DOI: 10.32517/2221-1993-2022-21-3-36-48

М. А. Плаксин
дипломант Всероссийского конкурса «Школьная информатика: вчера, сегодня, завтра» в номинации «Школьная информатика: вчера, сегодня, завтра», Национальный исследовательский университет «Высшая школа экономики» (Пермский филиал); Пермский государственный национальный исследовательский университет, г. Пермь, Россия

ИГРЫ-ХОДИЛКИ ДЛЯ ИЗУЧЕНИЯ БЛОК-СХЕМ*

Аннотация

В статье описывается методика освоения блок-схем с помощью настольных игр-ходилок.

Игровое поле в такой игре представляет собой блок-схему. В ходе игры учащиеся бросают кубик и передвигают фишку по игровому полю в зависимости от выпавшего числа. Но при этом они должны соблюдать правила исполнения входящих в блок-схему алгоритмических конструкций.

Цель предлагаемого подхода — интенсифицировать освоение учебного материала и освободить время для его расширения. Это актуально для всех школьных предметов, но для информатики с ее крайне быстрыми темпами развития — особенно.

Для игры класс разбивается на команды по два—четыре человека. Все команды играют одновременно. В ходе игры каждый участник многократно проходит все изученные алгоритмические конструкции. Таким образом достигается стопроцентный охват учащихся и многократное повторение материала.

За счет этого появляется возможность продвинуться в изучении алгоритмических конструкций достаточно далеко. А именно: кроме традиционных развилок и двух видов циклов (с предусловием и с заданным числом повторений) изучаются команды ввода, подпрограммы (вспомогательные алгоритмы) и параметры (в играх — только входные).

Знакомство с механизмом подпрограмм, причем параметризованных, позволит в будущем рассматривать достаточно сложные алгоритмы и плавно перейти к пропедевтике технологии программирования (наращивание мощности простых исполнителей за счет разработки мощных команд-подпрограмм, проектирование алгоритмов «сверху вниз», построение алгоритмов из крупных заготовок).

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_03-2022/

Контактная информация

Плаксин Михаил Александрович, канд. физ.-мат. наук;
доцент кафедры информационных технологий в бизнесе, Национальный исследовательский университет «Высшая школа экономики» (Пермский филиал), г. Пермь, Россия; *адрес:* 614070, г. Пермь, ул. Студенческая, д. 38;
доцент кафедры математического обеспечения вычислительных систем, Пермский государственный национальный исследовательский университет, г. Пермь, Россия; *адрес:* 614990, г. Пермь, ул. Букирева, д. 15;
e-mail: mapl@list.ru

M. A. Plaksin,
National Research University Higher School of Economics, Perm Branch;
Perm State National Research University, Perm, Russia

BOARD GAMES FOR THE STUDY OF FLOW CHARTS

Abstract

The article presents the description of method of the study of flow charts using board games.

The playing field in such a game is a flow chart. During the game, students roll the dice and move the chip around the playing field, depending on the number that has fallen. But at the same time, they must comply with the rules for the execution of the algorithmic structures included in the flow chart.

The purpose of the proposed teaching method is to intensify the learning of educational material. This is especially important for informatics with its extremely fast rates of development.

For the game, the class is divided into teams of 2–4 people. All teams play at the same time. During the game, each participant repeatedly goes through all the studied algorithmic constructions. Thus, one hundred percent coverage of students and repeated repetition of the material is achieved.

Due to this, it becomes possible to advance far enough in the study of algorithmic constructions. In addition to traditional forks and two types of loops (with a precondition and with a given number of repetitions), input commands, subprograms and parameters are studied.

The methodic were prepared as part of the Perm version of the propaedeutic course of informatics (team of authors: M. A. Plaksin, N. G. Ivanova, O. L. Rusakova; the working title of the course is "TRIZformatika"). But the technique can be applied in any other course.

The article provides a description and playing fields of four board games for mastering the mechanism of flow charts, and also gives a number of methodological comments.

Keywords: informatics, secondary school, primary school, teaching methods, flow chart, algorithmic constructions, board game, TRIZformatics, TRIZformatika.

Предлагаемые методические материалы подготовлены в рамках работ над Пермской версией пропедевтического курса информатики (авторский коллектив: М. А. Плаксин, Н. Г. Иванова, О. Л. Русакова; рабочее название курса — «ТРИЗформатика»). Но методика может применяться в любом другом курсе.

В статье приведены описание и игровые поля четырех игр-ходилок для освоения механизма блок-схем, а также дан ряд методических замечаний.

Ключевые слова: информатика, средняя школа, начальная школа, методика обучения, блок-схема, алгоритмические конструкции, игра-ходилка, ТРИЗформатика.

1. Введение

Стремительное развитие информационных технологий требует постоянного обновления школьного курса информатики и включения в него нового материала. При этом старый материал в значительной мере должен сохраниться. То есть речь идет о содержательном расширении курса без увеличения отводимых на него часов. Сделать это можно только за счет интенсификации образовательного процесса, только научившись давать учащимся за то же время больше учебного материала, научившись прежнее содержание излагать за меньшее количество уроков. Это актуально для всех школьных предметов. Но для информатики с ее чрезвычайно высокими темпами развития — особенно.

Настоящая статья предлагает для изучения такой традиционной темы, как представление алгоритмов в виде блок-схем, использовать нетрадиционный прием — игры-ходилки. Игровое поле в такой игре представляет собой блок-схему. В ходе игры учащиеся бросают кубик и передвигают фишку по игровому полю в зависимости от выпавшего числа. Но при этом они должны соблюдать правила исполнения входящих в блок-схему алгоритмических конструкций.

Цель предлагаемого подхода — интенсифицировать освоение учебного материала.

Для игры класс разбивается на команды по два—четыре человека. Все команды играют одновременно. В ходе игры каждый участник многократно проходит все изученные алгоритмические конструкции. Таким образом достигается стопроцентный охват учащихся и многократное повторение материала.

За счет этого появляется возможность продвинуться в изучении алгоритмических конструкций достаточно далеко. А именно: кроме традиционных развилок и двух видов циклов (с предусловием и с заданным числом повторений) изучаются команды ввода, подпрограммы (вспомогательные алгоритмы) и параметры.

Несколько слов о параметрах. При обсуждении этого понятия в примерах возникают параметры как входные, так и выходные (например, подпрограмме сложения многозначных чисел нужен выходной параметр для передачи результата). Однако внимание на этом не акцентируется. Про деление параметров на входные и выходные и способы передачи параметров ничего не говорится. В играх используются только входные параметры.

Знакомство с механизмом подпрограмм, причем параметризованных, позволит в будущем рассматривать достаточно сложные алгоритмы и плавно перейти к пропедевтике технологии программирования (наращивание мощности простых исполнителей за счет разработки мощных команд-подпрограмм, проектирование алгоритмов «сверху вниз», построение алгоритмов из крупных заготовок).

В данный момент от учащихся не требуется построения блок-схем, нужны только чтение и исполнение алгоритмов. Построение алгоритмов — материал более поздний. (Почему-то при изучении естественных языков кажется нормальным, что ребенок сначала научается читать, а потом в течение многих лет в школе осваивает письмо. А вот при изучении алгоритмики ребенку сразу предлагают писать, хотя он еще читать не научился.)

Предлагаемые методические материалы подготовлены в рамках работ над Пермской версией пропедевтического курса информатики (авторский коллектив: М. А. Плаксин, Н. Г. Иванова, О. Л. Русакова; рабочее название курса — «ТРИЗформатика») [9, 10]. Но применение данной методики с курсом ТРИЗформатики никак не связано.

Далее приводятся описание и игровые поля четырех игр-ходилок для освоения механизма блок-схем и еще несколько методических замечаний.

2. Правила игры «Блок-схема-Бродилка»

«Блок-схема-Бродилка» — игра-ходилка. Цель игры — переместить фишку по шагам от начала до конца блок-схемы. Количество шагов на каждом ходе определяется бросанием кубика. Развилки определяются правилами записи блок-схем: условие (ветвление и цикл ПОКА), цикл ПОВТОРИТЬ.

Блоки перенумерованы. Номера играют чисто информационную роль, служат «именами» блоков и предназначены для однозначного определения позиции.

Играть может любое количество игроков. Разумное число — от двух до четырех.

Каждому игроку полезно завести карточку для ведения записей в ходе игры.

Каждый игрок двигает свою фишку, отличную от других.

Первоначально все фишки ставятся на блок «Начало».

Игроки по очереди бросают кубик и передвигают свою фишку по блокам на выпавшее количество шагов.

Если на пути встречается заголовок цикла ПОВТОРИТЬ (блок-шестиугольник), то фишка должна зайти в тело цикла и повторить цикл указанное число раз. Игрок должен отслеживать (отмечать на своей карточке), сколько повторов цикла он уже выполнил.

Если на пути встречается блок условия (блок-ромб), то число, выпавшее на кубике (количество шагов на данном ходе), подставляется в записанное в ромбе условие, которое и определяет направление дальнейшего движения. Если фишка попала на блок условия на последнем шаге данного хода, то направление движения будет определяться следующим броском кубика (на следующем ходе).

На рисунке 1 приведено игровое поле игры «Бродилка-1». Рассмотрим пример игры.

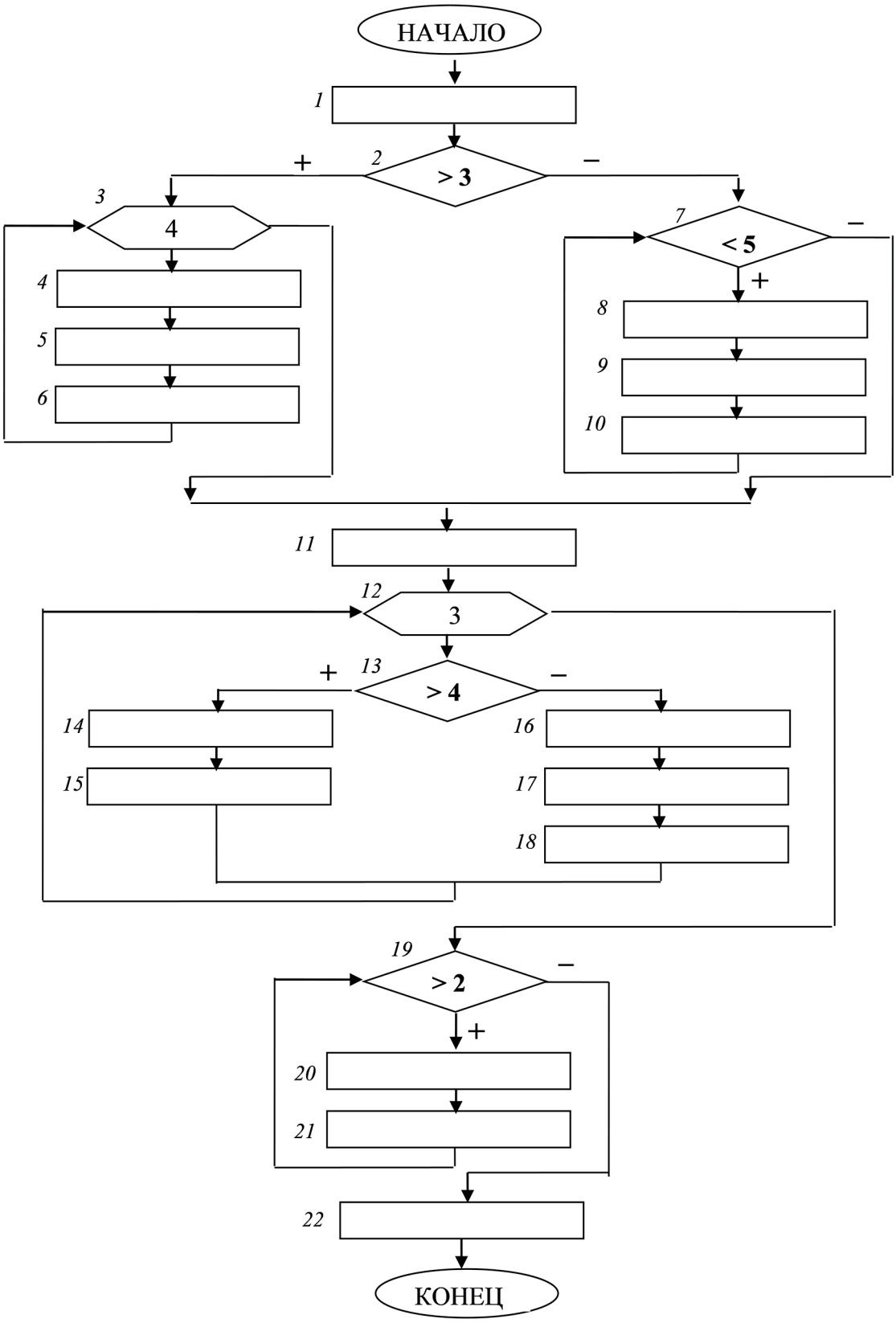

Рис. 1. Игровое поле (блок-схема) игры «Бродилка-1»

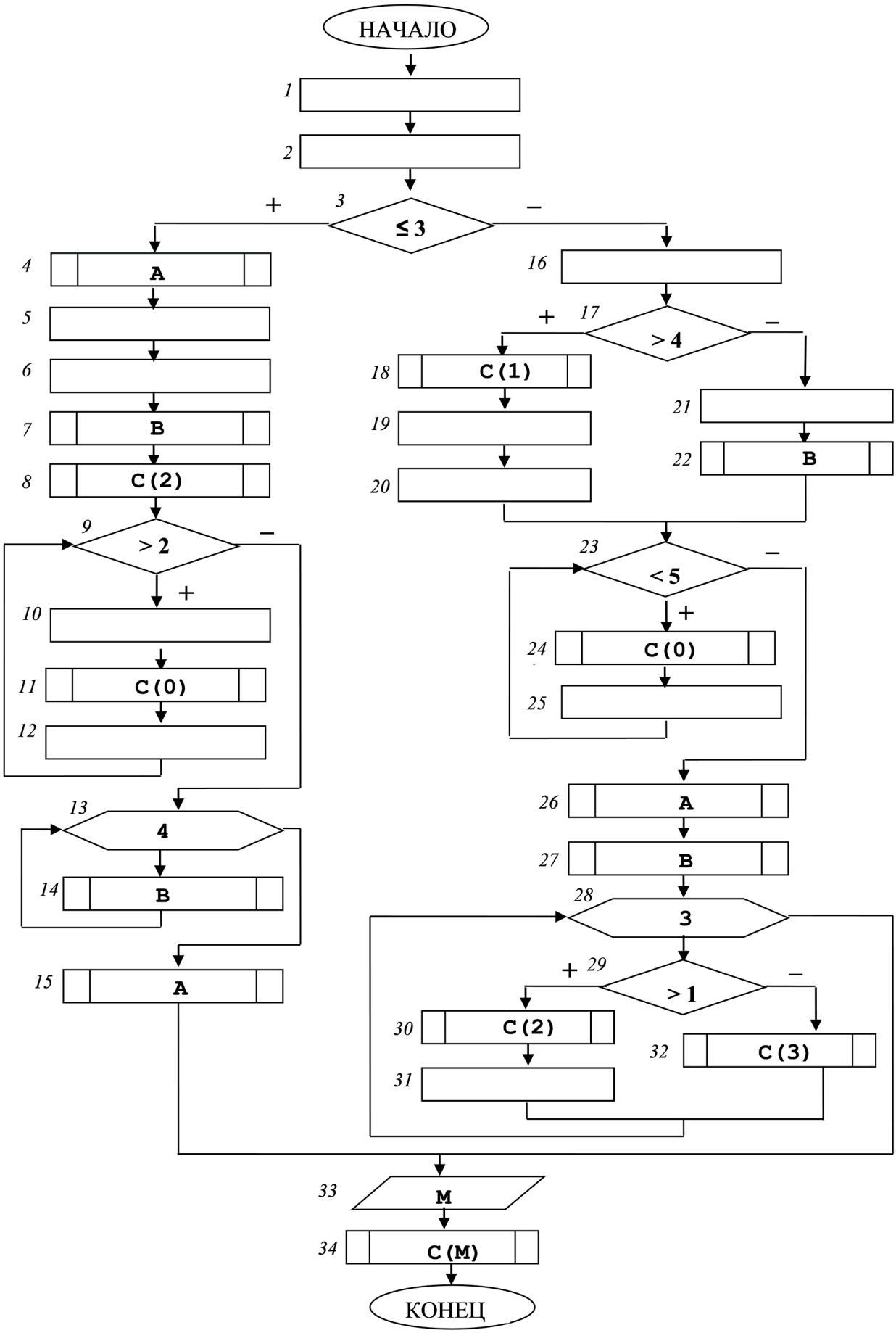

Рис. 2. Игровое поле игры «Бродилка-2». Блок-схема главной программы

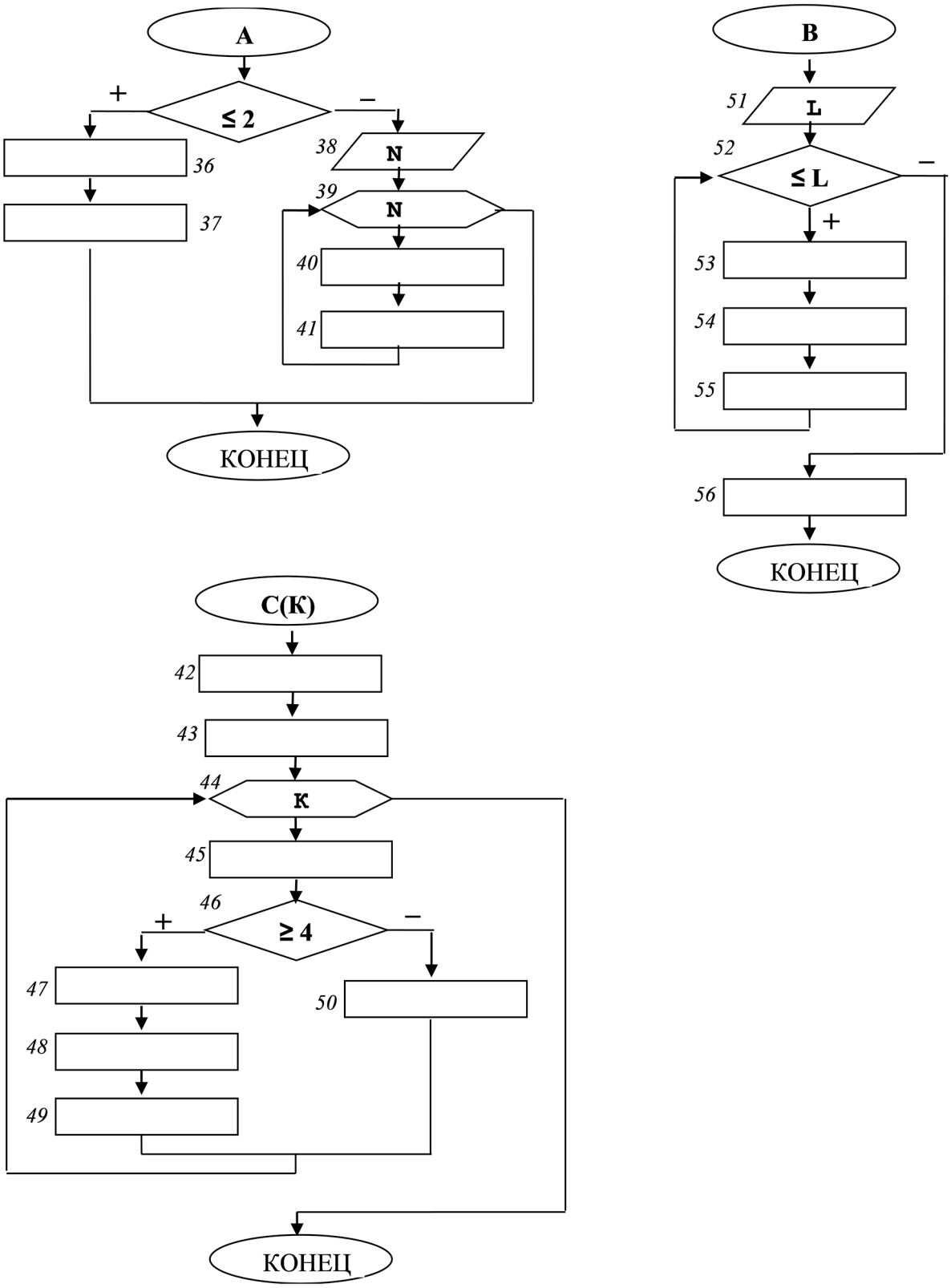

Рис. 3. Игровое поле игры «Бродилка-2». Блок-схемы подпрограмм

Пусть при первом броске на кубике выпало число 3. Игрок передвигает фишку по блоку 1 и доходит до блока 2. В блоке 2 стоит условие «> 3». В это условие надо подставить число, выпавшее на кубике, т. е. 3. Получится условие «3 > 3». Это условие ложно, поэтому движение продолжается по ветке «минус». Третьим шагом фишка встанет на блок 7. Это блок-условие. Фишка попала на него на последнем шаге данного хода. Право хода переходит к другому игроку. Направление движения данной фишки будет определено только на следующем ходе. Если на кубике выпадет меньше 5, фишка должна войти в цикл (на блок 8). Если выпадет больше или равно 5, фишка должна пойти на блок 11 и далее.

Продолжение правил для игры «Бродилка-2».

Если на пути встречается блок вызова подпрограммы (блок-прямоугольник с двойными стенками), то игрок должен увести свою фишку на блок-схему названной подпрограммы, пройти ее от начала до конца и после этого продолжить выполнение с блока, следующего за блоком вызова подпрограммы.

Если подпрограмма параметризована, игроку имеет смысл записать значение параметра на свою карточку. (Подпрограммы вызываются по нескольку раз с разными значениями параметров.)

Последний вид блока, который встречается на блок-схеме, это блок ввода данных (блок-параллелограмм). В параллелограмме указано имя переменной, значение которой надо ввести. Для этого игрок бросает кубик. Выпавшее значение становится значением этой переменной и используется в следующем блоке. Имя и значение переменной игрок записывает на свою карточку. Переменная может использоваться для сравнения в блоке-условии, для определения количества повторов цикла ПОВТОРИТЬ и для передачи в подпрограмму в качестве параметра. Обратим внимание, что ввести значение переменной надо даже в том случае, если фишка просто прошла через блок ввода данных, не остановившись на нем. Дело в том, что введенное значение будет задействовано в следующих блоках.

На рисунках 2 и 3 приведено игровое поле игры «Бродилка-2» (на рисунке 2 — блок-схема главной программы, на рисунке 3 — блок-схемы подпрограмм).

3. Правила игры «Блок-схема-Вычислялка»

«Блок-схема-Вычислялка» — игра-ходилка. Игровое поле представляет собой блок-схему. Цель игры — переместить фишку по шагам от начала до конца блок-схемы. Количество шагов на каждом ходе определяется бросанием кубика. Развилки определяются правилами записи блок-схем: условие (ветвление и цикл ПОКА), цикл ПОВТОРИТЬ.

Играть может любое количество игроков. Разумное число — от двух до четырех.

Каждый игрок должен завести карточку для ведения записей в ходе игры.

Каждый игрок двигает свою фишку, отличную от других.

Первоначально все фишки ставятся на блок «Начало».

Игроки по очереди бросают кубик и передвигают свою фишку по блокам на выпавшее количество шагов.

По ходу игры будут использоваться «переменные величины». Каждая такая величина имеет имя (в этой игре оно обозначается одной латинской буквой) и значение. Значение величины может изменяться по ходу игры (на то она и переменная!). У каждого игрока есть карточка, на которой обязательно записывается значение «величины N». Остальные переменные будут записываться в том случае, когда фишка игрока проходит через блок ввода данных, в котором записано имя соответствующей переменной.

Перед началом игры игрок бросает кубик. Выпавшее число является начальным значением величины N этого игрока.

В блоках-действиях записаны выражения, состоящие из знаков +/– и числа.

В блоках-условиях записаны выражения, состоящие из N, знака сравнения и числа.

Если при движении по игровому полю фишка останавливается на блоке-действии, то указанное действие надо выполнить над величиной N (прибавить к N или отнять от N указанное число). Результат операции становится новым значением числа N.

Если на пути встречается заголовок цикла ПОВТОРИТЬ (блок-шестиугольник), то фишка должна зайти в тело цикла и повторить цикл указанное число раз. Игрок должен отслеживать (отмечать на своей карточке), сколько повторов цикла он уже выполнил.

Если при движении фишка проходит через блок-условие (блок-ромб), то величина N подставляется в операцию сравнения и направление движения определяется результатом сравнения.

На рисунке 4 приведено игровое поле игры «Вычислялка-1». Рассмотрим пример игры.

Пусть начальное значение N, определенное броском кубика, равно 5.

Пусть при первом броске на кубике выпало число 6. Игрок передвигает фишку по блоку 1 и доходит до блока 2. В блоке 2 стоит условие «N < 10». В это условие надо подставить значение переменной N, т. е. 6. Получится условие «6 < 10». Это условие истинно, поэтому движение продолжается по ветке «плюс». То есть фишка входит в цикл. Третьим и четвертым шагами фишка проходит блоки, составляющие тело цикла («+3» и «–1»). И пятым шагом опять оказывается на заголовке цикла — блоке условия «N < 10». Поскольку значение переменной N не изменилось, фишка на шестом шаге опять входит в цикл и заканчивает движение на первом блоке тела цикла «+3». Это блок-действие. Это действие — «+3» — выполняется с переменной N. Ее значение увеличивается на 3 и становится равно 8. На этом ход данного игрока заканчивается. Право броска кубика переходит к следующему игроку.

Продолжение правил для игры «Вычислялка-2».

Если на пути встречается блок вызова подпрограммы (блок-прямоугольник с двойными стенками), то игрок должен увести свою фишку на блок-схему названной подпрограммы, пройти ее от начала до конца и после этого продолжить выполнение с блока, следующего за блоком вызова подпрограммы.

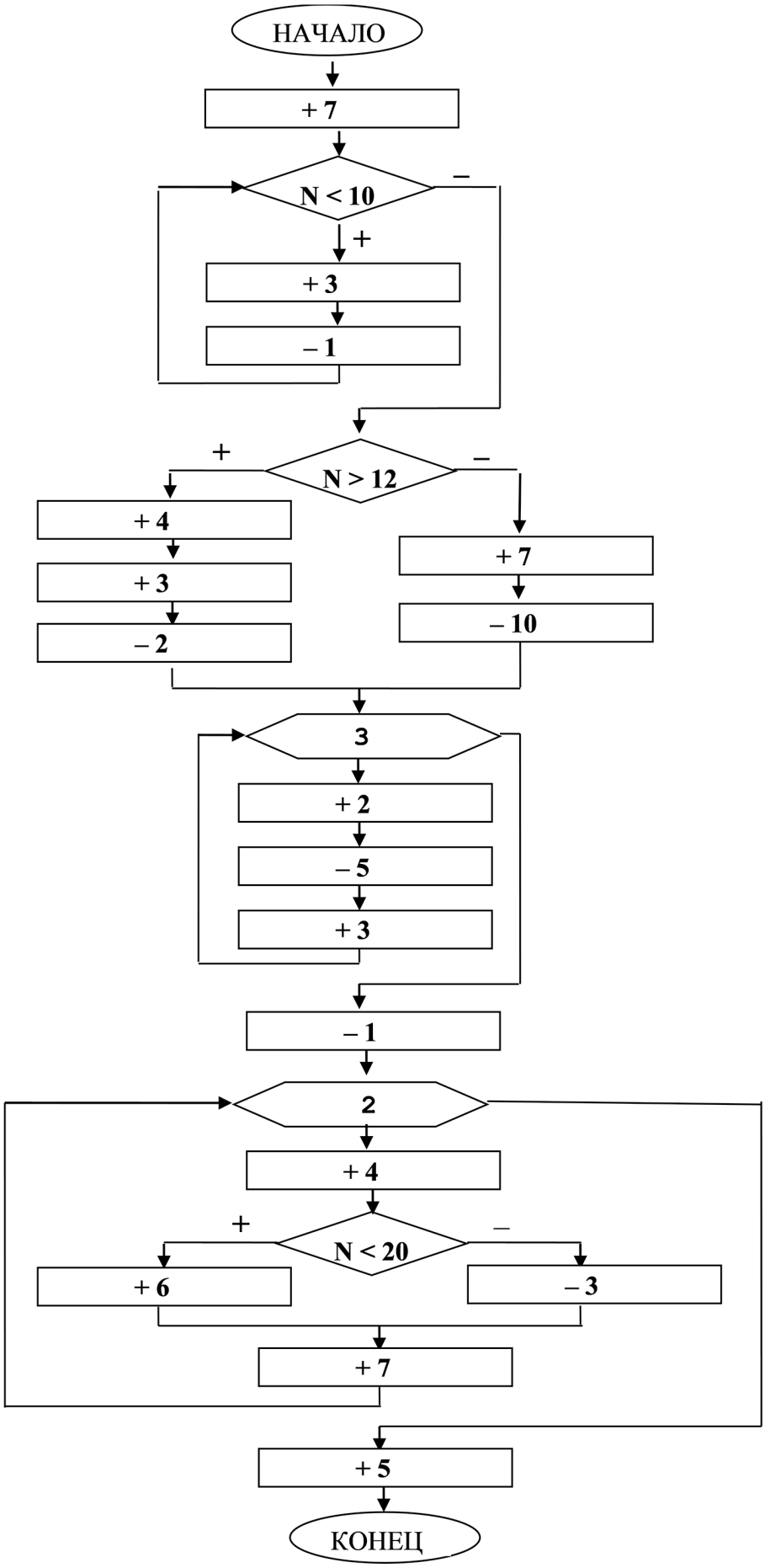

Рис. 4. Игровое поле (блок-схема) игры «Вычислялка-1»

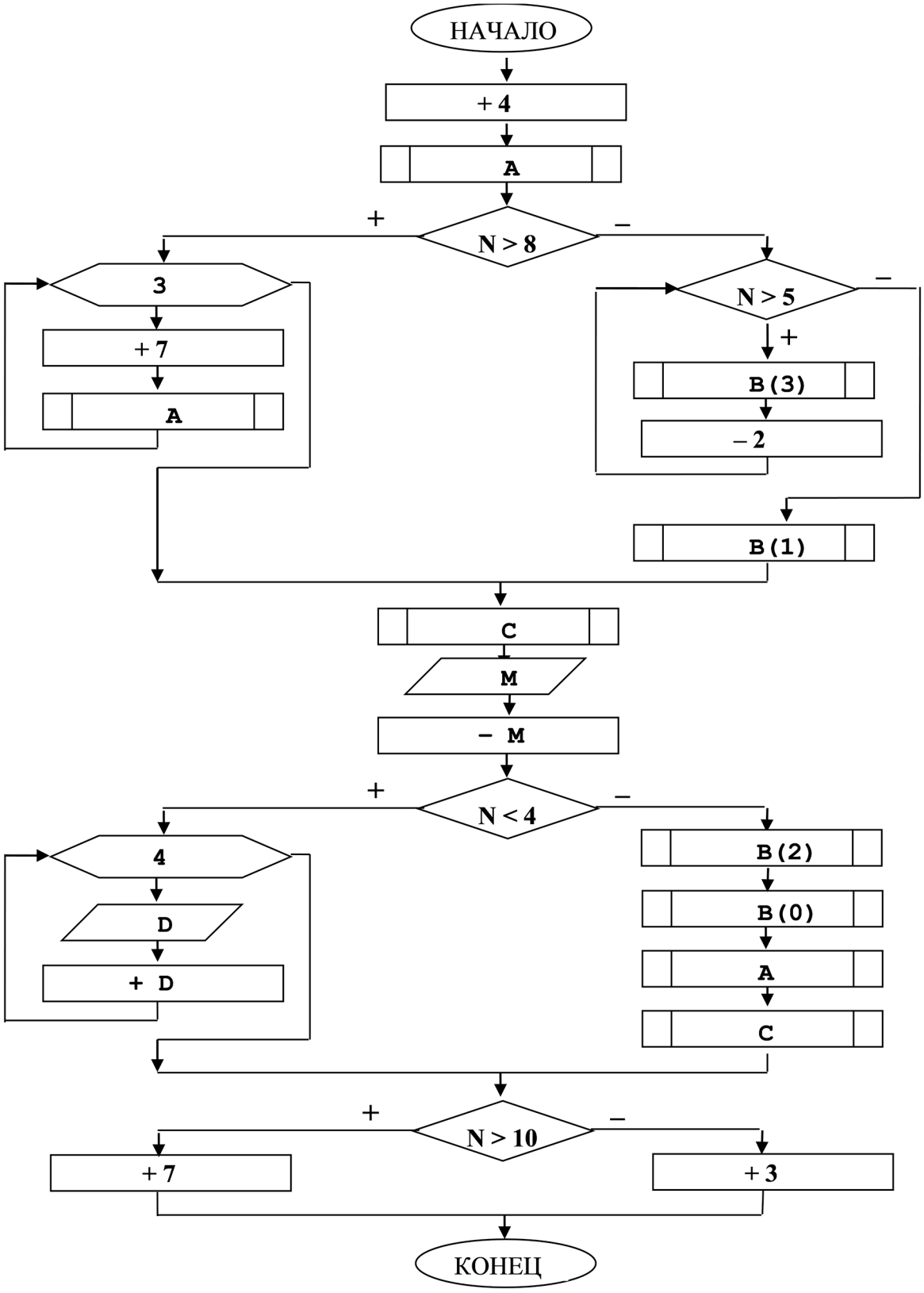

Рис. 5. Игровое поле игры «Вычислялка-2». Блок-схема главной программы

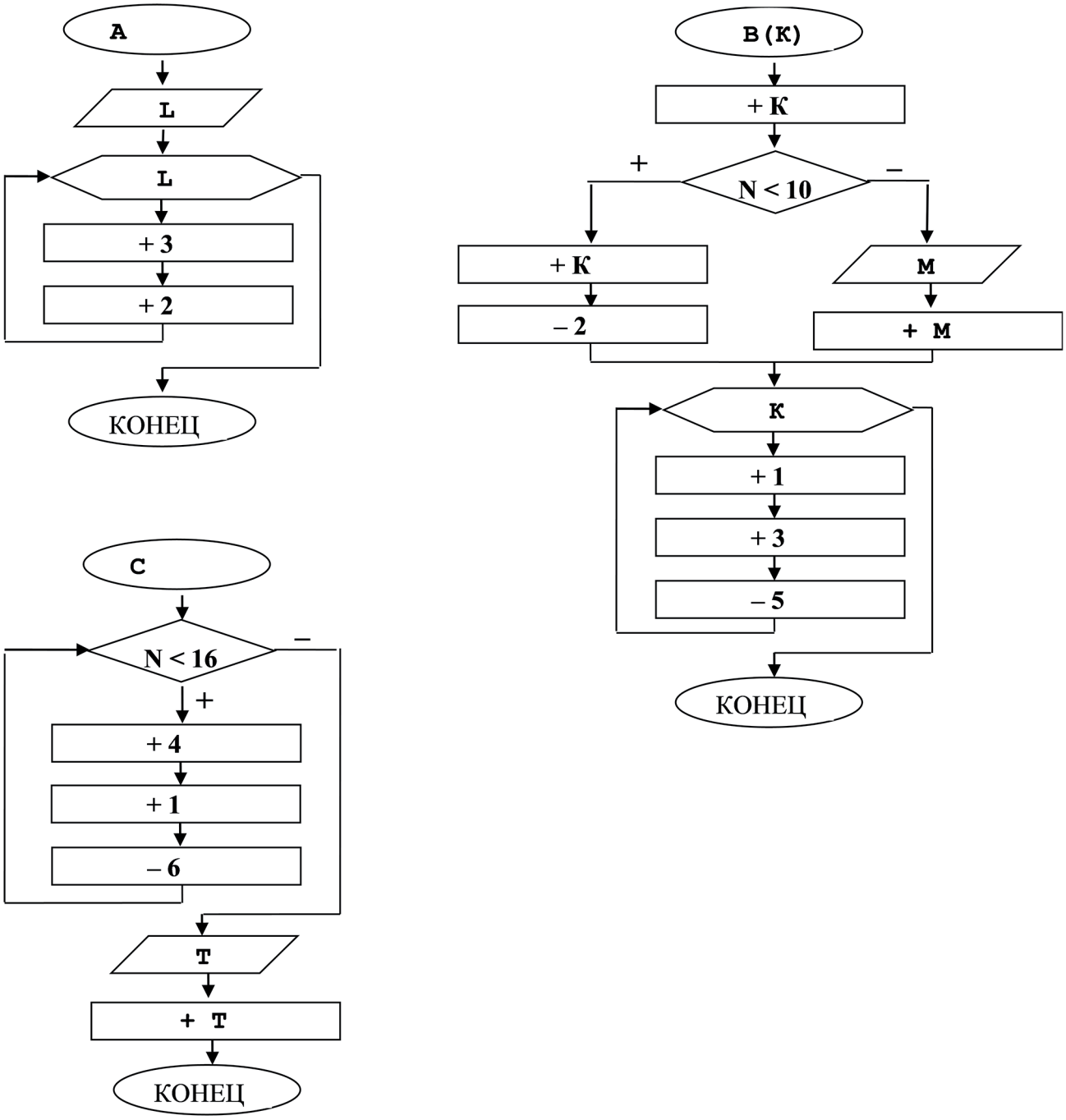

Рис. 6. Игровое поле игры «Вычислялка-2». Блок-схемы подпрограмм

Если подпрограмма параметризована, игроку имеет смысл записать значение параметра на свою карточку. (Подпрограммы вызываются по нескольку раз с разными значениями параметров.)

Последний вид блока, который встречается на блок-схеме, это блок ввода данных (блок-параллелограмм). В параллелограмме указано имя переменной, значение которой надо ввести. Для этого игрок бросает кубик. Выпавшее значение становится значением этой переменной и используется в следующем блоке. Имя и значение переменной игрок записывает на свою карточку. Переменная может использоваться для вычисления нового значения переменной N в блоке-действии, для сравнения в блоке-условии, для определения количества повторов цикла ПОВТОРИТЬ и для передачи в подпрограмму в качестве параметра. Обратим внимание, что ввести значение переменной надо даже в том случае, если фишка просто прошла через блок ввода данных, не остановившись на нем. Дело в том, что введенное значение будет задействовано в следующих блоках.

На рисунках 5 и 6 приведено игровое поле игры «Вычислялка-2» (на рисунке 5 — блок-схема главной программы, на рисунке 6 — блок-схемы подпрограмм).

4. Методические замечания

4.1. Разбивка материала по урокам

Всего на тему «Представление алгоритмов с помощью блок-схем» планируется четыре часа:

- Первый урок — «Знакомство с видами алгоритмов, ветвлениями и циклами. Игра “Бродилка-1”». О зацикливании речь пока не идет.
- Второй урок — «Знакомство с конструкциями ввода, подпрограммами и параметрами. Продолжение работы с управляющими конструкциями. Игра “Бродилка-2”». (Пока тоже без зацикливания.)
- Третий урок — «Понятие зацикливания на примере подпрограммы В из игры “Бродилка-2”. Закрепление материала. Игры “Вычислялка-1”, “Вычислялка-2”».
- Четвертый урок — «Повторение. Проверочная работа».

Начинать надо с той игры, которая проще. К сожалению, универсального способа оценить сложность игр не существует. Выше предложено начинать с «Бродилки». Но автор встречался с ситуацией, когда «Вычислялка» оказывалась для аудитории проще.

4.2. Использование «рукописных» игровых полей

Автор проводил занятия с детьми на игровых полях, нарисованных от руки. Специально для данной публикации (для лучшего отображения материала при типографской печати) были подготовлены «машинописные» игровые поля, выполненные средствами компьютерной графики. Автор настоятельно рекомендует использовать для занятий с детьми поля «рукописные» (рис. 7). Причина — та же, по которой рукописные конспекты лекций предпочтительней машинописных. «Рукописные» поля — «живые». С ними интересно. Каждая блок-схема, каждый блок уникальны, неповторимы. Это позволяет при освоении информации активно задействовать правое полушарие головного мозга (отвечающее за конкретно-образное мышление). «Рукописные» блок-схемы «хорошо видно». При использовании «рукописных» полей эффективность работы возрастает.

«Машинописные» блок-схемы, разумеется, много красивее. Но они — «мертвые», неинтересные. Все они выглядят одинаково. От однообразных изображений попросту рябит в глазах. Воспринимать их приходится левым — абстрактно-логическим — полушарием. Правое используется минимально. С такими блок-схемами дети играть не захотят.

В данном случае мы имеем дело с часто встречающимся противоречием: эффектность или эффективность? Что важнее: красивый вид или доходчивость учебного материала?

Оба комплекта игровых полей — «машинописный» и «рукописный» — можно скачать как дополнительный материал к данной статье на сайте ИНФО.

4.3. Логика исполнения алгоритма vs логика игры-ходилки

В правилах обеих игр возникает противоречие между «логикой игры-ходилки» и «логикой исполнения алгоритма». По логике игры-ходилки «срабатывать» должен только тот блок, на котором фишка остановилась. Блок, через который фишка прошла без остановки, «срабатывать» не должен. Это значит, например, что если фишка не остановилась на блоке вызова подпрограммы, а просто прошла через этот блок, то и заходить в эту подпрограмму она не должна. С точки зрения логики исполнения алгоритма деление продвижения фишки по блок-схеме на ходы — вещь совершенно условная. Поэтому «срабатывать» должен каждый блок, через который проходит фишка. В описанных выше правилах игры вход в подпрограмму выполняется всегда. То есть в этом случае применяется логика исполнения алгоритма. Аналогично всегда выполняется ввод данных при прохождении фишки через блок ввода (введенное значение может потребоваться далее). В то же время в «Вычислялке» предлагается выполнять не все вычислительные блоки (действия с переменной N), а только те, на которых фишка остановится. В этом случае применяется логика игры-ходилки.

4.4. Пропедевтика трассировки переменных

Запись последовательности значений переменной N в игре «Вычислялка», по сути, является трассировкой. Впоследствии ее можно будет вспомнить при изучении трассировочных таблиц.

4.5. Сравнение циклов ПОКА и ПОВТОРИТЬ

В игре «Бродилка-1» дается возможность сравнить циклы с предусловием и с заданным числом повторений. В первой развилке (блок 2) на ветви «плюс» располагается цикл ПОВТОРИТЬ (блоки 3–6), а на ветви «минус» — цикл ПОКА (блоки 7–10). У обоих циклов тело имеет одинаковую длину. В первом цикле количество повторений задано точно — 4. Во втором стоит условие «Пока < 5» (т. е. для ответа «Да» на кубике могут выпасть

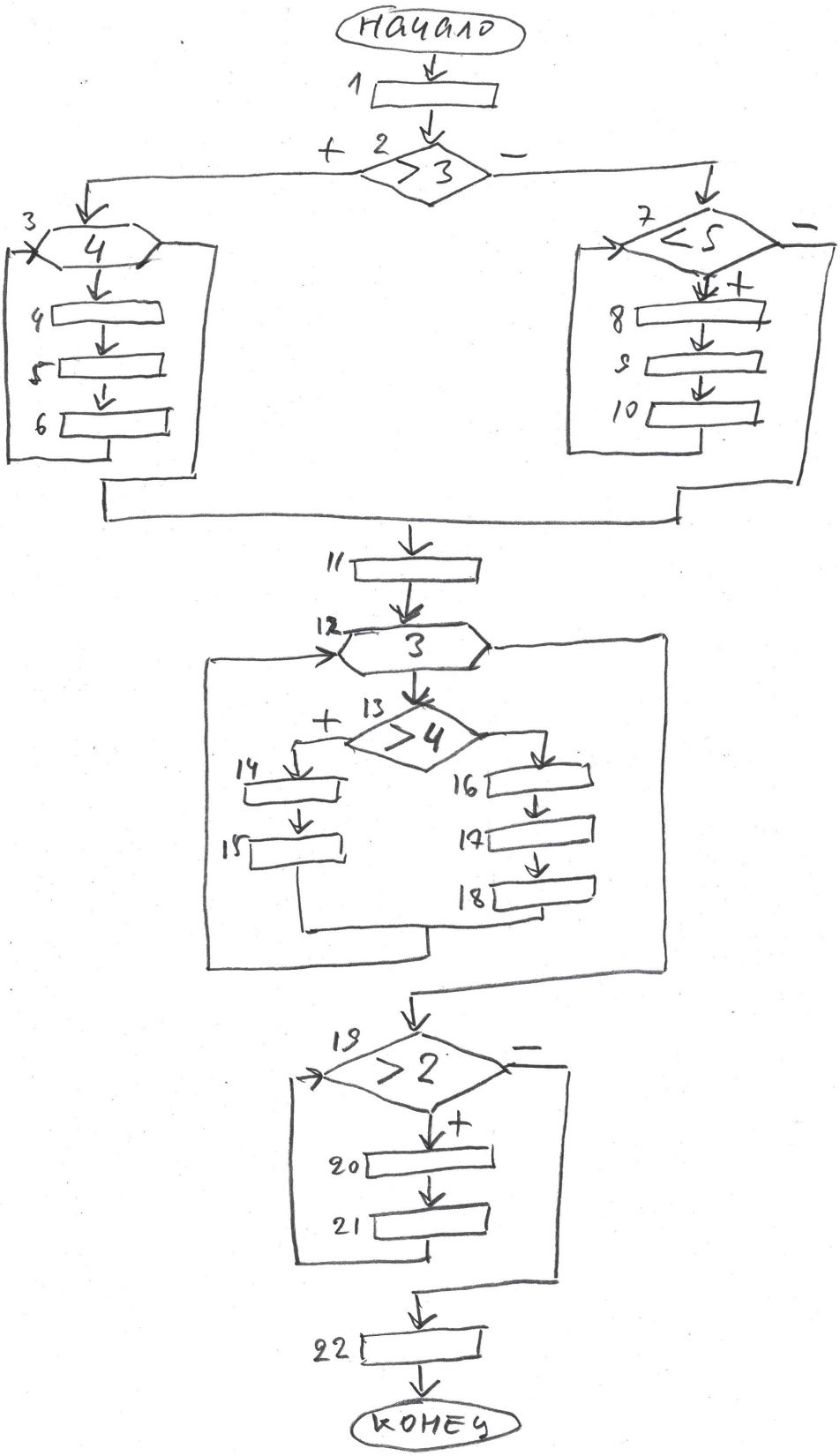

Рис. 7. Игровое поле (блок-схема) игры «Бродилка-1»

числа от 1 до 4). Можно проследить отличие в исполнении этих циклов.

4.6. Зацикливание

В игре «Бродилка-2» в подпрограмме B нарочно допущена возможность зацикливания. Заголовок цикла имеет вид «ПОКА значение на кубике ≤ L», где значение переменной L определено предыдущим броском кубика. Если L выпадает равным 6, выйти из цикла невозможно. Хорошо, если такая ситуация реально возникнет в игре. На нее надо обратить внимание и ввести понятие «зацикливание». Если ситуация не возникнет «естественным» путем, следует специально обратить внимание на ее возможность.

Вообще говоря, понятие «зацикливание» в данный момент выглядит некоторым излишеством. Нет никакой необходимости изучать его именно сейчас. Вполне можно отложить знакомство на потом.

4.7. Отсутствие вложенных подпрограмм

Для простоты в описанных играх отсутствуют вложенные вызовы подпрограмм. Подпрограммы вызываются только из главной программы. Обучение манипуляциям со стеком не входит в задачу данных игр.

4.8. Варьирование правил игры

Предлагаемые правила игр являются результатом нескольких проб, проведенных как на ученической, так и на учительской аудитории. Автор считает полезным поделиться описанием опробованных вариантов. Обсуждались следующие идеи.

Не считать за шаги блоки-ветвления и заголовки циклов, считать только блоки-действия. Опыт показал, что необходимость разделять блоки на те, которые надо считать, и на те, которые считать не надо, излишне усложняет игру. Проще считать все блоки подряд.

Был вопрос: надо ли продолжать движение после входа в подпрограмму, если на данном ходе еще остались неизрасходованные шаги, или движение в подпрограмме надо начинать со следующего броска кубика? Вариант «продолжать движение» проще.

Значительные сложности были связаны с прохождением условий в игре «Бродилка». Один из операндов указан в блоке-условии. Откуда взять второй, который должен принести в игру случайность? Кроме варианта, описанного выше, рассматривались следующие:

1) Игрок бросает сразу два кубика: «кубик-шагалку» и «кубик-решалку». Кубик-шагалка определяет количество шагов, на которые должна быть передвинута фишка. Кубик-решалка задает число, которое будет подставлено в блок-условие.
2) Игра ведется одним кубиком. Его первоначальный бросок определяет количество шагов. При попадании фишки на блок-условие игрок бросает кубик еще раз. Выпавшее число подставляется в условие. Этот вариант делает игру существенно сложнее.
3) Перед началом игры разыгрывается на кубике и записывается ряд случайных чисел. Далее они по очереди будут использоваться в блоках-условиях. При исчерпании ряда его можно либо продолжить, либо зациклить.

4.9. Необходимость специальных мер для контроля знаний

Слабым местом игровой методики является сложность контроля знаний. В тот момент, когда играет весь класс, контроль правильности действий каждого учащегося обеспечить физически невозможно. Может возникнуть ситуация, когда *все* игроки одной команды неправильно поняли какую-то конструкцию либо неправильно понял кто-то один, но этот один сумел убедить в своей правоте всех остальных. В результате окажется, что вся команда затвердила неправильное понимание. Для проверки правильности понимания необходимы домашние задания, причем легкопроверяемые. Лучше всего, если это будут блок-схемы алгоритмов черчения, которые надо выполнить (создать соответствующий чертеж). В этом случае правильность ответов можно оценить «с одного взгляда». В данный момент учащимся еще не введены строго (хотя уже упомянуты) понятия «исполнитель» и «система команд исполнителя» (СКИ). Поэтому углубляться в их строгое описание не стоит.

Самую первую апробацию описанные игры прошли в лицее № 10 города Перми (см. фото). Больше фотографий — в архиве дополнительных материалов к данной статье, который можно скачать на сайте ИНФО.

Фото. Игры-бродилки на уроке информатики в лицее № 10 города Перми

5. Заключение

В заключение несколько слов об оценке оригинальности описанных игр.

Сама по себе идея использования игровых моментов в изучении информатики ведет начало от Сеймура Пейперта с его черепахой Лого. Затем она была подхвачена и реализована в превеликом множестве ползающих, шагающих и прыгающих исполнителей. Вслед за многочисленными версиями Лого (см., например, [7]) были Таркан [4], Муравей [3], Кенгуренок [2]. Предназначались они для освоения алгоритмики. В настоящее время это направление развивается очень активно. Перечислять надо уже не отдельные игры и исполнителей, а обзоры таких игр и исполнителей (см., например, [1, 5, 6, 8, 11]).

Описанные в данной статье игры придуманы в 2013 году и с тех пор многократно использовались при работе как с учащимися, так и учителями информатики. Аналогов автор пока не встречал. Игры оригинальны, во-первых, по теме (освоение блок-схем), во-вторых, по направленности на «чтение алгоритмов», а не их написание, в-третьих, по «настольной реализации». Традиционно считается, что игры для обучения информатике должны иметь компьютерную реализацию. Исключения — редки. Признавая достоинства компьютерных игр, автор считает, что современные дети слишком много времени проводят в мире виртуальном. И любую возможность вернуть их в мир материальный и заставить поработать руками надо использовать.

Список источников

1. *Бордик В.* 15 бесплатных игр и сайтов, которые научат ребенка программировать // Летидор. https://letidor.ru/obrazovanie/15-besplatnykh-igr-i-saitov-kotorye-nauchat-rebenka-programmirovat.htm

2. *Вараксин Г. С.* Информатика в начальной школе. Кенгуренок (графический исполнитель). https://ikt-det.jimdofree.com/алгоритмика/исполнители/графический-исполнитель-кенгуренок/

3. *Гутман Г. М., Карпилова О. М.* Муравьиные сказки. М.: Просвещение, 1993. 98 с.

4. *Дуванов А. А., Первин Ю. А.* Необыкновенные приключения Пети Кука в Роботландии. М.: Финансы и статистика, 1997. 272 с.

5. Игры с алгоритмами // POLYCENT. https://polycent.ru/blog/igryi-s-algoritmami/

6. *Карпов Д.* 15 сайтов и игр, которые научат детей программированию. Почему и как стоит учить информатику с дошкольного возраста // Мел. https://mel.fm/ucheba/fakultativ/5297184-computer_science

7. ЛогоМиры 3.0. Интегрированная творческая среда для освоения азов программирования. https://www.softslot.com/software-2581-logomiry.html

8. *Мартынов А.* 10 игр, которые научат ребенка программированию // Лайфхакер. https://lifehacker.ru/10-igr-nauchat-programmirovat/

9. *Плаксин М. А.* ТРИЗформатика — метапредмет, объединяющий компьютерные и интеллектуальные технологии работы с информацией (ответ на вызов информационного общества) // Программирование. 2011. № 6. С. 26–32.

10. *Плаксин М. А., Иванова Н. Г., Русакова О. Л.* Информатика: учебник для 4 класса. М.: БИНОМ. Лаборатория знаний, 2012. 128 с.

11. ТОП-12 игр, с которыми ребенок научится программировать // Proglib. https://proglib.io/p/games-for-kids