При решении подавляющего большинства задач (в том числе и весьма несложных) в программе практически невозможно задать в явном виде все операции, которые необходимо выполнить. В самом деле, пусть необходимо вычислить сумму первых n членов гармонического ряда:

Y= 1+ 1/2+ 1/3 + …+ 1/ n

Очевидно, что с использованием только рассмотренных выше типов операторов можно составить программу лишь для фиксированного значения n. Например, при n=5 требуемые вычисления можно задать с помощью оператора присваивания вида:

Y:= 1+1/2+1/3+1/4+1/5

Если же значение n не фиксируется, а является исходным данным, вводимым в процессе выполнения программы (и даже константой, описанной в программе), то аналогичный оператор присваивания записать невозможно. Ибо запись вида Y:= 1+1/2+1/3+…+1/ n в языках программирования недопустима.

Для устранения возникающих трудностей служат **операторы цикла**. Они позволяют повторять выполнение отдельных частей программы. Можно выделить четыре о**ператора цикла**, присутствующих в том или ином виде во всех языках программирования: **простой арифметический оператор цикла** (цикл с параметром с шагом 1), **сложный арифметический оператор цикла** (цикл с параметром произвольного шага), **итерационный оператор цикла с предусловием**, **итерационный оператор цикла с постусловием**.

Простой арифметический оператор цикла Паскаля (цикл с параметром)

Вернемся к рассмотренной выше задаче вычисления суммы первых n членов гармонического ряда, правила которой невозможно задать в виде арифметического выражения, если значение n заранее не фиксировано.

На самом деле вычисление этой суммы можно осуществить по очень простому и компактному алгоритму: предварительно положим y=0 (с помощью оператора присваивания y:=0), а затем выполним оператор присваивания y:= y+1/ i для последовательных значений i= 1,2,…, n. При каждом очередном выполнении этого оператора к текущему значению y будет прибавляться очередное слагаемое. Как видно, в этом случае процесс вычислений будет носить циклический характер: оператор **y:= y+1/i** должен выполняться многократно, т.е. циклически, при различных значениях i.

Этот пример циклического вычислительного процесса является весьма типичным; его характерные особенности состоят в том, что

* число повторений цикла известно к началу его выполнения (в данном случае оно равно значению n, которое предполагается заданным к этому времени);
* управление циклом осуществляется с помощью переменной порядкового типа, которая в этом циклическом процессе принимает последовательные значения от заданного начального до заданного конечного значений (в нашем случае – это целочисленная переменная i, принимающая последовательные значения от 1 до n).

Для компактного задания подобного рода вычислительных процессов и служит **оператор цикла с параметром**. Чаще всего используется следующий вид этого оператора В Паскале:

For V:= E1 to E2 do S,

где **for** (для), **to** (увеличиваясь к) и **do** (выполнять, делать) – служебные слова, **V** – переменная порядкового типа, называемая параметром цикла, **Е1** и **Е2** – выражения того же типа, что и параметр цикла, **S** – оператор, который и выполняется многократно в цикле, называемый телом цикла.

Заметим, что в Паскале после **do** должен стоять один оператор, если необходимо выполнить несколько действий, то они должны быть объединены в один составной оператор путем заключения в операторные скобки.

Этот оператор цикла Паскаля предусматривает присваивание параметру цикла V последовательных значений от начального значения, равного значению выражения Е1, до конечного значения, равного значению выражения Е2, т.е. при каждом повторении выполняется оператор присваивания **V:= succ(V)**, и выполнение оператора S при каждом значении параметра цикла V. При этом значения выражений Е1 и Е2 вычисляются один раз, при входе в оператор цикла, а значение параметра цикла V не должно изменяться в результате выполнения оператора S. Если заданное конечное значение меньше начального значения (что допустимо), то оператор S не выполняется ни разу.

В Паскале считается, что при нормальном завершении выполнения оператора цикла значение параметра цикла не определено.

С использованием **оператора цикла с параметром** алгоритм вычисления суммы первых n членов гармонического ряда может быть задан следующим образом:

Пример кода программы для суммирования первых n членов гармонического ряда

Readln(n);

Y:= 0;

For i:= 1 to n do y:= y+1/i;

В некоторых случаях бывает удобно, чтобы параметр цикла Паскаля принимал последовательные, но не возрастающие, а убывающие значения. Для таких случаев в Паскале предусмотрен оператор цикла с параметром следующего вида:

For V:= E1 downto E2 do S,

где **downto** (уменьшаясь к) – служебное слово, а все остальные слова и выражения имеют прежний смысл. Изменение параметра цикла от большего значения к меньшему происходит при выполнении присваивания V:=pred(V). Заметим, что начальное значение может быть меньше конечного значения. В этом случае оператор S не выполнится ни разу. Значение параметра цикла по завершении выполнения такого цикла так же считается неопределенным.

Следует запомнить и то, что для обоих вариантов записи **цикла с параметром** справедливо: если начальное и конечное значения равны, то тело цикла (оператор S) выполнится один раз.

Заметим так же, что **параметр цикла** может и не использоваться в теле цикла, так что основное его назначение – это управление числом повторений цикла. Например, значение y= x n, где n>=0 – целое, можно вычислить по следующему алгоритму: предварительно положить y=1, а затем n раз домножить это значение на x:

Пример кода программы цикла Паскаля

Readln(n);

Readln(x);

Y:= 1;

For i:= 1 to n do y:= y\*x;

Как видно, здесь параметр цикла i служит лишь для того, чтобы тело цикла (оператор y:= y\* x) выполнилось нужное число раз.

Арифметический оператор цикла Паскаля с произвольным шагом

Естественным усложнением простого арифметического цикла Паскаля, является цикл, в котором параметр цикла изменяется не на 1, а на произвольную величину – **шаг приращения**. При этом в процессе выполнения цикла шаг изменяется по заданному закону. Стандартные операторы для реализации такого цикла есть в Форте, в других языках их приходится организовывать из простейшего арифметического цикла.

Итерационные операторы цикла Паскаля

Итерационные циклы отличаются от циклов с параметром тем, что в них заранее неизвестно число повторений.

Пусть мы отправляемся за грибами и возвращаемся домой, когда корзина наполнится. Все грибники делятся на 2 категории:

* Смотрят, есть ли место в корзине, а уже потом срывают грибы, если их можно поместить в корзину. (Правда, в жизни таких грибников встречать не приходилось)
* Сначала срывают грибы, а уже потом думают, как их положить в корзину.

Отсюда получаются два варианта реализации итерационных циклов:

с предусловием и с постусловием.

В цикле с предусловием сначала проверяется условие, а потом делается шаг. Грибник придет с полной или почти полной корзиной. В цикле с постусловием – сначала шаг, а потом проверка. Как всякий нормальный грибник, этот принесет полную или слегка переполненную корзину.

Какой алгоритм выбрать? Это зависит от конкретной задачи.

Если, сделав шаг без проверки, можно свалиться в яму, то лучше проверка вначале (как слепой с палочкой). Ну, а если шаг без проверки вас не пугает, то можно отложить ее до завершения шага.

Нужно также проанализировать событие, которого мы ожидаем. Если оно может случиться до первого шага, то нужен цикл с предусловием. А если событие не может случиться до первого шага, то нужен цикл с постусловием.

Оператор цикла Паскаля с постусловием

Рассмотрим теперь математическую задачу. Пусть нам необходимо вычислить сумму первых членов гармонического ряда, удовлетворяющих условию 1/i>= e, где 0< e<1, а i=1,2,3…...Эту задачу можно решить по следующему алгоритму: положить предварительно y=0 и i=0, а затем в цикле увеличивать i на 1, к значению y добавлять очередное слагаемое 1/ i до тех пор, пока текущее значение 1/ i впервые окажется больше заданного значения 0< e<1.

Очевидно, что число повторений этого цикла заранее не известно. В подобного рода случаях мы можем лишь сформулировать условие, при выполнении которого процесс добавления к сумме очередного слагаемого должен завершиться.

Для задания таких вычислительных процессов и служит оператор цикла Паскаля с постусловием. Этот оператор имеет вид:

Repeat S1; S2;…; Si until B,

где **repeat** (повторять) и **until** (до) – служебные слова, через **Si** обозначен любой оператор Паскаля, а через **В** – логическое выражение.

При выполнении этого оператора цикла последовательность операторов, находящихся между словами repeat и until, выполнится один или более раз. Этот процесс завершается, когда после очередного выполнения заданной последовательности операторов логическое выражение В примет (впервые) значение true. Таким образом, с помощью логического выражения В задается условие завершения выполнения оператора цикла. Поскольку в данном случае проверка условия производится после выполнения последовательности операторов (тела цикла), этот оператор цикла и называется оператором цикла с постусловием.

С использованием этого вида оператора цикла Паскаля задача о суммировании первых членов гармонического ряда, удовлетворяющих заданному условию, может быть реализована следующим образом:

Пример кода оператора цикла Паскаля с постусловием

readln(e);

i:=0;

y:=0;

Repeat

i:=i+1;

y:=y+1/i;

Until 1/i<e;

Заметим, что оператор цикла с постусловием является более общим, чем оператор цикла с параметром — любой циклический процесс, задаваемый с помощью цикла с параметром можно представить в виде цикла с постусловием. Обратное утверждение неверно. Например, задача о суммировании первых n членов гармонического ряда, рассмотренная ранее, с оператором цикла с постусловием будет выглядеть так:

Пример кода оператора цикла Паскаля с постусловием

Readln(n);

i:=0;

y:=0;

Repeat

i:=i+1;

y:=y+1/i;

Until i>n;

Оператор цикла Паскаля с предусловием

В случае оператора цикла Паскаля с постусловием входящая в него последовательность операторов заведомо будет выполняться хотя бы один раз. Между тем довольно часто встречаются такие циклические процессы, когда число повторений цикла тоже неизвестно заранее, но при некоторых значениях исходных данных предусмотренные в цикле действия вообще не должны выполняться, и даже однократное выполнение этих действий может привести к неверным или неопределенным результатам.

Пусть, например, дано вещественное число М. Требуется найти наименьшее целое неотрицательное число k, при котором 3 k> M. Эту задачу можно решить по следующему алгоритму: предварительно положить y=1 и k=0; затем в цикле домножать значение y на 3 и увеличивать значение k на 1 до тех пор, пока текущее значение y впервые окажется больше значения М. На первый взгляд, здесь можно воспользоваться оператором цикла с постусловием:

Пример кода оператора цикла Паскаля с постусловием

y:=1; k:=0;

Repeat

y:=y\*3;

k:=k+1;

Until y> M;

Однако нетрудно убедиться в том, что при M<1 будет получен неправильный результат k=1, тогда как должно быть получено k=0: в этом случае предварительно сформированное значение k=0 является окончательным результатом и действия, предусмотренные в цикле, выполняться не должны.

Для задания подобного рода вычислительных процессов, когда число повторений цикла заранее неизвестно и действия, предусмотренные в цикле, могут вообще не выполняться, и служит оператор цикла с предусловием. Этот оператор цикла имеет в Паскале следующий вид:

While B do S,

где **while** (пока), **do** (делать, выполнять) – служебные слова, **В** – логическое выражение, **S** – оператор. Здесь оператор S выполняется ноль или более раз, но перед каждым очередным его выполнением вычисляется значение выражения В, и оператор S выполняется только в том случае, когда значение выражения В true. Выполнение оператора цикла завершается, когда выражение В впервые принимает значение false. Если это значение выражение В принимает при первом же его вычислении, то оператор S не выполнится ни разу.

В рассматриваемой нами задаче правильное значение k при любом значении М может быть получено следующим образом:

Пример кода оператора цикла Паскаля с предусловием

y:=1; k:=0;

While y<=M do

Begin

y:=y\*3;

k:=k+1;

End;

Оператор цикла Паскаля с предусловием можно считать наиболее универсальным – с использованием таких операторов можно задать и циклические процессы, определяемые операторами цикла с параметром и постусловием.

Отметим **отличия и особенности хорошего стиля работы** с рассмотренными циклическими операторами.

|  |  |
| --- | --- |
| **Цикл с предусловием While (пока условие истинно)**  | **Цикл с постусловием Repeat (до истинности условия)**  |
| * До начала цикла должны быть сделаны начальные установки переменных, управляющих условием цикла, для корректного входа в цикл
 |
| 1. В теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций завершился
 |
| 1. Цикл работает пока условие истинно (пока True)
 | 1. Цикл работает пока условие ложно (пока False)
 |
| 1. Цикл завершается, когда условие становится ложным (до False)
 | 1. Цикл завершается, когда условие становится истинным (до True)
 |
| 1. Цикл может не выполниться ни разу, если исходное значение условия при входе в цикл False
 | 1. Цикл обязательно выполнится как минимум один раз
 |
| 1. Если в теле цикла требуется выполнить более одного оператора, то необходимо использовать составной оператор
 | 1. Независимо от количества операторов в теле цикла, использование составного оператора не требуется
 |
| **Цикл со счетчиком (с параметром) For** |
| * Начальная установка переменной счетчика цикла до заголовка не требуется
 |
| * Изменение в теле цикла значений переменных, стоящих в заголовке не допускается
 |
| * Количество итераций цикла неизменно и точно определяется значениями нижней и верхней границ и шага приращения
 |
| * Нормальный ход работы цикла может быть нарушен оператором goto или процедурами Break и Continue
 |
| * Цикл может не выполниться ни разу, если шаг цикла будет изменять значение счетчика от нижней границы в направлении, противоположном верхней границе
 |

Оператор, который выполняется в цикле, сам может быть циклом. Это относится ко всем видам циклов. В результате мы получаем **вложенные циклы**. Механизм работы вложенных циклов удобнее всего рассмотреть на примере вложенных циклов с параметром. Пусть нам нужно описать работу электронных часов, начиная с момента времени 0 часов, 0 минут, 0 секунд. Значение минут станет равным 1 только после того, как секунды «пробегут» все последовательные значения от 0 до 59. Часы изменят свое значение на 1 только после того, как минуты «пробегут» все последовательные значения от 0 до 59. Таким образом, вывод всех значений времени от начала суток до конца суток может быть представлен следующим фрагментом программы:

For h:=0 to 23 do

For m:=0 to 59 do

For s:=0 to 59 do

Writeln(h,’:’,m,’:’,s);

Для удобства реализации циклических структур на Паскале в последних версиях языка введены операторы **break** и **continue**, применяемые внутри циклов. Они расширяют возможности использования циклов и улучшают структуру программы.

В процессе выполнения тела цикла до полного завершения цикла могут возникнуть дополнительные условия, требующие завершения цикла. В этом случае цикл может быть прекращен оператором break.

В ходе выполнения цикла может возникнуть условие, при котором необходимо пропустить все или некоторые действия, предусмотренные в цикле, не прекращая работу цикла совсем. Для этого используется оператор continue, который передает управление в ту точку программы, где проверяется условие продолжения или прекращения цикла.

Циклический вычислительный процесс (ЦВП) характеризуется повторением одних и тех же вычислений над некоторым набором данных. Числом повторений цикла управляет специальная переменная, называемая его *счетчиком* или *управляющей переменной* цикла. На счетчик накладывается условие, определяющее, до каких пор следует выполнять цикл.

Повторяемый блок вычислений называют *телом* цикла. В теле цикла должно быть обеспечено изменение значения счетчика, чтобы он мог завершиться. Если тело цикла состоит более чем из одного оператора, оно заключается в операторные скобки begin... end;. Однократное выполнение тела цикла называют его *шагом*.

Таким образом, для программирования цикла достаточно определить условие, управляющее числом его повторений и описать операторы, образующие тело цикла. С этой точки зрения, теоретически возможны всего два вида циклов -- проверка условия либо предшествует выполнению тела цикла, либо происходит после него. Изобразим эти циклы в виде блок-схем (рис. 9.1).

В цикле с предусловием сначала проверяется условие, затем, в зависимости от того, истинно оно или ложно, либо выполняется тело цикла, либо следует переход к оператору, следующему за телом цикла. После завершения тела цикла управление вновь передается на проверку условия. Естественно, предполагается, что в теле цикла было обеспечено некоторое изменение входящих в условие переменных -- в противном случае произойдет *зацикливание* и программа "зависнет".

*Рис. 9.1. Блок-схемы циклов с предусловием и постусловием*

Для цикла с постусловием сначала выполняется тело цикла, затем управление передается на проверку условия. В зависимости от истинности или ложности условия, тело цикла выполняется повторно или же происходит переход к оператору, следующему за телом цикла. Всё, сказанное о возможном зацикливании для цикла с предусловием, справедливо и для цикла с постусловием.

Исходя из приведенных блок-схем, очевидно основное различие двух циклов: цикл с постусловием гарантированно *выполняется хотя бы раз*, а цикл с предусловием может не выполняться ни разу, если условие сразу же окажется ложным.

В языке Паскаль реализованы оба вида циклов. Цикл с предусловием имеет следующий общий вид:

while логическое\_выражение do begin

{операторы тела цикла}

end;

Работу цикла можно описать словами: "пока логическое выражение истинно, повторяется тело цикла".

Логическое выражение строится по правилам, изученным в гл. 7. Тело цикла могут образовывать любые операторы Паскаля. Если в цикле находится всего один оператор, операторные скобки, показывающие начало и конец тела цикла, можно не писать.

Общая запись цикла с постусловием следующая:

repeat

{операторы тела цикла}

until логическое\_выражение;

Работает цикл с постусловием следующим образом: "тело цикла повторяется до тех пор, пока логическое выражение не станет истинным". Обратите внимание, что, в отличие от while, цикл repeat в Паскале работает, пока условие *ложно*. Это отличие подчеркивается использованием ключевого слова until ("до тех пор, пока не") вместо while ("до тех пор, пока"). Кроме того, в виде исключения, тело цикла repeat, даже если оно состоит из нескольких операторов, можно не заключать в операторные скобки.

Довольно часто циклы взаимозаменяемы. Представим, например, что для каждого из значений переменной x=1, 2,...,20, нужно выполнить некоторый расчет (математически этот закон изменения x можно записать как или ). Это можно сделать как в цикле while:

x:=1;

while x<=20 do begin

{операторы расчета}

x:=x+1;

end;

так и с помощью repeat:

x:=1;

repeat

{операторы расчета}

x:=x+1;

until x>20;

Как видно из листинга, управляющей переменной x в обоих случаях присвоено начальное значение 1, оба цикла изменяют значение x и, соответственно, условие цикла, оператором x:=x+1;, но для цикла repeat условие "перевернуто" ("пока x не станет больше 20"), а тело не заключено в операторные скобки.

Зачастую использование одного из циклов выглядит предпочтительней. Например, обработка ввода пользователя с клавиатуры удобней с помощью repeat (сначала пользователь должен нажать клавишу, затем следуют проверки и обработка).