



Мурашкина Ксения Борисовна

*Преподаватель подготовительных курсов
факультета вычислительной математики и кибернетики
МГУ имени М.В. Ломоносова.*

Как не заблудиться в лабиринтах блок-схем

Чтобы выполнить любую задачу из повседневной жизни (например, пойти покататься на велосипеде) или решить математическую задачу (найти корни квадратного уравнения), требуется произвести определённую последовательность действий. Для выезда на велосипеде – проверить техническое состояние велосипеда, должным ли образом накачаны колёса, если нет, то подкачать, выйти на улицу, определить маршрут. Для решения квадратного уравнения – найти дискриминант, определить его знак, вычислить корни, если таковые имеются. Такие последовательности точных и понятных исполнителю команд, которые определяют порядок действий, необходимых для решения поставленной задачи, а также их содержание, называются алгоритмами. Один из способов описания алгоритмов – блок-схемы. О них и пойдёт речь в предлагаемой статье.

Существуют различные способы представления алгоритмов. Графический способ (с помощью блок-схем) является, пожалуй, самым наглядным и компактным. Каждой команде алгоритма соответствует свой графический элемент – геометрическая фигура, а сами элементы соединены между собой линиями со стрелками, которые определяют порядок выполнения действий. Главных графических элементов только два – прямоугольник и ромб, а для ввода начальных данных и вывода результатов обычно используются овалы или круги.

Давайте разберём блок-схему самого простого *линейного* алгоритма, в котором команды следуют одна за другой, на примере решения обыденной задачи вскипятить воду, для того чтобы заварить чай. Какие команды нам нужно выполнить и каков их порядок? Сначала опишем наши действия словами, а потом нарисуем блок-схему.

1. Найти чайник.
2. Налить в него воды.
3. Зажечь газ.
4. Поставить чайник на плиту.
5. Дождаться, пока чайник закипит.
6. Выключить газ.
7. Заварить чай.



Таким образом, на каждом шаге нам просто нужно выполнять конкретное действие.

В блок-схеме мы каждое конкретное действие поместим в графический элемент – прямоугольник, затем последовательно соединим прямоугольники линиями со стрелочками, указав тем самым порядок выполнения команд (рис. 1).

Как видите, каждый прямоугольник характеризуется тем, что имеет только по одной стрелке на входе и выходе (кроме первого и последнего), что предполагает однозначность перехода от предыдущей инструкции к последующей.

За более сложную операцию – *ветвление* алгоритма в блок-схемах – отвечает графический элемент ромб. Слово «ветвление» означает, что в зависимости от выполнения или невыполнения условия, которое указывается внутри ромба, дальше реализация алгоритма пойдёт по одной из двух возможных ветвей. У ромба всегда один вход и два выхода, на которых написаны слова «да» и «нет» (или «истина» и «ложь»), которые являются ответом на вопрос, выполнено условие в ромбе или нет.

На рис. 2 представлен общий случай процесса ветвления. Если условие, записанное в ромбе, выполняется, то осуществляется переход

к действиям, которые идут по ветви «да», если же не выполняется, то переход к следующей команде нужно делать по ветви «нет».



Рис. 1

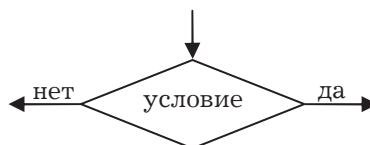


Рис. 2

Рассмотрим блок-схему, содержащую простое ветвление, на примере задачи вычисления модуля числа (рис. 3). Вводится число A , требуется переменной B присвоить значение модуля A , т. е. если $A > 0$, то $B := A$, в противном случае (когда $A < 0$, другого быть не может, в данной задаче мы не рассматриваем

случай $A = 0$) $B := -A$. Знак « $:=$ » в языке программирования Паскаль обозначает «присвоить», или занести значение переменной A в ячейку B . Будем использовать этот знак, чтобы не путать математический знак равенства с операцией присваивания.

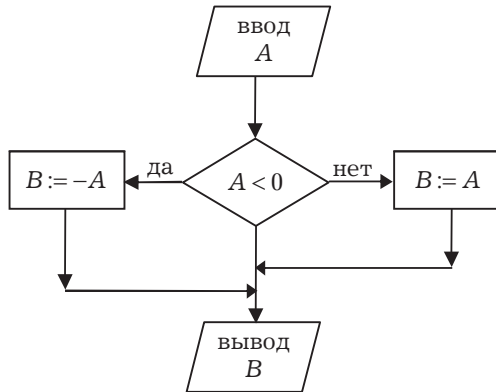


Рис. 3

У нас на блок-схеме появился новый графический элемент – параллелограмм. Он обычно используется для того, чтобы показать, какие данные вводятся и значения каких переменных являются результатом работы программы.

В этой блок-схеме, вне зависимости от того, по какой ветке мы двигаемся, нужно выполнить только один оператор присваивания, хотя в прямоугольниках могут находиться и несколько команд, а после проверки первого условия может стоять другое условие.

Рассмотрим более сложный пример. Требуется определить значение переменной Y после прохода по блок-схеме (рис. 4), в которой мы видим ещё один новый графический объект – овал, который обозначает начало и конец работы блок-схемы. (Знак $*$ означает умножение.)

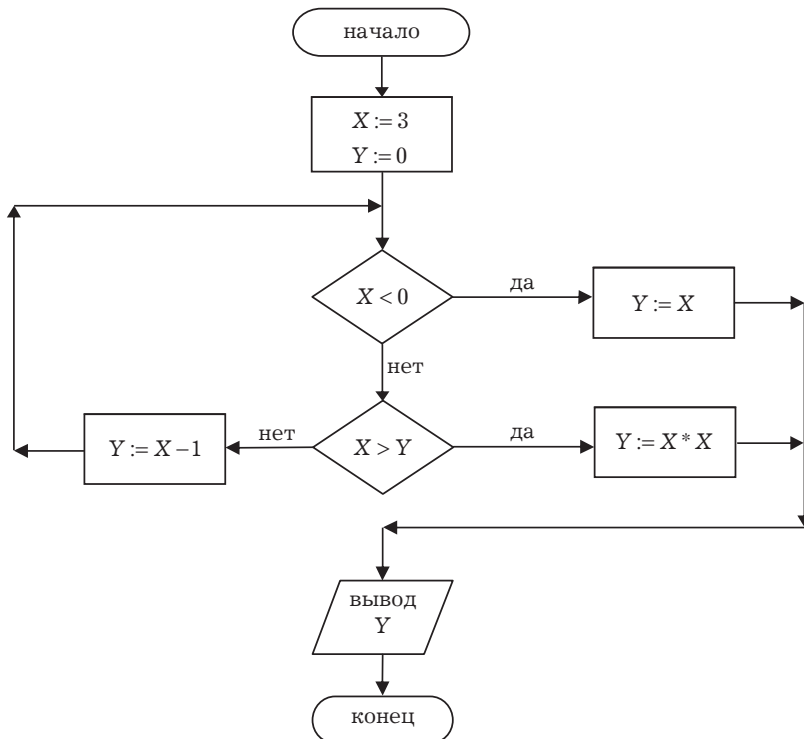


Рис. 4

Работа этой блок-схемы начинается с того, что переменным X и Y мы присваиваем начальные значения $X=3$, $Y=0$. Затем переходим к проверке условия в первом ромбе, это условие у нас не выполняется ($X=3>0$), поэтому мы должны двигаться по ветви «нет» вниз, и попадаем опять на проверку, но уже другого условия $X>Y$. Это условие выполняется, поэтому мы должны перейти к выполнению присваивания $Y:=X*X$, т. е. занести в ячейку Y квадрат переменной X . На этом работа блок-схемы заканчивается, результатом является число 9. Как видим, на выполнение двух команд $Y:=X$ и $Y:=X-1$ при таком образом заданных начальных значениях Y и X мы не попали. Проверьте, какое значение Y получится в результате при других начальных данных, например, $X=-5$, $Y=10$. В этом примере мы выполнили проверку уже двух условий в строгом соответствии с очередностью их записи в блок-схеме. Схема, которая реализует проверку многих условий и разветвление алгоритма по многим ветвям, представлена на рис. 5.

Самой сложной и важной в теории алгоритмов (и в программировании) является структура, которая называется «цикл». В ней определённая последовательность команд, именуемая телом цикла, должна выполняться несколько раз. Циклические структуры бывают двух типов – циклы с условием и циклы со счётчиком. Разберём каждый тип отдельно. Если мы заранее знаем количество повторений тела цикла, нужно использовать цикл со счётчиком. Поясним, как всё происходит, на примере блок-схемы, в которой наша задача – определить значение переменной Y (рис. 6).

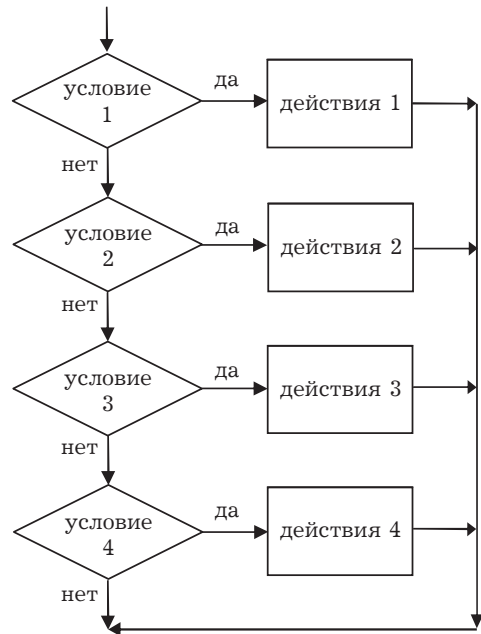


Рис. 5

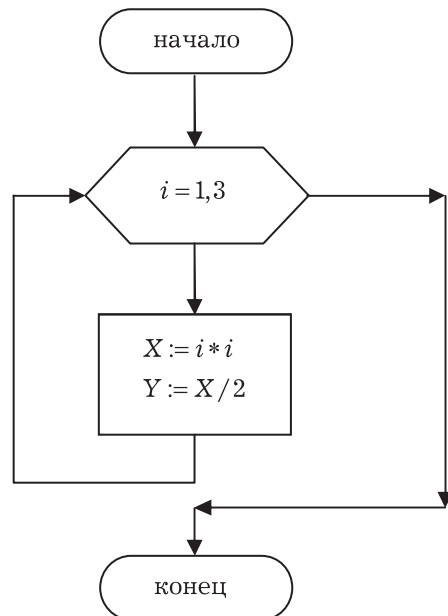


Рис. 6

В графическом элементе шестиугольной формы заданы через запятую начальное и конечное значения счётчика (или параметра) цикла i .

После каждого прохождения тела цикла (на приведённой на рис. 6 блок-схеме это действия, записанные в прямоугольнике) значение параметра цикла увеличивается на 1, говорят, что шаг цикла равен 1. Можно задавать и другое изменение счётчика цикла на каждом шаге (например, 2).

При первом прохождении тела цикла при $i=1$ выполняются следующие операторы присваивания: $X := i * i$, $Y := X / 2$. В переменную X заносится значение 1, а значение Y становится равным $1/2$. Далее двигаемся по стрелочке и увеличиваем значение i на 1. При втором прохождении тела цикла (при $i=2$) получаем $X=4$ и $Y=2$. Затем третий раз попадаем в шестиугольник, опять увеличиваем значение i на 1 ($i=3$) и последний раз прокручиваем тело цикла. В результате имеем $X=9$ и $Y=4,5$. На этом наше выполнение тела цикла заканчивается, мы дошли до верхней границы изменения параметра цикла и идём по стрелочке к концу блок-схемы. Очень важное замечание: в циклах с параметром нельзя изменять значение параметра внутри тела цикла, начальное, конечное значения счёт-



чика и шаг его изменения задаются один раз при входе в цикл.

В тех случаях, когда нам заранее неизвестно, какое количество раз следует выполнить тело цикла, используются циклы с условием. На каждом шаге производится проверка условия, причём такая проверка может проводиться и до, и после выполнения тела цикла. Общий вид этих двух видов цикла с условием показан на рис. 7.

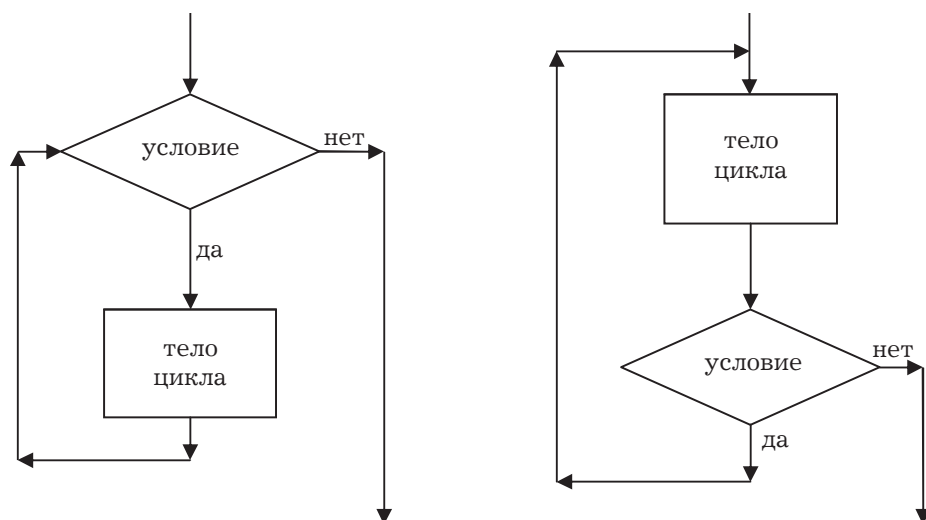


Рис. 7

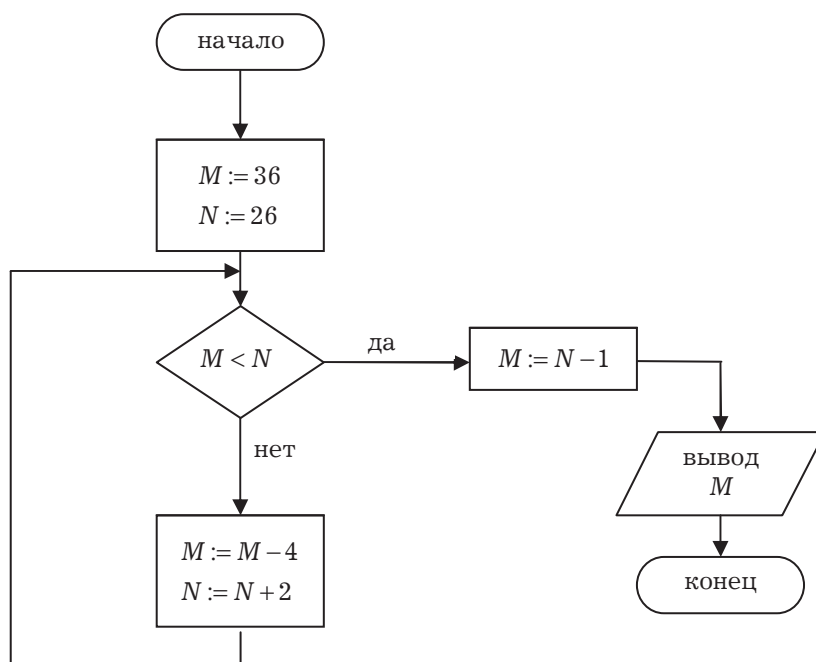


Рис. 8

Рассмотрим на примерах оба вида. Определим значение переменной M после выполнения блок-схемы (рис. 8).

Начальные значения переменных M и N полагаются равными 36 и 26 соответственно, далее идём по стрелочке на проверку условия $M < N$, оно не выполняется, поэтому значения переменных изменяются, значение M становится равным 32 ($36 - 4 = 32$), а $N = 28$. При этих новых значениях опять попадаем на проверку, условие не выполняется ($32 < 28$ – неверно), продолжаем изменять значения обеих переменных: $M = 32 - 4 = 28$, $N = 28 + 2 = 30$. Теперь третий раз проверяем условие, оно выполняется, осуществляем переход по ветви «да» в блок-схеме, выполняем команду $M := N - 1$ и получаем результат $M = 29$. Таким образом, мы три раза «прокрутили» тело цикла, в котором у нас стоял блок из двух операторов присваивания,

после чего вышли из цикла, получив искомое значение переменной M .

Как упоминалось, проверка условия, при котором осуществляется выход из цикла, может стоять после тела цикла. Рассмотрим задачу определения значения переменной S по следующей блок-схеме (рис. 9).



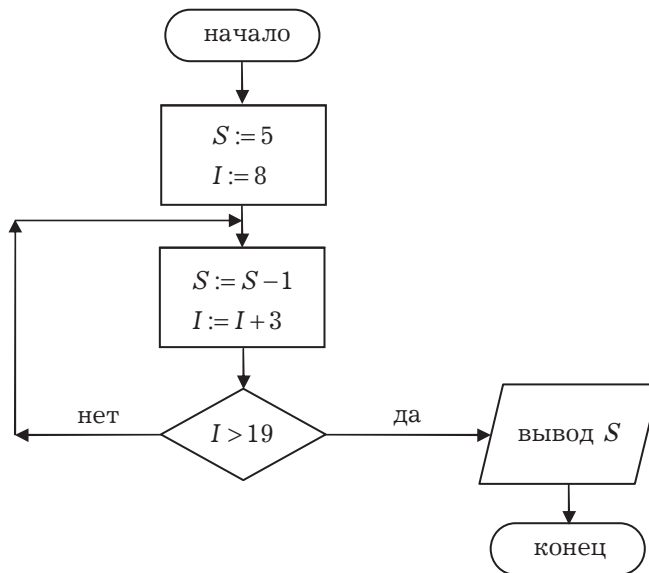


Рис. 9

Начинаем с занесения в ячейки S и I соответственно чисел 5 и 8. Теперь начинается тело цикла. Изменяем значения переменных: $S = 5 - 1 = 4$, $I = 8 + 3 = 11$, идём на проверку условия $I > 19$, оно не выполняется ($I < 19$), двигаемся по ветви «нет» к блоку изменения значений переменных S и I . Получаем новые значения $S = 3$, $I = 14$, невыполнение условия в ромбе приводит нас по ветви «нет» опять к выполнению операторов присваивания, и мы получаем $S = 2$, $I = 17$. Очевидно, что нам требуется ещё раз (последний) выполнить команды тела цикла, получаем $S = 1$, $I = 20$, и теперь условие $I > 19$ выполняется, попадаем на ветку «да» в нашей блок-схеме и выводим значение переменной $S = 1$.

А теперь попробуем разобраться в блок-схеме, которая включает в себя все разобранные компоненты. Требуется найти значения переменных X и Y после выполнения блок-схемы, представленной на рис. 10. Начинаем с присвоения начальных значений переменным X , Y , Z ($X = 5$, $Y = -1$,

$Z = 10$). Теперь осуществляем проверку условия в первом ромбе ($Z > 5$), оно выполняется, идём по ветви «да» и попадаем на структуру «цикл со счётчиком». Значение счётчика изменяется от 2 до 4 и автоматически увеличивается на 1 после каждого выполнения тела цикла, а тело цикла состоит из двух операторов, первый уменьшает значение Z на 1, а второй увеличивает Y в два раза, и их нам нужно выполнить три раза. Таким образом, «прокрутив» цикл, мы получаем $Z = 7$, $Y = -8$. Следуя по стрелочке после выполнения цикла, опять приходим на проверку условия $Z > 5$, оно ещё выполняется (хотя значение Z изменилось), и поэтому нам ещё раз нужно выполнить тот же самый цикл. После второго его выполнения в ячейках Z и Y появятся значения 4 и -64 соответственно. Проверка условия $Z > 5$ направляет нас по ветви «нет», т. е. теперь условие уже не выполняется. Во втором ромбе блок-схемы стоит условие $Y > 0$, на его проверку мы попали при $Y = -64$, и это обстоятельство

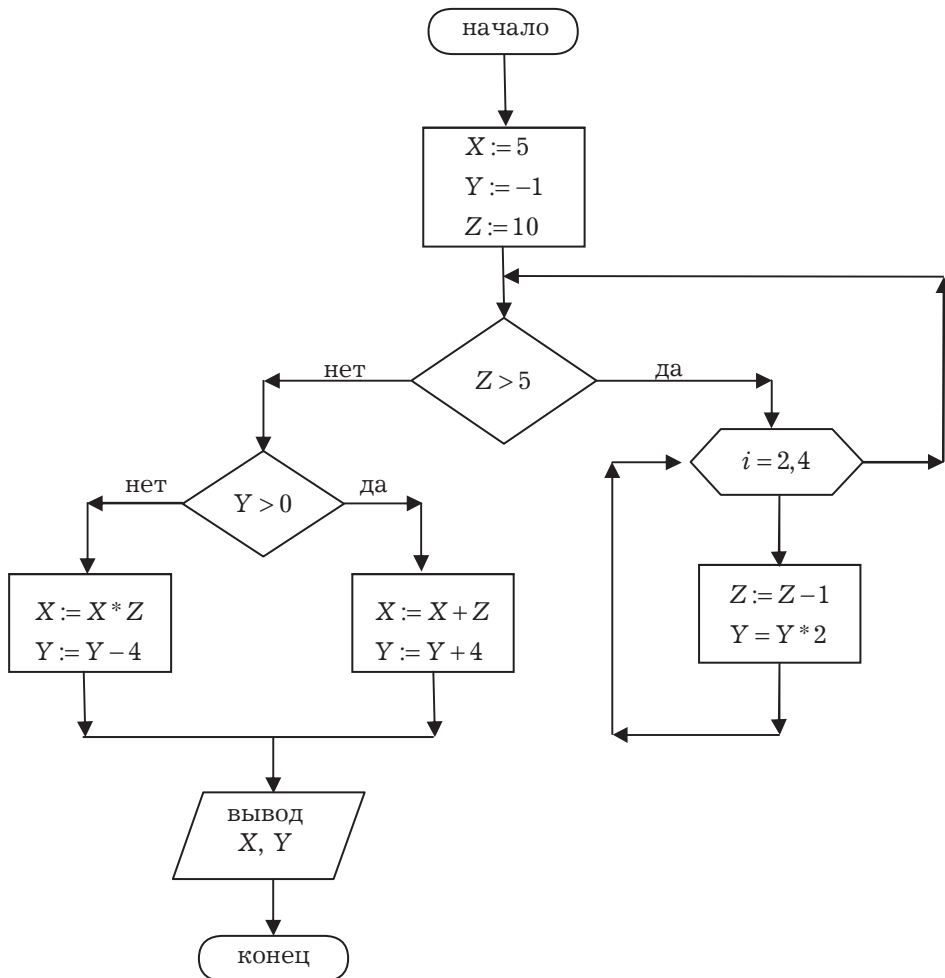


Рис. 10

приводит нас по ветви «нет» к выполнению операторов, стоящих в самом левом прямоугольнике схемы. Уменьшаем значение Y на 4, а в ячейку X заносим произведение X и Y . Результатом работы блок-схемы являются значения $X = 20$ ($5 * 4 = 20$) и $Y = -68$ ($-64 - 4 = -68$).

Подведём итоги. Мы познакомились с основными графическими элементами блок-схем и их назначением, рассмотрели такие сложные алгоритмические понятия, как

ветвление и *цикл*, научились «пробегать» по блок-схемам. Задача ЕГЭ по информатике, в которой используются блок-схемы, требует внимательного к себе отношения. Главное, за чем нужно следить, чтобы не заблудиться в лабиринте блок-схемы, – это по какой ветке двигаться при проверке очередного условия по ходу выполнения алгоритма и не забывать значения переменных, полученных на предыдущих шагах. Успехов!