Министерство образования и науки РФ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»

Институт математики, физики, информатики

Кафедра информатики, информационных технологий
и методики обучения информатике

ОБУЧЕНИЕ АЛГОРИТМАМ КОМПЬЮТЕРНОЙ ГРАФИКИ СРЕДСТВАМИ БИБЛИОТЕКИ TURTLE В PYTHON УЧАЩИХСЯ 7-х КЛАССОВ

Выпускная квалификационная работа

Допущено к защите
Зав. кафедрой Сардак Л. В.
«26» июня 2023 г. _____

Исполнитель: Поснова Екатерина Федоровна
обучающийся группы МиИ-1801
_____

Руководитель: Сардак Л.В.
кандидат педагогических наук,
доцент, зав. кафедрой ИИТ и МОИ
_____

Екатеринбург – 2023

Оглавление

Введение

В настоящее время, одной из основных задач школы является не просто обучить школьника навыкам программирования, а сформировать и, в последствии, развить логическое и алгоритмическое типы мышления, как одни из необходимых, в современном обществе, для дальнейшей профессиональной деятельности, и не только в сфере IT-технологий. Для этого, в школьную дисциплину «информатика» включён раздел «алгоритмизация и программирование». И, своё знакомство с ним, школьники, начинают со знакомства и работы с учебными исполнителями, а именно построения каких-либо изображений их средствами: то есть развивают умения написания алгоритмов для создания компьютерной графики.

Большинство школьных учебно-методических комплексов уже не менялись на протяжении довольно долгого периода времени поэтому, основной раздел, который предполагает знакомство школьников с алгоритмизацией, на начальных этапах обучения, основан на использовании различных учебных исполнителей и их возможностей для построения изображений в таких средах как: КуМир, ЛогоМиры, Scratch и другие. И, вместе с обучением школьников навыкам управления учебными исполнителями, происходит изучение одного из языков программирования, используемого в данных средах чаще всего – алгоритмического, либо ему подобного языка, использующего русскую нотацию и являющегося структурным. Что является объективным минусом, так как такой язык программирования, в дельнейшем, нигде не используются, являясь по своей сути «мёртвым» языком. А обучать школьников алгоритмизации можно на объектах разной природы, например, используя компьютерную графику. Поэтому возникает необходимость использования одного из современных, востребованных и объектных языков программирования, который позволят через реализацию написания алгоритмов компьютерной графики сформировать и развить у школьников соответствующие навыки алгоритмизации, а также послужат надёжной базой в дальнейшем, при изучении тем по программированию. Этим и обусловлена актуальность данной работы.

Для того, чтобы выбрать язык программирования, преподавателю важно учитывать не только его простоту и популярность, но и возможности для реализации алгоритмов управления учебным исполнителем, в частности для написания алгоритмов создания компьютерной графики. Одним из таких языков программирования может послужить Python, который, являясь одним из самых популярных, содержит в себе модуль turtle – являющийся неким аналогом учебного исполнителя Черепашка из других сред.

Объект исследования – процесс обучения программированию учащихся 7-х классов.

Предмет исследования – методика обучения алгоритмам компьютерной графики средствами программирования на Python учащихся 7-х классов.

Цель работы – разработать серию уроков по освоению алгоритмов создания растровых изображений путем программирования исполнителя «Черепашка» в Python.

Задачи:

1. Проанализировать использование различных программных решений для построения растровых изображений средствами программирования в утвержденных учебных программах по информатике.
2. Сопоставление инструментария среды КуМир и библиотеки turtle языка программирования Python.
3. В соответствии с выбранным содержанием и языком программирования провести разработку серии уроков по теме «Алгоритмизация. Исполнитель Черепашка» инструментами языка программирования Python.
4. Подготовить сопроводительную документацию по использованию серии уроков для учителей школы.

Глава 1. «Алгоритмизация» в школьном курсе информатики и выбор базового языка программирования

1.1 Место раздела «Алгоритмизация» в школьном курсе информатики

При подготовке рабочих программ и разработке поурочного планирования по предмету «информатика» педагогу необходимо опираться на ФГОС основного общего образования, который закрепляет «развитие алгоритмического мышления как необходимого условия профессиональной деятельности в современном обществе; понимание сущности алгоритма и его свойств» [45, с. 82] как один из результатов освоения предмета «информатика».

Для достижения этого результата педагогу также необходимо сформировать у школьников «умение составлять, выполнять вручную и на компьютере несложные алгоритмы для управления исполнителями (Черепашка, Чертёжник); создавать и отлаживать программы на одном из языков программирования (Python, C++, Паскаль Java, C#, Школьный Алгоритмический Язык), реализующие несложные алгоритмы обработки числовых данных с использованием циклов и ветвлений; умение разбивать задачи на подзадачи, использовать константы, переменные и выражения различных типов (числовых, логических, символьных); анализировать предложенный алгоритм, определять, какие результаты возможны при заданном множестве исходных значений» [45, с. 82].

По завершению изучения предмета «информатика» у обучающихся должны быть сформированы навыки алгоритмического мышления, под которыми будем понимать «совокупность мыслительных действий и приемов, нацеленных на решение задач, в результате которых создается алгоритм, являющийся специфическим продуктом человеческой деятельности» [46, с. 46]. В современных реалиях, данный навык, будет полезен не только в сфере программирования, но и в любой другой сфере профессиональной деятельности.

Для формирования навыков алгоритмического мышления, в школьном предмете «информатика» существует раздел, посвященный алгоритмизации и программированию, в котором школьники знакомятся с одними из основных и важных понятий – алгоритм и исполнитель. Безусловно, «исполнитель – центральное понятие школьного курса информатики, изучаемое на всех ступенях обучения» [38, с. 210]. Так как позволяет наглядно демонстрировать выполнение некоторых алгоритмических конструкций, что значительно упрощает процесс формирования навыков алгоритмизации у обучающихся.

Также отметим, что, согласно обновлённому ФГОС основного общего образования 2022 года, весь школьный курс информатики был разделён на 4 больших блока:

1. Цифровая грамотность.
2. Теоретические основы информатики.
3. Алгоритмы и программирование.
4. Информационные технологии.

Рассматриваемый нами раздел, который касается исполнителя «Черепашка», алгоритмизации и компьютерной графики, в полной мере относиться к третьему блоку, а также затрагивает второй.

Некоторые учебно-методические комплексы предполагают, что изучение предмета «информатика» может быть начато с 5-го класса, а, следовательно, и знакомство школьников с алгоритмами и исполнителями происходит раньше, но большинство из них составлены для учеников 7-9 классов, так, темы, которые напрямую касаются алгоритмов, компьютерной графики, исполнителей и управления ими посредством написания программного кода, могут быть изучены школьниками гораздо позднее.

На фоне этого, появляется необходимость провести анализ действующих рабочих программ и используемых в школах учебно-методических комплексов, для выявления количества часов, которое отводится на изучение и работу с исполнителями и тему, посвящённую алгоритмам в целом, а также класса в котором ученики начинают своё знакомство с ними.

Рассмотрим УМК авторов Босовой Л.Л. и Босовой А.Ю., у данных авторов существуют учебники для 5-9 классов, причём, стоит отметить, что темы, изучаемые в 5-6 классах, дублируются в 7-8-х классах – это вызвано тем, что не во всех школах изучение информатики начинается с 5 класса. В данном УМК, разделы, посвященные алгоритмам, носят название «Алгоритмика», в учебниках для 6 класса, и «Алгоритмы и элементы программирования», в учебниках для 8 класса. В целом, на изучение данной темы отводиться 10 и 20 часов соответственно, а на знакомство с исполнителями, средами и алгоритмами их программирования, отводят 3-5 часов в 6-ом классе и 4 часа для изучения в 8-ом классе.

Авторы Поляков К.Ю. и Еремин Е.А., предлагают УМК по информатике для 7-9 классов, полагая что изучения информатики начинается только с 7 класса, иными словами, учебники для 5-6 классов отсутствуют. В данном УМК раздел, который посвящён алгоритмам, носит название «Алгоритмы и программирования», и изучается на протяжении всего периода обучения, то есть с 7 по 9 классы. На всю тему в целом выделяют 27 часов, а для знакомства школьников с исполнителями и обучению работы с ними, отводится всего 3 часа, и исключительно в 7 классе, далее этот раздел никак не рассматривается.

В своём УМК Семакин И.Г. и др., который также предназначен только для 7-9 классов, раздел, посвящённый алгоритмам, называет «Управление и алгоритмы», изучение которого, школьниками, начинается только в 9 классе, автор обосновал это тем, что все темы, касающиеся алгоритмов, школьники должны изучать последовательно и только за один год обучения. Так как, согласно данному УМК, алгоритмизацию школьники начиняют и заканчивают изучать в 9 классе, а, следовательно, и с исполнителями знакомятся именно в этот период, и из общего объёма – 12 часов, на исполнителей отводиться 3-4 часа.

Автор УМК для 7-9 классов, Гейн А.Г., предлагает начать изучение алгоритмизации в 8 классе, в разделе, который носит название «Алгоритмы и исполнители». В целом, логика тем и их последовательность довольно похожа на логику изложения темы в УМК Босовой Л.Л. и Босовой А.Ю. А на изучение данного раздела, автор выделяет 14 учебных часов, из них, на знакомство и работу с исполнителями отводится всего 2-3 часа.

Угринович Н.Д. и др., в своём УМК, предназначенном для 7-9 классов, не использует стандартных исполнителей (Черепашка, Чертёжник, Робот и др.), и, в разделе, «Основы алгоритмизации и ООП», сразу переходит к использованию сред и языков программирования, выделяя для этого 15 часов в 9 классе.

Таким образом, можно заключить, что в большинстве УМК, знакомство обучающихся с исполнителями и формирование у них навыков алгоритмизации, в целом, начинается в 7-8 классе, то есть с началом изучения предмета «информатика». Для наглядности проделанной работы, представим результаты проделанного анализа в виде сравнительной таблицы (табл. 1).

Таблица 1.

Сравнительный анализ УМК

Критерий / **Учебник**	**Класс**		**Количество часов для изучения темы «Алгоритмы»**		**Количество аудиторных часов на изучения исполнителей**	
Босова Л.Л. Босова А.Ю.	6	8	10	20	3-5	4
Поляков К.Ю. Еремина Е.А.	7		27		3	
Семакин И.Г. и др.	9		12		3-4	
Гейн А.Г.	8		14		2-3	
Угринович Н.Д. и др.	9		15		-	

Независимо от класса и выбранного УМК, формирование алгоритмических навыков у школьников начинается с введения центрального понятия «алгоритм», которое происходит от латинского «algorithmi», которое берёт своё начало от имени среднеазиатского математика Аль-Хорезми и означает «конечную совокупность точно заданных правил решения некоторого класса задач или набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи» [10]. И от этого понятия школьники плавно переходят к знакомству с исполнителем алгоритмов, под которым понимают «средство (или инструмент) для решения поставленной задачи» [38, с. 210].

При переходе к изучению исполнителей и работы с ними, заметим, что каждый автор учебников в своих УМК использует различных исполнителей. «Босова Л.Л. в своих УМК в качестве примеров формальных исполнителей использует следующие: «Черепаха», «Чертёжник» и «Робот». В УМК Макаровой Н.В. присутствует только учебный исполнитель «Черепашка». Семакин И.Г. для выполнения учащимися практических работ использует исполнитель «Стрелочка», входящий в комплекс ЕКЦОР. Гейн А.Г. предлагает учебного исполнителя «Паркетчик» - простой и наглядный исполнитель, реализующий принцип «от простого к сложному». В УМК Угринович Н.Д. средой обучения алгоритмизации является VBASIC (QBASIC), при этом вышеперечисленные исполнители автором учебника не используются. Поляков К.Ю. предлагает познакомить школьников с таким учебным исполнителем, как «Черепашка»» [29, с. 39].

На основе сказанного можно заключить, что учебный исполнитель «Черепашка» является одним из самых популярным среди всех. И, чаще всего, когда школьники начинают своё знакомство и работу с данным исполнителем сталкиваются с такими понятиями как: «компьютерная графика», «алгоритмы построения компьютерной графики», «черепашья графика».

Под компьютерной графикой понимают «технологии, методы и средства для создания компьютерных изображений различного характера» [35]. Для этого используют различные алгоритмы, способы создания изображений: рисование при помощи уже имеющихся примитивов (векторная графика) и рисование при помощи отдельных пикселей (растровая графика).

Выделим основные алгоритмы создания компьютерной графики, под чем будем понимать набор инструкций и процедур, которые используются для создания, обработки и отображения изображений на компьютере. Алгоритмы компьютерной графики используют различные языки программирования (Python, C++, Java и др.), написанный на них код будет определять, как изображения будут создаваться и обрабатываться на компьютере. Все они основываются на математических принципах и используются в различных областях, включая игровую индустрию, медицину, архитектуру и другие.

В целом, с опорой на учебное пособие «Основные алгоритмы компьютерной графики» автора Вельтмандера П.В., алгоритмы создания компьютерной графики можно разделить на несколько категорий [21]:

- точечное (пиксельное) построение изображения;
- использование векторной алгебры для построения изображений;
- заливки замкнутой области;
- трансформации изображения – изменения размеров, поворот и перенос изображения;
- использование освещения;
- анимация изображения.

Рассмотрим алгоритм пиксельного построения графики, который по-другому можно назвать точечным построением растрового изображения. При данном алгоритме построение изображения происходит следующим образом:

1 Шаг. Определяется размер изображения в пикселях, в зависимости от пропорций изображения (ширина и высота в пикселях).

2 Шаг. Создаётся матрицы пикселей, которая представляет собой двумерный массив, в нём каждый элемент соответствует одному пикселю изображения.

3 Шаг. Задаётся цветовая характеристика для каждого пикселя. Это происходит через кодирование, которое осуществляется с помощью задания определённых числовых значений либо, в некоторых языках программирования, через указание конкретного цвета. Обычно используется палитра RGB (Red-Green-Blue), а для задания самого цвета используют HTML формат.

4 Шаг. Расположение пикселей на экране. Сопоставление координат пикселя из матрицы с реальным положением пикселя на экране.

5 Шаг. Отображается изображение на экране. Для этого компьютером происходит обработка информации из предыдущих шагов, то есть информация о цвете пикселя и его расположении.

В результате точечного (пиксельного) построения получается изображение, состоящее из множества маленьких точек (пикселей), каждый из которых имеет свой цвет и расположение на экране.

Рассмотрим, как используется векторная алгебра для создания изображений на экране компьютера: первым шагом такие геометрические объекты, как линии, кривые, простые фигуры определяются в виде математических векторов, затем используются различные математические операции, такие как сложение, вычитание, умножение и деление, для изменения формы и размера объектов на изображении. Например, для создания круга используется формула окружности, которая задается радиусом и координатами центра.

После определения геометрических объектов и применения математических операций происходит рендеринг изображения. Рендеринг – это процесс преобразования геометрических объектов и свойств, таких как цвет и освещение, в пиксели на экране. Для этого используются алгоритмы растеризации, которые преобразуют векторные объекты в пиксели на экране.

После растеризации происходит обработка изображения, такая как наложение текстур и эффектов, чтобы дать изображению более реалистичный вид. Затем изображение отображается на экране. Для создания эффекта тени используются вектора нормали, которые определяют направление света и его интенсивность. Для перемещения объекта на изображении используются вектора направления и скорости, которые определяют его движение.

Далее разберём алгоритмы заливки замкнутого объекта, под которым будем понимать создание какой-либо области и заполнения её отдельным цветом или текстурой. Иными словами, процесс, при котором всем пикселям внутри конкретной фигуры присваивается какой-либо новый цвет, отличный от начального. Существует несколько вариантов реализации данного алгоритма:

- заливка по точке – внутри замкнутой области выбирается начальный пиксель, у него и его соседних пикселей, чей цвет совпадает с выбранным пикселем, меняется цвет на новый. Затем выбирается следующий пиксель и алгоритм повторяется несколько раз, пока все пиксели в области не будут перекрашены в выбранный цвет;
- заливка по ребру – внутри замкнутой области выбирается какое-либо ребро фигуры и меняется цвет всех пикселей на новый, что лежат между начальной и конечной точками этого ребра. Затем выбирается следующее ребро и алгоритм повторяется несколько раз, пока все пиксели в области не будут перекрашены в выбранный цвет.

В результате выполнения алгоритмов заливки получается изображение, которое состоит из цветной фигуры, ограниченной контуром.

Теперь рассмотрим алгоритмы трансформации изображения, под которым будем понимать изменения размеров, поворот и перенос изображения. То есть, процесс, при котором меняется размер, ориентация, форма и другие характеристики исходного изображения. Рассмотрим реализацию некоторых алгоритмов:

- изменения размера начинается с выбора новых размеров изображения и создания пустого холста с этими размерами. Затем применяется метод интерполяции, чтобы заполнить новые пиксели на холсте на основе значений пикселей исходного изображения;
- поворот изображения начинается с выбора угла поворота и создания пустого холста с таким же размером, как и исходное изображение. Затем применяется метод аффинного преобразования, чтобы повернуть каждый пиксель на холсте на заданный угол. Этот алгоритм может использоваться для изменения ориентации изображения.

В результате выполнения алгоритмов трансформации получается изображение, которое отличается размером, формой и положением от исходного изображения.

Теперь ознакомимся с алгоритмами освещения, под которыми будем понимать расчёт и моделирование того, как свет взаимодействует с объектами на изображении, что позволяет создавать реалистичные и эффективные изображения, имитируя различные источники света и их влияние на объекты. Самыми популярными алгоритмами освящения можно назвать:

- модель Фонга – основана на том, что свет отражается от поверхности объекта в разных направлениях, и эти отражения создают различные уровни яркости и тени на объекте;

- модель Ламберта – основана на том, что свет падает на поверхность объекта и рассеивается во всех направлениях. Этот алгоритм используется для создания мягких и равномерных теней на объектах.

В результате использования алгоритмов освещения можно получить реалистичные изображения с тенями и добиться такого эффекта картинки как 3D.

Рассмотрим последний из выделенных алгоритмов – анимации изображения, под которым будем понимать процесс создания движущихся изображений. Одними из самых распространенных являются:

- интерполяция, она основана на том, что два или более изображения объединяются в одно, и затем между ними создается плавный переход. Это может быть достигнуто путем изменения позиции, размера, цвета или других свойств объектов на изображении;
- скелетная анимация, она основана на том, что объекты на изображении разбиваются на отдельные части, которые могут быть перемещены и повернуты независимо друг от друга.

Учебный исполнитель «Черепашка» позволяет реализовать алгоритмы построения изображения на основе растровой (пиксельной) графики, тем самым вводя новое понятие – черепашья графика, то есть «принцип организации <…> графического вывода, построенный на метафоре Черепахи, воображаемого <…> роботоподобного устройства, которое перемещается по экрану или бумаге и поворачивается в заданных направлениях, при этом оставляя (или, по выбору, не оставляя) за собой нарисованный след заданного цвета и ширины» [50]. То есть при написании алгоритмов для управления Черепашкой, ученики параллельно осваивают алгоритмы построение пиксельной графики – изменения цветовой характеристики конкретному числу пикселей в заданном направлении.

Для того, чтобы не только сформировать у обучающихся умения управлять учебным исполнителем «Черепашка», а также обучить алгоритмам компьютерной графики, нужно определить используемую среду, так как возможности для создания алгоритмов предоставляют ряд программных средств (например, КуМир, ЛогоМиры), а также языков программирования (например, Scratch, Python), ещё существует множество онлайн сервисов, которые можно использовать совершенно бесплатно и без предварительной установки (например, Blockly.ru).

В связи с таким разнообразием программ и сервисов возникает вопрос: какой инструментарий целесообразно использовать, для формирования навыков алгоритмизации и умения составлять алгоритмы управления учебным исполнителем «Черепашка» и обучения учеников алгоритмам построения компьютерной графики, на основе данного исполнителя. В действительности, чтобы дать обоснованный и более точный ответ на данный вопрос, «необходимо учитывать различные факторы: интуитивную понятность системы команд исполнителя, читаемость получаемой программы, наличие программного обеспечения, использование исполнителя в заданиях ОГЭ и ЕГЭ» [44, с. 11]. Поэтому, проведём сопоставительный анализ возможностей и инструментария КуМир, ЛогоМиры, Scratch и Python, оформив полученные результаты в виде таблицы (табл. 2).

Таблица 2.

Сопоставительный анализ возможностей

Возможность	**КуМир**	**ЛогоМиры**	**Scratch**	**Python**
Возможность использования на ЕГЭ	+	-	-	+
Перспективы использования в дальнейшей профессиональной деятельности	-	-	-	+
Удобство редактора кода	-	-	+	+
Простота и понятность кода	-	-	+	+
Единица перемещения Черепашки равна одному пикселю	-	+	+	+
Язык кода	русский	русский	русский	английский
Возможность использовать онлайн среду	-	-	+	+

Для того, чтобы провести сопоставительный анализ сложности написанного кода, составим алгоритм для создания одинакового изображения в рассматриваемых средах, используем для этого изображение «домика» (рис. 1).

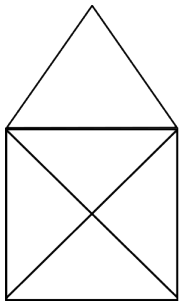


Рисунок 1. Изображение, для создания которого, нужно написать алгоритм

Так как исполнитель «Черепашка» направлен на реализацию алгоритма построения пиксельного изображения, который был рассмотрен ранее, то и её рисование будем рассматривать как изменение цветовой характеристики определённого количества пикселей в заданном направлении. Таким образом, задача на построение рассмотренного выше изображения, из создания непрерывной ломаной кривой переходит в задачу на непрерывное изменение цветовой характеристики конкретного количества пикселей – длина отрезков, в заданных направлениях – угол между отрезками.

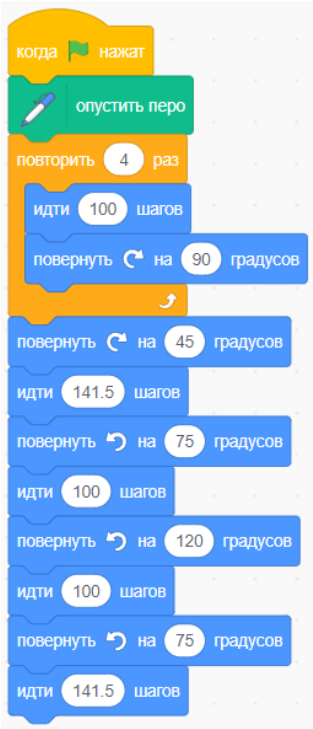
Напишем такой алгоритм для создания точечного изображения в средах КуМир (рис. 2.а), ЛогоМиры (рис. 2.б), Scratch (рис. 2.в) и Python (рис. 2.г) для проведения сопоставления сложности получившихся листингов.

```
использовать Черепаха
алг
нач
 . опустить хвост
 . нц 4 раз
 . . вперед(100)
 . . вправо(90)
 . кц
 . вправо(45)
 . вперед(141.5)
 . влево(75)
 . вперед(100)
 . влево(120)
 . вперед(100)
 . влево(75)
 . вперед(141.5)
кон
```

а.

```
по
повтори 4 [вперед 100 направо 90]
направо 45
вперед 141,5
налево 75
вперед 100
налево 120
вперед 100
налево 75
вперед 141,5
```

б.



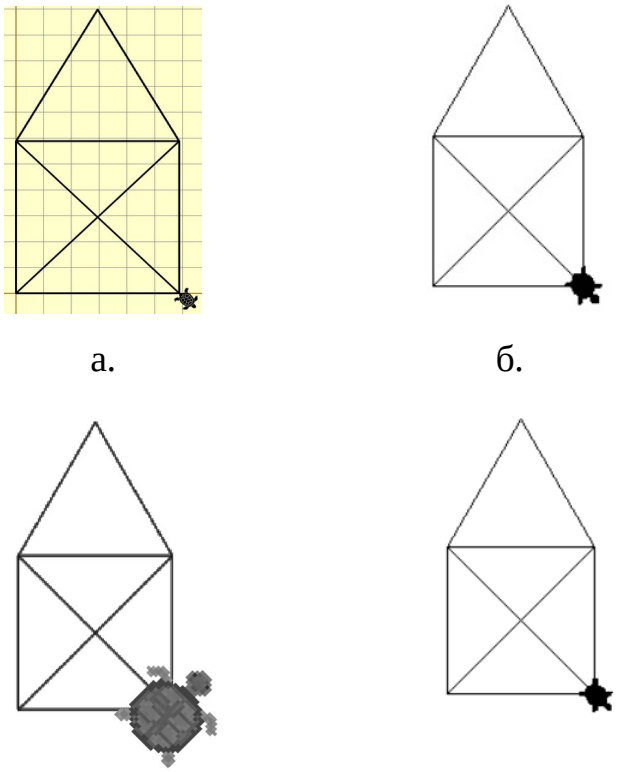
в.

```
from turtle import *
for i in range(4):
    fd(100)
    rt(90)
rt(45)
fd(141.5)
lt(75)
fd(100)
lt(120)
fd(100)
lt(75)
fd(141.5)
```

г.

Рисунок 2. Листинги программ, для создания изображения
а. Алгоритм, написанный в КуМир
б. Алгоритм, написанный в ЛогоМирах
в. Алгоритм, написанный в Scratch
г. Алгоритм, написанный в Python

Далее, для сопоставления результатов выполнения кода обратим своё внимание на полученные изображения в КуМир (рис. 3.а), ЛогоМиры (рис. 3.б), Scratch (рис. 3.в) и Python (рис. 3.г) и убедимся в их полной идентичности.



а. б.

в. г.

Рисунок 3. Результат выполнения программ
а. Изображение, полученное в КуМир
б. Изображение, полученное в ЛогоМирах
в. Изображение, полученное в Scratch
г. Изображение, полученное в Python

На основе написанных программ и полученных результатов, можно провести сопоставление сложности написанных кодов, под которым будем понимать не только простоту и понятность написанного для школьников, но и объём конструкций всего листинга в целом, а также необходимость каких-либо дополнительных подключений. Так, для работы с Черепашкой в КуМир и Python нет необходимости в дополнительных настройках – есть возможность сразу писать код, и, при запуске, сама Черепашка и её окружение сразу отобразятся на экране. А в ЛогоМирах и Scratch необходимо предварительно настроить саму Черепашку – создать её на поле и, при необходимости, разместить её нужным образом, иными словами, несмотря на довольно маленький алгоритм управления, написанный в ЛогоМирах, и простоту блочного кода в Scratch сложности могут возникнуть с настройкой и созданием самой Черепашки – школьники больше времени будут тратить на её настройку, чем на работу с ней. Также, обратим внимание на автоматизированный ввода команд, подсветку синтаксиса и выделение ошибок, которые в полном объёме присутствует только в Python, что в разы ускоряет написание программ и поиск семантических и синтаксических ошибок, иными словами – экономит время, а в остальных средах прописывать команды приходиться полностью самостоятельно, за исключением Scratch, где нужно использовать блоки. Также обратим внимание, что все вышеперечисленные удобства полностью отсутствуют в программе ЛогоМиры. Далее, стоит отметить, что только Python использует английскую нотацию, однако это не может служить минусом, так как используемый набор команд достаточно прост, и под силу школьнику, даже немного знающему английский язык. А также, в дальнейшем, при изучении различных языков программирования, все они используют английскую нотацию, поэтому не стоит приучать учеников к русскоязычному листингу.

Так как черепашья графика напрямую связана с компьютерной графикой и алгоритмами её создания, то целесообразно использовать среды, где единица перемещения Черепашки равна одному пикселю, то есть позволяет реализовать алгоритмы пиксельной (растровой) графики. КуМир же позволяет только передать идею, но не суть такой графики, так как в редакторе используется искусственно увеличенное поле для Черепашки, то есть её единица перемещения – это не один пиксель. Таким образом передвижение Черепашки в КуМир можно трактовать, как перемещение на шаг с некоторой длиной и приданием цвета некоторому количеству пикселей на экране, что важно – не равному длине шага, что не укладывается в концепт алгоритмов пиксельного создания изображения.

Для дельнейшей профессиональной деятельности, в сфере программирования, ученикам важно передавать актуальные знания, в связи со стремительно меняющимся современным миром, поэтому обратимся к статистике по самым популярным языкам программирования.

На одном из популярных сайтов, с ежемесячно обновляемой статистикой – TIOBE, язык программиования Python занимает первое место, имея рейтинг 14,51% (рис. 4). На данном сайте статистика собирается автоматически и рейтинги формируются в зависимости от: количества программистов, знающих язык, количества предлагаемых курсов на различных платформах, для изучения языка, количества вакансий от работодателей на различных сервисах, количества поисковых запросах пользователей в сети Интернет.

Apr 2023	Apr 2022	Change	Programming Language	Ratings	Change
1	1		Python	14.51%	+0.59%
2	2		C	14.41%	+1.71%
3	3		Java	13.23%	+2.41%
4	4		C++	12.96%	+4.68%
5	5		C#	8.21%	+1.39%

Рисунок 4. Рейтинги некоторых языков программирования на сайте TIOBE

Обратимся к другому, не менее популярному сайту – PYPL, на нём, статиистика по языкам, также обновляется ежемесячно и собирается в зависимости от частоты запросов пользователей в поисковых системах. Согластно этому рэйтингу, в момент обращения, Python, также занимает лидирующую позицию – первое место по количеству запросов в сети Интернет. (рис. 5).

Worldwide, Apr 2023 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.43 %	-0.8 %
2		Java	16.41 %	-1.7 %
3		JavaScript	9.57 %	+0.3 %
4		C#	6.9 %	-0.3 %
5		C/C++	6.65 %	-0.5 %

Рисунок 5. Рейтинги некоторых языков программирования на сайте PYPL.

Теперь рассмотрим сервис IEEE Spectrum, который собирает информацию не только исходя из поисковых запросов, но и обращается к другим статистическим сервисам, то есть систематизирует всю информацию в одном месте. И, не трудно убедиться, что Python, также, занимает лидирующую позицию (рис. 6).

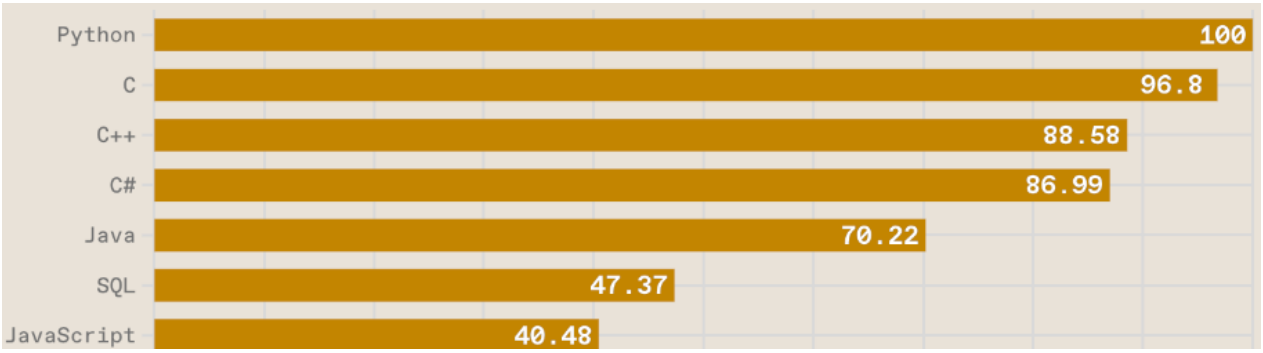


Рисунок 6. Рейтинги некоторых языков программирования на сайте IEEE Spectrum

Помимо англоязычных сайтов, которые просто отражают статистику популярности языков, обратимся к русскоязычному сервису Zarplan.com, который составляет рейтинг языков, исходя из возможных вакансий, предлагаемых на территории России. Отметим, хоть Python не занимает первого места, он все равно остается в тройке самых популярных языков (рис. 7).

Рейтинг языков программирования по количеству вакансий

Язык программирования	Количество вакансий для языка программирования
SQL	16713
Python	8199
JavaScript	6607
Java	5246
C++	3476

Рисунок 7. Рейтинги некоторых языков программирования на сайте Zarplan.com

Таким образом, можно сделать выводы, что язык Python, являясь одним из популярных и востребованных языков программирования, что безусловно важно для дальнейшей профессиональной деятельности учеников в сфере программирования, и, позволяя реализовать ровно те же функции, что и КуМир, ЛогоМиры и Scratch, для управления исполнителем Черепашка, может послужить им полноценной заменой.

На основе проделанного анализа можно сделать заключение, что создание изображений при помощи учебного исполнителя Черепашка напрямую связано с алгоритмами точечной (пиксельной) графики, то есть процессом изменением цвета конкретного числа пикселей в заданном направлении. Что и позволяет, в полной мере, реализовать язык программирования Python, поэтому, на основе представленного выше материала, нами обоснован выбор данного языка программирования.

1.2 Сопоставление возможностей реализации «Черепашьей графики» в графическом модуле Turtle языка программирования Python с учебным алгоритмическим языком в КуМир

Несмотря на то, что алгоритмы по созданию «черепашьей графики» позволяют организовать множество сервисов и сред, в нашей дальнейшей работе будем проводить сопоставительный анализ возможностей среды КуМир и языка программирования Python. Данный выбор сделан на основе того, что в дальнейшем обучающимся, которые выбрали сдавать ОГЭ по информатике, для решения ряда некоторых задач, вынуждены использовать знания либо алгоритмического языка, который как раз и используется для написания программ в среде КуМир, либо одного из языков программирования (Python, Pascal, C++, Basic), на выбор ученика, а также, для решения заданий из второй части им необходимо написать алгоритм для управления таким исполнителем как Робот в среде КуМир, либо программу на выбранном языка программирования. На основе вышесказанного, можно заключить, что в школах, чаще всего, для введения в раздел, посвящённый алгоритмам, педагоги, используют учебную среду КуМир.

Для того, чтобы писать алгоритмы по управлению учебным исполнителем Черепашка, в среде КуМир, обучающимся необходимо знание алгоритмического языка программирования: синтаксиса, основных конструкций, а также специальных команд для управления Черепашкой. При запуске среды программирования появляется окно программы, поделённое на две равные части: левая половина предназначена для написания программного кода и его отладки, а правая для демонстрации самой Черепашки, её окружения и координатной сетки (рис. 8).

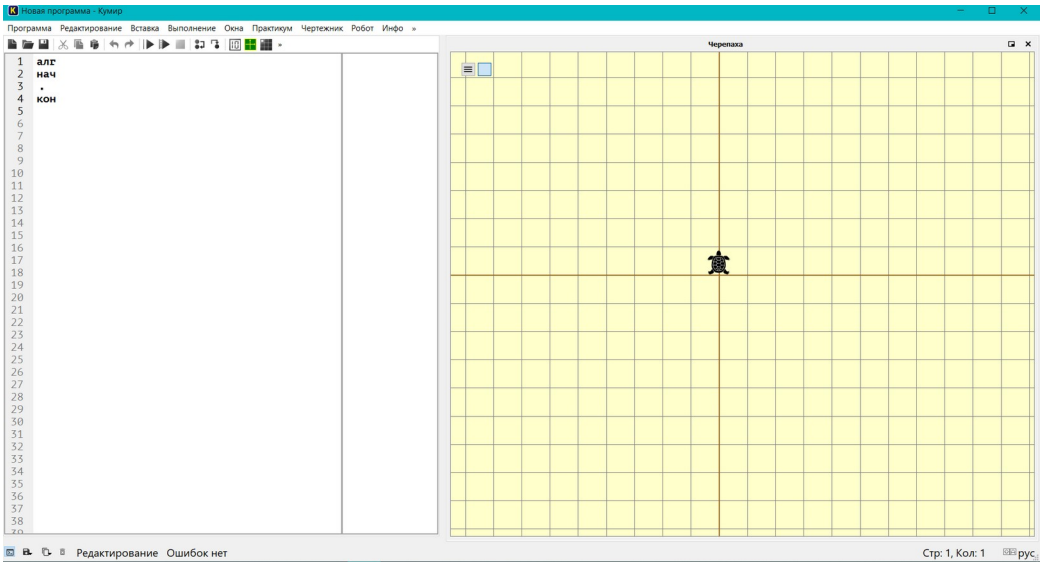
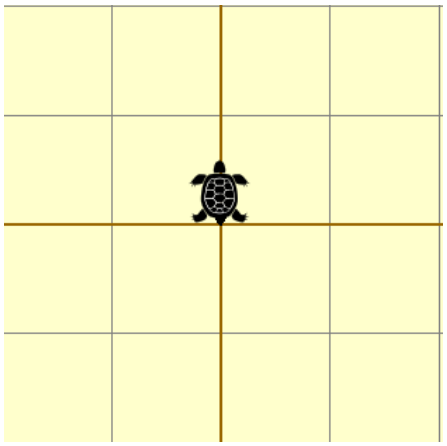
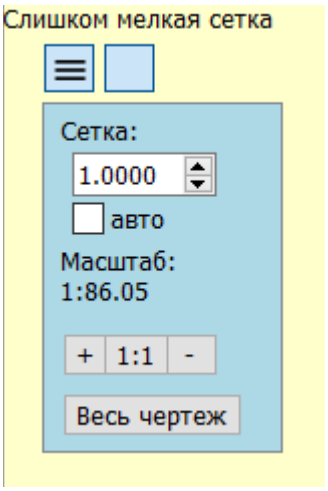


Рисунок 8. Окно при запуске КуМир.

Рассмотрим подробнее правую часть окна, отводимую для отображения Черепашки, начальное расположение хвоста которой совпадает с перекрестьем координатных осей – начальной точкой с координатами (0; 0) (рис. 9.а). Также отметим, что размер сетки и единица перемещения Черепашки не совпадают с реальным размером одного пикселя на экране монитора, несмотря на то, что в справочной информации по исполнителю единицей его передвижения указан пиксель. Если очень необходимо, данного соответствия возможно добиться искусственным путём, но на приближенном уровне, используя для этого настройку масштаба изображения, чем мы и попробовали заняться в следствии чего получили характеристики, которые изображены на рисунке ниже (рис. 9.б), но, при такой искусственной настройке, достаточно сложно оценить достоверность полученного результата, то есть равенство единиц перемещение черепашки по экрану и одного пикселя на мониторе. Поэтому, можно опытным путём убедиться, что Черепашка из КуМир позволяет только передавать идею пиксельной графики, но не её суть в целом.

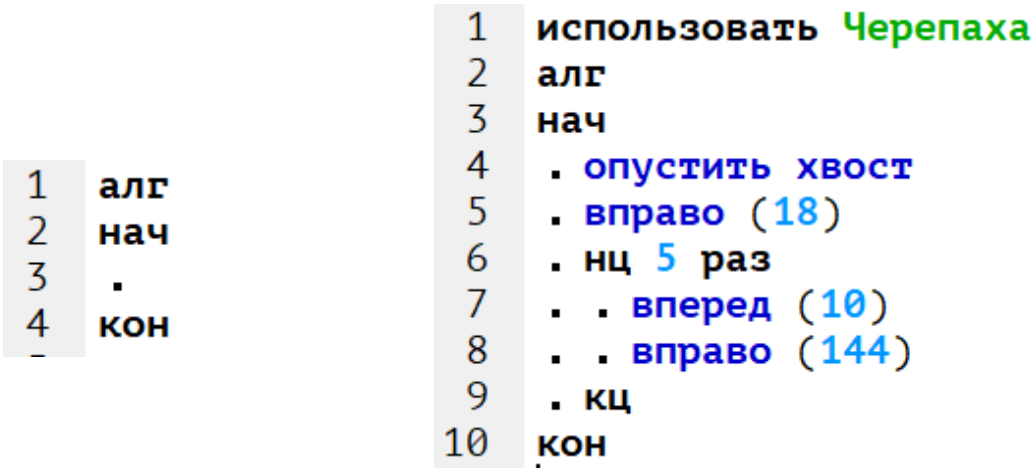




а. б.

Рисунок 9. Левая половина окна программы КуМир.
а. Расположение Черепашки и координатные оси.
б. Настройка масштаба и размеров сетки.

Теперь рассмотрим левую часть экрана, ту, которая предназначена для написания программного кода и его отладки, а также для управления учебными исполнителями. В ней уже есть совсем небольшая заготовка программного кода, который появляется всегда, в независимости от исполнителя (рис. 10.а). Однако, для того, чтобы любая написанная программа могла быть запущенной с тем условием, чтобы написанные команды не считались ошибочными, в КуМир, необходимо обязательно указывать, какого исполнителя нужно использовать. В нашем случае нужно подключить Черепашку, и поэтому необходимо использовать следующую строчку: «**использовать Черепаха**» и далее уже можно, используя её синтаксис, конструкции алгоритмического языка и команды для управления Черепашкой, чтобы писать программы (рис. 10.б). Отметим, что присутствует довольно приятная подсветка синтаксиса, что безусловно удобно, так как ученик сразу может оценить правильность ввода команд и конструкций языка.



б **а**

Рисунок 10. Правая половина окна программы КуМир.
а. Начальный макет программы.
б. Пример самостоятельно написанной программы, которая реализует алгоритм рисования «звёздочки».

Отметим, что, для запуска программ в КуМир, нет необходимости в предварительном сохранении файла программы. Это означает, если ученику необходимо написать небольшую программу, нет нужды в её сохранении для запуска.

Такой, минимально-необходимый, набор возможностей предоставляет среда КуМир для реализации управления учебным исполнителем Черепашка. Являясь более популярной программной средой, используемой в школьном курсе информатики, нежели другие аналоги, позволяющие также реализовать алгоритмы Черепашьей графики.

Далее рассмотрим возможности языка программирования Python. Для того чтобы писать алгоритмы управления Черепашкой обучающимся нет необходимости в глубоких знаниях данного языка, так как основные команды довольно просты и даже могут рассматриваться учениками, как английские слова, то есть чтобы понять назначение команды ученику лишь необходимо понимать значение соответствующего слова на английском языке. При запуске среды программирования, на экране появляется окно, которое содержит в себе небольшую информацию о версии продукта (рис. 11).

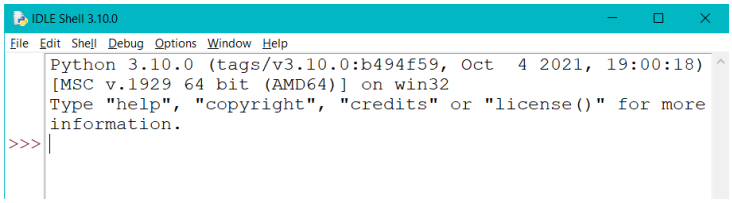


Рисунок 11. Стартовое окно при запуске IDLE Python.

Данное окно уже содержит в себе возможность написания программного кода и его интерпретации, и, написанные таким образом программы не требуют предварительного сохранения, но имеют довольно неудобный вид и, при самостоятельном сохранении, в файл кода, также записываются результаты запуска программы. Поэтому, для дальнейшего написания алгоритмов стоит создавать новый файл, через меню File – New File, и писать код в появившемся, пустом окне (рис. 12).

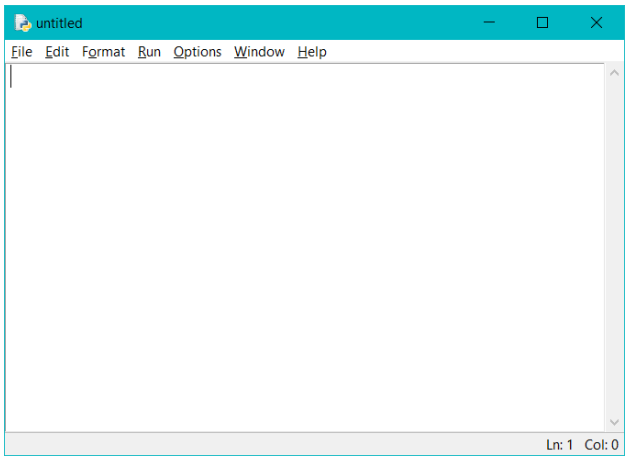


Рисунок 12. Окно для написания программного кода в IDLE Python.

Окно, которое демонстрирует Черепашку и её окружение отсутствует и появляется только после запуска, интерпретации и выполнения написанной программы, при условии, что программный код был написан правильно и без синтаксических и семантических ошибок.

На Python, также, как и в КуМир, для того, чтобы использовать возможности языка по управлению исполнителем Черепашка, нужно первой строчкой, условно, указывать какой исполнитель должен быть использован, а в терминологии языка программирования – это называется подключить библиотеку (модуль). Для этого нужно использовать строку – **`import turtle`**. Но, при таком подключении, для написания кода, придётся каждый раз указывать имя используемого модуля, то есть каждый раз прописывать строку вида: **`turtle.command_name()`**, а также создавать объект данной библиотеки. Это не всегда удобно и будет выглядеть довольно громоздко, поэтому стоит использовать импорт следующего вида: **`from turtle import *`**, подключение таким образом позволяет использовать альтернативный импорт библиотеки (модуля), и при написании кода можно использовать все команды без многократного указания имени самого модуля и дополнительного создания объекта, то есть использовать строки вида: **`command_name()`**. В силу особенности используемого языка, также, можно добавить строчки, которые позволяют провести настройку окна, в котором появиться Черепашка, используя для этого следующие строки вида: **`turtle.setup(200,200)`** и **`turtle.screensize(150,150)`**, либо **`setup (200,200)`** и **`screensize(150,150)`** (в зависимости от использованного импорта библиотеки), которые позволяют задать размеры окна, которое появиться при запуске, и объём области по которой может перемещаться Черепашка, то есть количество пикселей рабочего поля. Если выйдет так, что размеры рабочей области гораздо больше размеров рабочего окна, то появиться полоса прокрутки. Также важно помнить, при задании размеров окна, что всё размеры задаются в пикселях и только в них.

Для того, чтобы увидеть окно с Черепашкой и её окружением (рис. 13), а также сравнить его с окружением Черепашки из КуМир, напишем небольшую программу, используя вышеуказанные команды:

```
from turtle import *
```

```
setup(500,500)
screensize(500,500)
shape("turtle")
```

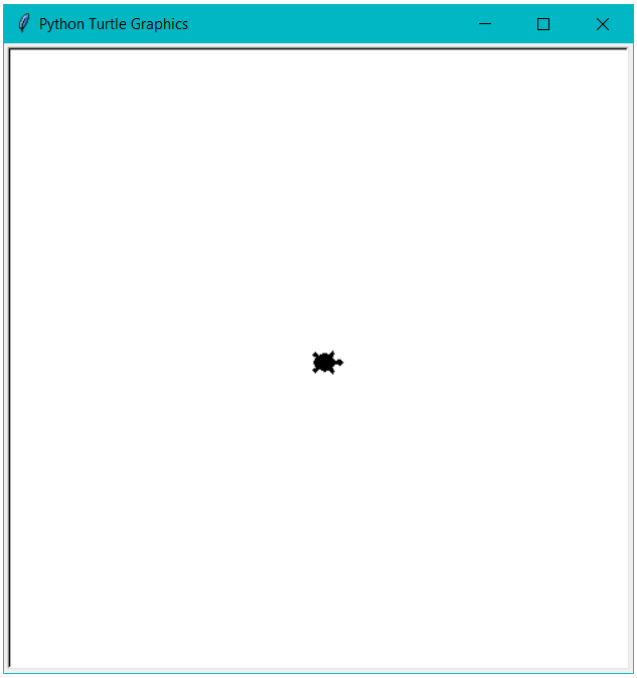


Рисунок 13. Окно с «Черепашкой» в Python

Заметим, что данное поле уже отличает отсутствие сетки и возможности его масштабирования, а также Черепашка направлена вправо, в то время как в КуМир она направлена вверх, но это нужно учитывать, только лишь при написании алгоритмов.

Заметим, что была использована строка «**`shape("turtle")`**». Всё из-за того, что после написания программы, которая подразумевает не только настройку окна, а и алгоритм для рисования изображения, и её запуска, Черепашка, которая отобразится на экране будет совсем на неё не похожа, а, по умолчанию, будет иметь форму стрелочки. Для того, чтобы эту форму изменить, нужно использовать команду: **`shape("name")`**, где на месте **`name`**, в кавычках, указывается один из возможных видов Черепашки. Продемонстрируем основные виды Черепашек и команды, которые нужно использовать, чтобы их применить, результаты, для наглядности соберём в таблице (табл. 3).

Таблица 3.

Команды изменяющие внешний вид черепашки

Команда	Внешний вид Черепашки
`shape ("classic")` (По умолчанию)	
`shape ("arrow")`	
`shape ("turtle")`	
`shape ("circle")`	
`shape ("square")`	
`shape ("triangle")`	

Такой достаточный набор инструментов предоставляет модуль turtle языка Python – окно для написания программного кода, также с подсветкой синтаксиса, и окно с Черепашкой, размеры которого можно самостоятельно настроить.

Прежде чем перейти к анализу возможных команд и конструкций языков, проанализируем, задачный материал из школьных учебников различных авторов. Каждый из авторов учебно-методических комплексов выбирает какого-нибудь исполнителя и делает на нём больший акцент, чем на других. Так, например, Босова Л.Л. больше акцентирована на исполнителе «Робот». На фоне этого, на исполнителя «Черепашка» отводиться довольно мало не только часов, но и задачного материала.

Поэтому все задачи, рассматриваемые в учебниках, которые нам удалось найти, для исполнителя Черепашка, выглядят подобным образом: «подумайте, какая фигура появиться на экране после выполнения Черепашкой следующего алгоритма: повтори 12 направо 45 вперед 20 направо 45» [17]. То есть, как таковые, задания на самостоятельное построение алгоритмов и работу в среде КуМир, практически отсутствуют, обучающимся просто нужно прочитать алгоритм и нарисовать картинку у себя на листочке. Исходя из этого, что бы не выбрал учитель КуМир или Python ему придётся самостоятельно подбирать и формулировать задания для учеников, либо, используя сеть Интернет, использовать наработки других учителей.

За исключением УМК Полякова К.Ю., который не только предлагает на выбор несколько самостоятельно разработанных сред с исполнителями, но и даёт возможность ученикам поработать с ними, посредством активного использования его сайта. Где, для Черепашки, выделен целый раздел и комплекс различных заданий, решаемых в онлайн среде Blockly. Там, задания, разбиты на несколько модулей (тем): линейные алгоритмы, циклы «повтори N раз», вложенные циклы, процедуры, переменные, процедуры с параметрами. И каждый из таких модулей содержит в себе 10-12 заданий, в которых присутствует градация сложности. В среде Blockly, алгоритмы для управления Черепашкой создаются при помощи блоков, как в Scratch, но внизу присутствует окно, которое демонстрирует как бы выглядели команды, если бы они были написаны на одном из языков программирования. Так основной выбор идёт за учителем – либо использовать данную среду с готовыми задачами и писать алгоритмы в ней, либо использовать задачный материал и адаптировать его для использования в среде КуМир, либо Python.

Поскольку, в первом параграфе было заключено, что Python может полностью заменить Черепашку из различных сред и программ, в частности и Черепашки из КуМир, то для большей наглядности, проведём сопоставление команд, которые использует язык программирования Python. Для удобства все результаты систематизируем и представим в виде таблицы, чтобы более наглядно продемонстрировать их отличие (табл. 4).

Таблица 4.
Сопоставление команд, используемых в КуМир на язык Python

Назначение команды	КуМир	Python
Подключение исполнителя	**использовать Черепаху**	**import turtle** **from turtle import ***
Не оставлять след на экране	**поднять хвост**	**up()** *либо* **pu()**
Оставлять след на экране	**опустить хвост**	**down()** *либо* **pd()**
Показывать Черепашку	нажать кнопку в верхнем левом углу поля с черепашкой	**showturtle()** *либо* **st()**
Скрыть Черепашку	нажать кнопку в верхнем левом углу поля с черепашкой	**hideturtle()** *либо* **ht()**
Движение вперёд	**вперед (цел а)**	**forward(int or float)** **fd(int or float)**
Движение назад	**назад (цел а)**	**back(int or float)** **bk(int or float)** **backward(int or float)** **fd(-int or -float)**
Поворот вправо	**вправо (вещ угол)**	**right(int or float)** **rt(int or float)**
Поворот влево	**влево (вещ угол)**	**left(int or float)** **lt(int or float)**
Перемещение в начало	-	**home()**
Перемещение в точку	-	**setposition(x, y)** **setpos(x, y)** **goto(x, y)** **x, y – int or float**
Конструкция цикла for	**нц цел а раз** **кц**	**for i in range(int):**
Конструкция цикла while	**нц пока *условие*** **тело цикла** **кц**	**while (*условие*)** #тело цикла
Конструкция подпрограмм и их вызов	**алг название_алг_1** **нач** **название_ алг_2** **кон** **алг название_ алг_2** **нач** **кон**	**def name (var)** #тело функции #вызов функции **name (var)**
Единица в шаге	Клетка настраиваемой сетки	Пиксель

На основе данной таблицы можно сделать выводы, что модуль turtle языка Python, предоставляет ровно такой же набор команд, для управления исполнителем Черепашка, местами позволяя реализовать больше (перемещение в точку, начало координат, настройка размеров окна, рабочей области). Но, основным преимуществом является возможность реализации алгоритмов построения пиксельной (растровой или точечной) графики в чистом виде, так как единицей передвижения Черепашки является пиксель, а не искусственно увеличенный пиксель сетки, как в КуМир.

Таким образом, на основе проделанной работы, можно заключить что модуль turtle языка Python может полностью заменить алгоритмический язык среды программирования КуМир, предоставляя ровно такой же ряд возможностей и команд для управления учебным исполнителем Черепашка и обладая рядом некоторых существенных преимуществ, таких как: автоматизированный ввод команд, их простота и минимализм, соответствие единицы перемещение черепашки одному пикселю, использование онлайн компиляторов без предварительной установки каких-либо программ на компьютер, перспективы и возможность использования в будущем при изучении языков программирования и возможная ориентация на будущую профессиональную деятельность учеников. А самое главное, возможности реализации алгоритмов построения точечной (пиксельной) графики и формирования навыков их написания у обучающихся.

Глава 2. Реализация изучения темы «Алгоритмизация. Исполнитель Черепашка» инструментами Python

2.1 Тематическое планирование

Тематическое планирование – это один из методов планирования учебного процесса, который основан на выборе центральной темы и выделения подтем, которые будут изучаться в течение определенного периода времени. Этот метод позволяет организовать обучение более эффективно и целенаправленно, а также, в некоторой степени, повысить мотивацию учащихся.

Тематическое планирование может быть использовано в любом предмете и на любом уровне обучения. Оно позволяет учителю более гибко подходить к организации учебного процесса и адаптировать его к потребностям и интересам учащихся. Кроме того, тематическое планирование помогает учителю более эффективно использовать учебное время и повысить качество образования.

Тематическое планирование по информатике является важным инструментом для организации учебного процесса. Оно позволяет структурировать материал, определить цели и задачи, выбрать методы и формы работы, а также, в конечном итоге, оценить результаты обучения.

Основными задачами тематического планирования по информатике являются:

- формирование у учащихся навыков работы с компьютером и программным обеспечением;
- развитие логического мышления и умения решать задачи;
- овладение базовыми знаниями в области информационных технологий;
- повышение культуры безопасности в сети Интернет.

Далее выделим основные этапы тематического и поурочного планирования, которых будем придерживаться при их составлении:

1. Выбор центральной темы и выделение её подтем.
2. Определение цели и задач, которые необходимо достичь в рамках изучения данной темы.
3. Определение методов и форм работы, которые будут использоваться для достижения поставленных целей и задач.
4. Разработка примерного учебного плана, включающего в себя темы уроков, виды работ, необходимые материалы и ресурсы.
5. Оценка результатов обучения и корректировка учебного плана при необходимости (выполняется после апробации разработанной серии уроков).

Центральная тема: Алгоритмизация. Исполнитель Черепашка.

Так как, при анализе учебно-методических комплексов было выявлено, что отводиться разное количество часов, которое, варьируется от 2 до 5, то при разбиении на подтемы, и дальнейшем составлении примерных планов уроков будем, в рамках нашей работы, выделять 4 учебных часа.

Для этого, в начале, выделим *подтемы*, которые станут базой для дальнейших примерных планов уроков:

1. Знакомство со средой программирования. Основы работы с модулем turtle в Python. Возможности сохранения озображения.
2. Использование циклов for и while для автоматизации рисования.
3. Использование подпрограммы (функций) для создания более сложных изображений. Введение понятий параметр и масштаб изображения.
4. Создание пиксельных изображений средствами библиотеки turtle в Python.

Далее для приведённых подтем выделим количество часов, которое стоит отвести на их изучение, учитывая, как аудиторную работу в классе, так и самостоятельную домашнюю работу. Также опираясь на то, что для изучения этой темы было выделено 4 учебных часа (без учёта домашней самостоятельной работы школьников) (табл. 5).

Таблица 5.

Тематическое планирование

№	**Тема**	**Количество часов**		
		Всего	**Аудиторная работа**	**Домашняя работа**
1	Основы работы с модулем turtle в Python.	2	1	1
2	Циклы.	2	1	1
3	Подпрограммы. Понятие параметра и масштаба.	2	1	1
4	Алгоритмы создания пиксельной графики.	2	1	1
	ИТОГ	8		

Отметим, что главной целью разрабатываемых уроков является обучение учеников алгоритмам компьютерной графики, а также программированию. Иными словами, такой раздел как «Алгоритмизация» можно изучать на объектах разной природы, в нашем случае – это алгоритмы построения растрового изображения.

Так, на основе проделанной работы в рамках данного параграфа и анализа результатов первой главы нами было составлено тематическое планирование по теме «Алгоритмизация. Исполнитель Черепашка», общим объёмом 8 часов, из которых 4 часа – классная работа и 4 часа – самостоятельная домашняя работа.

2.2 Примерные планы уроков по изучению темы «Алгоритмизация. Исполнитель Черепашка» инструментами языка программирования Python

На основе приведённого выше тематического планирования и анализа учебно-методических материалов составим примерные планы серии из 4-х уроков по информатике, в рамках темы «Алгоритмизация. Исполнитель Черепашка». Данные планы, для удобства их чтения и использования в дальнейшей профессиональной деятельности представим в формате технологических карт уроков (табл. 6-13). Также, все рабочие файлы, которые ученики будут использовать на уроке, примеры выполнения домашних работ и листинги демонстрационных фалов разместим отдельно, под каждой из технологических карт уроков.

Таблица 6.

Технологиечская карта первого урока

Тема урока	Основы работы с модулем turtle в Python.
Тип (форма) урока	Комбинированный.
Цель урока	Формирование у обучающихся представление о алгоритмах построения пиксельной графики и программировании, при помощи модуля turtle в Python.
Этапы и задачи урока	1 этап (мотивация): Задача этапа: мотивация учеников к дальнейшей практической деятельности. 2 этап (изучение нового материала): 2.1. Задача: познакомить учеников со средой программирования Python; 2.2. Задача: продемонстрировать, как подключается Черепашка, рассмотреть основные типы команд: - показать/спрятать черепашку; - поднять опустить перо; - вперёд; - назад; - поворот влево и вправо; - перемещение в конкретную точку; - сменить цвет пера; - установить цвет фона; 2.3. Задача: продемонстрировать связь построения растрового изображения и темы «Алгоритмизация», на примере написания небольшой программы, которая позволит изобразить рисунок 3 на 3 пикселя. 2.4. Задача: продемонстрировать способ сохранения изображения (средствами Paint). 3 этап (первичное закрепление изученного материала): 3.1. Задача: выдать ученикам задание для самостоятельного решения. 3.2. Задача: выступая в роли тьютора, при необходимости оказывать помощь ученикам. 4 этап (рефлексивный): подвести итоги урока и предоставить домашнее задание;
Образовательные ресурсы	презентация, мультимедийный проектор, проекционный экран, демонстрационные файлы программ, рабочий файл программы, персональные компьютеры, онлайн компилятор.
Формы и методы работы	Формы: фронтальная (Ф), индивидуальная (И).

	Методы: наглядные, словесные, практические.
Основные понятия	учебный исполнитель Черепашка, среда программирования Python, модуль turtle, пиксельное изображение.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ		
Предметные	**Метапредметные**	**Личностные**
Формирование умения составлять и выполнять на компьютере несложные алгоритмы для управления исполнителем Черепашка средствами библиотеки turtle в Python. Формирование представления о связи программирования и создания растровых изображений.	Формирование умения самостоятельно составлять алгоритмы решения задачи.	Формирование уважительного отношения к символам России.

Таблица 7.

Структура первого урока					
Этапы урока	**Время**	**Деятельность учителя**	**Деятельность обучающихся**	**Формы организации взаимодействия**	**Результаты этапа и формы контроля**
I Этап (мотивация)	1-2	Отмечает отсутствующих, сообщает тему и цель сегодняшнего урока. Настраивает учеников на дальнейшую деятельность.	Рассаживаются по местам. Внимательно слушают учителя.	(Ф)	**Результат:** мотивация учеников к дальнейшей учебной деятельности. **Форма контроля:** фронтальная

II Этап (изучение нового материала)	10-15	Демонстрирует окно среды программирования IDLE Python, даёт пояснения; Демонстрирует как подключить Черепашку, а также основные команды с пояснениями: - **st(), ht();** - **pu(), pd();** - **fd();** - **bk();** - **rt(), lt();** - **goto();** - **pencolor();** - **bgcolor();** - **fillcolor();** Для демонстрации связи создания растрового изображения и алгоритмизации в целом, показывает и поясняет алгоритм, который нарисует маленький флаг Российской Федерации, также поясняет, что цвета задаются в HTML формате. Показывает, как найти таблицу с этими кодами в сети Интернет. Демонстрирует как сохранить изображение – создание скриншота окна и размещение его в Paint. Увеличивает сохранённое изображение и наглядно демонстрирует, что рисунок, созданный по алгоритму, получился пиксельным.	Внимательно слушают учителя, при необходимости задают вопросы.	(Ф)	**Результат:** начало формирования представления о возможностях библиотеки turtle, для создания пиксельных изображений; формирование представления об алгоритмах создания пиксельной графики. **Форма контроля:** фронтальная

III Этап (первичное закрепление)	15-20	Выдаёт ученикам рабочий файл, в котором уже записаны строки кода, подключающие Черепашку, настраивающие размеры окна и рассмотренные ранее команды с пояснениями. Выдаёт задание: самостоятельно придумать небольшое изображение, максимум 10 на 10 пикселей и реализовать алгоритм его рисования. Скопировать полученное изображение в Paint и продемонстрировать результат учителю. При необходимости помогает ученикам.	Сдаться за компьютеры и приступают к выполнению задания используя рабочий файл, выданный учителем. При необходимости просят помощи у учителя.	(И) или (Ф)	**Результат:** начало формирования навыков написания алгоритмов по управлению учебным исполнителем Черепашка. Развитие представления об алгоритмах создания пиксельной графики. **Форма контроля:** индивидуальная
IV Рефлексивный	2-3	Кратко повторяет пройденное за урок и совместно с учениками формулирует выводы.	Внимательно слушают учителя и формулирую выводы.	(Ф)	**Результат:** закрепление пройденного материала за урок. **Форма контроля:** фронтальная
Домашнее задание	1-2	Выдаёт домашнее задание: придумайте небольшой логотип, максимум 10 на 10 пикселей и напишите алгоритм для его создания. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.	Записывают домашнее задание.	(Ф)	**Результат:** записанное домашнее задание.

Содержание рабочего файла, для первого урока:

```
from turtle import * # подключение Черепашки
setup (200, 200)     # размеры окна
screensize(150,150)  # размеры холста
bgcolor('#003333')   # задать цвет холсту
pencolor('#FFFFFF')  # задать цвет перу
pd()                 # опустить перо
pu()                 # поднять перо
fd(1)                # вперёд на 1 пиксель
bk(1)                # назад на 1 пиксель
rt(90)               # вправо на  градусов
lt(90)               # влево на 90 градусов
goto(0,0)            # переместиться в (0;0)
ht()                 # скрыть Черепашку
st()                 # показать Черепашку
```

Содержание демонстрационного файла и результат его выполнения (рис. 14):

```
from turtle import *
setup (200, 200)
screensize(150,150)
bgcolor('#003333')
pencolor('#FFFFFF')
goto(0,0)
pd(), fd(5), pu()
pencolor('#0039A6')
```

```
goto(0,-1)
pd(), fd(5), pu()
pencolor('#D52B1E')
goto(0,-2)
pd(), fd(5), pu()
```

Рисунок 14. Пиксельный флаг Российской Федерации

Домашнее задание: придумайте небольшой логотип, максимум 10 на 10 пикселей и напишите алгоритм для его создания. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.

Пример выполнения: – листинг и полученное изображение (рис. 15):

```
from turtle import *
bgcolor('#003333')
pencolor('#FFFFFF')
fd(10), rt(90), fd(10), rt(90), fd(10), rt(90), fd(10), rt(90)
goto(1, -1)
pencolor('#0033FF')
fd(8), rt(90), fd(8), rt(90), fd(8), rt(90), fd(8), rt(90)
goto(2, -2)
pencolor('#66FFFF')
fd(6), rt(90), fd(6), rt(90), fd(6), rt(90), fd(6), rt(90)
goto(3, -3)
pencolor('#0033FF')
fd(4), rt(90), fd(4), rt(90), fd(4), rt(90), fd(4), rt(90)
goto(4, -4)
```

```
fd(2), rt(90), fd(2), rt(90), fd(2), rt(90), fd(2), rt(90)
goto(5, -5)
fd(1)
ht()
```

Рисунок 15. Пример небольшого логотипа

Таблица 8.

Технологиечская карта второго урока

Тема урока	Использование циклов для автоматизации рисования.
Тип (форма) урока	Комбинированный.
Цель урока	Формирование у обучающихся представление об алгоритмах построения пиксельной графики и программировании, через управление исполнителем Черепашка с использованием циклов.
Этапы и задачи урока	1 этап (мотивация): Задача этапа: мотивация учеников к дальнейшей практической деятельности. 2 этап (изучение нового материала): 2.1. Задача: продемонстрировать написание алгоритма, который, не отрывая пера, позволяет нарисовать небольшой элемент славянского узора. 2.2. Задача: организовать обсуждение на тему – если нужно нарисовать это же изображение ещё несколько раз, что можно сделать? Что бы мы делали, если бы использовали какой-либо редактор изображений? 2.3 Задача: выслушав предложения учеников, продемонстрировать, как при помощи конечного (или бесконечного) цикла можно размножить нарисованный элемент и получить небольшой славянский орнамент, поясняет, что циклы, позволяют реализовать функцию копирования. 2.4 Задача: организовать небольшое обсуждение по теме – что будет если использовать разные циклы (for или while) вместо того, что был уже использован, как будет меняться результат выполнения программы. 3 этап (первичное закрепление изученного материала): 3.1. Задача: выдать ученикам задание для самостоятельного решения, которое подразумевает использование циклов для создания изображения. 3.2. Задача: выступая в роли тьютора, при необходимости оказывать помощь ученикам в их самостоятельной деятельности. 4 этап (рефлексивный): подвести итоги урока и предоставить домашнее задание;
Образовательные ресурсы	презентация, мультимедийный проектор, проекционный экран, демонстрационные файлы программ, рабочий файл программы, персональные компьютеры, онлайн компилятор.
Формы и методы работы	Формы: фронтальная (Ф), индивидуальная(И).

	Методы: наглядные, словесные, практические.
Основные понятия	Конструкции циклов for и while, модуль turtle, алгоритм построение растрового изображения.
ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ	

Предметные	**Метапредметные**	**Личностные**
Формирование умения создавать и отлаживать программы для управления исполнителем Черепашка средствами библиотеки turtle в Python, реализующие несложные алгоритмы с использованием циклов. Развитие представления о связи программирования и создания растровых изображений.	формирование умения самостоятельно составлять алгоритмы решения задачи.	Развитие восприимчивости к разным видам искусства, традициям и творчеству своего и других народов.

Таблица 9.

Структура второго урока

Этапы урока	**Время**	**Деятельность учителя**	**Деятельность обучающихся**	**Формы организации взаимодействия**	**Результаты этапа и формы контроля**
I Этап (мотивация)	1-2	Отмечает отсутствующих, сообщает тему и цель урока. Настраивает учеников на дальнейшую деятельность.	Рассаживаются по местам. Внимательно слушают учителя.	(Ф)	**Результат:** мотивация учеников к дальнейшей учебной деятельности. **Форма контроля:** фронтальная
II Этап (изучение нового материала)	10-15	Демонстрирует и поясняет написание алгоритма, который сначала нарисует небольшой элемент узора, не отрывая пера. Далее спрашивает учеников: если мне необходимо нарисовать этот элемент несколько раз, что мне стоит сделать? Поясняет, что в программировании, для того, чтобы размножить какой-либо элемент есть циклы, дополняет код циклом и демонстрирует результат. Организует небольшое обсуждение – как измениться результат выполнения программы и листинг, если использовать не цикл for, а цикл while.	Внимательно слушают учителя, при необходимости задают вопросы. Предполагаемый ответ: - как-нибудь его размножить (скопировать и вставить несколько раз). Участвуют в обсуждении и выдвигают свои идеи.	(Ф)	**Результат:** развитие представления о возможностях библиотеки turtle, для создания пиксельных изображений; формирование представления о циклах, и возможностях их использования. **Форма контроля:** фронтальная

III Этап (первичное закрепление)	15-20	Выдаёт ученикам рабочий файл, с прошлого урока, который дополнен конструкциями циклов. Выдаёт задание: придумаете свой элемент узора, который возможно нарисовать, не отрывая пера и напишите алгоритм, который позволит его изобразить, а затем используйте цикл, чтобы размножить это изображение. При необходимости помогает ученикам.	Сдаться за компьютеры и приступают к выполнению задания используя рабочий файл, выданный учителем. При необходимости просят помощи у учителя.	(И) или (Ф)	**Результат:** развитие навыков написания алгоритмов по управлению учебным исполнителем Черепашка с использованием циклов; развитие представления о связи алгоритмизации и создания растровых изображений. **Форма контроля:** индивидуальная
IV Рефлексивный	2-3	Кратко повторяет пройденное за урок и совместно с учениками формулирует выводы.	Внимательно слушают учителя и формулирую выводы.	(Ф)	**Результат:** закрепление пройденного материала за урок. **Форма контроля:** фронтальная
Домашнее задание	1-2	Выдаёт домашнее задание: выберите любой элемент какого-либо народного орнамента и составьте алгоритм, который позволит его нарисовать. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.	Записывают домашнее задание.	(Ф)	**Результат:** записанное домашнее задание.

Содержание рабочего файла, для второго урока:

```
from turtle import * # подключение Черепашки
setup (200, 200)     # размеры окна
screensize(150,150)  # размеры холста
bgcolor('#003333')   # задать цвет холсту
pencolor('#FFFFFF')  # задать цвет перу
pd()                 # опустить перо
pu()                 # поднять перо
fd(1)                # вперёд на 1 пиксель
bk(1)                # назад на 1 пиксель
rt(90)               # вправо на  градусов
lt(90)               # влево на 90 градусов
goto(0,0)            # переместиться в (0;0)
ht()                 # скрыть Черепашку
st()                 # показать Черепашку

for _ in range(n):   # конечный цикл
    #body

while True:          # бесконечный цикл
    #body
```

Пример демонстрационного файла и результат его выполнения (рис. 16):

```
from turtle import *
bgcolor('#003333')
```

```
pencolor('#FFFFFF')
speed(50)
for _ in range(5):
    rt(45)
    fd(10)
    lt(90)
    fd(10)
    for _ in range(4):
        fd(10), lt(90)
    rt(90)
    fd(10)
    lt(90)
    fd(10)
    rt(45)
ht()
```

Рисунок 16. Результат выполнения программы с циклом

Домашнее задание: выберите любой элемент какого-либо народного орнамента и составьте алгоритм, который позволит его нарисовать и используя конструкции циклов получите узор. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.

Пример выполнения: – листинг и полученное изображение (рис. 17):

```
from turtle import *
bgcolor('#003333')
pencolor('#FFFFFF')
speed(50)
```

```
for _ in range(5):
    lt(45)
    fd(20)
    lt(135)
    fd(10)
    lt(135)
    fd(20)
    lt(90)
    fd(20)
    for _ in range(4):
        fd(10), lt(90)
    rt(90)
    fd(20)
    lt(45)
ht()
```

Рисунок 17. Пример орнамента

Таблица 10.

Технологиечская карта третьего урока

Тема урока	Подпрограммы. Понятия: параметр и масштаб изображения.
Тип (форма) урока	Комбинированный.
Цель урока	Формирование у обучающихся представление о связи алгоритмов построения пиксельной графики с программированием, через управление исполнителем Черепашка с использованием подпрограмм.
Этапы и задачи урока	1 этап (мотивация): Задача этапа: мотивация учеников к дальнейшей практической деятельности. 2 этап (изучение нового материала): 2.1. Задача: ввести понятие масштаба изображения (выделить в качестве параметра значение размера стороны фигуры и продемонстрировать, какие изменения будут происходить при изменении этого параметра). 2.2. Задача: продемонстрировать использование параметра для создания разных геометрических фигур (например, выделить в качестве параметра количество углов фигуры и продемонстрировать алгоритм, который позволяет его задействовать). 2.3. Задача: продемонстрировать конструкцию функции (подпрограммы) в Python и пояснить, какие параметры в неё входят и как её использовать (вызывать); 2.4. Задача: пояснить, как определить блок кода, который целесообразно выделить в качестве функции на примере создания изображения какой-нибудь фигуры – выделить в качестве параметров масштаб и количество углов. 3 этап (первичное закрепление изученного материала): 3.1. Задача: выдать ученикам задание для самостоятельного решения. 3.2. Задача: выступая в роли тьютора, при необходимости оказывать помощь ученикам. 4 этап (рефлексивный): подвести итоги урока и предоставить домашнее задание;
Образовательные ресурсы	презентация, мультимедийный проектор, проекционный экран, демонстрационные файлы программ, рабочий файл программы, персональные компьютеры, онлайн компилятор.
Формы и методы работы	Формы: фронтальная (Ф), индивидуальная(И). Методы: наглядные, словесные, практические.

Основные понятия	подпрограмма, параметр, масштаб изображения, учебный исполнитель Черепашка, модуль turtle.	
ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ		
Предметные	**Метапредметные**	**Личностные**
Формирование умения создавать и отлаживать программы для управления исполнителем Черепашка средствами библиотеки turtle в Python, реализующие несложные алгоритмы с использованием функций. Развитие представления о связи программирования и создания растровых изображений.	Формирование умения определять ситуации, в которых целесообразно выделить последовательность действий в отдельный вызываемый алгоритм.	Развитие восприимчивости к разным видам искусства.

Таблица 11.

Структура первого урока

Этапы урока	Время	Деятельность учителя	Деятельность обучающихся	Формы организации взаимодействия	Результаты этапа и формы контроля
I Этап (мотивация)	1-2	Отмечает отсутствующих, сообщает тему и цель урока. Настраивает учеников на дальнейшую деятельность.	Рассаживаются по местам. Внимательно слушают учителя.	(Ф)	**Результат:** мотивация учеников к дальнейшей учебной деятельности. **Форма контроля:** фронтальная

II Этап (изучение нового материала)	10-15	Демонстрирует и поясняет написание алгоритма, который позволяет изобразить треугольник. Организует обсуждение и задаёт вопросы: – как можно назвать процесс увеличения размеров изображения? – что нам необходимо задать, для масштабирования? – можно ли для масштабирования использовать какой-нибудь коэффициент? Поясняет, что в программирование, такие коэффициенты называются параметрами. Организует обсуждение и задаёт вопрос: – если я хочу рисовать одну и ту же фигуру, но разного размера, рационально ли производить копирование? Поясняет, что существует понятие подпрограммы, которая позволяет реализовать «копирование» изображения – демонстрирует конструкцию функции в Python, поясняет о входящих в неё параметрах.	Внимательно слушают учителя. Предполагаемые ответы на вопросы: – масштабирование; – новые размеры; – да; – не рационально, нужно что-то придумать.	(Ф)	**Результат:** начало формирования представления о таких понятиях как: параметр, масштаб изображения и подпрограмма (функция) в Python. **Форма контроля:** фронтальная

III Этап (первичное закрепление)	15-20	Выдаёт ученикам рабочий файл. Выдаёт задание: придумайте алгоритм, который позволит изобразить элемент узора в различных масштабах, получите таким способом небольшой орнамент. Поэкспериментируйте и используйте цикл. При необходимости помогает ученикам.	Сдаться за компьютеры и приступают к выполнению задания используя рабочий файл, выданный учителем. При необходимости просят помощи у учителя.	(И) или (Ф)	**Результат:** начало формирования навыков написания алгоритмов с использованием функций и параметров; развитие представления о связи алгоритмизации и создания растровых изображений. **Форма контроля:** индивидуальная
IV Рефлексивный	2-3	Кратко повторяет пройденное за урок и совместно с учениками формулирует выводы.	Внимательно слушают учителя и формулирую выводы.	(Ф)	**Результат:** закрепление пройденного материала за урок. **Форма контроля:** фронтальная
Домашнее задание	1-2	Выдаёт домашнее задание: придумайте свой орнамент, или выберите любой, предложенный учителем, составьте программу содержащую функцию для его создания. Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.	Записывают домашнее задание.	-	**Результат:** записанное домашнее задание.

Содержание рабочего файла, для первого урока:

```
from turtle import *    # подключение Черепашки
setup (200, 200)        # размеры окна
screensize(150,150)     # размеры холста
bgcolor('#003333')      # задать цвет холсту
pencolor('#FFFFFF')     # задать цвет перу
pd()                    # опустить перо
pu()                    # поднять перо
fd(1)                   # вперёд на 1 пиксель
bk(1)                   # назад на 1 пиксель
rt(90)                  # вправо на  градусов
lt(90)                  # влево на 90 градусов
goto(0,0)               # переместиться в (0;0)
ht()                    # скрыть Черепашку
st()                    # показать Черепашку

for _ in range():       # конечный цикл
    #body

while True:             # бесконечный цикл
    #body

def name (var):         # функция
    #body
```

```
name(value var)          # вызов функции
```

Пример демонстрационного файла и результат его выполнения (рис.18):

```
from turtle import *

def figure (n, size):
    angle = 360//n
    for _ in range(0, n):
        fd(10*size)
        lt(angle)
ht()

figure(3, 5)
figure(4, 10)
figure(5, 15)
```

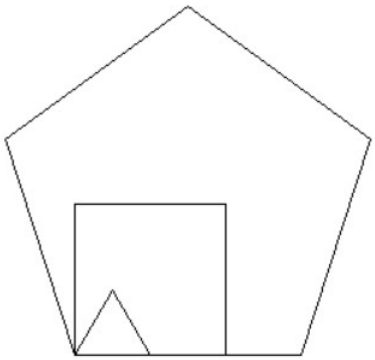


Рисунок 18. Результат выполнения программы с функцией

Домашнее задание: придумайте свой орнамент, или выберите любой, предложенный учителем (рис. 19), составьте программу содержащую функцию для его создания. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.

Пример выполнения: – листинг для первого рисунка (рис. 19.а):

```
from turtle import *
speed(50)
def two_line(size):
    pu(), goto(-5,-5), pd()
    fd(size)
```

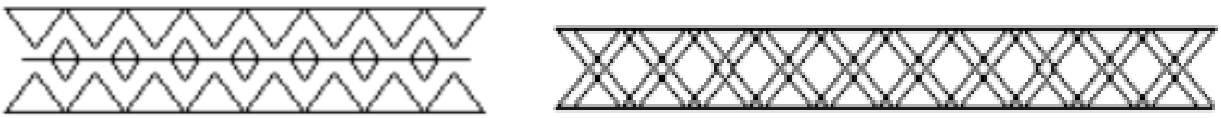
```
    pu(), goto(-5,20), pd()
    fd(size)
    pu(), goto(0,0), pd()

def cross (size):
    fd(20)
    lt(135)
    fd(5)
    lt(135)
    fd(20)
    rt(135)
    fd(5)
    rt(135)

two_line(180)
lt(45)

for _ in range(10):
  cross(20)
```

а. б.



в. г.

Рисунок 19. Пример логотипа.
а. Узор с завитком
б. Узор с квадратами
в. Узор с линиями
г. Узор с треугольниками

Таблица 12.

Технологиечская карта четвёртого урока

Тема урока	Алгоритмы создания пиксельной графики.
Тип (форма) урока	Применение полученных знаний и умений.
Цель урока	Развить у обучающихся представление о связи алгоритмов построения пиксельной графики с программированием, через управление исполнителем Черепашка, при помощи модуля turtle в Python.
Этапы и задачи урока	1 этап (мотивация): Задача этапа: мотивация учеников к дальнейшей практической деятельности. 2 этап (актуализация материала): 2.1. Задача: организовать совместное обсуждение вопроса – как вы считаете, что интересного можно нарисовать, используя пиксельную графику и turtle, а также все ранее изученные алгоритмические конструкции? 2.2. Задача: продемонстрировать не слишком сложный алгоритм создания пиксельной графики, который позволяет рисовать интересный узор, используя ранее изученные конструкции. 2.3. Задача: к написанному алгоритму дать комментарии, отражающие связь создания компьютерной графики и программирования в целом: - использование параметров(переменных) позволяет масштабировать изображение; - использование циклических конструкций позволяет воспроизводить изображения несколько раз (конечно или бесконечно); - использование функций позволяет выделить какую-либо часть изображения, условно, скопировать её и вставлять при необходимости (написать и вызывать функцию); 3 этап (применение изученного материала): 3.1. Задача: выдать ученикам задание для их индивидуального творчества. 3.2. Задача: выступая в роли тьютора, при необходимости помогать ученикам. 4 этап (рефлексивный): подвести итоги урока и предоставить домашнее задание;
Образовательные ресурсы	презентация, мультимедийный проектор, проекционный экран, демонстрационные файлы программ, рабочий файл программы, персональные компьютеры, онлайн компилятор.
Формы и методы работы	Формы: фронтальная (Ф), индивидуальная(И). Методы: наглядные, словесные, практические.

Основные понятия	Алгоритмы пиксельной графики, учебный исполнитель Черепашка, модуль turtle в Python.	
ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ		
Предметные	**Метапредметные**	**Личностные**
Развитие умения создавать и отлаживать программы для управления исполнителем Черепашка средствами библиотеки turtle в Python. Развитие представления о связи программирования и создания растровых изображений.	Развитие умения самостоятельно составлять алгоритмы решения задачи.	Развитие восприимчивости к различным видам искусства.

Таблица 13.

Структура второго урока

Этапы урока	Время	Деятельность учителя	Деятельность обучающихся	Формы организации взаимодействия	Результаты этапа и формы контроля
I Этап (мотивация)	1-2	Отмечает отсутствующих, сообщает тему и цель урока. Настраивает учеников на дальнейшую деятельность.	Рассаживаются по местам. Внимательно слушают учителя.	(Ф)	**Результат:** мотивация учеников к дальнейшей учебной деятельности. **Форма контроля:** фронтальная

II Этап (актуализация знаний)	10-15	Предлагает ученикам совместно обсуждая составить алгоритм, который позволит нарисовать какой-нибудь элемент (это может быть какая-нибудь геометрическая фигура, либо, как в нашем примере - звезда). Предлагает поэкспериментировать и использовать циклы и функции для рисования этого элемента, организует обсуждение – что мы хотим получить на экране компьютера и что мы будем для этого использовать? – если хотим получить тоже изображение в разных масштабах, что стоит использовать? – если хотим рисовать одно и тоже изображение несколько раз, что нам необходимо? После обсуждения и составления алгоритма, даёт комментарии к написанному коду: – у нас с вами, был один фрагмент, который мы использовали несколько раз, для этого мы использовали функцию, которая нам позволила воссоздать копирование изображения; – далее, мы использовали параметр (переменную), которая позволила нам реализовать масштабирование изображения; – и наконец, мы использовали циклы, что также позволило нам воспроизводить элемент изображения несколько раз.	Внимательно слушают учителя. Активно учувствуют в обсуждении – выдвигают идеи. Предположительные ответы на вопросы: – параметр (переменная); – функция (цикл);	(Ф)	**Результат:** развитие представления о связи программирования и алгоритмов создания пиксельной графики. **Форма контроля:** фронтальная

III Этап (применение изученного)	15-20	Выдаёт рабочий файл с прошлого урока. Предлагает ученикам пофантазировать и придумать собственные рисунки-шедевры, для которых они самостоятельно должны написать алгоритм – используя все ранее приобретённые знания. При необходимости помогает ученикам.	Сдаться за компьютеры и приступают к выполнению задания используя рабочий файл, выданный учителем. При необходимости просят помощи у учителя.	(И) или (Ф)	**Результат:** развитие навыков написания алгоритмов по управлению учебным исполнителем Черепашка. **Форма контроля:** индивидуальная
IV Рефлексивный	2-3	Кратко повторяет пройденное за урок и совместно с учениками формулирует выводы.	Внимательно слушают учителя и формулирую выводы.	(Ф)	**Результат:** закрепление пройденного материала за урок. **Форма контроля:** фронтальная
Домашнее задание	1-2	Выдаёт домашнее задание: если не успели за урок – доделать дома сои изображения. Если на компьютере не установлен Python, можно воспользоваться онлайн компилятором: https://trinket.io/turtle.	Записывают домашнее задание.	-	**Результат:** записанное домашнее задание.

Содержание рабочего файла, для второго урока:

```
from turtle import *    # подключение Черепашки
setup (200, 200)        # размеры окна
screensize(150,150)     # размеры холста
bgcolor('#003333')      # задать цвет холсту
pencolor('#FFFFFF')     # задать цвет перу
pd()                    # опустить перо
pu()                    # поднять перо
fd(1)                   # вперёд на 1 пиксель
bk(1)                   # назад на 1 пиксель
rt(90)                  # вправо на  градусов
lt(90)                  # влево на 90 градусов
goto(0,0)               # переместиться в (0;0)
ht()                    # скрыть Черепашку
st()                    # показать Черепашку

for _ in range():       # конечный цикл
    #body

while True:             # бесконечный цикл
    #body

def name (var):         # функция
    #body
```

```
name(value var)          # вызов функции
```

Пример демонстрационного файла и результат его выполнения (рис. 20):

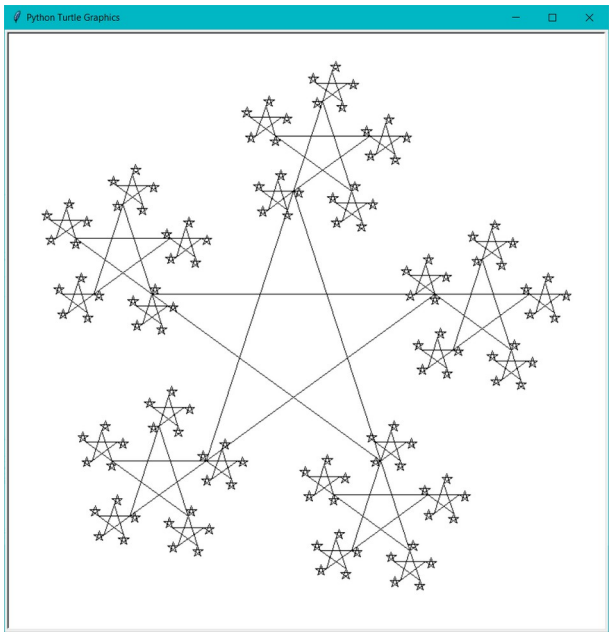


Рисунок 19. Результат выполнения программы

```
from turtle import *
speed(50)
pu()
goto(-200, 50)
pd()

def star(size):
    if size <= 10:
        return
    else:
        for i in range(5):
            fd(size)
            star(size/3)
            lt(216)

star(360)
ht()
```

Так, на основе проделанной работы, были составлены технологические карты четырёх уроков по теме «Алгоритмизация. Исполнитель Черепашка», целью которых является отразить возможность изучения программирования и алгоритмизации на примере создания растровой графики, средствами библиотеки turtle языка программирования Python.

2.3 Методические рекомендации

Основной целью составления данных методических рекомендаций является оказание помощи учителям информатики для успешного внедрения и использованная разработанной серии из четрыёх уроков, посвященных управлению учебным исполнителем Черепашка средствами библиотеки turtle в Python и обучению школьников алгоритмам создания компьютерной графики, а именно – растровой.

Для того, чтобы данные рекомендации выглядели структурированно и понятно, при их составлении будем руководствоваться следующей структуры:

1. Название урока.
2. Особенности изложения содержания темы урока.
3. Подготовка и использование материалов на уроке.

Урок 1: Знакомство со средой программирования. Основы работы с модулем turtle в Python.

Особенности изложения содержания темы урока:

- так как ученики, возможно, ещё не только незнакомы с алгоритмическими конструкциями, но ещё и не знают ни одного языка программирования, включая Python, то стоит выделить время для знакомства с ним и его средой программирования;
- при переходе к знакомству с модулем turtle, стоит заострить внимание на том, что единицы движения Черепашки указываются в пикселях и Черепашка позволяет реализовать пиксельную графику, сделать это можно через написание алгоритма, который просто перекрасит цвет одному-двум пикселям, а затем сделать скриншот окна с результатом выполнения программы и увеличить его размеры в каком-нибудь редакторе;
- стоит прокомментировать, что для создания графики таким образом используются один из алгоритмов создания пиксельной график – придание цветовой характеристики некоторому количеству пикселей в заданном направлении;

- так как это первый урок по теме, не стоит ожидать от учеников чего-то большего, основная задача – понять принцип создания пиксельного изображения с использованием модуля turtle, на примере создания простых, довольно маленьких изображений;
- рекомендуется использовать краткую нотацию команд по управлению Черепашкой – основная задача не натаскать школьников в наборе строк кода, а понять основные принципы создания пиксельной графики.

Подготовка и использование материалов на уроке:

- необходимо заранее продумать и подготовить алгоритм, который будет демонстрироваться как пример для учеников, он должен наглядно демонстрировать суть использования алгоритма по созданию пиксельной графики;
- необходимо заранее подготовить рабочий файл для учеников, содержащий основные команды по управлению Черепашкой., так как нет нужны в том, чтобы натренировать учеников в наборе кода.

Урок 2: Использование циклов для автоматизации рисования.

Особенности изложения содержания темы урока:

- вначале стоит озадачить учеников вопросом: что можно сделать, чтобы размножить один и тот же элемент (изображение) и внимательно выслушать их ответы;
- также стоит провести аналогию с компьютерной графикой, часть изображения, которую мы нарисуем – мотив;
- далее можно демонстрировать конструкции циклов языка программирования Python, давать пояснения для них;
- обязательно нужно провести аналогию с копированием изображения в любом редакторе и использованием цикла в Python;

- также стоит дать обучающимся поэкспериментировать с использованием разных циклов для одной и той же программы, чтобы они почувствовали их разницу;
- снова отметим, что использовать стоит краткую нотацию команд по управлению Черепашкой.

Подготовка и использование материалов на уроке:

- необходимо заранее продумать и подготовить алгоритм, который позволит рисовать небольшой элемент орнамента, не отрывая пера;
- необходимо заранее подготовить рабочий файл для учеников, содержащий основные команды по управлению Черепашкой – добавить в прошлый файл конструкции циклов и комментарии к ним.

Урок 3: Подпрограммы. Понятия параметр и масштаб изображения.

Особенности изложения содержания темы урока:

- не стоит использовать всё понятия сразу, стоит вводить их постепенно, рекомендуемый порядок – масштаб, параметр, функция, где каждое понятие должно плавно вытекать из следующего;
- обязательно нужно проводить аналогию с алгоритмами компьютерной графики:
    - параметр (переменная) позволяет реализовать процесс масштабирования;
    - функция позволяет реализовать процесс копирования элемента изображения и последующей её вставки;

- отметим, что именно на примерах создания изображений целесообразно вводить понятие функции, так как обучающимся будет гораздо проще понять основной смысл и посыл данной конструкции, тем более результат её использования будет подкреплён визуально, что, несомненно, является огромным плюсом;
- также, используем краткую нотацию команд – наша задача не натренировать детей в наборе кода.

Подготовка и использование материалов на уроке:

- необходимо заранее продумать и подготовить алгоритм, в котором чётко просматривается часть, которую можно выделить в функцию;
- необходимо заранее подготовить рабочий файл для учеников, содержащий основные команды по управлению Черепашкой – добавить в прошлый файл конструкцию подпрограммы.

Урок 4: Создание пиксельных изображений.

Особенности изложения содержания темы урока:

- так как это последний урок в серии, то тут можно предложить ученикам пофантазировать и используя ранее приобретённые знания изобразить всё что угодно, но если у обучающихся совсем не возникнет идей, можно использовать и демонстрировать собственные;
- в процессе написания алгоритма, обязательно вспоминаем всё изученное на прошлых уроках и проводим аналогии с алгоритмами компьютерной графики;
- для демонстрации можно подготовить алгоритм, который содержит в себе комбинацию всех ранее изученных команд и позволяет нарисовать что-то действительно красивое и интересное, чтобы вовлечь школьников (можно использовать фрактальную графику);

- используем краткую нотацию команд – наша задача не натренировать детей в наборе кода.

Подготовка и использование материалов на уроке:

- необходимо заранее продумать и подготовить алгоритм, в котором использованы все ранее изученные команды (параметр, цикл, функция);
- в качестве рабочего файла можно использовать файл с прошлого урока.

Также, для помощи учителям, привыкшим к использованию алгоритмического языка среды КуМир, но решившим перейти на модуль turtle языка Python, был составлен словарь для перевода команд Черепашки из КуМир на «язык» Черепашки из Python:

Таблица 14.

Словарь команд Черепашки

Назначение команды	КуМир	Python
Подключение	**использовать Черепаху**	**from turtle import ***
Поднять перо	**поднять хвост**	**up()** *либо* **pu()**
Опустить перо	**опустить хвост**	**down()** *либо* **pd()**
Показать Черепашку	нажатие спец. кнопки	**showturtle()** *либо* **st()**
Скрыть Черепашку	нажатие спец. кнопки	**hideturtle()** *либо* **ht()**
Вперёд	**вперед (цел а)**	**forward(int or float)** **fd(int or float)**
Назад	**назад (цел а)**	**back(int or float)** **bk(int or float)** **backward(int or float)** **fd(-int or -float)**
Вправо	**вправо (вещ угол)**	**right(int or float)** **rt(int or float)**
Влево	**влево (вещ угол)**	**left(int or float)** **lt(int or float)**
Перемещение в точку	-	**setposition(x, y)** **setpos(x, y)** **goto(x, y)** **x, y – int or float**
Цикл for	**нц цел а раз** **кц**	**for i in range(int):**
Цикл while	**нц пока *условие*** **тело цикла** **кц**	**while (*условие*)** #тело цикла
Подпрограмма и её вызов	**алг название_алг_1** **нач** **название_ алг_2** **кон** **алг название_ алг_2** **нач** **кон**	**def name (var)** #тело функции **name (var)** #вызов функции

Таким образом, на основе проделанной работы нами были составлены методические рекомендации и словарь для перевода команд, главная цель которых оказать помощь учителям, при использовании разработанной ранее серии уроков и отразить все тонкости изучения темы «Алгоритмизация. Исполнитель Черепашка» через создание растровых изображений средствами библиотеки turtle языка программирования Python.

Заключение

Итоги решения поставленных задач; оценка соответствия продукта техническому заданию; выводы.

В рамках сформулированной цели, и процессе выполнения работы было выполнено следующие:

1. На основе проведённого анализа различных программных решений для построения растровых изображений средствами программирования в утвержденных учебных программах по информатике был проведёт их сопоставительный анализ, с целью выявления наиболее оптимального из них. На основе этого был обоснован выбор языка программирования Python, поскольку он позволяет реализовать алгоритмы создания пиксельной (растровой) компьютерной графики.
2. В результате сопоставление инструментария среды КуМир и библиотеки turtle языка программирования Python был сделаны вывод, что модуль turtle языка Python может полностью заменить алгоритмический язык среды программирования КуМир, предоставляя ровно такой же ряд возможностей и команд для управления учебным исполнителем Черепашка и обладая рядом некоторых существенных преимуществ.
3. В соответствии с выбранным содержанием и языком программирования произведена разработка серии уроков по теме «Алгоритмизация. Исполнитель Черепашка» инструментами языка программирования Python.
4. Подготовлена техническая и сопроводительная документация по использованию серии уроков для учителей школы.

Таким образом, разработанный продукт соответствует всем требованиям технического задания, все поставленные задачи выполнены; поставленная цель достигнута. Работа носит законченный характер.

Список информационных источников

1. Blockly.Ru – Для будущих программистов. // Blockly.Ru: [сайт]. – URL: http://blockly.ru/ (дата обращения: 15.03.2023).

2. PYPL PopularitY of Programming Language. – Текст : электронный // PYPL index: [сайт]. – URL: https://pypl.github.io/PYPL.html (дата обращения: 15.01.2023).

3. Scratch – Imagine, Program, Share. // Scratch: [сайт]. – URL: https://scratch.mit.edu/projects/editor/?tutorial=getStarted (дата обращения: 18.05.2023).

4. TIOBE index. – Текст : электронный // TIOBE: [сайт]. – URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 15.01.2023).

5. Top Programming Languages 2022. – Текст : электронный // IEEE Spectrum: [сайт]. – URL: https://spectrum.ieee.org/top-programming-languages-2022 (дата обращения: 15.01.2023).

6. turtle – Черепашья графика. – Текст : электронный // turtle – Черепашья графика | Python 3 : [сайт]. – URL: https://digitology.tech/docs/python_3/library/turtle.html (дата обращения: 03.04.2023).

7. Turtle graphics. – Текст : электронный // Python 3.11.4 documentation: [сайт]. – URL: https://docs.python.org/3/library/turtle.html (дата обращения: 15.03.2023).

8. Turtle Programming in Python. – Текст : электронный // GeeksforGeeks : [сайт]. – URL: https://www.geeksforgeeks.org/turtle-programming-python/ (дата обращения: 05.04.2023).

9. Your Python Trinket. – Текст : электронный // Trinket: [сайт]. – URL: https://trinket.io/python (дата обращения: 18.05.2023).

10. Алгоритм. – Текст : электронный // ВикипедиЯ. Свободная энциклопедия : [сайт]. – URL: https://ru.wikipedia.org/wiki/Алгоритм (дата обращения: 15.12.2022).

11.Андреева, Д. Д. Развитие алгоритмического мышления на уроках информатики как ключевой навык обучению программирования школьников / Д. Д. Андреева. – Текст : непосредственный // Личность и общество. – 2020. – № 9. – С. 35-39.
12.Батракова, Л. В. Графический модуль turtle в Python / Л. В. Батракова. – Текст : электронный // Учебник питон. Начало. Черепашка. : [сайт]. – URL: https://digitology.tech/docs/python_3/library/turtle.html (дата обращения: 06.03.2023).
13.Боброва, Н. С. Развитие познавательного интереса к языку python программирования через графические возможности языка / Н. С. Боброва. – Текст : непосредственный // Цифровизация современной школы. – Киров : Кировское региональное отделение Межрегиональной Ассоциации учителей и преподавателей информатики, 2021. – С. 9-12.
14.Босова, Л. Л. Информатика : учебник для 5 класса / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2016. – 184 с. – Текст : непосредственный.
15.Босова, Л. Л. Информатика : учебник для 6 класса / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2016. – 224 с. – Текст : непосредственный.
16.Босова, Л. Л. Информатика : учебник для 7 класса / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2016. – 176 с. – Текст : непосредственный.
17.Босова, Л. Л. Информатика : учебник для 8 класса / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2016. – 176 с. – Текст : непосредственный.
18.Босова, Л. Л. Информатика. Программа для основной школы: 5-6 классы. 7-9 классы / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ. Лаборатория знаний, 2015. – 88 с. – Текст : непосредственный.

19.Босова, Л. Л. Программирование графики и анимации в курсе информатики основной школы / Л. Л. Босова, В. И. Филиппов, А. Ю. Босова. – Текст : непосредственный // Информатика в школе. – 2022. – № 5. – С. 5-15.
20.Бутягина, К. Л. Информатика. Примерные рабочие программы 5-9 классы: учебно-методическое пособие / К. Л. Бутягина. – М. : БИНОМ. Лаборатория знаний, 2018. – 224 с. – Текст : непосредственный.
21.Вельтмандер, П. В. Учебное пособие "Основные алгоритмы компьютерной графики" / П. В. Вельтмандер. – Текст : электронный // : [сайт]. – URL: https://ciu.nstu.ru/kaf/vt (дата обращения: 09.05.2023).
22.Вячина, А. Н. Применение модуля turtle при изучении программирования на языке Python в школьном курсе информатики / А. Н. Вячина. – Текст : непосредственный // Образование. Технологии. Качество. – М. : Издательство "Перо", 2021. – С. 58-63.
23.Гейн, А. Г. Информатика. Методическое пособие к завершённой предметной линии учебников А. Г. Гейна и др. «Информатика 7 класс», «Информатика 8 класс», «Информатика 9 класс» / А. Г. Гейн. – М. : Просвящение, 2020. – 40 с. – Текст : непосредственный.
24.ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы: межгосударственный стандарт: утв. и введ. в действие Постановлением Государственного комитета СССР по стандартам от 24 марта 1989 г. No 661: дата введ. 1990-01-01: переиздание июнь 2009 г. / разработан и внесен Государственным комитетом СССР по стандартам, Министерством приборостроения, средств автоматизации и систем управления СССР. – М.: Стандартинформ, 2009. – Текст : электронный // Электронный фонт правовых и нормативно-технических документов [сайт]. – URL: https://docs.cntd.ru/document/1200006924 (дата обращения 17.01.2023).
25.ГОСТ Р 7.0.100 – 2018. Библиографическая запись. Библиографическое описание: общие требования и правила составления : нац. стандарт Рос.

Федерации : изд. офиц. : утв. и введ. в действие Приказом Федер. агентства по техн. регулированию и метрологии от 3 дек. 2018 г. No 1050-ст : введ. впервые : дата введ. 2019-07-01 / разраб. Федер. гос. унитар. предприятием «Информ. телеграф. агентство России (ИТАР-ТАСС)» фил. «Рос. кн. палата», Федер. гос. бюджет. учреждением «Рос. гос. б-ка», Фе-дер. гос. бюджет. учреждением «Рос. нац. б-ка». – Москва : Стандартинформ, 2018. – 128 с. – (Система стандартов по информации, библиотечному и издательскому делу). – Текст : непосредственный.

26.ГОСТ Р 7.0.100 – 2018: новые правила описания статей и книг. Примеры библиографического описания для списка литературы. Памятка. – Текст: электронный // URL: http://spf-vshni.ru/upload/medialibrary/9df/PAMYATKA-i-PRIMERY-bibliograficheskogo-opisaniya-dlya-spiska-literatury.pdf (дата обращения : 9.06.2023).

27.Графика в Python при помощи модуля Turtle. Часть 1. – Текст : электронный // GeekStand : [сайт]. – URL: https://geekstand.top/development/grafika-v-python-pri-pomoshhi-modulja-turtle/ (дата обращения: 04.03.2023).

28.Графика модуля Turtle. – Текст : электронный //: Основы программирования на языке Python 3: [сайт]. – URL: https://clck.ru/34f28n (дата обращения: 11.01.2023).

29.Громова, С. Ф. Учебные исполнители / С. Ф. Громова, К. А. Богданова. – Текст : непосредственный // Цифровые инструменты в образовании. – Сургут : РИО БУ «Сургутский государственный педагогический университет», 2021. – С. 39-41.

30.Егоров, Е. Знакомство с модулем Turtle | Программирование на Python : [видео] / Е. Егоров // Youtube : [видеохостинг]. – URL: https://www.youtube.com/watch?v=WIWHPDN7CTI (дата обращения: 21.01.2023).

31.Казырбаева, А. А. Графика модуля turtle в python / А. А. Казырбаева, Л. Ф. Розанова. – Текст : непосредственный // Начало в науке. – Уфа : РИЦ БашГУ, 2022. – С. 63-64.
32.Качула, Е. Е. Обучение математике и программированию на языке Python учащихся 5-6 классов с помощью turtle / Е. Е. Качула. – Текст : непосредственный // Информационные технологии в образовании. – 2021. – № 4. – С. 118-122.
33.Качула, Е. Е. Обучение первоначальным навыкам программирования на языке Python учащихся 5-6 классов на уроках информатики с помощью исполнителя turtle / Е. Е. Качула. – Текст : непосредственный // Образование. Технологии. Качество. – М. : Издательство "Перо", 2021. – С. 108-114.
34.Качулина, Е. Е. Использование графической библиотеки turtle языка python для обучения программированию учащихся 5-6 классов на уроках информатики / Е. Е. Качулина. – Текст : непосредственный // Информационные технологии в образовании. – Саратов : ООО Издательский центр "Наука", 2018. – С. 150-156.
35.Компьютерная графика. – Текст : электронный // ВикипедиЯ. Свободная энциклопедия: [сайт]. – URL: https://ru.wikipedia.org/wiki/Компьютерная_графика (дата обращения: 24.12.2022).
36.Кривопляосва, Е. В. Методика обучения основам программирования на языке Python / Е. В. Кривопляосва, В. Ю. Нефёдова, А. В. Прилепина. – Текст : непосредственный // Информатика в школе. – 2020. – № 3. – С. 24-30.
37.Крылова, Т. И. Алгоритм творчества - элективные курсы компьютерной графики в системе изучения информатики и ИКТ (из опыта работы) / Т. И. Крылова, А. М. Кожанова. – Текст : непосредственный // Информатика в школе: прошлое, настоящее и будущее. – Пермь : Пермский государственный национальный исследовательский университет, 2014. – С. 108-114.
38.Лебедева, Т. Н. Понятие «исполнитель» в школьном курсе информатики / Т. Н. Лебедева. – Текст : непосредственный // Методология и методика

формирования научных понятий у учащихся школ и студентов вузов. – Челябинск : Край Ра, 2013. – С. 209-213.

39.Мельничук, А. В. Основные алгоритмы компьютерной графики / А. В. Мельничук. – Текст : непосредственный // Научный взгляд в будущее. – 2020. – № 17. – С. 37-45.

40.Программирование. – Текст : электронный // ВикипедиЯ. Свободная энциклопедия: [сайт]. – URL: https://ru.wikipedia.org/wiki/Программирование (дата обращения: 18.12.2022).

41.Рабинович, В. В. Python для детей: Основные команды черепашьей графики / В. В. Рабинович. – М. : Издательское решение, 2020. – 22 с. – Текст : непосредственный.

42.Сорокина, Т. Е. Использование графической библиотеки turtle graphics языка python для плавного перехода от блочного программирования к текстовому / Т. Е. Сорокина. – Текст : непосредственный // Информатика в школе. – 2018. – № 3. – С. 6-10.

43.Сорокина, Т. Е. Использование графической библиотеки Turtle graphics языка Python для плавного перехода от блочного программирования к текстовому / Т. Е. Сорокина. – Текст : непосредственный // Информатика в школе. – 2018. – № 3. – С. 6-10.

44.Тузов, А. А. Опыт использования модуля Turtle языка Python при изучении темы «Алгоритмизация и программирование» / А. А. Тузов. – Текст : непосредственный // Информатика в школе. – 2015. – № 8. – С. 11-13.

45.ФГОС Основное общее образование. – Текст : электронный // ФГОС : [сайт]. – URL: https://fgos.ru/fgos/fgos-ooo/ (дата обращения: 15.12.2022).

46.Чебурина, О. В. Формирование алгоритмического мышления в обучении программированию игр / О. В. Чебурина. – Текст : непосредственный // Наука и перспективы. – 2017. – № 2. – С. 12-17.

47.Черепаха-Blockly. – Текст : электронный // Учебная среда «Исполнители»: [сайт]. – URL: https://kpolyakov.spb.ru/school/blockly/trt-blockly.htm (дата обращения: 18.02.2023).
48.Черепашья графика при помощи turtle, рисование при помощи алгоритма. Часть 1. – Текст : электронный // Черепашья графика turtle : [сайт]. – URL: https://gvard.github.io/py/turtle/ (дата обращения: 16.12.2022).
49.Черепашья графика при помощи turtle, рисование при помощи алгоритма. Часть 2. – Текст : электронный // Черепашья графика turtle : [сайт]. – URL: https://gvard.github.io/py/turtle/ (дата обращения: 05.02.2023).
50.Черепашья графика. – Текст : электронный // ВикипедиЯ. Свободная энциклопедия: [сайт]. – URL: https://ru.wikipedia.org/wiki/Черепашья_графика (дата обращения: 24.12.2022).