Задание линии 2 в его современном виде появилось в Демоверсии-2016 и за 8 лет не претерпело особых изменений. Постепенно условие трансформировалось из задачи на анализ полной таблицы истинности в анализ частично заполненного фрагмента таблицы истинности.

Демоверсия-2016

Логическая функция F задаётся выражением (¬z)/\x \/ x/\y. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z.

Перем.1	**Перем.2**	**Перем.3**	**Функция**
???	**???**	**???**	**F**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

В ответе напишите буквы x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала – буква, соответствующая 1-му столбцу; затем – буква, соответствующая 2-му столбцу; затем – буква, соответствующая 3-му столбцу). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Демоверсия-2017

Логическая функция F задаётся выражением x /\ ¬y /\ (¬z \/ w). Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных x, y, z.

Перем.1	**Перем.2**	**Перем.3**	**Перем.4**	**Функция**
???	**???**	**???**	**???**	**F**
0	0	1	0	1
0	0	1	1	1
1	0	1	1	1

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы (сначала – буква, соответствующая 1-му столбцу; затем – буква, соответствующая 2-му столбцу; затем – буква, соответствующая 3-му столбцу; затем – буква, соответствующая 4-му столбцу). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Демоверсия-2024

Миша заполнял таблицу истинности логической функции $F=(x \wedge \neg y) \vee (y \equiv z) \vee \neg w$, но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных *w*,*x*,*y*,*z*.

				F
		0	0	0
1	0		0	0
1	0	1		0

Определите, какому столбцу таблицы соответствует каждая из переменных *w*,*x*,*y*,*z*.

Согласно спецификации задание считается аналитическим, то есть для его выполнения не требуется дополнительных программных средств. Мы же рассмотрим как аналитические способы решения, так и программные.

Аналитический метод решения

Данный метод сводится к следующей последовательности действий:

1) Найти наборы переменных, соответствующие данным в таблице,
2) Сопоставить найденные наборы и строки из фрагмента,
3) Определить порядок столбцов во фрагменте

Для использования метода нужно знать, какие бывают логические операции и их приоритет, и уметь строить таблицу истинности.

Логические операции и их приоритет выполнения

На сегодня (ЕГЭ-2024) в заданиях встречается 5 операций: отрицание, логическое умножение, логическое сложение, логическое следование и логическая равнозначность. Так же эти операции могут называться инверсией, конъюнкцией, дизъюнкцией, импликацией и тождество/эквивалентностью соответственно.

Обозначения, используемые в ЕГЭ:

Операция	**Обозначение**	**Пример**
Отрицание	¬	¬A
Логическое умножение	∧	A ∧ B
Логическое сложение	∨	A ∨ B
Логическое следование	→	A → B
Логическая равнозначность	≡	A ≡ B

Как нетрудно заметить, отрицание является одноместной операцией (для одного значения), остальные четыре операции – двуместные (для двух значений).

Логические операции обрабатывают логические значения – ИСТИНА и ЛОЖЬ. Для более краткой записи принято истинное значение записывать, как 1, ложное – 0. Соответственно, значение, которое возвращает логическая операция, тоже логическое (может быть либо 0, либо 1).

Представим все возможные результаты логических операций с помощью таблиц.

Отрицание	
A	¬A
0	1
1	0

Логическое умножение		
A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Логическое сложение		
A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

Логическое следование		
A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Логическая равнозначность		
A	B	A ≡ B
0	0	1
0	1	0
1	0	0
1	1	1

В заданиях 2 линии в ЕГЭ вычисления регламентированы скобками. Как минимум пока что не было заданий, где нужно было бы сначала решить в каком порядке выполняются операции. Однако никто не застрахован от подобных сюрпризов. Поэтому рассмотрим и этот небольшой блок.

Приоритет выполнения операций (меньше номер – больший приоритет)

Приоритет	**Операция**
1	Отрицание
2	Логическое умножение
3	Логическое сложение
4	Логическое следование
5	Логическая равнозначность

Для выполнения заданий уровня ЕГЭ необходимо запомнить только одно правило – сначала выполняются отрицания, затем двуместные операции. Например, в выражении из демоверсии-2024 последовательность действий может быть такой

$$\underset{2\ 1}{(x \wedge \neg y)} \vee \underset{4}{(y \equiv z)} \vee \underset{3\quad 6\ 5}{\neg w}$$

Этот же принцип важно соблюдать при программных решениях (далее рассмотрим подробно эту идею).

Построение полной таблицы истинности

Первый этап – расставить в выражении порядок выполнения операций.

Второй этап – перечислить все наборы переменных.

Для этого надо определить количество наборов N. Определяется просто – 2 в степени количества переменных. Три переменных – 2^3, 4 переменных – 2^4.

Способа заполнения есть два

1 способ: перевести числа от 0 до N-1 в k-знаковое двоичное число, где k – количество переменных.

Например для трех переменных будут следующие наборы.

Десятичное число	0	1	2	3	4	5	6	7
Двоичное число	000	001	010	011	100	101	110	111

2 способ: левый столбец заполнить по правилу – первая половина 0, вторая 1. В следующем столбце группы 0 и единиц также разделить пополам и первую половину заполнить 0, вторую – 1. Так продолжать до последнего столбца.

Для трех переменных

П1
0
0
0
0
1
1
1
1

П1	П2
0	0
0	0
0	1
0	1
1	0
1	0
1	1
1	1

П1	П2	П3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Либо этот же способ в обратном направлении: сначала заполняем группами по одному значению правый столбец, потом группами по 2, далее группами по 4 и т.д. каждый раз увеличивая размер группы вдвое.

П1	П2	П3		П1	П2	П3		П1	П2	П3
		0			**0**	0		**0**	0	0
		1			**0**	1		**0**	0	1
		0			**1**	0		**0**	1	0
		1			**1**	1		**0**	1	1
		0			**0**	0		**1**	0	0
		1			**0**	1		**1**	0	1
		0			**1**	0		**1**	1	0
		1			**1**	1		**1**	1	1
		По 1			**По 2**			**По 4**		

Теперь под каждую операцию нужно выделить отдельный столбец и пошагово вычислить значение F для каждого набора переменных.

Рассмотрим построение полной таблицы истинности на примере выражения из демоверсии-2024.

$$\underset{(x \wedge \neg y)}{2\,1} \vee \underset{(y \equiv z)}{4} \vee \underset{\neg w}{3} \vee \underset{}{6\,5}$$

x	**y**	**z**	**w**	1	2 = (x ∧ (1))	3 = (y ≡ z)	4 = ((2) ∨ (3))	5 = ¬w	F = ((4) ∨ (5))
0	0	0	0	1	0	1	1	1	1
0	0	0	1	1	0	1	1	0	1
0	0	1	0	1	0	0	0	1	1
0	0	1	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1	1
0	1	0	1	0	0	0	0	0	0
0	1	1	0	0	0	1	1	1	1
0	1	1	1	0	0	1	1	0	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1
1	0	1	0	1	1	0	1	1	1
1	0	1	1	1	1	0	1	0	1
1	1	0	0	0	0	0	0	1	1
1	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	1
1	1	1	1	0	0	1	1	0	1

В условиях задач линии 2 в таблице истинности приведены некоторые наборы и значения из столбца с результатом.

Также таблицу истинности можно заполнять построчно, подставив значения переменных в выражение. Например, для набора (x, y, z ,w) = (0, 1, 1, 0) получим выражение

$$(0 \wedge \neg 1) \vee (1 \equiv 1) \vee \neg 0 = 0 \vee 1 \vee 1 = 1$$

Метод эквивалентен пошаговому заполнению таблицы по результату. Однако, на мой личный взгляд, требует значительно больше времени на его применение.

Подбор вариантов по значению

В заданиях ЕГЭ встречались задания, в которых значения функции для фрагмента таблицы были одинаковые (либо 0, либо 1). Поэтому для более удобного и быстрого нахождения наборов можно их подобрать исходя из значений результата.

Обычно одну переменную можно определить однозначно сразу, либо сделав несложное наблюдение. Например, в Демоверсии-2024 это значение w равное 1 (так как для других значений выражение будет истинно). В демоверсии-2022 было выражение ¬(*y*→(*x*≡*w*))∧(*z*→*x*)=1. В котором также можно определить однозначно значение y = 1, так как логическое произведение должно быть равно 1, следовательно, выражение под отрицанием в первой скобке должно принимать ложное значение, это возможно только при y = 1. В варианте резервного дня в 2022 году однозначно определялось уже 2 переменных $w \to ((x \to z) \to y) = 0$. w = 1, y = 0.

Поэтому первый прием – попробовать найти переменную (или пару переменных), которые принимают строго заданное значение.

Второй прием – разделить выражение на подвыражения и потом из найденных значений составить наборы переменных.

Пример $(x \wedge \neg y) \vee (y \equiv z) \vee \neg w = 0$

Чтобы результат логического сложения был равен 0, надо чтобы все его части (подвыражения) были также равны 0.

Или $(x \wedge \neg y) = 0,\ (y \equiv z) = 0,\ \neg w = 0$

x	y	$x \wedge \neg y$
0	0	0
0	1	0
1	1	0

y	z	$y \equiv z$
0	1	0
1	0	0

w	$\neg w$
1	0

Теперь объединим полученные наборы, учитывая, что переменная y должна принимать значение одинаковое как в первом подвыражении, так и во втором.

x	y	$x \wedge \neg y$
0	0	0
0	1	0
1	1	0

y	z	$y \equiv z$
0	1	0
1	0	0

w	$\neg w$
1	0

x	y	z	w	F
0	0	1	1	0
0	1	0	1	0
1	1	0	1	0

Рассмотрим чуть более сложное задание.

(x ∧ (y ∨ ¬z) ∧ w) ≡ (x → ¬y ∧ z) = 1

Понятно, что подвыражения (x ∧ (y ∨ ¬z) ∧ w) и (x → ¬y ∧ z) должны принимать одинаковые значения.

Если x = 0, то левое подвыражение будет точно ложным, в то время как правое – истинным. Значит x = 1 для всех искомых наборов.

Заметим, что (y ∨ ¬z) = ¬ (¬y ∧ z), то есть, когда значение (¬y ∧ z) равно 0 значение y ∨ ¬z равно 1. В таком случае, w = 0, чтобы сохранить равнозначность левой и правой частей.

В обратном случае части не могут быть равнозначными, так как значение импликации будет равно 1, в то время как значение конъюнкции будет ложным.

Следовательно, имеем следующие наборы:

x	y	z	w	F
1	0	0	0	1
1	1	0	0	1
1	1	1	0	1

Полезные приемы поиска строк и столбцов

Последний этап решения задачи уже не связан непосредственно с алгеброй логики. Необходимо сопоставить два фрагмента таблицы и определить, в каком порядке должны следовать столбцы.

Важно помнить, что переставлены могут быть не только столбцы, но и строки. То есть строки в полученном нами фрагменте таблицы истинности могут располагать в таблице, приведённой в задании, в другом порядке.

Прием 1 (столбцы с одним значением)

Часто на ЕГЭ попадается задание, где в одном или двух столбцах повторяется одно и тоже значение. Такое наблюдение может отсеять из потенциальных вариантов те столбцы, в которых в задании указаны разные значения.

Например, в следующей таблице столбец со всеми единицами может быть только первым из столбцов, а столбец со всеми нулями либо 2, либо 4 столбцом.

				F
		0	0	0
1	0		0	0
1	0	1		0

Прием 2 (количество единиц/нулей в столбце)

Аналогичное наблюдение можно делать, когда известно, что в столбце должно быть, например, 2 единицы. Если в фрагменте невозможно в одном или нескольких столбцах разместить 2 единицы, то они не подходят.

Пример фрагмента из задания и полученных наборов.

x	y	z	w	F
1	0	0	0	1
1	1	0	0	1
1	1	1	0	1

				F
	1			1
1	1			1
1	1	1		1

Нетрудно заметить, что второй столбец – x, первый – y, третий – z и, по остаточному принципу, четвертый – w.

Прием 3 (один 0 или одна 1 в строке)

Если в найденных наборах ровно одна строка, где находится один 0 или одна 1, при этом строка однозначно определяется во фрагменте из задания, то можно определить столбец, который содержит в себе уникальное значение.

Пример фрагмента из задания и полученных наборов.

x	y	z	w	F
0	0	1	0	1
1	0	0	1	1
1	1	1	0	1

				F
		1		1
1	1			1
	1	1		1

В полученных наборах в первой строке три нуля и одна единица. Эта строка может быть только первой строкой во фрагменте из условия. Следовательно, где 1 там столбец z.

Прием 4 (разнозначные столбцы)

Обычно, такой прием работает, когда одно из подвыражений содержит равнозначность. Тогда мы можем определить пару столбцов, которые нужно выделить под такие переменные, и распределять оставшиеся переменные по другим столбцам.

Пример фрагмента из задания и полученных наборов.

x	y	z	w	F
0	0	1	1	1
1	0	0	0	1
1	1	1	0	1

				F
0	1	1	0	1
1	1	0		1
	0		0	1

В таком случае мы можем заключить, что столбцы x и w не могут быть столбцами 1-2, 1-4, 2-3, 2-4, 3-4. Следовательно, это пара столбцов 1-3.

Про преобразование

Данный способ обычно усложняет решение задания (задания простые и описанных выше приемов хватает), однако, можно привести выражение в сумме произведений, когда имеем дело с 1 в результате, или с произведением сумм, когда имеем дело с 0 в результате).

Данный метод сводится к приведению выражения к форме, которая включает три операции: отрицание, дизъюнкция, конъюнкция. Для этого пользуются правилами и законами, позволяющими переходить от одной формы записи к другой.

Основные правила и законы представлены в следующей таблице.

Название	**Формула (ы)**
Закон снятия двойного отрицания	¬¬A = A
Правила работы с константами	$1 \vee A = 1$ $0 \vee A = A$ $0 \wedge A = 0$ $1 \wedge A = A$
Закон исключённого третьего	A ⋁ ¬A = 1
Закон противоречия	A ⋀ ¬A = 0
Правила идемпотентности	$A \wedge A = A \vee A = A$
Раскрытие импликации	A → B = ¬A ⋁ B
Раскрытие равнозначности	A ≡ B = ¬A ⋀ ¬B ⋁ A ⋀ B
Законы де Моргана	$\neg(A \wedge B) = \neg A \vee \neg B$ $\neg(A \vee B) = \neg A \wedge \neg B$
Закон поглощения	A ⋁ (A ⋀ B) = A A ⋀ (A ⋁ B) = A
Правило свертки	¬A ⋁ (A ⋀ B) = ¬A ⋁ B ¬A ⋀ (A ⋁ B) = ¬A ⋀ B
Правило раскрытия скобок	(A ⋁ B) ⋀ (A ⋁ C) = A ⋁ B ⋀ C (A ⋀ B) ⋁ (A ⋀ C) = A ⋀ (B ⋁ C)

Рассмотрим данный метод на нескольких авторских задачах, в которых его применение может иметь смысл.

Получение дизъюнктивной нормальной формы

Данный метод применим для постановок, в которых в столбце результата все 1.

Статград 08.02.2022

Логическая функция *F* задаётся выражением:(¬*y*→(*z*≡*w*))∧((*z*→*x*)≡*w*)

Дан частично заполненный фрагмент, содержащий неповторяющиеся строки таблицы истинности функции *F*. Определите, какому столбцу таблицы истинности соответствует каждая из переменных *w*,*x*,*y*,*z*.

?	**?**	**?**	**?**	**F**
1	1	0	1	1
0	1	1	1	1
0			0	1

В ответе напишите буквы *w*,*x*,*y*,*z* в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т. д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Раскроем импликации

$(y \vee (z \equiv w)) \wedge ((\neg z \vee x) \equiv w)$

Раскроем равнозначности

$(y \vee (z \wedge w \vee \neg z \wedge \neg w)) \wedge ((\neg z \vee x) \wedge w \vee \neg(\neg z \vee x) \wedge \neg w)$

Раскроем скобки в правом подвыражении

$(y \vee z \wedge w \vee \neg z \wedge \neg w) \wedge (\neg z \wedge w \vee x \wedge w \vee z \wedge \neg x \wedge \neg w)$

Раскроем оставшиеся скобки

$y \wedge \neg z \wedge w \vee y \wedge x \wedge w \vee y \wedge z \wedge \neg x \wedge \neg w \vee z \wedge w \wedge \neg z \wedge w \vee z \wedge w \wedge x \wedge w \vee z \wedge w \wedge z \wedge \neg x \wedge \neg w \vee$

$\vee \neg z \wedge \neg w \wedge \neg z \wedge w \vee \neg z \wedge \neg w \wedge x \wedge w \vee \neg z \wedge \neg w \wedge z \wedge \neg x \wedge \neg w$

Сократим все произведения по закону противоречия и правилу идемподентности

$y \wedge \neg z \wedge w \vee y \wedge x \wedge w \vee y \wedge z \wedge \neg x \wedge \neg w \vee \sout{z \wedge w \wedge \neg z \wedge w} \vee z \wedge w \wedge x \sout{\wedge w} \vee \sout{z \wedge w \wedge z \wedge \neg x \wedge \neg w} \vee$

$\vee \sout{\neg z \wedge \neg w \wedge \neg z \wedge w} \vee \sout{\neg z \wedge \neg w \wedge x \wedge w} \vee \sout{\neg z \wedge \neg w \wedge z \wedge \neg x \wedge \neg w}$

Получим

$y \wedge \neg z \wedge w \vee y \wedge x \wedge w \vee y \wedge z \wedge \neg x \wedge \neg w \vee z \wedge w \wedge x$

Теперь мы без труда можем найти все наборы, приравняв единице каждое из полученных произведений

$y \wedge \neg z \wedge w$			
x	y	z	w
0	1	0	1
1	1	0	1

$y \wedge x \wedge w$			
x	y	z	w
1	1	0	1
1	1	1	1

$y \wedge z \wedge \neg x \wedge \neg w$			
x	y	z	w
0	1	1	0

$z \wedge w \wedge x$			
x	y	z	w
1	0	1	1
1	1	1	1

Итого имеем 5 **различных** наборов.

x	**y**	**z**	**w**
0	1	0	1
1	1	0	1
1	1	1	1
0	1	1	0
1	0	1	1

Сопоставим полученные наборы и фрагмент из задания

	x	y	z	w
1	0	1	0	1
2	1	1	0	1
3	1	1	1	1
4	0	1	1	0
5	1	0	1	1

	?	?	?	?	F
1	1	1	0	1	1
2	0	1	1	1	1
3	0			0	1

1 и 2 строки фрагмента из задания это 2 и 4 строки из полученного фрагмента (без точного соответствия). 3 строка в полученном фрагменте отсутствует в фрагменте из задания. В одном из столбцов должно быть два нуля. Следовательно, третьей строке из фрагмента соответствует 1 строка из полученного фрагмента.

	x	y	z	w
1	0	1	0	1
2	1	1	0	1
3	1	0	1	1

	?	?	?	?	F
1	1	1	0	1	1
2	0	1	1	1	1
3	0	1	1	0	1

w – столбец №2, z – столбец №1, y – столбец №3 (противоположные y значения), x – столбец №4.

Ответ: zwyx

Получение конъюнктивной нормальной формы

Данный метод применим для постановок, в которых в столбце результата все 0.

Суть метода в следующем – проинвертировать выражение, преобразовать результат в дизъюнктивную нормальную форму, проинвертировать полученную запись. Метод можно объяснить законом снятия двойного отрицания F -> ¬F = {преобразование} -> ¬¬F = F

> ***А.Богданов***
> Миша заполнял таблицу истинности логической функции: $(w \vee x \vee y) \rightarrow ((y \vee z) \wedge x \vee y \wedge (w \vee z))$, но успел заполнить лишь фрагмент из трёх различных её строк даже не указав, какому столбцу таблицы соответствует каждая из переменных *w*,*x*,*y*,*z*.
>
> | | | | | F |
> |---|---|---|---|---|
> | 0 | 0 | 0 | | 0 |
> | | 1 | 1 | | 0 |
> | | 1 | | 1 | 0 |
>
> Фрагмент таблицы истинности функции *F*, содержит неповторяющиеся строки. Определите, какому столбцу таблицы истинности функции *F* соответствует каждая из переменных *x*,*y*,*z*,*w*.

Раскроем импликацию

$\neg (w \vee x \vee y) \vee ((y \vee z) \wedge x \vee y \wedge (w \vee z))$

Проинвертируем выражение

$\neg (\neg(w \vee x \vee y) \vee ((y \vee z) \wedge x \vee y \wedge (w \vee z))) = \neg\neg(w \vee x \vee y) \wedge \neg((y \vee z) \wedge x \vee y \wedge (w \vee z))) = (w \vee x \vee y) \wedge (\neg((y \vee z) \wedge x) \wedge \neg (y \wedge (w \vee z)))) = (w \vee x \vee y) \wedge ((\neg(y \vee z) \vee \neg x) \wedge (\neg y \vee \neg (w \vee z))) = (w \vee x \vee y) \wedge (((\neg y \wedge \neg z) \vee \neg x) \wedge (\neg y \vee (\neg w \wedge \neg z))) = (w \vee x \vee y) \wedge (\neg y \wedge \neg z \vee \neg x) \wedge (\neg y \vee \neg w \wedge \neg z)$

Раскроем скобки

Для удобного внесения первого подвыражения в остальные скобки часть ч x и y внесем во второе подвыражение, часть с w – в третье.

$(w \vee x \vee y) \wedge (\neg y \wedge \neg z \vee \neg x) \wedge (\neg y \vee \neg w \wedge \neg z) = (x \vee y) \wedge (\neg y \wedge \neg z \vee \neg x) \wedge (\neg y \vee \neg w \wedge \neg z) \vee w \wedge (\neg y \wedge \neg z \vee \neg x) \wedge (\neg y \vee \neg w \wedge \neg z) = (\neg y \wedge \neg z \wedge x \vee \neg x \wedge y) \wedge (\neg y \vee \neg w \wedge \neg z) \vee (\neg y \wedge \neg z \vee \neg x) \wedge w \wedge \neg y$

$(\neg y \wedge \neg z \wedge x \vee \neg x \wedge y) \wedge (\neg y \vee \neg w \wedge \neg z) = \neg y \wedge \neg z \wedge x \vee \neg x \wedge y \wedge \neg w \wedge \neg z \vee x \wedge \neg y \wedge w \wedge \neg z$

$(\neg y \wedge \neg z \vee \neg x) \wedge w \wedge \neg y = \neg y \wedge \neg z \wedge w \vee \neg x \wedge w \wedge \neg y$

$\neg F = \neg y \wedge \neg z \wedge x \vee \neg x \wedge y \wedge \neg w \wedge \neg z \vee x \wedge \neg y \wedge w \wedge \neg z \vee \neg y \wedge \neg z \wedge w \vee \neg x \wedge w \wedge \neg y$

Инвертируем результат

$\neg (\neg y \wedge \neg z \wedge x \vee \neg x \wedge y \wedge \neg w \wedge \neg z \vee x \wedge \neg y \wedge w \wedge \neg z \vee \neg y \wedge \neg z \wedge w \vee \neg x \wedge w \wedge \neg y) = (y \vee z \vee \neg x) \wedge (x \vee \neg y \vee w \vee z) \wedge (\neg x \vee y \vee \neg w \vee z) \wedge (z \vee \neg w \vee y) \wedge (x \vee \neg w \vee y)$

Чтобы результат выражения был равен нулю, надо чтобы хотя бы одно подвыражение было ложным

$y \vee z \vee \neg x$			
x	y	z	w
1	0	0	0
1	0	0	1

$x \vee \neg y \vee w \vee z$			
x	y	z	w
0	1	0	0

$\neg x \vee y \vee \neg w \vee z$			
x	y	z	w
1	0	0	1

$z \vee \neg w \vee y$			
x	y	z	w
0	0	0	1
1	0	0	1

$x \vee \neg w \vee y$			
x	y	z	w
0	0	0	1
0	0	1	1

Итого 5 **различных** наборов.

x	y	z	w
1	0	0	0
1	0	0	1
0	1	0	0
0	0	0	1
0	0	1	1

Две единицы во втором столбце из фрагмента в условии могут быть только в столбце x или w. Однако строки, соответствующие строкам с 1 у x, не соответствуют фрагменту (1 единица в одной из строк). Следовательно, это строки 2 и 5, а первая строка в условии соответствует первой строке в полученном фрагменте (единица должна стоять над 1 в одной из следующих строк)

x	y	z	w
1	0	0	0
1	0	0	1
0	0	1	1

0	0	0	1
0	1	1	0
0	1	0	1

Столбец со всеми нулями – y, столбец с одной единицей – z, противоположный ему столбец – x, оставшийся – w.

Ответ: ywzx

Работа со свойствами таблицы истинности

Что может быть полезно:

1) В каждом столбце поровну 0 и 1,
2) Количество наборов в таблице 2^k, где k – количество переменных,
3) Если выражение сложное, то можно сначала обработать значения подвыражений, а затем работать со всем выражением полностью,
4) Обрабатывать строки из фрагмента и строки вне фрагмента отдельно,
5) Если спрашивают максимальное/минимальное количество строк с определенным значением, то для всех строк с неопределенными значениями можно принять значение, которое наиболее выгодно по условию задачи.

№80 из документа на сайте К.Ю.Полякова

Дан фрагмент таблицы истинности для выражения F:

x1	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**F**
0	0	1	1	0	0	1	0	0
0	1	0	0	1	1	0	1	1
0	0	0	0	1	1	1	1	1
1	0	1	0	1	1	0	1	1
0	1	1	1	0	1	0	0	1

Укажите максимально возможное число различных строк полной таблицы истинности этого выражения, в которых значение x6 не совпадает с F.

Всего строк в таблице истинности $2^8 = 256$. Значит в 128 строках x6 = 1, и в 128 строках x6 = 0. Во всех из них, кроме представленных во фрагменте, значение F может не совпадать с x6.

Поэтому подсчитаем, сколько строк во фрагменте не удовлетворяет условию. Все значения F совпадают с соответствующими им значениями x6. Следовательно, минимум 5 строк не удовлетворяют описанному условию.

Ответ: 256 – 5 = 251

Задание с сайта К.Ю.Полякова

Каждое логическое выражение A и B зависит от одного и того же набора из 6 переменных. В таблицах истинности каждого из этих выражений в столбце значений стоит ровно по 4 единицы. Каково минимально возможное число единиц в столбце значений таблицы истинности выражения $A \vee B$?

В данной задаче выражения A и B независимые. Наиболее выгодный случай для нас, когда и A и B одновременно принимают значение 1. То есть минимальное количество 1 для выражения $A \vee B$ в описанном случае может быть 4.

Ответ: 4

Про количество перестановок

Несколько вариантов расстановки столбцов

Программные методы решения

Запись выражения

Первое, с чем необходимо разобраться, это приоритет операций, которые мы будем использовать в языке программирования (ЯП). Этот пункт важен, так как обычно в ЯП не реализованы в полном смысле операции импликации и логической равнозначности.

Для наиболее популярных на ЕГЭ ЯП можно описать логические операции (и их эквиваленты) следующим образом.

Операция	Python	C++	PascalABC.net
Инверсия	not	!	not
Инверсия (эквивалент)	1 - a	1 - a	
Конъюнкция	and	&&	and
Дизъюнкция	or	||	or
Импликация (эквивалент)	<=	<=	<= (только логические значения)
Лог.равнозначность (эквивалент)	==	==	= (только логические значения)

Приписка «эквивалент» означает, что используемая операция имеет аналогичный логической операции результат, однако, в силу отличия таблиц приоритета нужно очень внимательно подходить к их применению.

Таблица приоритетов.

Операция	Алгебра логики	Python	C++	PascalABC.net
Инверсия	1	3	1	1
- (бинарный)		1	2	3
Конъюнкция	2	4	5	2
Дизъюнкция	3	5	6	3
Импликация/ <=	4	2	3	4
Равнозначность/ =	5	2	4	4

Как видно из таблицы, ни один ЯП не имеет необходимой таблицы приоритетов. Конкретно для использования лог.выражений с экиввалентными операциями удобнее всего использовать PascalABC.net, так как в нем необходимо контролировать только порядок выполнения операций сравнения. Так как на ЕГЭ обычно такие операции используются внутри скобочных групп, можно записать выражение без дополнительных манипуляций со скобками.

Сложнее всего следить за порядком операций на ЯП Python, так как если в одной скобочной группе используются и операции сравнения, и отрицание, необходимо с помощью дополнительных скобок строго устанавливать приоритет взятия отрицания выше операции сравнения.

Построение таблицы истинности

Перебор значений с помощью вложенных циклов

Выражение x → (¬y ≡ z)

Python

```
print("x y z F")
for x in 0, 1:
  for y in 0, 1:
    for z in 0, 1:
      # + перед выражением для преобразования True\False в 1\0
      print(x, y, z, +(x <= ((not y) == z)))
```

C++

```
cout << "x y z F" << endl;
for(int x=0; x <= 1; ++x)
  for(int y=0; y <= 1; ++y)
    for(int z=0; z<=1; ++z){
      cout << x << " " << y << " " << z << " ";
      cout << ((!x || y) <= z) << endl;
    }
```

PascalABC.net

```
println('x y z F')
for var x := False to True do
  for var y := False to True do
    for var z := False to True do
      // функция ord() нужна для преобразования в 0 или 1
      println(ord(x), ord(y), ord(z), ord(x <= (not y = z)))
// либо с помощью foreach
foreach var x in |False, True| do
  foreach var y in |False, True| do
    foreach var z in |False, True| do
      println(ord(x), ord(y), ord(z), ord(x <= (not y = z)))
```

Рекомендации по оформлению

Так как обычно необходимо найти наборы переменных для определенных значений функции, то удобнее сохранить значение в переменную. Также можно описать функцию, которая будет возвращать значение выражения. Лично я обычно использую функцию, потому что так удобнее сравнивать оригинальное выражение и выражение в задании.

Также такой способ гораздо безопаснее, так как не нужно контролировать приоритет операции сравнения с необходимым значением значения выражения (оборачивать логическое выражение в скобки).

Пример с переменной для ложного значения функции

Python

```
print("x y z F")
for x in 0, 1:
  for y in 0, 1:
    for z in 0, 1:
      f = x <= ((not y) == z)
      if f == 0: print(x, y, z, +f)
```

C++

```
cout << "x y z F" << endl;
for(int x=0; x <= 1; ++x)
  for(int y=0; y <= 1; ++y)
    for(int z=0; z <= 1; ++z){
      int F = (!x || y) <= z;
      if(f == 0)
```

```
        cout << x << " " << y << " " << z << " " << F << endl;
      }
```

PascalABC.net

```
println('x y z F')
for var x := False to True do
  for var y := False to True do
    for var z := False to True do begin
      var f := x <= (not y = z);
      if f = False then println(ord(x), ord(y), ord(z), ord(f))
    end;
```

Процедура на PascalABC.net

Неправильно было бы обойти вниманием специальную функцию TrueTablePrint, которая выводит на экран отформатированную таблицу истинности (полученную при помощи функции TrueTable).

Эти подпрограммы можно импортировать из встроенного модуля school.

```
###
uses school;
TrueTablePrint(TrueTable((x, y, z) -> x <= (not y = z)));
```

Функция TrueTablePrint также имеет еще один аргумент, для каких значений логического выражения необходимо вывести наборы значений переменных – 0 для ложных значений и 1 для истинных.

Так следующая запись выведет только те наборы, при которых функция будет истинной.

```
###
uses school;
TrueTablePrint(TrueTable((x, y, z) -> x <= (not y = z)), 1);
```

До версии 3.9.0.3379 у данного метода было ограничение на количество переменных – нельзя было построить таблицу с большим, чем 6 количеством переменных. В обозначенной выше версии добавили перегрузку данной функции, где первым параметром указывается количество переменных. Если в выражении до 6 переменных включительно, можно использовать описанный выше вариант формирования таблицы истинности.

```
### Пример для трех переменных
uses school;
TrueTablePrint(TrueTable(3, \(x, y, z) -> x <= (not y = z)), 1);
```

Универсальный решатор

Программный метод решения, предложенный 14 января 2022 года Хирьяновым Т.Ф.

Метод заключается в переборе всех возможных вариантов расстановки переменных для всех возможных вариантов заполнения пропусков во фрагменте таблице истинности. И поиск подходящих под условие перебранных вариантов. Программа будет представлена на ЯП Python, так как и предложено решение было на нем, и я не знаю, как это можно перенести на другой ЯП. Вероятно, в будущем данный материал будет дополнен аналогичными решаторами на других ЯП.

Рассмотрим сначала перебор всех вариантов расстановки переменных, после чего расширим алгоритм заполнением пропусков во фрагменте таблицы истинности.

Перебор вариантов расстановки переменных

> Миша заполнял таблицу истинности функции $F=((x\to y)\vee(z\equiv x))\wedge(w\to z)$, но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных *w*,*x*,*y*,*z*.
>
> | ? | ? | ? | ? | F |
> |---|---|---|---|---|
> | 0 | 0 | 1 | 1 | 1 |
> | 0 | 0 | 1 | 0 | 0 |
> | 0 | 1 | 1 | 1 | 0 |
>
> Определите, какому столбцу таблицы истинности функции *F* соответствует каждая из переменных *x*,*y*,*z*,*w*.
> В ответе напишите буквы *w*,*x*,*y*,*z* в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Для удобства решения вводится функция, которая вычисляет значение выражения.

```
def f(x, y, z, w):
  return ((x <= y) or (z == x)) and (w <= z)
```

Для перебора всех вариантов расстановки переменных воспользуемся функцией permutations из встроенного модуля itertools.

```
from itertools import permutations

# описываем наборы из фрагмента таблицы истинности
table = [(0, 0, 1, 1), (0, 0, 1, 0), (0, 1, 1, 1)]
# соответствующие строкам выше значения функции
# (порядок такой же, как в table)
res = [1, 0, 0]
# перебираем все перестановки имен переменных
for p in permutations('xyzw'):
  # список для хранения значений функций для всех строк table
  resP = []
  # перебираем все строки
  for row in table:
     # формируем пары (имя переменной, значение)
     # для ('x', 'z', 'w', 'y') и (0, 0, 1, 0) получится
     # (('x', 0), ('z', 0), ('w', 1), ('y', 0))
```

```
      pairs = zip(p, row)
      # собираем из полученных пар словарь
      # для (('x', 0), ('z', 0), ('w', 1), ('y', 0)) получится
      # {'x': 0, 'z': 0, 'w': 1, 'y': 0}
      d_pars = dict(pairs)
      # передаем ключ-значения в функцию, как значения аргументов
      # для {'x': 0, 'z': 0, 'w': 1, 'y': 0} получится
      # f(x=0, z=0, w=1, y=0)
      r = f(**d_pars)
      # добавляем значение в итоговый список
      resP.append(r)
  # если список результатов для перестановки и таблицы равен res
  # значит найдена нужная перестановка
  if resP == res:
    print(*p, sep ='')
```

Формирование списка с результатами можно выполнить с помощью list-comprehension и записать следующим образом.

```
    resP = [f(**dict(zip(p, row))) for row in table]
```

Соответственно, можно сразу сравнить результат этого выражения с res

```
    if [f(**dict(zip(p, row))) for row in table] == res:
      print(*p, sep='')
```

Короткая версия представленного выше алгоритма.

```
def f(x, y, z, w):
  return ((x <= y) or (z == x)) and (w <= z)

from itertools import permutations

table = [(0, 0, 1, 1), (0, 0, 1, 0), (0, 1, 1, 1)]
for p in permutations('xyzw'):
  if [f(**dict(zip(p, row))) for row in table] == [1, 0, 0]:
    print(*p, sep ='')
```

Перебор пропусков во фрагменте таблицы истинности

Демоверсия 2024 (Уровень: Базовый)

Миша заполнял таблицу истинности логической функции $F=(x \wedge \neg y) \vee (y \equiv z) \vee \neg w$, но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных w,x,y,z.

				F
		0	0	0
1	0		0	0
1	0	1		0

Определите, какому столбцу таблицы соответствует каждая из переменных w,x,y,z.

```
def f(x, y, z, w):
  return (x and not y) or (y == z) or not w
```

Для перебора пропусков сгенерируем все возможные варианты заполнения соответствующих ячеек с помощью функции product из модуля itertools. Для этого можно сначала описать таблицу с пропусками, а затем перебрать указанные на местах пропуска имена в цикле.

```
from itertools import permutations, product
# перебор всех возможных вариантов заполнения пропусков
for a1, a2, a3, a4 in product([0, 1], repeat=4):
  # формирование таблицы с заполненными пропусками
  table = [(a1, a2, 0, 0), (1, 0, a3, 0), (1, 0, 1, a4)]
  # если количество строк в таблице не равно количеству уникальных
  # значит есть дублирующиеся строки
  if len(table) != len(set(table)):
    # поэтому перебираем следующий вариант заполнения пропусков
    continue
  for p in permutations('xyzw'):
    if [f(**dict(zip(p, row))) for row in table] == [0, 0, 0]:
      print(*p, sep ='')
```

Такой алгоритм позволяет решить задачи уровня ЕГЭ и более сложные постановки, в том числе и с количеством возможных перестановок.

С.Чайкин

Логическая функция F задаётся выражением $x \to y \wedge z$. На рисунке приведён фрагмент таблицы истинности функции F, содержащий неповторяющиеся наборы аргументов.

?	?	?	F
0	1	0	0
1	1	0	0

Определите, сколько существует различных способов расстановки переменных *x*,*y*,*z*, подходящих для данной таблицы истинности?

```
def f(x, y, z):
  return x <= (y and z)

ps = set()
from itertools import permutations
table = [(0, 1, 0), (1, 1, 0)]
for p in permutations('xyz'):
  if [f(**dict(zip(p, row))) for row in table] == [0, 0]:
    ps.add(p)

print(len(ps))
```

Построение таблицы истинности в электронных таблицах

Этот способ уже для сильных духом, кто по каким-то причинам категорически отказывается от программирования. Что ж, и для вас есть автоматизированный способ построения таблицы истинности.

Во-первых, логические функции в эл.таблицах реализованы в виде функций. То есть в эл.таблицах нет логических операторов, привычных по аналитическому и программному решениям.

Логическая функция	Реализация в эл.таблицах	Лог.функция/эквивалент
¬A	НЕ(A)	Лог.функция
$A \wedge B$	И(A; B)	Лог.функция
$A \vee B$	ИЛИ(A; B)	Лог.функция
$A \rightarrow B$	A <= B	Эквивалент
$A \equiv B$	A = B	Эквивалент

Также стоит помнить, что в эл.таблицах приоритет выполнения операторов сравнения одинаковый.

Например, выражение $F=(x \rightarrow (z \equiv w)) \vee \neg(y \rightarrow w)$, будет записано с помощью формулы в эл.таблице как =ИЛИ(x <= (z = w); НЕ(y <= w)), где на месте переменных будут соответствующие переменным адреса ячеек.

Также важно помнить, что значения должны быть именно логическими, иначе придется учитывать ранги типов данных. Так, например, всегда выполняется следующее отношение, не зависимо от значений соответствующих частей, boolean > string > number или date.

На мой взгляд сильно сложнее, чем записать выражение на любом ЯП. Но на вкус и цвет все фломастеры разные 😊

Заполнение таблицы истинности

Можно заполнить вручную, зачастую это будет, пожалуй, даже проще. Но можно и автоматизировать этот процесс. Пример будем приводить на построении таблицы истинности для выражения $F=(x\rightarrow(z\equiv w))\vee\neg(y\rightarrow w)$

Способ 1

1) Заполнить столбец от 0 до 2^{k-1}, где k – количество переменных.
2) Написать все степени 2 от нулевой до (k-1) в первой строке
3) С помощью операций ОСТАТ и ЧАСТНОЕ найти все наборы переменных
4) Расписать по действиям выражение в столбцах
5) Заполнить каждое действие

Такой способ позволяет как не озадачиваться рангами данных, так и гораздо проще реализовывать лог.операции отдельно и все выражение целиком, так как нет необходимости переводить инфиксное представление в функциональное.

1)

	A	B
1		
2	0	
3	1	
4	2	
5	3	
6	4	
7	5	
8	6	
9	7	
10	8	
11	9	
12	10	
13	11	
14	12	
15	13	
16	14	
17	15	

	A	B	C	D	E
1		8	4	2	1
2					
3	0				
4	1				
5	2				
6	3				
7	4				
8	5				
9	6				
10	7				
11	8				
12	9				
13	10				
14	11				
15	12				
16	13				
17	14				
18	15				

2)

3) Формула заключается в нахождении i-го слева разряда в двоичном числе B3=ОСТАТ(ЧАСТНОЕ($A3; B$1); 2). Адреса зафиксированы так, чтобы формулу можно было скопировать во все ячейки таблиц и работа осуществлялась всегда с числом из первого столбца и делителем (степенью 2) из первой строки.

	A	B	C	D	E
1		8	4	2	1
2		x	y	z	w
3	0	0	0	0	0
4	1	0	0	0	1
5	2	0	0	1	0
6	3	0	0	1	1
7	4	0	1	0	0
8	5	0	1	0	1
9	6	0	1	1	0
10	7	0	1	1	1
11	8	1	0	0	0
12	9	1	0	0	1
13	10	1	0	1	0
14	11	1	0	1	1
15	12	1	1	0	0
16	13	1	1	0	1
17	14	1	1	1	0
18	15	1	1	1	1

4) Расписываем выражение по действиям (первая строка для 0).
   ($z \equiv w$) F3 =D3=E3
   ($x \to (z \equiv w)$) G3 =И(B3) <= F3
   ($y \to w$) H3 =C3 <= E3
   $\neg(y \to w)$ I3 =НЕ(H3)
   $(x \to (z \equiv w)) \vee \neg(y \to w)$ J3 =ИЛИ(G3; I3)
5) Заполнить все ячейки

	A	B	C	D	E	F	G	H	I	J
1		8	4	2	1					
2		x	y	z	w	(z≡w)	(x→(z≡w))	(y→w)	¬(y→w)	(x→(z≡w))∨¬(y→w)
3	0	0	0	0	0	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
4	1	0	0	0	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
5	2	0	0	1	0	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
6	3	0	0	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
7	4	0	1	0	0	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
8	5	0	1	0	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
9	6	0	1	1	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
10	7	0	1	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
11	8	1	0	0	0	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
12	9	1	0	0	1	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
13	10	1	0	1	0	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
14	11	1	0	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
15	12	1	1	0	0	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
16	13	1	1	0	1	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
17	14	1	1	1	0	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ИСТИНА
18	15	1	1	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА

Способ 2

Алгоритм:

1) Заполнить столбец значениями от 0 до 2^{k-1}, где k – количество переменных.
2) С помощью функции =ДЕС.В.ДВ(значение; разрядность) представить значение из п.1 как k-разрядное двоичное число.
3) С помощью меню разбивки по столбцам разделить на отдельные символы.
4) Перевести строковые значения в логические
5) Записать формулу
6) Скопировать строку с формулами на весь диапазон

1)

	A	B
1		
2	0	
3	1	
4	2	
5	3	
6	4	
7	5	
8	6	
9	7	
10	8	
11	9	
12	10	
13	11	
14	12	
15	13	
16	14	
17	15	

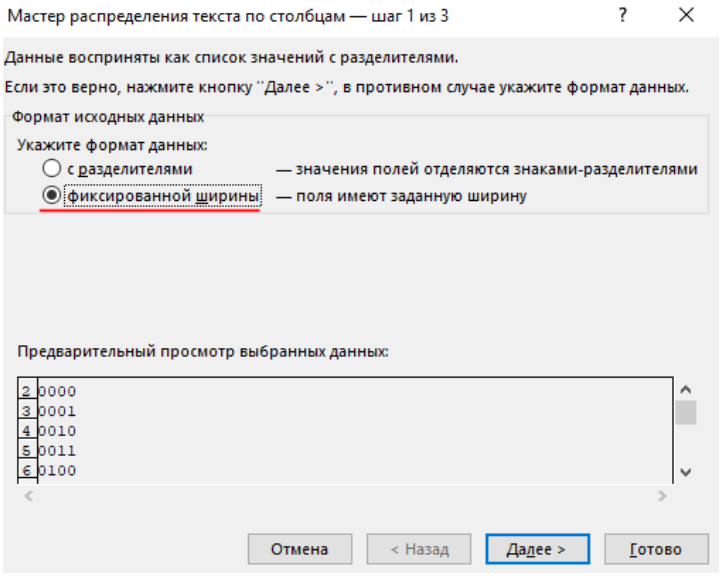
2)

	A	B	C
1			
2	0	0000	
3	1	0001	
4	2	0010	
5	3	0011	
6	4	0100	
7	5	0101	
8	6	0110	
9	7	0111	
10	8	1000	
11	9	1001	
12	10	1010	
13	11	1011	
14	12	1100	
15	13	1101	
16	14	1110	
17	15	1111	

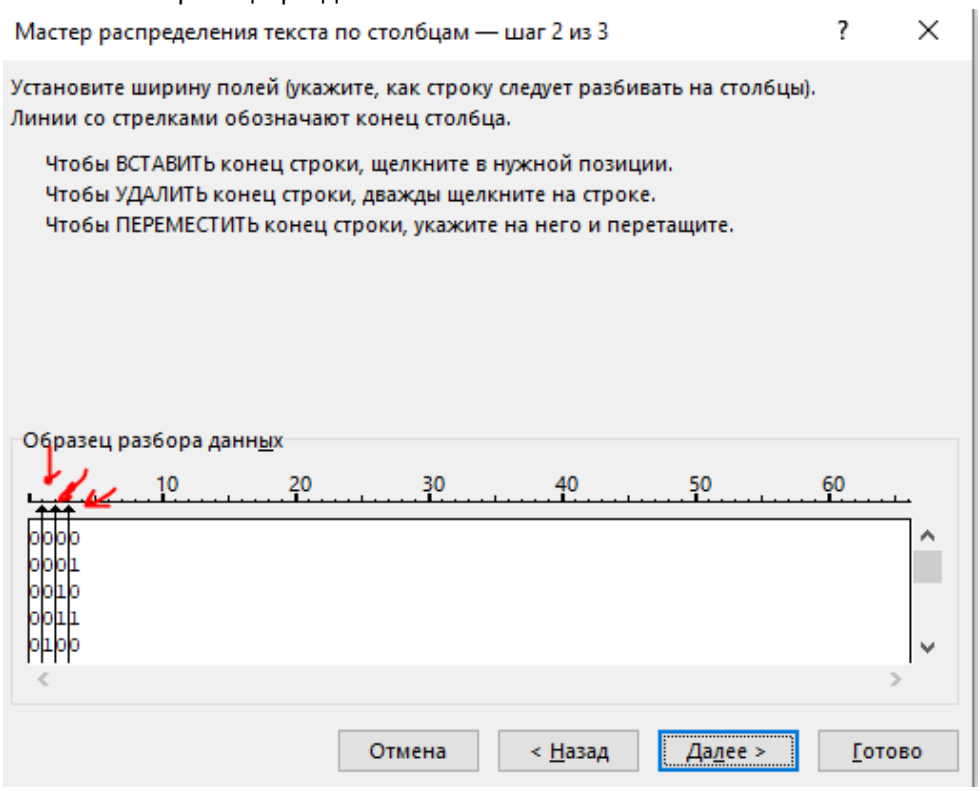
1			
2	0	0000	0000
3	1	0001	0001
4	2	0010	0010
5	3	0011	0011
6	4	0100	0100
7	5	0101	0101
8	6	0110	0110
9	7	0111	0111
10	8	1000	1000
11	9	1001	1001
12	10	1010	1010
13	11	1011	1011
14	12	1100	1100
15	13	1101	1101
16	14	1110	1110
17	15	1111	1111

3) Копируем только значения

Выбираем поля фиксированной длины



Расставляем границы разделения



4)

	A	B	C	D	E	F
1						
2	0	0000	0	0	0	0
3	1	0001	0	0	0	1
4	2	0010	0	0	1	0
5	3	0011	0	0	1	1
6	4	0100	0	1	0	0
7	5	0101	0	1	0	1
8	6	0110	0	1	1	0
9	7	0111	0	1	1	1
10	8	1000	1	0	0	0
11	9	1001	1	0	0	1
12	10	1010	1	0	1	0
13	11	1011	1	0	1	1
14	12	1100	1	1	0	0
15	13	1101	1	1	0	1
16	14	1110	1	1	1	0
17	15	1111	1	1	1	1

5) С помощью, например, функции И преобразуем строковые значения в логические

1										
2	0	0000	0	0	0	0	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
3	1	0001	0	0	0	1	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА
4	2	0010	0	0	1	0	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ
5	3	0011	0	0	1	1	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ИСТИНА
6	4	0100	0	1	0	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
7	5	0101	0	1	0	1	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ИСТИНА
8	6	0110	0	1	1	0	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ
9	7	0111	0	1	1	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ИСТИНА
10	8	1000	1	0	0	0	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
11	9	1001	1	0	0	1	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА
12	10	1010	1	0	1	0	ИСТИНА	ЛОЖЬ	ИСТИНА	ЛОЖЬ
13	11	1011	1	0	1	1	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
14	12	1100	1	1	0	0	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
15	13	1101	1	1	0	1	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
16	14	1110	1	1	1	0	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ
17	15	1111	1	1	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА

6) Записываем формулу

1										
2	0	0000	0	0	0	0	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА
3	1	0001	0	0	0	1	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ИСТИНА
4	2	0010	0	0	1	0	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ИСТИНА
5	3	0011	0	0	1	1	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ИСТИНА	ИСТИНА
6	4	0100	0	1	0	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА
7	5	0101	0	1	0	1	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
8	6	0110	0	1	1	0	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
9	7	0111	0	1	1	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА
10	8	1000	1	0	0	0	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА
11	9	1001	1	0	0	1	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ
12	10	1010	1	0	1	0	ИСТИНА	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
13	11	1011	1	0	1	1	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА	ИСТИНА
14	12	1100	1	1	0	0	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА
15	13	1101	1	1	0	1	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА	ЛОЖЬ
16	14	1110	1	1	1	0	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ИСТИНА
17	15	1111	1	1	1	1	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА

ВАЖНО: если не произвести преобразование в логический тип, то формула будет работать неправильно.

Пример неправильной работы при той же функции

	A	B	C	D	E	F	G
1							
2	0	0000	0	0	0	0	ИСТИНА
3	1	0001	0	0	0	1	ИСТИНА
4	2	0010	0	0	1	0	ИСТИНА
5	3	0011	0	0	1	1	ИСТИНА
6	4	0100	0	1	0	0	ИСТИНА
7	5	0101	0	1	0	1	ИСТИНА
8	6	0110	0	1	1	0	ИСТИНА
9	7	0111	0	1	1	1	ИСТИНА
10	8	1000	1	0	0	0	ИСТИНА
11	9	1001	1	0	0	1	ИСТИНА
12	10	1010	1	0	1	0	ИСТИНА
13	11	1011	1	0	1	1	ИСТИНА
14	12	1100	1	1	0	0	ИСТИНА
15	13	1101	1	1	0	1	ИСТИНА
16	14	1110	1	1	1	0	ИСТИНА
17	15	1111	1	1	1	1	ИСТИНА

Единственное удобство, которое я могу выделить, это возможность фильтровать по значению и копировать. Но ведь можно скопировать вывод программы и также вставить в таблицу.