ТУБ №3:
Словесное описание алгоритмов

Как Вы уже знаете, естественные языки с их нестрогими правилами не слишком подходят для описания алгоритмов. Тем не менее, словесное описание алгоритмов находит широкое применение, если значительно ограничить набор используемых слов и выражений.

Единых общепринятых правил для словесного описания не существует (оно и понятно, весь мир на разных языках разговаривает), поэтому попробуем придумать свой набор правил, а заодно посмотрим, какие подводные камни есть на пути к своему собственному способу описания алгоритмов.

Поскольку любой алгоритм представляет собой описание этапов, образующих решение задачи, разработка своих правил словесного описания заключается в том, чтобы понять, какие этапы могут потребоваться, и предложить способ такой их записи, который не будет допускать разночтений. Но для начала давайте введём несколько полезных понятий.

Программы используются для обработки данных, а эти данные нужно где-то хранить (то есть куда-то записывать, чтобы потом оттуда прочитать и использовать). При описании алгоритма нам придётся давать имена отдельным величинам, участвующим в решении задачи. При этом некоторые из них могут в ходе решения изменять свои значения, а некоторые будут оставаться неизменными.

> Переменная — элемент данных, который может изменять своё значение в ходе решения задачи.
>
> Константа — элемент данных, имеющий фиксированное значение в ходе решения задачи.

Обратите внимание, что эти понятия в программировании несколько отличаются от аналогичных понятий в математике. В математике переменная обычно имеет одно и то же значение на протяжении всего решения задачи. Скажем, при решении уравнения

> x + 5 = 8

подразумевается, что x равен 3 и это значение не изменяется по ходу решения задачи. В уравнении

> x2 + 5x + 6 = 0

переменная x может принимать одно из двух значений, но эти значения тоже фиксированные в рамках конкретной задачи. В программировании же переменная может изменять своё значение произвольно. В каком-то смысле переменные в программировании даже более переменны, чем в математике.

Похожая история и с константами. В математике константами обычно называют величины, которые принципиально не могут иметь других значений: например, π всегда равно 3.141592…, а e всегда равно 2.71828… В программировании же константа — это элемент данных (величина), которая имеет определённое значение только в рамках конкретной задачи, в другой задаче её значение может быть принято совсем другим.

В компьютере для хранения значений переменных отводятся области ОЗУ достаточного размера. Запись в такую область памяти называется присваиванием. При обсуждении алгоритма под присваиванием понимают действие, суть которого в том, чтобы начиная с определённого этапа вычислений считать, что переменная равна тому или иному (чаще всего новому) значению.

1) Этап обработки (вычисления)

Иногда действительно нужно просто что-то посчитать или выполнить какое-то элементарное действие. В компьютерах же вообще любые действия можно свести к вычислениям и копированию данных, поэтому для обозначения этапа обработки обычно бывает достаточно указать выражение, значение которого нужно вычислить, и переменную, в которую следует записать результат вычислений.

Мы уже сказали, что запись в переменную ещё называется присваиванием. Для обозначения присваивания был придумал специальный символ присваивания. Он выглядит так:

:=

С его использованием, например, этап алгоритма, отвечающий за вычисление дискриминанта квадратного уравнения, можно записать так:

$$D := b^2 - 4ac$$

Некоторые недальновидные программисты предпочитают использовать для обозначения присваивания просто знак равенства, т.е. пишут так:

$$D = b^2 - 4ac$$

но такое обозначение не совсем корректно, т.к. символ присваивания обозначает вычисление значения выражения в своей правой части и запись результата в левую, тогда как знак равенства используется для обозначения соотношения между значениями двух выражений. Особенно заметна эта проблема, если мы хотим записать, например, увеличение значения переменной x на 5:

$$x = x + 5$$

Любой математик скажет Вам, что перед нами уравнение относительно переменной x. Причём уравнение, которое не имеет решений, т.к. нет такого числа, которое после увеличения на 5 было бы равно самому себе. Поэтому для грамотного человека такая запись выглядит очень странно. Проблема полностью устраняется использованием символа присваивания:

$$x := x + 5$$

Такую запись уже нельзя спутать с уравнением или попыткой сравнить две величины.

2) Проверка условия

Часто ход вычислений изменяется в зависимости от получаемых промежуточных результатов. Например, если при решении квадратного уравнения получен отрицательный дискриминант, как правило, говорят об отсутствии корней, в противном же случае переходят к их вычислению.

В таких случаях договоримся использовать следующее обозначение:

Если условие, идти к N

Вместо слова «условие» будем записывать выражение, по которому принимается решение, а вместо «N» будем указывать, к какому этапу алгоритма нужно перейти.

3) Безусловный переход

Иногда бывает нужно указать, что дальнейшее решение задачи следует продолжить с определённого этапа, который уже записан в другой части словесного описания алгоритма, причём в отличие от предыдущего случая переход должен произойти всегда, независимо от каких бы то ни было условий. В этом случае будем использовать такое обозначение:

Идти к N

4) Конец вычислений

Как Вы помните, среди свойств правильного алгоритма есть такие, как результативность и определённость. Первое означает, что алгоритм должен однажды завершиться получением искомого результата, а второе — что на каждом шаге мы должны чётко понимать, куда двигаться дальше.

Для обеспечения этих свойств алгоритмов необходимо иметь обозначение для специального этапа, соответствующего окончанию вычислений. Будем обозначать его так:

Останов.

Приведённых обозначений будет достаточно для записи алгоритма любой сложности. Это можно даже доказать строго математически, но мы этого сейчас делать не будем, придётся поверить на слово. Для удобства договоримся также нумеровать этапы.

Попробуем с использованием придуманных нами обозначений описать алгоритм решения квадратного уравнения $ax^2 + bx + c = 0$:

> 1) Ввод a, b, c.
> 2) $D := b^2 - 4ac$.
> 3) Если $D < 0$, идти к 7.
> 4) $x_1 := \dfrac{-b + \sqrt{D}}{2a}$, $x_2 := \dfrac{-b - \sqrt{D}}{2a}$.
> 5) Вывод x_1, x_2.
> 6) Идти к 8.
> 7) Вывод «Нет корней».
> 8) Останов.

Главный недостаток словесного описания алгоритмов — его малая наглядность. Когда алгоритм становится достаточно сложным, количество переходов (как условных, так и безусловных) резко возрастает и само описание становится достаточно запутанным.

Вместе с тем безусловное преимущество заключается в том, что этот способ можно использовать при обсуждении алгоритма решения задачи с заказчиком: в отличие от других способов он не требует специальной подготовки, а используемые обозначения понятны без особых пояснений.