Рекурсивная графика в школьном курсе программирования

Аннотация

В работе рассмотрена методика изучения рекурсивной и фрактальной графики в школьном курсе информатики (раздел «Алгоритмы и программирование») на основе Черепашьей графики. Предложенный подход позволяет знакомить школьников на практике с рекурсией, рекурсивной графикой и фракталами как в старших классах, так и на ранних этапах изучения программирования. Для школьников старших классов используется язык программирования Python, модуль turtle. Для школьников, ещё не знакомых с языком программирования Python, для приложения Исполнитель-Черепаха (автор приложения — К. Ю. Поляков) разработан набор заданий для программирования в блочно-визуальной среде. Показано, что изучение программируемой графики и, в частности, рекурсивной графики, способствует развитию у школьника широкого спектра полезных навыков и умений. Методика успешно применялась на занятиях в STEM-лагере с учениками 5 классов и на уроках информатики в 10-м классе в школе № 550 Центрального района Санкт-Петербурга.

Введение

«Программирование – вторая грамотность», — этот тезис был сформулирован более 40 лет назад академиком А. П. Ершовым, автором первых школьных учебников по информатике, и не утратил свою актуальность в наше время. Современное общество всё чаще использует цифровые технологии и всё больше нуждается в специалистах, имеющих навыки программирования. Но и для тех учеников, которые не станут специалистами в области IT, изучение программирования является важным в образовании и развитии.

Особое место в программировании занимает тема рекурсии. Исследования учёных, в частности А. Р. Есаяна [1], показали, что развитие навыков рекурсивного программирования способствует более интенсивному формированию алгоритмического стиля мышления. Кроме того, программирование и наука в целом — не единственные области, где встречается рекурсия. Прикоснувшись к теме рекурсивных построений на уроках информатики, школьник затем сможет увидеть подобные структуры в природе, литературе, архитектуре, произведениях живописи и графики, услышать в музыкальных произведениях. Это является важным шагом в развитии его как гармоничной личности.

Тема рекурсии включена в федеральную рабочую программу по учебному предмету Информатика 10-11 (темы: рекурсия, рекурсивные алгоритмы (фракталы), рекурсивные процедуры и функции), но одной из проблем обучения методам рекурсивного программирования, как правильно отмечено в работе Е. В. Хламова «Рекурсивная графика в школьной информатике» [2], является состав учебного материала. Список изучаемых рекурсивных алгоритмов состоит, в основном, из вычислительных задач. Такой подход не мотивирует учащихся на изучение рекурсии как метода эффективного программирования.

Расширить базу школьных рекурсивных алгоритмов можно за счёт рекурсивной графики с её эффектными рисунками, которые невозможно получить другими способами. Тот факт, что школьник на основе простых инструкций воспроизводит узор сложной структуры (Рис. 1), может послужить сильной мотивацией к обучению и изучению программирования.

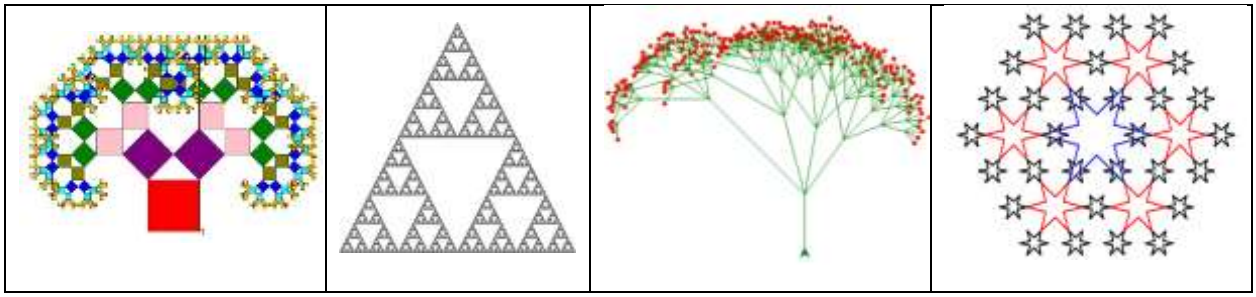


Рис.1 Примеры рекурсивной графики

Важно, что при работе с рекурсивной графикой выполняется множество различных учебных действий:

- анализ, декомпозиция и поиск паттернов;
- описание объекта на языке геометрических терминов;
- определение свойств и отношений отдельных фрагментов (пропорции, масштаб, взаимное положение);
- описание рисунка с помощью детализированного алгоритма, позволяющего на следующем шаге это описание перевести в компьютерную программу;
- разработка и отладка компьютерных программ.

Рекурсия и рекурсивная графика в частности, считаются сложными для понимания и освоения. Рекурсия в школьном курсе рассматривается только в 10-11 классах, причём на базовом уровне — в рекомендательной форме. Вместе с тем, российской школой накоплен большой и интересный опыт изучения рекурсивной графики совсем юными школьниками — 12-14 лет. Для этого использовался язык Лого с его Черепашьей графикой, разработанный в 60-х годах прошлого века и имевший огромную популярность во всём мире, в том числе, в России.[3], [4]

Достоинство Черепашьей графики состоит в том, что ребёнок управляет исполнителем Черепашкой с помощью простых, понятных и «осязаемых» команд, таких как «вперёд», «назад», «повернись», и при этом получает немедленный отклик в виде рисунка (следа Черепашки), который легко сопоставить с желаемым образцом. При этом активизируются различные виды мыслительной деятельности, и их спектр шире, чем при решении вычислительных задач.

В современной школе Лого уже не используется, уступив место другим средам и языкам программирования, но разработчики популярного языка Python включили в Python библиотечный модуль Черепашьей графики ***turtle***, сохранив все атрибуты и функции Лого-черепашки. Это позволяет адаптировать богатый и ценный педагогический опыт, накопленный ранее, к современным условиям и познакомить школьников уже на ранних этапах освоения программирования с такими сложными понятиями как рекурсия и фракталы. Не только познакомить, но и освоить на практике алгоритмы построения рекурсивных рисунков.

Методика использования Черепашьей графики не ограничена модулем ***turtle*** языка программирования Python и может быть успешно применена в других средах, основанных на Черепашьей графике, например, в приложении Черепашка-Blockly (разработчик – К. Ю. Поляков) [5], где используется блочно-визуальное программирование.

Про Черепашью графику говорят, что у неё низкий порог и высокий потолок. Новичку нетрудно этот порог перешагнуть, а профессионал не будет ограничен низким потолком возможностей. Приведённые в работе приёмы и примеры не претендуют на полноту, но могут служить отправной точкой для дальнейшего более глубокого изучения рекурсии и рекурсивной графики.

Практическая часть

Подготовительные упражнения

Конструированию алгоритмов рекурсивных рисунков предшествуют занятия, на которых школьники приобретают и закрепляют необходимые навыки работы с программируемой графикой. Кроме базовых навыков программирования (управление Черепашкой, программирование условных операторов и циклов, описание и вызов функций), осваиваются и специфические, связанные непосредственно с графикой и — в перспективе — с рекурсивной графикой. Компьютерная графика органично вписывается в «стандартный» курс школьного программирования наряду с решением вычислительных задач и задач обработки строковых данных и списков. Более того, компьютерная графика «включает» новые способы мышления и, как уже было сказано, расширяет спектр навыков, причём важное значение здесь имеет визуализация, наглядность процессов. Для иллюстрации этого приведём кратко ключевые моменты занятия по теме «Функции», на котором ученики на практике знакомятся с понятиями «подпрограмма», «функция», программируя рисунки из повторяющихся элементов (подготовка к рекурсии).

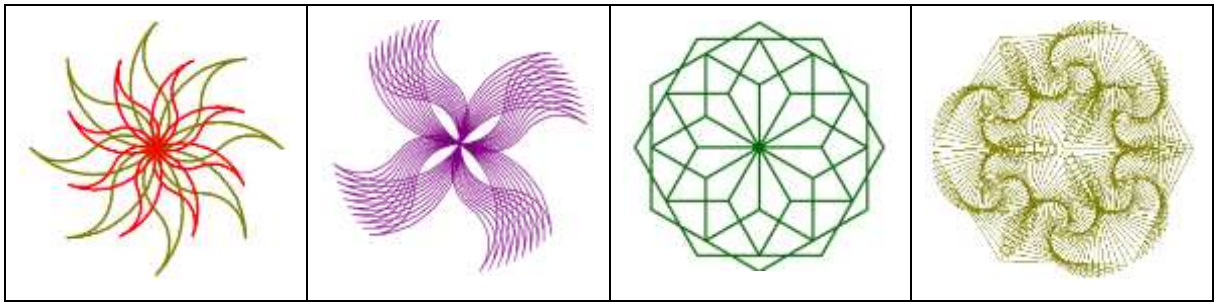


Рис. 2 «Поиск паттернов»

Работа разделена на несколько последовательных этапов, на каждом из которых выполняются определённые учебные действия.

1. Анализ, декомпозиция, поиск паттернов

Задача ученика на этом этапе: научиться анализировать сложную геометрическую конструкцию (Рис. 2), находить и выделять

повторяющиеся элементы. Рисунок – наглядная иллюстрация общего подхода к решению задач. В ходе обсуждения ученики должны прийти к пониманию того, что для программирования сложного рисунка (в общем случае – для решения большой сложной задачи) следует разделить его на более простые части, что облегчит и сам процесс программирования, и отладку программы.

2. Построение геометрической модели

Задача: построить модель объекта-паттерна, описав его с помощью геометрических фигур.

3. Разработка алгоритма построения паттерна для Черепашки

Задача: описать алгоритм построения паттерна, ориентированный на исполнителя (Черепашку), по которому на следующем этапе разрабатывается программный код. На каждом шаге алгоритма выполняются важные мыслительные операции: определяются начальные условия (положение и направление исполнителя на плоскости), требуемые действия и новые условия — после выполнения этого действия. На этом этапе алгоритм описывает фрагмент рисунка, он является вспомогательным, а будущий программный код, ему соответствующий — подпрограммой

4. Синтез частичного решения: разработка подпрограммы

Задача: разработать программный код, соответствующий алгоритму, составленному на предыдущем шаге. Так как на этом занятии подпрограмма – новое понятие, то сопутствующей задачей является освоение синтаксиса и правила оформления и вызова функции.

5. Синтез решения исходной задачи

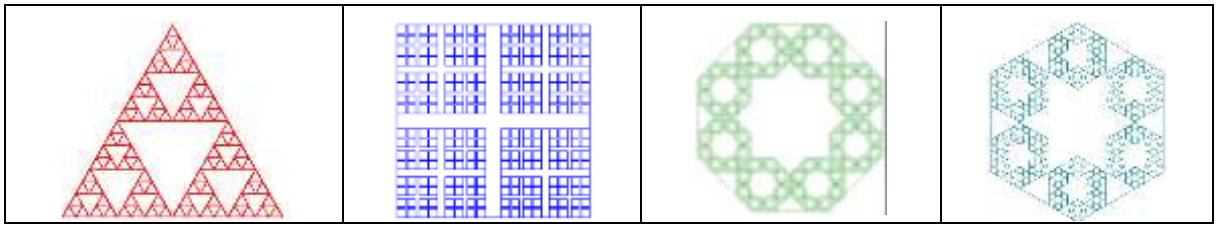
Задача: разработать алгоритм и программу решения исходной задачи с использованием вспомогательных подпрограмм, причём подпрограмм может быть несколько и, если говорить о рисунке, то его «строительные блоки» могут укрупняться постепенно. Итогом должна быть хорошо структурированная программа с понятным «прозрачным» кодом.

Построение рекурсивных графических алгоритмов. Занятие 1

На первом занятии ученики знакомятся с природными рекурсивными объектами, примерами из литературы, архитектуры, других видов искусства, и на простых примерах – вычисление факториала, последовательности Фибоначчи, рекурсивно описанной спирали исследуют, как работают рекурсивные вызовы функции (презентация, слайды 2-7)

Построение рекурсивных графических алгоритмов. Занятие 2

Рассмотрим следующий шаг – построение рекурсивных объектов сложной структуры.



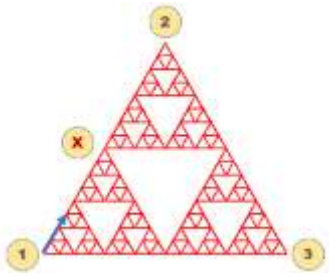
Работа над построением рекурсивных алгоритмов имеет те же этапы, что и приведённые выше: поиск паттерна, построение геометрической модели, разработка вспомогательных алгоритмов, синтез общего решения, но с некоторыми отличиями. Главное отличие: геометрическая модель – рекурсивная, её надо формулировать рекурсивно.

Учитель в беседе с учениками проходит все этапы построения одного из рекурсивных рисунков, направляя беседу и акцентируя внимание на ключевых моментах. Другие рисунки ученики выполняют самостоятельно по аналогии.

Рассмотрим первый рисунок, который носит название «треугольник Серпинского». Ученики могут описать треугольник Серпинского разными способами, и для разных словесных описаний алгоритмы рисования будут различными. Остановимся на такой модели: треугольник Серпинского — это правильный треугольник, в каждой вершине которого нарисован ***такой***

же треугольник, но в 2 раза меньше, в каждой вершине которого нарисован ***такой же треугольник***, но в 2 раза меньше, и так далее, и так далее.

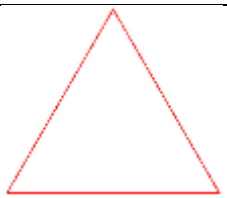
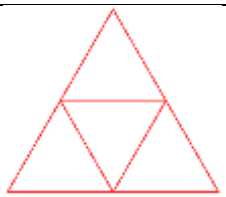
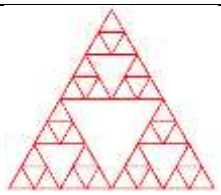
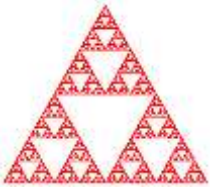
При построении алгоритма по этому описанию надо понимать, что слова «такой же треугольник» означают: если за начало рисования мы приняли, например, левую нижнюю вершину (по отношению к Черепашке) большого треугольника, а направление его обхода — по часовой стрелке, то и начинать рисовать «такой же треугольник» нужно с его левой нижней вершины и обходить его также по часовой стрелке. Таким образом, алгоритм, соответствующий модели, описывается так:



1. Нарисовать в вершине 1 «такой же треугольник», со стороной в 2 раза меньше.
2. Переместить Черепашку в соседнюю вершину 2, повернуть направо на 120 градусов. Перемещать надо именно в вершину 2, не в точку X, иначе нарушается правило «такой же».
3. В этой вершине нарисовать «такой же треугольник» со стороной в 2 раза меньше.
4. Переместить Черепашку в соседнюю вершину 3, повернуть направо на 120 градусов.
5. В новой вершине нарисовать «такой же треугольник» со стороной в 2 раза меньше.
6. Вернуть Черепашку в вершину 1, повернуть направо на 120 градусов. (*Черепашка должна заканчивать рисунок в исходной позиции).*

Выполняя заданный таким образом алгоритм, Черепашка никогда не закончит работу, надо позаботиться об ***условии остановки***. Можно остановить процесс, если треугольники становятся слишком маленькие, и наш глаз уже не может различить изменения. Программисты, как правило, используют другой способ остановки процесса рекурсивных вызовов — задают *глубину рекурсии*. Процесс останавливается, когда достигнут

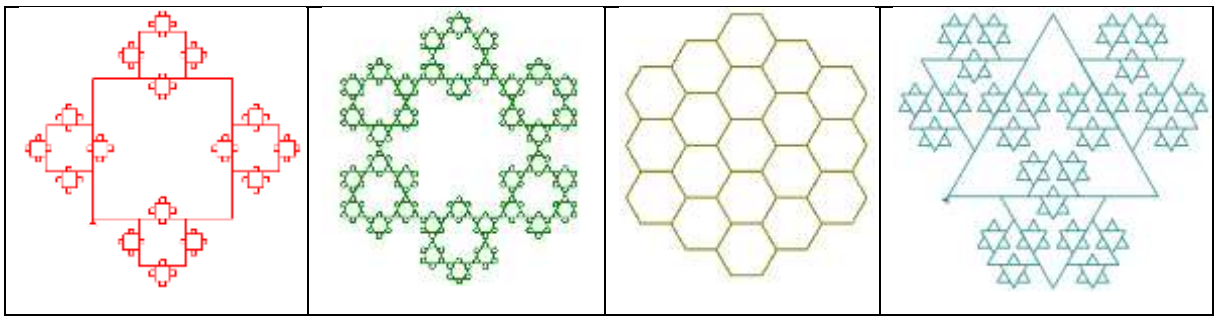
заданный уровень глубины рекурсии. В нашем случае заданная глубина рекурсии – это количество треугольников внутри каждой из вершин «большого» треугольника. На рисунке буквой n обозначена глубина рекурсии.

			
n = 1	n = 2	n = 3	n = 6

Трудно было представить вначале, что сложному рисунку соответствует простой и лаконичный код, но это так (здесь и дальше для экономии места несколько команд пишутся на одной строке):

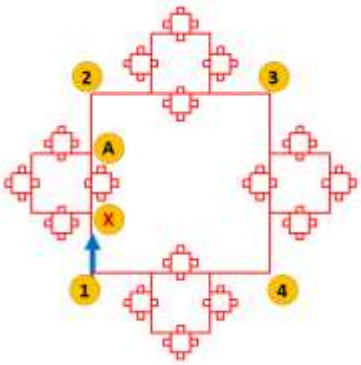
```
def treug_serp (a, n): # параметр n отвечает за глубину рекурсии
  if n > 0:    #  в противном случае завершаем работу
    for _ in range (3):
      treug_serp (a / 2, n - 1)    # рекурсивный вызов
      forward (a); right (120)   # переход к очередной вершине
left (60): speed (0)       # начальные установки для Черепашки
treug_serp (300, 4)    # вызов функции, глубина рекурсии n = 4
```

Построение рекурсивных графических алгоритмов. Занятие 3



Задачи этого занятия похожи на предыдущие, но побуждают учеников глубже исследовать правила рекурсивного вызова функции. Рисунки имеют общее отличие — построение «таких же», то есть подобных, объектов выполняется не в вершине «объекта-родителя», а на его стороне. Это

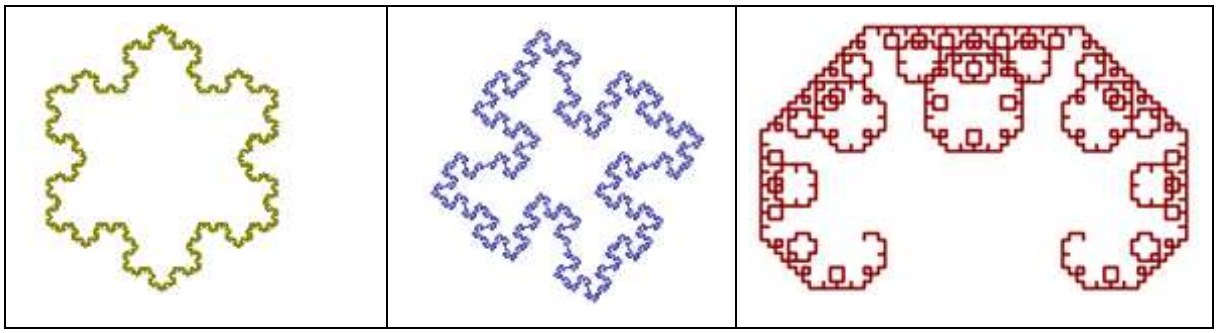
отличие несколько усложняет поиск точки, которая должна стать точкой рекурсивного вызова.



На примере одного из рисунков, например, конструкции из квадратов, учитель вместе с учениками разбирает возможное решение. Другие рисунки выполняются по аналогии. Ключевым моментом является понимание того, что, если обход квадрата начинается из вершины 1 и выполняется по часовой стрелке, то точкой первого рекурсивного вызова является вершина А (не Х!), и при этом Черепашку в точке А надо повернуть на 180 градусов. Таким образом, код рекурсивной функции записывается так:

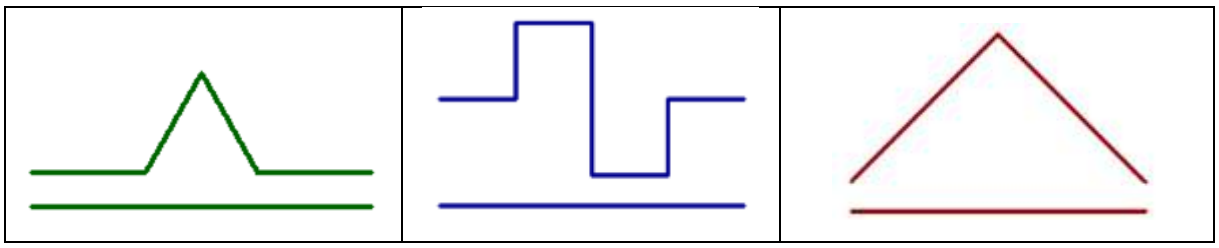
```
def rec_square (a, n): # параметр n отвечает за глубину рекурсии
    if n > 0:   # в противном случае завершаем работу
        for _ in range (4):   # Черепашка обходит квадрат
            forward (a / 3 * 2) # подводим Черепашку к нужной вершине
            right (180)  # и поворачиваем в нужном направлении
            rec_square( a / 3, n-1)     # рекурсивный вызов
            right (180) # направляем Черепашку к следующей вершине
            forward (a / 3);   right (90)      # и перемещаем к этой вершине
```

Построение рекурсивных графических алгоритмов. Занятие 4



Это – известные рисунки, они даже имеют имена, данные им в честь учёных, впервые их описавших: снежинка Коха, кривая Минковского, кривая Леви. Рисунки существенно отличаются от предыдущих. Если

раньше можно было без труда выделить базовый элемент и правило его изменения при переходе на новый уровень рекурсии, то здесь это сделать крайне затруднительно, если вообще возможно. Для конструирования этих и подобных рисунков используется другой подход: базовый элемент (он называется инициатором) и правило перехода на новый уровень рекурсии, который описывается графически и называется генератором, даны. По этим исходным данным строится рекурсивный рисунок. Для рисунков, которые представлены в начале раздела, инициатором является отрезок прямой, а генераторы — ломаные простой структуры — приведены в таблице:



Перед разработкой программного кода учитель вместе с учениками анализирует, как работает пара инициатор – генератор на примере кривой Коха. Для этой кривой генератор состоит из 4-х звеньев, длина каждого звена в три раза меньше родительского отрезка. Звенья соединены между собой под углами 120, 60, 120 градусов. На рисунках n – уровень рекурсии.

n = 0	n = 1	n = 2	n = 3

Всё начинается с отрезка прямой (инициатора). При переходе на следующий уровень рекурсии отрезок превращается в ломаную линию по правилу, который описан генератором. Затем каждый из полученных отрезков превращается по ***тому же правилу*** в ломаную линию, затем каждый из новых отрезков превращается по ***тому же правилу*** в ломаную, и так далее, и так далее… Для того, чтобы составить алгоритм, а затем и программу, эту модель надо детализировать, выполнив визуально-алгоритмическое исследование: определить, в каких точках выполняется

рекурсивный вызов, в каком положении находится Черепашка в этот момент, в каком положении она находится после «рекурсивного путешествия» и т. п. На языке Python алгоритм записывается так:

```
from turtle import *    # подключение библиотеки
def koch (a, n):    # рекурсивная функция для кривой Коха
    if n == 0:  forward(a); return   # останавливаемся, нарисовав инициатор
    koch (a / 3, n - 1)    # обход ломаной с заменой её звеньев
    left (60);   koch (a / 3, n - 1)     # рекурсивными вызовами
    right (120);   koch (a / 3, n - 1)    # третий рекурсивный вызов
    left (60);   koch (a / 3, n - 1)    # четвёртый рекурсивный вызов
koch (200, 4) # вызов функции с глубиной рекурсии n = 4
```

Снежинка получится, если соединить кривые Коха треугольником:

```
for _ in range (3):
    koch (200, 4);  right (120)
```

Другие задачи этого занятия ученики решают самостоятельно по аналогии.

Заключение

В этой работе рассмотрено лишь небольшое количество примеров рекурсивных рисунков. Дополнить учебную коллекцию можно образцами, которых в сети множество, например, фрактальные деревья (есть даже «обдуваемое ветром дерево Пифагора»). Некоторое количество авторских рисунков собрано в «Галерее рекурсивной графики» [3].

Рекурсия завораживает, рекурсия учит, рекурсия вдохновляет. Освоив рассмотренные приёмы и техники, ученик получает хорошую базу и инструменты для дальнейших самостоятельных исследований и творчества.

Список литературы

1. Есаян А. Р. Часть автореферата диссертации на тему «Теория и методика обучения алгоритмизации на основе рекурсии в курсе информатики педагогического вуза» // URL: https://www.dissercat.com/content/teoriya-i-metodika-obucheniya-algoritmizatsii-na-osnove-rekursii-v-kurse-informatiki-pedagog (Дата обращения 25.11.2023)
2. Хламов Е. В. Рекурсивная графика в школьном курсе информатики // URL: https://cyberleninka.ru/article/n/rekursivnaya-grafika-v-shkolnoy-informatike/viewer (Дата обращения 25.11.2023)
3. Тузова О. А. Программируем на Лого // URL: https://portal.ort.spb.ru/lib/Documents/LogoTextbook/index.html (Дата обращения 25.11.2023)
4. Кузнецова И. Н. Моделирование в среде Лого. Занятие 5. Создание микромиров // URL: https://cyberleninka.ru/article/n/modelirovanie-v-srede-logo-zanyatie-5-sozdanie-mikromirov/viewer (Дата обращения 25.11.2023)
5. Поляков К. Ю. Черепаха-Blockly // URL: https://kpolyakov.spb.ru/school/blockly/trt-blockly.htm (Дата обращения 25.11.2023)