1. Что такое алгоритм

Теория:

Повседневно человек выполняет большое количество самых разнообразных задач. Мы об этом даже не задумываемся, потому что некоторые задачи для нас стали автоматизированными действиями. *Например, «почистить зубы», «перейти дорогу», «собрать портфель»  и т.д.*

Иные задачи, наоборот, бывают настолько трудными, что требуют от нас длительных размышлений и немалых усилий для получения результата. Например, задача *«выучить английский язык»* требует от нас большое количество сложных действий, чем решение задачи *«купить воды».*

Любую, даже самую простую задачу мы всегда решаем за несколько последовательных шагов.

Рассмотрим задачу *«вскипяти чайник»* как последовательность действий:

1. взять чайник;
2. открыть крышку;
3. налить воды;
4. закрыть крышку;
5. включить плиту;
6. поставить чайник на плиту;
7. дождаться, пока чайник закипит;
8. выключить плиту.

Таким образом можно описать процесс решения любой задачи. Например, задачи, которые ты решаешь в школе *«найти сумму двух чисел», «вычислить площадь прямоугольника», «выполнить синтаксический разбор предложения», «найти размер компьютерного файла».*

Последовательность действий в решении задачи называется **алгоритмом**. Исполнитель — это объект, который может выполнить алгоритм. Исполнителем может быть человек, животное или какое-то устройство: компьютер, стиральная машина.

**Алгоритм — это описание последовательности действий, приводящих к решению задачи.**

Свойства алгоритма.

1. **Понятность**. Алгоритм должен быть написан на понятном для исполнителя языке. Действия должны быть точными, ясными, однозначными.
2. **Прерывность** (раздельность). Алгоритм должен представлять собой отдельные шаги. Необходимо использовать минимальное количество шагов. Каждый шаг должен приносить определенный результат.
3. **Результативность**. Каждый алгоритм должен приводить к обязательному решению поставленной задачи.
4. **Обобщенность** (массовость). Алгоритм должен решать не одну какую-то задачу, а некоторый класс однотипных задач. Например, написали алгоритм для вычисления суммы двух чисел. Этот алгоритм должен работать для сложения любых двух чисел.

# 2. Разнообразие исполнителей

### Теория:

Ранее мы говорили о том, что алгоритм предназначен для какого-то конкретного исполнителя. Приводили примеры для исполнителя — человек. Но в современном мире нас окружает большое количество разных электронных устройств: смартфон, компьютер, цифровой фотоаппарат, автомобиль, стиральная машина, робот-пылесос и т.д. Все эти устройства предназначены для решения какой-то конкретной задачи и способны выполнять ограниченный набор команд.

**Исполнитель — это устройство, способное выполнять определённый набор действий (команд).**

Каждый исполнитель имеет свою **систему команд исполнителя** (СКИ). СКИ — это список команд, которые способен выполнять исполнитель.

Рассмотрим простой пример. Команды, которые может выполнять монитор компьютера: включение/выключение, настройки яркости, контрастности и т.п. Для каждой такой команды на мониторе есть отдельная кнопка.

Есть и более сложные исполнители. Например, стиральная машина. В современных стиральных машинах заложено очень много различных функций: замачивание, стирка, сушка, отжим, полоскание, отложить старт и т.д. Все эти функции описаны командами, и хранятся в электронной памяти устройства. Стиральная машина выполняет весь процесс стирки автоматически без участия человека. Нам только нужно выбрать программу стирки, нажать на кнопку, и процесс начнется.

Самыми совершенными исполнителями считаются **роботы**.

Робот выполняет свою работу точно и быстро. Роботы используются на сложных производствах, могут управлять самолётами и поездами, применяются в медицине, в военных технологиях, обеспечивают безопасность людей, заменяют труд человека в быту.





Самым распространенным и универсальным исполнителем является **компьютер**.

Компьютер может работать с любым видом информации: текстовой, числовой, графической, аудио и видеоинформацией.

Компьютер управляется специальными компьютерными программами, каждая из которых направлена на выполнение своих задач и целей.

Например, чтобы обработать фотографию, нужно воспользоваться графическим редактором.

Для отправки письма по электронной почте необходимо воспользоваться специальной программой — браузером, которая позволяет нам пользоваться Интернетом.

# 3. Формальные и неформальные исполнители

### Теория:



Каждому формальному исполнителю можно дать следующие характеристики.



**Разработка алгоритма** — это очень сложная задача, которая требует от человека большого количества знаний и много времени.

Исполнитель, выполняя алгоритм, следует по строго указанным командам, не понимая зачем и почему надо сделать именно так. Другими словами, исполнитель работает **формально**. Как только люди научились разрабатывать точные алгоритмы, появилась возможность автоматизировать труд человека, тем самым освободить его от рутинной работы. Для этого процесс решения некоторой задачи описывают в виде последовательности простейших действий, дальше создается автоматическое электронное устройство, которое выполняет все эти действия. Тем самым человек делает свою жизнь комфортнее.

4. Формы записи алгоритмов

Теория:

Существует несколько способов записи алгоритмов.

1. **Словесный способ**. Алгоритм записывается в виде нумерованного текста. Текст должен быть понятен исполнителю.
2. **Графический способ**. Алгоритм изображается с помощью **блок-схемы** — последовательности геометрических фигур, в которых записываются команды. Элементы блок-схемы между собой соединяются линиями и стрелками, которые показывают ход выполнения алгоритма.



Алгоритм «Собери портфель», записанный блок-схемой.



**Программа — это алгоритм, записанный на языке, понятном исполнителю.**

Каждый алгоритм разрабатывается для решения некоторого класса задач.

План разработки алгоритма:

1. выделить главные объекты в задаче и установить связь между ними;
2. определить исходные данные;
3. описать точную последовательность действий исполнителя, которая приведет к нужному результату;
4. действия должны быть понятны конкретному исполнителю, для которого пишется алгоритм.

5. Линейные алгоритмы и алгоритмы с ветвлениями

Теория:

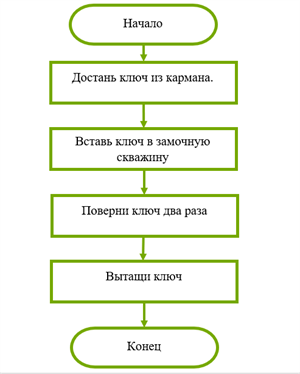
**Линейный алгоритм (следование) — это алгоритм, который описывает последовательно выполняющиеся действия.**

Рассмотрим простой пример линейного алгоритма.

Алгоритм «Открой дверь».

1. Начало.
2. Достань ключ из кармана.
3. Вставь ключ в замочную скважину.
4. Поверни ключ два раза.
5. Вытащи ключ.
6. Конец.

Изобразим данный алгоритм графически, с помощью блок-схемы.

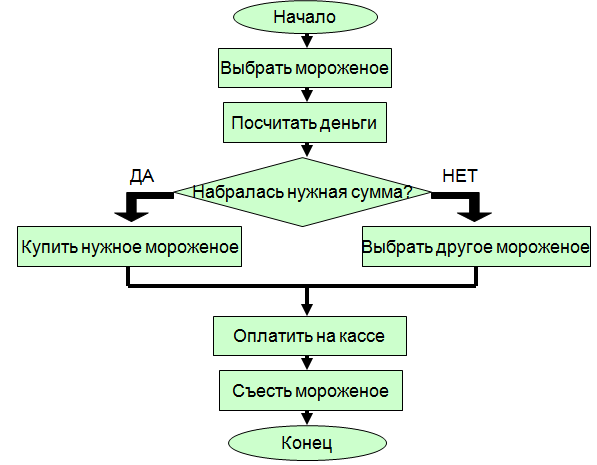


**Алгоритм с ветвлением (разветвляющийся) — это алгоритм, в котором в зависимости от результатов проверки условия выполняется либо одно действие, либо другое.**

Редко в нашей жизни встречаются ситуации, когда известна чёткая последовательность действий. Часто мы стоим перед выбором и принимает решение в зависимости от ситуации. Если на улице светит солнце, то зонт и дождевик оставим дома, иначе всё это возьмем с собой. Но выбор не всегда бывает таким простым.

**Общий вид: ЕСЛИ <условие> ТО <действие**1**> ИНАЧЕ <действие**2**>.**

Рассмотрим пример алгоритма «Купи мороженное».



# 6. Алгоритмы с повторениями

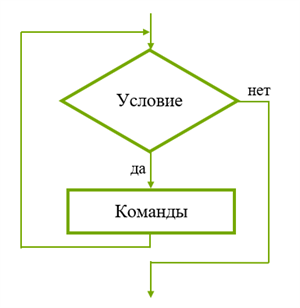
### Теория:

Часто в задачах одно или несколько действий необходимо повторять несколько раз, пока не выполнится заранее обговоренное условие.

**Алгоритм с повторением или цикл — это алгоритм, в котором одно или несколько действий повторяются до тех пор, пока не будет выполнено условие.**

**Зацикливание — это бесконечное повторение одного и того же действия, то есть выполнение цикла никогда не заканчивается.**

Графически алгоритм с повторением можно представить следующим образом.



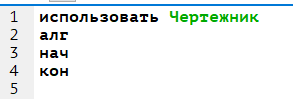
Если условие выполняется, то мы двигаемся по стрелочке «да» и выполняем ряд команд, после этого условие проверяется до тех пор, пока оно не перестанет выполнятся, тогда по стрелочке «нет» можно будет переходить к последующим действиям.

7. Знакомимся с Чертёжником

Теория:

**Главной функцией исполнителя Чертёжник является построение рисунков на координатной плоскости.**

Для того, чтобы перемещать **Перо** по координатной плоскости необходимо написать алгоритм, используя **систему команд исполнителя Чертёжник**.



Команды исполнителя Чертёжник:

* опустить перо;
* поднять перо;
* сместиться в точку (X,Y);
* сместиться на вектор (dX,dY);
* установить цвет (цвет);
* надпись (ширина, текст).

*Обрати внимание!*

Координаты точки x и y разделяются запятой.

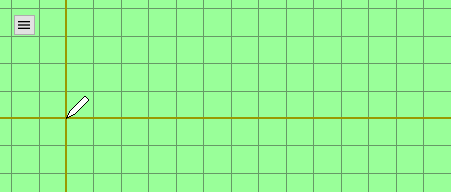
Рассмотрим подробнее каждую команду.

**Опустить перо**. Если перо Чертёжника опущено, то оно будет оставлять за собой след от предыдущей координаты к следующей в виде линии. С помощью этого можно рисовать в Чертёжнике различные фигуры.

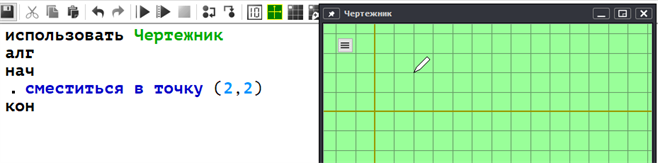
**Поднять перо**. Когда перо поднято, след за ним не остаётся. Изначально перо всегда поднято.

**Сместиться в точку**(X,Y). Выполняя эту команду, перо Чертёжника перемещается в указанную точку независимо от предыдущего положения.

Начальное положение Чертёжника (0,0).



Выполним команду *сместиться в точку*(2,2).



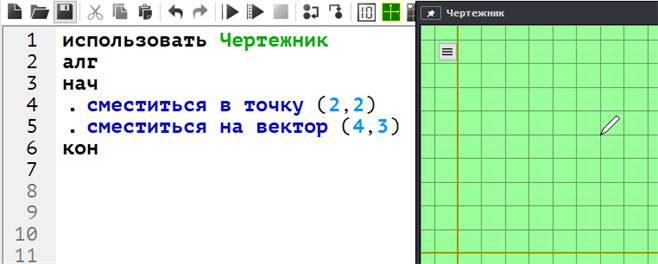
**Сместиться на вектор**(dX,dY).

**Векторы — это направленные отрезки.**

При выполнении данной команды перемещение пера зависит от предыдущего положения.

Например, после того как Чертёжник сместился в точку (2,2), дадим ему команду *сместиться на вектор*(4,3).

Перо окажется в точке (2+4,2+3), т.е. (6,5).



*Обрати внимание!*

Команды для исполнителя нужно записывать **правильно**, иначе он не сможет их выполнить.

# 8. Пример алгоритма управления Чертёжником

### Теория:

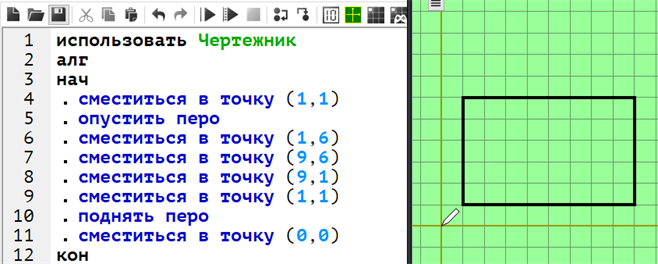
Построим с помощью Чертёжника прямоугольник с координатами вершин (1,1),(1,6),(9,6),(9,1).

Начальное положение Чертёжника в точке (0,0), перо поднято.

Переместим перо в точку (1,1)  (строка 4).

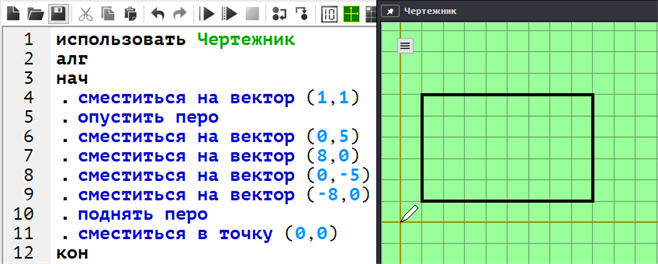
Опустим перо (строка 5), чтобы начать рисовать прямоугольник. Далее будем давать команды перемещения.

Перо Чертёжника необходимо вернуть в исходное положения, поэтому после того, как мы окажемся в точке (1,1), нужно поднять перо и вернуться в точку (0,0)  (строки 10,11).



Данный алгоритм не является массовым. Если изменить координаты одной из вершин прямоугольника, то придётся пересчитывать координаты других вершин вручную разработчику алгоритма.

Попробуем изобразить этот же прямоугольник командой сместиться на вектор.



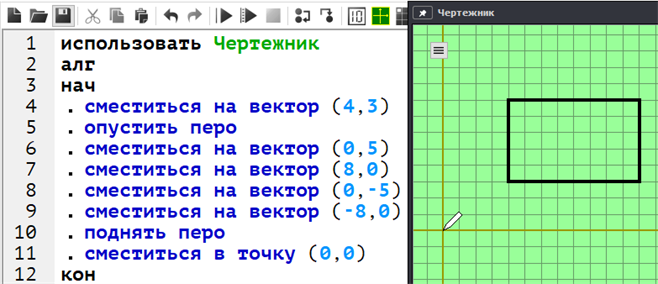
Для перемещения пера нужно указывать не координаты следующей вершины, а на сколько нужно сместиться из данной точки.

Сначала перемещаемся в начальную точку с координатами (1,1).

Чтобы оказаться в следующей вершине (1,6), из точки (1,1) нужно сместиться на 5 по оси Y, по оси X изменений не будет, поэтому запишем 0. Получится (0,1+5) (строка 6). Окажемся в точке (1,6). Чтобы попасть следующую вершину (9,6) нужно переместиться по оси X на 8  (строка 7). Дальше сместимся на вектор (0,−5)(строка 8) и окажемся в точке (9,1). И вернемся в первую вершину командой сместиться на вектор(−8,0)(строка 9).

Команда сместиться на вектор позволяет не привязываться к начальной точке, а даёт возможность построить данный прямоугольник в любом месте координатной плоскости.

Изменим координаты начальной точки на (4,3).



# 9. Использование вспомогательных алгоритмов

### Теория:

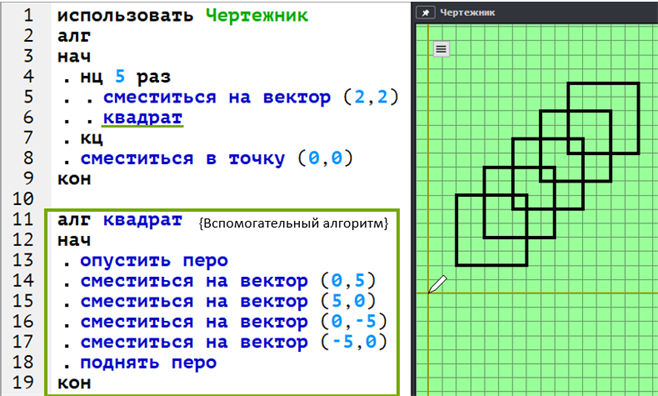
**Вспомогательный алгоритм — это алгоритм, который можно использовать в составе другого алгоритма.**

Нарисуем рисунок из квадратов.

Для того, чтобы не писать несколько раз один и тот же алгоритм, напишем алг квадрат.

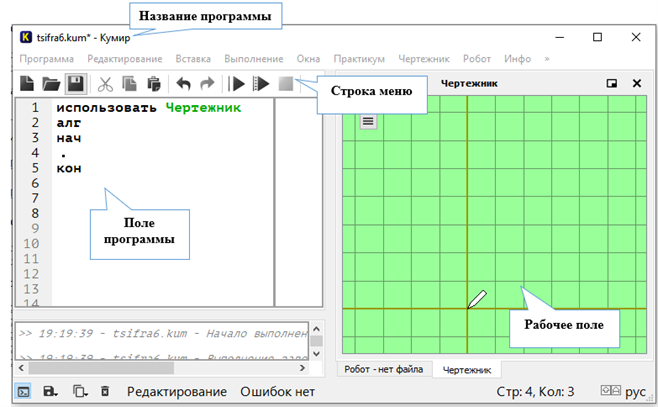
Далее напишем основной алгоритм.

Будем использовать цикл, т. е. повторим некоторые действия несколько раз. В данном случае будем повторять алг квадрат 5 раз.

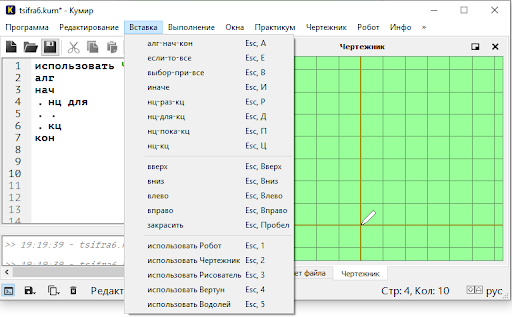


# 10. Чертёжник

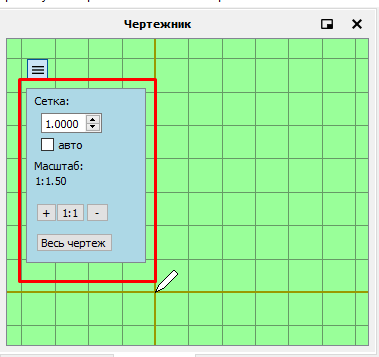
### Теория:



Что умеет делать Чертёжник?



Настройка → Параметры



С помощью специальных команд можно установить цвет линий, нарисовать окружность и залить замкнутую область. Установить один из стандартных цветов:



# 11. Цикл повторить n раз

### Теория:

**Цикл — это многократное повторение некоторой последовательности действий.**

Нарисуем с помощью Чертёжника узор из звёзд.

Сначала напишем **вспомогательный алгоритм**, который будет чертить одну звезду.

Далее в основном алгоритме будем использовать цикл.

**Общий вид цикла:  
нц <число повторений> раз**

**<тело цикла>**

**кц**

**Число повторений** показывает, сколько раз будет выполняться цикл, а **тело цикла**какие действия будут повторяться.

