М. Р. Закирова

Информатика и информационные технологии 4 класс

Учебник для средних общеобразовательных школ с русским языком обучения



Новое издание

Рекомендовано Научно-методическим советом при Центре развития образования

ПРОВЕРЕНО АНТИПЛАГИАТ ПРОВЕРЕНО

novda edutainment

Ташкент – 2026

УДК: 004(075.3)
ББК 32.81я72
3-18

Закирова М. Р.
Информатика и информационные технологии, 4 класс.
Учебник для средних общеобразовательных школ с русским языком обучения / М. Р. Закирова – Ташкент: «Novda Edutainment», 2026 – 128 с.

Рецензенты:

Н. М. Усманова – учитель начальных классов высшей категории СОШ № 190 г. Ташкента;

О. Ш. Евстафьева – учитель начальных классов высшей категории СОШ № 190 г. Ташкента.

Условные обозначения



Ответьте на вопросы



Логическое задание



Это интересно знать!



Работа за компьютером



Работа в парах



Домашнее задание



Конец урока

ISBN 978-9910-295-16-4

© М. Р. Закирова
© «Novda Edutainment», 2026

Дорогие ребята!

Меня зовут **Шустрик**, я – робот. Мы с вами уже сделали первые шаги в мир информатики, узнали о видах информации и способах её обработки. Теперь нас ждёт новое увлекательное путешествие!

В этом учебнике мы познакомимся с новыми технологиями и компьютерами, разберёмся с современными терминами и продолжим учиться составлять алгоритмы. Вы будете развивать логическое мышление, научитесь рассуждать, находить решения и систематизировать знания.

На страницах учебника вас ждут познавательные истории, интересные задания и множество открытий. Готовы отправиться в путешествие?

Тогда вперёд!



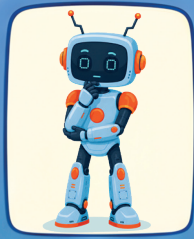
Важные обозначения



Запомни!

В тексте с таким обозначением я сообщу вам новые термины и понятия, которые следует запомнить.

В этом блоке я предложу ситуации и задачи, к которым нужно будет придумать объяснение или решение.



Подумай!



Осторожно!

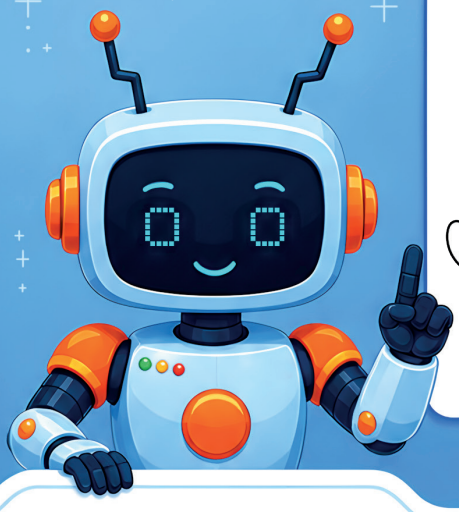
Так я буду предупреждать вас о необходимости соблюдать правила безопасности!

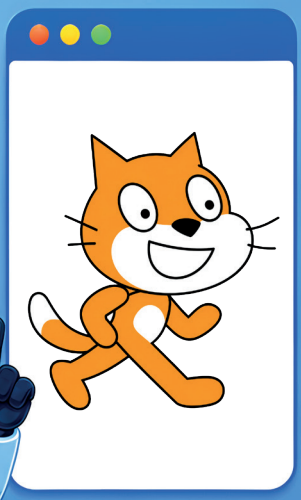
В этом блоке я расскажу вам интересные факты.



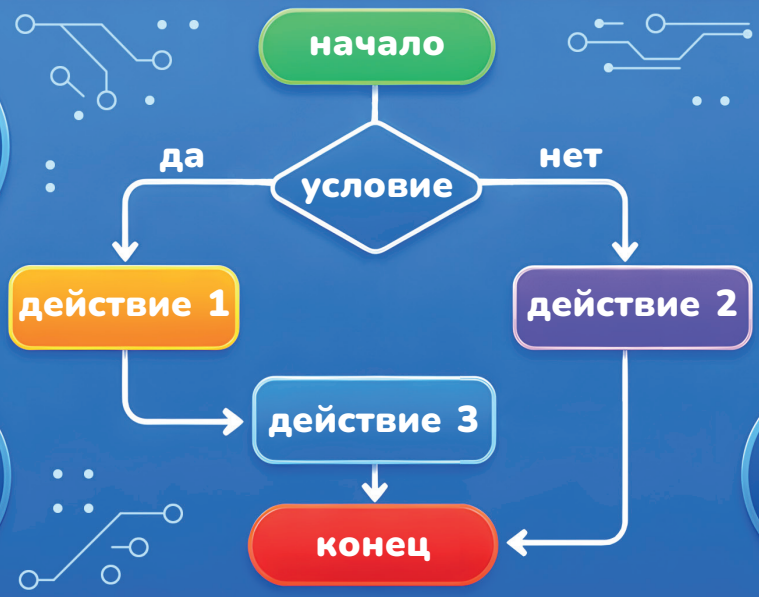
Это интересно!







Основы алгоритмов и программирования



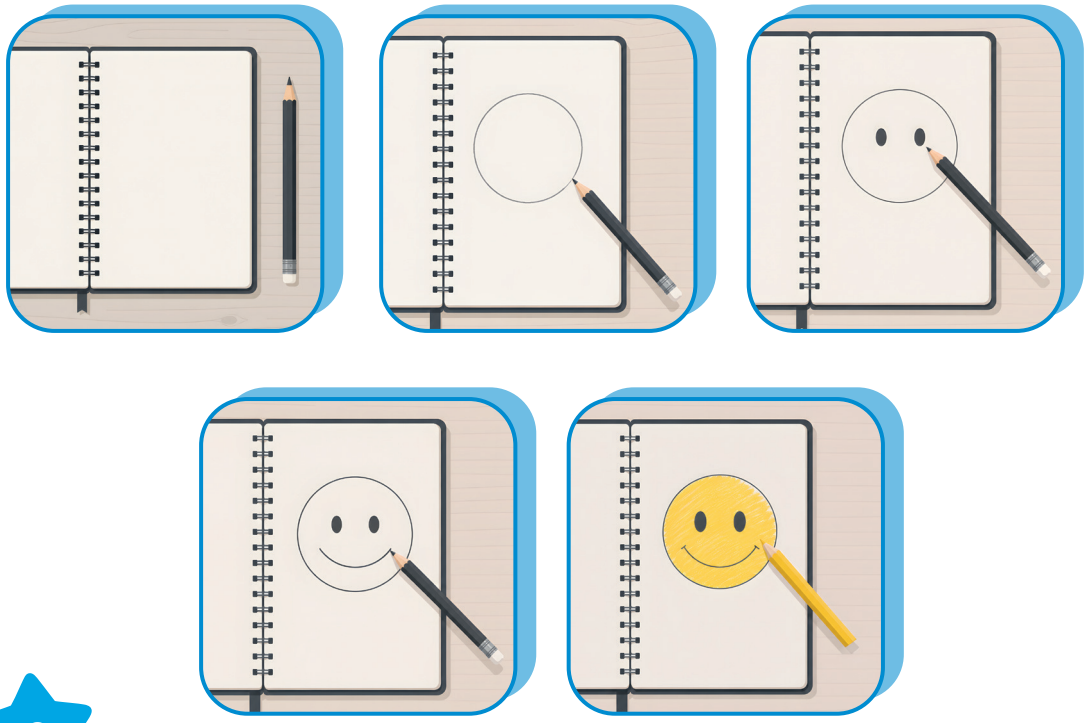


Алгоритмы

- Что вы знаете об алгоритмах?
- Можно ли любую последовательность действий назвать алгоритмом?
- Зачем нужны алгоритмы?

Каждый день человек решает разные задачи. Чтобы успеть собраться в школу, созвониться с другом, приготовить еду или выполнить задание, нужно соблюдать правильную последовательность действий.

1. Опишите последовательность шагов в рисовании смайлика:



Связанные друг с другом действия, которые выполняются по порядку и приводят к определённому результату, называются **последовательностью действий**.





2. Умид решил научить младшего брата завязывать шнурки. Он объяснил порядок действий по шагам.

а) Возьми шнурки в обе руки.

б) Перекрести шнурки.

в) Сделай петлю.

г) Оберни второй шнурок вокруг петли.

д) Протяни шнурок в отверстие.

е) Затяни бантик.



Брат Умида перепутал некоторые шаги местами, и бантик не получился.

Как вы думаете:

- Почему результат оказался неправильным?
- Нужно ли выполнять все действия по порядку?
- Можно ли пропускать шаги в последовательности действий?

Во многих делах очень важно соблюдать правильный порядок действий. Если поменять действия местами или пропустить какой-нибудь шаг, задача может быть решена неправильно.

Когда последовательность действий помогает выполнить задачу и получить нужный результат, она называется **алгоритмом**.

Каждое отдельное действие в алгоритме называется **шагом алгоритма**.





Значит, любая последовательность действий является алгоритмом?



Конечно же, нет! Чтобы последовательность стала алгоритмом, нужно выполнить ряд условий.

Чтобы последовательность действий считалась алгоритмом, она должна соответствовать следующим правилам:

- каждое действие должно быть описано точно и понятно;
- алгоритм должен состоять из определённого числа шагов;
- выполнение алгоритма должно приводить к заданному результату.

Алгоритмы могут быть представлены разными способами.

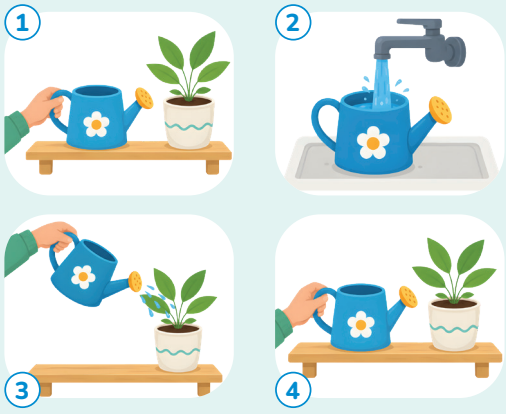
Представление алгоритма

- словесное описание
- с помощью рисунков
- с помощью блок-схемы

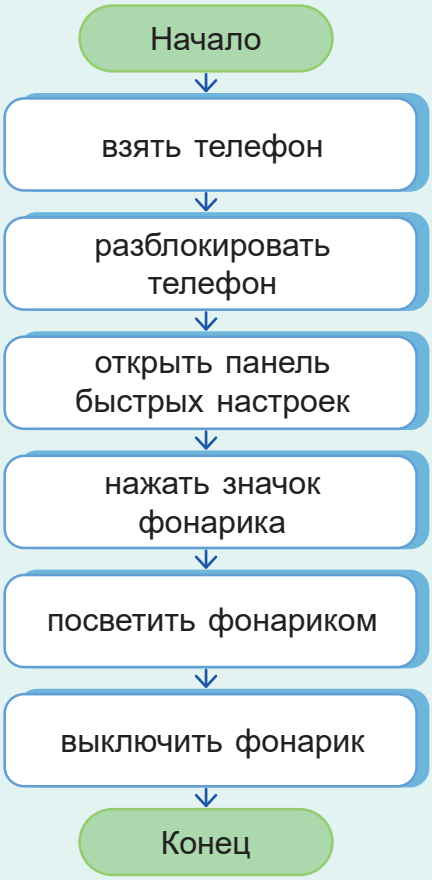
Словесный алгоритм

1. Взять телефон.
2. Открыть камеру.
3. Включить фронтальную камеру.
4. Улыбнуться.
5. Нажать кнопку съёмки.
6. Посмотреть фотографию.

Алгоритм, заданный рисунками

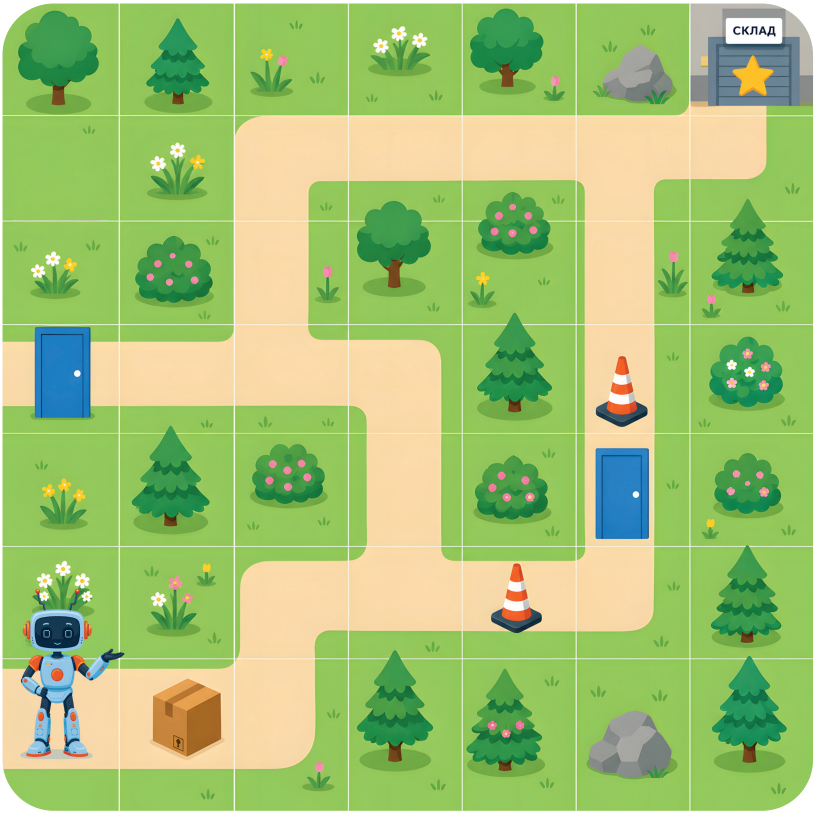


Алгоритм, заданный блок-схемой



3. Приведите примеры, когда удобнее использовать словесное описание алгоритма, а когда – графическое описание с помощью рисунков или блок-схем.

4. Помогите роботу доставить коробку на склад. Составьте алгоритм его движения.



Робот умеет выполнять команды:

- шаг вверх
- шаг вниз
- шаг влево
- шаг вправо
- взять коробку
- открыть дверь
- перепрыгнуть
- положить коробку

5. Составьте правильный алгоритм действий для решения задачи. Как получить 1 литр воды, используя только 3-литровую и 5-литровую ёмкости?



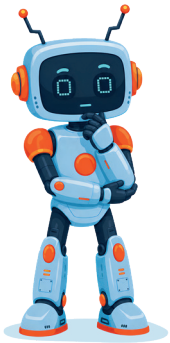
Слово «алгоритм» связано с именем известного учёного аль-Хорезми. Он жил в IX веке и изучал математику, способы вычислений. На латинском языке его имя произносили как «Алгоритми». Позже от этого имени появилось слово «алгоритм». Сегодня алгоритмы используются в компьютерах, телефонах и роботах.



6. Ответьте на вопросы:

- Какая последовательность действий будет являться **алгоритмом**?
- Что такое шаг алгоритма?
- Почему в алгоритме важен порядок шагов?
- Какие существуют способы записи алгоритмов?
- Приведите пример алгоритма из своей жизни.

Разделение задачи на шаги



Вспомните, как вы решали сложные задачи. Составляли ли вы план решения? Решали задачу по действиям или сразу записывали решение одним выражением?

Акмаль и Азиза собирают большой конструктор. В инструкции показано, что модель собирают не сразу, а по частям.



1. По рисунку расскажите, что они собрали сначала, а что – потом.

Когда люди решают сложную задачу, её удобно разделять на небольшие действия или части.

Это помогает:

- сделать задачу понятнее;
- выполнять разные части по очереди или одновременно;
- легче находить и исправлять ошибки;
- лучше планировать свои действия.

В информатике разделение сложной задачи на простые части называется **декомпозицией**.

Декомпозиция помогает выстроить правильную последовательность действий и составить строгий алгоритм.



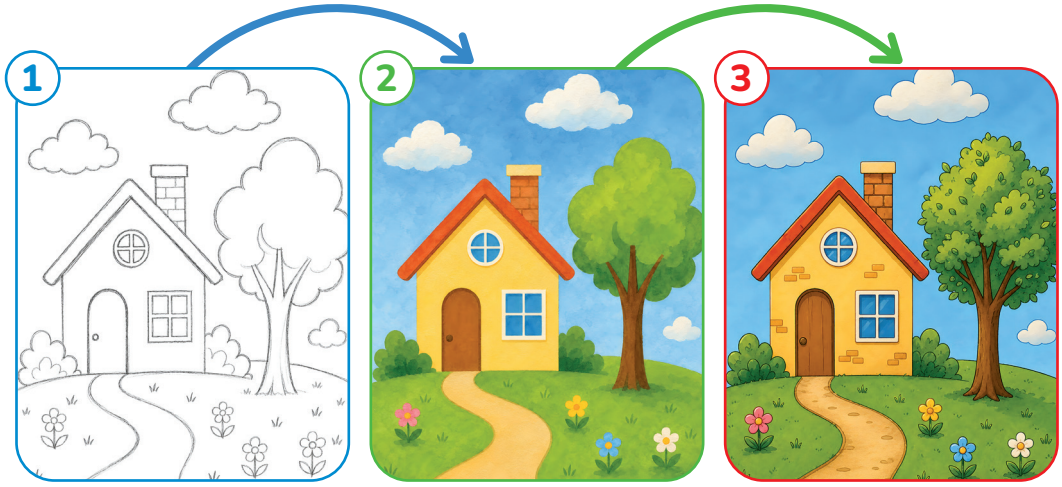


2. Представьте, что вашему классу нужно подготовить праздник. Сразу выполнить такую большую задачу сложно, поэтому её нужно разделить на маленькие действия.

- Обсудите, какие действия нужно выполнить, чтобы организовать праздник.
- Разделите большую задачу на более мелкие шаги.
- Запишите шаги в правильном порядке.

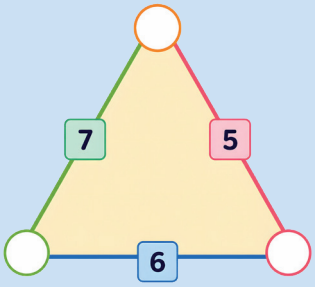


3. Приведите примеры, когда вы разделяете сложную задачу на простые части. Воспользуйтесь рисунком-подсказкой.



4. Подумайте, как разделить эту задачу на отдельные простые шаги.

Какие числа должны стоять в пустых кругах, чтобы число на каждой стороне равнялось сумме двух соседних с ним кругов?



Обсудите:

- Почему задачу удобнее решать по шагам?
- Можно ли решить её сразу?
- Что произойдёт, если ошибиться в одном шаге?

«Ханойская башня» – известная логическая задача. Её название связано с городом Ханоем во Вьетнаме. Существует легенда о монахах, которые перекладывают золотые диски между башнями по строгим правилам. По легенде, когда работа будет закончена, наступит конец света.
На самом деле эту задачу придумал французский математик Эдуард Люка в XIX веке. Эта головоломка показывает, как сложную задачу можно решить постепенно, шаг за шагом, с помощью алгоритма и декомпозиции.



5. Ответьте на вопросы:

- Что такое **декомпозиция**?
- Почему сложную задачу удобно делить на маленькие действия?
- Что может произойти, если не разбивать сложную задачу на шаги?
- Приведите свой пример декомпозиции (из жизни или учёбы).

Исполнение алгоритма



- Можно ли дать одну и ту же команду человеку, роботу и собаке и получить одинаковый результат?
- Почему даже после подробного объяснения человек может выполнить задание неправильно?



1. Проведите игру в паре «Рисуем по алгоритму».

Ученик А – автор алгоритма

- Придумайте простой рисунок.
- Не показывайте его партнёру.
- Запишите точные команды для рисования.





Ученик В – исполнитель

- Внимательно выполняйте команды партнёра.
- Не задавайте дополнительные вопросы.

Обсудите:

- Получился ли рисунок таким, как задумал автор?
- Какие команды были удачными, а какие – не очень?
- Почему исполнителю так важны точные команды?
- Может ли исполнитель выполнить действие по-своему?



Исполнитель – это тот, кто умеет выполнять команды. Исполнителями могут быть люди, животные, компьютеры, машины и роботы. Каждый исполнитель понимает только определённые команды.

Алгоритм – это точная последовательность действий для определённого исполнителя. Один и тот же алгоритм может быть понятен одному исполнителю, но непонятен другому.



Собака понимает команды сидеть, ко мне.



Машинка на пульте управления понимает только команды: вперёд, назад, влево, вправо.

2. Подумайте, кто на рисунке является исполнителем команд. Какие команды понимает каждый из этих исполнителей?



Набор команд, которые понимает исполнитель, называется **системой команд исполнителя**. Человек понимает слова и жесты, компьютер – команды программ.

Чтобы исполнитель правильно выполнил алгоритм, команды должны быть:

- понятными исполнителю;
- точными;
- заданными в правильном порядке.

Если команда будет неточной или непонятной, исполнитель не сможет правильно выполнить задание.



Значит, для каждого исполнителя нужно создавать свой алгоритм, который он может исполнить?



Именно! Исполнитель понимает только те команды, которые входят в его систему команд.

3. Придумайте алгоритм рисования квадрата для:
а) человека, который понимает любые слова и жесты;
б) компьютера, который понимает команды «начни рисовать», «вверх», «вниз», «влево», «вправо», «закончи рисовать».

В современном мире человеку помогают роботы и компьютеры. Робот – это машина, которая выполняет команды человека. Для робота, как и для компьютера, очень важно правильно составить алгоритм и программу.

4. Изучите алгоритм для робота. Попадёт ли арбуз в ящик в результате этого алгоритма?





5. Ответьте на вопросы:

- Кто или что может быть **исполнителем** алгоритма? Приведите примеры.
- Что называется системой команд исполнителя?
- Почему команды должны быть точными и понятными исполнителю? Приведите пример, когда неточная команда может привести к неверному результату.



Циклические алгоритмы

В нашей жизни многие действия повторяются снова и снова. Приведите примеры таких действий.

Каждое утро Солнце встаёт – наступает день. Вечером Солнце садится – наступает ночь. Эти действия повторяются снова и снова.

Как происходит смена дня и ночи



Каждое **утро** Солнце встаёт – наступает **день**.

Вечером Солнце садится – наступает **ночь**.

Повторения можно встретить не только в природе, но и в действиях человека.

Например, при ходьбе человек выполняет повторяющиеся действия: шаг правой ногой, затем – шаг левой ногой, потом – снова шаг правой ногой. Такие повторы встречаются и в повседневной жизни: при умывании, приёме пищи, работе.

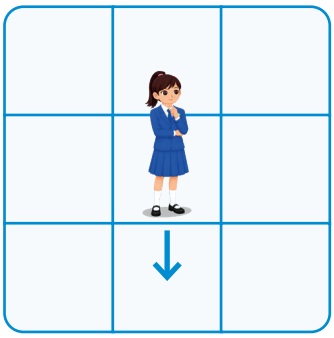
Иногда в алгоритмах некоторые действия повторяются несколько раз. Последовательность таких повторяющихся действий называют **циклом**. А алгоритмы, содержащие цикл, называют **циклическими алгоритмами**.



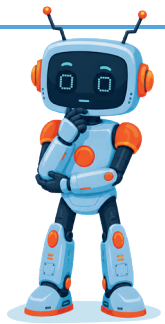
Составляя план действий, нам часто приходится указывать, что некоторые действия нужно выполнить несколько раз. Вместо того, чтобы много раз записывать одинаковые команды, можно использовать **повторение**.

1. Выполните алгоритм:

1. Встаньте прямо.
2. Повторите 4 раза:
   - сделайте шаг вперёд;
   - повернитесь направо.
3. Остановитесь.



- Где вы оказались в конце этого алгоритма?
- Что произойдёт, если в цикле поменять шаги местами: сначала «повернитесь направо», потом «сделайте шаг вперёд»?



Использование повторения в алгоритме помогает:

- сократить его запись;
- сделать алгоритм понятнее;
- не повторять одинаковые команды много раз.

2. Выберите 3–4 упражнения, укажите, сколько раз необходимо повторять каждое действие. А затем составьте свой алгоритм выполнения утренней зарядки.

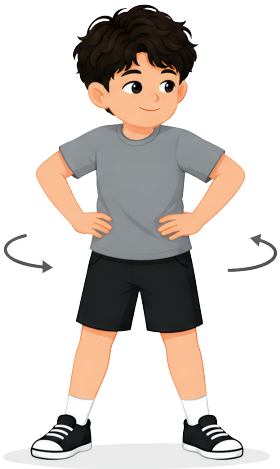
Упражнения для утренней зарядки



наклоны вперёд



приседания



повороты туловища



выпады



отжимания



наклоны в стороны

Циклические алгоритмы используются в компьютерных программах, чтобы многократно повторить одно и то же действие.

Алгоритм рисования узора

1. Начало.
2. Установи синий цвет кисти.
3. Переместись к верхнему краю экрана.
4. Повтори 8 раз
   - нарисуй квадрат со стороной 1 см;
   - установи красный цвет кисти;
   - нарисуй красный ромб внутри квадрата;
   - установи синий цвет кисти;
   - нарисуй вниз от синего квадрата линию длиной 5 мм;
   - переместись в конец синей линии.
5. Конец.





3. Придумайте свой циклический алгоритм для рисования узора.

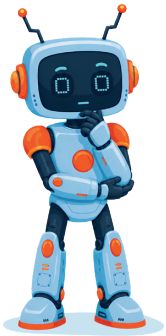


4. Ответьте на вопросы:

- Что называется **циклом**?
- Какие алгоритмы называют циклическими?
- Почему **повторение** помогает упростить алгоритм?
- Где в жизни можно встретить повторяющиеся действия?



Отладка алгоритма



- Бывает ли так, что ты выполняешь инструкцию, а результат получается совсем не таким, как ожидалось?
- Что обычно мешает получить правильный результат?

Составляя алгоритм, важно убедиться, что он работает верно. Для этого разработчики алгоритмов сами последовательно выполняют все действия алгоритма.

Обсудите.

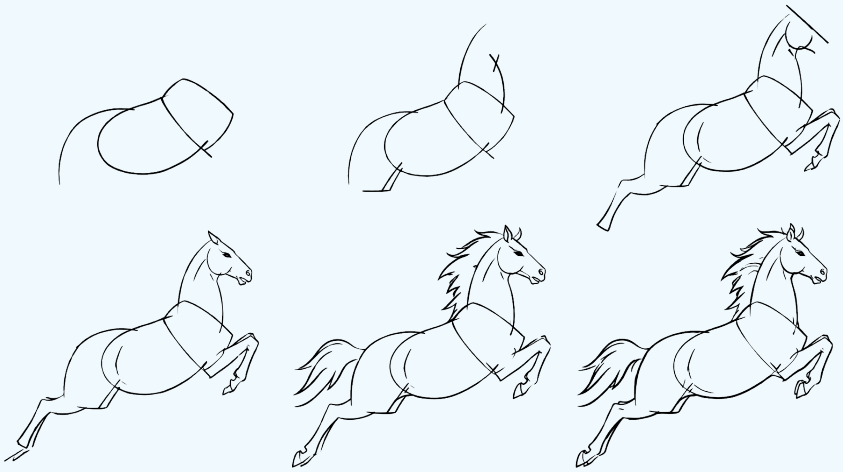
Получится ли отправить голосовое сообщение по этой инструкции? Почему?



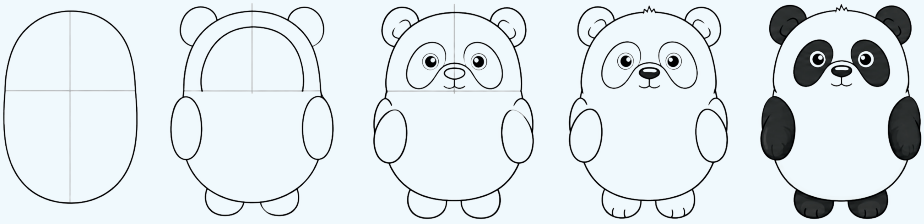


1. Выполните алгоритмы по предложенным шагам. Получился ли у вас такой же результат?

1



2



Иногда алгоритм работает неправильно. Это может происходить, если нарушен порядок действий, перепутаны команды или пропущен важный шаг.

Алгоритм также может работать неправильно, если действия записаны неточно.



Чтобы алгоритм выполнялся правильно, его нужно проверить и исправить ошибки. Такой процесс называется **отладкой алгоритма**.

Даже маленькая ошибка может привести к неправильному результату. Лучше найти ошибку заранее, чем потом переделывать всю работу.

2. Найдите все ошибки и соберите правильный алгоритм приготовления попкорна.

1


2


3


4


5


6


7


8


9




3. Проведите исследование.

- Придумайте и запишите небольшой алгоритм.
- «Сломайте» его – добавьте в него ошибки.
- Поменяйтесь «сломанными» алгоритмами с одноклассником.
- Попробуйте выполнить его алгоритм.
- Проведите отладку: найдите все ошибки, исправьте их и запишите правильную версию алгоритма.
- Обсудите результаты: какие ошибки было легче всего найти, а какие – сложнее.



Даже опытные программисты иногда допускают ошибки. Поэтому программы много раз проверяют и тестируют перед использованием.

Ошибки в программах называют «багами». Слово «баг» в переводе с английского означает «жук». По одной из легенд, в старый компьютер однажды действительно залетел маленький мотылёк и помешал его работе.

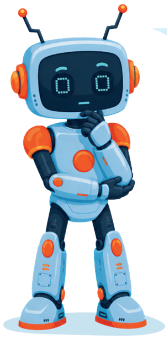


4. Ответьте на вопросы.

- Что такое **отладка алгоритма**?
- Какие ошибки могут встречаться в алгоритмах?
- Почему важно соблюдать порядок действий?
- Как можно проверить алгоритм?



Среда программирования Scratch



Человека окружает множество автоматизированных устройств: компьютеры, роботы-пылесосы, стиральные машины, сотовые телефоны и другие устройства. Как эти устройства понимают команды и инструкции?

Внутри каждого из автоматизированных устройств находится маленький компьютер, который умеет очень точно и быстро исполнять алгоритмы.

Алгоритм, записанный на понятном компьютеру языке, называют **программой**.

Специальный язык, который понимает компьютер, называют **языком программирования**.



Программы для компьютеров создают люди, которые знают языки программирования. Таких людей называют **программистами**.

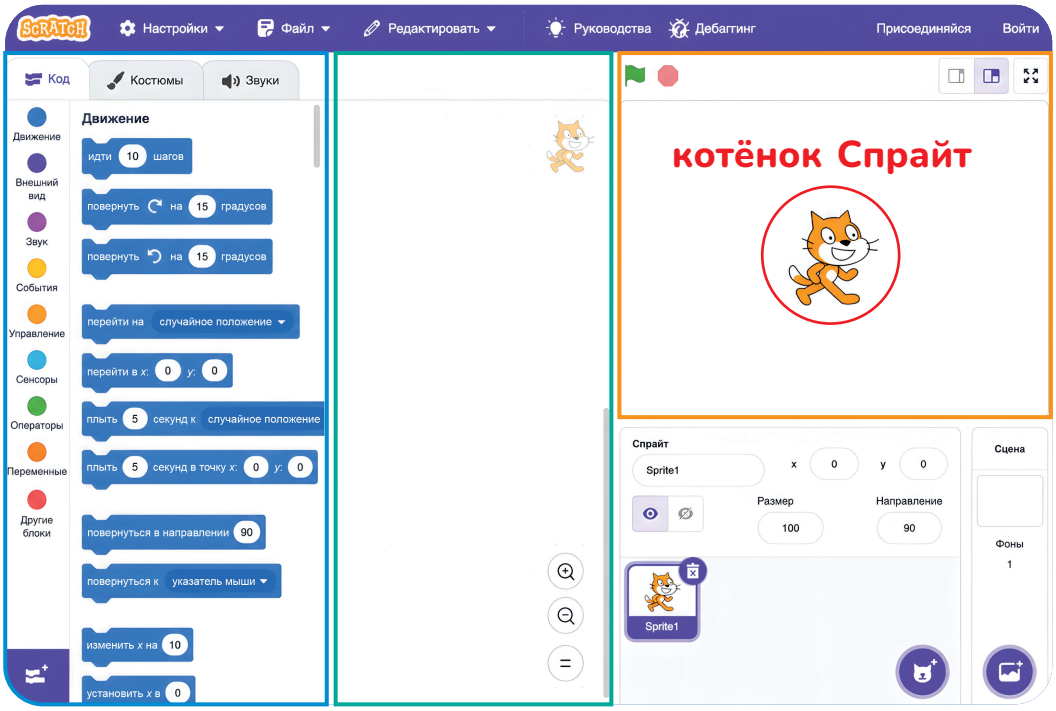
Scratch – это один из языков программирования. В **Scratch** не нужно писать команды вручную. Программа собирается из разноцветных блоков, как пазл или конструктор.

Чтобы написать программу на языке **Scratch**, нужно открыть среду программирования **Scratch**.

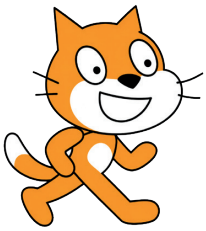
Окно программы **Scratch** разделено на три поля:

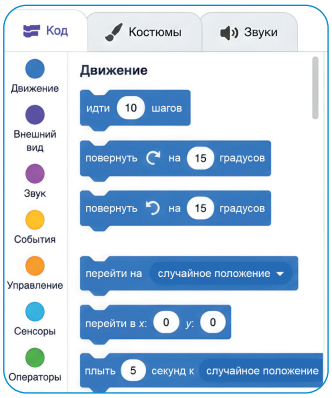
- Команды;
- Рабочая область;
- Сцена.

Команды **Рабочая область** **Сцена**



Один из персонажей среды **Scratch** – котёнок по имени **Спрайт**. Персонажи не умеют действовать самостоятельно. Они выполняют только те команды, которые мы им задаём.





блок Команды

Все команды, которые понимают персонажи, находятся слева в поле **Команды**.

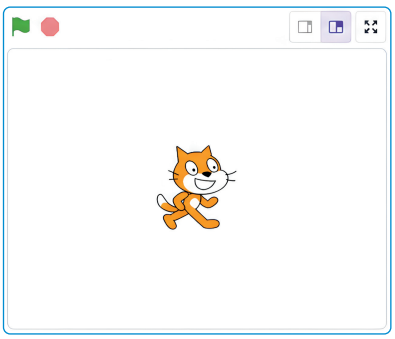
Команды собраны по цветам:
синие – команды движения;
фиолетовые – команды внешности;
розовые – команды звука;
жёлтые – команды событий.

Из этих команд мы будем составлять алгоритмы для персонажей. Для этого в **Рабочую область** нужно перетащить блоки команд и расставить их в правильном порядке.

Алгоритмы, записанные в рабочей области, называются **скриптами**.



Скрипт



Сцена

Сцена – это место, где будут происходить действия, которые вы пропишете своему герою. В самом начале на сцене находится котёнок Спрайт.

Чтобы запустить программу, в окне **Сцены** находятся специальные кнопки:

– запуск скрипта; – остановка скрипта.

Спрайт – персонаж, который выполняет команды.
Скрипт – программа, собранная из блоков для одного спрайта.
Сцена – фон, на котором всё происходит.



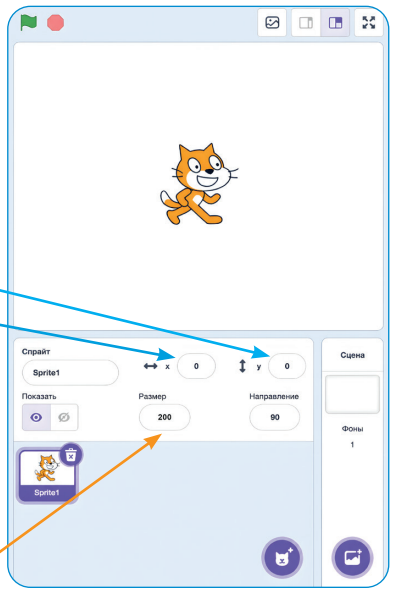
1. Откройте программу **Scratch**.

- Переименуйте персонажа. Для этого найдите под сценой панель с настройками, щёлкните по его имени мышкой и напишите **Кот**.
- Укажите местоположение персонажа ($x = 0$; $y = 0$), его размер (200).

Где оказался персонаж? Как он изменился?

Местоположение персонажа задаётся двумя координатами: значением по горизонтали (ось x), значением по вертикали (ось y). Центр сцены имеет координаты (0;0).

Размер персонажа задаётся в экранных точках – **пикселях**.



Чтобы заставить объект двигаться, нужно указать событие, при котором это происходит.

Первое событие происходит при запуске скрипта. Поэтому, чтобы объект начал двигаться с самого начала, нужно перетащить из блока **Команды** в **Рабочую область** событие **когда**  **нажат**. Чтобы переместить команду, нужно нажать на неё левой кнопкой мыши и, удерживая кнопку, перетащить команду в **Рабочую область**.



Как я смогу понять, в каком списке находится нужная мне команда?



Тебе поможет цвет команды. Если она связана с каким-то событием, команда будет **жёлтого** цвета. Если она связана с движением, цвет команды будет **синим**.



2. Перетащите в **Рабочую область** команды:

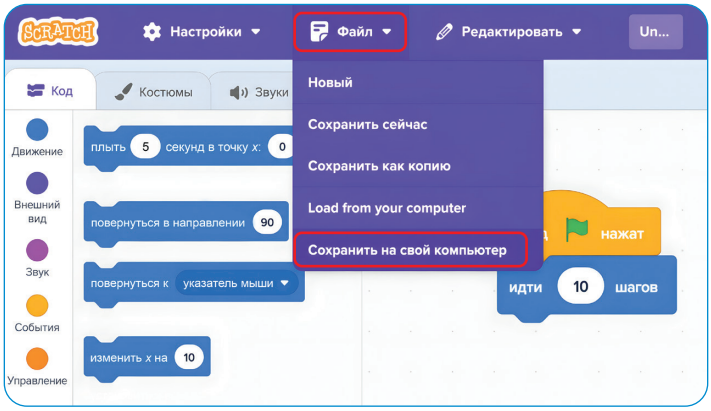
- когда  нажат;
- идти (10) шагов.



Запустите скрипт. Что произошло?

Количество шагов во второй команде можно изменить. Для этого наведите указатель мыши на число 10. При этом указатель изменится со значка руки на вертикальную черту. Затем щёлкните левой кнопкой мыши и с помощью клавиатуры измените количество шагов.

Чтобы скрипт остался в памяти компьютера, его надо сохранить. Найдите в меню кнопку **Файл**, нажмите на неё левой кнопкой мыши и в появившемся списке выберите **Сохранить на свой компьютер**.



3. Создайте в программе **Scratch** скрипт, как на рисунке.



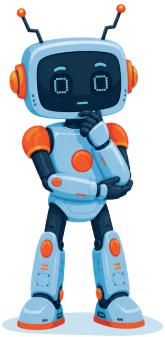
- Что произойдёт после запуска скрипта?
- Придумайте персонажу новое имя и переименуйте его.
- Измените алгоритм так, чтобы персонаж поворачивался на другой угол.

4. Составьте скрипт для персонажа, при котором он будет двигаться по квадрату. Сохраните ваш скрипт под названием «Поворот».

5. Ответьте на вопросы.

- Что такое **Scratch**?
- Кто такой **спрайт**?
- Что называется **скриптом**?
- Для чего нужна **Сцена**?
- Какие группы команд есть в **Scratch**?

Спрайты и управление объектами



- В играх и мультфильмах обычно действует не один герой, а сразу несколько персонажей.
- Придумайте, как персонажи могут взаимодействовать друг с другом.

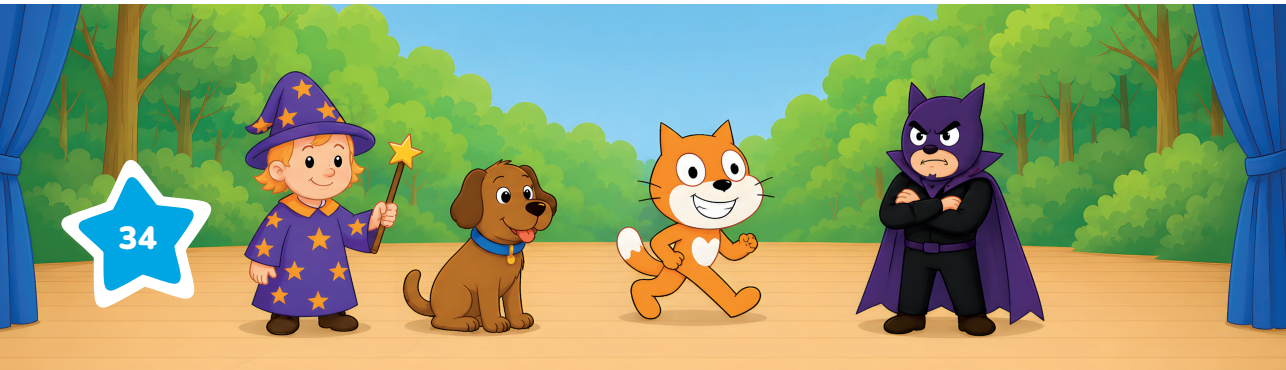
Представьте настоящий театр. На сцене выступают разные актёры: главный герой, его друзья и помощники, злодеи, животные, волшебники. Каждый актёр играет свою роль и взаимодействует с другими персонажами.

Такую театральную постановку в **Scratch** называют проектом – в нём собраны все персонажи, их скрипты и сцена. Проект можно сохранить и открыть снова, чтобы продолжить работу.

Спрайт – это любой объект (персонаж), который может двигаться, говорить, танцевать и выполнять команды. Кроме котёнка Спрайта, в Scratch есть много других персонажей. И все эти персонажи могут находиться на одной сцене одновременно.

Каждый спрайт:

- живёт своей жизнью;
- имеет свой собственный набор скриптов;
- может взаимодействовать с другими спрайтами.

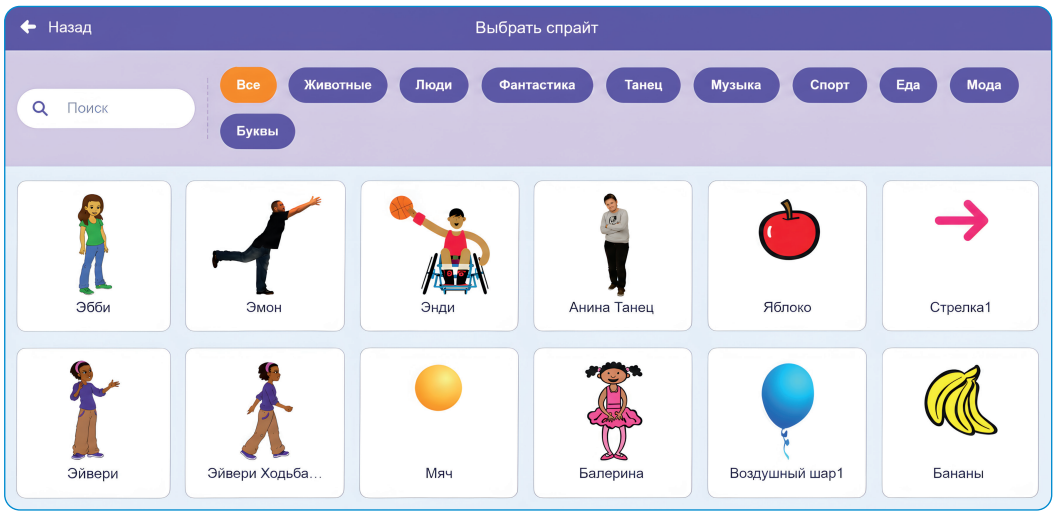


Как добавить новый спрайт

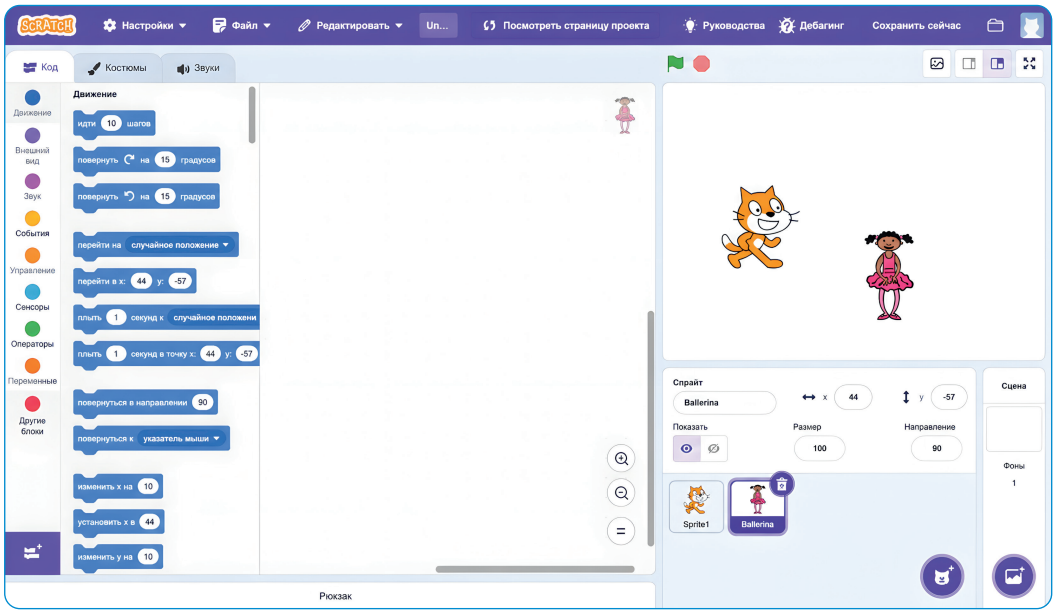
Для добавления нового спрайта на **Сцену** нужно внизу экрана нажать на кнопку **«Выбрать спрайт»**.



Затем откроется библиотека спрайтов, в которой нужно выбрать новый спрайт.



И новый герой добавляется к вам на **Сцену**.



Спрайт также добавляется в списке снизу.

Скрипты для каждого спрайта пишутся отдельно.

Чтобы написать скрипт для нужного персонажа, сначала щёлкните по нему в списке внизу экрана.

Если спрайты – это актёры, то Сцена – это место, где происходят все события. Героев мы уже научились добавлять и задавать им набор инструкций.

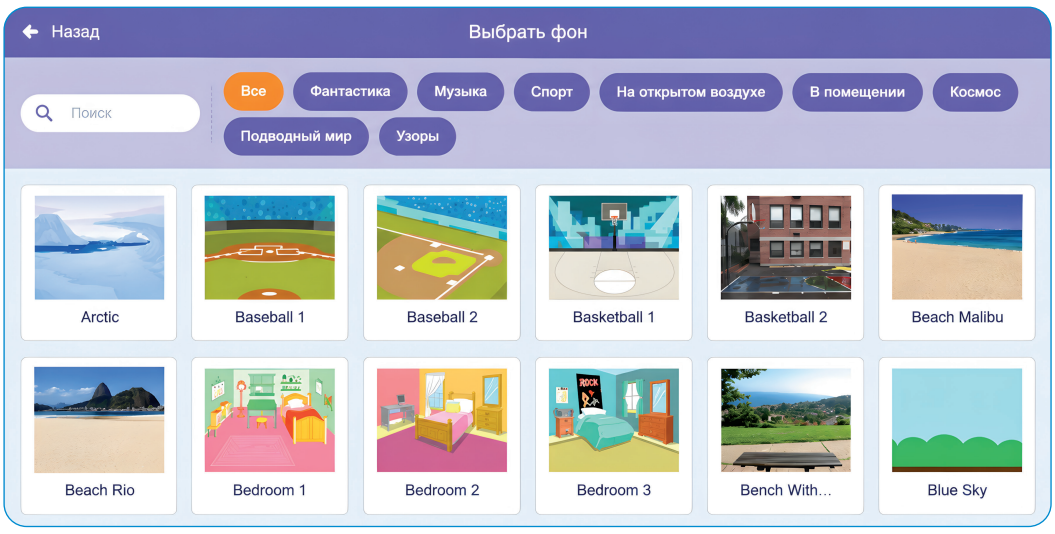
Сцену можно сравнить с театральными декорациями. Она может менять свой вид с помощью разных фонов.

В **Scratch** мы также можем поменять Сцену:

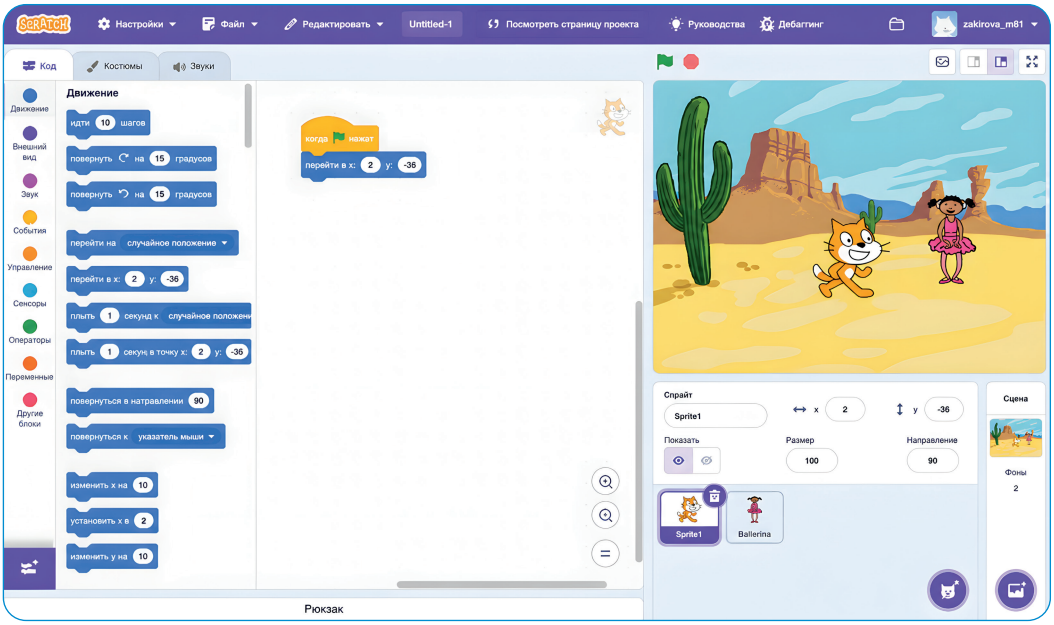
- можно менять фон (комната, лес, космос, школа, море и т. д.);
- на Сцене может быть несколько фонов – их можно переключать во время программы;
- Сцена также может выполнять команды (менять фон, играть музыку).

Как работать со Сценой

Для изменения фона сцены нужно внизу экрана справа нажать на иконку «**Сцена**». Затем откроется окно с библиотекой фонов, в которой нужно выбрать фон.



И ваша **Сцена** меняет вид.



В одном проекте Scratch можно добавить до 1000 спрайтов, но лучше использовать не больше 10–15, чтобы программа работала быстро.

Ответьте на вопросы.

- Сколько **спрайтов** может быть на одной сцене?
- Где располагаются **скрипты** разных спрайтов?
- Как добавить новый спрайт?

Урок 8. Контрольная работа № 1

Урок 9. Проект «Создание мультфильма в Scratch»

Мы узнали:

Алгоритм – это точная последовательность шагов, которая приводит к нужному результату.

Шаг алгоритма – каждое отдельное действие в алгоритме.

Декомпозиция – это разделение сложной задачи на более простые части. Это помогает выстроить правильную последовательность и составить алгоритм.

Исполнитель – это тот, кто выполняет алгоритм (человек, робот, компьютер, спрайт). У каждого исполнителя своя система команд, поэтому команды должны быть точными и понятными.

Цикл – повторение одних и тех же действий несколько раз.

Отладка – поиск и исправление ошибок в алгоритме.

Scratch – блочная среда программирования, где программы собираются из цветных блоков, как конструктор.

Спрайт – персонаж в Scratch, который выполняет команды.