешение основных его противоречий [1]. Таким образом, исходным содержанием понятия «подход» является определенная идея, концепция, совокупность принципов, обусловливающих организацию того или иного явления, процесса, например, процесса обучения программированию.

В настоящее время существует ряд российских и западных исследований, посвященных разработке методических подходов к обучению программированию, среди которых можно выделить: системный подход

(И.О. Одинцов и др.), деятельностный подход (Е.А. Ракитина и др.), когнитивный подход (J. Reinfelds и др.), проблемный подход (Е.В. Касьянова, К.Ю. Поляков и др.), семиотический подход (Р. Andersen, К. И. Баумане, Н.И. Рыжова и др.).[4]

Проведем сравнительный анализ обозначенных методических подходов, уделяя особое внимание системному и деятельностному подходу в связи с тем, что согласно ФГОС, они являются основополагающими в общеобразовательной школе.

В основе системного подхода лежит рассмотрение объектов как систем. Он ориентирует исследование на раскрытие целостности объекта, на выявление многообразных типов связей в нем и сведение их в единую теоретическую картину [4]. Как показал анализ научной литературы, основными категориями системного подхода является система, структура и среда.

Система – множество элементов, находящихся в отношениях и связях друг с другом, образующих определенную целостность, единство [22].

Также исследователи системного подхода (В.С. Леднев, В.Г. Афанасьев и др.), подчеркивают, что система – это совокупность объектов, взаимодействие которых вызывает появление новых интегративных качеств, не свойственных отдельно взятым образующим систему компонентам [28].

Структура – совокупность устойчивых связей между элементами объекта, которые обеспечивают его целостность и тождественность самому себе, то есть сохранение основных свойств при различных внешних и внутренних изменениях [27].

Структурность является главным свойством реальности – «всё как-то устроено: все объекты реальности из чего-то состоят, имеют составляющие их части и в тоже время, сколь бы сложными они ни были, сами являются составными частями чего-то более сложного. И так до бесконечности как в

сторону увеличения масс и масштабов, так и в сторону их уменьшения. Притом всё находится в движении – всё течёт, всё изменяется» [6].

Поэтому любые объекты реальности требуют двоякого их рассмотрения: в статике и динамике.

Среда – совокупность всех объектов/субъектов, не входящих в систему, изменение свойств и/или поведение которых влияет на изучаемую систему, а также тех объектов/субъектов, чьи свойства и/или поведение которых меняются в зависимости от поведения системы [9].

Следует отметить, что системный подход к познанию и преобразованию любого объекта является ведущим общенаучным подходом. Применение же данного подхода в обучении программированию позволяет выявить такой компонент как система обучения программированию со всеми ее характеристиками: целостность, связь, структура и организация, уровни системы и их иерархия, управление, самоорганизация системы, ее функционирование и развитие.

Идеям и принципам применения системного подхода в обучении информатике и программированию посвящены работы И.О. Одинцова, Н.В. Макаровой и др.

Основная идея системного подхода в контексте обучения программированию состоит в рассмотрении учебных задач в тесной взаимосвязи со средствами, методами программирования и в целом с технологическим процессом. Изучение каждого из этих направлений должно происходить не изолировано, а в тесных связях и зависимостях между ними.

Учитывая межпредметные связи программирования с другими школьными предметами и областями научного знания, можно утверждать, что системный подход является основополагающим в его преподавании, и другие методические подходы, которые могут быть применены к обучению программированию, должны дополнять и расширять его основные идеи и принципы, а не противоречить ему.

Деятельностный подход в обучении основан на «принципиальном положении о том, что психика человека неразрывно связана с его деятельностью и деятельностью обусловлена». Категория деятельности является основной в этом подходе. При этом деятельность понимается как преднамеренная активность человека, проявляемая в процессе его взаимодействия с окружающим миром, и это взаимодействие заключается в решении жизненно важных задач, определяющих существование и развитие человека [20].

В обучении информатике и программированию концепция деятельностного подхода предполагает:

– формирование готовности школьника на активное, адекватное и эффективное применение информационных и телекоммуникационных технологий в процессе обучения информатике,

– выявление и формирование творческой индивидуальности школьника,

– развитие его будущих профессиональных взглядов [24].

В настоящее время системный и деятельностный подходы интегрированы и положены в основу ФГОС второго поколения. Системно-деятельностный подход направлен на обеспечение реализации идеи непрерывного образования на уровне школы при условии сформированности у обучающихся универсальных учебных действий (УУД): регулятивных, познавательных, коммуникативных и личностных.

Под универсальными учебными действиями, в широком значении понимают «умение учиться, то есть способность субъекта к саморазвитию и самосовершенствованию путем сознательного и активного присвоения нового социального опыта» [35]. В более узком (собственно психологическом) значении термин «универсальные учебные действия» определяется как совокупность способов действия учащегося (а также связанных с ними навыков учебной работы), обеспечивающих его

способность к самостоятельному усвоению новых знаний и умений, включая организацию этого процесса [26].

В повышении эффективности изучения программирования приоритетное место занимают познавательные УУД [36]. Познавательные универсальные учебные действия формируют систему способов познания окружающего мира, построения самостоятельного процесса поиска, исследования и совокупность операций по обработке, систематизации, обобщению и использованию полученной информации.

В блоке познавательных УУД выделяют общеучебные действия, знаково-символические, логические действия и действия постановки и решения проблем.

Сформированность общеучебных универсальных действий позволяет учащимся:

– самостоятельно выделять и формулировать проблему, ставить цели, задачи, находить необходимую информацию, осуществлять поиск решения поставленной задачи;

– структурировать знания;

– осознанно использовать термины программирования в речевых высказываниях в различных предметных областях;

– выбирать наиболее эффективный способ решения задачи по программированию в зависимости от конкретных условий.

Особую роль для повышения эффективности обучения программированию играют знаково-символические действия, которые обеспечивают конкретные способы преобразования учебного материала, представляют действия моделирования, выполняющие функцию отображения учебного материала; выделение существенного, отрыва от конкретных ситуативных значений, формирования обобщенных знаний. К ним относят:

– замещение – перенос существенных свойств объекта на модель;

– кодирование – написание алгоритма на понятном для исполнителя языке;

– схематизацию – представление результатов двух предыдущих этапов в наглядной, удобной для восприятия форме.

– моделирование – преобразование объекта из чувственной формы в модель (пространственно-графическую или знаково-символическую), где выделены существенные характеристики объекта

При выборе методов и способов решения задач, определении оптимальных алгоритмов, существенную роль играют логические универсальные действия. Уровень их развития влияет на выполнение учащимися действий по переводу записи алгоритма решения задачи из одной знаковой системы в другую. К этим действиям относятся:

– анализ объектов с целью выделения признаков (существенных, несущественных);

– синтез – составление целого из частей, в том числе самостоятельное достраивание с восполнением недостающих компонентов;

– выбор оснований и критериев для сравнения, сериации, классификации объектов;

– подведение под понятие, выведение следствий.

Многообразие и емкость содержания каждого из перечисленных учебных действий создают необходимые предпосылки успешного решения задач по программированию. Учитель должен быть уверен, что учащиеся владеют всей необходимой системой действий, составляющих умение программировать.

Однако в процессе обучения программированию необходимо использование не только общего системно-деятельностного подхода, так как он не учитывает выделенных нами особенностей программирования: высокую степень абстрактности базовых понятий программирования, метапредметность и мультимодальность учебного материла. Следовательно,

обозначенный во ФГОС системно-деятельностный подход должен быть дополнен идеями и принципами других методических подходов.

Как показал анализ научной и методической литературы, учебников по информатике в школьной практике в настоящее время распространены когнитивный и проблемный подходы к обучению программированию.

Когнитивный подход основывается на таксономии Б. Блума и предполагает овладение учащимися операциями программирования, начиная с элементарных, постепенно продвигаясь к сложным (рис. 1).

Основная цель когнитивного подхода – развитие когнитивных качеств обучающихся: быстроты и экономичности мышления, способности к решению нестандартных проблем, готовности воспринимать и анализировать противоречивую информацию. Обобщенным показателем уровня развития перечисленных качеств является интеллект – относительно устойчивая структура умственных способностей индивида [34].

В основные задачи обучения программированию на основе когнитивного подхода, помимо развития интеллекта, входит передача знаний учащимся. Однако это должны быть не формальные, а действенные знания, дающие возможность широкого и разностороннего оперирования ими.

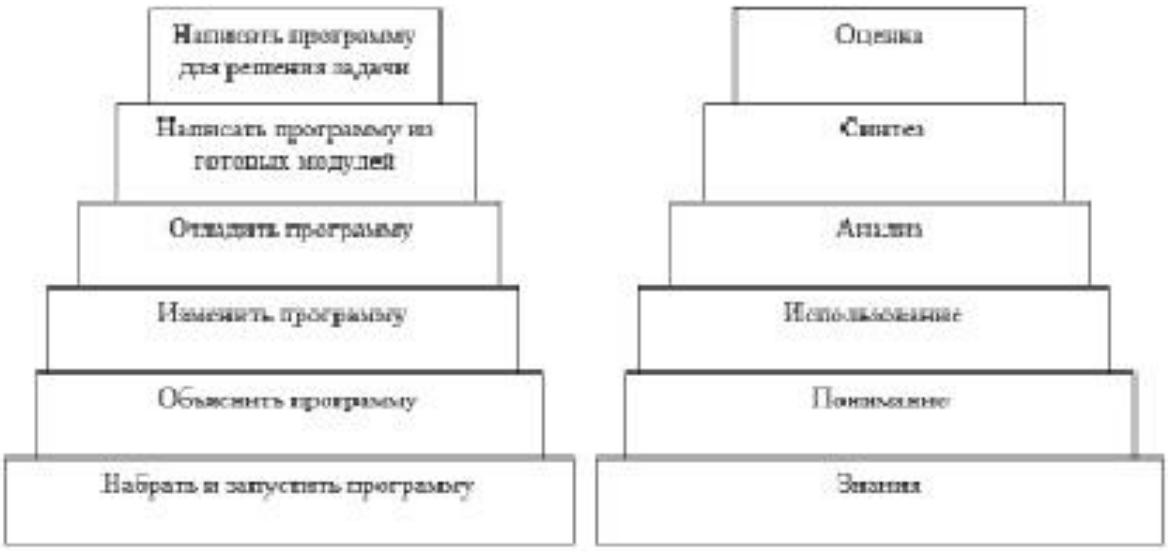

Рис. 1. Схема модели когнитивной таксономии целей при обучении программированию

Анализ учебной литературы по информатике и ИКТ для 7-9 классов показывает, что когнитивный подход к обучению программированию является основным. В рекомендованных Министерством образования и науки РФ учебниках по информатике [22] в качестве самостоятельной работы учащимся предлагаются, в основном, задачи трех первых уровней: на знание синтаксиса языка программирования и умение работать в выбранной среде, на объяснение программы и ее модификацию в соответствии с изменившейся задачей. Наиболее полно все типы учебных задач рассмотрены в учебнике И.Г. Семакина. Так, учащимся предлагается объяснить действие набранной программы, проанализировать ее действие при разных входных параметрах. Следует также отметить, что автор накладывает некоторые ограничения на решение задач, предлагает учащимся решить одну задачу несколькими способами. Данные приемы способствуют развитию алгоритмического мышления учащихся.

Реализация когнитивного подхода возможна с помощью метода демонстрационных примеров, метода ключевых задач, метода раскрутки задач и др. Метод демонстрационных примеров, предложенный в исследовании М.В. Швецкого [24], основан на идее Н.О. Вирта «...программирование представляет собой обширную и разнообразную деятельность, часто требующую сложной умственной работы. Ошибочно считать, что ее можно свести лишь к использованию готовых рецептов. В качестве метода обучения остается тщательный выбор и рассмотрение характерных примеров» [25]. Метод демонстрационных примеров имеет определенные преимущества: позволяет научить учеников чтению программ, использовать ключевые задачи при программировании других задач и др.

Однако когнитивный подход, по мнению И.А. Бабушкиной, В.Ю. Нефедовой, А.Ю. Петухова и др., требует значительных временных затрат и предполагает решение большого числа однотипных вычислительных задач, которое приводит к снижению мотивации [26, 27, 28]. Следует отметить

также, что при обучении программированию на основе когнитивного подхода, несмотря на работу учащихся с разными знаковыми системами, нет целенаправленного развития их знаково-символических действий.

Проблемный подход предполагает, что весь курс программирования строится на основе системы специально разработанных «проблемных» задач или ситуаций, содержание которых интересно для учащихся и, по возможности, связано с их будущей профессиональной деятельностью. Решая такого вида задачи, ученики расширяют свой опыт программирования.

Схема проблемной модели обучения программированию представлена на рисунке 2.

Рис. 2. Схема проблемной модели обучения программированию.

Наиболее четко и последовательно понятие проблемной ситуации представлено психологом А.М. Матюшкиным, который выделяет следующие компоненты:

- неизвестное достигаемое знание или способ действия;
- познавательная потребность, побуждающая человека к интеллектуальной деятельности;
- интеллектуальные возможности человека, включающие его творческие способности и прошлый опыт.

Проблемному подходу соответствуют такие методы обучения, как проблемное изложение, исследовательский метод, эвристический метод и др.

По сравнению с когнитивным подходом в проблемном обучении программированию подчеркивается понимание процесса изучения учебного материала с точки зрения знаний и личного опыта учащихся. Как отмечает И. Н. Скопин, при изучении программирования на основе проблемного подхода «…изучение основ курса становится мотивированным, а не догматически преподнесенным ученикам как непререкаемая истина».

С нашей точки зрения, в полном объеме проблемный подход не может быть использован на первом этапе обучения программированию, поскольку опирается на опыт деятельности в данном направлении у учащихся. Однако элементы проблемных ситуаций при обучении программированию успешно применяются и в начале обучения.

Обобщая вышесказанное, можно сделать вывод, что проблема изучения специфичной работы с новыми, не используемыми в естественном языке знаками, характерной для программирования, остается недостаточно исследованной. В связи с этим, как показывает анализ, при обучении программированию до сих пор возникают значительные трудности. Таким образом, в обучении программированию возникает проблема учета закономерностей взаимодействия человека со знаками и знаковыми системами, рассматриваемых семиотикой.

Семиотический подход к обучению программированию. В настоящее время все большую популярность в обучении информатике приобретает семиотический подход. Его основа разрабатывалась в философии и лингвистике (Ч.С. Пирс, Ч. Морис, Ф. де Соссюр и др.). Однако в связи с тем, что любая предметная область может быть описана как система знаков, а «...все информационные процессы представляют собой процесс прохождения информации по цепочке – иерархии кодов внутри некоторой системы…, т.е. процесс перекодирования (перевода) информации», семиотический подход постепенно становится общеметодологическим и развивается как в рамках

специальной науки о знаках – семиотике, так и междисциплинарных областях знаний (теория коммуникаций, кибернетика и др.).

Целесообразность применения семиотического подхода к обучению программированию может быть обоснована тем, что язык программирования изначально имеет знаковую природу. Как всякий искусственный язык он имеет «повторный перевод», другими словами, знания о мире формализуются, «переводятся» на обычный язык, а затем находят свое отражение в алгоритмических конструкциях, встроенных функциях и др., что влечет за собой постепенное развитие языка программирования.

Применение семиотического подхода в информатике рассматривается О.Ф. Брыксиной, Н.А. Кургановой, К.Р. Пиотровской, В.И. Фоминым и др. Так, О.Ф. Брыксина рассматривает семиотический подход к формированию ИКТ-компетентности выпускника ВУЗа [35]. В. И. Фомин на основе данного подхода рассматривает профессиональную компетентность специалиста через умения оперировать различными знаковыми системами в контексте задач предметной области [36]. В диссертационном исследовании Н. А. Кургановой рассматривается семиотический подход к изучению содержательной линии «Формализация и моделирование» школьного курса информатики и развитию знаково-символической деятельности учащихся [29].

К основным категориям семиотического подхода относят знак, символ и образ.

Сущность семиотического подхода к обучению программированию заключается в целенаправленном развитии у учащихся знаково-символических действий (замещения, кодирования, схематизации, моделирования). При этом учитывается, что обучающийся, уже знаком с некоторыми знаковыми системами (русским языком, дорожными знаками, позиционными системами счисления) и имеющиеся у него знания, опыт

деятельности можно эффективно использовать при изучении особенностей языков программирования как знаковых систем.

Под замещением понимают самый простой уровень знаково-символической деятельности, когда функции (признаки, свойства) замещаемого объекта переносятся на знаково-символическое средство. Так, при решении учебных задач учащимся нужно определить входные и выходные данные, на основе которых определить переменные, их тип и свойства.

В кодировании знаково-символические средства выполняют коммуникативную функцию, в случае решения задач средствами программирования: между учеником и компьютером. Ученику следует выделить алгоритмические конструкции: командные слова, встроенные функции, а также подпрограммы, которые необходимы для решения задачи.

В схематизации знаково-символические средства выполняют ориентировочную роль, заключающуюся в структурировании реальности, выявлении связей между явлениями. В контексте обучения программированию схемы используются как средства активной наглядности (выступают в функции материализации), например, в качестве наглядной опоры плана деятельности по созданию программного кода. Схема может быть представлена учащимся в удобной для него форме (в виде блок-схемы, псевдокода, плана решения задачи и др.) и вмещать в себя результаты предыдущих этапов.

После создания схемы учащийся переходит к следующему этапу составления программы: ее записи на языке программирования с соблюдением синтаксических и семантических правил.

Потребность в применении семиотического подхода к обучению программированию и детализации процесса перевода учащимися алгоритма решения с естественного языка на язык программирования на основе выделения подэтапов (замещение, кодирование, схематизация,

моделирование) особенно ощутима при решении учебных задач, где используются «новые» непривычные рассуждения, из которых сразу же выделить алгоритмические структуру достаточно сложно.

Анализ методических подходов к обучению программированию позволил сделать вывод о дидактическом потенциале идей для повышения эффективности обучения программированию. Определены методические основания использования семиотического подхода в обучении программированию учащихся 7–9 классов: согласованность с основополагающим системно-деятельностным подходом, целенаправленное развитие знаково-символических действий учащихся, повышение уровня учебной мотивации и уровня сформированности предметных знаний.

1.3 Технология И.Р. Дединского при обучении программированию.

Углубленный подход к преподаванию информатики в большинстве случаев применяется в учебных заведениях или группах физико-математической направленности и предполагает курс программирования, что связано с дальнейшим обучением по этому профилю в ВУЗе. В большинстве случаев способом реализации курса является решение большого количества изолированных алгоритмических задач (так называемый олимпиадный подход).

Однако, если ограничиваться только этим и игнорировать современные тенденции развития процесса разработки программного обеспечения, может получиться, что даже успешный олимпиадник будет испытывать серьезные проблемы с успешностью при попытках выйти за пределы олимпиадной стилистики. Это связано с тем, что участие в разработке ПО, как для научных целей, так и в качестве инженерной профессии – процесс проектно-ориентированный, а это требует многих качеств, которые в олимпиадном подходе не нужны и, как следствие, не развиваются.

В результате характерный для каждой профессии диссонанс между «тем, чему учили», «тем, что надо в работе», описывается непустым множеством образовательных разрывов, которые в настоящее время учащийся и студент должен преодолевать сам, и которые составляет его личный опыт. Такая ситуация существует и в школе, и в ВУЗе. В то же время, большинство разрывов типичны и легко обнаруживаются в ходе внимательного анализа.

Цель данной технологии – проанализировать образовательные разрывы и построить курс программирования таким образом, чтобы минимизировать такие разрывы и максимизировать набор конструктивного положительного опыта, не ограничивающимся лишь конкретными приемами, шаблонами и средствами. Это позволяет учащимся в дальнейшем ориентироваться в меняющемся мире ИТ-технологий, которые часто успевают развиться и умереть до того, как по ним выйдет первый учебник. В таких условиях главная учебная задача, и не только в сфере ИТ, – научить ученика действовать грамотно и самостоятельно. Под грамотностью здесь понимается умение классифицировать проблемы, знать типовые решения, выбирать из них спектр адекватных решений, комбинировать их, придумывать новые решения, контролировать качество, мыслить не рецептами, а как минимум технологиями [10].

Для этого автором вводится понятие когнитивно-технологической единицы (КТЕ), как единицы действительного усвоения знаний, определенной следующим образом:

- Зачем это надо,
- Что это такое,
- На чем основано и с чем связано,
- Как это применять,
- Где это можно и где нельзя использовать,
- Чем придется пожертвовать,

- Что будет, если этого не делать,
- Какие в этом «подводные камни» (чего опасаться).

Разработанный курс рассчитан на учащихся 7(8) – 10(11) классов, нагрузку минимум 4 учебных часа в неделю и систему факультативов. Он учитывает разнородную предварительную подготовку учащихся, и тот факт, что часть из них не изучали информатику и программирование вовсе. По этой причине в начале курса преподавание ведется «с нуля», в предположении, что учащийся не обладает какими-либо специальными знаниями в области программирования. По этой причине используются следующие принципы:

Во главу угла ставится задача, понимаемая как часть проекта, и, главное, путь от задачи к решению, а не кодирование алгоритма.

Для записи алгоритма на языке программирования выбирается минимальное подмножество средств языка, чтобы не акцентировать внимания на кодировании и для более легкого перехода на другие языки программирования.

Самостоятельность решения является ключевым условием, которое необходимо доказать при сдаче работы.

Понимание учащимся тех средств, с помощью которых он решил задачу, ставится выше уровня самих средств решения.

Аккуратность и надежность решения ставятся выше «программистских трюков», иногда позволяющих в отдельных случаях добиться несколько лучших результатов.

Задачи ставятся в нескольких вариантах различной сложности (от базового до творческого), при сдаче работы засчитывается решение на любом уровне. Уровень сложности фиксируется и используется как дополнительная информация к оценке, для выяснения и повышения уровня профессионализма ученика.

Главным методологическим принципом является системный подход.

В обучении активно применяются парные и групповые техники (обмен кодом и документацией, перекрестные peer review и тестирование, групповая разработка стандартов взаимодействия участников проекта). Эти же техники используются при подготовке к ЕГЭ по информатике.

Важнейшей задачей курса является формирование системы профессиональных ценностей (предпочтений) ученика. В конечном счете, это формирование и есть основная инвариантная методологическая задача курса, так как все остальное – технология и будет неотвратимо изменяться с течением времени.

Принятый подход, ориентированный на проектную работу, сильно увлекает многих учеников и дает не только высокие проектные результаты (призовые места на Всероссийских и международных конкурсах), но и высокие олимпиадные (победителей и призеров Всероссийского и регионального уровня). Однако надо отметить, что он не совпадает с традиционным подходом (ставящим во главу угла олимпиадное программирование) и не всегда одобряется приверженцами чисто олимпиадного подхода, которые зачастую хотят получить ученика "целиком и полностью" и воспринимают проектную работу такого уровня как конкуренцию. Тем не менее, действительно сильные олимпиадные школы России видят в нем большую перспективу.

Результатом прохождения курса становится не только понимание основных принципов программирования и владение основными алгоритмическими конструкциями, но и серьезные концептуальные и технологические навыки, позволяющие самостоятельно разрабатывать проекты достаточно большого для школьников объема (порядка курсовой работы 2-3 курса ВУЗа), успешно работать в групповых проектах, требующих активного взаимодействия участников, а некоторым – участвовать и побеждать в различных конкурсах Всероссийского и международного уровней, участвовать в научных конференциях РАН.

Выводы по г лаве 1.

1. Проведен обзор и анализ состояния обучения программированию в основной школе и установлено, что наиболее важной проблемой обучения программированию в основной школе является незаинтересованность учеников программированием и как научить понимать и решать задачи.

2. Рассмотрены различные подходы обучения программированию в основной школе: системный подход (И.О. Одинцов и др.), деятельностный подход (Е.А. Ракитина и др.), когнитивный подход (J. Reinfelds и др.), проблемный подход (Е.В. Касьянова, К.Ю. Поляков и др.), семиотический подход (Р. Andersen, К. И. Баумане, Н.И. Рыжова и др.).

3. Рассмотрена технология И.Р.Дединского, особенностью которой является не обучение учеников программированию на каком то определенном языке программирования, а понимание основных принципов программирования, владение основными алгоритмическими конструкциями, формирование серьезных концептуальных и технологических навыков на основе системного подхода.

Глава 2. Методические рекомендации обучения программированию с использованием технологии Дединского И.Р.

2.1 Методическое планирование системы уроков обучения программированию с использованием технологии Дединского И.Р.

Таблица 2. Методическое планирование системы уроков обучения программированию с использованием технологии Дединского И.Р.

Алгоритмы и элементы программирования. Цель темы «Алгоритмы и элементы программирования» состоит в формировании базовых знаний по алгоритмизации и программированию и умений создаватьпрограммы для решения задач, возникающих в процессе учебы и вне ее. **По завершению темы «Алгоритмы и элементы программирования» учащийся должен знать:** • понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции; • основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти; • подпрограммы, составление библиотек подпрограмм; • объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов. **Учащиеся должны уметь:** • составлять алгоритмы для решения учебных задач различных типов; • выражать алгоритм решения задачи различными способами (словесным, графическим, в том числе и в виде блок-схемы, с помощью формальных языков и др.); • определять наиболее оптимальный способ выражения алгоритма для решения конкретных задач (словесный, графический, с помощью формальных языков); • определять результат выполнения заданного алгоритма или его фрагмента; • составлять несложные алгоритмы управления исполнителями и анализа числовых и текстовых данных с использованием основных управляющих конструкций	**Класс: 9**

последовательного программирования и записывать их в видепрограмм на выбранном языке программирования; выполнять эти программы на компьютере; • использовать величины (переменные) различных типов, табличные величины (массивы), а также выражения, составленные из этих величин; использовать оператор присваивания; • анализировать предложенный алгоритм, например, определять какие результаты возможны при заданном множестве исходных значений; • использовать логические значения, операции и выражения с ними; • записывать на выбранном языке программирования арифметические и логические выражения и вычислять их значения.	

№ урока в системе	**Тема урока**	**Содержание урока** (основные понятия, способы действий)
1.	Введение в язык программирования Python.	Программирование, Python, особенности Python, история Python.
2.	Графика в Python.	Программирование, Python, графика, окно, графические обекты.
3.	Движение графического объекта в Python.	Python, графика, графические объекты, движение графических объектов.
4.	Отражение графического объекта от стен окна.	Python, графика, графические объекты, отражение графических объектов от стен окна, окно, неполный условный оператор if.
5.	Управление графического объекта, с помощью нажатия левой или правой кнопки мыши.	Графика, графические объекты, движение графических объектов, управление графических объектов, событие мыши, условный оператор if.
6.	Создание случайного множества графических объектов.	Графика, графические объекты, движения графических объектов, списки, случайное множество графических объектов, цикл while.
7.	Столкновение движущегося графического объекта с другими графическими объектами.	Графические объекты, движение графического объекта, управлене графического объекта, столкновение движущегося графического объекта с другими графическими объектами, условный оператор if, ввод и вывод данных.

2.2 Методическое планирование уроков по программированию с использованием технологии Дединского И. Р.

Урок 1. Введение в язык программирования Python.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: познакомить учащихся языком программирования, с особенностями языка; сформировать у учащихся первичные знания по применению изученного материала.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: владение информацией о языке программирования Python, представление об особенностях языка; владение понятиями «Python».

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: фронтальная

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Рефлексия.

Ход урока:

1. Организационный этап.

Здравствуйте, ребята! Сегодня мы с вами перейдем к изучению нового раздела информатики - программированию.

Тема нашего занятия «Знакомство с Python и средой программирования».

2. Усвоение нового материала.

Программирование – это создание компьютерных программ. Все программы: игры, антивирусы, текстовые редакторы на компьютере были написаны программистами. Мы конечно, не сможем создать такую большую программу как антивирус или редактор Microsoft Office Word, но небольшие игры сделать постараемся.

Компьютерные программы пишут на специальных языках программирования. Язык программирования – это язык, понятный компьютеру.

Мы будем изучать программирование на языке Python. Это современный язык, он постоянно развивается и дорабатывается. Этот язык используется в таких проектах, как Google, YouTube, Instagram, Яндекс, Facebook и других. Он легок и прост в использовании.

История

Язык программирования Python был создан примерно в 1991 году голландцем Гвидо ван Россумом.

Свое имя - Пайтон (или Питон) - получил от названия телесериала.

После того, как Россум разработал язык, он выложил его в Интернет, где уже целое сообщество программистов присоединилось к его улучшению. Python активно совершенствуется и в настоящее время. Часто выходят его новые версии.

Вот его официальный сайт: http://python.org.

Особенности

Python – это интерпретируемый язык программирования: исходный код частями преобразуется в машинный в процессе выполнения специальной программой — интерпретатором.

Python характеризуется ясным синтаксисом. Читать код на этом языке программирования достаточно легко, т.к. в нем мало вспомогательных элементов, а правила языка заставляют программистов делать отступы, ведь хорошо оформленный текст с малым количеством отвлекающих элементов читать и понимать легче.

А теперь перейдем к основной части нашего урока.

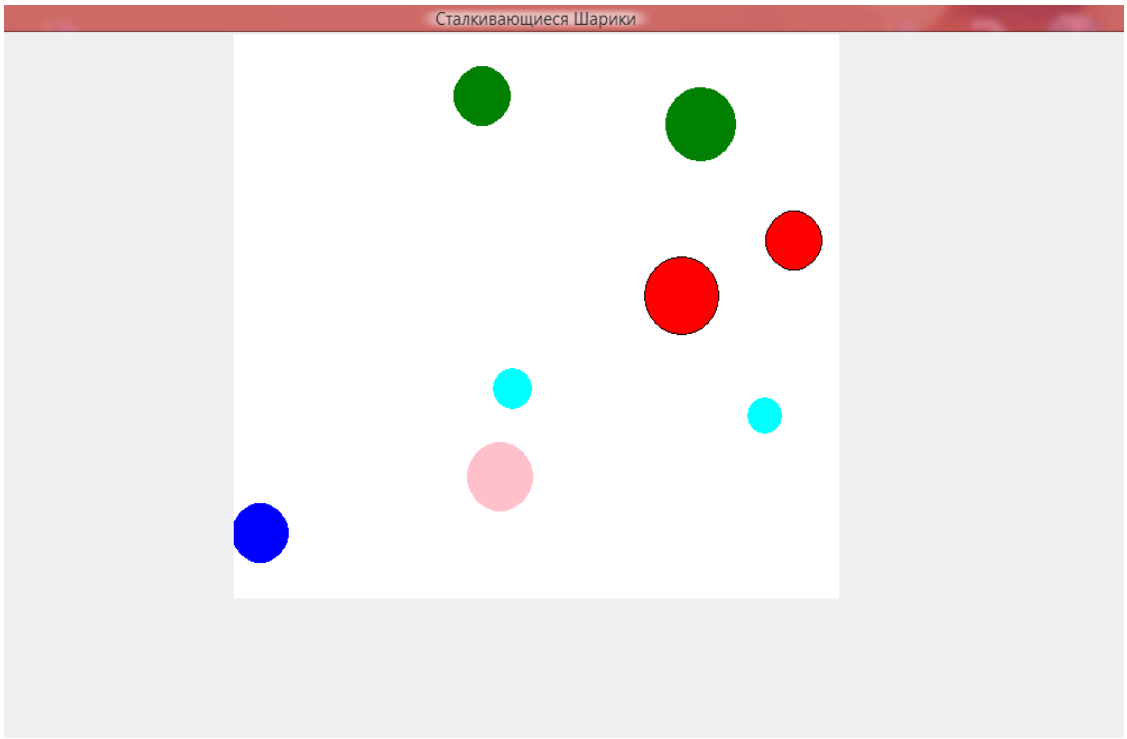

Рис. 3. Скриншот игры

На рисунке 3 представлен скриншот игры, код которой был написан на языке Python.

Суть игры:

При нажатии левой кнопки мыши появляется шарик и движется, после, нажимая правую кнопку мыши он меняет свое направление вверх - вправо,

вниз - вправо, нажимая левую кнопку мыши он меняет направление вверх - влево, вниз -влево.

Когда шарик прикасается к другим шарикам разных цветов, кроме красного, они исчезают, если шарик прикасается к шарикам красного цвета, то исчезнет он сам и игра будет окончена.

Цель игры прикоснуться ко всем шарикам разных цветов, не прикоснувшись при этом к красному.

А теперь давайте посмотрим код этой игры. Приложение А.

С первого взгляда он кажется сложным, но на самом деле каждая программа состоит из подпрограмм и если внимательно посмотреть на каждую подпрограмму, то можно увидеть, что она состоит из самых основных алгоритмов языка Python. И мы в этом убедимся!

Данная программа состоит из 5 основных фрагментов:

1. Создание графического объекта;
2. Движение графического объекта;
3. Отскакивание графического объекта от стен окна и его управление;
4. Создание списка случайных неподвижных графических объектов;
5. Столкновение графических объектов.

Ваша задача разделиться на 4 группы.

На каждом уроке мы будем поэтапно рассматривать фрагменты программы и выполнять соответствующее задание, то есть пошагово писать игру. К концу четверти у каждой группы получится своя собственная полноценная игра. Она будет подобна, но отлична от моей.

К концу каждого урока будут выставляться оценки. Самым важным критерием оценивания будет понимание изученного материала и применение его на практике.

3. Подведение итогов. Рефлексия.

1. На каком языке программирования мы будем писать программу?
2. В чем особенность этого языка?
3. Возникли ли у вас вопросы по дальнейшей работе?
4. Заинтересовала ли вас дальнейшая деятельность?
5. Хотели ли бы вы научиться программировать?
6. Что именно бы вы хотели программировать?
7. Хотите ли вы создать свою собственную игру?

Урок 2. Графика в Python.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: сформировать умения создавать графические объекты на языке программирования Python.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: сформированность умений создавать графические объекты на языке программирования Python.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.
4. Сообщение домашнего задания.
5. Подведение итогов. Рефлексия.

Ход урока:

1. Организационный этап.

Здравствуйте, ребята! На прошлом уроке мы с вами познакомились с языком программирования Python. Вам была представлена игра, написанная на данном языке. Сегодня мы перейдем к рассмотрению первого фрагмента программы, с помощью которого создается графический объект.

Это и будет тема нашего занятия: «Графика в Python». Запишите ее в тетрадь.

2. Усвоение нового материала.

С помощью графики в Python можно рисовать фигуры и изображения, создавать анимацию, визуализировать математические вычисления в Python и использовать элементы графики в компьютерных играх на Python. Для работы с графикой на любом языке в том числе и Python нужно импортировать модуль графики, для Python это graphics.py.

Давайте посмотрим на игру, как вы думаете с чего нужно начинать, при рисовании данной игры? *Ученики перечисляют возможные варианты.*

Чтобы начать работу с графикой в Python, нужно создать окно для графики.

Это применимо для программировании на любом языке.

Графический объект = GraphWin("Название окна для графики", ширина окна для графики в пикселях, высота окна для графики в пикселях) GraphWin это ключевое слово, которое задаёт окно графической области, в котором будут отображаться графические объекты.

Вся работа с графикой будет осуществляться нами через графические объекты.

Общая структура

Графический_объект.Вызов_команды()

Общая структура графической программы в Python.

```
# импортируем библиотеку graphics
from graphics import *
# создаём окно для графики
win = GraphWin("Окно для графики", 400, 400)
# ...рисуем все объекты...
win.getMouse() # ждём нажатия кнопки мыши
win.close() # закрываем окно для графики
```

В этой программе мы определили объект графическое окно win и открыли его с размерами 400 на 400 пикселей.

win.getMouse() останавливает выполнение программы до тех пор, пока пользователь не нажмёт на любую кнопку мыши, наведённую на область окна win.

win.close() закрывает окно для графических объектов win.

Что будет следовать после создания окна? *Ученики перечисляют варианты.*

Создание Графических объектов.

С помощью графических модулей в программах можно отобразить точку, линию, окружность, прямоугольник, эллипс и многоугольник, можно вывести текст на экран. Как говорилось ранее, для Python - graphics.py

Чтобы задать расположение объекта в графическом окне Python, необходимо указать его координаты в системе координат Python.

Начало координат находится в левом верхнем углу окна для графики. Положительное направление оси X слева направо, оси Y сверху вниз. Чем больше X, тем правее точка, чем больше Y, тем точка ниже. Чтобы нарисовать заданный объект obj в окне для графики win, нужно использовать процедуру obj.draw(win). Остановить выполнение программы можно процедурой win.getMouse(), программа не будет выполняться, пока пользователь не нажмёт на любую кнопку мыши. Окно для графики закрывается с помощью win.close().

Перед тем, как рисовать графические объекты в заданном графическом окне, нужно их задать.

Для задания точки в Python используется функция Point(x, y)

obj = Point(x, y)

x, y – координаты точки.

Пример программы на Python, которая задаёт и отображает точку в графическом окне.

```
from graphics import * # импортируем библиотеку graphics
win = GraphWin("Окно для графики", 400, 400) # создаём окно для графики размером 400 на 400 пикселей
obj = Point(50, 50) # создаём точку в координатах (50, 50)
obj.draw(win) # отображаем точку в окне для графики
win.getMouse() # ждём нажатия кнопки мыши
win.close() # закрываем окно для графики
```

Для задания отрезка в Python используется функция Line(объект точка первого конца, объект точка второго конца)

obj = Line(Point(x1, y1), Point(x2, y2))

x1, y1 – координаты начала отрезка линии,

x2, y2 – координаты конца отрезка линии.

Пример программы на Python, которая отображает линию в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Line(Point(50, 50), Point(350, 350))
obj.draw(win)
win.getMouse()
win.close()
```

Для отображения окружности в Python используется

obj = Circle(Point(x, y), R)

x, y – координаты центра окружности,

R – радиус окружности.

Пример программы на Python, которая отображает окружность в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Circle(Point(200, 200), 50)
obj.draw(win)
win.getMouse()
win.close()
```

Для отображения прямоугольника в Python используется процедура

obj = Rectangle(Point(x1, y1), Point(x2, y2))

x1, y1 – координаты левого верхнего угла прямоугольника,

x2, y2 – координаты правого верхнего угла прямоугольника

Пример программы на Python, которая отображает прямоугольник в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Rectangle(Point(50, 50), Point(200, 300))
```

```
obj.draw(win)
win.getMouse()
win.close()
```

Для отображения эллипса в Python используется процедура

```
obj = Oval(Point(x1, y1), Point(x2, y2))
```

x1, y1 – координаты первого фокуса эллипса,

x2, y2 – координаты второго фокуса эллипса.

Пример программы на Python, которая отображает эллипс в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Oval(Point(200, 200), Point(300, 350))
obj.draw(win)
win.getMouse()
win.close()
```

Для отображения многоугольника в Python используется процедура

```
obj = Polygon(Point(x1, y1), Point(x2, y2),…, Point(xn, yn))
```

x1, y1, x2, y2,…, xn, yn – координаты вершин многоугольника.

Пример программы на Python, которая отображает пятиугольник в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Polygon(Point(10, 10), Point(300, 50), Point(200, 300), Point(150, 150),
Point(70, 70))
obj.draw(win)
win.getMouse()
win.close()
```

Для редактирования границ объектов в Python используются процедуры setOutline(“цвет границы”) и

setWidth(ширина границы).

obj.setOutline("blue") – объект obj отображается с границей синего цвета.

obj.setWidth(5) – объект obj отображается с шириной границы 5 пикселей.

Если не редактировать границы, графический объект в Python будет отображаться с границами чёрного цвета шириной 1 пиксель.

Пример программы на Python, которая отображает фигуру с нестандартной границей и заливкой в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Polygon(Point(10, 10), Point(300, 50), Point(200, 300), Point(150, 150), Point(70, 70))
obj.setOutline("blue")
obj.setWidth(5)
obj.setFill("cyan")
obj.draw(win)
win.getMouse()
win.close()
```

Есть ли у кого-то вопросы по пройденной теме?

Перейдем к практической части, присаживайтесь за свои компьютеры.

3. Применение полученных знаний.

А сейчас вам предстоит самим создать графические объекты на языке Python. Графический объект можно выбрать любой, на свое усмотрение.

Ученики, работая в минигруппах, создают часть программы. Показывают учителю. Учитель проверяет.

4. Сообщение домашнего задания.

Запишите домашнее задание: повторить изученную тему, если не доделали задание, доделать его дома.

5. Подведение итогов. Рефлексия.

1. Какую тему мы сегодня изучали?
2. Чему вы научились на данном уроке?
3. С чем у вас возникли трудности?
4. Для чего вам может пригодиться создание графических объектов?
5. Понравилось ли вам этим заниматься?

Урок 3. Движение графического объекта в Python.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: сформировать умения движения графического объекта на языке программирования Python.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: сформированность умений движения графического объекта на языке программирования Python.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.
4. Сообщение домашнего задания.
5. Подведение итогов. Рефлексия.

Ход урока:

1. Организационный этап.

Здравствуйте ребята! На прошлом уроке мы научились создавать графические объекты. На сегодняшнем уроке мы научимся, как задавать движение графических объектов в Python. Собственно, это и есть тема нашего урока, запишите ее в тетрадь.

2. Изучение нового материала.

Как вы думаете, как осуществляется передвижение графических объектов?

Чтобы переместить графический объект в Python, используйте процедуру move(dx, dy), которая перемещает объект на dx пикселей вправо и dy пикселей вниз.obj.move(50, 50) смещает объект obj на 50 пикселей вправо и 50 пикселей вниз.

Для клонирования объектов используется процедура clone() newObj = obj.clone(). Создаётся новый графический объект newObj, который идентичен объекту obj, который мы клонировали.

Для удаления фигур с экрана используется процедура undraw(). Объект удаляется с графического окна, но не удаляется из программы. obj.undraw()

Пример программы на Python, которая удаляет, перемещает и копирует объект в графическом окне.

```
from graphics import *
win = GraphWin("Окно для графики", 400, 400)
obj = Polygon(Point(30, 10), Point(30, 50), Point(20, 30), Point(15, 30), Point(7, 7))
obj.setOutline("blue")
obj.setWidth(2)
obj.setFill("cyan")
obj.draw(win)
win.getMouse()
obj.undraw()
win.getMouse()
obj.draw(win)
obj.move(100, 100)
win.getMouse()
shape = obj.clone()
shape.move(-100, -100)
shape.draw(win)
win.getMouse()
win.close()
```

Есть ли в классе такие ребята, которые не поняли, как осуществляется движение графических объектов?

Рассаживайтесь за свои рабочие места.

3. Применение полученных знаний.

На прошлом уроке вы создали свои графические объекты, сегодня ваша задача состоит в том, чтобы придать этим объектам движение.

Ученики, работая в минигруппах, создают часть программы. Показывают учителю. Учитель проверяет.

4. Сообщение домашнего задания.

Запишите домашнее задание: Повторить изученный материал. Кто не успел доделать задание, доделать его дома.

5. Подведение итогов. Рефлексия.

1. Что мы изучили на данном уроке? Что научились делать?
2. Какие трудности у вас возникли?
3. Как вы думаете, для чего вам может пригодиться умение задавать движение графическим объектам?
4. Нравится ли вам то, чем мы занимаемся?
5. Какого результата мы хотим достигнуть?
6. Хотите ли вы научиться создавать игры?

Урок 4. Отражение графического объекта от стен окна.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: сформировать умения отражения графического объекта от стен окна на языке программирования Python.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: сформированность умения отражения графического объекта от стен окна на языке программирования Python.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.
4. Сообщение домашнего задания.
5. Подведение итогов. Рефлексия.

Ход урока:

1 Организационный этап.

На прошлых уроках мы научились создавать графические объекты, так же научились осуществлять движение объекта, но как вы заметили он улетает за границу окна. Сегодня нам предстоит научиться, как сделать так, чтобы он отражался от стен окна, а не уходил за его пределы.

2 Изучение нового материала.

Для того, чтобы графический объект отскакивал от стен окна, нам необходимо изучить условный оператор if.

Конструкция условного оператора означает следующее:

Если <выполняется условие> делать: какие-то действия.

Например:

if a>b:

print(a)

«Если а больше b, то вывести а».

Давайте посмотрим на фрагмент нашего кода, который отвечает за то, чтобы шарики отскакивали от стен окна:

```
def move(self):  #движение шара

    # столкновение со стенами
if (self.x + self.r + self.dx >= WIDTH) or (self.x - self.r + self.dx <= ZERO):
self.dx = -self.dx #центр шара + радиус + ожидаемое смещение >= ширине экрана, то шар отражается
if (self.y + self.r + self.dy >= HEIGHT) or (self.y - self.r + self.dy <= ZERO):
self.dy = -self.dy #центр шара + радиус + ожидаемое смещение >= высоте, то шар отражается
```

В нашем случае, если центр шара + радиус + ожидаемое смещение больше или равен ширине экрана, то шар отражается.

А так же если центр шара + радиус + ожидаемое смещение больше или равен высоте, то шар, так же отражается.

Ребята, скажите пожалуйста, всем понятно, как осуществляется отражение графического объекта от стен окна?

Прошу присесть вас за свои рабочие места и приступить к практической части урока.

3 Применение полученных знаний.

Ваша задача сегодня, сделать так, чтобы ваш графический объект, который движется и улетает за стенки окна, отражался от них.

Ученики, работая в минигруппах, создают часть программы. Показывают учителю. Учитель проверяет.

4 Сообщение домашнего задания.

Запишите домашнее задание: повторить изученный материал. Выучить конструкцию условного оператора if. Завершить задание, если кто-то не успел.

5 Подведение итогов. Рефлексия.

1. Какую тему мы сегодня изучили?
2. С помощью какого оператора, мы можем сделать так, чтобы графический объект отражался от стен окна?
3. Что такое оператор ветвления if?
4. Как выглядит его конструкция?
5. Для чего они могут использоваться?
6. Как они могут помочь вам для создания игры?
7. Понравилась ли вам тема нашего урока?
8. Как вы считаете, чем мы будем заниматься на следующем уроке?

Всем спасибо за урок! Все свободны.

Урок 5. Управление графическим объектом, с помощью нажатия левой или правой кнопки мыши.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: сформировать умения управления графического объекта на языке программирования Python.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: сформированность умений управления графического объекта на языке программирования Python.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.
4. Сообщение домашнего задания.
5. Подведение итогов. Рефлексия.

Ход урока:

1. Организационный этап.

Здравствуйте, дорогие ребята! Тема нашего урока "Управление графическим объектом, с помощью нажатия левой или правой кнопки мыши." Запишите ее в тетрадь.

2. Изучение нового материала.

Давайте посмотрим на фрагмент программы, который отвечает за управление графического объекта, с помощью нажатия левой или правой кнопки мыши.

```
#событиемыши

def mouse_click(event):# параметр event-событие
global main_ball #объявление объекта main ball глобальным, при повторном обращении ее нет если она локальная(был локальным)

    if event.num == 1:  # при нажатии левой кнопки рисует шар
        if 'main_ball' not in globals():  #проверяем, если шар не создан, то мы его рисуем 1
main_ball = Ball(event.x, event.y, MAIN_BALL_RADIUS,
MAIN_BALL_COLOR, INIT_DX,
INIT_DY)#появлениешарикапринажатиимыши
            if main_ball.x > WIDTH / 2:
                main_ball.dx = -main_ball.dx
            if main_ball.y > HEIGHT / 2:
                main_ball.dy = -main_ball.dy
main_ball.draw()#1-появление шара, после нажатия мыши
        else: # если шар есть, то поворачиваем его налево
            if main_ball.dy * (main_ball.dx *10) > 0:#математическое вычисление поворота шара
                main_ball.dy = -main_ball.dy #математическое вычисление поворота шара
else:
                main_ball.dx = -main_ball.dx
elif event.num == 3:  # если нажата правая кнопка мыши то поворот на право
if main_ball.dy * main_ball.dx > 0:
            main_ball.dx = -main_ball.dx
        else:
            main_ball.dy = -main_ball.dy
```

Для того, чтобы мы могли управлять нашим графическим объектом мы должны применить условный оператор if. На прошлом уроке мы изучили неполную форму условного оператора. Но у условного оператора также есть и полная форма. Выглядит она так:

Если <выполняется условие> : делать какие-то действия. Иначе: делать другиедействия.

Иначе означает «если условие не выполняется».

Например:

```
if a>b:
print(a)
else:
print(b)
```

«Если а больше b, то вывести а, иначе вывести b.

В нашем случае, если шар есть, то при нажатии левой кнопки мыши поворот налево, иначе создается шар.

Второе условие:если нажата правая кнопка мыши, то поворот на право.

Всем понятно, как работает полный условный оператор if? Как мы можем применить его, для управления графического объекта, с помощью нажатия левой или правой кнопки мыши?

Присаживайтесь за свои рабочие места.

3. Применение полученных знаний.

Ваша задача сделать так, чтобы вы могли управлять своим графическим объектом. Приступайте к выполнению.

4. Сообщение домашнего задания.

Домашнее задание - повторить законспектированную тему урока, выучить конструкцию полного условного оператора if и кто не успел сделать задание на уроке, доделать его дома.

5. Подведение итогов. Рефлексия.

Понравился ли вам сегодняшний урок? Что вы научились делать? Получилось ли у вас? С чем возникли трудности? Как вы их решили? Что нам осталось сделать, чтобы наша игра была завершенной?

Урок 5. Создание случайного множества графических объектов.

Предмет: «Основы программирования на Python»

Класс: 9

Тип урока: изучение новых знаний.

Цели урока:

Образовательные: сформировать умение создавать случайное множество графических объектов на языке программирования Python.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

Предметные: сформированность умения создавать случайное множество графических объектов на языке программирования Python.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.

4. Сообщение домашнего задания.
5. Подведение итогов. Рефлексия.

Ход урока:

1 Организационный этап.

Всем здравствуйте!

На сегодняшний день у нас есть движущийся графический объект, который отражается от стен окна и которым мы можем управлять, нажимая правую или левую кнопку мыши.

Нам осталось создать множество неподвижных, случайных графических объектов и описать столкновение управляемого объекта с неподвижными.

На данном уроке мы будем учиться создавать множество случайных, неподвижных объектов.

2 Изучение нового материала.

Для создание случайного множества графических объектов нам понадобятся знания о списках.

Список (list) представляет тип данных, который хранит набор или последовательность элементов. Для создания списка в квадратных скобках ([]) через запятую перечисляются все его элементы.

Например, определим список чисел:

```
numbers = [1, 2, 3, 4, 5]
```

Также для создания списка можно использовать конструктор list():

```
numbers1 = []
numbers2 = list()
```

Оба этих определения списка аналогичны - они создают пустой список.

Конструктор list для создания списока может принимать другой список:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(numbers)
```

Перед тем, как мы начнем разбирать код программы, я заранее скажу, что для создания в нашей программе списков графических объектов нам необходимо изучить циклы.

Конспектируйте в тетрадь.

Циклы — это инструкции, выполняющие одну и туже последовательность действий, пока действует заданное условие.

Универсальным организатором цикла в языке программирования Python (как и во многих других языках) является конструкция while. Слово "while" с английского языка переводится как "пока" ("пока логическое выражение возвращает истину, выполнять определенные операции").

Когда выполнение программного кода доходит до цикла while, выполняется логическое выражение в заголовке, и, если было получено True (истина), выполняются вложенные выражения. После поток выполнения программы снова возвращается в заголовок цикла while, и снова проверяется условие.

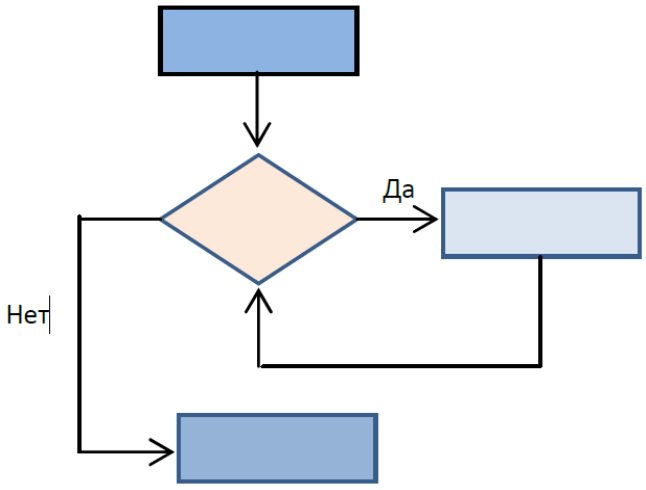

Рис. 4 Схема цикла while

На рисунке 4 представлена схема цикла while.

Простейший цикл на языке программирования Python может выглядеть так:

```
strl = "+"
i = 0
while i < 10;
print (strl)
        i = i + 1
```

Ну а теперь вернемся к фрагмент программы.

Сначала нам необходимо создать список шариков:

```
def create_list_of_balls(number):# указываем количество создаваемых шариков
```

После создать пустой список:

```
    lst = []#создание пустого списка
```

После этого, нам необходимо задать цикл:

```
    while len(lst) < number:#пока длина списка меньше заданного нами количества
```

То мы создаем следующий шарик:

```
        next_ball = Ball(random.choice(range(MAX_RADIUS, WIDTH - MAX_RADIUS)),#ширина #x
random.choice(range(MAX_RADIUS, HEIGHT - MAX_RADIUS)),#высота#y
  random.choice(range(MIN_RADIUS, MAX_RADIUS)),#радиус #r
 random.choice(COLORS)) #рандомныецвета
```

(random.choice - рандомно выбирает значение из заданных значений)

Добавляем и рисуем следующий шарик:

```
lst.append(next_ball)#добавление шара в список
            next_ball.draw()#рисуется шар из списка
    return lst
```

Остались ли у кого-то вопросы?

Теперь вы знаете, как создаются списки графических объектов. Осталось только научиться делать это на практике, можете пройти к своим компьютерам.

3 Применение полученных знаний.

Ваша задача создать в своей программе список графических объектов. На выполнение работы у вас есть 20 минут.

Ученики, работая в минигруппах, создают часть программы. Показывают учителю. Учитель проверяет.

4 Сообщение домашнего задания.

Запишите домашнее задание: Повторить изученный материал. Доделать задание, кто не успел.

5 Подведение итогов. Рефлексия.

1. Что мы изучили сегодня на уроке?
2. Что вы научились делать?
3. В чем у вас возникли трудности?
4. Что такое списки? Какие они бывают?
5. Для чего вы можете их использовать?
6. Все ли вы доделали?
7. Что нам осталось доделать для завершения игры?
8. Хотите ли вы получить уже готовый продукт?

Всем спасибо за внимание! До свидания.

Урок 6. Столкновение движущегося графического объекта с другими графическими объектами.

Класс: 9

Тип урока: повторение ранее изученных знаний

Цели урока:

Образовательные: сформировать понятие, как сделать так, чтобы движущийся графический объект отражался от других графических объектов.

Воспитательные: развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.

Развивающие: совершенствование умения анализировать, сравнивать, систематизировать и обобщать, развитие коммуникативных умений обучающихся.

Планируемые результаты:

*Предметные:*сформированность умения отражать движущийся графический объект от других графических объектов.

Личностные: сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Метапредметные: умение контролировать и корректировать учебную деятельность.

Форма обучения: Фронтальная, групповая.

Этапы урока:

1. Организационный этап.
2. Изучение нового материала.
3. Применение полученных знаний.
4. Подведение итогов. Рефлексия.

Ход урока:

1. Организационный этап.

Всем здравствуйте! Присаживайтесь, открывайте тетради. Сегодня у нас завершающий урок, у нас есть движущийся графический объект, который отражается от стен окна и которым мы умеем управлять, с помощью кнопок

мыши. Так же у нас есть случайное множества графических объектов, нам осталось описать столкновение движущегося графического объекта с другими графическими объектами и наша игра будет выполнима, следовательно закончена.

Запишите тему нашего урока в тетрадь: "Столкновение движущегося графического объекта с другими графическими объектами"

2. Изучение нового материала.

Ну что же, смотрим на последний фрагмент программы. Как вы видите, для описания столкновения движущегося шара с другими нам понадобится условный оператор if.

Если движущийся шар столкнулся с неподвижным и если это хороший шар(то есть не красный), то мы скрываем его, окрашиваем в цвет фона и меняем направление нашего шара.

Иначе (если шар плохой), наш шар останавливается.

Self - это движущийся шар, ball - это неподвижный шар

```
        # столкновение с шарами
        for ball in balls:# каждый элемент balls попадает в ball

if self.is_collision(ball):#еслишарстолкнулся
if ball.color != BAD_COLOR:  #определяется что шар хороший
                    ball.hide()#1 действие-скрытие шара
                    balls.remove(ball)#2 действие-удаление выбитого шара из списка
                    self.dx = -self.dx#3 действие-смена направления
                    self.dy = -self.dy#3 действие-смена направления
                else:  # шар плохой
                    self.dx = self.dy = 0#остановка нашего шара
        self.hide()
        self.x += self.dx #смена координат относительно центра
        self.y += self.dy #смена координат относительно центра
if self.dx * self.dy != 0:
self.draw()
```

```
#считаем плохие шары
def count_bad_balls(list_of_balls):#сосписккашаров
    result = 0
    for ball in list_of_balls:# беретсяшаризсписккашаров
        if ball.color == BAD_COLOR:#есликрасный
            result += 1
    return result
```

Теперь нам нужно вывести на экран "ВЫ ВЫИГРАЛИ" или "ВЫ ПРОИГРАЛИ" давайте подумаем, как мы можем это сделать.

Как мы уже говорили, цель игры - это прикоснуться ко всем шарам разных цветов, кроме красного.

```
# движениешара
def main():
    if 'main_ball' in globals(): #проверка, естьливглобале main ball
main_ball.move() # если создан, то шарик двигается
```

Написать мы это можем так:

```
    if len(balls) - num_of_bad_balls == 0: #если длинна списка - количество плохих шаров =0, то
```

Выводим по середине экрана "ВЫ ВЫИГРАЛИ", выбираем шрифт и размер.

```
canvas.create_text(WIDTH / 2, HEIGHT / 2, text="ВЫВЫИГРАЛИ!",
font="Arial 45", fill="lime")
```

Так же мы можем сделать и с красным цветом, только отличие будет в том, что когда наш шар прикасается к красному он останавливается, это мы и используем в нашем условие:

```
main_ball.dx = main_ball.dy = 0 #остановкашара
elif main_ball.dx * main_ball.dy == 0:#если смещение = 0, значит столкнулся с красным шаром, то
```

Выводим по середине экрана "ВЫ ПРОИГРАЛИ", выбираем шрифт и размер.

```
canvas.create_text(WIDTH / 2, HEIGHT / 2, text="ВЫПРОИГРАЛИ!",
font="Arial 45", fill="red")
```

И записываем функцию, которая вызывает сама себя:

```
root.after(DELAY, main) # DELAY-скорость передвижения шара.
```

А теперь присаживайтесь за свои компьютеры и дописывайте вашу игру!

3. Применение полученных знаний.

Ваша задача описать столкновение движущегося графического объекта с другими графическими объектами, если столкновение происходит с "хорошим объектом" вывести на экран сообщение о победе, если с "плохим объектом", вывести сообщение о проигрыше.

4. Подведение итогов. Рефлексия.

Ну вот вы и написали свои игры! У всех групп они получились совершенно разные, что и интересно.

1. Расскажите пожалуйста, чему вы научились за время написания игры?
2. Что вы узнали о синтаксисе в Python, с какими операторами познакомились?
3. Понравилось ли вам писать игры? Хотели ли бы вы, чтобы уроки были продолжены в такой форме?
4. Чему бы вы еще хотели научиться на уроках программирования?

Всем спасибо за проделанную работу. До свидания.

Выводы по главе 2

1. Разработано методическое планирование уроков обучения программированию с использованием технологии Дединского И.Р.
2. Сформированы конспекты уроков с использованием технологии Дединского И.Р.

Заключение.

В качестве заключения сформулируем основные результаты выпускной квалификационной работы.

Во-первых, в ходе проведенного исследования были определены теоретические основы технологий при обучении программированию в основной школе.

Во-вторых был проведен обзор и анализ состояния обучения программированию в основной школе и установлено, что наиболее важной проблемой обучения программированию в основной школе является незаинтересованность учеников программированием, так как в разделе «Алгоритмизация и программирование» школьного курса учащимся предлагается для решения множество различных задач, эти задачи обычно математические, результатом которых являются цифры - решения квадратного уравнения на черном экране, что совершенно не наглядно. Наибольший интерес у учащихся вызывает графика, в связи с этим возникла идея давать большую часть материала, используя графику. Отсюда вытекает и изменение учебного плана занятий: на первое место можно сразу поставить изучение графических операторов, а затем уже с их помощью объяснять (по возможности) весь остальной материал.

В-третьих были рассмотрены различные подходы обучения программированию в основной школе: системный подход (И.О. Одинцов и др.), деятельностный подход (Е.А. Ракитина и др.), когнитивный подход (J. Reinfelds и др.), проблемный подход (Е.В. Касьянова, К.Ю. Поляков и др.), семиотический подход (P. Andersen, К. И. Баумане, Н.И. Рыжова и др.).

В-четвертых, особенностью технологии Дединского И.Р является не обучение учеников программированию на каком то определенном языке программирования, а понимание основных принципов программирования, владение основными алгоритмическими конструкциями, формирование

серьезных концептуальных и технологических навыков на основе системного подхода.

Наконец, были разработаны и представлены методические рекомендации обучения программированию с использованием технологии Дединского И.Р

Таким образом, на основании выше изложенного можно утверждать, что цель выпускной квалификационной работы достигнута.

Список используемой литературы:

1. Бабанский, Ю.К. Оптимизация педагогического процесса: Общедидактический аспект [Текст] / Ю.К. Бабанский. -М.: Педагогика, 1977.

2. Бабанский, Ю.К. Проблемы повышения эффективности педагогических исследований [Текст] / Ю.К. Бабанский. -М.: 1982.

3. Бежанова, М.М. Практическое программирование. Структуры данных и алгоритмы : учебник [Текст] / М.М. Бежанова, Л.А. Москвина, И.В. Поттосин. - М. : Логос, 2001.

4. Беспалько, В.П. Программированное обучение: дидактические основы [Текст] / В.П. Беспалько. -М.: Высш. шк., 1970.

5. Вирт, Н. Алгоритмы + структуры данных = программы [Текст] / Н. Вирт. -М.: Мир, 1985.

6. Гейн, А.Г. Информатика и ИКТ (базовый и профильный уровни) [Текст] / А.Г. Гейн [и др.]. - М.: Просвещение, 2008.

7. Давыдова, Н.А. Программирование [Текст] / Н.А. Давыдова, Е.В. Боровская. - М.: Бином, 2009.

8. Дейкстра, Э. Дисциплина программирования = A discipline of programming [Текст] / Э. Дейкстра. - 1-е изд. - М.: Мир, 1978.

9. Ершов, А.П. Основы информатики и вычислительной техники: пробный учебник для сред. учеб. заведений [Текст] / Под ред. А.П. Ершова. - М.: Просвещение, 1988.

10. Ершов, А.П. Программирование - вторая грамотность [Электронный ресурс] / А.П. Ершов // Режим доступа: http://www. ershov.ras.ru/russian/second_literacy/article.html.

11. Ершов, А.П. Школьная информатика (концепции, состояние, перспективы) [Текст] / А.П. Ершов, Г.А. Звенигородский, Ю.А. Первин. - Новосибирск, 1979.

12. Жемчужников, Д.Г. Разработка динамических игр как средства обучения программированию [Текст] / Д.Г. Жемчужников // Вестник

13. Жемчужников, Д.Г. Создание компьютерных игр как средство обучения школьников программированию [Текст] / Д. Г. Жемчужников // Информатика и образование. - 2012. -№8. - С. 49-51.

14. Жемчужников, Д.Г. Обучение программированию на основе создания динамических игр : учебно-методическое пособие [Текст] / Д.Г.Жемчужников. - М.: Мэйлер, 2010. - 60 с.

15. Жемчужников, Д.Г. Программирование компьютерных игр: метод повышения мотивации учащихся к изучению информатики [Текст] / Д. Г. Жемчужников // Сборник II педагогических чтений научной школы управления образованием. - М.: МПГУ, 2010. - С. 214-220.

16. Жемчужников, Д.Г. Методика создания компьютерных игр в обучении школьников программированию [Текст] / Д. Г. Жемчужников // Сборник тезисов II Международной конференции «Информационные технологии в образовании». - Москва: МИРЭА, 2010. - Часть 1, С. 43-46.

17. Жемчужников, Д.Г. Создание компьютерных игр в обучении школьников программированию [Текст] / Д. Г. Жемчужников // Сборник XV Всероссийской научно-практической конференции «Новые информационные технологии». - Рязань: РГРУ, 2010. - С. 31-33.

18. Жемчужников, Д.Г. Методика программирования динамических игр: аспект визуализации алгебраических функций [Текст] / Д. Г. Жемчужников // Сборник статей II Международной научно-практической конференции «Проблемы и возможности современной науки». - Тамбов: ТМБпринт, 2011.- С. 60-61.

19. Жемчужников, Д.Г. Формирование учебно-логических компетенций в процессе обучения программированию игр [Текст] / Д. Г. Жемчужников // Сборник научных трудов «Актуальные проблемы информатизации образования». - Воронеж: Научная книга, 2012. - С. 72-74.

20. Жемчужников, Д.Г. Методика преподавания программирования при помощи сквозной проектной задачи [Текст] / Д. Г. Жемчужников // Инновации и качество лицейского образования. - 2012. - №1. - С. 27-31.

21. Звенигородский, Г.А. Первые уроки программирования [Текст] / Под ред. А.П.Ершова. - М.: Наука, 1985.

22. Информатика, 9 класс. Учебник для учащихся средней школы [Текст] / под ред. Н.В. Макаровой. - СПб.: Питер, 2000.

23. Касаткин, В.Н. Программирование как элемент общего образования [Текст] / В.Н. Касаткин // Кибернетика. - 1973. - № 2.

24. Кузнецов, А.А. О разработке стандарта школьного образования по информатике [Текст] / А.А. Кузнецов // Информатика и образование. -1994. - №1.

25. Лапчик, М.П. Основы программирования: учеб. пособие для учащихся [Текст] / М.П. Лапчик. - М.: НИИ СИМО АПН СССР, 1972.

26. Левченко, И.В. Развитие системы методической подготовки учителей информатики в условиях фундаментализации образования : автореферат дис. ... доктора педагогических наук : 13.00.02 [Текст] / Ирина Витальевна Левченко; [Место защиты: Тул. гос. пед. ун-т им. Л.Н. Толстого] . -М., 2009.

27. Методика преподавания информатики: учеб. пособие для студ. пед. вузов [Текст] / М.П. Лапчик, И.Г. Семакин, Е.К. Хеннер; под общей ред. М.П. Лапчика. - М.: Издательский центр «Академия», 2001.

28. Методы системного педагогического исследования: учеб. пособие [Текст] / под ред. Н.В.Кузьминой.- Л., 1980.

29. Новиков, А.М. Методология научного исследования [Текст] / А.М. Новиков., Д.А. Новиков. -М.: Либроком, 2009.

30. Новиков, А.М. О развитии методических систем. [Электронный ресурс] / А.М. Новиков. - Режим доступа: http://www.anovikov.ru/artikle/met_sys.htm.

31. Новые педагогические и информационные технологии в системе образования: учеб. пособие для студ. пед. вузов и системы повыш. квалиф.

пед. кадров [Текст] / Е.С.Полат, М.Ю.Бухаркина, М.В.Моисеева, А.Е.Петров; под ред. Е.С. Полат. - М.: Издательский центр «Академия», 2000.

32. Попов, О.А. Новая классификация компьютерных игр [Электронный ресурс] / О.А. Попов // Режим доступа: http://psystat.at. ua/publ/4-1-0-30.

33. Проблемы теоретического и системного программирования: сб. науч. тр. [Текст] / под ред. А.П. Ершова. - Новосибирск, 1982.

34. Работа со школьниками в области информатики: Опыт Сиб. отд. АН СССР [Текст] / А.П. Ершов, Г.А. Звенигородский, С.И. Литерат, Ю.А. Первин // Математика в школе. - 1981. - №1. - С.20.

35. Федеральный государственный образовательный стандарт среднего (общего) образования [Электронный ресурс] // Режим доступа: http://standart.edu.ru/catalog.aspx?CatalogId=2588.

36. Федеральный государственный образовательный стандарт среднего (полного) образования [Электронный ресурс] // Режим доступа: http://standart.edu.ru/catalog.aspx?CatalogId=6408.

Приложение А. Код игры.

```
importtkinter
import random

# константы
WIDTH = 540
HEIGHT = 480
BG_COLOR = 'white'
MAIN_BALL_COLOR = 'blue'
MAIN_BALL_RADIUS = 25
BAD_COLOR = 'red'
COLORS = ['aqua', 'black', 'pink', 'yellow', 'gold', 'green', BAD_COLOR]
NUM_OF_BALLS = 10
MAX_RADIUS = 35
MIN_RADIUS = 15
DELAY = 10
INIT_DX = 0.1#смещение на 1 единицу
INIT_DY = 1#смещение на 1 единицу
ZERO = 0


# классшаров
class Ball():
    def __init__(self, x, y, r, color, dx=0, dy=0):#self-объекткласса (init-
методконструктора)
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.dx = dx
        self.dy = dy

    def draw(self): #рисуемшарик
        canvas.create_oval(self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r,
fill=self.color, outline=self.color if self.color != BAD_COLOR else
'black')#задаем окружность координатами расположенных друг на против
друга углов прямоугольника(наша рабочая область).
fill=self.color заполнение шара.

def hide(self):
```

```
        canvas.create_oval(self.x - self.r, self.y - self.r, self.x + self.r, self.y + self.r,
fill=BG_COLOR, outline=BG_COLOR) #скрывает нарисованный шар в цвет
фона

    def is_collision(self, ball): #обработка столкновений шаров, эта функция
проверяет сталкивается ли текущий объект с тем объектом который мы
задали
a = abs(self.x + self.dx - ball.x)
        b = abs(self.y + self.dy - ball.y)
return (a * a + b * b) ** 0.5 <= self.r + ball.r #возвращаем результат **-
квадратный корень

    def move(self):  #движение шара

        # столкновение со стенами
if (self.x + self.r + self.dx >= WIDTH) or (self.x - self.r + self.dx <= ZERO):
self.dx = -self.dx #центр шара + радиус + ожидаемое смещение >= ширины
экрана, то отражается
if (self.y + self.r + self.dy >= HEIGHT) or (self.y - self.r + self.dy <= ZERO):
self.dy = -self.dy #центр шара + радиус + ожидаемое смещение >= высоты, то
отражается

        # столкновение с шарами
        for ball in balls:# каждый элемент balls попадает в ball
if self.is_collision(ball):#еслишарстолкнулся
if ball.color != BAD_COLOR:  #определяется что шар хороший
                    ball.hide()#1 действие-скрытие шара
                    balls.remove(ball)#2 действие-удаление выбитого шара из списка
                    self.dx = -self.dx#3 действие-смена направления
                    self.dy = -self.dy#3 действие-смена направления
                else:  # шар плохой
                    self.dx = self.dy = 0#остановка нашего шара
        self.hide()
        self.x += self.dx #смена координат относительно центра
        self.y += self.dy #смена координат относительно центра
if self.dx * self.dy != 0:
            self.draw()


#событиемыши
def mouse_click(event):# параметр event-событие
global main_ball #объявление объекта main ball глобальным, при повторном
обращении ее нет если она локальная(был локальным)
```

```
    if event.num == 1:  # при нажатии левой кнопки рисует шар
        if 'main_ball' not in globals():  #проверяем, если шар не создан, то мы его
рисуем 1
main_ball = Ball(event.x, event.y, MAIN_BALL_RADIUS,
MAIN_BALL_COLOR, INIT_DX,
INIT_DY)#появлениешарикапринажатиимыши
            if main_ball.x > WIDTH / 2:
                main_ball.dx = -main_ball.dx
            if main_ball.y > HEIGHT / 2:
                main_ball.dy = -main_ball.dy
main_ball.draw()#1-появление шара, после нажатия мыши
        else: # если шар есть, то поворачиваем его налево
            if main_ball.dy * (main_ball.dx *10) > 0:#математическое вычисление
поворота шара
                main_ball.dy = -main_ball.dy #математическое вычисление поворота
шара
else:
                main_ball.dx = -main_ball.dx
elif event.num == 3:  # если нажата правая кнопка мыши то поворот на право
if main_ball.dy * main_ball.dx > 0:
            main_ball.dx = -main_ball.dx
        else:
            main_ball.dy = -main_ball.dy


# создание списка шариков
def create_list_of_balls(number):#количество создаваемых шариков указываем
    lst = []#создание пустого списка
    while len(lst) < number:#пока длина списка меньше заданного нами
количества
        next_ball = Ball(random.choice(range(MAX_RADIUS, WIDTH -
MAX_RADIUS)),#то мы создаем следующий шарик #ширина #x
random.choice(range(MAX_RADIUS, HEIGHT - MAX_RADIUS)),#высота #y
                         random.choice(range(MIN_RADIUS, MAX_RADIUS)),#радиус
#r
                         random.choice(COLORS)) #рандомныецвета
        is_collision = False
        for ball in lst:
            if next_ball.is_collision(ball):
                is_collision = True
                break
        if not is_collision:
lst.append(next_ball)#добавление шара в список
```

```
            next_ball.draw()#рисуется шар из списка
    return lst


#считаем плохие шары
def count_bad_balls(list_of_balls):#сосписскашаров
    result = 0
    for ball in list_of_balls:# беретсяшаризспискашаров
        if ball.color == BAD_COLOR:#есликрасный
            result += 1
    return result


# движениешара
def main():
    if 'main_ball' in globals(): #проверка, естьливглобале main ball
main_ball.move() # если создан, то шарик двигается
        if len(balls) - num_of_bad_balls == 0: #если длинна списка - количество
плохих шаров =0
canvas.create_text(WIDTH / 2, HEIGHT / 2, text="ВЫВЫИГРАЛИ!",
font="Arial 45", fill="lime")
            main_ball.dx = main_ball.dy = 0 #остановкашара
elif main_ball.dx * main_ball.dy == 0:#если смещение = 0, столкнулся с
красным шаром, то ВЫ ПРОИГРАЛИ
canvas.create_text(WIDTH / 2, HEIGHT / 2, text="ВЫПРОИГРАЛИ!",
font="Arial 45", fill="red")
root.after(DELAY, main) #функция main_ball вызывает сама себя(DELAY-
скорость передвижения шара)



root = tkinter.Tk() #создание корневого окна
root.title("Сталкивающиеся Шарики") #заголовок коневого окна
canvas = tkinter.Canvas(root, width=WIDTH, height=HEIGHT, bg=BG_COLOR)
#заданиеразмеровокна
canvas.pack() #команда отражения окна
canvas.bind('<Button-1>', mouse_click)#вводим управление клавишами мыши
canvas.bind('<Button-3>', mouse_click, '+')#вводим управление клавишами
мыши
balls = create_list_of_balls(NUM_OF_BALLS)#вызов списка шаров и задаем
их количество с помощью NUM_OF_BALLS
num_of_bad_balls = count_bad_balls(balls)#количествоплохихшаров
```

```
if 'main_ball' in globals():  #проверка если объект есть то его удаляем, для того чтоб при аварийном прерывании программы шарик
del main_ball
main() #вызовфункции
root.mainloop() #программа работает совместно с tkinter
#canvas-объкт на котором мы рисуем
```