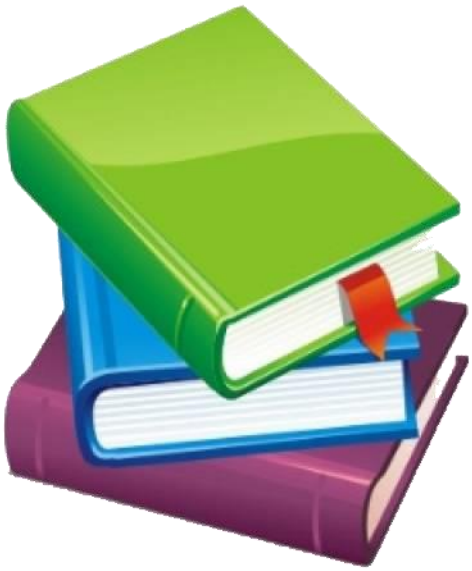
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Волгоградский государственный социально-педагогический университет»

ГЛОССАРИЙ НА ТЕМУ
«АЛГОРИТМЫ»

студентка I курса группы МИБ-12
Мухина Жанна



Волгоград
2013 г

Оглавление

Источники: Информатика. Базовый курс. 7-9 кл. Н. В. Макарова.
Информатика. Базовый курс. 9 кл. Н.Угринович.

Алгоритм

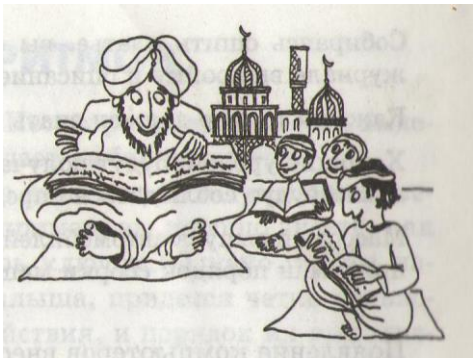
Алгоритм – описание последовательности действий (план), строе исполнение которых приводит к решению поставленной задачи за конечное число шагов.

Алгоритмизация – процесс разработки алгоритма (плана действий) для решения задачи

Понятие алгоритма можно рассмотреть с помощью следующего примера. Предположим, вы хотите вылепить из пластилина дракона. Результат во многом будет зависеть от вашего умения и опыта. Однако достичь поставленной цели окажется гораздо легче, если вы предварительно наметите план действий, например следующий:

1. Изучить образ дракона по имеющейся картинке.
2. Вылепить голову.
3. Вылепить туловище.
4. Вылепить хвост.
5. Вылепить четыре ноги.
6. Сравнивая с картинкой, уточнить детали каждой вылепленной части дракона.

Следуя подготовленному плану, любой человек, даже не обладающий художественными способностями, но имеющий терпение, обязательно получит хороший результат.



Появление алгоритмов связывают с зарождением математики. Более тысячи лет назад (в 825 году) ученый Абдулла аль-Хорезми создал книгу по математики, в которой описал способы выполнения арифметических действий над многозначными числами. Само слово

«алгоритм» возникло в Европе после перевода на латынь книги этого узбекского математика, в которой его имя писалось как «Алгоритми».

Область математики, известная как теория алгоритмов, посвященная исследованию свойств, способов записи, видов и сферы применения различных алгоритмов, созданию новых алгоритмов. Научное определение понятия алгоритма дал А. Черч в 1930 году.

Появление компьютеров внесло свою лепту в теорию алгоритмов. При работе на компьютере важно знать и понимать, что такое алгоритм и для чего он нужен. Прежде чем поручить компьютеру выполнение определенной работы, следует составить для него план действий – алгоритм. В нем необходимо предусмотреть порядок ввода и преобразования исходных данных, а также очередность и форму вывода результата.

Разрабатывая алгоритм, вы всегда будете проходить минимум *две стадии* – сначала алгоритм должен быть понятен тому, кто его разрабатывает, а затем его следует преобразовать с учетом специфики среды.

Первая стадия – алгоритм должен быть представлен в форме, понятной человеку, который его разрабатывает.

Вторая стадия – алгоритм должен быть представлен в форме, понятной тому объекту (в том числе и человеку), который будет выполнять описанные в алгоритме действия. В том случае, если эти действия станет выполнять сам разработчик алгоритма, вторая стадия будет отсутствовать.

Запомните правила разработки любого алгоритма.

Первая стадия – разработка приближенного алгоритма, ориентированного на создающего его человека:

- определить цель, для достижения которой будет создан алгоритм;
- наметить приблизительный план действий для достижения поставленной цели.

Вторая стадия – детализация алгоритма с учетом специфики среды и других объектов:

- выбрать среду и объекты, посредством которых алгоритм будет реализован;
- детализировать алгоритм с учетом особенностей выбранной среды.

Вернуться к оглавлению →

Детерминированность

Детерминированность. Это свойство указывает, что любое действие алгоритма должно быть строго и недвусмысленно определено в каждом случае.

Например, при управлении самолетом используются сложные алгоритмы, исполнителями которых являются пилот или бортовой компьютер. Последовательность выполнения действий, например, при взлете должна быть строго определенной (например, нельзя отрываться от взлетной полосы, пока самолет не набрал взлетную скорость). Исполнитель алгоритма, выполнив очередную команду, должен точно знать, какую команду необходимо исполнять следующей.

Вернуться к оглавлению →

Дискретность

Дискретность - это свойство алгоритма, указывающий он должен состоять из конкретных действий, следующих в определенном порядке. Если переставить действия в алгоритме, то он не может быть выполнен.

Рассмотрим алгоритм открытия двери ключом.

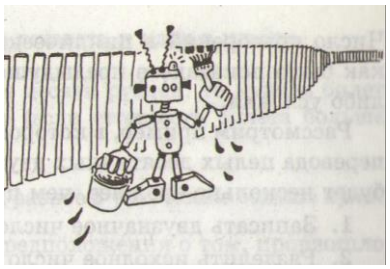
1. Достать ключ из кармана.
2. Вставить ключ в замочную скважину.
3. Повернуть ключ два раза против часовой стрелки.
4. Вынуть ключ.

В приведенном выше алгоритме, как и в любом алгоритме, необходимым является строгое соблюдение последовательности выполнения действий. Попробуем переставить второе и третье действие. Вы, конечно, сможете выполнить и этот алгоритм, но дверь вряд ли откроется. Поэтому алгоритм в целом является не выполнимым.

Вернуться к оглавлению →

Конечность

Конечность. Это свойство определяет, что каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения. Алгоритм должен иметь предел, т.е. быть конечным.



Пусть робот обучен красить забор. Для работы составлен алгоритм:

1. Покрасить доску.
2. Переместиться к следующей доске.
3. Перейти к действию 1.

Робот, закрасив одну доску, перейдет ко второй, затем к следующей и т.д. Робот не может закончить работу, так как алгоритм не предусматривает окончания работ. Поэтому алгоритм необходимо составить так, чтобы работы робота было завершение:

1. Покрасить доску.
2. Если есть еще доска, переместиться к следующей; перейти к действию 1.
3. Если доски закончились, завершить работу.

Вернуться к оглавлению →

Массовость

Массовость. Это свойство показывает, что один и тот же алгоритм можно использовать с различными исходными данными.

Рассмотрим алгоритм нахождения корней квадратного уравнения $ax+bx+c=0$.

1. Найдем дискриминант по формуле $D=b^2 - 4*a*c$.
2. Определим количество корней.
    - Если D>0, то уравнение имеет два разных корня.
    - Если D=0, то уравнение имеет два одинаковых корня.
    - Если D<0, то уравнение не имеет корней.
3. Найдем корни уравнения по формуле $x = \frac{-b \pm \sqrt{D}}{2a}$.

Таким образом, данный алгоритм можно применять к любому квадратному уравнению, вне зависимости от того, чему равны a,b,c.

Вернуться к оглавлению →

Результативность

Результативность. Это свойство требует, чтобы в алгоритме не было ошибок, и был обязательный конечный результат.

Рассмотрим алгоритм нахождения большего из двух заданных чисел A и B:

1. Из числа A вычесть число B.
2. Если получилось отрицательное значение, от сообщить, что число B больше.
3. Если получилось положительное значение, то сообщить, что число A больше.

При всей простоте и очевидности алгоритма, не каждый сразу поймет его ошибочность. Ведь если оба числа равны, то не получится никого сообщения. Значит, надо обязательно предусмотреть этот вариант, например:

1. Из числа A вычесть число B.
2. Если получилось отрицательное значение, от сообщить, что число B больше.
3. Если получилось положительное значение, то сообщить, что число A больше.
4. Если получился ноль, то сообщить, что числа равны.

Вернуться к оглавлению →

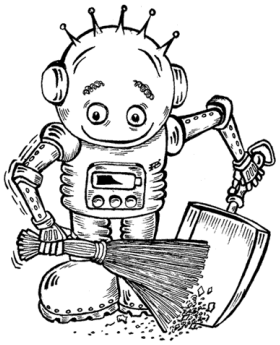
Исполнитель алгоритма

Действия, описываемые в алгоритме, прежде всего, должны быть понятны самому разработчику алгоритма. Только тогда алгоритм можно будет преобразовать в форму, понятную тому, кто его будет осуществлять.

Объект, который выполняет разработанный человеком алгоритм, называю ***исполнителем.*** Его предназначение – точно выполнять предписания алгоритма, не задумываясь о результатах и целях.

Каждый исполнитель обладает определенным набором команд, который называют ***системой команд исполнителя (СКИ)***. Алгоритм должен быть понятен исполнителю, т.е. должен содержать только те команды, которые входят в систему команд исполнителя.

Исполнитель не обязан понимать цели и методы достижения этой цели, пропускать действия или менять их местами по своему усмотрению, искать какую-то замену, если действие выполнить невозможно. Идеальными исполнителями являются машины, роботы, компьютеры. Они в состоянии выполнять указанные команды, не обсуждая целесообразность.



Вернуться к оглавлению →

Линейный алгоритм

Линейный (последовательный) алгоритм – описание действий, которые выполняются однократно в заданном порядке.

Предположим, требуется составить алгоритм вычисления результата выражения:

100+15 – 40+20

1. Сложить числа 100 и 15.
2. Из полученной суммы вычесть 40.
3. К результату 20.

В этом примере действия выполняются в том порядке, в котором записаны, т.е. последовательно, в линейном порядке.

Линейными алгоритмами являются алгоритмы отпирания дверей, заваривания чая, приготовления одного бутерброда. Линейный алгоритм применяется при вычислении арифметического выражения, если в нем используются только сложение и вычитание.

Вернуться к оглавлению →

Разветвляющийся алгоритм

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

Ситуации, заставляющие нас принимать решение в зависимости от некоторого условия, постоянно встречаются в повседневной жизни.

Если пошел дождь, то надо открыть зонт.

Если болит горло, то прогулку следует отменить.

Если прозвенел будильник, то надо вставать и идти в школу.

Эти предложения начинаются вопроса о том, произошло или нет некоторое событие: пошел ли дождь, болит ли горло, прозвенел ли будильник. Приведенные примеры отражают суть нашего мышления. Делая какие-то предположения, мы неизбежно приходим к определенным выводам.

Условие, под которым понимается предположение, начинающееся со слова *«если»* и заканчивающееся словом *«то»*. Условие может принимать значение *«истина»*, когда оно выполнено, или *«ложь»*, когда оно не выполнено. От значения условия зависит наше дальнейшее поведение.

Например, в предложении «Если пошел дождь, то надо открыть зонт» условие «пошел дождь» может быть и истинным, и ложным. Поэтому в конкретной ситуации предполагается либо выполнение действия «открыть зонт», либо его пропуск – если нет дождя, то зонтик открыть незачем.

Условие – выражение, находящееся между словом «если» и словом «то» и принимающее значение «истина» или «ложь».

В общем случае схема разветвляющего алгоритма будет выглядеть так: *«если условие, то…, иначе…»*. Такое представление алгоритма получило название ***полной формы.***

Вспомните кота из сказки А. С. Пушкина «Идет направо – песнь заводит, налево – сказку говорит…».

В разветвляющемся алгоритме при невыполнении условия действия могут не предусматриваться. Тогда это будет неполная форма, в которой действия пропускаются: *«если условие, то…»*. Неполная форма разветвляющегося алгоритма напоминает поведение водителя, едущего по шоссе: если бензин на исходе, то водитель заезжает на ближайшую АЗС.

Вернуться к оглавлению →

Циклический алгоритм

Циклический алгоритм – описание действий, которые должны повторяться указанное число раз или пока не выполнится заданное условие. Перечень повторяющихся действий называется телом цикла. Циклы могут быть: *с параметром* (со счетчиком), *с предусловием* (когда условие идет раньше тела цикла, т.е. если выполняется условие, выполняется цикл), *с постусловием* (когда условие стоит после цикла, т.е. цикл выполняется до тех пор, пока не выполнится условие).

Многие процессы в окружающем мире основаны многократном повторении одной и той же последовательности действий. Каждый год наступает весна, лето, осень и зима. Жизнь растений в течение года проходит одни и те же циклы. Подсчитывая число полных поворотов минутной или часовой стрелки, человек измеряет время.

Допустим, робот обучен красить забор. Он последовательно закрашивает доску за доской. Для робота составлен следующий алгоритм:

4. Покрасить доску.
5. Переместиться к следующей доске.
6. Перейти к действию 1.

Робот, закрасив одну доску, перейдет ко второй, затем к следующей и т.д. Робот не может закончить работу, так как алгоритм не предусматривает окончания работ. В приведенном примере необходимо добавить в алгоритм действие по анализу результата:

4. Покрасить доску.
5. Если есть еще доска, переместиться к следующей; перейти к действию 1.
6. Если доски закончились, завершить работу.

Вернуться к оглавлению →

Вспомогательный алгоритм

Вспомогательный алгоритм – алгоритм, который можно использовать в других алгоритмах, указав только его имя. Вспомогательному алгоритму должно быть присвоено имя.

Допустим, вы хотите научиться жонглировать двумя или даже тремя мячами. Если внимательно приглядеться к действиям профессионального артиста и попытаться понять, как это ему удается делать, то оказывается – секрет в том, что надо научиться искусно выполнять несколько определенных движений, которым присвоим соответствующие названия:

Бросок левой – подбросить мяч левой рукой**.**

Бросок правой – подбрось мяч правой рукой.

Захват левой – поймать мяч левой рукой.

Захват правой – поймать мяч правой рукой.

Выполняться каждое такое действие будет по собственному алгоритму. Научившись таким действиям, вы сможете применить свое умение и в другом деле, например, показывая фокусы или участвуя в соревнованиях.

Алгоритм жонглирования можно записать с помощью вспомогательных алгоритмов выполнения отдельных действий в следующем виде:

1. Когда летящий шарик начинает поворачивать в правой руке, выполнить **Бросок правой** и **Захват правой**
2. Когда летящий шарик начинает поворачивать в левой руке, выполнить **Бросок левой** и **Захват левой.**

Вернуться к оглавлению →

Словестный способ представления алгоритма

Словесный способ. Способ описания алгоритма на естественном языке. Он очень удобен, когда следует приближенно описать суть алгоритма. Например, алгоритм открытия двери.

1) Достать ключ из кармана.

2) Вставить ключ в замочную скважину.

3) Повернуть ключ два раза против часовой стрелки.

4) Вынуть ключ.

Вернуться к оглавлению →

Графический способ представления алгоритма

Графический способ (блок-схема). Для более наглядного представления алгоритма широко используется графическая форма – блок-схема, которая составляется из стандартных графических объектов:

 - начало и конец алгоритма

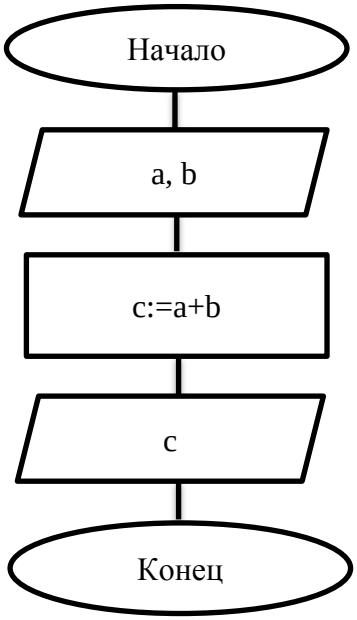
 - ввод и вывод данных

 - выполнение действий

 - условие

 - последовательность выполнения действий

Пример:



Вернуться к оглавлению →

Табличный способ представления алгоритма

Табличный способ. Например, расписание занятий.

№	Время	Название предмета
1	8:00-8:45	Математика
2	8:55-9:40	Информатика
3	10:00-10:45	Русский язык
4	11:05-11:50	Литература

Вернуться к оглавлению →

Способ представления алгоритма на алгоритмическом языке.

Алгоритмический язык (псевдокод). Осуществляется с помощью специальных слов: АЛГ – название алгоритма, АРГ – аргументы, т.е. переменные, РЕЗ – результат, т.е. переменная результата, НАЧ и КОН – начало и конец алгоритма.

Пример:

```
АЛГ сложение
АРГ a, b
РЕЗ c
НАЧ
    c:=a+b
КОН
```

Вернуться к оглавлению →

Способ представления алгоритма на языке программирования

Язык программирования.

Пример (Pascal):

```
program slozhenie;
var  a,b,c:integer;
begin
     Readln (a,b);
     c:=a+b;
     Write(‘c = ’, c);
end.
```

Вернуться к оглавлению →