ТУБ №9:
Циклические вычислительные процессы

Виды циклов

Давным-давно, когда вечером пятницы попить чаю с чизкейками и посмотреть «Поле чудес» собирались не люди, а динозавры и мамонты, жил да был программист по имени Эдик. Сейчас уже никто толком и не помнит, на каком языке программирования он писал программы — то ли на Pascal'е, то ли на JavaScript'е, — но все сходятся во мнении, что на работе.

И вот на эту самую работу он однажды не пришёл. А потом не пришёл и на следующий день. И на следующий. Коллеги его, конечно же, стали за него волноваться. А в те далёкие времена не было ещё не то что телефонов — даже голубиной почты не было, поэтому стали звонить ему в Viber. Набирают его раз, другой, третий — нет ответа.

Приехали к нему домой, позвонили в дверь к соседям. «Нет ли, — спрашивают, — у вас ключа запасного от квартиры Эдика?» А все ведь знают, что сосед Эдик — это всегда хороший соседик: и о здоровье справится, и о погоде разговор поддержит, и за морской свинкой присмотрит, если Вы в отъезде. Жил наш Эдик с соседями дружно и ключ запасной им, конечно же, оставил.

Зашли коллеги к Эдику в квартиру — вроде никого дома нет. Прислушались — а в ванной вода журчит. Подёргали дверь — закрыто. Ну, делать нечего, стали ломать. Выломали дверь, а там сидит Эдик в ванне, весь в мыльной пене, от холода трясётся, в руках бутылка шампуня, а на ней инструкция напечатана:

«Нанести небольшое количество шампуня на влажные волосы, помассировать 2–3 минуты, смыть, повторить».

Тут и сказочке конец, а кто слушал — пусть читает ещё раз.

Критично настроенный читатель скажет, что не все Эдики такие уж хорошие соседики, и, пожалуй, будет прав, но дело здесь, конечно, не в Эдиках, а в том, что приведённая выше инструкция по использованию шампуня — это как раз пример циклического вычислительного процесса (или, проще говоря, циклического алгоритма), а именно о таких вычислительных процессах мы сейчас и поговорим.

Без циклов не обходится ни одна хоть сколько-нибудь сложная программа. Но не бывает так, чтобы один и тот же инструмент подходил на все случаи жизни, поэтому и циклы тоже бывают разными.

По взаимному расположению выделяют:

* простые циклы;
* сложные циклы.

Сложным называется цикл, внутри которого содержится другой цикл, или даже несколько. Соответственно простой цикл — такой цикл, внутри которого других циклов нет.

Если перед нами сложный цикл, то где-то там есть ещё одна классификация. В этом случае по взаимному расположению циклы также могут быть:

* вложенные (внутренние);
* внешние.

Здесь всё предсказуемо: вложенный цикл — тот, который входит в состав других циклов, внешний — тот, который содержит внутри себя другие циклы, но сам в другие циклы не входящий.

Применительно к циклам используются такие понятия, как тело цикла и итерация.

> Тело цикла — часть цикла, многократно выполняющаяся последовательность действий.
>
> Итерация — одно из повторений цикла.

Главное требование к любому циклу — это наличие в нём некоторого условия, в зависимости от которого цикл либо продолжает выполняться, либо завершается. Отсутствие такого условия в инструкции к шампуню Эдика как раз и стало причиной неловкой ситуации. Если же сформулировать это более строго, то алгоритм, заданный в форме инструкции, оказался неправильным: у него отсутствовало свойство результативности.

В зависимости от местоположения условия циклы бывают:
* с предусловием;
* с постусловием.

> Цикл с предусловием — цикл, в котором проверка условия цикла осуществляется до выполнения тела цикла.
>
> Цикл с постусловием — цикл, в котором проверка условия цикла осуществляется после выполнения тела цикла.

Важное отличие между циклами с предусловием и с постусловием заключается в том, какой смысл несёт в себе условие цикла: цикл с предусловием завершается, когда условие цикла оказывается ложным, а цикл с постусловием — когда условие оказывается истинным. Об этом ещё говорят так: в цикле с предусловием используется условие входа в цикл, а в цикле с постусловием — условие выхода из цикла.

В зависимости от вида условия выделяют циклы:
* с параметром;
* итерационные.

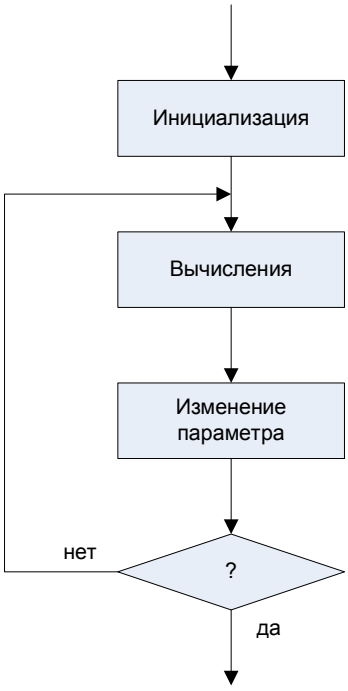
Циклы с параметром

Циклы с параметром — самый часто используемый вид циклов. По-другому они ещё могут называться «цикл со счётчиком» или «циклический процесс с известным числом повторений».

Как и в любом другом цикле, в цикле с параметром можно выделить одну или несколько переменных, от которых зависит, будет ли продолжаться выполнение цикла. Такие переменные в цикле с параметром ещё называются параметрами цикла. Отличительной особенностью цикла с параметром является то, что здесь заранее известно количество повторений цикла.

Когда речь идёт о заранее известном количестве повторений, это необязательно должно быть конкретное число. Скажем, если Ваш алгоритм вычисляет средний балл студентов учебной группы по итогам сессии и количество студентов является для алгоритма исходными данными, то назвать точное количество повторений будет проблематично. Но мы точно можем сказать, что для численности группы в N человек количество повторений будет равно N. Иногда соотношение между исходными данными и количеством повторений оказывается несколько сложнее, но если перед Вами цикл с параметром, то оно по крайней мере поддаётся какой-то оценке.

Рассмотрим обобщённую схему циклического алгоритма с параметром:



При построении любого цикла с параметром следует следить за наличием в нём четырёх составляющих.

Во-первых, редкий цикл обходится без того, чтобы перед его началом не потребовалась инициализация. Инициализацией называют задание переменным, участвующим в вычислениях, некоторых начальных значений. Сама инициализация не входит в тело цикла, т.е. ту часть, которая будет повторяться, однако без неё значения переменных на первой итерации цикла (т.е. при первом выполнении его тела) могут оказаться не теми, которые нужны для получения правильного результата.

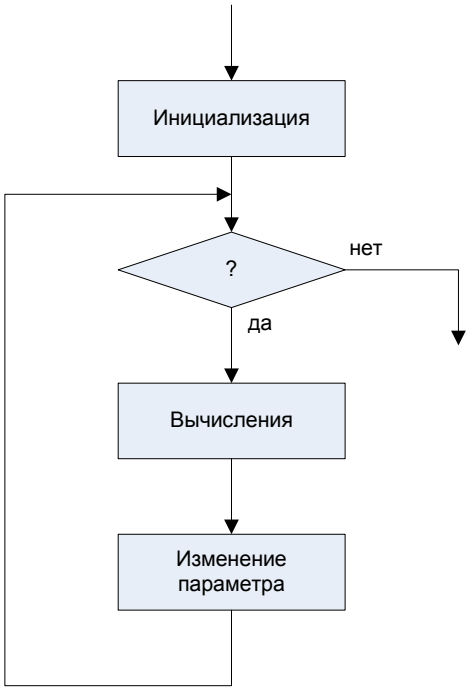
Вторая составляющая — полезные вычисления, те самые, ради которых цикл вообще-то и задумывался.

Третья составляющая — изменение параметра цикла. Эта часть цикла нужна по двум причинам. Во-первых, чаще всего значение параметра цикла является исходными данными для тех вычислений, которые происходят в теле цикла. Во-вторых, в циклах с параметром именно от значения параметра зависит, когда цикл перестанет выполняться. Получается, если бы значение параметра цикла не изменялось, мы могли бы получить бесконечный цикл, а Эдик бы такое не одобрил.

Наконец четвёртая составляющая — условие цикла. В циклах с параметром условие опирается на значение параметра цикла.

В зависимости от решаемой задачи некоторые части цикла с параметром могут отсутствовать или объединяться. Например, если в результате вычислений, которые выполнялись до цикла с параметром, все переменные уже получили необходимые начальные значений, может быть так, что дополнительной инициализации не потребуется. В некоторых случаях изменение параметра цикла происходит как часть полезных вычислений. Тем не менее, при построении любого цикла следует ответить для себя на вопрос о том, где в нём реализована каждая из вышеназванных частей.

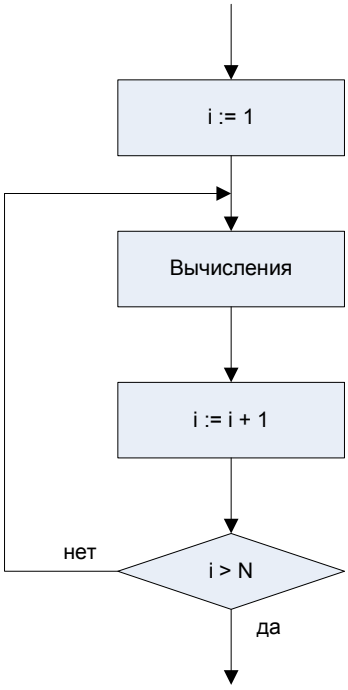
Как Вы помните, помимо разделения на циклы с параметром и итерационные циклы существует также отдельная независимая классификация на циклы с предусловием и с постусловием. В приведённом примере был использован цикл с постусловием: в нём условие цикла проверялось после выполнения тела цикла. Но цикл с параметром может быть реализован и как цикл с предусловием:



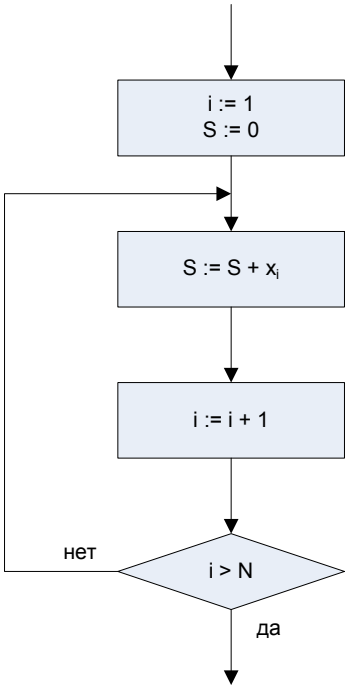
Рассмотрим циклы с параметром на примере несложной задачи.

Дан набор из N чисел x_i. Найти $S = \sum_{i=1}^{N} x_i$.

В данном случае перед нами пример типичного цикла с параметром: количество повторений заранее известно и равно количеству слагаемых, участвующих в вычислениях. Чтобы организовать цикл с правильным количеством повторений, используем переменную i следующим образом:

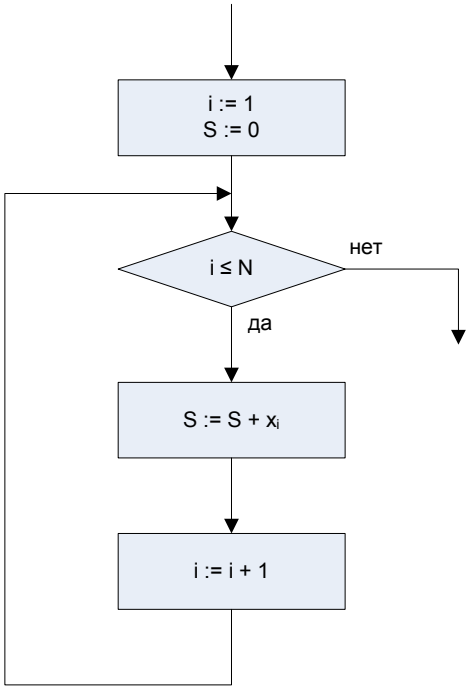


Обратите внимание, что в таком виде этот цикл может быть использован для любой задачи, в которой требуется цикл на N итераций. По сути, мы только что реализовали техническую, вспомогательную логику работы этого цикла. Теперь остаётся только дополнить его вычислениями, которые нужны для решения конкретной задачи.



Вычисление суммы осуществляется её накоплением: на каждой итерации к уже накопленной сумме прибавляется значение очередного слагаемого. Инициализация до начала цикла дополнилась обнулением переменной S, т.к. к этому моменту не было просмотрено ни одного слагаемого, а значит, накопленная сумма пока равна нулю.

Тот же самый алгоритм можно было реализовать и в виде цикла с предусловием:



При замене цикла с постусловием на цикл с предусловием изменения оказались незначительными: проверка условия переместилась к началу тела цикла, а само условие заменилось на противоположное.

Циклы с предусловием и с постусловием абсолютно взаимозаменяемы. Выбор между ними осуществляется исходя из удобства использования цикла того или иного вида. Так, например, нетрудно заметить, что цикл с постусловием выполняется как минимум один раз, тогда как цикл с предусловием может не выполниться ни разу.

Оценим полученные решения в точки зрения циклов с параметром. В обоих случаях таким параметром была переменная i. От того, каким образом она изменяла своё значение, зависело число повторений.

Итерационные циклы

Предположим, Вам встретилась следующая задача.

Дано число x. Найти $y = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

Может показаться, что мы дошли до того случая, когда алгоритмы бессильны. Действительно, ведь одно из свойств правильного алгоритма — результативность — требует, чтобы выполнение алгоритма завершалось за конечное число шагов, а в этой задаче перед нами бесконечная сумма, а бесконечная сумма будет вычисляться бесконечно долго. Но нет: именно в таких задачах приходят на помощь итерационные циклы.

Давайте возьмём какое-нибудь значение переменной x и проанализируем отдельные слагаемые нашей бесконечной суммы. Например, при $x = 1$ получится следующая сумма:

$$y = 1 + \frac{1}{1} + \frac{1 \cdot 1}{1 \cdot 2} + \frac{1 \cdot 1 \cdot 1}{1 \cdot 2 \cdot 3} + \frac{1 \cdot 1 \cdot 1 \cdot 1}{1 \cdot 2 \cdot 3 \cdot 4} + \ldots = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \ldots$$

Замечаете интересное свойство? Давайте попробуем с $x = 2$:

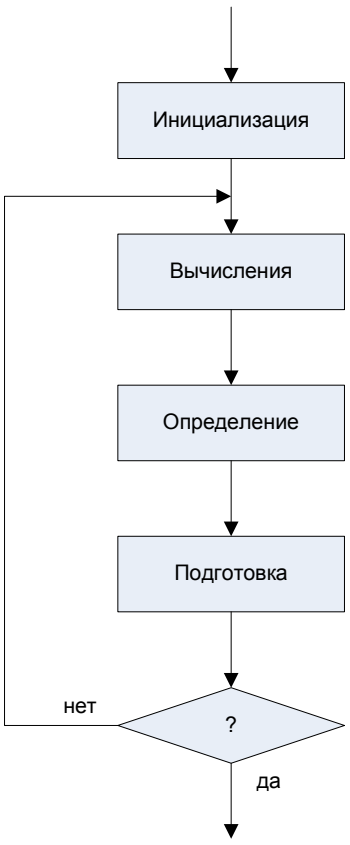
$$y = 1 + \frac{2}{1} + \frac{2 \cdot 2}{1 \cdot 2} + \frac{2 \cdot 2 \cdot 2}{1 \cdot 2 \cdot 3} + \frac{2 \cdot 2 \cdot 2 \cdot 2}{1 \cdot 2 \cdot 3 \cdot 4} + \ldots = 1 + 2 + 2 + \frac{8}{6} + \frac{16}{24} + \ldots$$

Ещё не заметили? Тогда давайте посмотрим, как связаны между собой соседние слагаемые. Самую первую единицу обозначим s_0. Тогда каждое последующее слагаемое можно получить из предыдущего по такой формуле:

$$s_k = s_{k-1} \cdot \frac{x}{k}, k > 0$$

Другими словами, каждое следующее слагаемое получается из предыдущего его домножением на x и делением на порядковый номер. Но ведь x у нас при вычислении суммы не меняется, а вот порядковый номер будет всё больше и больше. Значит, начиная с какого-то момента каждое следующее слагаемое начнёт получаться меньше предыдущего ($\frac{x}{k}$ станет меньше 1). А значит, какое-нибудь десятимиллионное по счёту слагаемое будет настолько малым, что перестанет оказывать на сумму хоть сколько-нибудь значимое влияние. Другими словами, начиная с какого-то момента всеми последующими слагаемыми можно будет просто пренебречь.

На этой идее основаны итерационные циклы. В общем виде итерационный цикл можно представить следующим образом:

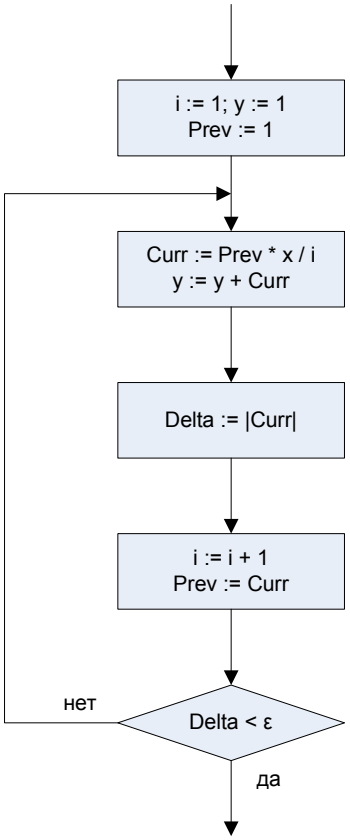


В блоке инициализации по-прежнему будут задаваться начальные значения величин, разве что их здесь обычно будет больше, чем в цикле с параметром. Как и в цикле с параметром, будут выполняться полезные вычисления.

Изменения коснутся того, как задаётся условие выхода из цикла. Для этого необходимо будет вычислить некоторую величину, которая будет характеризовать значимость полученного на текущей итерации промежуточного результата (на схеме этому соответствует блок «Определение»). Блок «Подготовка» обозначает действия, которые необходимо выполнить, чтобы на следующей итерации вычисления происходили с правильными значениями переменных.

Основное отличие от цикла с параметром будет заключаться в условии: здесь величина, вычисленная на этапе «Определение», будет сравниваться с требуемой по условию точностью. Результат сравнения и будет определять, нужно ли продолжать цикл: очевидно, продолжать имеет смысл только тогда, когда вклад очередной итерации в общий результат превышает некоторую достаточно малую величину (её обычно обозначают ε).

Схема алгоритма вычисления приведённой ранее бесконечной суммы будет иметь такой вид:



Переменные Prev и Curr используются для хранения значения предыдущего и текущего слагаемых. В блоке «Подготовка» значение слагаемого, полученное на текущей итерации, присваивается переменной Prev, чтобы на следующей итерации оно использовалось как значение предыдущего слагаемого (а оно к тому моменту именно предыдущим и будет). Каждое новое слагаемое получается по формуле, которую мы уже обсудили ранее.

Величина Delta, по значению которой принимается решение о продолжении или выходе из цикла, определяется как модуль очередного слагаемого. Дело в том, что значения слагаемых не обязаны быть положительным. Например, при отрицательных значениях x даже в нашей формуле знаки слагаемых будут чередоваться. Но вклад слагаемого в общую сумму не зависит от его знака: важен только его модуль.

Обратите внимание: здесь, как и в циклах с параметром, у нас была переменная, которая использовалась для подсчёта итераций. Ключевое отличие, делающее цикл итерационным, заключается в том, что здесь эта переменная-счётчик не участвует в организации цикла: от неё не зависит количество итераций, она нужна только для удобства вычислений.

Как видим, с итерационными циклами программистам подвластна даже бесконечность.