Основы алгоритмизации и программирования

Лектор: преподаватель колледжа АлтГУ

Москаленко Елена Валерьевна

Список рекомендуемой литературы по дисциплине:

1.Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для среднего профессионального образования / Д. Ю. Федоров. — 2-е изд. — Москва : Издательство Юрайт, 2021. — 161 с. — (Профессиональное образование).
2. Семакин Основы алгоритмизации и программирования: учебник для студ.учреждений сред. проф. образования / И. Г. Семакин, А. П. Шестаков.— 3-е изд., стер. — М. : Издательский центр «Академия», 2013. — 304 с.
3. Трофимов, В. В.Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2018. — 137 с.
4. Черпаков И. В. Основы программирования:учебник и практикум для СПО: учебник и практикум для СПО - Юрайт, 2017 https://urait.ru/book/osnovy-programmirovaniya-414541

Алгоритмизация – это процесс построения алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения.

Понятие алгоритма

Алгоритм – система четких однозначных указаний, которая определяет последовательность действий над некоторыми объектами и после конечного числа шагов приводит к получению требуемого результата.

Название «алгоритм» произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда ибн Муса ал-Хорезми (Alhorithmi), жившего в 783—850 гг, который в IX веке в своей книге «Об индийском счете» сформулировал правила выполнения четырех арифметических действий над десятичными числами. В латинских переводах с арабского арифметического трактата ал-Хорезми его имя транскрибировалось как *algorismi*. Откуда и пошло слово «алгоритм».

- Термин «алгоритм», впервые употребленный в современном значении. Лейбницем (1646–1716).
- Научное определение алгоритма дал А. Чёрч в 1930 году. В наше время понятие алгоритма является одним из основополагающих понятий вычислительной математики и информатики.
- Область математики, известная как *теория алгоритмов*, посвящена исследованию свойств, способов записи, области применения различных алгоритмов, а также созданию новых алгоритмов. Теория алгоритмов находит широкое применение в различных областях деятельности человека — в технике, производстве, медицине, образовании и т. д.

- решение задач в информатике всегда связано с преобразованием информации, а значит, исходными данными и результатом работы алгоритма должна быть информация. Это может быть представлено в виде схемы:

- алгоритмы в информатике предназначены для реализации в виде компьютерных программ или для создания некоторой компьютерной технологии. Для выполнения алгоритма требуется конечный объем оперативной памяти и конечное время.

На этапе разработки алгоритма рекомендуется придерживаться следующих правил его составления:

- Алгоритм должен быть максимально прост и понятен.
- Алгоритм должен состоять из мелких шагов.
- Сложная задача должна разбиваться на достаточно простые, легко воспринимаемые части (блоки).
- Логика алгоритма должна опираться на минимальное число достаточно простых базовых управляющих структур.

В итоге процесс разработки алгоритма должен быть направлен на получение четкой структуры алгоритмических конструкций.

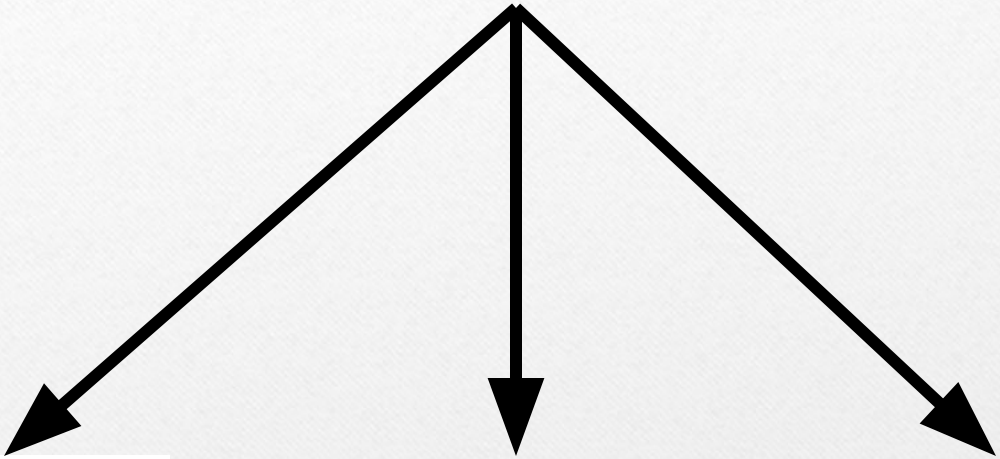
Разработать алгоритм решения означает разбить задачу на последовательно выполняемые этапы. Можно сказать, что алгоритм описывает процесс преобразования исходных данных в результаты, т. к. для решения любой задачи необходимо:

1) ввести исходные данные;

2) преобразовать исходные данные в результаты (выходные данные);

3) вывести результаты.

ИСПОЛНИТЕЛИ АЛГОРИТМОВ



ЧЕЛОВЕК **РОБОТ** **КОМПЬЮТЕР**

Алгоритм предназначен всегда для определенного ***исполнителя*** – человека, робота, компьютера, языка программирования и т. д.

Исполнитель алгоритма — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Исполнителя характеризуют:

- среда;
- элементарные действия;
- система команд;
- отказы.

Умение выполнять определенные команды является ***свойством***, характеризующим любого исполнителя.

Система команд исполнителя – совокупность всех команд, которые данный исполнитель может выполнять. Соответственно алгоритм описывается в командах определенного исполнителя, который будет его реализовывать.

Объекты, над которыми исполнитель может совершать действия, образуют ***среду исполнителя***.

Описание алгоритма решения задачи выполняется в соответствии со следующими правилами:

1. Определяются исходные данные задачи.
2. Процесс решения задачи разбивается на этапы, понятные и однозначные для исполнителя.
3. Указывается порядок, в котором выполняются этапы, а также признак завершения процесса.
4. Определяется, что является результатом решения задачи.

Алгоритмы бывают численными и логическими.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к арифметическим действиям, называются ***численными алгоритмами***.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к логическим действиям, называются ***логическими алгоритмами*** (алгоритмы поиска минимального числа, поиска пути в лабиринте).

Задача: Исполнитель умеет, заменить в слове ровно одну букву на любую другую, причем при замене должно получиться осмысленное слово. Составьте алгоритм для преобразования слова **САД** в слово **КОТ**.

Основные свойства алгоритма:

- дискретность;
- детерминированность (определенность);
- результативность (конечность);
- массовость (универсальность);
- понятность;
- формальность.

Основные свойства алгоритма:

*1.**Дискретность*** – разделение выполнения решения задачи на отдельные операции.

Под *дискретностью* понимается то, что алгоритм состоит из описания последовательности шагов обработки, организованных таким образом, что в начальный момент задаётся исходная ситуация, а после каждого следующего шага ситуация преобразуется на основе данных, полученные в предшествующие шаги обработки.

Основные свойства алгоритма:

2. **Детерминированность (определенность)** – каждая команда алгоритма должна однозначно определять действия исполнителя.

3. **Результативность (конечность) -** завершение работы алгоритма за конечное число шагов (при этом количество шагов может быть заранее не известным и различным для разных исходных данных).

Основные свойства алгоритма:

4. **Массовость (универсальность)** - алгоритм решения задачи разрабатывается в общем виде, то есть возможность решения класса задач, различающихся лишь исходными данными. При этом исходные данные выбираются из некоторой области, называемой областью применимости алгоритма.

5. **Понятность** – содержание допустимого набора команд, понятного конкретному исполнителю.

Основные свойства алгоритма:

6. **Формальность** – это свойство указывает на то, что любой исполнитель, способный воспринимать и выполнять инструкции алгоритма, действует формально, т. е. отвлекается от содержания поставленной задачи и лишь строго выполняет инструкции.

Для оценки и сравнения алгоритмов существует много критериев. Чаще всего *анализ алгоритма* (***анализ сложности алгоритма***) состоит в оценке временных затрат на решение задачи в зависимости от объема исходных данных. Используются также термины «**временная сложность**», «трудоемкость» алгоритма. Фактически эта оценка сводится к подсчету количества основных операций в алгоритме, поскольку каждая из них выполняется за заранее известное конечное время. Кроме временной сложности, должна оцениваться также **емкостная сложность**, т. е. увеличение затрат памяти в зависимости от размера исходных данных.

Оценка сложности дает количественный критерий для сравнения алгоритмов, предназначенных для решения одной и той же задачи. Оптимальным (наилучшим) считается алгоритм, который невозможно значительно улучшить в плане временных и емкостных затрат.

Сущность алгоритмизации вычислительного процесса проявляется в следующих действиях, отражающих его свойства:

- выделении законченных частей вычислительного процесса;
- формальной записи каждого из них;
- назначении определенного порядка выполнения выделенных частей;
- проверки правильности выбранного алгоритма по реализации заданного метода вычислений.

СПОСОБЫ ОПИСАНИЯ АЛГОРИТМОВ:

- словесное описание;
- формульно-словесное описание;
- псевдокод;
- графический способ (блок-схема);
- программа (способ описания с помощью языков программирования).

Словесное описание

Словесное описание представляет алгоритм как инструкцию о выполнении действий в определенной последовательности с помощью слов и предложений естественного языка. Форма изложения произвольна и устанавливается разработчиком.

Этот способ описания не имеет широкого распространения, т. к. строго не формализуем.

Словесное описание

Пример: Алгоритм нахождения наибольшего общего делителя (НОД) двух натуральных чисел:

1. задать два числа;
2. если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
3. определить большее из чисел;
4. заменить большее из чисел разностью большего и меньшего из чисел;
5. повторить алгоритм с п. 2.

Формульно-словесный способ:

Формульно-словесный способ записи действий содержит формальные символы и выражения (формулы) в сочетании со словесными пояснениями. Т.е. алгоритм записывается в виде текста с формулами по пунктам, определяющим последовательность действий.

Формульно-словесный способ:

П р и м е р . Составить алгоритм вычисления площади трапеции с основаниями a,b и высотой h.

Р е ш е н и е . Известно, что площадь трапеции вычисляется по формуле $S = \frac{a+b}{2} h$. Поэтому можно записать алгоритм:

1. Задать численное значение a, b, h.
2. Вычислить выражение $S = \frac{a+b}{2} h$.
3. Записать ответ S.

Псевдокод

Псевдокод представляет собой описание структуры алгоритма на естественном, частично-формализованном языке, позволяющее выявить основные этапы решения задачи перед точной его записью на языке программирования.

В псевдокоде используются некоторые формальные конструкции и общепринятая математическая символика.

Псевдокоды бывают разные.

Способы задания алгоритма

Псевдокод – пошагово-словесная запись алгоритма по определенным правилам или соглашениям. В псевдокоде используется общепринятая математическая символика и конструкции. Он занимает промежуточное место между естественным и формальным языками.

Рассмотрим алгоритм умножения двух чисел, записанный с помощью псевдокода.

1. Ввод a, b.
2. P=a · b.
3. Вывод P.
4. Конец.

Примером псевдокода является школьный алгоритмический язык (АЯ).

Алфавит **учебного алгоритмического языка** является открытым. В него могут быть введены любые понятные всем символы: русские и латинские буквы, знаки математических операций, знаки отношений, специальные знаки и т. д. Кроме алфавита, в алгоритмической нотации определяются *служебные слова*, которые являются неделимыми. Служебные слова обычно выделяются жирным шрифтом или подчеркиванием. К служебным словам относятся:

алг — заголовок алгоритма	**нц** — начало цикла	**знач**
нач — начало алгоритма	**кц** — конец цикла	**и**
кон — конец алгоритма	**дано**	**или**
арг — аргумент	**надо**	**не**
рез — результат	**если**	**да**
цел — целый	**то**	**нет**
сим — символьный	**иначе**	**при**
лит — литерный	**всё**	**выбор**
лог — логический	**пока**	**утв**
вещ — вещественный	**для**	**ввод**
таб — таблица	**от**	**вывод**
длин — длина	**до**	

Общий вид записи алгоритма на псевдокоде:

алг – название алгоритма (аргументы и результаты)

дано – условие применимости алгоритма

надо – цель выполнения алгоритма

нач – описание промежуточных величин

последовательность команд (тело алгоритма)

кон

Часть алгоритма от слова **алг** до слова **нач** называется *заголовком*, а часть, заключенная между словами **нач** и **кон,** — *телом алгоритма* (исполняемой частью алгоритма).

В предложении **алг** после названия алгоритма в круглых скобках указываются *характеристики* (**арг, рез**) и *тип значения* (**цел, вещ, сим, лит** или **лог**) всех входных (аргументы) и выходных (результаты) переменных. При описании массивов (таблиц) используется служебное слово **таб**, дополненное именем массива и граничными парами по каждому индексу элементов массива.

Команды учебного языка:

1. ***Оператор присваивания***, который обозначается «:=» и служит для вычисления выражений, стоящих справа, и присваивания их значений переменным, указанным в левой части. Например, если переменная a имела значение 5, то после выполнения оператора присваивания a := a + 1, значение переменной a изменится на 6.

2. ***Операторы ввода/вывода:***

ввод (список имен переменных)

вывод (список вывода)

Список вывода может содержать комментарии, которые заключаются в кавычки.

3. ***Оператор ветвления*** (с использованием команды **если...то… иначе…всё; выбор**);

4. ***Операторы цикла*** (с использованием команд **для, пока, до**).

Запись алгоритма на псевдокоде:

```
алг Сумма квадратов целых чисел до n включительно (арг цел n рез цел S)
    дано |n > 0
    надо |S = 1 * 1 + 2 * 2 + 3 * 3 + ... + n * n
нач
    цел i
    ввод n;
        S := 0
    нц для i от 1 до n
        S := S + i * i
    кц
    вывод "S = ", S
кон
```

Здесь в предложениях **дано** и **надо** после знака «|» записаны *комментарии.* Комментарии можно помещать в конце любой строки, они существенно облегчают понимание алгоритма.

Графический способ записи алгоритма

Графическая запись, или **блок-схема**, – описание структуры алгоритма с помощью геометрических фигур с линиями связями, показывающими порядок выполнения отдельных инструкций.

Этот способ имеет ряд преимуществ перед остальными:

- наглядное отображение базовых конструкций алгоритма;
- концентрация внимания на структуре алгоритма;
- использование принципа блочности при коллективном решении сложной задачи;
- преобразование алгоритма методом укрупнения (сведения к единому блоку) или детализации (разбиения на ряд блоков);
- быстрая проверка разработанного алгоритма.

Правила выполнения блок-схем:

Блок-схемой называется наглядное изображение алгоритма, когда отдельные действия (этапы алгоритма) изображаются при помощи различных геометрических фигур (блоков), а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок, соединяющие эти фигуры.

Выполнение блок-схем осуществляется по ГОСТ 19.701–90.

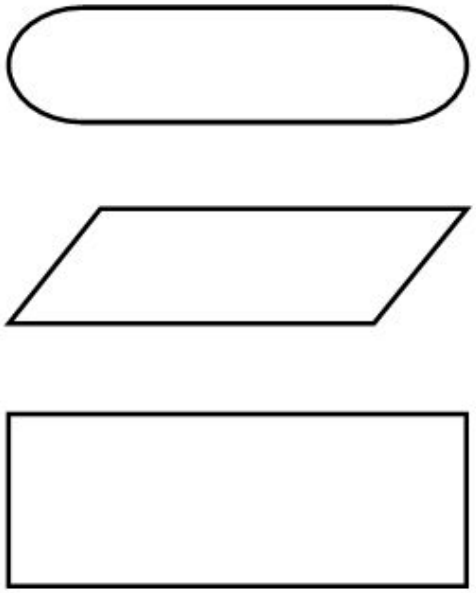
Правила выполнения соединений:

- Стандартное направление линий потока – слева направо и сверху вниз.
- Направление потока указывается стрелками.
- В схемах следует избегать пересечения линий.
- Вход в блок и выход из блока следует размещать по центру символа.
- Если две или более входящих линии объединяются в одну исходящую линию, то место объединения линий смещается.
- Количество входящих линий не ограничено, выходящая линия из блока должна быть одна, за исключением логического блока.

Правила построения алгоритмов на языке блок-схем

1. Блок-схема строится сверху вниз.
2. В любой блок-схеме имеется один элемент, соответствующий началу, и один элемент, соответствующий концу.
3. Должен быть хотя бы один путь из начала блок-схемы к любому элементу.
4. Должен быть хотя бы один путь от каждого элемента блок-схемы в конец блок-схемы.

Основные элементы блок-схем:

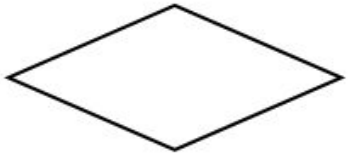


- начало (конец) алгоритма

- блок ввода-вывода данных

- блок вычислений

Основные элементы блок-схем:

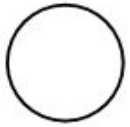


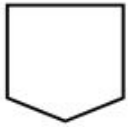
- логический блок, в котором направление потока информации выбирается в зависимости от некоторого условия

- процесс пользователя (подпрограмма)

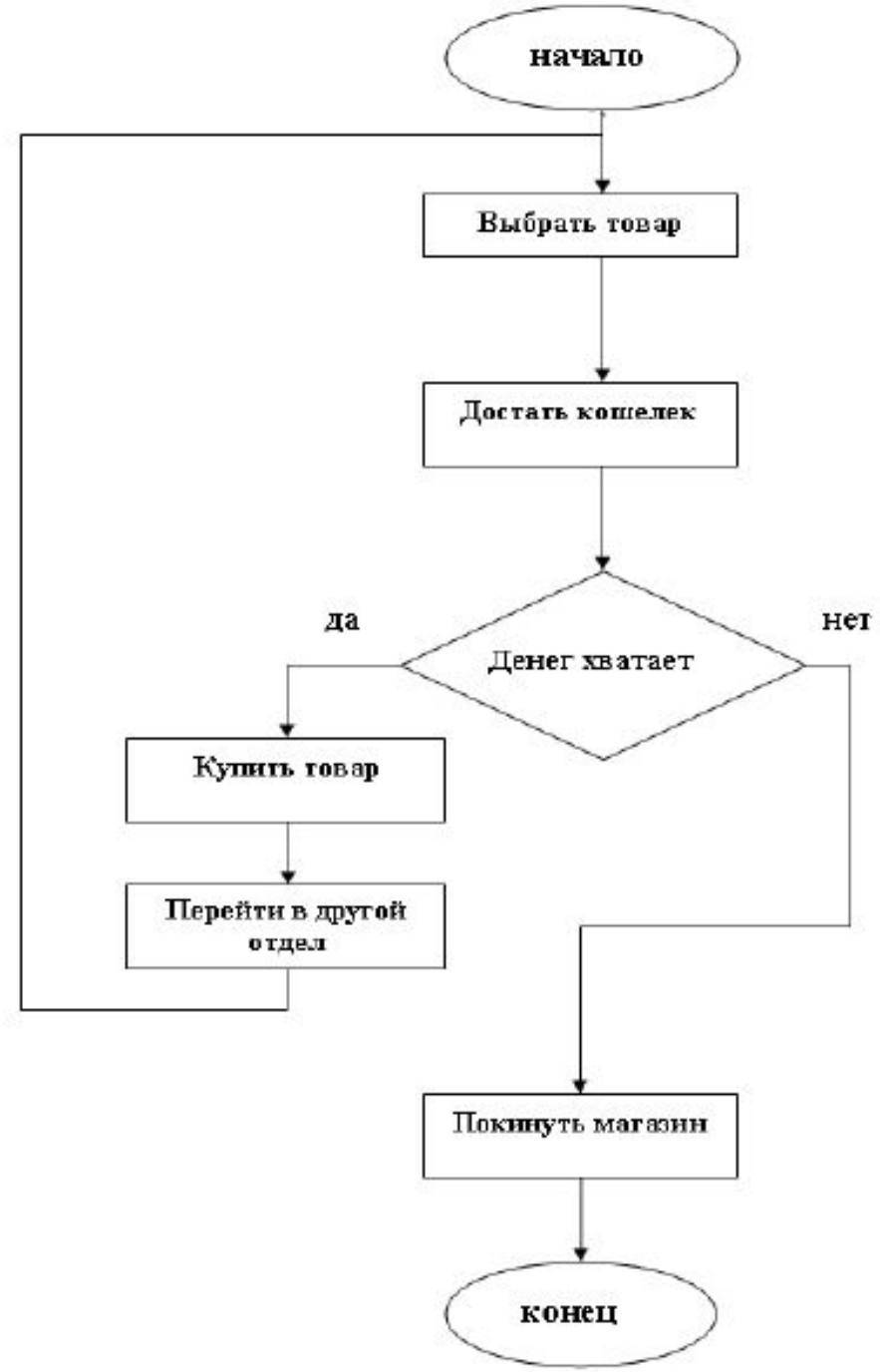
- блок модификации, в котором функция выполняет действия, изменяющие пункты (например, заголовок цикла)

Основные элементы блок-схем:

 - соединитель, используется для указания связи между потоками информации в пределах одного листа

 - межстраничный соединитель, т.е. указание связи между информацией на разных листах

Пример: Алгоритм совершения покупок в магазине



Программа

Программа - это алгоритм, записанный в виде последовательности команд, понятных ЭВМ (машинных команд).

При записи алгоритмов в виде программ для ЭВМ используются языки программирования - системы кодирования предписаний и правила их использования. Такие языки являются искусственными языками со строго определенными синтаксисом и пунктуацией.

Базовые алгоритмические структуры

В зависимости от особенностей своего построения алгоритмы можно разделить на следующие группы:

1) линейные (последовательные);

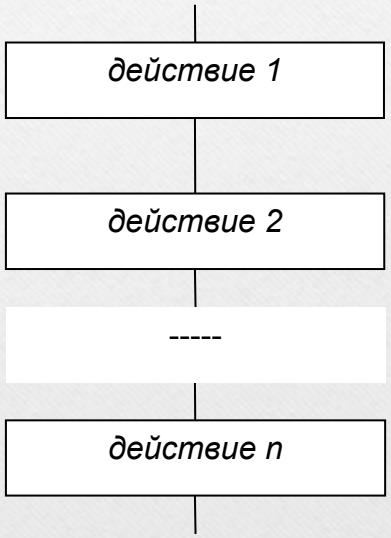
2) разветвляющиеся;

3) циклические;

4) рекурсивные.

Следование (линейные алгоритмы)

Блок-схема

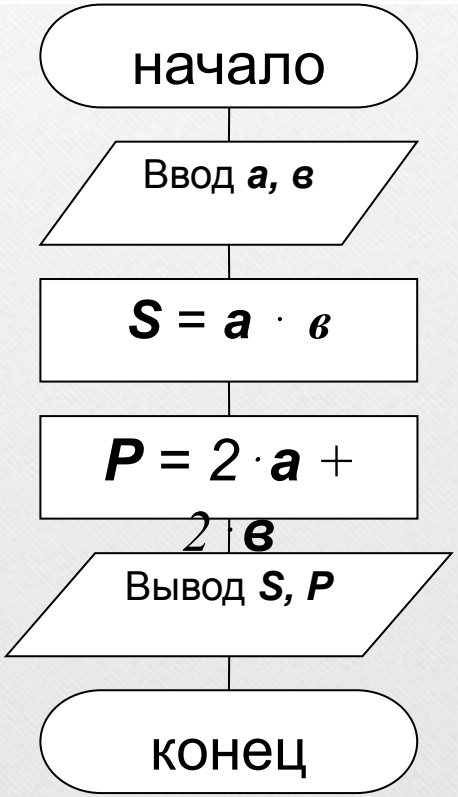


Следование (линейный алгоритм)

При разработке линейного алгоритма необходимо учитывать что:

- данный вид алгоритма является простейшим;
- он чаще используется для реализации простых вычислений по формулам;
- инструкции в нем выполняются последовательно, одна за другой.

Пример: Разработать блок-схему алгоритма вычисления площади и периметра прямоугольника по двум заданным сторонам ***а*** и ***в***.



Разветвляющаяся алгоритмическая конструкция (ветвление)

Разветвляющийся алгоритм — это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

Структура ВЕТВЛЕНИЕ существует в двух основных вариантах:

ВЕТВЛЕНИЕ:

- **Полное**
- **Неполное**
- **Выбор (неполный)**
- **Выбор – иначе (полный)**

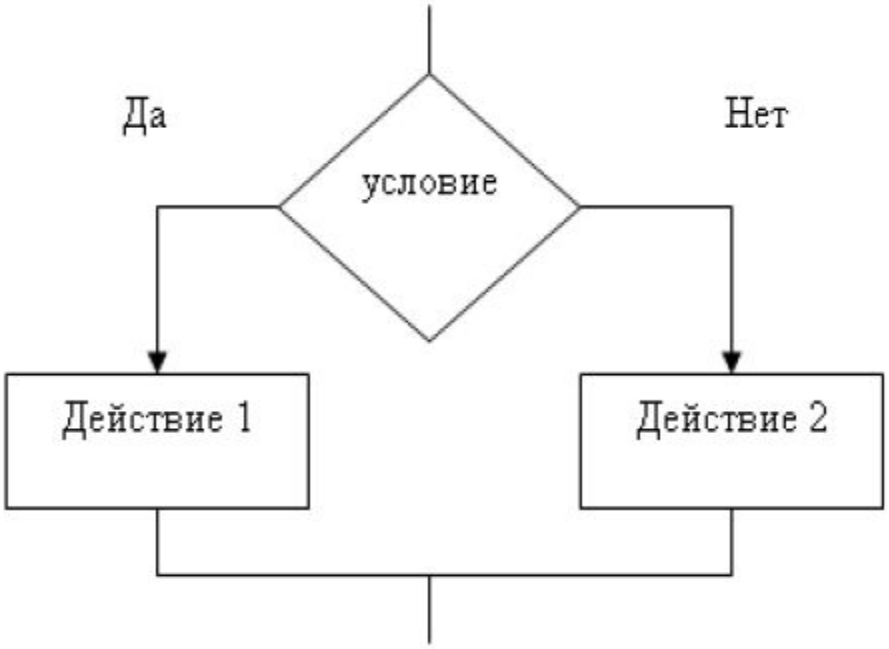
Полное ветвление

Предполагает выполнение действий для обеих веток в алгоритме:

Если [условие], то [действие 1], иначе [действие 2]

Структура полного ветвления:

Блок-схема



Псевдокод

Если [условие]
то [Действие 1]
иначе [Действие 2]
конец если

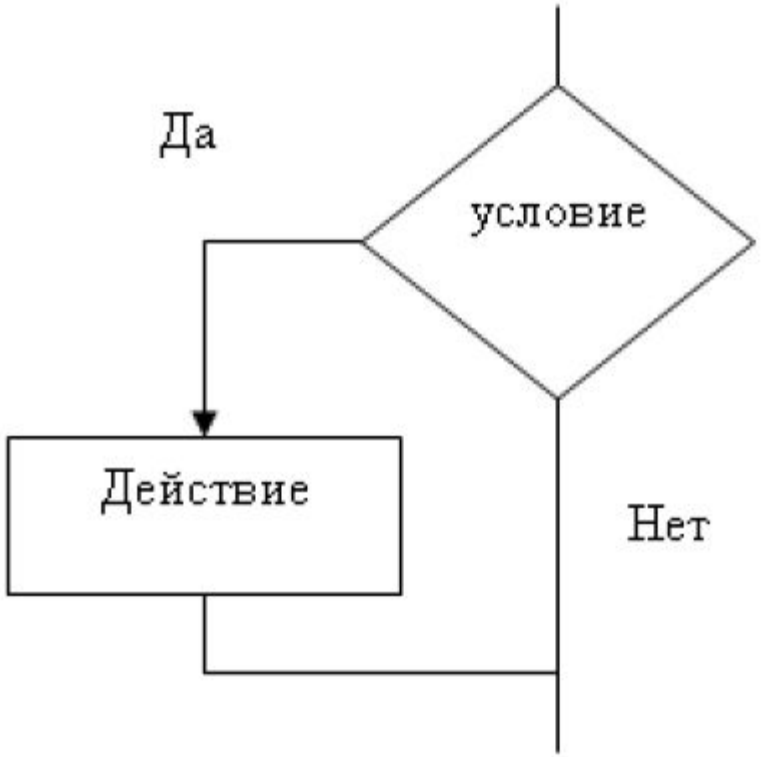
Неполное ветвление

Предполагает выполнение действий только на одной ветви алгоритма (вторая отсутствует):

Если [условие], то [действие]

Структура неполного ветвления:

Блок-схема



Псевдокод

Если [условие]
то [Действие]
конец если

Выбор

выбор

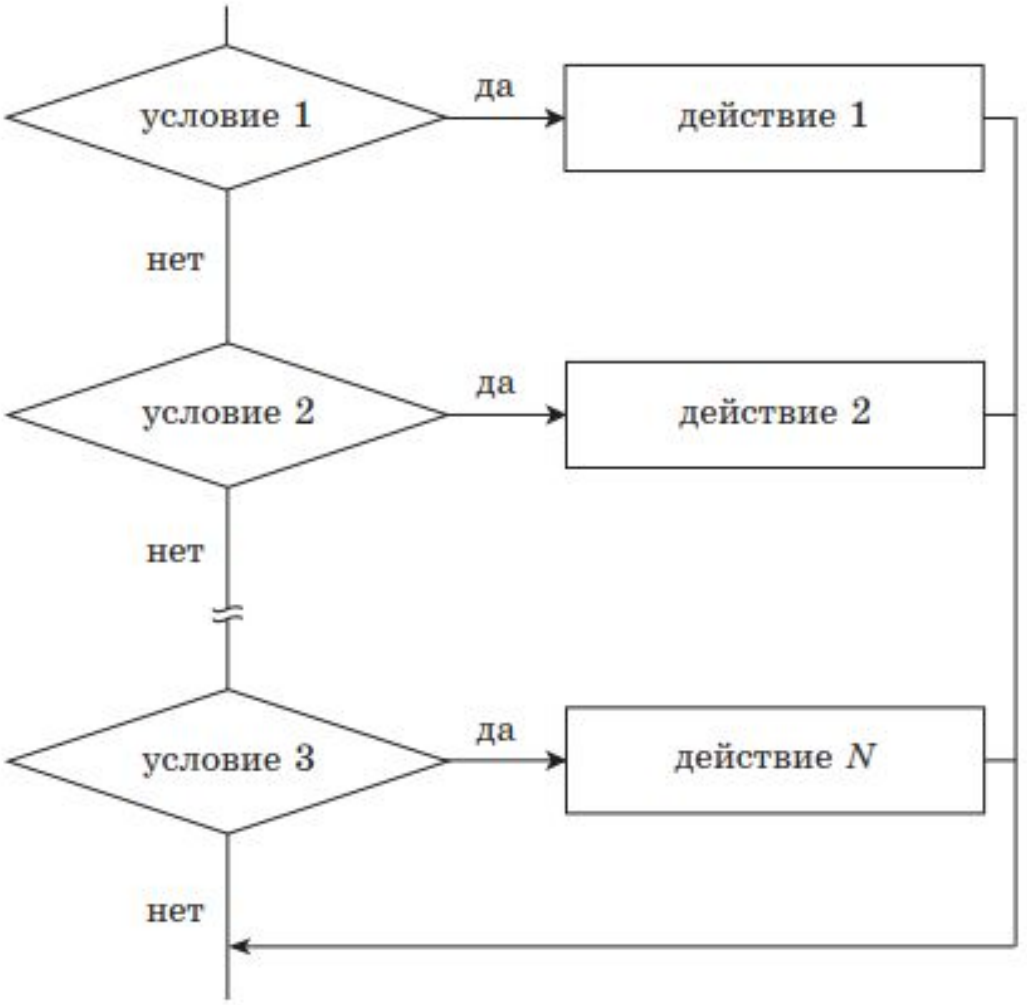
при условие 1: действие 1

при условии 2: действие 2

...

при условие N: действие N

всё



Выбор - иначе

выбор

при условие 1: действие 1

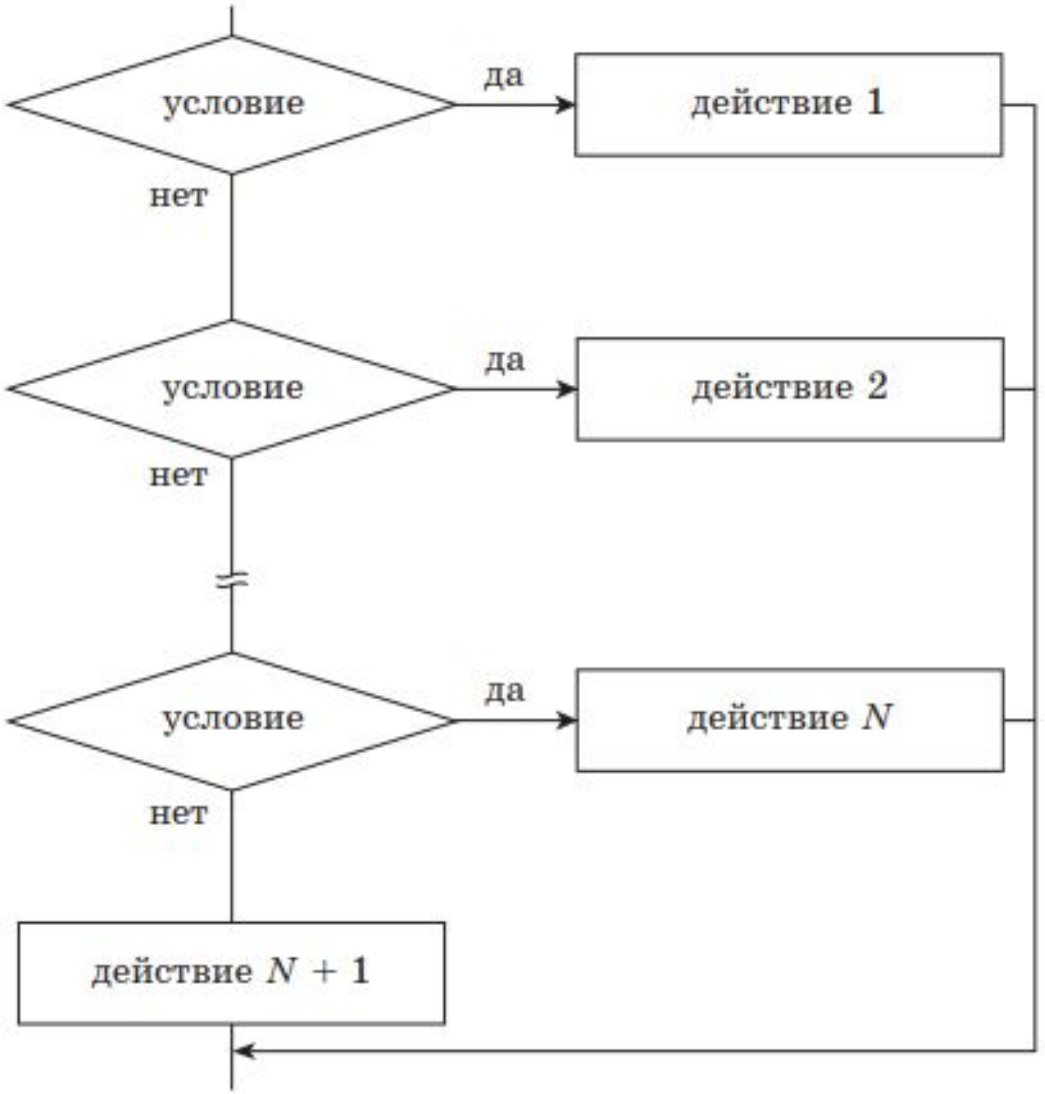
при условие 2: действие 2

...

при условие N: действие N + 1

иначе действия N + 1

всё

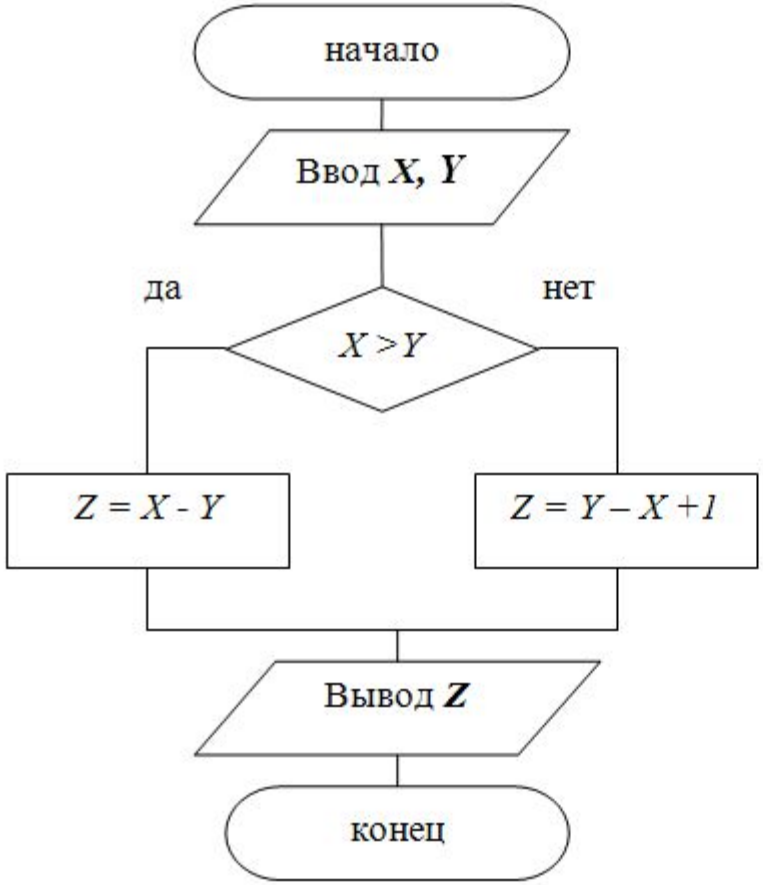


При разработке разветвляющегося алгоритма необходимо учитывать:

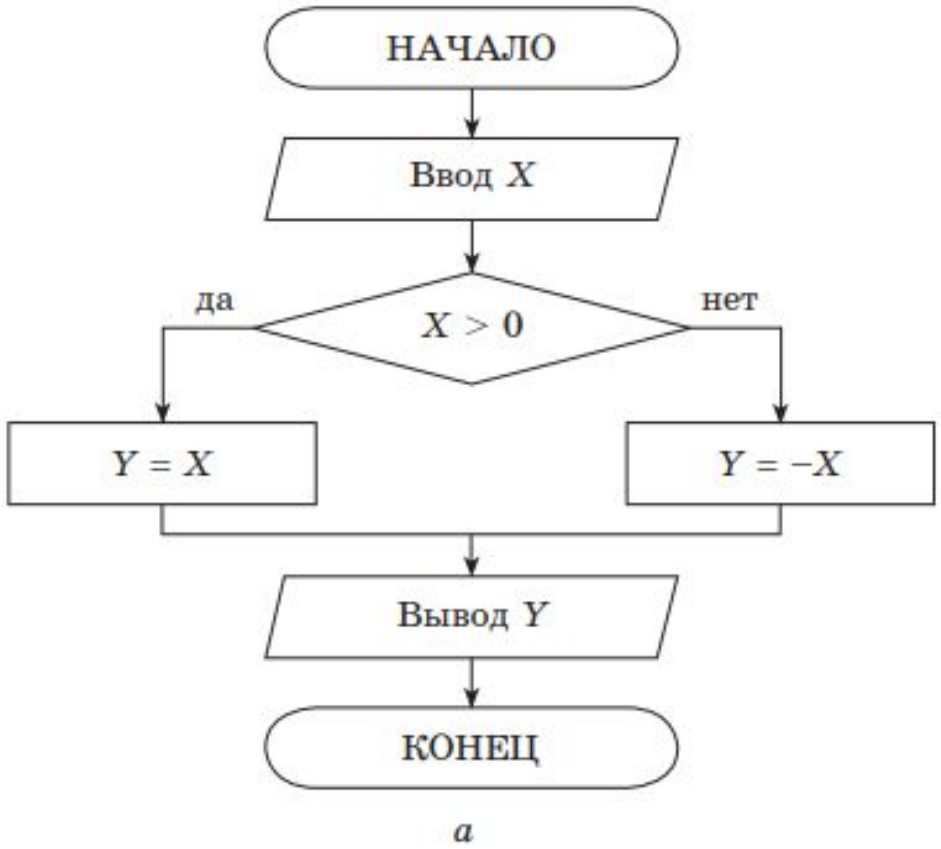
- что данный вид алгоритма применяется при наличии операций условного перехода;
- он чаще используется для вычислений функций, заданных несколькими арифметическими выражениями (формулами);
- инструкции в нем выполняются в зависимости от значения условия.

Пример: Разработать блок-схему алгоритма вычисления ***Z*** , если даны два действительных числа ***x*** и ***y*** .

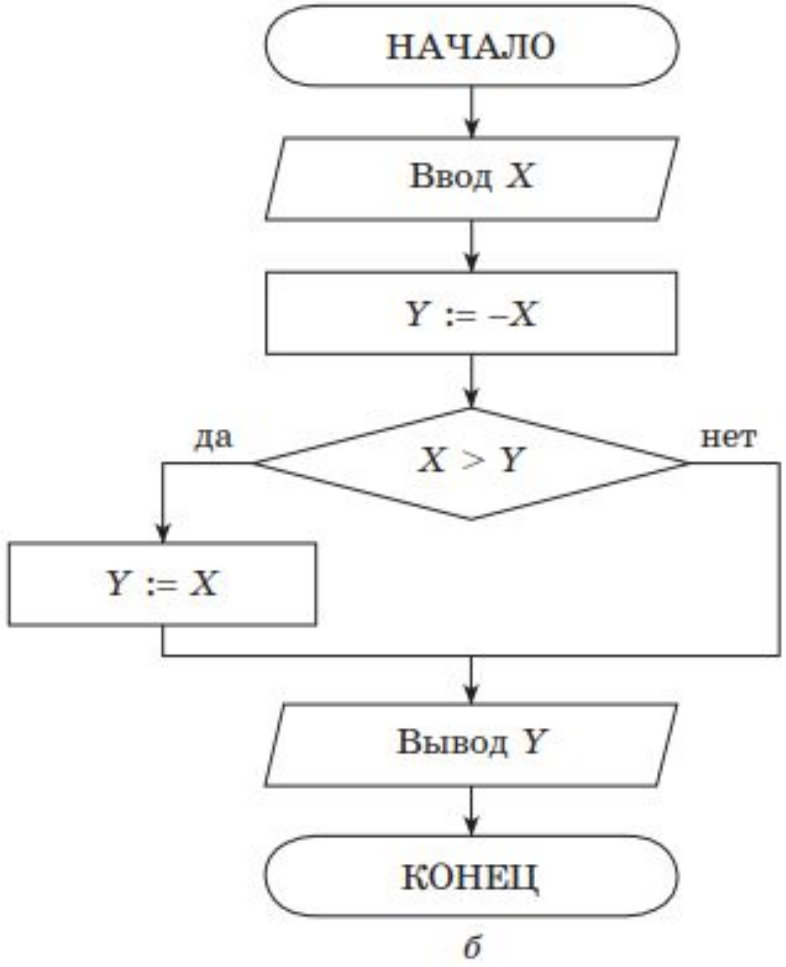
$$Z = \begin{cases} X - Y, & \text{если } X > Y \\ Y - X + 1, & \text{если } X <= Y \end{cases}$$



Пример: Нахождение модуля числа y(x) = |x|. Решение для случаев полной (а) и неполной (б) структур ветвления.



а



б

Пример: (найдите ошибку)

Пример решения квадратного уравнения $ax^2+bx+c=0$