**2. Решение логических задач с использованием аппарата алгебры логики**

Знакомство с математическими основами логического мышления необходимы специалисту в любой области человеческой деятельности. Не удивительно, что логические задачи традиционно присутствуют в экзаменационных заданий по информатике. Их можно условно разделить на три группы: задачи с формальным условием, задачи с неформальным условием и задачи с числами. Некоторые типовые примеры предлагаем вниманию наших читателей.

**ЗАДАЧА 7.** Логическая функция задана в форме выражения с использованием базиса из трех операций: дизъюнкции, конъюнкции и отрицания. Построить эквивалентное выражение для данной функции с использованием базиса из двух операций: импликации и стрелки Пирса (операции Вебба).

**F(A,B) = A & B + ¬A & ¬B**

Для решения подобных задач необходимо хорошо знать все основные тождества логической алгебры и уметь выражать одни операции с помощью других.

В нашем случае ситуация не является слишком сложной. Прежде всего, обратим внимание на то, что функция представляет собой запись операции эквивалентности с помощью логических отрицания, сложения и умножения. Но операцию эквивалентности можно легко выразить с помощью импликации.

A & B + ¬A & ¬B = A ↔ B = (A → B) & (B → A)

Таким образом, нам остается заменить операцию логического умножения и задача будет решена. Известно, что все основные логические операции можно выразить с помощью стрелки Пирса (операции Вебба) или штриха Шеффера. С помощью стрелки Пирса операция умножения представляется следующим образом:

X & Y = (X↓ X) ↓ ( Y↓ Y)

После подстановки (A → B) и (B → A) получим:

F(A,B) = ( (A → B) ↓ (A → B) ) ↓ ( (B → A) ↓ (B → A) )

**Ответ: F(A,B) = ( (A → B) ↓ (A → B) ) ↓ ( (B → A) ↓ (B → A) )**

**ЗАДАЧА 8.** Логическая функция от четырех переменных имеет значение 1 (истина) для следующих наборов аргументов в таблице истинности: 0, 1, 2, 3, 4, 8, 9, 12. Определить логическое выражение для данной функции. Упростить полученное выражение. Полученное выражение должно содержать только три типа логических операций: отрицание, конъюнкцию и дизъюнкцию.

Это типичная задача на построение логической функции по заданной таблице ее значений. Методы решения данной задачи разработаны в математической логике очень хорошо. Основными являются метод нормальных форм и метод таблиц Вейча-Карно. Решим задачу двумя способами.

**Метод 1. Нормальные формы логических выражений.**

Попробуем найти выражение функции в виде совершенной дизъюнктивной нормальной формы (СДНФ). Для большего удобства запишем строки таблицы истинности, в которых функция принимает значение 1 (истина).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Набор | A | B | C | D |
| 0  1  2  3  4  8  9  12 | 0  0  0  0  0  1  1  1 | 0  0  0  0  1  0  0  1 | 0  0  1  1  0  0  0  0 | 0  1  0  1  0  0  1  0 |

Теперь составим выражение в виде СДНФ по известному правилу: запишем сумму логических произведений, по одному произведению для каждой строки таблицы с единичным значением функции. При этом каждая переменная входит в состав произведения либо с отрицанием, либо без него, в зависимости от того, каким значением она обладает в строке таблицы: отрицание используется для той переменной, которая имеет значение нуля.

¬A¬B¬C¬D + ¬A¬B¬CD + ¬A¬BC¬D + ¬A¬BCD +

¬AB¬C¬D + A¬B¬C¬D + A¬B¬CD + AB¬C¬D

Фактически выражение для функции уже получено. Остается только упростить выражение с помощью тождественных преобразований.

¬A¬B¬C¬D + ¬A¬B¬CD + ¬A¬BC¬D + ¬A¬BCD +

¬AB¬C¬D + A¬B¬C¬D + A¬B¬CD + AB¬C¬D =

¬A¬B¬C(¬D+D) + ¬A¬BC(¬D+D) + B¬C¬D(¬A+A) + A¬B¬C(¬D+D) =

¬A¬B¬C + ¬A¬BC + B¬C¬D + A¬B¬C = ¬B¬C(¬A+A) + ¬A¬BC + B¬C¬D =

¬B¬C + ¬A¬BC + B¬C¬D = ¬B(¬C+¬AC) + B¬C¬D = ¬B(¬C+¬A) + B¬C¬D =

¬B¬C + ¬A¬B + B¬C¬D = ¬C(¬B+B¬D) + ¬A¬B = ¬C(¬B+¬D) + ¬A¬B =

¬A¬B + ¬B¬C + ¬C¬D

В результате преобразований получаем окончательно:

**F(A,B,C,D) = ¬A¬B + ¬B¬C + ¬C¬D**

**Метод 2. Таблицы Вейча-Карно.**

Оптимальный вид логического выражения для заданной логической функции можно также получить с помощью специальной технологии, которая известна под названием «таблицы Вейча-Карно». Для функции от четырех переменных таблица Вейча-Карно имеет следующий вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | B | B | ¬B | ¬B |  |
| A | 12 | 13 | 9 | 8 | ¬C |
| A | 14 | 15 | 11 | 10 | C |
| ¬A | 6 | 7 | 3 | 2 | C |
| ¬A | 4 | 5 | 1 | 0 | ¬C |
|  | ¬D | D | D | ¬D |  |

Заполним ячейки таблицы в соответствии с определением функции: поставим единицы в клетках, которые соответствуют значениям функции 0,1,2,3,4,8,9,12, в остальных клетках запишем нули.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | B | B | ¬B | ¬B |  |
| A | 1 | 0 | 1 | 1 | ¬C |
| A | 0 | 0 | 0 | 0 | C |
| ¬A | 0 | 0 | 1 | 1 | C |
| ¬A | 1 | 0 | 1 | 1 | ¬C |
|  | ¬D | D | D | ¬D |  |

Теперь найдем единицы, образующие группы соседних клеток в количестве 8, 4, 2 или 1. При этом следует учитывать, что верхний и нижний, а также левый и правый края таблицы являются смежными. Для каждой такой группы можно образовать логическое произведение переменных, значения которых не изменяются в пределах группы. Каждая переменная входит в состав произведения либо с отрицанием, либо без него. Для данной таблицы можно образовать 4 группы по 4 единицы. Покажем состав групп и соответствующие им логические произведения.

Группа 1: { 0, 1, 2, 3 } → ¬A¬B

Группа 2: { 0, 1, 8, 9 } → ¬B¬C

Группа 3: { 0, 4, 8, 12 } → ¬C¬D

Объединяя произведения с помощью дизъюнкции, получим выражение для логической функции.

**F(A,B,C,D) = ¬A¬B + ¬B¬C + ¬C¬D**

Не удивительно, что полученное выражение совпадает с полученным в результате тождественных преобразований. В данном случае это служит дополнительным подтверждением правильности результата.

**Ответ: F(A,B,C,D) = ¬A¬B + ¬B¬C + ¬C¬D**

**ЗАДАЧА 9.** При обсуждении своих летних планов на отдых четыре подруги высказали следующие утверждения.

1) Света: Вика или Настя не поедет в Италию только тогда, когда туда поеду я;

2) Даша: Мы вместе с Викой обязательно едем в Италию;

3) Настя: Мы вместе со Светой ехать не можем; Или едет кто-то один, или никто;

Вика подумала немного и сделала следующий вывод: Значит, если Настя не едет в Италию, то туда поедут Света и Даша. Настя сказала, что из высказанных утверждений такой вывод вовсе не следует. Кто из двоих прав? Проверить правильность вывода с помощью логической алгебры.

Основой для решения задачи является следующее. Если на основании нескольких логических высказываний-посылок, например, X, Y, Z, делается некоторый вывод W, то его истинность равносильна тождеству следующего вида:

X & Y & Z → W = 1

Другими словами, импликация из конъюнкции посылок в сторону утверждения-вывода должна быть истинной при любых значениях аргументов. Такие тождественно истинные высказывания называются тавтологиями. Таким образом, если X, Y, Z – это утверждения подруг, высказанные в ходе обсуждения, а W – вывод, сделанный Викой, то для доказательства правильности вывода надо доказать, что логическая функция вида X&Y&Z→W есть тавтология.

Составим высказывания, необходимые для анализа.

Введем следующие обозначения:

А: Настя едет в Италию;

В: Вика едет в Италию;

С: Света едет в Италию;

Д: Даша едет в Италию;

Тогда исходные высказывания можно записать следующим образом:

Света: ( ¬А + ¬В ) → С

Даша: В & Д

Настя: ¬ ( А & С )

Аналогично, вывод, сделанный Викой, будет иметь следующее формальное выражение:

¬А → СД

Теперь достаточно составить импликацию и проверить ее тождественную истинность.

(¬А+¬В → С) & (ВД) & (¬(АС)) → ( ¬А → СД ) = 1 ?

Для проверки можно составить таблицу истинности логической функции или упростить выражение с помощью тождественных преобразований. Найдем решение двумя способами.

**Метод 1. Тождественные преобразования.**

¬ ( ( ¬ ( ¬ А + ¬ В ) + С ) (ВД) ( ¬ (АС) ) + А + СД =

¬ ( (АВ + С ) ВД ( ¬ (АС) ) + А + СД =

¬ ( АВ + С ) + ¬ (ВД) + АС + А + СД =

(¬ (АВ) ¬ С ) + ¬ В + ¬ Д + А + СД =

( ¬ А + ¬ В ) ¬ С + ¬ В + ¬ Д + А + С =

¬ А ¬ С + ¬ В ¬ С + ¬ В + ¬ Д + А + С =

¬ А + ¬ В + ¬ Д + А + C = 1

**Метод 2. Таблица истинности.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| А | В | С | Д | ¬А | ¬В | 1+2 | 3→С | АС | ¬5 | ВД | 467 | СД | 1→9 | 8→10 |
| 0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1 | 0  0  0  0  1  1  1  1  0  0  0  0  1  1  1  1 | 0  0  1  1  0  0  1  1  0  0  1  1  0  0  1  1 | 0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1 | 1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0 | 1  1  1  1  0  0  0  0  1  1  1  1  0  0  0  0 | 1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0 | 0  0  1  1  0  0  1  1  0  0  1  1  1  1  1  1 | 0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  1 | 1  1  1  1  1  1  1  1  1  1  0  0  1  1  0  0 | 0  0  0  0  0  1  0  1  0  0  0  0  0  1  0  1 | 0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0 | 0  0  0  1  0  0  0  1  0  0  0  1  0  0  0  1 | 0  0  0  1  0  0  0  1  1  1  1  1  1  1  1  1 | 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 |

Разумеется, что при отсутствии ошибок оба метода дают одинаковые результаты.

Подведем итоги. Построенная импликация является тождественно истинным логическим выражением. Следовательно, в нашей ситуации Вика оказалась права.

**Ответ: Вика сделала правильный вывод из всех высказанных логических утверждений.**

**ЗАДАЧА 10.** Целое число M получено в результате выполнения логических и арифметических операций над целыми числами. Определить значение числа и записать его в формате компьютера, как целое число со знаком. Длина числа шестнадцать двоичных разрядов. Результат представить в двоичной и шестнадцатиричной системах счисления.

**M = 7722(8) – (((21 or 22 or … or 220 ) × 121(3) ) xor 1FFF000(16))**

Для решения задач с логическими операциями над числами следует вспомнить два ключевых вопроса. Логические операции выполняются по обычным правилам выполнения над одноименными двоичными разрядами чисел. Числа в запоминающих устройствах компьютера имеют фиксированную разрядность и определенные особенности представления.

Начинать решение задачи надо с выполнения операций над числами. Прежде всего, вычислим логическую сумму степеней двойки от первой до двадцатой. Конечно, для получения результата не надо складывать все числа. Обратим внимание лишь на то, что любая степень двойки записывается в двоичной системе счисления, как единица с последующими нулями.

21 = 10

22 = 100

23 = 1000

…

220 = 1000…0

Таким образом, в каждом разряде, кроме последнего, хотя бы в одном числе присутствует значение единицы. Следовательно, при логическом сложении мы получим в результате 20 единиц в старших разрядах и нуль в последнем разряде числа.

1 1111 1111 1111 1111 1110(2)

Троичное число 121 – это 16 в десятичной системе счисления или 10000 в двоичной записи:

121(8) = 1 × 32 + 2×3 + 1 = 9 + 6 + 1 = 16(10)

16(10) = 24 = 10000(2)

Умножение двоичного числа на 10000(2) равносильно добавлению к числу четырех двоичных нулей справа. Для полученного значения и для шестнадцатиричного числа 1FFF000, выполняем логическую операцию ИЛИ-ИЛИ (xor). Разумеется, оба числа должны быть записаны в двоичной системе счисления.

1 1111 1111 1111 1111 1110 0000

1 1111 1111 1111 0000 0000 0000

-------------------------------

0 0000 0000 0000 1111 1110 0000

Операцию вычитания можно выполнить в любой системе счисления, главное, чтобы оба числа были записаны с одним и тем же основанием.

M = 7722(8) – 111111100000(2) = 7722(8) - 7740(8) =

= - (7740(8) – 7722(8)) = - 16(8) = - 1110(2)

Теперь надо записать полученное отрицательное число в формате компьютера. Отрицательные целые числа хранятся в форме двоичных дополнений. Дополнительный код Д(М) для двоичного числа длиной в 16 разрядов определяется следующим образом.

Д(M) = 216 – M

Для нашего числа получим:

Д(М) = 1 0000 0000 0000 0000(2) – 1110(2)

или в шестнадцатиричной форме: Д(М) = 10000(16) – E(16)

Операцию вычитания можно не выполнять. Вместо этого можно просто заменить противоположными значениями все двоичные разряды, кроме самой последней единицы и всех нулей справа от нее. Получим:

Прямой код числа: 0000 0000 0000 1110

Дополнительный код числа: 1111 1111 1111 0010

Это и есть дополнительный двоичный код отрицательного числа в формате компьютера. Для окончательного решения задачи остается получить шестнадцатиричную запись полученного значения: FFF2.

**Ответ:**

**1111 1111 1111 0010(2)**

**FFF2(16)**

**3. Методика анализа программ**

Основы программирования – обязательная часть любого информационного образования. Но наряду с традиционными заданиями, в которых требуется написать программу решения задачи, многие учебные заведения предлагают задания противоположного характера. Речь идет о том, чтобы определить результаты выполнения алгоритма по его записи в форме текста или графической схемы. Проблема анализа программы (алгоритма) сводится к ответу на главный вопрос: что получится в результате выполнения? К сожалению формального решения этой проблемы не существует и это утверждают достижения современной математики, в частности теорема о проблеме останова машины Тьюринга, знаменитые теоремы Геделя о неполноте, теоремы Райса и Тарского. Другими словами, не существует алгоритма, с помощью которого можно определить результат работы другого алгоритма. Единственный метод определения результатов работы программы – это выполнение программы, либо реальное, либо «виртуальное», т.е. «на бумаге». В последнем случае можно использовать любые изобразительные средства для отражения основных ходов алгоритма, например, таблицы для записи значений переменных и логических условий.

Условия для всех задач этого набора могут быть сформулированы одинаковым образом:

Дана запись программы на языке Паскаль. Определить результат выполнения программы. Если процесс выполнения завершается выводом значений величин на экран монитора, то указать эти значения. Если процесс выполнения завершается аварийно или не может завершиться за конечное время, то указать причину возникновения исключительной ситуации.

**ЗАДАЧА 11.**

program primer(input,output);

var x, y, z: integer;

BEGIN

x := 123;

y := 321;

while x > y do

begin

if x>y then

x := x – y

else

y := y – x;

z := x div y

end;

writeln(z)

END.

Как уже было замечено, никакой формальной технологии для решения задач анализа нет и быть не может. Но для отражения всех изменений, происходящих в ходе выполнения программы, можно использовать специальные изобразительные средства, например, таблицы. Покажем, как это делается.

В нашей программе используются три переменные, которые изменяют свои значения внутри цикла с логическим условием. Следовательно, надо следить за изменением значений этих переменных. При этом переменная z зависит только от x и y, поэтому вычислять ее при каждом повторении не обязательно – достаточно сделать это один раз перед выходом из цикла. Таким образом, для наблюдения за программой достаточно совсем простой таблицы с тремя колонками.

|  |  |  |
| --- | --- | --- |
| x | y | x > y |
| 123  123  123  48  48  21  21  15  9  3  3  3 | 321  198  75  75  27  27  6  6  6  6  3  0 | false  false  true  false  true  false  true  true  true  false  false  true  Деление  на 0 |

Вычисляя значения x и y, надо помнить о таком важном математическом понятии, как область допустимых значений. Переменная y используется в качестве знаменателя при вычислении z, причем никакой проверки на равенство нулю не делается. Следовательно, нулевое значение y не является допустимым внутри данного цикла. Но именно это значение получит величина y в результате выполнения операций. Попытка выполнить операцию деления на нуль всегда приводит к возникновению аварийного завершения программы (без результата).

**Ответ: При выполнении программы возникает исключительная ситуация «Деление на 0». Программа завершается аварийно.**

**ЗАДАЧА 12.**

program primer(input,output);

var i, n: integer;

BEGIN

For i := 49 to 53 do

begin

n := i;

while n <> 34 do

if n mod 2 = 0 then

n := n div 2

else

n := n\*3 + 1

end;

writeln(n)

END.

Методы анализа в этой задаче будут точно такими же, как в предыдущей. Составляем таблицу и следим за изменениями значений переменной n.

|  |  |  |  |
| --- | --- | --- | --- |
| n=49 | n=50 | n=51 | n=52 |
| 148  74  37  112  56  28  14  7  22  11  34 | 25  76  38  19  58  29  88  44  22  11  34 | 154  77  232  116  58  29  88  44  22  11  34 | 26  13  40  20  10  5  16  8  4  2  1  Период  (4,2,1) |

Отличие этой задачи от задачи №11 только в трудоемкости вычислений. При первом выполнении цикла while при i=49 происходит выход из цикла из-за нарушения логического условия. Аналогичный результат дает выполнение цикла при i=50 и i=51. При i=52 ситуация меняется. Возникает периодическое повторение значений величины, причем значение 34 не достигается. Фактически это означает, что выполнение цикла не может быть завершено – возникает ситуация бесконечного цикла.

**Ответ: При выполнении программы возникает исключительная ситуация «Бесконечный цикл». Программа зацикливается.**

**ЗАДАЧА 13.**

program primer(input,output);

var x: integer;

function f(n:integer):integer;

begin

if n = 0 then

f := 0

else

f := f(n div 10)\*10 + n

end;

BEGIN

x := (f(123)+ f(321)) div 10;

writeln(x)

END.

Программы, в которых используются рекурсивные функции, это самый трудоемкий объект для анализа. Но и здесь можно разобраться с логикой алгоритма. Надо только понять, что анализ рекурсии надо производить «с конца», а не «с начала». Это неизбежно, потому что рекурсивная функция вызывает саму себя и мы не можем понять, какое значение она вычисляет, пока не дойдем до конца этой цепочки.

К функции f(n) происходят два обращения, одно с параметром 123, другое – с параметром 321. Посмотрим, что происходит при первом обращении. Если значение параметра не равно нулю, то функция вычисляет некоторый результат, используя обращение к самой себе, но с другим значением параметра. Поскольку значение целого числа все время делится на 10 нацело, то на каком-то этапе это значение становится равным нулю и с этого момента функция уже не вызывает себя, а возвращает значение 0. Вот с этой точки мы и можем начать анализировать процесс рекурсии.

После обращения со значением 0 функция возвращает нулевое значение в ту функцию, которая соответствует обращению со значением n=1 (старшая цифра в записи числа 123). Используя это нулевое значение, функция вычисляет результат по формуле и может вернуть его на вышестоящий уровень обращения к ней, и т.д. В таблице показано, какие значения принимает параметр n и что вычисляет функция f(n) на каждом уровне рекурсивного обращения.

|  |  |
| --- | --- |
| n | f(n) |
| 0  1  12  123  0  3  32  321 | 0  f(0)\*10+1=0+1=1  f(1)\*10+12=10+12=22  f(12)\*10+123=220+123=343  0  f(0)\*10+3=0+3=3  f(3)\*10+32=30+32=62  f(32)\*10+321=620+321=941 |

После того, как результаты обращения к рекурсивной функции становятся известными вычислить окончательные итоги работы программы совсем просто. После подстановки значений получим значение переменной x, которое будет показано на экране монитора.

(343 + 941) div 10 =

1284 div 10 = 128

**Ответ: 128.**

**ЗАДАЧА 14.**

program primer(input,output);

var

a: array [1..100] of integer;

i, n, z, k: integer;

BEGIN

n := 8;

for i := 1 to n do

a[i] := i \* i;

i := 0;

repeat

i := i + 1;

z := a[i];

a[i] := a[a[i] div 4 + 1];

a[a[i] div 4 + 1] := z

until i > n div 2;

i := 1;

while a[a[i] div i] = a[i] do

i := i + 1;

for k := 1 to n do

a[k] := a[k] mod i;

for i := 1 to n do

writeln(a[i])

END.

Программы с массивами требуют, как правило, больше места для отслеживания всех изменений: следить надо не за одной или двумя переменными, а за множеством значений элементов массива. Кроме этого, обращение к элементу массива происходит с помощью индексного выражения, а оно может быть достаточно сложным. Пример такого сложного обращения мы видим на примере данной задачи. Индексом для обращения к элементу массива является значение одного из элементов, такой способ индексации называется косвенным.

В самом начале работы программы происходит просто заполнение одномерного массива квадратами натуральных чисел от 1 до 8. После выхода из цикла for состояние массива выглядит следующим образом:

(1, 4, 9, 16, 25, 36, 49, 64)

Дальше начинается наиболее трудоемкая для анализа часть алгоритма и здесь не обойтись без таблиц. Таблицы помогут нам не запутаться в косвенной индексации и аккуратно проследить за каждым изменением в состоянии массива.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Операторы цикла | i | z | a | Состояние массива a |
| 1  2  3  4 | i:=i+1;  z:=a[i];  a[i]:=a[a[i] div 4+1];  a[a[i] div 4+1]:=z | 1 | 1 | a[1]:=a[1]  a[1]:=1 | 1,4,9,16,25,36,49,64  1,4,9,16,25,36,49,64 |

Для экономии места не будем повторять форму каждого оператора внутри цикла, а ограничимся только их номерами.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | i | z | a | Состояние массива a |
| 1  2  3  4 | 2 | 4 | a[2]:=a[2]  a[2]:=4 | 1,4,9,16,25,36,49,64  1,4,9,16,25,36,49,64 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | i | z | a | Состояние массива a |
| 1  2  3  4 | 3 | 9 | a[3]:=a[3]  a[3]:=9 | 1,4,9,16,25,36,49,64  1,4,9,16,25,36,49,64 |

Пока с элементами массива ничего особенного не происходит: элементы с индексами 1,2,3 получили свои собственные значения. Но, начиная с четвертого элемента ситуация меняется. Индексные выражения в операторе № 3 слева и справа отличаются друг от друга. Четвертый элемент массива получает значение пятого и его значение меняется. Аналогично, следующий оператор меняет значение седьмого элемента. Здесь надо быть особенно внимательным: индекс для обращения к элементу использует значение элемента, которое было изменено предыдущим оператором программы! В таких «хитросплетениях» алгоритма очень легко запутаться.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | i | z | a | Состояние массива a |
| 1  2  3  4 | 4 | 16 | a[4]:=a[5]  a[7]:=16 | 1,4,9,16,25,36,49,64  1,4,9,25,25,36,49,64  1,4,9,25,25,36,16,64  1,4,9,25,25,36,16,64 |

При i=5 с элементами массива происходят аналогичные преобразования. Сначала пятый элемент получит значение седьмого. Но уже следующий оператор вернет ему прежнее значение (25).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | i | z | a | Состояние массива a |
| 1  2  3  4 | 5 | 25 | a[5]:=a[7]  a[5]:=25 | 1,4,9,25,25,36,16,64  1,4,9,25,16,36,16,64  1,4,9,25,25,36,16,64  1,4,9,25,25,36,16,64 |

При значении i=5 условие цикла repeat-until становится истинным и происходит выход из цикла. После выполнения цикла элементы массива будут иметь следующие значения.

( 1, 4, 9, 25, 25, 36, 16, 64 )

Определить результат последнего цикла программы значительно проще. Мы переходим к очередному элементу массива до тех пор, пока элементы a[i] и a[a[i] div i] равны между собой. Это равенство будет нарушено для четвертого элемента a[4]. Понять механизм работы в этом случае поможет таблица следующей формы.

|  |  |  |  |
| --- | --- | --- | --- |
| i | a[i] | a[a[i] div i] | a[a[i] div i]=a[i] |
| 1  2  3  4 | 1  4  9  25 | a[1]=1  a[2]=4  a[3]=9  a[6]=36 | true  true  true  false |

|  |  |  |
| --- | --- | --- |
| k | a[ k ] | a[ k ] mod 4 |
| 1  2  3  4  5  6  7  8 | 1  4  9  25  25  36  16  64 | 1  0  1  1  1  0  0  0 |

При i=4 условие цикла нарушается и происходит выход из него. Дальше это значение переменной используется для изменения значений элементов массива. Элементы массива при делении на 4 или дают остаток 1, или делятся без остатка. Таким образом, результатом выполнения программы является последовательность целых чисел вида: 1,0,1,1,1,0,0,0.

**Ответ: 1, 0, 1, 1, 1, 0, 0, 0.**