

Содержание

1	Задание 1. Анализ информационных моделей	3
1.1	Спецификация	3
1.2	Теория	3
1.3	Однозначное соотнесение таблицы и графа	4
1.4	Неоднозначное соотнесение таблицы и графа	9
2	Задание 2. Построение таблиц истинности логических выражений	12
2.1	Спецификация	12
2.2	Теория	12
2.2.1	Логическое отрицание	13
2.2.2	Логическое сложение	13
2.2.3	Логическое умножение	13
2.2.4	Логическая функция	13
2.2.5	Таблица истинности	14
2.2.6	Импликация	15
2.2.7	Эквиваленция	16
2.2.8	Исключающее ИЛИ	16
2.2.9	Преобразования и правила	17
2.2.10	Булева логика в Python	17
2.2.11	Перебор значений	18
2.3	Строки с пропущенными значениями	18
3	Задание 3. Поиск информации в реляционных базах данных	27
3.1	Спецификация	27
3.2	Теория	27
3.2.1	Знакомство с редактором таблиц	27
3.2.2	Формулы	29
3.2.3	Реляционные базы данных	31
3.2.4	Фильтры	32
3.3	Задания для подготовки	33
4	Задание 14. Кодирование чисел. Системы счисления	34
4.1	Теория	34
4.1.1	Унарная СС	34
4.1.2	Непозиционная СС	34
4.1.3	Позиционная СС	35
4.1.4	Системы счисления с основанием больше 10	36
4.1.5	Перевод из n-ой системы счисления в 10-ую	37
4.1.6	Перевод из 10-ой системы счисления в n-ую	37
4.1.7	Перевод из n-ой СС в m-ую СС	38
4.1.8	Связь 2-ой, 4-ой, 8-ой и 16-ой систем счислений	38
4.1.9	Системы счисления в Python	39
4.2	Операции в разных СС с одной переменной	41

4.3	Операции в разных СС с двумя переменными	43
4.4	Операции в одной СС	46

1 Задание 1. Анализ информационных моделей

1.1 Спецификация

Задание проверяет: Умение представлять и считывать данные в разных типах информационных моделей (схемы, карты, таблицы, графики и формулы)

Уровень сложности: Базовый

Максимальный балл: 1

Примерное время: 3 минуты (1,5 минуты)

1.2 Теория

Определение 1.1. *Граф — множество точек (вершин, узлов), которые соединяются множеством линий (ребрами).*

Примером графа из реального мира является карта страны: города это вершины, дороги между ними — ребра. Графы могут быть ориентированными и неориентированными.

Определение 1.2. *Ориентированный граф — граф, в котором у каждого ребра есть направление. Ребра неориентированного графа не имеют направлений.*

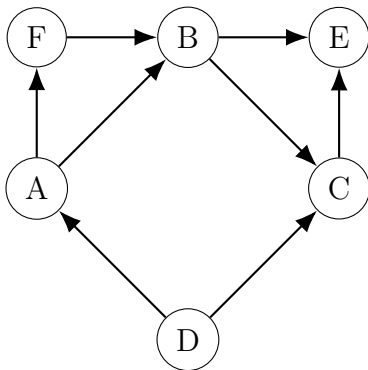


Рис. 1.1: Ориентированный граф

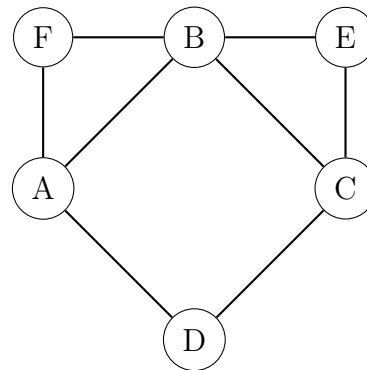


Рис. 1.2: Неориентированный граф

Определение 1.3. *Матрица смежности — представление графа в виде квадратной матрицы, где каждая строка и столбец соответствуют вершинам графа. Значение в определенной ячейке может соответствовать расстоянию или говорить о наличии/отсутствии ребра между вершинами.*

Например, матрица смежности для Рис. 1.2 может выглядеть следующим образом: Как пример, данная таблица показывает, что расстояние между пунктами А и D равняется 4, а между В и F — 7. Отсутствие значения в ячейке, говорит об отсутствии ребра между точками (напр. А и С).

Обратите внимание, что данная матрица смежности симметрична относительно диагонали, это, как правило, говорит, что граф — неориентированный.

Стоит держать в голове, что расстояния, данные в матрице смежности не должны быть пропорциональны длинам ребер в графе.

	a	b	c	d	e	f
a	-	12		4		5
b	12	-	2		9	7
c		2	-	6	10	
d	4		6	-		
e		9	10		-	
f	5	7				-

Таблица 1.1: Матрица смежности для Рис. 1.2

Определение 1.4. *Степень вершины графа (в неориентированном) — количество ребер, выходящих из вершины. В ориентированном, выделяют степени входа и выхода, количества входящих и выходящих ребер, соответственно. Для вершины V степень будем записывать как $\deg(V)$.*

В заданиях первого типа дается матрица смежности, которую необходимо сопоставить с данным графом. При этом, есть графы и матрицы смежности, которые могут сопоставлены единственным образом (1.3) и те, которые не сопоставляются единственным образом (1.4). Однако в обоих случаях есть возможность прийти к ответу на вопрос задачи.

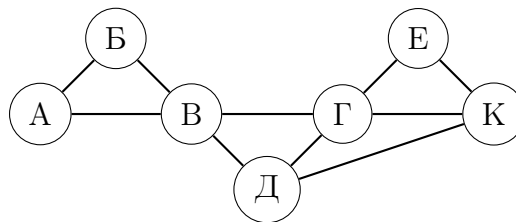
1.3 Однозначное соотнесение таблицы и графа

Посмотрим на задачу данного подтипа:

Задание 1.3.1

На рисунке справа схема дорог Н-ского района изображена в виде графа; в таблице слева содержатся сведения о протяжённости каждой из этих дорог (в километрах).

	П1	П2	П3	П4	П5	П6	П7
П1	-	11	5		12		
П2	11	-	8	15		23	
П3	5	8	-		10		7
П4		15		-		10	
П5	12		10		-		11
П6		23		10		-	
П7			7		11		-

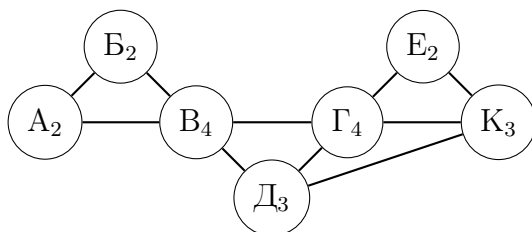


Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова длина дороги из пункта **В** в пункт **Г**. В ответе запишите целое число — так, как оно указано в таблице.

(Источник: РешуЕГЭ)

Решение 1.3.1

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



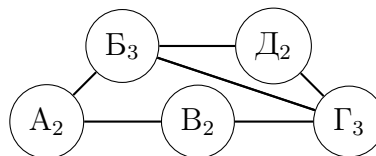
Как правило, в задачах данного типа, в графе существуют вершины, степени которых отличаются от большинства остальных. В данной задаче, можно увидеть, что вершины В и Г - единственные, чьи степени равняются 4. Значит, в матрице смежности два пункта, имеющие по четыре значения в строке, соответствуют этим пунктам.

Не трудно заметить, что пункты П2 и П3 имеют по 4 значения в строчках. Так как, требуется лишь определить расстояние от В до Г, нам необязательно устанавливать четкое соответствие между ними (не важно, ищем ли мы расстояние от П2 до П3 или от П3 до П2). Из таблицы мы видим, что расстояние между этими пунктами равно 8.

Ответ: 8

Решение заданий данного типа строится на поиске взаимосвязей в графе и матрице смежности. Самой очевидной взаимосвязью являются вершины с уникальными степенями, их всегда легко увидеть. Следующим шагом постарайтесь найти взаимосвязи между конкретными вершинами. Представим, что дан следующий граф и матрица смежности к нему:

	П1	П2	П3	П4	П5
П1	-	11	5		
П2	11	-	8	15	
П3	5	8	-		1
П4		15		-	9
П5			1	9	-



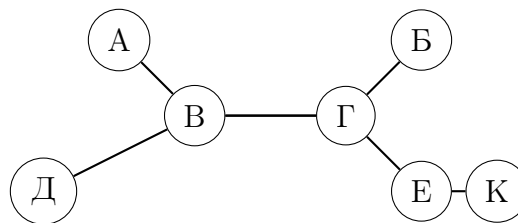
Несложно, догадаться, что вершины В и Г соответствуют пунктам П2 и П3 (или наоборот), так как это единственные, чьи степени равны 3 (по три записи в таблице). Однако, чтобы восстановить соответствие, этого недостаточно. Посмотрев на остальные вершины (А, В и Д), можно заметить, что только одна из них соединена с обеими вершинами, степени которых равны 3, это вершина Д. Найдем в таблице пункт, который соединён только с П2 и П3 - это П1, следовательно, П1 = Д.

Этот пример иллюстрирует, каким образом мы можем находить соответствия. Решим еще пару задач.

Задание 1.3.2

На рисунке схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о длине этих дорог в километрах:

	П1	П2	П3	П4	П5	П6	П7
П1	-		10				
П2		-	20				
П3	10	20	-	8			
П4			8	-	15	12	
П5				15	-		
П6				12		-	18
П7						18	-

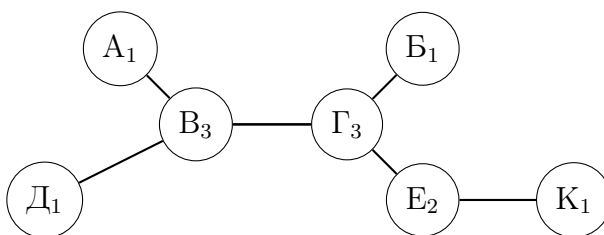


Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите длину дороги из пункта **Г** в пункт **Е**. **ВНИМАНИЕ!** Длины отрезков на схеме не отражают длины дорог.

(Источник: РешуЕГЭ)

Решение 1.3.2

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



Очевидно, вершиной с уникальной степенью (2) является Е. В таблице ей соответствует П6. Обратим внимание, что только вершины В и Г имеют степени 3, при этом, только одна из них соединена с Е. Найдем в таблице вершину, которая имеет три ребра и соединена с П6 (Е) - П4. Следовательно, вершине Г соответствует П4.

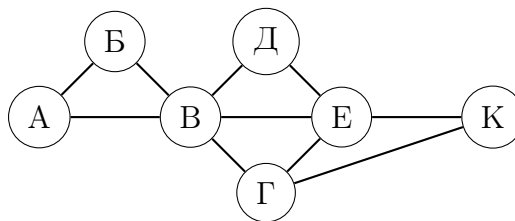
Обе вершины (Г и Е) установлены, из таблицы найдем расстояние между ними (П4 и П6) - 12.

Ответ: 12

Задание 1.3.3

На рисунке схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о длине этих дорог в километрах:

	П1	П2	П3	П4	П5	П6	П7
П1	-	45	-	10	-	-	-
П2	45	-	-	40	-	55	-
П3	-	-	-	-	15	60	-
П4	10	40	-	-	-	20	35
П5	-	-	15	-	-	55	-
П6	-	55	60	20	55	-	45
П7	-	-	-	35	-	45	-

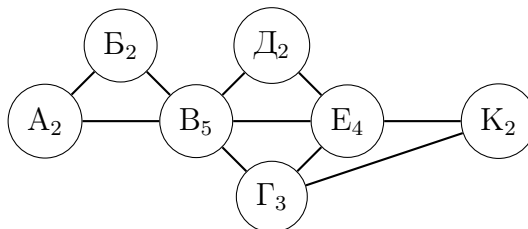


Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова длина дороги из пункта **К** в пункт **Е**. В ответе запишите целое число – так, как оно указано в таблице.

(Источник: РешуЕГЭ)

Решение 1.3.3

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



Обе вершины Г и Е имеют уникальные степени 3 и 4, соответственно. В таблице это пункты П2 и П4. Так как вершина К только соединена с Е и Г, в матрице достаточно найти пункт, соединяющийся только с П2 и П4 – это П1.

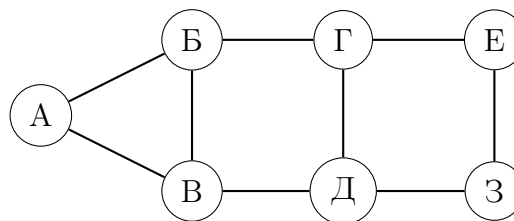
Обе вершины (К и Е) установлены, из таблицы найдем расстояние между ними (П4 и П1) – 10.

Ответ: 10

Задание 1.3.4

На рисунке справа схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о длинах этих дорог (в километрах).

	П1	П2	П3	П4	П5	П6	П7
П1	-					12	7
П2		-			10	11	9
П3			-	5	6	3	
П4			5	-	15		
П5		10	6	15	-		
П6	12	11	3			-	
П7	7	9					-

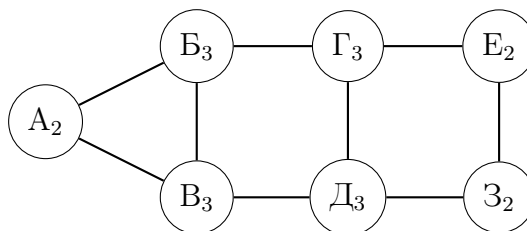


Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. В таблице в левом столбце указаны номера пунктов, откуда совершается движение, в первой строке — куда. Найдите сумму длин дорог из пункта Г в пункт Е и из пункта Д в З:

(Источник: РешуЕГЭ)

Решение 1.3.4

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



Вершины А, Е и З имеют уникальные степени (2) (в таблице: П1, П4, П7), однако заметим, что из всех трех, только вершина А не имеет общего ребра с вершиной, у которой степень тоже 2 (Е и З соединены между собой). Соответственно, в таблице это - П4.

Тогда, пункты П1 и П7 соответствуют Е и З (без понимания, кто есть кто). Тогда по таблице нам известно, что есть соответствие между ребрами ГЕ, ЕЗ, ЗД и П7-П1, П7-П2, П1-П6). Сложим три значения из таблицы ($12 + 7 + 9 = 28$) и получим суммарную длину трех ребер. Однако ребро ЕЗ нас не интересует, но его мы можем установить наверняка П1-П7. Тогда $28 - 7 = 21$

Ответ: 21

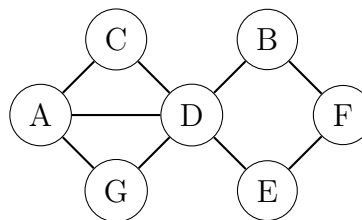
1.4 Неоднозначное соотнесение таблицы и графа

В задачах этого типа, как правило, матрица смежности заполняется звёздочками на тех местах, где подразумевается ребро. Разберем пример:

Задание 1.4.1

На рисунке слева изображена схема дорог N-ского района. В таблице звёздочкой обозначено наличие дороги из одного населённого пункта в другой. Отсутствие звёздочки означает, что такой дороги нет.

	1	2	3	4	5	6	7
1	-		*				*
2		-			*		*
3	*		-	*			*
4			*	-			*
5		*			-	*	
6					*	-	*
7	*	*	*	*		*	-

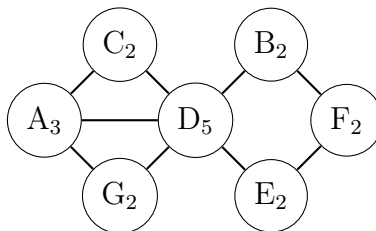


Каждому населённому пункту на схеме соответствует его номер в таблице, но неизвестно какой именно номер. Определите, какие номера населённых пунктов в таблице могут соответствовать населённым пунктам **В** и **Е** на схеме. В ответе запишите эти два номера в возрастающем порядке без пробелов и знаков препинания.

(Источник: РешуЕГЭ)

Решение 1.4.1

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



Вершина D - единственная, чья степень равна 5. Из таблицы, очевидно, что ей соответствует 7. Таким же образом, можно установить, что вершине A соответствует 3.

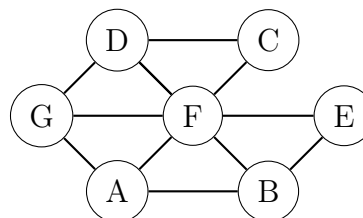
Обратим внимание, что интересующие нас вершины B и E соединены ребром с G (7) и не соединены с A (3), это свойство есть только у них. Найдём по таблице, для каких пунктов это выполняется: 2 и 6.

Ответ: 26

Задание 1.4.2

На рисунке слева изображена схема дорог Н-ского района, в таблице звёздочкой обозначено наличие дороги из одного населённого пункта в другой. Отсутствие звёздочки означает, что такой дороги нет.

	1	2	3	4	5	6	7
1	-		*	*			*
2		-	*		*	*	
3	*	*	-	*	*	*	*
4	*		*	-			
5		*	*		-		
6		*	*			-	*
7	*		*			*	-

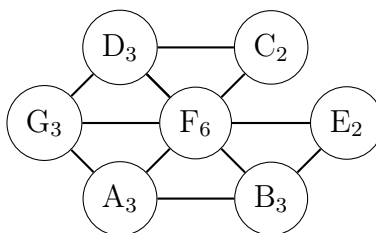


Каждому населённому пункту на схеме соответствует его номер в таблице, но неизвестно какой именно номер. Определите, какие номера населённых пунктов в таблице могут соответствовать населённым пунктам **A** и **G** на схеме. В ответе запишите эти два номера в возрастающем порядке без пробелов и знаков препинания.

(Источник: РешуЕГЭ)

Решение 1.4.2

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



Вершина D - единственная, чья степень равна 6. Из таблицы, очевидно, что ей соответствует 3. Таким же образом, можно установить, что вершинам C и E соответствуют пункты 4 и 5 (или наоборот).

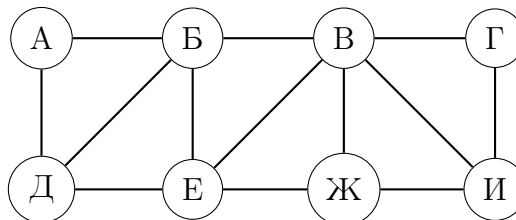
Обратим внимание, что интересующие нас вершины A и G соединены ребром только с теми вершинами, степени, которых больше или равны 3, следовательно, интересующие нас пункты не имеют ребер с пунктами 4 и 5 - это пункты 6 и 7.

Ответ: 67

Задание 1.4.3

На рисунке слева изображена схема дорог Н-ского района, в таблице звёздочкой обозначено наличие дороги из одного населённого пункта в другой. Отсутствие звёздочки означает, что такой дороги нет.

	П1	П2	П3	П4	П5	П6	П7	П8
П1	-			15	29	31		
П2		-	18		30		25	
П3		18	-				33	24
П4	15			-		21		
П5	29	30			-	14		27
П6	31			21	14	-	23	
П7		25	33			23	-	12
П8			24		27		12	-

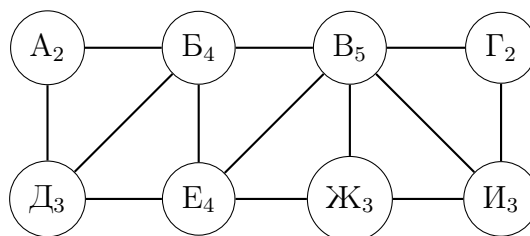


Так как таблицу и схему рисовали независимо друг от друга, нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Известно, что одна дорога в таблице отмечена неверно: из двух пунктов, которые соединяет эта дорога, правильно указан только один. В результате в одном из пунктов в таблице одной дороги не хватает, а в другом — появилась лишняя дорога. Определите длину дороги **ГИ**.

(Источник: РешуЕГЭ)

Решение 1.4.3

Решение всех задач данного типа должно начинаться с подсчитывания степени каждой вершины:



В таблице 1 пункт, в котором 2 дороги, 4 пункта, в которых по 3 дороги, 3 пункта, в которых по 4 дороги и 0 пунктов, в которых 5 дорог.

Пункту П4 соответствует А, тогда П6 соответствует Б, а П1 соответствует Д. Б и Д оба ведут в П5, значит, П5 соответствует Е. П7 имеет 4 дороги, очевидно, что это и есть пункт с недостающей дорогой, а сам П7 соответствует В.

Теперь возможны 2 варианта. Либо П8 соответствует Ж и в П2 лишняя дорога, либо П2 соответствует Ж и в П8 лишняя дорога. Если П8 это Ж, то П3 соответствует И. Тогда П2 соответствует Г, и расстояние ГИ равно 18. Если П2 это Ж, то П3 соответствует И. Тогда П8 соответствует Г, и расстояние ГИ равно 24.

Ответ: 24 или 18

2 Задание 2. Построение таблиц истинности логических выражений

2.1 Спецификация

Задание проверяет: Умение строить таблицы истинности и логические схемы

Уровень сложности: Базовый

Максимальный балл: 1

Примерное время: 3 минуты (1,5 минуты)

2.2 Теория

Определение 2.1. Булева алгебра – это раздел математической логики, в котором переменные могут иметь всего два значения: «истина» (обычно обозначается как 1) и «ложь» (обычно обозначается как 0). Булева алгебра также известна как булева логика, двоичная логика, двоичная алгебра, алгебра высказываний.

Основными объектами булевой алгебры являются логические высказывания.

Определение 2.2. Логическое высказывание — это повествовательное предложение, в отношении которого можно однозначно сказать, истинно (1) оно или ложно (0).

При этом, главным образом нас интересует значение выражение (истина или ложь), а не предмет высказывания. Пример высказывания: *На улице идет дождь*. Это выражение истинно, если на улице действительно идет дождь и ложно, если это не так.

В какой-то момент, при решении простейших арифметических задач мы начали использовать *переменные*, ведь гораздо удобнее записать просто x вместо *данное кол-во яблок*, поступим таким же образом.

Определение 2.3. Логические переменные – это переменные, которые могут принимать только два значения: истина (1) или ложь (0).

Далее мы будем работать именно с логическими переменными, как правило, они обозначаются заглавными латинскими буквами, например: A

Вспомним, что в начальной школе мы не остановились на том, что числа просто существуют, вместо этого мы продолжили изучать *операции* над числами. Так нас научили их складывать, вычитать, перемножать и тд. Последуем этому примеру и изучим основные операции над высказываниями, их всего три:

- Логическое отрицание (инверсия)
- Логическое сложение (дизъюнкция)
- Логическое умножение (конъюнкция)

2.2.1 Логическое отрицание

Определение 2.4. *Логическое отрицание — унарная операция над высказыванием, результатом которой является высказывание, «противоположное» исходному по значению.*

Логическое сложение аналогично поведению обычного "не" в повседневном языке: если значение высказывания было истинным (1), то после применения данной операции получится ложь (0), и наоборот. Например, для высказывания *На улице идет дождь*, логическим отрицанием будет высказывание *На улице НЕ идет дождь*.

Слово *унарная* говорит о возможности применения этой операции лишь к одному высказыванию. Эту операцию так же называют *инверсией* или *логическим НЕ*.

Есть несколько способов записи: $\neg A$; $\sim A$; \bar{A} ; *not A*

2.2.2 Логическое сложение

Определение 2.5. *Логическое сложение — это бинарная логическая операция, результатом которой является истинное значение (1), если хотя бы одно из входных высказываний истинно, и ложное значение (0), если оба входных высказывания ложны.*

Логическое сложение аналогично поведению обычного "или" в повседневном языке: оно утверждает, что если хотя бы одно из высказываний истинно, то итоговое утверждение также будет истинным. Например, если первое высказывание: *Сейчас день*, а второе: *Сейчас ночь*, то логическое сложение между ними будет истинным, так как хотя бы одно из высказываний истинно.

Эту операцию также называют *дизъюнкцией* или *логическим ИЛИ*. Она используется для объединения двух высказываний, проверяя хотя бы наличие одного из них.

Для высказываний A, B может быть записана разными способами: $A \vee B$; $A + B$; $A \text{ or } B$

2.2.3 Логическое умножение

Определение 2.6. *Логическое умножение — бинарная логическая операция, результатом которой является истинное значение (1) только в том случае, если оба входных высказывания истинны, в противном случае результат будет ложным (0).*

Логическое умножение аналогично поведению обычного "и" в повседневном языке: оно утверждает, что оба высказывания должны быть истинны, чтобы итоговое утверждение также было истинным. Например, если первое высказывание: *Мы живем в 21 веке*, а второе: *Мы живем в первой половине века*, то логическое И между ними будет истинным, так как оба высказывания истинны.

Эту операцию также называют *конъюнкцией* или *логическим И*. Для высказываний A, B может быть записана разными способами: $A \wedge B$; $A \cdot B$; $A \& B$; $A \text{ and } B$

2.2.4 Логическая функция

Как и арифметические операции, логические могут быть объединены: $(A \wedge B) \vee C$

Полученное выражение — новое высказывание, которое тоже принимает значение Ис-

тина или Ложь в зависимости от значений входных переменных. Для дальнейшего удобства, будем называть это новое высказывание — логической функцией (функция в алгебре — зависимость одной переменной величины, от другой/других переменных величин).

Определение 2.7. *Логическая функция — это функция, которая принимает одно или несколько логических значений в качестве входных данных и возвращает логическое значение в качестве результата.*

Пример записи логической функции: $F(x, y, z) = x \wedge (y \vee \neg z)$. Так как при записи функции мы часто будем использовать несколько операций вместе, обозначим их приоритет:

Инверсия	$\neg A$
Конъюнкция	$A \wedge B$
Дизъюнкция	$A \vee B$

Таблица 2.1: Приоритет основных логических операций

Определение 2.8. *Тавтология — логическая функция, которая принимает истинное значение при любой комбинации значений входных аргументов.*

2.2.5 Таблица истинности

Часто перебор всех значений функции в алгебре невозможен, так как их кол-во бесконечно. Однако значения логических функций часто перебираются без труда. Например, если функция принимает 3 логических значения на вход, то значений у данной функции 8, так как каждая входная переменная может принимать всего два значения, значит различных комбинаций входных значений $2^3 = 8$. В общем случае, логическая функция принимающая n входных значений, имеет 2^n выходных значений.

Самым удобным способом представления значений функции в зависимости от значений аргументов является таблица истинности.

Определение 2.9. *Таблица истинности — таблица содержащая все возможные комбинации входных переменных и соответствующее им значения функции на выходе.*

Составим таблицы истинности для трех основных операций:

A	$\neg A$
1	0
0	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 2.1: Таблицы истинности для основных операций

В таблице 1 соответствует истине и 0 соответствует лжи. Например, в предпоследней

строке в таблице с конъюнкцией записано: при $A = 1$ и $B = 0$, значении $A \wedge B = 0$.

Таблицы истинности позволяют нам найти значения какой-либо функции и проанализировать ее:

Пусть дана функция $F(x, y, z) = \neg x \wedge y \wedge (x \vee z)$. Построим для нее таблицу истинности следующим образом: сначала разобьём изначальную функции на части и заполним таблицу истинности для этих частей. После этого заполним таблицу для функции, по имеющимся значениям частей:

x	y	z	$\neg x$	$x \vee z$	$F(x, y, z)$
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	0

2.2.6 Импликация

В алгебре мы умеем возводить число в степень и считаем это арифметической операцией, хотя на самом деле это по сути, выполнение другой арифметической операции несколько раз (произведение). В математической логике так же существуют операции, которые были введены ради удобства, но все еще могут быть выражены через основные логические операции (инверсия, конъюнкция и дизъюнкция).

Определение 2.10. *Импликация — это бинарная логическая операция, результатом которой является ложное значение (0) только в том случае, если первое высказывание истинно (1), а второе ложно (0); в остальных случаях результат истинный (1).*

Импликация часто интерпретируется как утверждение "если А, то В". Она утверждает, что если первое высказывание А истинно, то второе высказывание В также должно быть истинным, чтобы вся импликация была истинной. Например, если первое высказывание: *Если идет дождь*, а второе: *Тогда улица мокрая*, то импликация будет ложной только в том случае, если идет дождь, а улица не мокрая. В остальных случаях она будет истинной.

Эту операцию также называют *условием* или *следованием*. Она используется для установления логической зависимости между двумя высказываниями. Импликация записывается как $A \Rightarrow B$.

Логическая функция $\neg A \vee B$ — запись импликации через основные логические операции, т.е. импликацию можно заменить этим выражением и функция не поменяет свои значения. Этим можно пользоваться при упрощении логического выражения.

Составим таблицу истинности для импликации и ее записи через стандартные операции:

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

A	B	$\neg A$	$\neg A \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

2.2.7 Эквиваленция

Определение 2.11. Эквиваленция — это бинарная логическая операция, результатом которой является истинное значение (1) в случае, если оба высказывания имеют одинаковые значения, и ложное значение (0) в случае, если значения высказываний различны.

Эквиваленция часто интерпретируется как утверждение "А тогда и только тогда, когда В". Она утверждает, что оба высказывания А и В должны быть либо истинными, либо ложными для того, чтобы вся эквиваленция была истинной. Например, если первое высказывание: *Сегодня вторник*, а второе: *Сегодня рабочий день*, то эквиваленция будет истинной, если оба высказывания истинны или оба ложны.

Эту операцию также называют *равнозначностью* или *логическим равенством*. Она используется для проверки эквивалентности двух высказываний. Эквиваленция записывается как $A \Leftrightarrow B$; $A \equiv B$.

Логическая функция $(A \wedge B) \vee (\neg A \wedge \neg B)$ — запись эквиваленции через основные логические операции, т.е. эквиваленцию можно заменить этим выражением, и функция не поменяет свои значения. Этим можно пользоваться при упрощении логического выражения. Составим таблицу истинности для эквиваленции и её записи через стандартные операции:

A	B	$A \Leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

A	B	$A \wedge B$	$\neg A \wedge \neg B$	$(A \wedge B) \vee (\neg A \wedge \neg B)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

2.2.8 Исключающее ИЛИ

Определение 2.12. Исключающее ИЛИ — это бинарная логическая операция, результатом которой является истинное значение (1) в случае, если одно из входных высказываний истинно, а другое ложно. Если оба высказывания либо истинны, либо ложны, результат будет ложным (0).

Исключающее ИЛИ часто интерпретируется как утверждение "либо А, либо В, но не оба". Оно утверждает, что одно из высказываний А или В должно быть истинным, а другое — ложным для того, чтобы итоговое утверждение было истинным. Например, если первое высказывание: *Сегодня пятница*, а второе: *Сегодня рабочий день*, то исключающее ИЛИ между ними будет истинным, если одно из высказываний истинно, а другое ложно.

Эту операцию также называют *исключающей дизъюнкцией*. Она используется для проверки, что ровно одно из двух высказываний истинно. Исключающее ИЛИ записывается как $A \oplus B$ или $A XOR B$.

Логическая функция $(A \wedge \neg B) \vee (\neg A \wedge B)$ — запись исключающего ИЛИ через основные логические операции, т.е. исключающее ИЛИ можно заменить этим выражением, и функция не поменяет свои значения. Этим можно пользоваться при упрощении логического выражения. Составим таблицу истинности для исключающего ИЛИ и его записи через стандартные операции:

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

A	B	$\neg A$	$\neg B$	$(A \wedge \neg B) \vee (\neg A \wedge B)$
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

2.2.9 Преобразования и правила

TODO

2.2.10 Булева логика в Python

В Python существует тип переменной, предназначенный для работы с логическими значениями — **булевый тип**. Переменные этого типа могут два значения: True (истина) и False (ложь). Поддерживается использование 1 вместо True и 0 вместо False.

```
1 a = True
2 b = 0
3 print(a, b)
```

В таблице ниже приведены операторы для выполнения логических операций: Результат

Операция	Общая запись	Запись в Python
Отрицание	$\neg a$	<code>not a</code>
Конъюнкция	$a \wedge b$	<code>a and b</code>
Дизъюнкция	$a \vee b$	<code>a or b</code>
Импликация	$a \Rightarrow b$	<code>a <= b</code>
Эквиваленция	$a \Leftrightarrow b$	<code>a == b</code>
Исключающее ИЛИ	$a \oplus b$	<code>a != b</code>

Таблица 2.2: Логические операции в Python

логических операций можно записывать в переменные, либо использовать в условии. Пусть дана следующая функция $F(a, b) = \neg a \wedge (a \vee \neg b)$, запишем ее в Python:

```
1 a = 0
2 b = 0
3 f = not a and (a or not b)
4 print(f)
```

Значение функции будет выписано на экран. Приоритет операций сохраняется и в Python.

При более сложных выражениях имеет смысл разделять все высказывание на части и записывать результаты в разные переменные. В конце достаточно их объединить самой последней операцией. Пример: $F(x, y, z) = \neg x \Rightarrow ((\neg z \Leftrightarrow y) \vee (y \Rightarrow x))$. Разобьем на следующие части: $A : \neg z \Leftrightarrow y$; $B : y \Rightarrow x$; $C : A \vee B$; $F : \neg x \Rightarrow C$

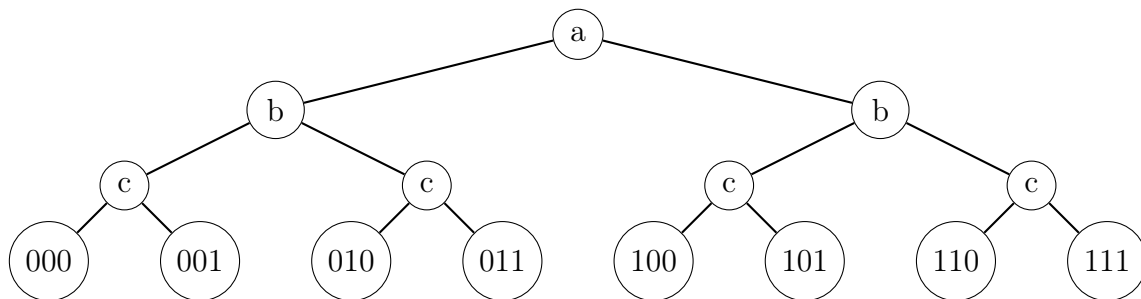
```
1 x = 1; y = 0; z = 1
2 a = not z == y
3 b = y <= x
4 c = a or b
5 f = not x <= c
6 print(f)
```

2.2.11 Перебор значений

В данной задаче будет необходимо перебирать все возможные комбинации значений входных переменных. При небольшом количестве переменных, это можно сделать с помощью вложенных циклов:

```
1 print(a, b, c)
2 for a in range(2):
3     for b in range(2):
4         for c in range(2):
5             print(a, b, c)
```

Работу данного кода можно представить в виде диаграммы:



2.3 Строки с пропущенными значениями

В данных задачах, как правило, необходимо найти верное расположение переменных в таблице истинности, чтобы известные значения соответствовали им и давали правильный результат при подстановке в функцию.

Задание 2.3.1

Логическая функция F задаётся выражением $(x \vee y) \Rightarrow (z \equiv x)$.

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

Переменная 1	Переменная 2	Переменная 3	Функция
???	???	???	F
	0	0	0
	0		0

Решение 2.3.1

Для решения задач этого типа будем использовать следующий шаблон:

```

1 def f(x, y, z): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = x or y
4     t2 = z == x
5     t3 = t1 <= t2
6     return t3 # Конечное значение возвращаем
7
8 print("x", "y", "z")
9 for x in range(2):
10     for y in range(2):
11         for z in range(2): # С помощью вложенных циклов перебираем значения
12             if f(x, y, z) == 0: # Если нужный результат, выводим значения
13                 print(x, y, z)

```

После запуска получаем:

```

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

```

Несложно заметить что только z дважды принимает значение 0, следовательно, 2 переменная - z . Когда z ноль, 3 переменная тоже обнуляется, следовательно, это y .

Ответ: xzy

На что обратить внимание:

- Чтобы не запутаться, выводим названия переменных (8-я строка) в таком же порядке, как и подходящую комбинацию (12-ая строка)
- Вложенных циклов должно быть столько, сколько и переменных в выражении. Для самопроверки можно вывести все значения без проверки, чтобы удостовериться в переборе всех возможных комбинаций
- Разделяем выражение на простейшие выражения без потери приоритетов (см. [Таблица 2.1](#))

Задание 2.3.2

Логическая функция F задаётся выражением $(x \wedge \neg y) \vee (y \equiv z) \vee \neg w$.

На рисунке приведён фрагмент таблицы истинности функции F , содержащий все наборы аргументов, при которых функция F ложна. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных w, x, y, z . Все строки в представленном фрагменте разные.

Перем. 1	Перем. 2	Перем. 3	Перем. 4
???	???	???	???
	0		
1	0		0
1		0	0

Решение 2.3.2

Используем шаблон, сделав необходимые изменения:

```

1 def f(x, y, z, w): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = x and not y
4     t2 = y == z
5     t3 = not w
6     t4 = t1 or t2 or t3
7     return t4 # Конечное значение возвращаем
8
9 print("x", "y", "z", "w")
10 for x in range(2):
11     for y in range(2):
12         for z in range(2):
13             for w in range(2): # Добавляем еще один цикл, так как добавилась переменная
14                 if f(x, y, z, w) == 0: # Из условия - функция ложна
15                     print(x, y, z, w)

```

После запуска получаем:

```

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

```

Так как в двух последних строках таблицы по два нуля, то первая строка в таблице может соответствовать только последнему выведенному ряду, следовательно, z - переменная 3. Переставим столбцы z и y и поменяем местами первую строку с последней, чтобы было проще сопоставлять:

```

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

```

Тогда, очевидно, что w - переменная 1 (первый столбец содержит две единицы), поменяем его местами с x . Поменяем местами первые две строчки и получим соответствие:

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

Обратите внимание, что выражение проверяем на ложность (14-ая строка) так как об этом сказано в условии (в прошлой задаче было указано в таблице): *"приведён фрагмент таблицы истинности функции F, содержащий все наборы аргументов, при которых функция F ложна"*

Ответ: wzyx

Задание 2.3.3

Логическая функция F задаётся как $((x \wedge y) \vee (y \wedge z)) \equiv ((x \Rightarrow w) \wedge (w \Rightarrow z))$.

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

Переменная 1	Переменная 2	Переменная 3	Переменная 4	Функция
???	???	???	???	F
0	1	1	1	1
0	1	0		1
0	1	0		1

В ответе напишите буквы x, y, z, w 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.

Решение 2.3.3

Используем шаблон, сделав необходимые изменения:

```
1 def f(x, y, z, w): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = x and y
4     t2 = y and z
5     t3 = t1 or t2
6
7     t4 = x <= w
8     t5 = w <= z
9     t6 = t4 and t5
10
11    t7 = t3 == t6
12    return t7 # Конечное значение возвращаем
13
14 print("x", "y", "z", "w")
15 for x in range(2):
16     for y in range(2):
17         for z in range(2):
18             for w in range(2): # Добавляем еще один цикл, так как добавилась переменная
19                 if f(x, y, z, w) == 1: # Из условия - функция ложна
20                     print(x, y, z, w)
```

После запуска получаем:

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

Так как строчка 0 1 1 1 - единственная, где три единицы, делаем вывод, что первая переменная правильно определена - x. Уберем все значения, где x = 1, так как таких нет в таблице:

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

Переставим местами y и w:

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

Таблица и вывод сопоставились.

Ответ: xwzy

Задание 2.3.4

Логическая функция F задаётся выражением:

$$(x \Rightarrow (y \equiv w)) \wedge (y \equiv (w \Rightarrow z)).$$

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

Переменная 1	Переменная 2	Переменная 3	Переменная 4	Функция
1		0	1	1
0	0		0	1
0	0	0	1	0

В ответе напишите буквы x, y, z, w 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.

Решение 2.3.4

В отличие от предыдущих задач данного типа, в таблице истинности значение функции меняется (последний столбец). В таких задачах, стоит разделить (условно) решение на две части: $F = 1$ и $F = 0$.

Выведем все комбинации, значение функции при которых 0, используя шаблон:

```

1 def f(x, y, z, w): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = y == w
4     t2 = x <= t1
5
6     t3 = w <= z
7     t4 = y == t3
8
9     t5 = t2 and t4
10    return int(t5) # Конечное значение возвращаем как число
11    # int(True) -> 1; int(False) -> 0; Возврат целого числа необходим для более понятной
12    #   записи
13
14    print("x", "y", "z", "w", "f") # Добавим вывод значения функции
15    for x in range(2):
16        for y in range(2):
17            for z in range(2):
18                for w in range(2): # Добавляем еще один цикл, так как добавилась переменная
19                    if f(x, y, z, w) == 0: # Из условия - функция ложна
20                        print(x, y, z, w, f(x, y, z, w))

```

Результат:

```

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

```

Строки, которые теоретически могут подходить под третью строку таблицы выделим. Следовательно, на последней позиции может быть либо x, либо z. *Предположим, что это z.*

Сделаем необходимые изменения в шаблоне и выведем комбинации при значении функции 1:

```

1 def f(x, y, z, w): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = y == w
4     t2 = x <= t1
5

```

```

6      t3 = w <= z
7      t4 = y == t3
8
9      t5 = t2 and t4
10     return int(t5) # Конечное значение возвращаем как число
11     # int(True) -> 1; int(False) -> 0; Возврат целого числа необходим для более понятной
      ↪ записи
12
13     print("x", "y", "w", "z", "f") # Добавим вывод значения функции
14     for x in range(2):
15         for y in range(2):
16             for z in range(2):
17                 for w in range(2): # Добавляем еще один цикл, так как добавилась переменная
18                     if f(x, y, z, w) == 1: # Из условия - функция ложна
19                         print(x, y, w, z, f(x, y, z, w))

```

Обратите внимание, что меняем местами переменные только в выводе на 13-ой строке и 19-ой (не меняем порядок параметров в функции ни при вызове, ни при декларации!).
Результат:

```

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

```

Заметим, что выделенная строка, подходит под 2 строку из таблицы, однако отсутствует первая строка из таблицы. Так как, z (согласно предположению) стоит на правильном месте, а w может стоять только на 3 позиции (иначе не подходит), то единственным вариантом перестановки является замена x и y (строка останется неизменной, но возможно будет найдена первая строка из таблицы):

```

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

```

Нашлась первая строка из таблицы, а значит такая последовательность нам подходит (изначальное предположение — верно).

Ответ: uhwz

В этом подтипе часто встречается необходимость в предположениях. В самом начале решения, мы увидели, что последнюю позицию должна занимать одна из двух возможных переменных. В таком случае, стоит сделать предположение и продолжить решение. Если таблица будет соответствовать данной, значит предположение — верно, в противном случае, предположение было неверно и другой вариант должен быть рассмотрен. Еще одной особенностью данного решения является поэтапный поиск строк. Сделав предположение, мы "нашли" одну из трех строк. Посмотрев на другое значение функции, мы "нашли" следующую строку. Изменили порядок и "нашли" последнюю строку. Главное, в таком процессе не "находить" новые строки, "теряя" ранее полученные.

Задание 2.3.5

Две логические функции заданы выражениями:

$$F1 = (x \Rightarrow y) \equiv (w \vee \neg z)$$

$$F2 = (x \Rightarrow y) \wedge (\neg w \equiv z)$$

Дан частично заполненный фрагмент, содержащий неповторяющиеся строки таблицы истинности обеих функций. Определите, какому столбцу таблицы истинности соответствует каждая из переменных w, x, y, z:

???	???	???	???	F_1	F_2
	1	0	1		0
	0	0	0	0	
0		0	0	0	1

Решение 2.3.5

Используем шаблон, сделав необходимые изменения:

```

1 def f1(x, y, z, w): # как параметры передаем все переменные из выражения
2     # Разделяем выражение на части и записываем в разные переменные
3     t1 = x <= y
4     t2 = w or not z
5     t3 = t1 == t2
6     return int(t3) # Конечное значение возвращаем
7
8 def f2(x, y, z, w): # как параметры передаем все переменные из выражения
9     # Разделяем выражение на части и записываем в разные переменные
10    t1 = x <= y
11    t2 = not w == z
12    t3 = t1 and t2
13    return int(t3) # Конечное значение возвращаем
14
15 print("x", "y", "z", "w", "#", "1", "2")
16 for x in range(2):
17     for y in range(2):
18         for z in range(2):
19             for w in range(2): # Добавляем еще один цикл, так как добавилась переменная
20                 print(x, y, z, w, '|', f1(x, y, z, w), f2(x, y, z, w))

```

После запуска получаем:

```

x y z w # 1 2
0 0 0 0 | 1 0
0 0 0 1 | 1 1
0 0 1 0 | 0 1
0 0 1 1 | 1 0
0 1 0 0 | 1 0
0 1 0 1 | 1 1
0 1 1 0 | 0 1
0 1 1 1 | 1 0
1 0 0 0 | 0 0

```

```

1 0 0 1 | 0 0
1 0 1 0 | 1 0
1 0 1 1 | 0 0
1 1 0 0 | 1 0
1 1 0 1 | 1 1
1 1 1 0 | 0 1
1 1 1 1 | 1 0

```

Заметим, что среди тех строчек, где F1 и F2 принимают значения 0 1, только одна подходит: 0 0 1 0 | 0 1. Следовательно, переменная 2 - z. Переставим местами два столбца в середине и уберем строки, где F1 и F2 не образуют нужных комбинаций или исходные переменные не могут быть теоретически использованы:

```

x z y w # 1 2
0 1 0 0 | 0 1
0 1 0 1 | 1 0
0 0 1 0 | 1 0
0 1 1 1 | 1 0
1 0 0 0 | 0 0
1 1 0 0 | 1 0
1 0 1 0 | 1 0
1 1 1 1 | 1 0

```

Выделенные строки соответствуют данной таблице.

Ответ: zxuw

При решении данного типа, часто не имеет смысл накладывать условия на F1 и F2. Вместо этого стоит самостоятельно вычеркивать линии, которые не подходят никаким образом (например, в данной таблице все строки содержат хотя бы 2 единицы, значит из вывода можно убрать все строки, где единиц нет совсем или только одна).

3 Задание 3. Поиск информации в реляционных базах данных

3.1 Спецификация

Задание проверяет: Умение поиска информации в реляционных базах данных.

Уровень сложности: Базовый

Максимальный балл: 1

Примерное время: 3 минуты (2,5 минуты)

3.2 Теория

Определение 3.1. База данных (БД) — это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

Как правило, БД представлена в виде таблицы, где одна строка соответствует одной записи. Каждый столбец отвечает за какое-то конкретное значение, определенное для записей. Пример базы данных, представленной в виде таблицы:

ID	Имя	Фамилия	Рост	Школа	Кол-во одноклассников
29	Степан	Сидоров	167	ГБОУ №1010	23
30	Анна	Петрова	160	ГБОУ №2345	25
31	Иван	Иванов	175	ГБОУ №678	22
32	Мария	Смирнова	162	ГБОУ №345	27
33	Алексей	Кузнецов	170	ГБОУ №9876	20

Таблица 3.1: Пример БД (таблица)

Мы будем работать с базами данных, которые представлены в виде таблицы в формате .xlsx (Excel таблица). Данный файл уже будет загружен на компьютер, для работы с ним понадобится редактор электронных таблиц. На экзамене хотя бы один будет уже предустановлен (скорее всего «МойОфис Таблица» или Microsoft Excel). Функционал большинства редакторов электронных таблиц похож, однако могут встречаться свои особенности. В данном блоке теории будет использоваться редактор Microsoft Excel.

3.2.1 Знакомство с редактором таблиц

Открыть файл можно либо двойным кликом по файлу с БД или открытием непосредственно в приложении редактора (в Excel: Открыть → Обзор → "находим файл в файловой системе").

	A	B	C	D	E	F	G
1	ID	Имя	Фамилия	Рост	Школа	Кол-во одноклассников	
2	29	Степан	Сидоров	167	ГБОУ №1010	23	
3	30	Анна	Петрова	160	ГБОУ №2345	25	
4	31	Иван	Иванов	175	ГБОУ №678	22	
5	32	Мария	Смирнова	162	ГБОУ №345	27	
6	33	Алексей	Кузнецов	170	ГБОУ №9876	20	
7	34	Наталья	Иванов	175	ГБОУ №345	22	
8	35	Дмитрий	Волков	180	ГБОУ №4321	25	
9	36	Степан	Смирнова	160	ГБОУ №345	23	
10	37	Наталья	Новикова	180	ГБОУ №2345	23	
11	38	Степан	Сидоров	175	ГБОУ №9876	25	
12	39	Мария	Новикова	175	ГБОУ №1234	23	
13	40	Дмитрий	Попова	170	ГБОУ №9876	23	
14	41	Степан	Смирнова	175	ГБОУ №9876	23	
15	42	Елена	Волков	160	ГБОУ №987	27	
16	43	Наталья	Новикова	172	ГБОУ №345	22	
17	44	Анна	Новикова	170	ГБОУ №1010	30	
18	45	Алексей	Волков	167	ГБОУ №987	25	
19	46	Сергей	Сидоров	167	ГБОУ №1010	27	
20	47	Анна	Петрова	167	ГБОУ №543	22	
21	48	Алексей	Сидоров	160	ГБОУ №1010	22	
22	49	Алексей	Петрова	167	ГБОУ №543	23	
23	50	Мария	Иванов	172	ГБОУ №9876	23	
24	51	Анна	Смирнова	167	ГБОУ №2345	25	
25	52	Ольга	Иванов	172	ГБОУ №1234	25	
26	53	Сергей	Петрова	178	ГБОУ №345	27	
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							

Ученики

Готово Специальные возможности: все в порядке

Рис. 3.1: Главный вид

Красным прямоугольником обозначено "имя" выделенной ячейки, изменять значение этой клетки можно в поле, выделенном желтым цветом. Имя ячейки используется как уникальный идентификатор и складывается из имени столбца (выделено фиолетовым) и номера строчки (выделено оранжевым). Снизу, в белой рамке выделена таблица, на который мы находимся и ее имя - в данном примере это *Ученики*.

Несколько ячеек можно выделять для копирования/вставки. Все стандартные комбинации работают и в этих редакторах: `ctrl + C`; `ctrl + V`; `ctrl + Z`; `ctrl + A`; Для

выделения столбца или строки, кликаем на соответствующее место в нумерации/наименовании строк/столбцов.

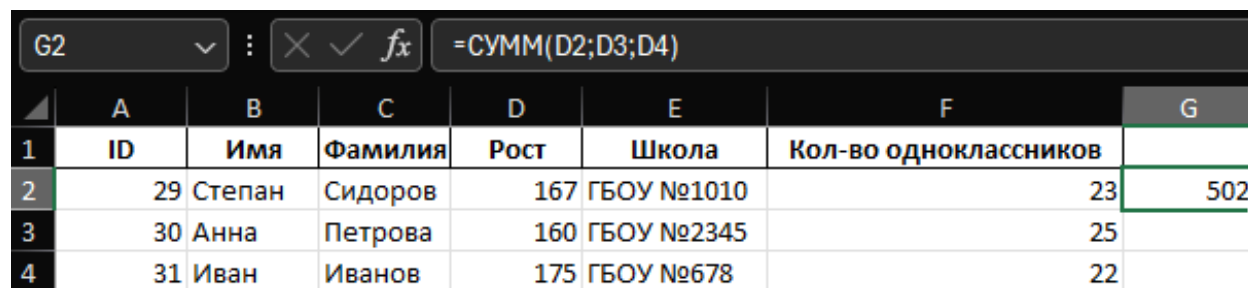
Редактирование стилизации ячеек можно осуществлять на панели инструментов (набор функциональных кнопок и меню в верхней части экрана), на вкладке *Главная*. В данном блоке теории, касательно стилизации, стоит лишь упомянуть, что размеры столбцов и строк можно изменять, двигая ползунок, появляющийся при наведении на границу двух строк/столбцов. Если какая-то информация, записанная в ячейке не видна из-за размеров ячейки, то ее стоит увеличить.

3.2.2 Формулы

Представим, что необходимо в какую-то ячейку записать средний рост всех учеников из таблицы. Как мы знаем, среднее значение вычисляется по формуле $(\sum_{i=0}^n x_i)/n$, то есть, необходимо посчитать сумму всех величин из столбца роста. Для небольшой таблицы, это сделать несложно, однако что если количество записей исчисляется тысячами? А если мы уберем пару записей, сумму нужно будет пересчитывать? Чтобы автоматизировать этот процесс, большинство редакторов электронных таблиц предлагают использовать формулы (функции).

Определение 3.2. *Формула (функция) в редакторе электронных таблиц — это выражение, которое выполняет вычисления на основе данных, введенных в ячейки.*

Формулы могут включать числа, ссылки на другие ячейки, арифметические операторы и встроенные функции. Результат вычисления формулы автоматически отображается в ячейке, где формула была введена, и обновляется при изменении данных, на которые она ссылается. В большинстве редакторов, запись функции начинается со знака равно. Попробуем записать формулу, для подсчета суммы значений в некоторых ячейках:



	A	B	C	D	E	F	G
1	ID	Имя	Фамилия	Рост	Школа	Кол-во одноклассников	
2	29	Степан	Сидоров	167	ГБОУ №1010	23	502
3	30	Анна	Петрова	160	ГБОУ №2345	25	
4	31	Иван	Иванов	175	ГБОУ №678	22	

Рис. 3.2: Запись функции суммы

Обратите внимание, что каждая функция имеет свое собственное название (всегда заглавными буквами) и синтаксис. В [Рис. 3.2](#), функция имеет название СУММ, ее параметры — имена ячеек или константы (перечисляются через ";"), итоговое значение — сумма параметров.

В данном блоке теории не будет описана каждая используемая формула, но только те, чей результат и механизм работы не является очевидным. В [Таблица 3.2](#) приведены основные формулы, которые мы будем использовать.

Название	Пример	Описание
СУММ	=СУММ(A2:A10;C2:C10)	Складывает значения ячеек A2:10, а также ячеек C2:C10 (Документация)
ПРОИЗВЕД	=ПРОИЗВЕД(A2:A4; 2)	Перемножает числа из ячеек A2–A4, а затем умножает полученный результат на 2 (Документация)
ЧАСТНОЕ	=ЧАСТНОЕ(-10;A3)	Целая часть результата деления -10 на значение ячейки A3 (Документация)
КОРЕНЬ	=КОРЕНЬ(A2)	Квадратный корень значения ячейки A2 (Документация)
ЕСЛИ	=ЕСЛИ(A1>10;"Больше";"Не превосходит")	Если значение A1 больше 10, записать в данную ячейку <i>Больше</i> , иначе записать <i>Не превосходит</i> (Документация)
СЧЁТ	=СЧЁТ(A5:A7)	Подсчитывает количество ячеек, содержащих числа, в диапазоне A5:A7 (Документация)
СЧЁТЕСЛИ	=СЧЁТЕСЛИ(B2:B5;'>55')	Количество ячеек со значением больше 55 в ячейках B2–B5 (Документация)
СУММЕСЛИ	=СУММЕСЛИ(A2:A5;'>100';B2:B5)	Ячейки из B2:B5 являются слагаемым, если соответствующая ячейка из A2:A5 имеет значение больше 100 (Документация)
И	=И(A2>1;A2<100)	ИСТИНА, если число в ячейке A2 больше 1 И меньше 100, иначе ЛОЖЬ (Документация)
ИЛИ	=И(A2>1;A2<100)	ИСТИНА, если число в ячейке A2 больше 1 ИЛИ меньше 100, иначе ЛОЖЬ (Документация)
НЕ	=НЕ(A2>100)	A2 НЕ больше 100 (Документация)

Таблица 3.2: Основные функции в Excel

Это лишь основные операции, которые будут встречаться в решениях чаще всего. Однако существует много и других функций, которые будут использоваться, с ними необходимо познакомиться во время решения. На [Рис. 3.1](#), слева от области, выделенной желтым цветом располагается кнопка *Fx* - вставка функции. При нажатии открывается меню, в которой можно найти необходимую функцию и небольшую справку к ней.

3.2.3 Реляционные базы данных

При выполнении работ в школе, мы часто подписываемся с помощью имени и фамилии. Не сложно догадаться, что использование лишь имени усложнило бы процесс идентификации ученика при проверке работы, так как нередко имена в одном классе повторяются. Поэтому, комбинация *имя + фамилия* в контексте школьных работ — хороший идентификатор, так как почти всегда является *уникальным*.

Определение 3.3. *Идентификатор (ключ) – это значение, которое однозначно определяет запись. Так же обозначается как ID (от англ. identification)*

Однако при заполнении документов в МФЦ, указывается серия и номер паспорта, потому что комбинация *имя + фамилия* может легко совпасть у двух разных людей в контексте района. В данном случае серия и номер паспорта — хороший идентификатор, так как является уникальным для каждого гражданина (уникальность обеспечивает контролируемая выдача идентификаторов, которая исключает вероятность коллизии).

Определение 3.4. *Коллизия — ситуация, при которой идентификаторы у двух записей совпадают.*

Теперь представим, что у нас есть БД (таблица), с записями о школьниках и их средних баллах за три предмета:

ID	Имя	Фамилия	Математика	Информатика
29	Степан	Сидоров	3.78	3.75
30	Анна	Петрова	4.00	3.50
31	Иван	Иванов	4.70	4.80
32	Мария	Смирнова	5.00	4.90
33	Алексей	Кузнецов	3.66	3.80

Таблица 3.3: Записи об учениках и их оценках

Наверняка, школьник захочет проверить правильность внесения данных в данную таблицу. Значит, должна быть другая БД (таблица), с оценками за каждую работу, для каждого ученика. Однако [Таблица 3.3](#) уже содержит информацию о том, какой ID присвоен какому ученику. Соответственно, в таблице по отдельному предмету нет необходимости в такой информации, ученика будем идентифицировать только по ID. Посмотрим на эту таблицу:

ID Ученика	СР №1	ДЗ №1	СР №2	КР №1
29	4	5	3	3
30	4	4	4	4
31	5	4	3	4
32	3	4	3	5
33	5	5	4	5

Таблица 3.4: Записи о работах по Информатике

Таким образом, можно сказать, что [Таблица 3.3](#) и [Таблица 3.4](#) связаны между собой (имея ID школьника из [Таблица 3.3](#) можем посмотреть его оценки из [Таблица 3.4](#), и наоборот). Тогда, можно объединить эти две БД (таблицы) в одну БД — такая база данных называется реляционной.

Определение 3.5. Реляционная база данных – это совокупность таблиц, которые связаны между собой. Связь создается с помощью идентификаторов (ключевых полей).

Слово реляция основано от англ. relation (отношение, зависимость, связь). Часто, для репрезентации реляционных БД используются диаграммы, где поля, отвечающие за одно и то же значение, в разных таблицах соединены линиями - [Рис. 3.3](#).

Рис. 3.3: Реляция [Таблица 3.3](#) и [Таблица 3.4](#)

Каждая задача данного типа подразумевает умение работать с такими таблицами. При этом количество реляций в одной БД может быть больше 1 (зачастую 2).

3.2.4 Фильтры

Представим, что в БД находится 30 записей об учениках и их итоговых оценках за три предмета. Наша задача: выбрать в олимпиадный класс по информатике 10 учеников, чьи итоговые оценки за этот предмет - 4 или 5. Мы бы могли проходиться по каждой записи и смотреть, отвечает ли она нашим критериям. Если ученик подходит, запомним его ID.

Представим, что мы решили отбирать учеников еще и по итоговой оценке с математики. Тогда, нам придется пройтись еще раз по записям и уже отобрать по двум критериям. Такой подход займет сколько-то времени, но по итогу у нас будут необходимые нам

записи.

Если количество записей будет исчисляться сотнями, то описанный ранее способ уже не подойдет: будет потрачено слишком много времени. Чтобы автоматизировать данную задачу, большинство редакторов электронных таблиц имеют функцию фильтрации.

Определение 3.6. *Фильтр (в эл. таблицах) - инструмент, позволяющий отображать только те записи, которые отвечают заданным критериям.*

В данном тексте будет приведен небольшой пример использования фильтров (рекомендуется ознакомиться и с [документацией](#)) Чтобы воспользоваться фильтром в таблице, его необходимо сначала активировать, это можно сделать либо комбинацией клавиш CTRL + SHIFT + L, либо нажатием на кнопку Фильтр на панели инструментов Данные.

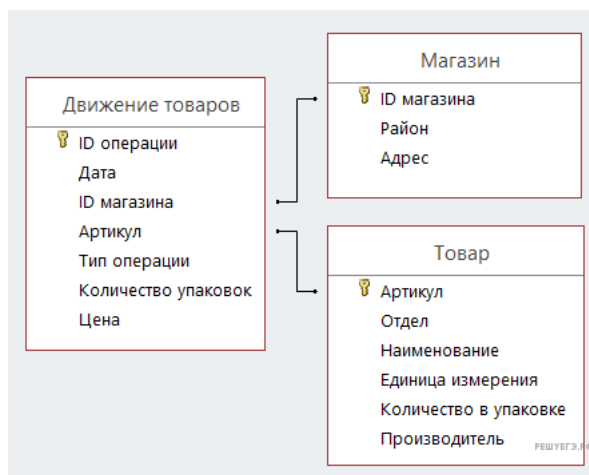
3.3 Задания для подготовки

Все файлы с которыми мы будем работать в данном разделе находятся в [./3_problems](#).

Задание 3.3.1

В [3_1.xlsx](#) приведён фрагмент базы данных «Продукты» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц.

Таблица «Движение товаров» содержит записи о поставках товаров в магазины в течение первой декады июня 2021 г., а также информацию о проданных товарах. Поле Тип операции содержит значение Поступление или Продажа, а в соответствующее поле Количество упаковок, шт. занесена информация о том, сколько упаковок товара поступило в магазин или было продано в течение дня. На рисунке приведена схема указанной базы данных:



Используя информацию из приведённой базы данных, определите, на сколько увеличилось количество упаковок яиц диетических, имеющихся в наличии в магазинах Заречного района за период с 1 по 10 июня.

В ответе запишите только число.

Решение 3.3.1

4 Задание 14. Кодирование чисел. Системы счисления

Задание проверяет: Знания касающиеся позиционных систем счисления.

Уровень сложности: Повышенный

Максимальный балл: 1

Примерное время: 3 минуты (3 минуты)

4.1 Теория

Определение 4.1. Система счисления — это знаковая система, в которой приняты определённые правила записи чисел.

Знаки, с помощью которых записываются числа, называются цифрами, а их совокупность — алфавитом системы счисления. Для удобства, будем использовать сокращение *СС* для системы счисления.

Важно понимать, что *СС* определяет лишь запись числа, а не его значение (количественный эквивалент), так Римляне, увидев четырнадцать барашков запишут: XIV, а мы запишем 14. Несмотря на то, что записи отличаются, мы говорим об одном и том же количественном эквиваленте.

Можно выделить три типа систем счисления:

- Унарная
- Непозиционная
- Позиционная

4.1.1 Унарная СС

Определение 4.2. Унарная *СС* — система счисления, в которой используется лишь одна цифра.

Количество записанных цифр и есть само значение числа. Например, если 1 — цифра в унарной *СС*, то число 1111 = 4. Такая *СС* примитивно простая, однако неудобна при выполнении арифметических операций, записи дробей и тд. С данной системой счисления мы работать не будем.

4.1.2 Непозиционная СС

Определение 4.3. Непозиционная *СС* — система счисления, в которой количественный эквивалент (количественное значение) цифры в числе **не** зависит от её положения в записи числа.

В большинстве непозиционных систем счисления числа образуются путём сложения "узловых" чисел (числа запись которых определена заранее, т.е. получена неалгоритмически).

Примером непозиционной *СС* служит римская система счисления:

Римская запись	Значение
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Таблица 4.1: Римские цифры

Числа получаются путём сложения и вычитания "узловых" чисел с учётом следующего правила: каждый меньший знак, поставленный справа от большего, прибавляется к его значению, а каждый меньший знак, поставленный слева от большего, вычитается из него. Например, $17 = XVII$, а $64 = LXIV$

Задание данного типа не подразумевает работу с такими СС.

4.1.3 Позиционная СС

Определение 4.4. *Позиционная СС — система счисления, в которой количественный эквивалент (количественное значение) цифры в числе зависит от её положения в записи числа.*

Для позиционных СС определено понятие основания СС.

Определение 4.5. *Основание позиционной системы счисления — количество уникальных цифр, составляющих её алфавит.*

Основанием СС может служить любое натуральное число $q > 1$. Алфавитом произвольной позиционной системы счисления с основанием q служат числа $1, \dots, q-1$, каждое из которых может быть записано с помощью одного уникального символа; младшей цифрой всегда является 0. Обратите внимание, что максимальная цифра в записи числа всегда меньше основания на 1.

В позиционной СС с основанием q любое число может быть представлено в виде:

$$A_q = \pm(a_{n-1} * q^{n-1} + a_{n-2} * q^{n-2} + \dots + a_0 * q^0) \quad (4.1)$$

В данной записи:

A_q - число (q говорит о записи в СС с основанием q);

q - основание СС;

n — количество целых разрядов числа;

a_i - цифры числа, стоящая на i -ом разряде;

q^i - "вес" i -го разряда;

Обратите внимание, что нумерация разряда идет справа налево, начиная с 0.

Запись можно расширить для определения и дробных чисел:

$$A_q = \pm(a_{n-1} * q^{n-1} + a_{n-2} * q^{n-2} + \dots + a_0 * q^0 + a_{-1} * q^{-1} + a_{-2} * q^{-2} + \dots + a_{-m} * q^{-m}) \quad (4.2)$$

Здесь, m - количество дробных разрядов числа.

Большинство чисел, которые нас окружают — числа записанные в позиционной СС с основанием 10.

Как пример, возьмем число 3425 и запишем его с помощью [Формула 4.1](#):

$$3425_{10} = 3 * 10^3 + 4 * 10^2 + 2 * 10^1 + 5 * 10^0 = 3000 + 400 + 20 + 5$$

Можно проделать то же самое и для дробного числа 3425,97 с помощью [Формула 4.2](#):

$$3425,97_{10} = 3 * 10^3 + 4 * 10^2 + 2 * 10^1 + 5 * 10^0 + 9 * 10^{-1} + 7 * 10^{-2} = 3000 + 400 + 20 + 5 + 0,9 + 0,07$$

Обратите внимание, что запись числа в виде A_i в данном контексте, говорит, что оно записано в системе счисления с основанием i . Так как в данных задачах используются только позиционные СС, то только о них мы и будем говорить, при этом слово позиционная будет опускаться.

Посмотрим на число не из десятичной СС, например: 11011_2 . Систему счисления с основанием 2 обычно называют *бинарной* или *двоичной* СС.

4.1.4 Системы счисления с основанием больше 10

Заметим, что максимальная цифра в СС с основанием десять - 9. Возникает логичный вопрос, какая максимальная цифра будет в СС с основанием 11? В количественном эквиваленте (т.е. по значению) это действительно 10, однако запись 10_{11} можно тогда расценить и как запись числа с двумя цифрами (1 и 0), так и как запись числа с одной цифрой (10). Чтобы решить такую ситуацию стали применять заглавные латинские буквы в записи числа. Соответствующее каждой букве значения приведены в [Таблица 4.2](#):

Запись	Эквивалент
A	10
B	11
C	12
D	13
E	14
F	15

Таблица 4.2: Запись цифр через латинские буквы

Теперь мы умеем различать оба варианта: A_{11} и 10_{11} .

Как выглядит запись максимального трехзначного числа в шестнадцатеричной СС? (FFF_{16}).

Обязательным навыком для решения данных задач является перевод из одной СС в другую СС. Разобьем эту операцию на две: $A_i \Rightarrow B_{10}$ и $B_{10} \Rightarrow C_j$. Будем проделывать эти действия используя строгие алгоритмы.

4.1.5 Перевод из n-ой системы счисления в 10-ую

По сути, этот перевод мы уже пару раз сделали используя [Формула 4.1](#). Выражая какое-то число по этой формуле, мы в процессе вычислений уже используем 10-чную СС, соответственно и результат получим в 10-ной СС, для примера, переведем число 1342_5 :

1. Пронумеруем разряды числа справа налево, начиная с 0: 1342_{3210} ;

2. Согласно нумерации выразим число через [Формула 4.1](#);

$$1342 = 1 \cdot 5^3 + 3 \cdot 5^2 + 4 \cdot 5^1 + 2 \cdot 5^0 = 1 \cdot 125 + 3 \cdot 25 + 4 \cdot 5 + 2 \cdot 1 = 125 + 75 + 20 + 2 = 222_{10}$$

Таким образом, $1342_5 = 222_{10}$. Как правило, визуально число должно уменьшиться, если перевод выполнялся из СС с основанием меньшим 10, и наоборот (так как вес каждого разряда увеличивается/уменьшается).

Перевод вещественного числа осуществляется с помощью [Формула 4.2](#):

$$123,22_4 = 1 \cdot 4^2 + 2 \cdot 4^1 + 3 \cdot 4^0 + 2 \cdot 4^{-1} + 2 \cdot 4^{-2} = 1 \cdot 16 + 2 \cdot 4 + 3 \cdot 1 + 2 \cdot 0,25 + 2 \cdot 0,0625$$

Часто будем пользоваться переводом из бинарной СС в 10-чную, при таком переводе прибегать к полной записи нет необходимости, ведь если на соответствующем месте стоит 0, значит разряд просто не учитывается, если стоит 1, то значение разряда - 2^i :

$$\begin{matrix} 101011 \\ 543210 \end{matrix} = 2^5 + 2^3 + 2^1 + 2^0 = 32 + 8 + 2 + 1 = 43_{10}$$

4.1.6 Перевод из 10-ой системы счисления в n-ую

Такой перевод осуществляется по следующему алгоритму:

1. Поделить число на основание нужной СС;
2. Если частное не 0, повторяем шаг 1 для частного;
3. Получившиеся остатки запишем в обратном порядке;

Приведем пример перевода 222_{10} в 5-ную СС:

$$\begin{array}{r|l} 222 & 5 \\ 20 & 44 \\ \hline 22 & \\ 20 & \\ \hline 2 & \end{array}$$

$$\begin{array}{r|l} 44 & 5 \\ 40 & 8 \\ \hline 4 & \end{array}$$

$$\begin{array}{r|l} 8 & 5 \\ 5 & 1 \\ \hline 3 & \end{array}$$

$$\begin{array}{r|l} 1 & 5 \\ 0 & 0 \\ \hline 1 & \end{array}$$

Таким образом, $222_{10} = 1342_5$. Заметим, что результат такой же, как и в [Подраздел 4.1.5](#). Перевод нецелого десятичного числа в другую СС осуществляется по следующему алгоритму:

1. Переведем целую часть в новую СС;

2. Нецелую часть умножаем на новое основание;
3. Целую часть результата запоминаем;
4. Повторяем шаг 2 и 3, если не достигнута точность или получился целый результат;

Для примера, переведем $222,37_{10}$ в пятеричную СС: целая часть $222_{10} = 1342_5$, нецелую $0,37$ переведем согласно алгоритму (точность: 3 знака после запятой):

$$0,37 * 5 = 1,85; 0,85 * 5 = 4,25; 0,25 * 5 = 1,25$$

Нецелая часть получается: $0,141$. Получаем: $222,37_{10} \approx 1342,141_5$. Используется знак *приблизительно равно*, так как мы остановились на вычислении определенной точности, т.е. возможно получить еще более точное значение.

4.1.7 Перевод из n-ой СС в m-ую СС

Используя техники из Подраздел 4.1.5 и Подраздел 4.1.6 попробуем перевести число 2855_9 в 16-ную СС:

1. Переведем число 2851_9 в 9-ную:

$$2855_9 = 2 * 9^3 + 8 * 9^2 + 5 * 9^1 + 5 * 9^0 = 2 * 729 + 8 * 81 + 5 * 9 + 5 = 2156_{10}$$

2. Теперь переведем получившееся число в 16-ную:

$$\begin{array}{r|l} 2156 & 16 \\ 16 & 134 \\ \hline & 55 \\ 48 & \\ \hline & 76 \\ 64 & \\ \hline & 12 \end{array}$$

$$\begin{array}{r|l} 134 & 16 \\ 128 & 8 \\ \hline & 6 \end{array}$$

$$\begin{array}{r|l} 8 & 16 \\ 0 & 0 \\ \hline & 8 \end{array}$$

Запишем остатки в обратном порядке: $86C_{16}$ (Помним, что если остаток больше 10, то просто используем соответствующую его значению цифру из Таблица 4.2). Таким образом, $2855_9 = 86C_{16}$.

4.1.8 Связь 2-ой, 4-ой, 8-ой и 16-ой систем счисления

(Мы знаем, что $2^1 = 2$; $2^2 = 4$; $2^3 = 8$; $2^4 = 16$)

При необходимости перевода из 2-ной СС в 4-ую, 8-ую или 16-ую, можно воспользоваться следующим свойством этих СС: числа, образованные каждыми m цифрами числа в бинарном представлении эквиваленты одной цифре в новой СС, m - показатель степени

с основанием 2 равный основанию СС, в которую переводим.

Для примера, переведем 101110100_2 в 8-ую ($2^3 = 8$, поэтому выделяем по 3 цифры):

$$\underbrace{101}_5 \underbrace{110}_6 \underbrace{100}_4$$

Следовательно, $101110100_2 = 564_8$. Числа образованные каждым блоком цифр переводим, как обычное число в 10-ую (Подраздел 4.1.5). Если число нельзя поделить на блоки по 2, 3 или 4 (в зависимости от основания СС, в которую переводим), то припишем необходимое количество незначащих нулей в начало число и будем переводить, главное начать деление с правого конца. Например, переведем 11100_2 в 16-чную:

1. Добавим еще три нуля в начало, чтобы число делилось на блоки по 4 цифры: 00011100_2 ;
2. Разделим на блоки по 4 и переведем:

$$\underbrace{0001}_{1} \underbrace{1100}_{12}$$

Следовательно, $11100_2 = 1C_{12}$ (не забываем, что 12 в данном контексте цифра, поэтому заменяем согласно Таблица 4.2).

Этим же свойством можно воспользоваться при обратном переводе: необходимо каждую цифру исходного числа представить как m цифр числа в нужной СС, m - показатель степени с основанием 2 равный основанию СС, в которую переводим. Для примера, переведем 3112_4 ($2^2 = 4$, следовательно, каждая цифра будет записана как две) в бинарную СС:

$$3112_4 = \underbrace{11}_3 \underbrace{01}_1 \underbrace{01}_1 \underbrace{10}_2$$

Следовательно, $3112_4 = 11010110_2$. Обратите внимание, что в отдельных блоках могут возникать незначащие нули, однако убрать их нельзя, так как в общей записи они являются значащими (кроме нулей в первом блоке).

Данное свойство работает не только при переводе из/в 2-ую, но и между, например, 16-ной и 8-ой или 8-ой и 4-ой, в таких случаях блок или цифры представляются как цифры/блоки в соответствующих СС.

4.1.9 Системы счисления в Python

По умолчанию в Python все вычисления происходят в 10-ой СС, однако существует ряд встроенных функций, которые позволяют использовать иные СС.

Например, для перевода 10-ного числа в двоичную СС, можно воспользоваться функцией `bin()`:

```
1 a = 124
2 b = bin(a)
3 print(b)
```

Результат:

```
0b11111100
```

Результат — строка с двоичным представлением числа. Первые два символа '0b' нужны лишь для обозначения записи, поэтому можем от них избавиться применив срез:

```
1 a = 124
2 b = bin(a)[2:]
3 print(b)
```

Результат:

```
11111100
```

Такие же функции предусмотрены и для 8-ной и 16-ой СС:

```
1 a = 239
2 b = oct(a)
3 c = hex(a)
4 print(b, c)
```

Результат:

```
0o357 0xef
```

Префикс у восьмеричного представления - '0o', а у шестнадцатеричного - '0x'. Их так же можно убрать используя срез.

Встроенных функций для перевода из 10-ной СС в другие СС (кроме оснований 2, 8, 16) не существует. Однако можно самостоятельно написать такую функцию:

```
1 def to_base(n, base):
2     digits = "0123456789ABCDEF"
3     r = ""
4     while n > 0:
5         r = digits[n % base] + r
6         n //= base
7     return r
```

Данная функция принимает на вход два параметра — само число и основание новой СС. Возвращает строку, с записью числа в нужной СС. Пример:

```
1 def to_base(n, base):
2     digits = "0123456789ABCDEF"
3     r = ""
4     while n > 0:
5         r = digits[n % base] + r
6         n //= base
7     return r
8
9 a = to_base(4522, 5)
10 print(a)
```

Результат:

```
121042
```


То есть, $4522_{10} = 121042_5$. Необходимо помнить, что все функции, переводящие в другие СС (встроенная и написанная самостоятельно) возвращают строки, следовательно, с ними нельзя производить арифметические операции.

Для перевода в СС с основанием 10, можем использовать встроенную функцию `int()`:

```
1 a = "124"
2 b = int(a, 5)
3 print(b)
```

Результат:

39

То есть $124_5 = 39_{10}$. Первым параметром является само число (как строка) в данной СС, а вторым параметром (как целое число) само основание исходной СС ($\in \langle 2; 36 \rangle$). Данная функция возвращает целое число, поэтому с ним можно производить арифметические операции.

4.2 Операции в разных СС с одной переменной

Задание 4.2.1

Операнды арифметического выражения записаны в системе счисления с основаниями 15 и 13:

$$4Cx4_{15} + x62A_{13}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита десятичной системы счисления. Определите наименьшее значение x , при котором значение данного арифметического выражения кратно 121. Для найденного значения x вычислите частное от деления значения арифметического выражения на 121 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение 4.2.1

Сначала решим без программирования. Чтобы не возникало путаницы, напомним себе, что запись \overline{abc} обозначает число, где на соответствующих позициях стоят цифры a , b и c .

Любое число в позиционной СС можно записать с помощью [Формула 4.1](#):

$$\begin{aligned} \text{Тогда, } \overline{4Cx4_{15}} &= 4 * 15^3 + 12 * 15^2 + x * 15 + 4 = 13500 + 2700 + 15x + 4 = 16204 + 15x \\ \overline{x62A_{13}} &= x * 13^3 + 6 * 13^2 + 2 * 13 + 10 = 2197x + 1014 + 26 + 10 = 1050 + 2197x \end{aligned}$$

Сложим две суммы и получим:

$$\overline{4Cx4_{15}} + \overline{x62A_{13}} = 16204 + 15x + 1050 + 2197x = 17254 + 2212x$$

Получившееся выражение записано в десятичной СС. Поставим ограничения на x :

- По условию x из алфавита десятичной СС, следовательно $0 < x < 10$;
- Основание СС у второго слагаемого 13, следовательно $x < 13$;
- Основание СС у первого слагаемого 15, следовательно $x < 15$;

- Во втором числе, x стоит на первом месте, а значит $x \neq 0$;

Получаем, $x \in \langle 1; 9 \rangle$. Переберем значения в порядке возрастания (так как ищем наименьший возможный x):

1. $x = 1$; $17254 + 2212 = 19466 \not\div 121$
2. $x = 2$; $17254 + 4424 = 21678 \not\div 121$
3. $x = 3$; $17254 + 6636 = 23890 \not\div 121$
4. $x = 4$; $17254 + 8848 = 26102 \not\div 121$
5. $x = 5$; $17254 + 11060 = 28314 \div 121$

Следовательно, $x = 5$; $28314 \div 121 = 234$.

Ответ: 234

Однако данный способ требует большого количества ручных вычислений, поэтому занимает много времени. Попробуем записать программу для автоматического подбора подходящего x :

```

1 for x in '123456789': # перебираем все возможные значения x
2     a = int('4C' + x + '4', 15) # составляем первое число и переводим в 10-ную СС
3     b = int(x + '62A', 13) # составляем второе число и переводим в 10-ную СС
4     t = a + b # вычисляем значение выражения
5     if t % 121 == 0: # проверяем, делится ли на 121
6         print(t // 121) # выводим результат при делении
7         break # выходим из цикла

```

Достаточно сказать о двух вещах:

1. Перебираем x строго из диапазона возможных значений ($x \in \langle 1; 9 \rangle$);
2. При первом найденном числе — выходим из цикла. Так как мы перебираем значения x в порядке возрастания, то первое подходящее — наименьшее значение x из возможных, а значит нет смысла искать следующее.

Задание 4.2.2

Операнды арифметического выражения записаны в системе счисления с основаниями 16 и 14:

$$3D4x_{16} + 4xC4_{14}$$

В записи чисел переменной x обозначены допустимые в данных системах счисления неизвестные цифры. Определите наименьшее значение x , при котором значение данного арифметического выражения кратно 154. Для найденного значения x вычислите частное от деления значения арифметического выражения на 154 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение 4.2.2

Поставим ограничения на x :

1. x присутствуют в записи числа в 16-ной СС, следовательно $0 \leq x < 16$
2. x присутствуют в записи числа в 14-ной СС, следовательно $0 \leq x < 14$

Следовательно, $x \in \langle 0; 13 \rangle$. Будем использовать шаблон:

```

1 for x in '0123456789ABCD': # перебираем все возможные значения x
2     a = int('3D4' + x, 16) # составляем первое число и переводим в 10-ную СС
3     b = int('4' + x + 'C4', 14) # составляем второе число и переводим в 10-ную СС
4     t = a + b # вычисляем значение выражения
5     if t % 154 == 0: # проверяем, делится ли на 154
6         print(t // 154) # выводим результат при делении
7         break # выходим из цикла

```

Результат:

187

Ответ: 187

При решении данного подтипа, используем следующий алгоритм действий:

1. Установить ограничения на x в зависимости от условия задачи и операндов;
2. Записать шаблон решения;
3. Записать правильный диапазон для перебора значений x . Если ищем минимальный подходящий — выставляем в порядке возрастания, в обратном случае - в порядке убывания;
4. Правильно записать составление каждого операнда и соответствующие им основания СС;
5. Правильно записать проверку на делимость и удостовериться, в выводе нужного значения и выхода из цикла;

4.3 Операции в разных СС с двумя переменными**Задание 4.3.1**

Операнды арифметического выражения записаны в системах счисления с основаниями 9 и 11:

$$88x4y_9 + 7x44y_{11}$$

В записи чисел переменными x и y обозначены допустимые в данных системах счисления неизвестные цифры. Определите значения x и y , при которых значение данного арифметического выражения будет наименьшим и кратно 61. Для найденных значений x и y вычислите частное от деления значения арифметического выражения на 61 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение 4.3.1

Данный подтип задачи похож на [Подраздел 4.2](#), однако теперь мы ищем минимальное значение выражения (не переменной), а так же количество переменных возрастает до двух.

Может показаться, что достаточно изменить шаблон, добавив вложенный цикл для подбора второй переменной, однако это не всегда верно. Так как позиция переменной в числе влияет на значение, то однозначно определить порядок перебора не всегда легко, рассмотрим следующий пример: $\overline{b3a}$ и $\overline{ab3}$. Очевидно, чем левее разряд, тем он весомее, однако переменные на самых левых позициях меняются, при этом позиции на следующих разрядах тоже не однозначные. Чтобы исключить возможность неправильного порядка перебора, будем его производить в любом порядке (главное перебрать все варианты) и каждый результат выражения записывать в список, интересующее нас значение — минимум в списке:

```

1 l = [] # создаем пустой список
2 for x in '012345678': # перебираем все возможные значения x
3     for y in '012345678': # перебираем все возможные значения y
4         a = int('88' + x + '4' + y, 9) # составляем первое число и переводим в 10-ную СС
5         b = int('7' + x + '44' + y, 11) # составляем второе число и переводим в 10-ную СС
6         t = a + b # вычисляем значение выражения
7         if t % 61 == 0: # проверяем, делится ли на 61
8             l.append(t) # добавляем результат выражения в список
9 print(min(l) // 61) # выводим частное от деления МИНимального значения на 61
10 # print(max(l) // 61) # если необходимо частное при делении МАКСимального значения на 61

```

Результат:

2715

Ответ: 2715

Обратите внимание на следующие отличия от шаблона из [Подраздел 4.2](#):

1. Перед циклом создается пустой список, туда будем добавлять значения выражений;
2. Добавляется вложенный цикл для подбора значения второй переменной (вложенный цикл обеспечивает перебор всех возможных комбинаций);
3. Если значение выражения делится на необходимое число, то оно просто добавляется в список: *не выводится и цикл не прерывается (не пишем `break`)*;
4. Находим минимум и вычисляем частное только после всех циклов (нет отступов);

При решении необходимо верно определить ограничения для каждой переменной *независимо от другой*. Когда определяем ограничения на x , представляем, что y просто какая-то цифра и не обращаем на нее внимание. То же самое повторяем для y .

Задание 4.3.2

Операнды арифметического выражения записаны в системах счисления с основаниями 8 и 11:

$$y04x5_{11} + 253xy_8$$

В записи чисел переменными x и y обозначены допустимые в данных системах счисления неизвестные цифры. Определите значения x и y , при которых значение данного арифметического выражения будет наименьшим и кратно 117. Для найденных значений x и y вычислите частное от деления значения арифметического выражения на 117 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение 4.3.2

Поставим ограничения на каждую из переменных по отдельности:

- Переменная x есть в обоих числах, минимальное основание СС — восемь, следовательно: $0 \leq x < 8$;
- Переменная y есть в обоих числах, минимальное основание СС — восемь. В первом операнде стоит на первом месте, следовательно: $1 \leq x < 8$;

Запишем программу используя шаблон:

```
1 l = [] # создаем пустой список
2 for x in '01234567': # перебираем все возможные значения x
3     for y in '1234567': # перебираем все возможные значения y
4         a = int(y + '04' + x + '5', 11) # составляем первое число и переводим в 10-ную СС
5         b = int('253' + x + y, 8) # составляем второе число и переводим в 10-ную СС
6         t = a + b # вычисляем значение выражения
7         if t % 117 == 0: # проверяем, делится ли на 117
8             l.append(t) # добавляем результат выражения в список
9 print(min(l) // 117) # выводим частное от деления минимального значения на 117
```

Результат:

224

Ответ: 224

Алгоритм решения данного подтипа следующий:

1. Найти ограничения на каждую из переменных по отдельности;
2. Записать шаблон решения;
3. В цикле для каждой переменной правильно записать ее диапазон значений;
4. Записать правильное условие (делимость на необходимое число);
5. Проверить, что добавляем *результат выражения* в список и *НЕ завершаем* цикл;
6. Проверить, ищем ли минимальное или максимальное значение и *вывести частное*;

4.4 Операции в одной СС

Задачи из данной секции обобщают идеи из [Подраздел 4.2](#) и [Подраздел 4.3](#), однако теперь вычисления происходят в одной конкретной СС, при этом количество переменных может меняться.

Не для всех задач данного раздела существует четкий шаблон. Однако самостоятельное решение в большинстве задач просто объединяет приемы из шаблонов и общих идей в программировании.

Задание 4.4.1

Операнды арифметического выражения записаны в системе счисления с основанием 15:

$$123x5_{15} + 1x233_{15}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 15-ричной системы счисления. Определите наименьшее значение x , при котором значение данного арифметического выражения кратно 14. Для найденного значения x вычислите частное от деления значения арифметического выражения на 14 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение 4.4.1

Подобные этой задаче решаются с шаблоном из [Подраздел 4.2](#):

```
1 for x in '0123456789ABCDE': # перебираем все возможные значения x
2     a = int('123' + x + '5', 15) # составляем первое число и переводим в 10-ную СС
3     b = int('1' + x + '233', 15) # составляем второе число и переводим в 10-ную СС
4     t = a + b # вычисляем значение выражения
5     if t % 14 == 0: # проверяем, делится ли на 14
6         print(t // 14) # выводим результат при делении
7         break # выходим из цикла
```

Результат:

8767

Важно обращать внимание на ограничения каждой переменной (в данном случае только одна - x):

- По условию, x из алфавита 15-ричной СС (цифры в 15-ричной СС), следовательно $x \in \langle 0; 14 \rangle$. Эквиваленты цифровым значениям найдем [Таблица 4.2](#);
- Перебираем значения x в порядке возрастания, выходим из цикла как только выражение подходит;
- Оба числа переводятся из 15-ричной СС;

Ответ: 8767

Задание 4.4.2

Числа M и N записаны в системе счисления с основанием 9 соответственно.

$$M = 842x5_9, N = 8x725_9$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита девятеричной системы счисления. Определите наименьшее значение натурального числа A , при котором существует такой x , что $M + A$ кратно N .

Решение 4.4.2

Подобрать сразу несколько значений можно с помощью вложенного цикла, однако стоит верно определить порядок перебора. Посмотрим на конструкцию:

```
for A in range(5):
    for x in range(3):
        print(A, x)
```

Полученные значения:

$(0, 0); (0, 1); (0, 2); (1, 0); (1, 1); (1, 2); \dots (4, 0); (4, 1); (4, 2)$

Можно сказать, что для каждого значения A перебираются все значения x :

$(A_0, x_0), (A_0, x_1), \dots (A_0, x_n); (A_1, x_0), (A_1, x_1), \dots (A_1, x_n); \dots; (A_m, x_0), (A_m, x_1), \dots (A_m, x_n);$

Если поменять местами циклы, получим, что для каждого значения x перебираются все значения A .

Так как в первую очередь нас интересует минимальный A , будем перебирать все значения x для каждого A в порядке возрастания. Как только какая-то пара (A, x) образует равенство $(M + A) \bmod N = 0$, завершим программу:

```
1 for A in range(1, 10000): # Перебираем возможные значения A
2     for x in '012345678': # Перебираем все возможные x
3         M = int('842' + x + '5', 9) # Составляем M
4         N = int('8' + x + '725', 9) # Составляем N
5         if (M + A) % N == 0: # Если (M + A) делится на N
6             print(A) # Выводим A
7             exit(0) # Завершаем программу
```

Следует обратить внимание на следующие пункты:

- $A \in \mathbb{N}$, следовательно, начинаем перебор с 1, не с 0 ($\mathbb{N} = \{1, 2, 3, \dots, \infty\}$). Верхняя граница не может быть больше, чем максимум из двух чисел (почему?), необходимо просто взять достаточно большое число;
- Правильно определяем ограничения на x , с учетом основания СС и позиций x . В данном случае $x \in \langle 0; 8 \rangle$;
- Используем `exit(0)` вместо `break`, так как завершаем сразу всю программу, а не только внутренний цикл (почему?);

