

DOI: 10.32517/2221-1993-2024-23-2-44-52

И. С. Орещенков

Интернет-портал «Банки РБ» (banks.of.by), г. Молодечно, Минская область, Беларусь

ИЗУЧЕНИЕ ОСНОВ ТЕМЫ «РЕКУРСИЯ» В СРЕДЕ ГРАФИЧЕСКОГО ИСПОЛНИТЕЛЯ «БУКАШКА»*

Аннотация

Одной из наиболее сложных тем школьного курса алгоритмизации и программирования является рекурсия. Использование среды графического исполнителя (ГРИС) позволяет сформировать у школьников ментальную алгоритмическую схему принципов построения и работы рекурсивных программ, которая упростит восприятие более сложных аспектов этой темы на дальнейших этапах обучения.

Не выходя за рамки ограниченных возможностей среды ГРИС, на простых наглядных примерах объясняются следующие понятия: рекурсивный вызов процедуры, отложенный возврат из процедуры, условие продолжения рекурсии, прямой и обратный шаги рекурсии. Показывается, как с помощью рекурсии могут быть реализованы цикл ПОКА и цикл ДО.

На примере нестандартной задачи рисования квадрата демонстрируется, как введение рекурсии приводит к сокращению размера программы и времени ее выполнения. Приводятся примеры задач для графического исполнителя, решения которых могут быть построены только с использованием рекурсии. Показывается, как рекурсия расширяет возможности позиционирования графического исполнителя.

Для проведения учебных занятий предлагается использовать независимую от вычислительной платформы реализацию ГРИС «Букашка», которая полностью совместима с ГРИС «Кенгурёнок РУ» и может быть интегрирована в информационную среду образовательной организации.

Ключевые слова: информатика, обучение алгоритмизации и программированию, графический исполнитель, нестандартные задачи, рекурсия.

1. Введение

На современном этапе развития общества возник разрыв между потребностью экономики в ИТ-специалистах и возможностями их подготовки школами, техникумами и вузами. В трудах отечественных и зарубежных исследователей высказываются мнения, что одним из элементов решения этой проблемы может

стать, возможно, раннее начало, обязательность и непрерывность обучения школьников алгоритмизации и программированию [2, 8]. В то же время отмечается, что применение на уроках промышленных языков программирования не является целесообразным, потому что у детей еще слабые навыки абстрактного мышления. Рекомендуется использовать адаптированные языки и среды программирования, специально разработанные

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_02-2024/

Контактная информация

Орещенков Игорь Сергеевич, администратор веб-сервера, интернет-портал «Банки РБ» (banks.of.by), г. Молодечно, Минская область, Беларусь; e-mail: iharsw@yandex.by

I. S. Areshchankau

"Banks of the Belarus" web site (banks.of.by), Molodechno, Minsk Region, Belarus

LEARNING THE BASICS OF THE THEME "RECURSION" USING THE "BUKASHKA" ("LADYBUG") DRAWING ACTOR SOFTWARE IMPLEMENTATION

Abstract

Recursion is one of the most difficult themes of the school course of informatics. The use of a drawing actor software allows schoolchildren to form a mental algorithmic scheme of the principles how recursive programs are designed and work. This will simplify the perception of more complex aspects of this theme at further stages of education.

The following concepts are explained using simple illustrative examples without going beyond the limited capabilities of the learning environment: recursive call of a procedure, deferred return from the procedure, head- middle- and tail-recursion. It is shown how a WHILE and a DO-WHILE loop can be implemented using recursion.

It is demonstrated how the use of recursion leads to a reduction in program size and its execution time by the example of the non-standard problem of drawing a square. Examples of problems for a graphic actor are given, solutions of which can only be designed using recursion. It shows how recursion increases the possibilities of positioning the drawing actor.

It is proposed to use the drawing actor software "Bukashka" ("Ladybug") for training sessions. It is computer platform agnostic, the implementation "Kangaroo Roo" compatible and can be integrated into the information environment of an educational organization.

Keywords: informatics, training in algorithmization and programming, drawing actor software, non-standard problems, recursion.

Таблица

Средний процент выполнения заданий ЕГЭ на рекурсию/рекуррентные соотношения в период с 2014 по 2023 год

Тема	Год									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Рекурсия	13,2	25,7	36,3	54,6	43,0	57,5	50,6	—	—	—
Рекуррентные соотношения	—	—	—	—	—	—	—	59,2	67,0	59,4

для обучения младших школьников с учетом их психофизиологического и интеллектуального развития, — интегрированные системы учебных исполнителей [1].

Для выработки у младших школьников начальных навыков алгоритмического мышления и предоставления учащимся возможности выражения их творческого потенциала на уроках информатики используются разнообразные графические исполнители, система команд которых позволяет создавать геометрические фигуры и рисунки [6]. Отдельный интерес представляют простейшие графические исполнители, которые не способны выполнять операции над числовыми величинами и непосредственно взаимодействуют со своей виртуальной средой. Исполнители этого класса получили наименование «графические исполнители», ГРИС [18, с. 22]. Эффективность их использования в образовательном процессе проверена временем: вошедшие в отечественные программно-методические комплекты на этапе перехода к «машинному варианту» преподавания информатики во второй половине 1980-х годов [16, с. 37], они уверенно сохраняют свои позиции в современных учебниках [см. 14] и образовательных программах [3].

В младшей школе применение ГРИС позволяет учесть психологические особенности учеников и внедрить в уроки игровые методики, например, графический диктант [15] в режиме непосредственного выполнения команд. Но у ребят постарше отношение к этому учебному инструменту становится неоднозначным. Благодаря возросшей доступности вычислительной техники и учебных материалов некоторые школьники приходят на уроки информатики, уже имея начальный опыт программирования на промышленных языках или разработки веб-сайтов. Они быстрее сверстников воспринимают учебный материал, предусмотренный программой, и теряют интерес к «игрушечной» среде исполнителя.

Чтобы восстановить у подготовленных учеников интерес к занятиям, а заодно и углубить их понимание предмета, надо продемонстрировать им эстетику простоты учебного исполнителя. Графический исполнитель действительно прост. Но пустяковая с виду безделушка, умеющая выполнять какие-нибудь хитрые функции, приводит в восторг. Снисходительность сменяется уважением, когда из предмета извлекается гораздо больше пользы, чем ожидалось [10].

Одним из лучших способов демонстрации возможностей ГРИС является решение нестандартных задач. Процесс решения приводит к творческой работе мысли учащихся, способствует их развитию и представляется сильнейшим средством активизации познавательной деятельности [17]. Нетипичные задания и упражнения

применимы не только для уроков, но и для внеурочной деятельности и олимпиад. Такие задания могут использоваться и как индивидуальные для наиболее успешных учеников, и как дополнительные задания для желающих.

Одной из наиболее сложных тем раздела «Алгоритмизация и программирование» школьного курса информатики является рекурсия. Федеральный институт педагогических измерений в «Методических рекомендациях для учителей, подготовленных на основе анализа типичных ошибок участников ЕГЭ» (<https://fipi.ru/ege/analiticheskie-i-metodicheskie-materialy>) за разные годы сообщает, что при выполнении заданий ЕГЭ, контролирующих умение исполнять рекурсивный алгоритм, у многих участников возникали затруднения вплоть до 2021 года (см. табл.). С переходом в 2021 году на модель К-ЕГЭ была проведена замена «бланкового» задания повышенного уровня сложности, проверяющего умение исполнять рекурсивный алгоритм, на задание, проверяющее умение вычислить значение по заданным рекуррентным соотношениям. При этом средний процент выполнения заданий этой линии вырос с 51 % (в 2020 году) до 59 % (в 2021 году).

Несмотря на заметное улучшение ситуации с 2017 года, обусловленное повышенным вниманием к преподаванию рекурсии, внутренняя сложность этой темы для учащихся никуда не исчезла.

В этой статье описывается возможный подход к изучению базовых понятий рекурсии при решении нестандартных задач в среде ГРИС. Он сочетает в себе использование описанных выше достоинств нетипичных задач с выработкой у школьников ментальной алгоритмической схемы рекурсивных алгоритмов, которая станет основой для объяснения более сложных аспектов в дальнейших курсах [7].

2. О реализации ГРИС «Букашка»

Среди множества реализаций ГРИС выделяется программа «Кенгурёнок Ру» (рис. 1), разработанная авторским коллективом, в который входили А. Г. Гейн, В. Г. Житомирский, Е. В. Линецкий, М. В. Сапир, В. Ф. Шолохович, и поставлявшаяся в составе программно-методического комплекта «Roo & Robby» для IBM PS/2, выпущенного в 1991 году «Компьютерным учебно-издательским центром» КУДИЦ, Москва. Предшественником «Кенгурёнка» была программа «Чертёжник» для КУВТ «Ямаха», которая сопровождала подготовленный теми же авторами учебник [4].

Забавный анимированный персонаж, попавший на экран компьютера из диснеевского мультсериала,

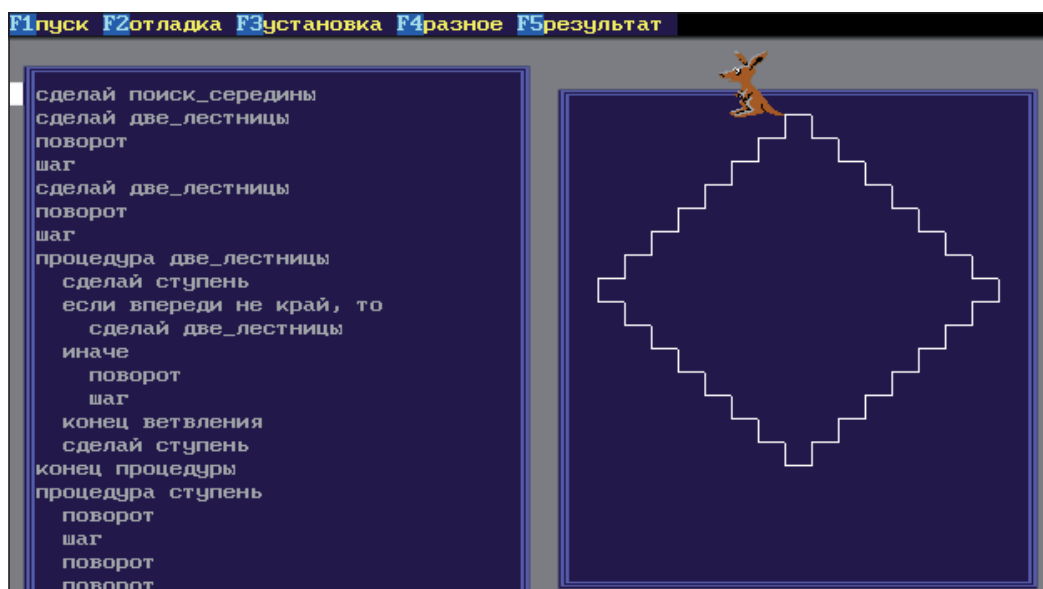


Рис. 1. Реализация ГРИС «Кенгурёнок Py» для операционной системы MS DOS долгое время продолжала эксплуатироваться в среде эмулятора DosBox под Windows

транслировавшегося в те годы по центральному телевидению, сразу завоевал симпатии как школьников, так и их учителей. Несмотря на повсеместный переход на операционные системы семейства Windows в первом десятилетии 2000-х, исполнитель «Кенгурёнок» продолжал использоваться в образовательном процессе, чему свидетельством являются инструкции о запуске программы goo.exe в эмуляторе DosBox.

Современное состояние вычислительной техники характеризуется большим разнообразием программно-аппаратных платформ. В конце первого десятилетия 2000-х годов получили широкое распространение новые типы персональных информационно-вычислительных устройств — смартфоны и планшеты, — которые ранее существовали только в виде концептуальных образцов. Набирает популярность архитектура ARM, отнимая долю рынка у традиционных игроков — Intel и AMD, — благодаря более низкой стоимости и высокой энергоэффективности при достаточной производительности в традиционных сферах использования. Снижается степень доминирования операционных систем производства Microsoft, уступая место различным реализациям UNIX, дистрибутивам Linux и версиям Android.

Чтобы избежать сложностей, сопряженных с использованием программы «Кенгурёнок» на современных вычислительных устройствах, автором статьи была разработана реализация простейшего графического исполнителя «Букашка», которая работает в большинстве веб-браузеров [11]. Помимо полной программной совместимости с прототипом преследовалась цель сохранить присущие ГРИС игровые элементы. В актуальной на данный момент версии 1.21 наряду с анимацией персонажа реализованы:

- звуковое сопровождение выполняемых команд;
- цветовая сигнализация результата проверки условий в инструкциях ветвления и циклов;
- отображение времени, затраченного на выполнение программы (для планирования уроков и сравнения различных вариантов решения задач).

Благодаря использованию стандартизированных веб-технологий в качестве основы для реализации ГРИС «Букашка», отсутствию привязки к веб-серверу и распространению по лицензии GPL v.3 этот продукт может использоваться локально или быть легко интегрирован в информационную среду образовательной организации [5, с. 177—183], например, может быть размещен на школьном интернет-сайте. При необходимости как интерфейс ГРИС «Букашка», так и его система команд без значительных трудностей могут быть переведены на национальный язык. Открытые исходные тексты позволяют самим школьникам превратить ГРИС «Букашка», например, в ГРИС «Лягушка», ГРИС «Черепашка» или ГРИС «Автомобиль» — для этого достаточно перерисовать 44 кадра с изображениями различных фаз исполнителя в файле img/sprite.gif. Такое задание может быть разделено между участниками для выработки навыков командной работы.

Тексты программ из этой статьи оформлены в виде HTML-файлов. Если при просмотре в веб-браузере нажать кнопку «Выполнить», то произойдет загрузка текста в среду ГРИС «Букашка» на веб-сервере разработчика (<https://iharsw.lgn.by/gris>). Для выполнения программы в обычном режиме остается выбрать пункт меню «F1пуск», а в режиме быстрого получения результата — «F5результат». Чтобы реализовать такую же интерактивность на школьном сервере, он должен быть настроен на выполнение РНР-сценариев. Кроме того, потребуется указать IP-адрес или имя сервера в HTML-файлах с текстами программ, которые сопровождают эту статью и которые можно загрузить по ссылке: https://iharsw.lgn.by/gris/files/gris_recursion.zip.

3. Изучение рекурсии

3.1. Подход к теме рекурсии

Подойти к теме рекурсии с использованием ГРИС можно по разным траекториям. Самая быстрая: *следование → подпрограммы → бесконечная рекурсия →*

ветвление и условие завершения рекурсии. Циклы при этом даются на заключительном этапе курса. Такой подход продемонстрирован в авторском видеокурсе по ГРИС «Букашка» (<https://youtube.com/playlist?list=PLp7gErb2baQBvPQhGiHsNEcXvsLaN86Tv>). Другая, более традиционная последовательность изучения темы: *следование → подпрограммы → циклы → ветвление → рекурсия*. Такая траектория используется в учебнике И. Г. Семакина и др. [18]. Оба варианта логически непротиворечивы и ведут к полному освоению материала по основам алгоритмизации.

В этой статье выбран традиционный путь с проведением аналогии между циклическим и рекурсивным подходами [13] для простейшей программы рисования лестницы. Преимущества рекурсии показываются на наглядном примере улучшения решения задачи рисования квадрата, которая сама по себе является нетривиальной [12, задача 5].

3.2. Решение задачи рисования квадрата

Условие задачи.

Пусть исполнитель нарисует на своем рабочем поле самый большой квадрат, который там может поместиться.

Желательно, чтобы учащиеся владели основами евклидовой геометрии: знали определение фигуры «квадрат» и имели представление о решении задач на построение. Кроме того, они должны изучить программу рисования лестницы (рис. 2), которая будет использоваться в качестве вспомогательного алгоритма для перемещения исполнителя по диагонали квадрата в противоположную вершину.

Ввиду того что поставленная задача не является типовой, **приведем возможный сценарий урока, подводящий школьников к догадке о решении.**

Учитель. Сегодня на занятии мы будем рисовать квадрат. Кстати, вы знаете, что такое квадрат?

Учащиеся. Конечно! Квадрат — это прямоугольник, у которого все стороны равны.

Учитель. Отлично! Тогда давайте напишем программу рисования самого большого квадрата, который может поместиться на поле графического исполнителя. Кто-нибудь может сразу это сделать?

Учащиеся. Надо записать 15 команд «шаг», чтобы нарисовать верхнюю сторону квадрата, потом 3 команды «поворот», чтобы повернуть вниз, потом снова 15 команд «шаг»...

Учитель. Да-да-да, так действительно можно сделать. Но согласитесь, что программа получится длинной и неинтересной. Можно ли сделать ее короче?

Учащиеся. Можно нарисовать верхнюю сторону квадрата с помощью цикла «пока впереди не край».

Учитель. Очень хорошо. А дальше?

Учащиеся. Потом надо повернуть вниз...

Учитель. Так. А что будем делать потом?

Учащиеся. Исполнитель должен сделать 15 шагов, чтобы нарисовать боковую сторону квадрата...

Учитель. И что же нам мешает это сделать? Почему мы не можем нарисовать боковую сторону так же, как нарисовали верхнюю?

Учащиеся. Мы не можем использовать цикл, потому что тогда исполнитель нарисует линию длиной 19 шагов, а не 15...

Учитель. То есть проблема в том, что исполнитель действует не на квадратном, а на прямоугольном поле? Наш исполнитель не умеет считать, и мы не можем придумать подходящее условие завершения цикла, которое заставило бы исполнителя остановиться после 15 шагов?

Учащиеся. Да, так и есть.

Учитель. Ни у кого нет идей, как заставить исполнителя опуститься вниз ровно на 15 шагов?



Рис. 2. Программа рисования лестницы используется в качестве вспомогательного алгоритма для перемещения исполнителя по диагонали квадрата. Пунктиром обведены команды, добавление которых позволяет значительно ускорить попадание исполнителя в конечную точку

Учащиеся. Можно с помощью цикла опустить исполнителя вниз до края и вернуть на 4 шага вверх прыжками! А потом нарисовать сторону квадрата, двигаясь вверх до самого края!

Учитель. Очень хорошее решение! Мы использовали наше знание о размерах рабочего поля исполнителя. Так можно и нужно делать. Однако сегодня давайте напишем универсальную программу, которая будет работать на рабочем поле любого размера, только бы его ширина была не больше высоты. Запомним идею с рисованием вверх — она будет работать и при таких ограничениях. Но как переместить исполнителя к вершине квадрата на его нижней стороне?

Учащиеся. ...

Учитель. Что ж, давайте думать вместе. А чтобы было нагляднее, возьмем прямоугольный лист бумаги, как поле нашего исполнителя. Как из него сделать квадрат, не производя никаких измерений?

Учащиеся. ...

Учитель. Давайте перегнем лист так, чтобы его меньшая сторона совместилась с большей. Тогда на большей стороне отложится длина меньшей, и мы сможем отметить на ней недостающую вершину квадрата (рис. 3).

Учащиеся. А чтобы найти другую вершину, надо совместить меньшую сторону со второй большей стороной!

Учитель. Правильно!

Учащиеся. Но как нам это поможет в написании программы?

Учитель. Посмотрите на лист бумаги. Какие еще линии кроме сторон проходят через вершины квадрата?

Учащиеся. Линии сгиба... Диагонали!

Учитель. Точно, диагонали квадрата! Наш исполнитель умеет рисовать диагональ?

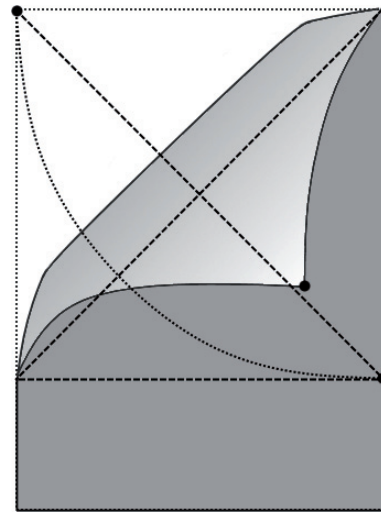


Рис. 3. Демонстрация диагоналей квадрата на сгибах прямоугольного листа бумаги

Учащиеся. Да, это алгоритм «лестница»!

Учитель. Тогда — за работу!

В результате должна получиться программа, подобная показанной на рисунке 4. В тело процедуры «диагональ» надо поместить текст программы рисования лестницы с единичными шагами. Чтобы диагональ не отображалась на рисунке, достаточно заменить в ней команды «шаг» на «прыжок». Такая программа выполняется в среде ГРИС «Букашка» более 12 минут, что ограничивает возможности ее демонстрации на уроке.

Существенно сократить это время можно, если выполнять рисование и перемещение отрезками утроенной

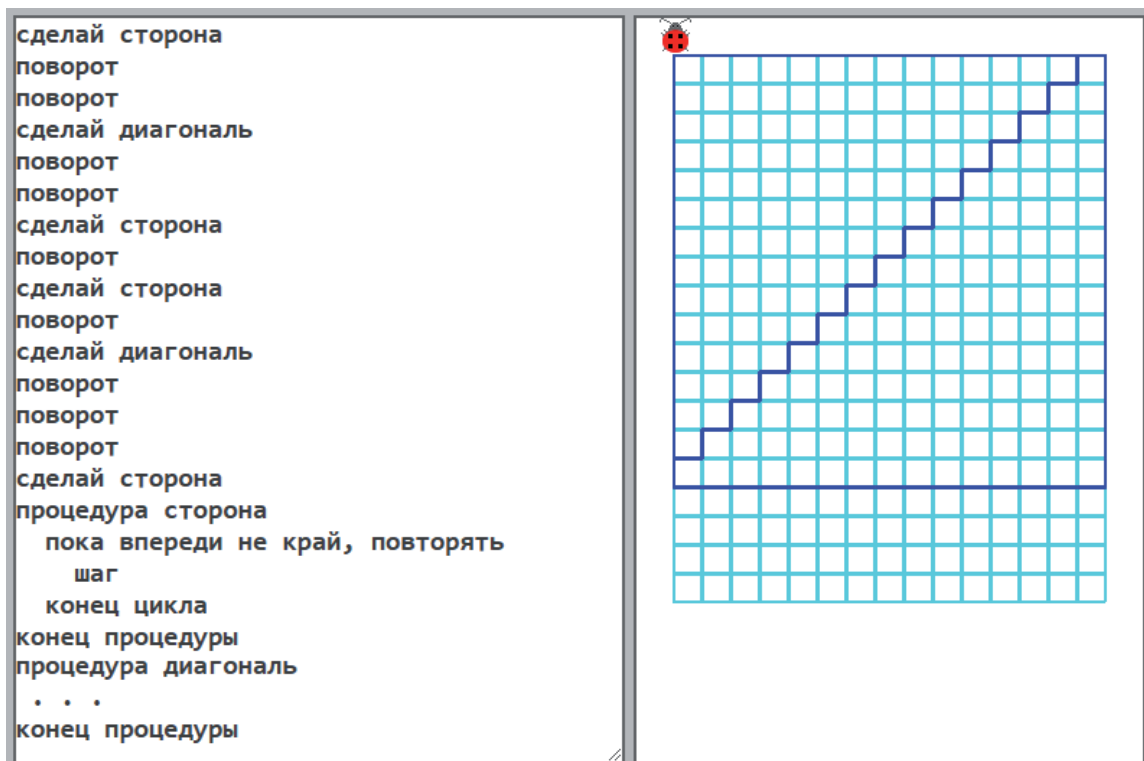


Рис. 4. Программа рисования квадрата отрезками единичной длины. В тело процедуры «диагональ» надо вставить текст программы рисования лестницы

длины. Для этого надо к каждой команде «шаг» и «прыжок» в процедурах «сторона» и «диагональ» дописать еще две такие же команды. Число 3 выбрано потому, что ему кратна ширина рабочего поля исполнителя. Это освобождает от необходимости дополнительных проверок достижения края в ходе рисования или перемещения.

Объяснить учащимся возможность такой оптимизации можно на примере программы рисования лестницы, показав, что исполнитель может достигнуть той же точки гораздо быстрее, рисуя ступени тройного размера (рис. 1). Имеет смысл обсудить с учащимися, за счет чего достигается экономия: из-за уменьшения количества итераций цикла с 15 до 5 происходит кратное уменьшение:

- числа «правых» поворотов, реализуемых тремя поворотами влево;
- числа выполнений управляющих конструкций, связанных с проверками условия и возвратами к началу тела цикла.

Можно отметить, что программы для реальных процессоров тоже содержат управляющие инструкции, которые влияют на производительность.

Для выполнения оптимизированной программы требуется уже шесть с половиной минут, что делает ее демонстрацию более реалистичной. А чтобы произвести дальнейшие улучшения, надо обратиться к рекурсии.

3.3. Введение в рекурсию

Начинать изучение рекурсии удобно с преобразования программы рисования лестницы. Лучше использовать ее вариант со ступенями не единичной, а тройной величины, потому что при этом существенно возрастает скорость демонстрации как обычного, так

и рекурсивного варианта. Помимо этого глубина рекурсии уменьшается с 15 до 5, что более чем достаточно для понимания сути, но при этом не наступает утомление от необходимости долго наблюдать монотонные операции.

Чтобы сохранить наглядность рекурсивного варианта программы, все непосредственные действия исполнителя лучше записать в процедуре «ступень», а рекурсивные вызовы осуществлять в процедуре «лестница». Это позволит очистить тело рекурсивной процедуры от несущественных в контексте изучаемого вопроса операций (особенно в случае трехкратных величин), кроме того, слово «лестница» несет в себе смысловую нагрузку множества ступеней и лучше отражает суть алгоритма: лестница является каркасом, на который устанавливаются ступени.

Первым можно продемонстрировать вариант программы с бесконечной рекурсией. В этом случае тело рекурсивной процедуры «лестница» (рис. 5) должно содержать только две команды:

- *прямой шаг рекурсии*, представленный вызовом процедуры «ступень»;
- *рекурсивный вызов* процедуры «лестница».

Этот пример убеждает учащихся в возможности многократного повторения действий без конструкции «цикл», но имеет существенную проблему — аварийное завершение.

Для решения проблемы команда рекурсивного вызова помещается в конструкцию неполной альтернативы, проверяющей условие недостижения края рабочей области. Этот вариант программы позволяет показать учащимся еще два элемента рекурсии:

- *условие продолжения* рекурсии;
- *отложенное выполнение* возвратов из процедуры.

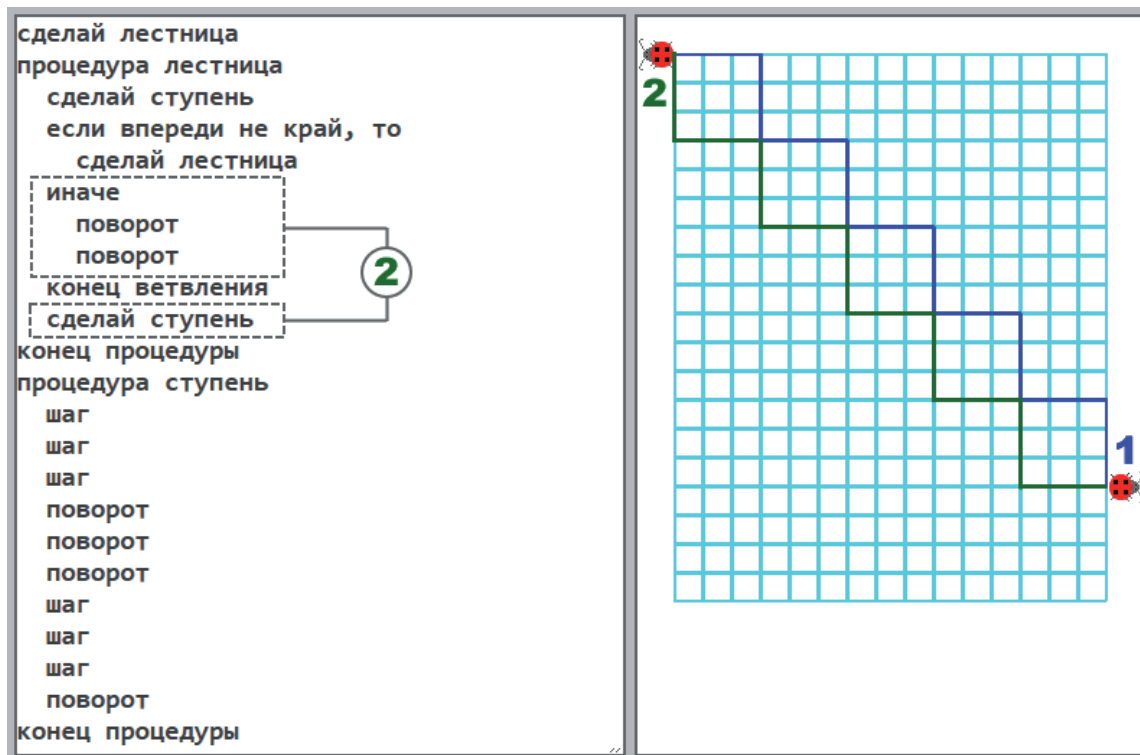


Рис. 5. Рекурсивная программа рисования лестницы:

1 — результат выполнения прямых шагов рекурсии, 2 — результат выполнения обратных шагов рекурсии

Чтобы раскрыть остальные возможности рекурсии, надо:

- расширить альтернативу до полной, добавив в нее ветвь, которая будет выполняться при нарушении условия рекурсии;
- после альтернативы добавить *обратный шаг рекурсии*, представленный вызовом процедуры «ступень».

Дополнения, которые надо внести в программу, и результат ее работы обозначены на рисунке 5 цифрой 2.

При демонстрации программы важно обратить внимание школьников на то, что количество как прямых, так и обратных шагов рекурсии в точности равно количеству рекурсивных вызовов.

В приведенном на рисунке 5 виде рекурсия реализует *аналог цикла ДО*, поскольку условие продолжения проверяется после выполнения прямого шага, который выступает аналогом тела цикла. Если необходимо реализовать *аналог цикла ПОКА*, то команды прямого шага рекурсии следует поместить в альтернативу перед командой рекурсивного вызова. А для сохранения баланса прямых и обратных вызовов и команды обратного вызова надо поместить в ту же ветвь альтернативы, но уже после команды рекурсивного вызова. При выполнении такой программы последний рекурсивный вызов будет приводить только к выполнению действий, предусмотренных для ситуации нарушения условия продолжения рекурсии, без прямого и обратного шагов.

3.4. Рекурсивная программа рисования квадрата

На данном этапе у учащихся имеются все знания, которые необходимы для разработки рекурсивной программы решения задачи рисования квадрата, сформулированной в начале этого раздела. Но, чтобы направить

их к решению, возможно, потребуется показать, что команды обратного шага не обязательно должны выполнять те же действия, что и команды прямого шага. Подпрограмма «ступень» в программе на рисунке 5 для обоих шагов рекурсии была использована, чтобы сократить текст программы. Замена вызова этой подпрограммы для обратного шага тремя командами «прыжок» приведет к перемещению исполнителя по горизонтальной линии к левой границе рабочего поля.

Теперь можно предложить ребятам написать программу рисования двух сторон квадрата, имеющих общую вершину. Решение приведено на рисунке 6 в процедуре «две_стороны». Скорее всего, ученики сразу догадаются, что для рисования оставшейся части квадрата надо повернуть исполнителя и выполнить процедуру «две_стороны» еще раз.

Рекурсивная программа рисования квадрата выполняется за три минуты сорок шесть секунд, что гораздо быстрее оптимизированного, но не рекурсивного варианта. При этом исполнитель не совершает ни одного лишнего действия, каждый шаг ведет к цели. Ради такого результата стоит приложить усилия для изучения рекурсии!

3.5. Дополнительные возможности рекурсии

Обычно говорят, что графический исполнитель не умеет считать. Но, как мы только что убедились, рекурсия позволяет если не считать, то *подсчитывать* количество выполненных прямых шагов и выполнять такое же количество обратных шагов. Уже отмечалось, что действия, выполняемые на обратном шаге, могут отличаться от действий на прямом шаге. А что, если на прямом шаге рекурсии перемещать исполнителя вперед на Р единиц, а на обратном шаге — возвращать обратно на Q единиц? Тогда в итоге исполнитель

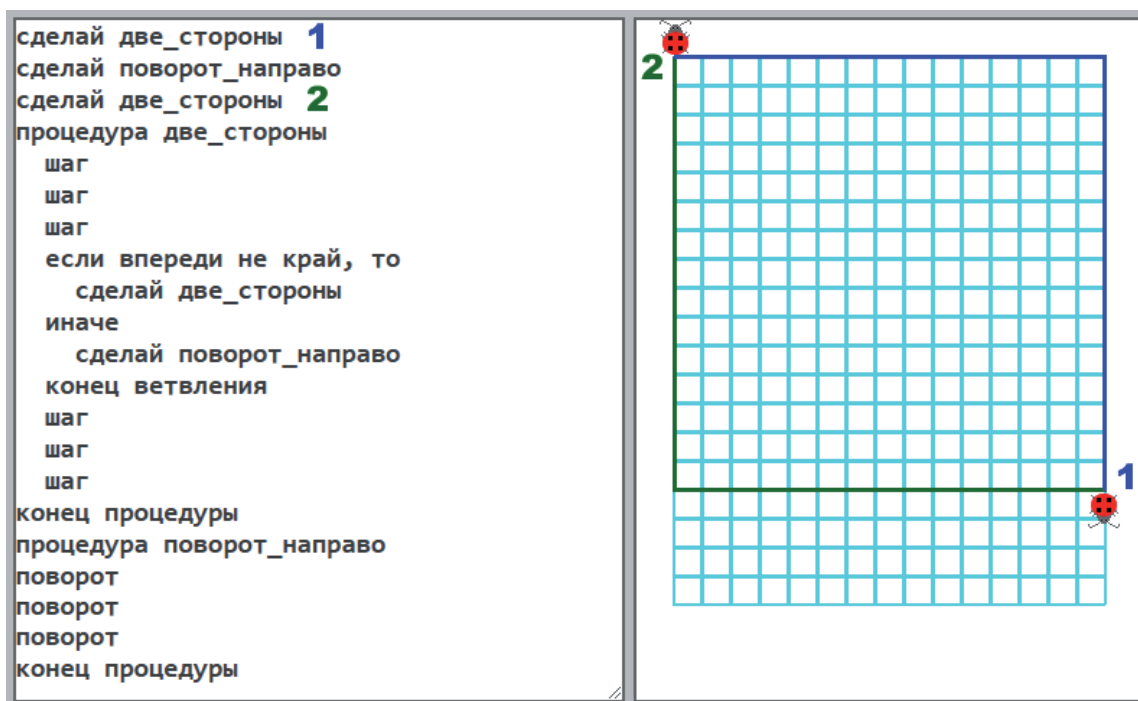


Рис. 6. Рекурсивная программа рисования квадрата:

1 — положение исполнителя после первого вызова процедуры «две_стороны», 2 — после второго вызова

должен оказаться в позиции, которая делит отрезок, по которому прошел его прямой путь, в отношении $(P - Q) : Q$ (рис. 7).

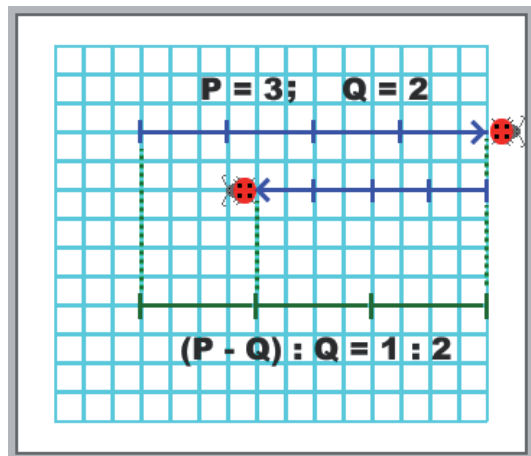


Рис. 7. Деление отрезка в результате прямых и обратных рекурсивных шагов разной длины

С помощью этого знания можно написать программу, которая устанавливает исполнителя в центре рабочего поля (рис. 8). Средняя точка делит отрезок в отношении 1:1, что реализуется выбором, например, $P = 2$ и $Q = 1$. Процедура «поиск_середины» перемещает исполнителя в середину отрезка от его текущего положения до края рабочей области. Перемещение в центр рабочей области обеспечивается последовательным применением процедуры «поиск_середины» для горизонтального и вертикального направлений.

Поскольку размеры рабочей области заданы нечетным числом клеток, ее геометрический центр лежит за

пределами возможных траекторий исполнителя. При выполнении процедуры «поиск_середины» последний прямой шаг рекурсии совершает неполное перемещение исполнителя (аварийная ситуация предотвращается специально введенной проверкой в процедуре «два_прыжка»), а все обратные шаги выполняются полностью. В результате исполнитель на последнем обратном шаге рекурсии перепрыгивает через геометрическую середину отрезка и оказывается в точке, соответствующей округлению результата деления до ближайшего целого, не превышающего округляемое значение: $[15 / 2] = 7$, $[19 / 2] = 9$.

В качестве эксперимента рекомендуется повторить выполнение программы (можно в режиме быстрого получения результата с помощью пункта меню «F5результат») и объяснить достигнутое после этого новое положение исполнителя. С помощью процедуры «поиск_середины» можно строить более сложные изображения, например, ромб, который изображен на снимке экрана ГРИС «Кенгурёнок Ру» (рис. 1).

Отметим, что при использовании описанного метода «деления отрезка в отношении» вычисления производит не исполнитель, а программист, выбирая значения P и Q , с учетом которых потом разрабатывается программа решения задачи. Такой прием адаптации исходной задачи к возможностям исполнителя применяется и в промышленном программировании. Например, работу с угловыми величинами на микроконтроллерах с ограниченными вычислительными возможностями организуют с помощью таблично-алгоритмических методов в целочисленной арифметике, для чего предварительно рассчитывают значения тригонометрических функций в узловых точках с шагом, обеспечивающим требуемую точность [9].

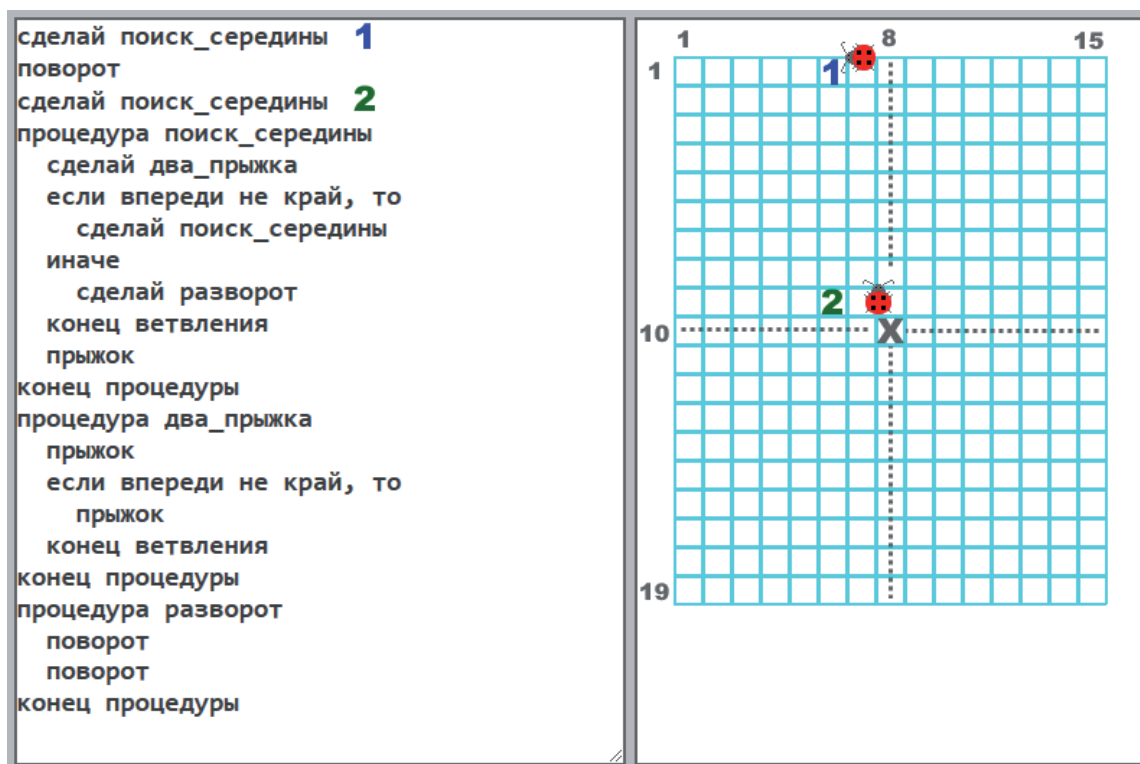


Рис. 8. Программа перемещения исполнителя в центр рабочей области

4. Заключение

В школьном курсе информатики целесообразно использовать программные средства учебного назначения, одной из категорий которых являются учебные исполнители. Простейшие графические исполнители способствуют выработке у школьников начальных навыков алгоритмического мышления и предоставляют возможность выражения их творческого потенциала, но для ребят постарше они могут показаться слишком простыми. Для восстановления интереса к изучению ГРИС и расширения образовательной области предлагается обратиться к решению нестандартных задач с помощью рекурсии.

Среда исполнителя ГРИС позволяет на наглядных примерах отработать следующие учебные вопросы:

- рекурсивный вызов процедуры;
- отложенный возврат из процедуры;
- условие продолжения рекурсии;
- прямой и обратный шаги рекурсии.

Проведение аналогии между циклическими программами и программами, использующими рекурсию, дает возможность объяснить различие между циклом ПОКА и циклом ДО, несмотря на то что в системе команд графического исполнителя конструкция цикла ДО отсутствует.

Преимущество использования рекурсии при решении некоторых задач наглядно продемонстрировано в ходе улучшения программы рисования квадрата. Использование рекурсии расширяет возможности позиционирования графического исполнителя, что открывает путь к решению более интересных задач.

За рамками рассмотрения этой статьи остались такие вопросы, как косвенная и параллельная рекурсии. Отсутствие в системе команд ГРИС процедур с параметрами и локальными переменными не позволяет продемонстрировать сохранение и восстановление контекста между рекурсивными вызовами. Однако обращение к теме рекурсии при изучении ГРИС позволяет сформировать у учащихся ментальную алгоритмическую схему рекурсивных алгоритмов и подготовить их к изучению более сложных вопросов на дальнейших стадиях обучения.

Для применения изложенной в статье методики на практике предлагается использовать независимую от вычислительной платформы реализацию ГРИС «Букашка», которая совместима с ГРИС «Кенгурёнок РУ» и может быть интегрирована в информационную среду образовательной организации.

Список источников

1. Боброва И. И., Новак А. В. Роль и место элементов программирования в школьном курсе информатики // Новые информационные технологии в образовании и науке НИТО-2017. Материалы X международной научно-практической конференции (Екатеринбург, 27 февраля — 3 марта 2017 г.). Екатеринбург: РГППУ, 2017. С. 449—453. EDN: YQFIYH.

2. Босова Л. Л. Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе // Информатика в школе. 2018. № 6. С. 3—11. EDN: XZOOJV.

3. Босова Л. Л. О новых подходах к изучению школьной информатики в условиях цифровой трансформации общества // Информатика в школе. 2022. № 4. С. 5—14. EDN: DKRLZV. DOI: 10.32517/2221-1993-2022-21-4-5-14.

4. Гейн А. Г., Житомирский В. Г., Линецкий Е. В., Сапир М. В., Шолохович В. Ф. Основы информатики и вычислительной техники: пробный учебник для 10—11 классов средних школ. М.: Просвещение, 1991. 254 с.

5. Григорьев С. Г., Гринишун В. В. Информатизация образования. Фундаментальные основы и практические приложения: учебник для студентов педагогических вузов и слушателей системы повышения квалификации педагогов. Воронеж: Научная книга, 2014. 232 с.

6. Гришаева А. Ю. Методические аспекты формирования алгоритмического и творческого мышления учащихся начальной школы // Ratio et Natura. 2022. № 1. EDN: NUNFUM.

7. Калинина В. В. Методика ментального обучения программированию студентов информационных направлений подготовки // Вестник КГПУ. 2015. № 1. С. 45—48. EDN: TLMJTI.

8. Кузенкова Г. В., Штанюк А. А., Шагбазян Д. В. Обучение программированию в средней и высшей школе: проблемы и пути решения // Современные проблемы науки и образования. 2021. № 1. EDN: NSQVNH. DOI: 10.17513/spno.30514.

9. Ледовской М. И. Алгоритм вычисления функций Sin(x) и Cos(x) для 16-разрядных микроконтроллеров // Известия ЮФУ. Технические науки. 2014. № 4. С. 164—170. EDN: SDJKKJ.

10. Мазда Дж. Законы простоты: Дизайн. Технологии. Бизнес. Жизнь. М.: Альпина Паблишер, 2016. 116 с.

11. Орещенко И. С. Реализация учебно-игровой среды «Букашка» средствами клиентских веб-технологий // Информатика в школе. 2021. № 6. С. 32—38. EDN: PVCZCL. DOI: 10.32517/2221-1993-2021-20-6-32-38.

12. Паклина Е. П. Задачи повышенной сложности с решениями в среде графического исполнителя Кенгурёнок // Фестиваль педагогических идей «Открытый урок». <https://urok.1sept.ru/articles/101993>

13. Пирогов В. Ю., Баландина И. В. Основы методики обучения рекурсивному программированию // Интернет-журнал «Мир науки». 2018. Т. 6. EDN: YLKSTB.

14. Прилепина А. В. Формирование и развитие алгоритмического и логического мышления в пропедевтическом и базовом курсе информатики // Современный урок в условиях внедрения ФГОС: опыт, проблемы, перспективы. Всероссийская научно-методическая конференция. Сборник статей (Оренбург, 28 ноября — 1 декабря 2016 г.). Оренбург: ОГПУ, 2017. С. 201—204. EDN: XSFTXB.

15. Пузанкова Л. В. Особенности преподавания информатики в начальной школе // Общество: социология, психология, педагогика. 2021. № 5. С. 155—159. EDN: AYNLIV. DOI: 10.24158/spp.2021.5.27.

16. Роберт И. В. Современные информационные технологии в образовании: дидактические проблемы; перспективы использования. М.: ИИО РАО, 2010. 140 с. EDN: MWKHQR.

17. Саблина О. М. О роли нестандартных задач в развитии логического мышления школьников // Молодой ученый. 2015. № 10. С. 1280—1283. EDN: TWARD.

18. Семакин И. Г., Залогова Л. А., Русаков С. В., Шестакова Л. В. Информатика. 9 класс: учебник. 3-е изд., стер. М.: Просвещение, 2022. 208 с.