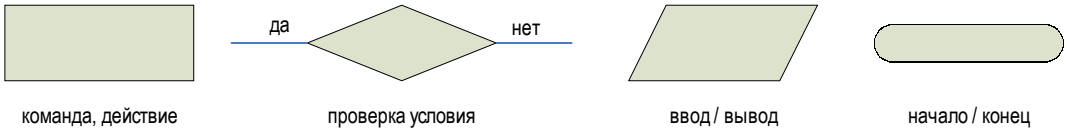
Запись алгоритмов в виде блок-схем

Вылиток А.А.

Блок-схема (нем. *Block*, голл. *blok* и греч. σχημα — наружный вид, форма) — графическое представление алгоритма (или управляющей структуры программы). Состоит из замкнутых фигур стандартной формы — блоков — и соединяющих их стрелок. В блоках записываются команды или названия действий, а стрелки указывают на порядок их выполнения.

Виды блоков:



В качестве примера рассмотрим алгоритм нахождения наибольшего общего делителя двух заданных натуральных чисел m и n. (Пусть m **mod** n означает остаток от деления m на n.)

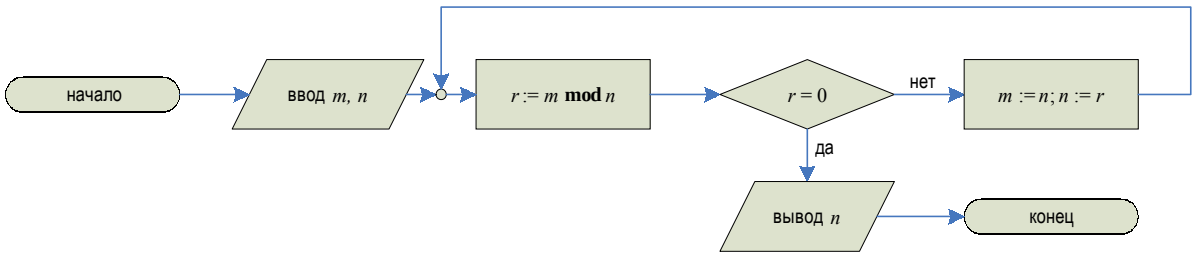
Алгоритм Евклида

Шаг 1. Присвоить r остаток от деления m на n.

Шаг 2. Если $r = 0$, то выполнение алгоритма прекращается; n — искомое значение.

Шаг 3. Присвоить m значение n, присвоить n значение r и вернуться к шагу 1.

Изобразим этот алгоритм в виде блок-схемы:



Покажем, что получаемое в результате число действительно является наибольшим общим делителем. После шага 1 имеем $m = qn + r$, где q — некоторое целое число. Если $r = 0$, то m кратно n и, очевидно, в этом случае n — наибольший общий делитель чисел m и n. Если $r \neq 0$, то любой делитель обоих чисел m и n должен быть также делителем числа $r = m - qn$, и любой делитель чисел n и r должен быть делителем числа $m = qn + r$. Таким образом, множество общих делителей чисел m и n совпадает с множеством общих делителей чисел n и r. Следовательно, пары чисел $<m, n>$ и $<n, r>$ имеют один и тот же наибольший общий делитель. Таким образом, шаг 3 не изменяет ответа исходной задачи. (Очевидно, что если первоначально $m < n$, то частное на шаге 1 оказывается равным нулю и на шаге 3 произойдет взаимный обмен значений переменных m и n.)

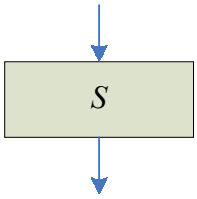
Алгоритм завершится после выполнения конечного числа шагов. Действительно, после шага 1 значение r меньше, чем n. Поэтому если $r \neq 0$, то на следующей итерации цикла значение n на шаге 1 уменьшается. Убывающая последовательность положительных целых чисел имеет конечное число членов,

поэтому шаг 1 может выполняться только конечное число раз для любого первоначально заданного числа *n*.

Хороший стиль программирования и конструирования алгоритмов предполагает использование *структурированных* схем.

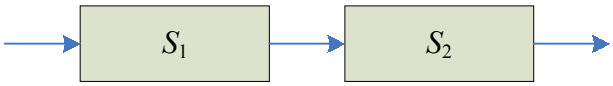
Структурированная схема строится из фрагментов, каждый из которых имеет одну входную и одну выходную стрелку. Простейший фрагмент — пустой — состоит из одной стрелки, входной и выходной одновременно:

Далее идет фрагмент, состоящий из одного оператора



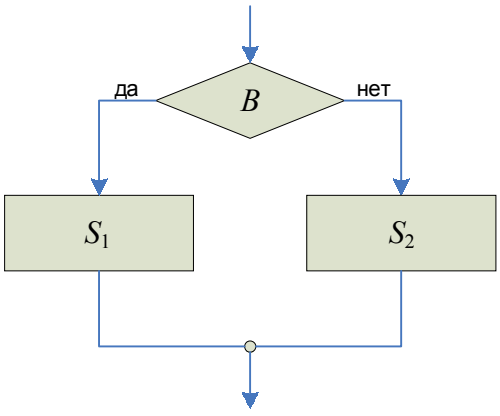
Фрагменты остальных видов (структурированные) получаются композицией двух или трех операторов. Внутренние операторы композиции могут быть простыми (простой оператор означает элементарное действие из системы команд исполнителя) или быть в свою очередь структурированными фрагментами.

При *последовательном* сочленении выходная стрелка одного из двух сочленяемых операторов совпадает со входной стрелкой другого оператора.



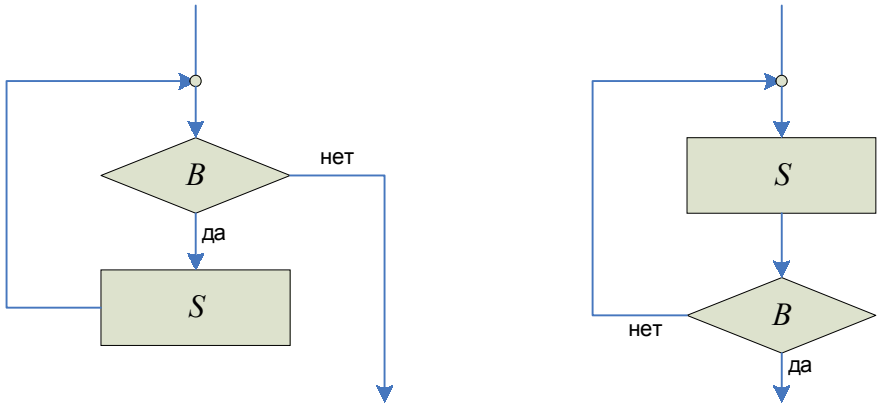
Оставшиеся виды композиции предусматривают наличие в составе фрагмента псевдооператора B, проверяющего выполнение некоторого условия и не осуществляющего иных действий, а поэтому не меняющего состояние программы (значений переменных). Такой оператор имеет две выходные стрелки. Переход по одной их них происходит, если условие удовлетворяется, по другой — если нет.

При альтернативной композиции одна из стрелок ведет к внутреннему оператору S_1, другая — к оператору S_2. Выходные стрелки этих операторов обязательно сливаются в одну и не имеют права вести к разным фрагментам схемы (точка слияния стрелок обозначается маленьким кружком):



Возможны варианты альтернативной композиции, в которых один из внутренних фрагментов пуст.

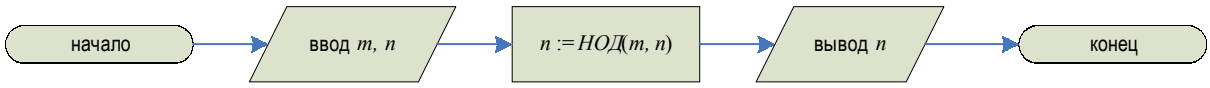
Последняя композиция — *циклическая*. Переход по одной из стрелок после проверки условия означает выход из данного фрагмента. Другая стрелка ведет к внутреннему оператору *S*, выходная стрелка которого ведет вновь к проверке условия. Два подвида этой композиции различаются тем, что входная стрелка фрагмента ведет в первом случае к проверке условия (цикл с *предпроверкой*), во втором — к оператору *S* (цикл с *послепроверкой*). В обоих случаях фрагмент с оператором *S* не может быть пустым.



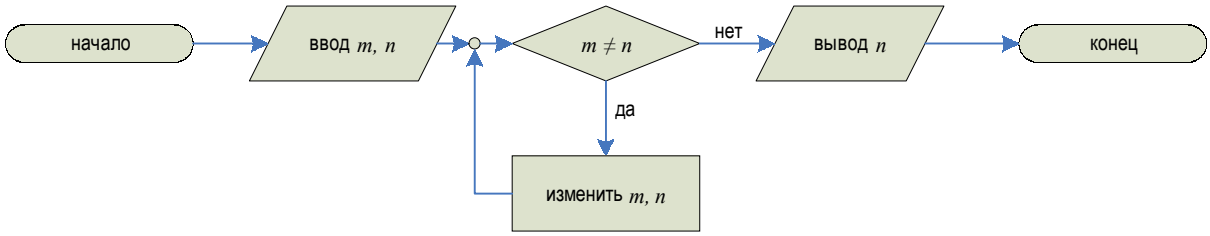
Можно модифицировать алгоритм нахождения наибольшего общего делителя (НОД) чисел *m* и *n*, заметив, что если $m > n$, то $НОД(m, n) = НОД(m - n, n)$, если $m < n$, то $НОД(m, n) = НОД(m, n - m)$, а если $m = n$, то *n* — наибольший общий делитель.

Построим структурированную блок-схему данной модификации алгоритма Евклида, используя *метод последовательных уточнений*.

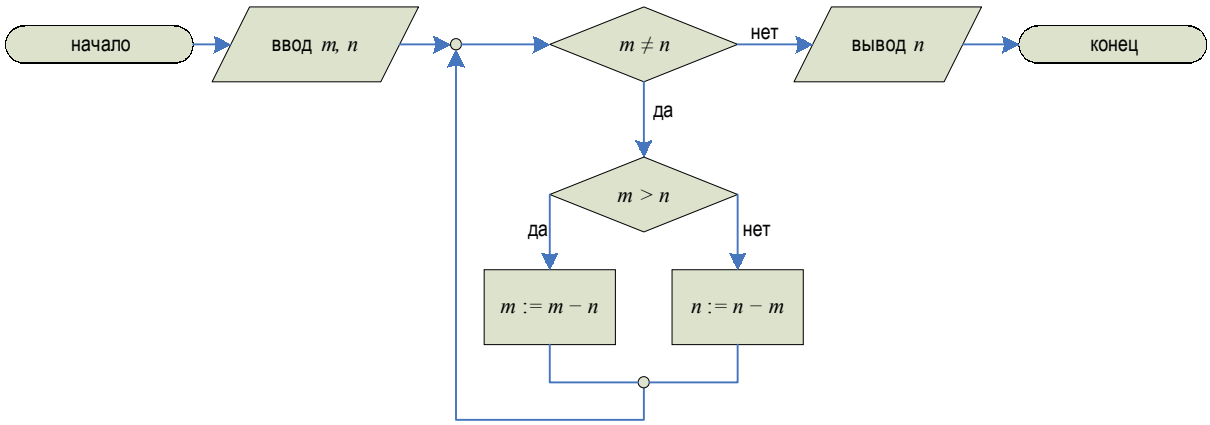
Если бы в системе команд исполнителя была операция $НОД(m, n)$ нахождения наибольшего общего делителя чисел *m* и *n*, блок-схема алгоритма выглядела бы так:



Детализируем оператор $n := НОД(m, n)$: пока $m \neq n$ изменять значения *m* и *n* в соответствии с написанным выше (если $m = n$, то *n* — результат):



Осталось уточнить, как именно следует изменять значения *m* и *n*, и мы получим окончательную схему алгоритма:


начало
ввод m, n
m ≠ n
нет
вывод n
конец
да
m > n
да
нет
m := m − n
n := n − m