

**ГЛАВА 2.
МЕТОДИЧЕСКИЙ АНАЛИЗ РЕЗУЛЬТАТОВ ЕГЭ ПО ИНФОРМАТИКЕ**

РАЗДЕЛ 1. ХАРАКТЕРИСТИКА УЧАСТНИКОВ ЕГЭ ПО ИНФОРМАТИКЕ

1.1. Количество участников ЕГЭ по учебному предмету (за 3 года)

Таблица 2-1

2023 г.		2024 г.		2025 г.	
чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
1415	13,6	1432	14,7	1560	15,3

1.2. Процентное соотношение юношей и девушек, участвующих в ЕГЭ (за 3 года)

Таблица 2-2

Пол	2023 г.		2024 г.		2025 г.	
	чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
Женский	310	21,91	328	22,91	357	22,88
Мужской	1105	78,09	1104	77,09	1203	77,12

1.3. Количество участников экзамена в регионе по категориям (за 3 года)

Таблица 2-3

Категория участника	2023 г.		2024 г.		2025 г.	
	чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
Выпускник общеобразовательной организации текущего года	1366	96,54	1392	97,21	1518	97,31
Обучающийся образовательной организации среднего	12	0,85	4	0,28	10	0,64

Категория участника	2023 г.		2024 г.		2025 г.	
	чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
профессионального образования						
Выпускник прошлых лет	36	2,54	36	2,51	32	2,05
Обучающийся иностранной образовательной организации	1	0,07				

1.4. Количество участников экзамена в регионе по типа ОО

Таблица 2-4

№ п/п	Категория участника	2023 г.		2024 г.		2025 г.	
		чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
1.	Средняя общеобразовательная школа	703	49,68	724	50,56	861	55,19
2.	Средняя общеобразовательная школа с углубленным изучением отдельных предметов	84	5,94	103	7,19	92	5,9
3.	Гимназия	266	18,8	272	18,99	242	15,51
4.	Лицей	272	19,22	246	17,18	274	17,56
5.	Средняя общеобразовательная школа-интернат					2	0,13
6.	Лицей-интернат	36	2,54	40	2,79	42	2,69
7.	Общеобразовательная школа-интернат с первоначальной летней подготовкой			2	0,14	2	0,13
8.	Специальная (коррекцион-	1	0,07				

№ п/п	Категория участника	2023 г.		2024 г.		2025 г.	
		чел.	% от общего числа участников	чел.	% от общего числа участников	чел.	% от общего числа участников
	ная) общеобразовательная школа						
9.	Специальная (коррекционная) школа-интернат	2	0,14				
10	Открытая (сменная) общеобразовательная школа	1	0,07	4	0,28	3	0,19
11.	Техникум			1	0,07	1	0,06
12.	Иное	49	3,46	40	2,79	41	2,63
13.	Иное	1	0,07				

1.5. Количество участников ЕГЭ по учебному предмету по АТЕ региона

Таблица 2-5

№ п/п	Наименование АТЕ	Количество участников ЕГЭ по учебному предмету	% от общего числа участников в регионе
1	Алейский район	1	0,06
2	Алтайский район	17	1,09
3	Бийский район	12	0,77
4	Благовещенский район	16	1,03
5	Бурлинский район	6	0,38
6	Быстроистокский район	1	0,06
7	Волчихинский район	2	0,13
8	Егорьевский район	7	0,45
9	Ельцовский район	2	0,13
10	Завьяловский район	6	0,38
11	Залесовский муниципальный округ	7	0,45

№ п/п	Наименование АТЕ	Количество участников ЕГЭ по учебному предмету	% от общего числа участников в регионе
12	Змеиногорский район	9	0,58
13	Зональный район	7	0,45
14	Калманский район	11	0,71
15	Каменский район	22	1,41
16	Ключевский район	5	0,32
17	Косихинский район	1	0,06
18	Красногорский район	11	0,71
19	Краснощековский район	2	0,13
20	Кулундинский район	8	0,51
21	Курьинский район	5	0,32
22	Кытмановский район	5	0,32
23	Локтевский район	3	0,19
24	Мамонтовский район	10	0,64
25	Михайловский район	9	0,58
26	Немецкий национальный район	2	0,13
27	Павловский район	14	0,90
28	Панкрушихинский район	6	0,38
29	Первомайский район	24	1,54
30	Петропавловский район	2	0,13
31	Поспелихинский район	10	0,64
32	Ребрихинский район	8	0,51
33	Родинский район	5	0,32
34	Романовский район	6	0,38
35	Рубцовский район	2	0,13
36	ЗАТО Сибирский	5	0,32
37	Смоленский район	6	0,38

№ п/п	Наименование АТЕ	Количество участников ЕГЭ по учебному предмету	% от общего числа участников в регионе
38	Советский район	3	0,19
39	Солонешенский район	6	0,38
40	Солтонский район	3	0,19
41	Табунский район	2	0,13
42	Тальменский район	14	0,90
43	Тогульский район	1	0,06
44	Топчихинский район	6	0,38
45	Третьяковский район	2	0,13
46	Троицкий район	6	0,38
47	Угловский район	6	0,38
48	Усть-Калманский район	6	0,38
49	Усть-Пристанский район	1	0,06
50	Хабарский район	6	0,38
51	Целинный район	6	0,38
52	Шипуновский район	2	0,13
53	Шелаболихинский район	5	0,32
54	г. Алейск	14	0,90
55	г. Барнаул	721	46,22
56	г. Белокуриха	8	0,51
57	г. Бийск	146	9,36
58	г. Заринск	24	1,54
59	г. Новоалтайск	43	2,76
60	г. Рубцовск	92	5,90
61	г. Славгород	22	1,41
62	г. Яровое	17	1,09
63	Краевые образовательные организации	70	4,49

№ п/п	Наименование АТЕ	Количество участников ЕГЭ по учебному предмету	% от общего числа участников в регионе
64	Краевые коррекционные образовательные организации	2	0,13
65	Негосударственные образовательные организации	9	0,58

1.6. Прочие характеристики участников экзаменационной кампании (при наличии)

не выделено.

1.7. ВЫВОДЫ о характере изменения количества участников ЕГЭ по учебному предмету

На основе приведенных в разделе данных отмечается стабильный рост как абсолютного числа участников ЕГЭ по информатике в 2025 году, относительно прошлых лет, так и доли участников ЕГЭ по информатике относительно общего числа выпускников образовательных организаций. Доля участников ЕГЭ по информатике в 2025 году на 0,6% выше, чем в 2024 году и на 1,7% выше, чем в 2023 году. Это можно объяснить рядом причин:

четкой и понятной системой проведения компьютерного ЕГЭ по информатике, удачно апробированной в 2021 году с предоставлением потенциальным участникам тестирования возможности тренировать экзамен в эмуляторе станции КЕГЭ;

популярностью сферы IT для выбора профессий, а также государственной поддержкой IT-отрасли; трендом на развитие цифрового сектора экономики в стране;

активным развитием в регионе направлений работы со школьниками в системе дополнительного образования в области IT, ИИ, робототехники, управления беспилотными летательными аппаратами и пр. (в точках роста, на кружках, в технопарках – IT-куб, Таланты22, Педагогический технопарк АлтГПУ «Кванториум», центр «Наследники Ползунова» АлтГТУ при поддержке Благотворительного Фонда Андрея Мельниченко).

Доли экзаменуемых женского и мужского пола остались теми же, что и в 2024 году.

Как и в предыдущие годы, основную часть участников ЕГЭ по информатике составили выпускники текущего года, обучавшиеся по программам среднего общего образования (97,31%). Преимущественно это выпускники СОШ, СОШ с углубленным изучением предмета, лицеев и гимназий, в общем их доля составляет 94,16%.

Сохраняется доминирование числа городских обучающихся над сельскими, что объясняется рядом причин – выбор информатики и не обязателен для сдачи ЕГЭ, в вузах введена практика выбора альтернативных вступительных

экзаменов, на селе не хватает специалистов данного профиля, профильное обучение в сельских школах с малым количеством обучающихся вести невозможно, а ЕГЭ по информатике требует углубленной подготовки по предмету, предусмотренной для программ обучения по соответствующему профилю. При этом, процент городских участников экзамена не существенно, но снижается по сравнению с 2023-2024 г.г. в пользу сельских учеников в пределах 1%. Всего городских участников экзамена по информатике 69,69%. Это стало возможно по причине того, что доступ к тренировочным материалам, обучающим курсам по информатике стал проще и понятнее для сельских школьников.

Как и в 2024 среди участников ЕГЭ по информатике 2025 года только на г. Барнаул и г. Бийск приходится 55,58%, т.е. более половины всех участников. Таким образом, можно констатировать сохранение проблем, связанных с отсутствием предметников и низким качеством подготовки по информатике на селе, и, наличием возможностей для профильной подготовки по информатике в городских ОО, в большей степени на это влияет наличие подготовленных кадров в городах и открытие IT-кубов, технопарков, кванториумов и пр.

Стоит отметить, что рост интереса среди школьников Алтайского края к выбору IT профиля, а следовательно и ЕГЭ по информатике, продиктован еще и тем, что по средней заработной плате регион входит в десятку отстающих (около 54,3 тыс. руб. за 2024 г.), а средняя заработная плата IT-специалистов в РФ достигает около 88,6 тыс. руб., эта статистика показывает предложения работодателей, а не реальные зарплаты, которые получают работники с учётом премий, надбавок и переработок, кроме того, в IT отрасли есть много возможностей работать удаленно.

РАЗДЕЛ 2. ОСНОВНЫЕ РЕЗУЛЬТАТЫ ЕГЭ ПО ПРЕДМЕТУ

2.1. Диаграмма распределения тестовых баллов участников ЕГЭ по предмету в 2025 г. (количество участников, получивших тот или иной тестовый балл)

1) Распределение тестовых баллов

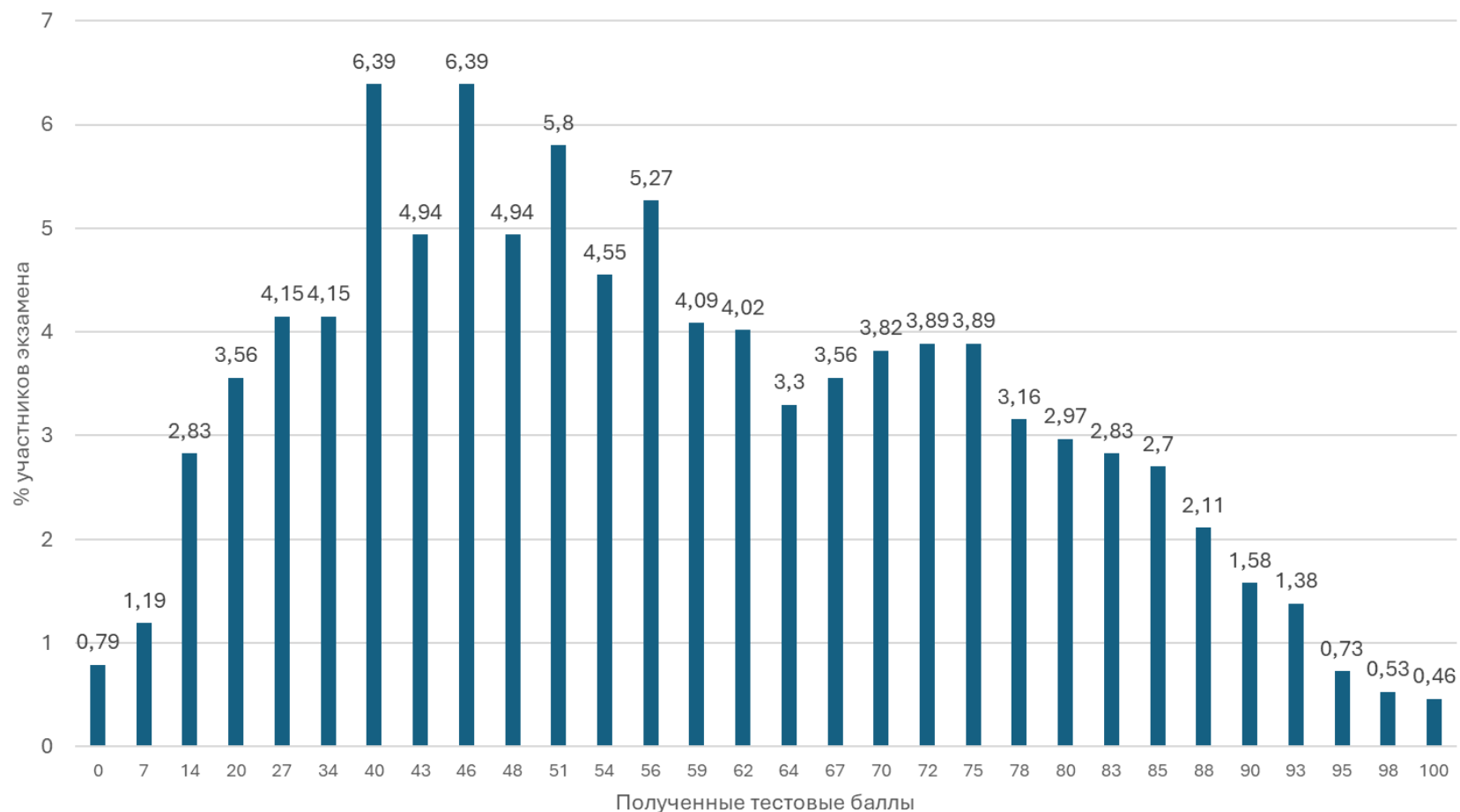


Рисунок 2-1. Распределение тестовых баллов

2) Сгруппированное распределение тестовых баллов

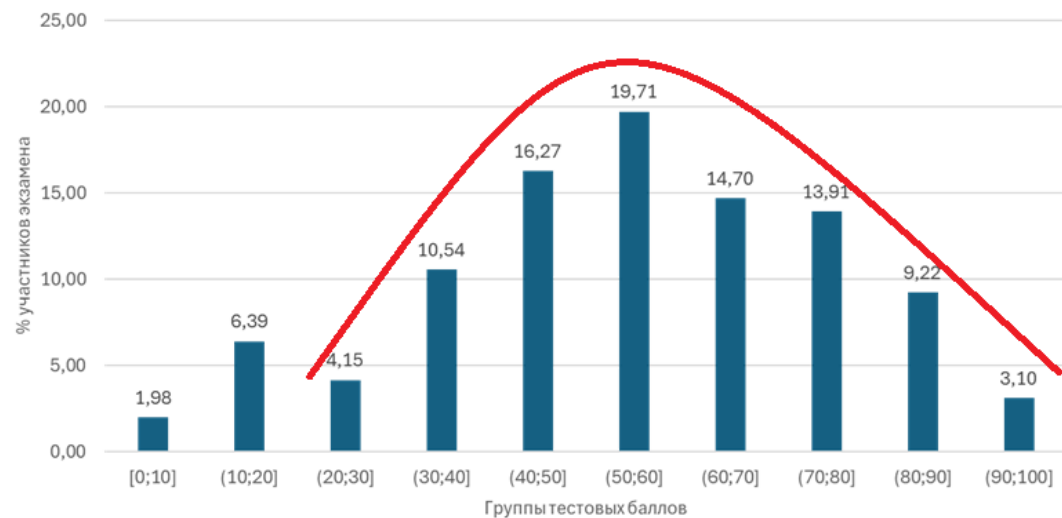


Рисунок 2-2. Сгруппированное распределение тестовых баллов

2.2. Динамика результатов ЕГЭ по предмету за последние 3 года

Таблица 2-6

№ п/п	Участников, набравших балл	Год проведения ГИА		
		2023 г.	2024 г.	2025 г.
1.	ниже минимального балла, %	13,50	16,83	16,67
2.	от минимального балла до 60 баллов, %	43,11	43,09	42,37
3.	от 61 до 80 баллов, %	32,30	30,31	28,61
4.	от 81 до 100 баллов, %	11,10	9,78	12,32
5.	Средний тестовый балл	56,86	54,21	55,37

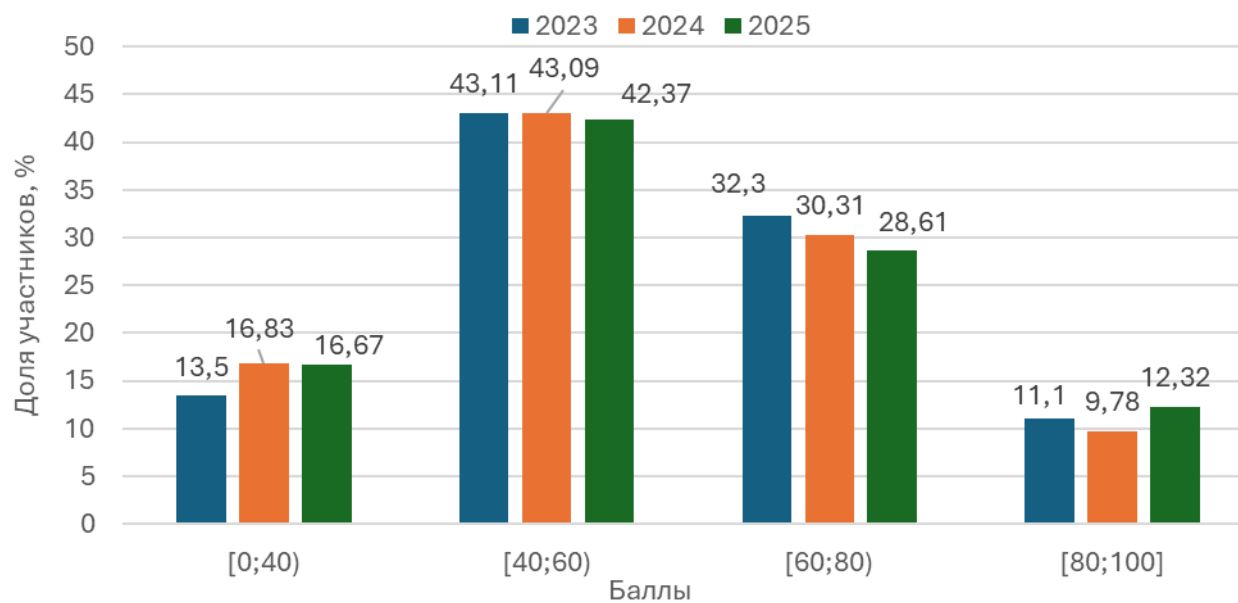


Рисунок 2-3. Сгруппированное распределение тестовых баллов в динамике за 3 года

2.3. Результаты ЕГЭ по учебному предмету по группам участников экзамена с различным уровнем подготовки

2.3.1. в разрезе категорий участников ЕГЭ

Таблица 2-7

№ п/п	Категории участников	Доля участников, у которых полученный тестовый балл			
		ниже минимального	от минимального балла до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов
1.	ВТГ, обучающиеся по программам СОО	1517	16,68	42,39	28,61
2.	ВТГ, обучающиеся по программам СПО	10	60	20	20
3.	ВПЛ	32	31,25	37,5	15,63
4.	Участники экзамена с ОВЗ	17	35,29	35,29	11,76

2.3.2. в разрезе типа ОО

Таблица 2-8

№ п/п	Тип ОО	Кол-во участников, чел.	Доля участников, получивших тестовый балл			
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов
1.	СОШ	859	20,61	44,7	26,43	8,27
2.	СОШ с УИОП	92	17,39	40,22	28,26	14,13
3.	Гимназии, лицеи	516	11,43	39,73	31,2	17,64
4.	Интернаты	42	0	28,57	42,86	28,57
5.	Вечерние и открытые (сменные) ОШ	3	0	66,67	33,33	0
6.	Другие	5	20	60	20	0

2.3.3. юношей и девушек

Таблица 2-9

№ п/п	Пол	Количество участников, чел.	Доля участников, получивших тестовый балл			
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов
1.	женский	357	14,29	42,02	30,53	13,17
2.	мужской	1202	18,14	42,18	27,62	12,06

2.3.4. в сравнении по АТЕ

Таблица 2-10

№ п/п	Наименование АТЕ	Количество участников, чел.	Доля участников, получивших тестовый балл				Количество участников, получивших 100 баллов
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов	
1	Алейский район	1	0,00	100,00	0,00	0,00	0
2	Алтайский район	17	17,65	58,82	5,88	17,65	1

№ п/п	Наименование АТЕ	Количество участников, чел.	Доля участников, получивших тестовый балл				Количество участников, получивших 100 баллов
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов	
3	Бийский район	12	33,33	50,00	16,67	0,00	0
4	Благовещенский район	16	25,00	25,00	50,00	0,00	0
5	Бурлинский район	6	16,67	50,00	33,33	0,00	0
6	Быстроистокский район	1	0,00	100,00	0,00	0,00	0
7	Волчихинский район	2	0,00	100,00	0,00	0,00	0
8	Егорьевский район	7	14,29	57,14	14,29	14,29	0
9	Ельцовский район	2	50,00	0,00	50,00	0,00	0
10	Завьяловский район	6	16,67	50,00	16,67	16,67	0
11	Залесовский муниципальный округ	7	0,00	57,14	28,57	14,29	1
12	Змеиногорский район	9	22,22	55,56	11,11	11,11	0
13	Зональный район	7	14,29	57,14	28,57	0,00	0
14	Калманский район	11	45,45	36,36	18,18	0,00	0
15	Каменский район	22	18,18	50,00	22,73	9,09	0
16	Ключевский район	5	20,00	40,00	40,00	0,00	0
17	Косихинский район	1	100,00	0,00	0,00	0,00	0
18	Красногорский район	11	9,09	36,36	45,45	9,09	0
19	Краснощековский район	2	0,00	0,00	100,00	0,00	0
20	Кулундинский район	8	25,00	50,00	25,00	0,00	0
21	Курьинский район	5	60,00	40,00	0,00	0,00	0
22	Кытмановский район	5	40,00	60,00	0,00	0,00	0
23	Локтевский район	3	33,33	0,00	0,00	66,67	0
24	Мамонтовский район	10	0,00	40,00	50,00	10,00	0

№ п/п	Наименование АТЕ	Количество участников, чел.	Доля участников, получивших тестовый балл				Количество участников, получивших 100 баллов
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов	
25	Михайловский район	9	0,00	66,67	22,22	11,11	0
26	Немецкий национальный район	2	0,00	50,00	50,00	0,00	0
27	Павловский район	14	28,57	50,00	21,43	0,00	0
28	Панкрушихинский район	6	33,33	16,67	33,33	16,67	0
29	Первомайский район	24	29,17	45,83	16,67	8,33	0
30	Петропавловский район	2	0,00	100,00	0,00	0,00	0
31	Поспелихинский район	10	20,00	30,00	50,00	0,00	0
32	Ребрихинский район	8	50,00	37,50	0,00	12,50	0
33	Родинский район	5	0,00	40,00	60,00	0,00	0
34	Романовский район	6	66,67	33,33	0,00	0,00	0
35	Рубцовский район	2	0,00	100,00	0,00	0,00	0
36	ЗАТО Сибирский	5	40,00	20,00	0,00	40,00	0
37	Смоленский район	6	66,67	16,67	16,67	0,00	0
38	Советский район	3	0,00	33,33	66,67	0,00	0
39	Солонешенский район	6	0,00	66,67	33,33	0,00	0
40	Солтонский район	3	0,00	33,33	66,67	0,00	0
41	Табунский район	2	0,00	0,00	100,00	0,00	0
42	Тальменский район	14	21,43	42,86	28,57	7,14	0
43	Тогульский район	1	0,00	100,00	0,00	0,00	0
44	Топчихинский район	6	16,67	83,33	0,00	0,00	0
45	Третьяковский район	2	50,00	0,00	50,00	0,00	0
46	Троицкий район	6	16,67	33,33	50,00	0,00	0

№ п/п	Наименование АТЕ	Количество участников, чел.	Доля участников, получивших тестовый балл				Количество участников, получивших 100 баллов
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов	
47	Угловский район	6	16,67	33,33	50,00	0,00	0
48	Усть-Калманский район	6	16,67	66,67	16,67	0,00	0
49	Усть-Пристанский район	1	100,00	0,00	0,00	0,00	0
50	Хабарский район	6	0,00	33,33	66,67	0,00	0
51	Целинный район	6	16,67	50,00	16,67	16,67	0
52	Шишунувский район	2	50,00	50,00	0,00	0,00	0
53	Шелаболихинский район	5	40,00	60,00	0,00	0,00	0
54	г. Алейск	14	28,57	35,71	14,29	21,43	0
55	г. Барнаул	721	11,93	40,64	31,35	16,09	5
56	г. Белокуриха	8	50,00	37,50	0,00	12,50	0
57	г. Бийск	146	26,71	42,47	21,23	9,59	0
58	г. Заринск	24	16,67	58,33	16,67	8,33	0
59	г. Новоалтайск	43	16,28	41,86	37,21	4,65	0
60	г. Рубцовск	91	17,58	51,65	24,18	6,59	0
61	г. Славгород	22	18,18	54,55	18,18	9,09	0
62	г. Яровое	17	17,65	35,29	41,18	5,88	0
63	Краевые образовательные организации	70	2,86	30,00	42,86	24,29	0
64	Краевые коррекционные образовательные организации	2	50,00	50,00	0,00	0,00	0

№ п/п	Наименование АТЕ	Количество участников, чел.	Доля участников, получивших тестовый балл				Количество участников, получивших 100 баллов
			ниже минимального	от минимального до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов	
65	Негосударственные образовательные организации	9	33,33	33,33	22,22	11,11	0

2.4. Выделение перечня ОО, продемонстрировавших наиболее высокие и низкие результаты ЕГЭ по предмету

2.4.1. Перечень ОО, продемонстрировавших наиболее высокие результаты ЕГЭ по предмету

Таблица 2-11

№ п/п	Наименование ОО	Количество ВТГ, чел.	Доля ВТГ, получивших тестовый балл			
			от 81 до 100 баллов	от 61 до 80 баллов	от минимального балла до 60 баллов	ниже минимального
1	МБОУ "Гимназия № 42" (г. Барнаул)	36	58,33	36,11	5,56	0
2	МБОУ "Лицей №124" (г. Барнаул)	46	52,17	39,13	8,7	0
3	МБОУ "СОШ №128" (г. Барнаул)	16	37,5	37,5	25	0
4	МАОУ "СОШ №132" им. Н.М. Малахова (г. Барнаул)	27	29,63	44,44	25,93	0
5	КГБОУ "Бийский лицей-интернат Алтайского края" (БЛИАК) (Краевые образовательные организации)	42	28,57	42,86	28,57	0
6	МБОУ "Гимназия № 27" (г. Барнаул)	17	23,53	47,06	29,41	0
7	МБОУ "Гимназия №123" (г. Барнаул)	14	21,43	42,86	35,71	0

№ п/п	Наименование ОО	Количество ВТГ, чел.	Доля ВТГ, получивших тестовый балл			
			от 81 до 100 баллов	от 61 до 80 баллов	от минимального балла до 60 баллов	ниже минимального
8	МБОУ "СОШ №113 имени Сергея Семенова" (г. Барнаул)	10	20	40	40	0
9	МБОУ "Лицей № 121" (г. Барнаул)	11	18,18	45,45	36,36	0

2.4.2. Перечень ОО, продемонстрировавших низкие результаты ЕГЭ по предмету

Таблица 2-12

№ п/п	Наименование ОО	Количество ВТГ, чел.	Доля ВТГ, получивших тестовый балл			
			ниже минимального	от минимального балла до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов
1	МБОУ "Гимназия № 1" (г. Бийск)	10	60	30	10	0
2	МБОУ "СОШ №117" (г. Барнаул)	10	40	40	10	10
3	МБОУ "СОШ №102" (г. Барнаул)	13	30,77	46,15	23,08	0
4	МБОУ "СОШ №89" (г. Барнаул)	13	30,77	38,46	30,77	0
5	МБОУ "Гимназия № 11" (г. Бийск)	17	29,41	58,82	5,88	5,88
6	МБОУ СОШ №19 (г. Яровое)	11	27,27	36,36	36,36	0
7	МБОУ СОШ №12 (г. Бийск)	16	25	25	25	25
8	МБОУ "Лицей Эрудит" (г. Рубцовск)	12	25	33,33	25	16,67
9	МБОУ Алтайская СОШ №5 (Алтайский район)	10	20	70	0	10

№ п/п	Наименование ОО	Количество ВТГ, чел.	Доля ВТГ, получивших тестовый балл			
			ниже минимального	от минимального балла до 60 баллов	от 61 до 80 баллов	от 81 до 100 баллов
10	МБОУ "Гимназия №45" (г. Барнаул)	10	20	70	10	0
11	МАОУ "СОШ №138" (г. Барнаул)	11	18,18	63,64	9,09	9,09
12	МБОУ "СОШ №107" (г. Барнаул)	12	16,67	66,67	16,67	0
13	МБОУ "Гимназия №5" (г. Барнаул)	19	15,79	42,11	10,53	31,58
14	МБОУ "Лицей №112" (г. Барнаул)	35	14,29	22,86	40	22,86
15	МБОУ "СОШ №118" (г. Барнаул)	16	12,5	56,25	18,75	12,5
16	МБОУ "СОШ № 1" (г. Бийск)	18	11,11	38,89	33,33	16,67
17	МБОУ "Лицей №129" (г. Барнаул)	38	10,53	26,32	47,37	15,79

2.5. ВЫВОДЫ о характере изменения результатов ЕГЭ по предмету

Распределение тестовых баллов участников ЕГЭ по информатике в текущем году более «гладко» аппроксимируется к нормальной кривой в отличие от распределений баллов в 2023 и 2024 годах, ранее оно имело выраженную правостороннюю асимметрию (сдвиг).

Распределение баллов таково, что привело к положению среднего значения между результатами 2023 и 2024 годов. Это значение в 2025 году ниже на 1,49 балла по сравнению с 2023 годом и выше на 1,16 балла по сравнению с 2024 годом, что является положительной характеристикой.

Доля участников, не преодолевших минимальный барьер осталась на уровне 2024 года, но это значение выше на 3,17% в сравнении с 2023 годом. Доля участников, набравших от 40 до 60 баллов, так же осталась на прежнем уровне. При этом доля участников, набравших от 60 до 80 баллов, имеет тенденцию к уменьшению, за два года на 3,69%, но это произошло за счет того, что доля высокобалльников увеличилась за два года на 1,22%, а за последний год на 2,54%. В целом – это положительные характеристики.

Участников ЕГЭ по информатике, набравших 100 баллов в текущем году 7, в 2024 году это был только 1 чело-

век, в 2023 году в Алтайском крае было 2 стобалльника.

Лучшие результаты, как и в 2023-2024 годах, демонстрируют выпускники текущего года, обучающиеся по программам СОО. По категориям ОО это СОО с углубленным изучением информатики, гимназии, лицеи. Выпускники СПО демонстрируют преимущественно низкую базовую подготовку. Выпускники прошлых лет и категория ВПЛ в основной массе либо не преодолели минимальный барьер, либо имеют низкую базовую подготовку.

Если по количеству девушек, сдававших ЕГЭ по информатике меньше, чем юношей, то доля высокобалльных результатов и результатов от 60 до 80 баллов, среди девушек выше суммарно на 4,02%, нежели среди участников – юношей. Что касается доли участников, не преодолевших минимальный барьер, то среди них больше юношей, чем девушек на 3,85%. Это говорит о том, что на экзамен идут только целенаправленно подготовленные девушки, а среди участников юношей большая доля неподготовленных, рассчитывающих либо на базовую подготовку по предмету, либо уверенные в «твердых» пятерках по предмету в школьном журнале.

В перечень ОО, продемонстрировавших наиболее высокие результаты ЕГЭ по предмету три года подряд включена организации МБОУ "Лицей №124" (г. Барнаул) и два года подряд МАОУ "СОШ №132" им. Н.М. Малахова (г. Барнаул) и МБОУ "Гимназия №123" (г. Барнаул). Вновь «вернулась» в этот список МБОУ "Гимназия № 42" (г. Барнаул). Вызывает настороженность отсутствие в этом списке второй год подряд КГБОУ "АКПЛ", потому как эта организация стабильно лидирует по количеству побед школьников на олимпиадах по программированию, наиболее массово участвуют в ЕГЭ по предмету, до 2024 года ежегодно демонстрировала большую долю участников экзамена с высокими результатами и считается организацией с высоким уровнем физико-математической подготовки.

Стабильно низкие результаты демонстрирует МБОУ "СОШ №89" (г. Барнаул), это должно быть учтено и исправлено администрацией и учителями информатики СОШ.

В целом, статистические результаты ЕГЭ по информатике удовлетворительны, однако имеется отрицательная динамика как в общем по ОО, так и по отдельным ОО.

В целом, результаты имеют положительную динамику. Возможно, на этом сказалось то, что задания КИМ практически полностью совпадали с примерами заданий демоверсии КИМ, не было значительных изменений ситуаций в формулировках. Новой в КИМ ЕГЭ была формулировка задания 27, ранее такого прототипа задания не было на экзамене. Однако это задание требует больших усилий при моделировании, отборе значений для обработки, но в разработке программы тривиально, по сравнению с заданиями №27 прошлых лет.

Раздел 3. АНАЛИЗ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ ЗАДАНИЙ КИМ

3.1. Анализ выполнения заданий КИМ

3.1.1. Статистический анализ выполнения заданий КИМ в 2025 году

3.1.1.1. Основные статистические характеристики выполнения заданий КИМ в 2025 году

Основные статистические характеристики выполнения заданий в целом представлены в Таблице 2-13. Информация о результатах оценивания выполнения заданий, в том числе в разрезе данных о получении того или иного балла по критерию оценивания выполнения каждого задания КИМ представлена в Таблице 2-14.

Таблица 2-13

Но- мер зада- ния в КИМ	Проверяемые элементы содержания / умения	Уровень сложно- сти зада- ния	Макс. балл за за- дание	Процент выполнения задания в Алтайском крае в группах участ- ников экзамена с разными уровнями подготовки				
				средний, %	в группе не преодо- левших минималь- ный балл, %	в группе от минималь- ного до 60 т.б.	в группе от 61 до 80 т.б.	в группе от 81 до 100 т.б.
1	Умение представлять и считывать данные в разных типах информационных моделей (схемы, карты, таблицы, графики и формулы)	базовый	1	90,38	63,24	94,09	96,54	100
2	Умение строить таблицы истинности и логические схемы	базовый	1	83,19	38,34	87,71	95,85	98,93
3	Умение поиска информации в реляционных базах данных	базовый	1	76,53	44,27	76,21	87,56	95,72
4	Умение кодировать и декодировать информацию	базовый	1	77,13	38,74	77,29	91,24	95,72
5	Формальное исполнение просто-	базовый	1	46,34	4,74	25,82	78,34	98,93

	го алгоритма, записанного на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные данные линейного алгоритма по результатам его работы							
6	Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов	базовый	1	39,82	2,37	28,46	59,91	82,89
7	Умение определять объём памяти, необходимый для хранения графической и звуковой информации	базовый	1	65,06	14,23	59,72	88,25	98,4
8	Кодирование данных, комбинаторика, системы счисления.	базовый	1	44,23	0,79	26,28	73,96	95,72
9	Умение обрабатывать числовую информацию в электронных таблицах	базовый	1	30,32	3,56	10,89	51,61	83,96
10	Информационный поиск средствами текстового процессора	базовый	1	82	62,45	81,03	90,09	93,05
11	Умение подсчитывать информационный объём сообщения	повышенный	1	30,26	3,16	15,86	45,16	81,82
12	Умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд	повышенный	1	45,75	1,58	27,22	77,65	95,19
13	Умение использовать маску под-	повы-	1	40,41	1,19	21,31	68,43	94,12

	сети	шенный						
14	Знание позиционных систем счисления	повышенный	1	31,31	0,4	11,82	54,15	87,17
15	Знание основных понятий и законов математической логики	повышенный	1	45,55	0,79	25,04	79,26	98,4
16	Вычисление рекуррентных выражений	повышенный	1	52,34	9,49	42,15	73,5	96,26
17	Умение составить алгоритм обработки числовой последовательности и записать его в виде простой программы (10–15 строк) на языке программирования	повышенный	1	24,92	0,4	2,95	42,4	93,05
18	Умение использовать электронные таблицы для обработки целочисленных данных	повышенный	1	41,73	4,35	26,91	65,21	88,77
19	Умение анализировать алгоритм логической игры	базовый	1	69,15	24,11	64,23	90,55	97,33
20	Умение найти выигрышную стратегию игры	повышенный	1	53,07	0,4	38,57	85,71	98,4
21	Умение построить дерево игры по заданному алгоритму и найти выигрышную стратегию	высокий	1	49,24	1,98	32,5	81,57	95,72
22	Построение математических моделей для решения практических задач. Архитектура современных компьютеров. Многопроцессорные системы.	повышенный	1	29,33	4,74	18,51	40,78	73,26
23	Умение анализировать ход исполнения алгоритма	повышенный	1	48,98	3,16	31,26	81,34	96,79

24	Умение создавать собственные программы (10–20 строк) для обработки символьной информации	высокий	1	5,67	0	0,47	4,15	34,76
25	Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации	высокий	1	9,43	0	0	8,29	57,22
26	Умение обрабатывать целочисленную информацию с использованием сортировки	высокий	2	3,99	0	0,08	1,73	28,07
27	Умение выполнять последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и построение модели, преобразование данных, визуализация данных, интерпретация результатов	высокий	2	11,73	0,4	0,47	11,41	66,58

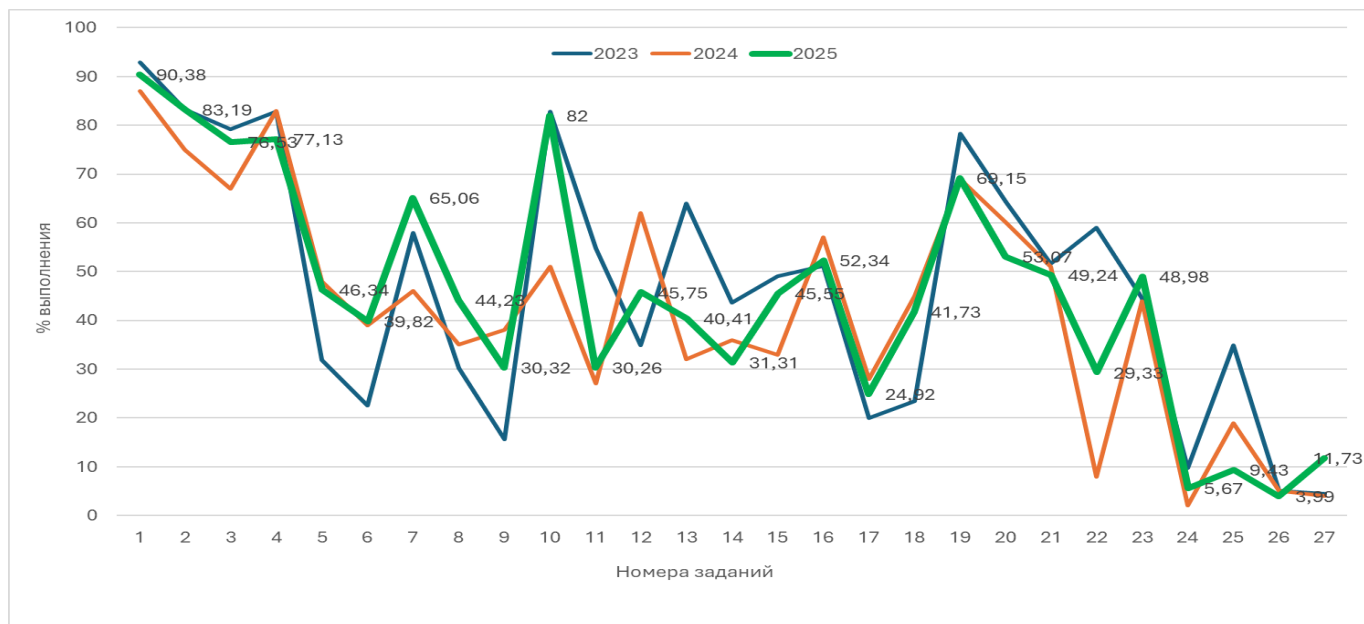


Рисунок 2-4. Динамика результатов выполнения заданий участниками ЕГЭ по информатике в 2023-2025 г.г.

Таблица 2-14

Номер задания / критерия оценивания в КИМ	Количество полученных первичных баллов	Процент участников экзамена в Алтайском крае, получивших соответствующий первичный балл за выполнения задания в группах участников экзамена с разными уровнями подготовки			
		в группе не преодолевших минимальный балл, %	в группе от минимального до 60 т.б., %	в группе от 61 до 80 т.б., %	в группе от 81 до 100 т.б., %
Количество участников в группе		253	643	434	187
1	X	2,37	0,62	0,00	0,00
	0	34,39	5,29	3,46	0,00
	1	63,24	94,09	96,54	100,00
2	X	3,95	0,78	0,00	0,00
	0	57,71	11,51	4,15	1,07
	1	38,34	87,71	95,85	98,93

Номер задания / критерия оценивания в КИМ	Количество полученных первичных баллов	Процент участников экзамена в Алтайском крае, получивших соответствующий первичный балл за выполнения задания в группах участников экзамена с разными уровнями подготовки			
		в группе не преодолевших минимальный балл, %	в группе от минимального до 60 т.б., %	в группе от 61 до 80 т.б., %	в группе от 81 до 100 т.б., %
3	X	5,14	1,09	0,00	0,00
	0	50,59	22,71	12,44	4,28
	1	44,27	76,21	87,56	95,72
4	X	1,58	0,00	0,23	0,00
	0	59,68	22,71	8,53	4,28
	1	38,74	77,29	91,24	95,72
5	X	35,57	25,51	2,76	0,00
	0	59,68	48,68	18,89	1,07
	1	4,74	25,82	78,34	98,93
6	X	17,79	6,07	0,46	0,00
	0	79,84	65,47	39,63	17,11
	1	2,37	28,46	59,91	82,89
7	X	14,62	4,51	0,92	0,00
	0	71,15	35,77	10,83	1,60
	1	14,23	59,72	88,25	98,40
8	X	28,06	20,53	2,07	0,00
	0	71,15	53,19	23,96	4,28
	1	0,79	26,28	73,96	95,72
9	X	45,85	46,50	10,37	0,53
	0	50,59	42,61	38,02	15,51
	1	3,56	10,89	51,61	83,96
10	X	2,37	0,00	0,00	0,00
	0	35,18	18,97	9,91	6,95
	1	62,45	81,03	90,09	93,05

Номер задания / критерия оценивания в КИМ	Количество полученных первичных баллов	Процент участников экзамена в Алтайском крае, получивших соответствующий первичный балл за выполнения задания в группах участников экзамена с разными уровнями подготовки			
		в группе не преодолевших минимальный балл, %	в группе от минимального до 60 т.б., %	в группе от 61 до 80 т.б., %	в группе от 81 до 100 т.б., %
11	X	25,69	14,93	0,92	0,00
	0	71,15	69,21	53,92	18,18
	1	3,16	15,86	45,16	81,82
12	X	49,41	25,66	2,30	0,00
	0	49,01	47,12	20,05	4,81
	1	1,58	27,22	77,65	95,19
13	X	41,90	28,77	6,22	0,00
	0	56,92	49,92	25,35	5,88
	1	1,19	21,31	68,43	94,12
14	X	55,73	43,86	9,22	1,60
	0	43,87	44,32	36,64	11,23
	1	0,40	11,82	54,15	87,17
15	X	47,43	38,88	8,76	0,00
	0	51,78	36,08	11,98	1,60
	1	0,79	25,04	79,26	98,40
16	X	48,62	29,39	7,60	1,07
	0	41,90	28,46	18,89	2,67
	1	9,49	42,15	73,50	96,26
17	X	69,96	73,56	23,27	0,00
	0	29,64	23,48	34,33	6,95
	1	0,40	2,95	42,40	93,05
18	X	51,38	25,04	3,92	0,53
	0	44,27	48,06	30,88	10,70
	1	4,35	26,91	65,21	88,77

Номер задания / критерия оценивания в КИМ	Количество полученных первичных баллов	Процент участников экзамена в Алтайском крае, получивших соответствующий первичный балл за выполнения задания в группах участников экзамена с разными уровнями подготовки			
		в группе не преодолевших минимальный балл, %	в группе от минимального до 60 т.б., %	в группе от 61 до 80 т.б., %	в группе от 81 до 100 т.б., %
19	X	30,04	11,51	0,46	0,00
	0	45,85	24,26	8,99	2,67
	1	24,11	64,23	90,55	97,33
20	X	44,27	20,06	0,69	0,00
	0	55,34	41,37	13,59	1,60
	1	0,40	38,57	85,71	98,40
21	X	45,06	24,26	2,07	0,00
	0	52,96	43,23	16,36	4,28
	1	1,98	32,50	81,57	95,72
22	X	50,99	35,15	12,44	3,21
	0	44,27	46,35	46,77	23,53
	1	4,74	18,51	40,78	73,26
23	X	49,80	31,88	3,46	0,00
	0	47,04	36,86	15,21	3,21
	1	3,16	31,26	81,34	96,79
24	X	73,52	74,81	63,82	18,18
	0	26,48	24,73	32,03	47,06
	1	0,00	0,47	4,15	34,76
25	X	91,30	91,29	69,59	13,90
	0	8,70	8,71	22,12	28,88
	1	0,00	0,00	8,29	57,22
26	X	81,03	85,38	76,50	33,69
	0	18,97	14,46	20,97	33,16
	1	0,00	0,16	1,61	10,16

Номер задания / критерия оценивания в КИМ	Количество полученных первичных баллов	Процент участников экзамена в Алтайском крае, получивших соответствующий первичный балл за выполнения задания в группах участников экзамена с разными уровнями подготовки			
		в группе не преодолевших минимальный балл, %	в группе от минимального до 60 т.б., %	в группе от 61 до 80 т.б., %	в группе от 81 до 100 т.б., %
	2	0,00	0,00	0,92	22,99
27	X	84,98	87,09	68,43	8,02
	0	14,62	11,98	14,52	8,02
	1	0,00	0,93	11,29	34,76
	2	0,40	0,00	5,76	49,20

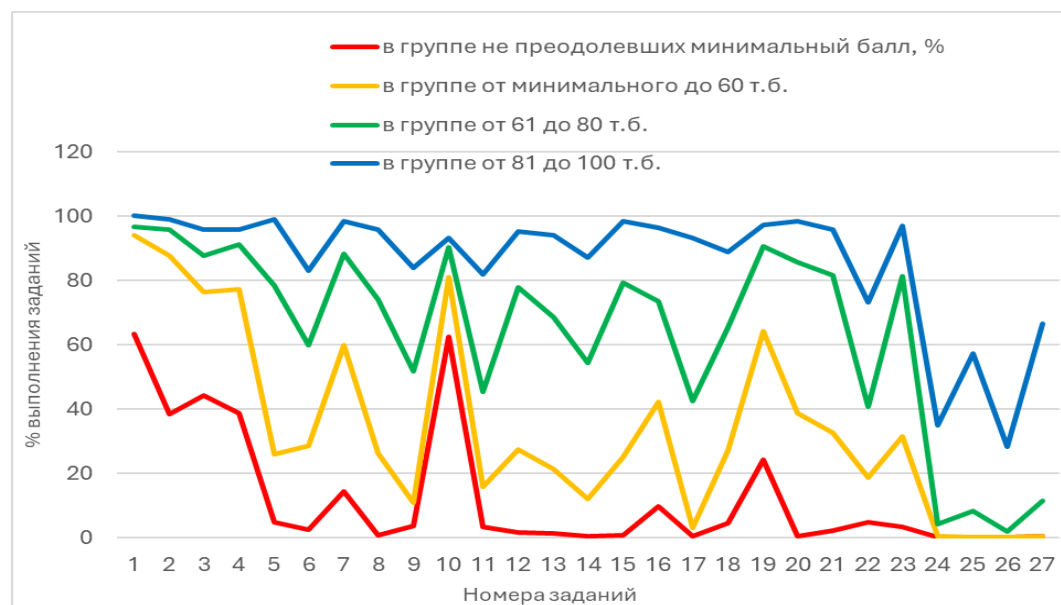


Рисунок 2-5. Выполнение заданий ЕГЭ 2025 г. участниками с разными уровнями подготовки

В целом показатели результативности выполнения большинства заданий либо сохранили значения предыдущего года, либо имеют положительную динамику (таблица 2-13, рисунок 2-4). Незначительное ухудшение результатов зафиксировано по заданиям 4 – на 5,87% (Умение кодировать и декодировать информацию), 9 – на 7,68% (Умение об-

рабатывать числовую информацию в электронных таблицах), 14 – на 4,69% (Знание позиционных систем счисления), 16 на 4,66% (Вычисление рекуррентных выражений), 20 – на 6,93% (Умение найти выигрышную стратегию игры). Более сильное ухудшение результатов зафиксировано в разрезе заданий 12 – на 16,25% (Умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд), 25 – на 9,57% (Умение создавать собственные программы для обработки целочисленной информации).

Все задания, которые были выделены в 2024 году как задания сложные для участников ЕГЭ в 2025 году, такими и остались, кроме задания 22: задания 5, 6, 8, 9 – задания базового уровня, задания 24, 25, 26, 27 – задания высокого уровня, по которым зафиксирован крайне низкий процент выполнения.

По линиям заданий, с которыми экзаменуемые справились на высоком уровне в прошлом году, результат сохранен или улучшен в текущем году. Особенно хорошо освоены всеми группами экзаменуемых (таблица 2-14, рисунок 2-5) следующие темы: «Графы» (задание 1), «Таблицы истинности» (задание 2), «Базы данных» (задание 3), «Кодирование и декодирование информации» (задание 4), «Информационный поиск в тексте» (задание 10).

При сравнении графиков с показателями выполнения заданий разными группами участников экзамена (рисунок 2-5) можно заметить, что линии по многим заданиям «пики» и «провалы» одинаково повторяются заданий. Таким образом, можно констатировать, что выявленные проблемы с изучением определенной темы касаются не только учеников, сдавших экзамен на «2» или «3», но и остальных участников экзамена. Например, очевидно несоответствие результатов выполнения заданий базового и повышенного уровня сложности 6, 9, 11, 14, 17 и 22, результатам выполнения остальных заданий тех же уровней сложности во всех группах (резкие провалы графиков вниз).

3.1.1.2. Выявление сложных для участников ЕГЭ заданий

Задания базового уровня (с процентом выполнения ниже 50)

Задание 5: средний процент выполнения 46,34%, что ниже на 1,66%, чем в 2024 г., но на 14,52% выше, чем в 2023 г., однако остается ниже допустимой границы.

Задание направлено на проверку умения формально исполнять простой алгоритм, записанный на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные данные линейного алгоритма по результатам его работы.

В группе участников, не преодолевших минимальный балл, с этим заданием справились лишь 4,74% ученика, при этом, совсем не приступали к его выполнению треть участников этой группы (35,57%), а 0 баллов получили более

половины (59,68%).

В группе участников, получивших на экзамене от минимального до 60 т.б. это задание также имеет низкий процент выполнения (25,82%) и четверть (25,51%) вовсе не приступали к его выполнению, остальные выполнили задание неверно. Эти значения имеют большой разрыв с высокими показателями выполнения этого задания в группе от 61 до 80 т.б. – 78,34%, а значит есть потенциал к повышению результата группы 41-60 т.б.

В группе получивших от 81 до 100 т.б. с этим заданием справились почти все участники – 98,93%.

Задание 6: средний процент выполнения 39,82%, что хоть и выше на 17,31%, чем в 2023 г., однако остается на том же низком уровне что и в 2024 г.

Задание направлено на проверку умения определять возможные результаты работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов. В частности, в формулировке задания требовался анализ алгоритма для исполнителя «Черепашка».

В группе участников, не преодолевших минимальный балл, с этим заданием справились лишь 2,37% учеников, при этом, менее, чем 1/5 часть не знали, как приступить к заданию (17,79%), остальные же приступили, но не справились. То есть, тема знакома даже слабо подготовленным ученикам, задание кажется доступным, но они допускают ошибки в решении, значит, необходимо проводить целенаправленную работу над этими ошибками при подготовке школьников.

При сравнении графиков с показателями выполнения заданий разными группами участников экзамена (рисунок 2-5) видно, что результаты выполнения задания 6 ниже результатов выполнения многих заданий базового уровня сложности во всех группах. Таким образом, выявлена общая низкая способность выпускников школ моделировать и анализировать несложные алгоритмы с циклами, условными операторами, связанные с построением чертежей.

В группе участников, получивших на экзамене от минимального до 60 т.б. это задание также имеет недопустимо низкий процент выполнения (28,46%), не приступали к заданию всего 6,07% учеников.

В группе от 61 до 80 т.б. результаты статистически допустимые – 59,91%, но близкие к пограничному значению. В группе получивших от 81 до 100 т.б. с этим заданием справились 82,89%.

Задание 8: средний процент выполнения 44,23%, этот показатель вырос на 14,02% за два года, с 2024 г. он вырос на 9,23%, однако остается ниже допустимой границы.

Задание направлено на проверку знаний и умений кодировать данные, применять комбинаторику и системы счисления при работе с равномерными кодами.

В группе участников, не преодолевших минимальный балл, с этим заданием справились лишь 0,79% ученика,

при этом, совсем не приступали к его выполнению больше четверти участников этой группы (28,06%), а 0 баллов получили 71,15%.

В группе участников, получивших на экзамене от минимального до 60 т.б. это задание также имеет низкий процент выполнения (26,28%) и половина (53,19%) вовсе не приступали к его выполнению, остальные выполнили задание неверно. Эти значения имеют большой разрыв с высокими показателями выполнения этого задания в группе от 61 до 80 т.б. – 73,96%, а значит есть потенциал к повышению результата группы 41-60 т.б.

В группе получивших от 81 до 100 т.б. с этим заданием справился высокий процент учеников – 95,75%.

Задание 9: это задание остается в течение трех лет самым сложным для выполнения среди участников ЕГЭ по информатике, средний процент выполнения 30,32%, на 14,78% выше, чем в 2023 г., но ниже, чем в 2024 г. на 7,68%.

Задание направлено на проверку базовых умений обрабатывать числовую информацию в электронных таблицах. В целом в задании необходимо разработать алгоритм вычисления, выделив нужные этапы. Как и в задании 6, здесь важным этапом является построение модели вычислений, в этом и проявляется главная сложность для учеников. Пользоваться функциями электронных таблиц и вычислять формулы они умеют, а вот в составлении необходимой формулы проблема.

В группе участников, не преодолевших минимальный балл, с этим заданием справились лишь 3,56% учеников, при этом, чуть менее половины не приступили к заданию (45,85%), остальная половина не справилась с решением.

При сравнении графиков с показателями выполнения заданий разными группами участников экзамена (рисунок 2-5) видно, что как и в задании 6 выполнения задания 9 ниже результатов всех заданий базового уровня сложности во всех группах. Таким образом, выявлена общая низкая способность выпускников школ моделировать и выполнять вычисления в электронных таблицах.

В группе участников, получивших на экзамене от минимального до 60 т.б. это задание также имеет недопустимо низкий процент выполнения (10,89%), не приступали к заданию половина участников группы (46,5%).

В группе от 61 до 80 т.б. результаты статистически допустимые – 51,61%, но близкие к пограничному значению.

В группе получивших от 81 до 100 т.б. с этим заданием справились 83,96%.

Задания повышенного и высокого уровня (с процентом выполнения ниже 15)

Среди заданий повышенного уровня сложности нет заданий, которые по среднему проценту выполнения не входят в статистически допустимые границы. Остальные сложные для выполнения задания являются имеют высокий уровень и направлены на проверку умений программировать.

Задание 24: это задание остается в течение трех лет самым сложным заданием высокого уровня для выполнения среди участников ЕГЭ по информатике, средний процент выполнения 5,67%, это на 4,01% ниже, чем в 2023 г., но на 3,67% выше, чем в 2024 г.

Задание направлено на проверку умений создавать собственные программы (10–20 строк) для обработки символической информации.

Бессмысленно анализировать задания высокого уровня сложности на программирование для группы учеников, не преодолевших минимальный балл, для них процент выполнимости этих заданий равен 0.

В группе участников, получивших на экзамене от минимального до 60 т.б., с этим заданием также никто не справился (0,47%), да и приступали к выполнению лишь четверть из группы (25,2%).

В группе от 61 до 80 т.б. с этим заданием справились лишь 4,15%, при том, что приступали к его выполнению только 35,18% человек из группы.

В группе высокобалльников (от 81 до 100 т.б.) с этим заданием справилась чуть больше трети учеников (34,76%), хотя приступали к выполнению уже 81,82% участников.

Задание 25: это задание в течение трех лет стабильно теряет показатель выполнимости: средний процент выполнения 9,43%, это на 25,54% ниже, чем в 2023 г. и на 9,57% ниже, чем в 2024 г.

Задание направлено на проверку умений создавать собственные программы (10–20 строк) для обработки целочисленной информации.

Интересно, что в группах учеников, не преодолевших минимальный балл и получивших на экзамене от минимального до 60 т.б., абсолютно одинаковые результаты выполнения задания 25: не приступали 91,3%, из остальных никто не справился с выполнением.

В группе от 61 до 80 т.б. с этим заданием справились лишь 8,29%, при том, что приступали к его выполнению только 30,41% человек из группы.

В группе высокобалльников (от 81 до 100 т.б.) с этим заданием справилась половина учеников (57,22%), не приступали к выполнению только 13,9% участников.

Задание 26: средний процент выполнения этого задания остается практически на том же уровне, что и в 2023–2024 г.г. – 3,99%, с незначительным понижением на 2% за два года. Задание оценивается максимально двумя баллами.

Задание направлено на проверку умений с помощью программирования обрабатывать целочисленную информацию с использованием сортировки.

В группах учеников, не преодолевших минимальный балл и получивших на экзамене от минимального до 60

т.б. никто не справился с выполнением задания.

В группе от 61 до 80 т.б. с этим заданием справились полностью лишь 0,92%, а на 1 балл – 1,61%. Приступали к выполнению задания только 23,5% человек из группы.

В группе высокобалльников (от 81 до 100 т.б.) с этим заданием справилась 22,9% учеников, на 1 балл задание выполнили 10,16%, не приступали к выполнению только 33,69% участников.

Задание 27: средний процент выполнения 11,73%, с повышением 7,33% за два года. Задание оценивается максимально двумя баллами.

Задание направлено на проверку умений выполнять последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и построение модели, преобразование данных, визуализация данных, интерпретация результатов. В 2025 г. Новой использовалась формулировка задания нового прототипа задания, отличающегося от прошлых лет, возможно, причина повышения показателя в этом. Это задание требует больших усилий при моделировании и отборе значений для обработки, но в разработке программы тривиально, по сравнению с заданиями №27 прошлых лет.

В группе учеников, не преодолевших минимальный балл, с выполнением задания справились 0,4%. Всего приступали к выполнению 14,02%.

В группе учеников, получивших на экзамене от 41- 60 т.б. справились с заданием на 1 балл 0,93%, но полностью верно задание не выполнил никто. Всего приступали к выполнению 13% из группы.

В группе от 61 до 80 т.б. с этим заданием справились полностью 5,76%, а на 1 балл – 11,29%. Всего приступали к выполнению задания только 31,57% человек из группы.

В группе высокобалльников (от 81 до 100 т.б.) с этим заданием полностью справилась 5,76% учеников из группы, на 1 балл задание выполнили 34,76%, не приступали к выполнению немногие, всего 8,02%.

Таким образом, очевидно, что программированием владеют лишь обучающиеся, вошедшие в группу высокобалльников.

Прочие задания

Среди прочих, стоит отметить **задание 17:** повышенный уровень сложности, проверяет базовые умения составлять алгоритм обработки числовой последовательности и записывать его в виде простой программы (10–15 строк) на языке программирования. В целом средний процент его выполнения соответствует норме, во всех группах участников экзамена (кроме высокобалльников) это задание имеет самый низкий показатель выполнимости, по сравнению с зада-

ниями повышенного уровня. По сравнению с 2024 г. процент выполнения понизился на 3,08%.

Ожидаемо низкий процент выполнения задания в группах учеников, не преодолевших минимальный балл и получивших на экзамене от минимального до 60 т.б. – 0,4% и 2,95% соответственно.

В группе от 61 до 80 т.б. с этим заданием достойно справилась 42,4% ученика, не приступали к выполнению около четверти участников группы (23,27%).

В группе высокобалльников (от 81 до 100 т.б.) с заданием справилась 93,05% ученика, причем нет ни единого человека, не приступившего к выполнению задания в этой группе.

3.1.1.3. Прочие результаты статистического анализа

Традиционно высокие показатели подготовки участники экзамена демонстрируют в разрезе заданий: №1 – средний процент выполнения 90,38% (базовый уровень, проверяет умение представлять и считывать данные в разных типах информационных моделей (схемы, карты, таблицы, графики и формулы)); №2 – средний процент выполнения 83,19% (базовый уровень, проверяет умение строить таблицы истинности и логические схемы); №3 – средний процент выполнения 76,53% (базовый уровень, проверяет умение поиска информации в реляционных базах данных); №4 – средний процент выполнения 77,13% (базовый уровень, проверяет умение кодировать и декодировать информацию, строить оптимальные неравномерные коды).

Стабильно хорошие знания и сформированные умения показывают школьники в области анализа алгоритмов игры и поиска выигрышной стратегии: задания №19-22 (базового, повышенного и высокого уровня) – средний процент выполнения соответственно 69,15%, 53,07%, 49,24%.

Отметим, что на 31% относительно результатов 2024 повысился результат выполнения задания базового уровня сложности №10, проверяющего умения информационного поиска средствами текстового процессора.

3.1.2. Содержательный анализ выполнения заданий КИМ

На основе данных, приведенных в п. 3.1.1. по каждому выявленному сложному заданию приведем характеристики задания, типичные при выполнении этих заданий ошибки, проведем анализ возможных причин получения выявленных типичных ошибочных ответов и путей их устранения в ходе обучения школьников предмету.

Примеры и характеристики заданий приводятся на основе открытого варианта, использованного в регионе №319.

Задание № 5

Тема: Выполнение и анализ простых алгоритмов.

Уровень сложности: базовый.

Рекомендуемое время выполнения: 4 минуты.

Проверяется умение формально исполнять простой алгоритм, записанный на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные данные линейного алгоритма по результатам его работы.

Проверяемые элементы содержания: Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов. Определение исходных данных, при которых алгоритм может дать требуемый результат.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение анализировать алгоритмы с использованием таблиц трассировки; определять без использования компьютера результаты выполнения несложных программ, включающих циклы, ветвления и подпрограммы, при заданных исходных данных.

Проверяемые метапредметные результаты: Базовые логические действия. Базовые исследовательские действия.

Что нужно знать:

сумма двух цифр в десятичной системе счисления находится в диапазоне от 0 до 18 (9+9);

в некоторых задачах нужно иметь представление о системах счисления (могут использоваться цифры восьмеричной, шестнадцатеричной и других систем счисления);

бит чётности – это дополнительный контрольный бит, который добавляется к двоичному коду так, чтобы количество единиц в полученном двоичном коде стало чётным; если в исходном коде уже было чётное количество единиц, дописывается 0, если нечётное – дописывается 1;

при добавлении к записи числа нуля справа число увеличивается в q раз, где q – это основание системы счисления;

чтобы отбросить последнюю цифру в записи числа, нужно разделить число на основание системы счисления нацело (остаток отбрасывается).

Пример формулировки задания

На вход алгоритма подаётся натуральное число N . Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N .
2. Далее эта запись обрабатывается по следующему правилу:
 - а) если число N делится на 3, то к этой записи дописываются три последние двоичные цифры;
 - б) если число N на 3 не делится, то остаток от деления умножается на 3, переводится в двоичную запись и дописывается в конец числа.

Полученная таким образом запись является двоичной записью искомого числа R .

3. Результат переводится в десятичную систему и выводится на экран.

Например, для исходного числа $12_{10} = 1100_2$ результатом является число $1100100_2 = 100_{10}$, а для исходного числа $4_{10} = 100_2$ это число $10011_2 = 19_{10}$.

Укажите максимальное число R , не превышающее 208, которое может быть получено с помощью описанного алгоритма.

В ответе запишите это число в десятичной системе счисления.

Решение (Вариант 1)

Очевидно, что простое перебирание чисел позволяет выявить закономерность. Явно исходные числа больше 4, так как у числа 3 в двоичной записи не найдется трех последних цифр в двоичном представлении.

$$4 = 100_2 = 10011_2 = 19$$

$$5 = 101_2 = 101110_2 = 46$$

$$6 = 110_2 = 110110_2 = 54$$

$$7 = 111_2 = 11111_2 = 31$$

$$8 = 1000_2 = 1000110_2 = 70$$

$$9 = 1001_2 = 1001001_2 = 73$$

$$10 = 1010_2 = 101011_2 = 43$$

$$11 = 1011_2 = 1011110_2 = 94$$

$$12 = 1100_2 = 1100100_2 = 100$$

И т.д.

Результат в двоичном представлении имеет вид: либо совпадают две последние тройки цифр (значит исходное число делилось на 3), либо последние две цифры «11» или «110» (значит исходное число делилось на 3 с остатком 1 или 2).

Рассмотрим двоичное представление чисел, меньших 208, и выберем большее из них, подходящее под описание

выше, для экономии времени можно воспользоваться средствами электронных таблиц для перевода чисел (=ОСНОВАНИЕ (число; основание) и =ДЕС (число; основание)):

	A	B	C
1	N	N2	Получено из:
2	207	11001111	
3	206	11001110	25
4	205	11001101	
5	204	11001100	
6	203	11001011	50
7	202	11001010	
8	201	11001001	25
9	200	11001000	
10	199	11000111	49
11	198	11000110	
12	197	11000101	
13	196	11000100	
14	195	11000011	
15	194	11000010	

Наибольшее похожее на подходящее значение в двоичной системе счисления $206 = 11001110_2$, оно заканчивается на «110», но если отбросить эти три цифры, то оставшееся значение $11001_2 = 25$ делится на 3 с остатком 1 и при выполнении заданного алгоритма к двоичному значению должно было добавиться «11», а не «110». Таким образом, число 206 не подходит.

При аналогичных рассуждениях находим ближайшее подходящее значение 199.

Ответ: 199

Решение (Вариант 2)

При выполнении этого задания, используя те же рассуждения, можно облегчить себе работу при переводе десятичных чисел в двоичную систему счисления если использовать Калькулятор Windows в режим Программист (Вид – Программист или Alt+3)

Решение (Вариант 3)

Программирующий ученик вполне может реализовать перебор чисел и проверку выполнения условий для преобразованных значений по описанному алгоритму с применением языка программирования.

Пример программы на языке Python:

```
def process_number(n):
    # Получаем двоичное представление числа n
    binary_representation = bin(n)[2:]
    if n % 3 == 0:
        # Добавляем последние три бита исходного числа
        result_binary = binary_representation + binary_representation[-3:]
    else:
        remainder = (n % 3) * 3
        # Преобразуем остаток в двоичный вид и добавляем к исходному числу
        remainder_binary = bin(remainder)[2:] # Убираем префикс '0b'
        result_binary = binary_representation + remainder_binary
    return int(result_binary, 2)

# Найти максимум R <= 208
max_R = 0
for i in range(1, 209):
    r = process_number(i)
    if r > max_R and r <= 208:
        max_R = r

print(max_R)
```

Ответ: 199

Анализ ошибок

Наиболее быстрый способ выполнения этого задания, это применение навыков программирования. Это возможно лишь при владении языками и методами программирования. В противном случае, выполнение задания

занимает гораздо больше времени.

Неверные ответы участников практически не повторяются, что говорит о случайном характере ошибок. Чаще, это неверное прочтение задания (ищут не минимальный, а максимальный результат, или результат в двоичной форме, а не в десятичной системе счисления, или исходное число, из которого получен нужный результат).

Способы избежать таких ошибок:

Если не выйти на проверку заданных условий, то неважно, какой инструмент решения задачи будет выбран, так как в случае правильной проверки необходимых условий задание легко выполняется разными способами при хорошем владении методами рассуждения или представленным инструментарием в виде доступного ПО.

При подготовке учеников важно сформировать комплекс умений: читать и выполнять алгоритм, переводить числа в разные системы счисления; отделять цифры числа, логически рассуждать.

В п. 3.1.1. отмечено, что тема знакома даже слабо подготовленным ученикам и задание кажется доступным, так как большинство участников экзамена приступало к его выполнению, но они допускали ошибки в решении, значит, необходимо проводить целенаправленную работу над этими ошибками при подготовке в школе, использовать разные способы перевода числе в разные системы счисления.

Задание № 6

Тема: Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов.

Уровень сложности: базовый.

Рекомендуемое время выполнения: 4 минуты.

Проверяется умение определять возможные результаты работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов.

Проверяемые элементы содержания: Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов. Определение исходных данных, при которых алгоритм может дать требуемый результат.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение анализировать алгоритмы с использованием таблиц трассировки; определять без использования компьютера результаты выполнения несложных программ, включающих циклы, ветвления и подпрограммы, при заданных исходных данных.

Проверяемые метапредметные результаты: Базовые логические действия. Базовые исследовательские действия.

Что нужно знать:

выполнять ручную прокрутку программы для исполнителя, в которой используется цикл; строить на координатной плоскости фигуру, которую нарисует Черепаха (при ее известном начальном положении);

анализировать чертеж и строить математическую модель вычисления количества целочисленных точек, площади и пр.

Пример формулировки задания

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует 6 команд: **Поднять хвост**, означающая переход к перемещению без рисования; **Опустить хвост**, означающая переход в режим рисования; **Вперёд n** (где n – целое число), вызывающая передвижение Черепахи на n единиц в том направлении, куда указывает её голова; **Назад n** (где n – целое число), вызывающая передвижение в противоположном голове направлении; **Направо m** (где m – целое число), вызывающая изменение направления движения на m градусов по часовой стрелке, **Налево m** (где m – целое число), вызывающая изменение направления движения на m градусов против часовой стрелки.

Запись **Повтори k [Команда1 Команда2 ... КомандаS]** означает, что последовательность из S команд повторится k раз.

Черепахе был дан для исполнения следующий алгоритм.

Повтори 2 [Вперёд 13 Направо 90 Вперёд 20 Направо 90]

Поднять хвост

Вперёд 8 Направо 90 Назад 3 Налево 90

Опустить хвост

Повтори 2 [Вперёд 16 Направо 90 Вперёд 8 Направо 90]

Определите, сколько точек с целочисленными координатами находятся внутри объединения фигур, ограниченного заданными алгоритмом линиями, включая точки на линиях.

Решение (Вариант 1)

Разберемся с задачей и поэтапно выполним предложенный алгоритм, определив вид полученного чертежа.

Исходные данные и состояние черепахи перед началом:

- Координаты (x; y): (0; 0)
- Ориентация головы: вверх (+Y)
- Хвост опущен.

Первый блок алгоритма:

Повтори 2 раза:

Вперед 13 → движение вверх: координата Y увеличивается на 13. Результат: (0;13).

Направо 90° → поворот направо, теперь ориентация вправо (+X).

Вперед 20 → движение вправо: координата X увеличивается на 20. Результат: (20;13).

Направо 90° → поворот направо, теперь ориентация вниз (-Y).

Вперед 13 → движение вниз: координата Y уменьшается на 13. Результат: (20; 0).

Направо 90° → поворот направо, теперь ориентация влево (-X).

Вперед 20 → движение влево: координата X уменьшается на 20. Результат: (0; 0).

Направо 90° → поворот направо, теперь ориентация вниз (+Y).

Этот блок создает прямоугольник размером 20×13.

Следующие команды вне цикла:

Поднять хвост → далее движемся без рисования линий.

Вперед 8 → движение вверх: координата Y увеличивается на 8. Результат: (0; 8).

Направо 90° → поворот направо, теперь ориентация вправо (+X).

Назад 3 → движение назад: координата X уменьшается на 3. Результат: (-3; 8).

Налево 90° → поворот налево, теперь ориентация вверх (+Y).

Опустить хвост → вновь начинаем рисовать линию.

Второй блок алгоритма:

Повтори 2 раза:

Вперед 16 → движение вверх: координата Y увеличивается на 16. Результат: (-3; 24).

Направо 90° → поворот направо, теперь ориентация вправо (+X).

Вперед 8 → движение вправо: координата X увеличивается на 8. Результат: (5; 24).

Направо 90° → поворот направо, теперь ориентация вниз (-Y).

Вперед 16 → движение вниз: координата Y уменьшается на 16. Результат: (5; 8).

Направо 90° → поворот направо, теперь ориентация влево (-X).

Вперед 8 → движение влево: координата X уменьшается на 8. Результат: (-3; 8).

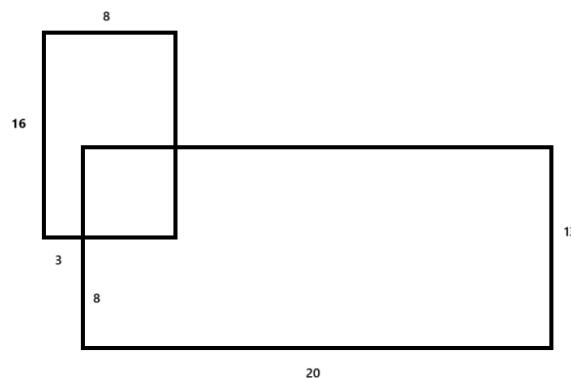
Направо 90° → поворот направо, теперь ориентация вниз (+Y).

Этот блок создаёт второй прямоугольник размером 8×16.

Результаты анализа:

- Мы получили два замкнутых контура: первый прямоугольник размером 20×13, второй прямоугольник размером 8×16.
- Эти контуры перекрываются частично.

Примерное изображение чертежв:



Необходимо посчитать количество целых точек внутри объединенной области двух фигур, включая границы.

Определение целочисленных точек:

Первый прямоугольник (20×13):

- Горизонтальная ось: от 0 до 20 включительно (21 точка).
- Вертикальная ось: от 0 до 13 включительно (14 точек).
- Количество точек в первом прямоугольнике: $21 \times 14 = 294$

Второй прямоугольник (8×16):

- Горизонтальная ось: от -3 до 5 включительно (9 точек).
- Вертикальная ось: от 8 до 24 включительно (17 точек).
- Количество точек во втором прямоугольнике: $9 \times 17 = 153$

Однако фигуры пересекаются частично, поэтому нужно учесть общие точки пересечения, чтобы избежать двойного счета.

Пересечение:

Определим пересечение:

Для нахождения точного пересечения этих прямоугольников обратимся к методике определения общих границ.

Верхняя граница первого прямоугольника – $y=13$.

Нижняя граница второго прямоугольника – $y=8$.

Значит, вертикальное пересечение идёт от $y=8$ до $y=13$.

Горизонтальное пересечение:

Левая сторона первого прямоугольника – $x=0$.

Правая сторона второго прямоугольника – $x=5$.

Значит, горизонтальное пересечение идёт от $x=0$ до $x=5$.

Границы пересечения:

Нижний левый угол пересечения: $(0,8)$, верхний правый угол пересечения: $(5,13)$.

Точные диапазоны пересеченных точек:

По оси x : от 0 до 5 включительно (всего 6 точек).

По оси y : от 8 до 13 включительно (всего 6 точек).

Количество точек пересечения: $6 \times 6 = 36$

Применение принципа включений и исключений:

Общее количество точек в объединении: $|A \cup B| = |A| + |B| - |A \cap B|$

Подставляем полученные значения и получим искомое количество точек: $294 + 153 - 36 = 411$

Ответ: 411

Решение (Вариант 2)

Этот вариант для программирующих учеников.

Пример программы на языке Python, включающей визуализацию чертежа с применением библиотеки Turtle и расчет количества целочисленных точек прямоугольников:

```
import turtle
from math import floor, ceil

# Создание окна для рисования
window = turtle.Screen()
window.bgcolor("lightyellow")
window.title("Рисунок Черепашки")

# Настроим Черепашку
t = turtle.Turtle()
t.shape("turtle") # Форма черепашки
t.color("black", "darkgreen") # Цвет линий и заливки
t.speed(0) # Максимально быстрая скорость

# Начинаем рисовать
t.begin_fill()

# Первая фигура (прямоугольник 20x13)
for _ in range(2):
    t.forward(13) # Вперед на 13 единиц
    t.right(90)  # Повернуть направо на 90 градусов
    t.forward(20) # Вперед на 20 единиц
    t.right(90)  # Повернуть направо на 90 градусов

# Подняли хвост (не будем рисовать дальше)
```

```

t.penup()
t.forward(8) # Переместились вперед на 8 единиц
t.right(90) # Повернули направо
t.backward(3) # Назад на 3 единицы
t.left(90) # Налево повернулись
t.pendown() # Опустили хвост (начали рисовать)

# Вторая фигура (прямоугольник 8x16)
for _ in range(2):
    t.forward(16) # Вперед на 16 единиц
    t.right(90) # Повернуть направо на 90 градусов
    t.forward(8) # Вперед на 8 единиц
    t.right(90) # Повернуть направо на 90 градусов

# Завершение рисунка
t.end_fill()

# Функция для подсчёта целочисленных точек в прямоугольнике
def count_points_in_rectangle(x_min, x_max, y_min, y_max):
    """
    Возвращает количество целочисленных точек внутри прямоугольника
    """
    return (ceil(x_max) - floor(x_min) + 1) * (ceil(y_max) - floor(y_min) + 1)

# Параметры прямоугольников
first_rect = {
    'x_min': 0, 'x_max': 20,
    'y_min': 0, 'y_max': 13
}

```

```

second_rect = {
    'x_min': -3, 'x_max': 5,
    'y_min': 8, 'y_max': 24
}

# Пересечение прямоугольников
intersection = {
    'x_min': max(first_rect['x_min'], second_rect['x_min']),
    'x_max': min(first_rect['x_max'], second_rect['x_max']),
    'y_min': max(first_rect['y_min'], second_rect['y_min']),
    'y_max': min(first_rect['y_max'], second_rect['y_max'])
}

# Количество точек в каждом прямоугольнике
points_first_rect = count_points_in_rectangle(
    first_rect['x_min'], first_rect['x_max'],
    first_rect['y_min'], first_rect['y_max']
)

points_second_rect = count_points_in_rectangle(
    second_rect['x_min'], second_rect['x_max'],
    second_rect['y_min'], second_rect['y_max']
)

# Если нет пересечения
if intersection['x_min'] >= intersection['x_max'] or intersection['y_min'] >= intersection['y_max']:
    points_intersection = 0
else:
    points_intersection = count_points_in_rectangle(

```

```

        intersection['x_min'], intersection['x_max'],
        intersection['y_min'], intersection['y_max']
    )

# Общее количество точек (формула включений и исключений)
total_points = points_first_rect + points_second_rect - points_intersection

# Вывод результата
print(f"Общее количество целочисленных точек: {total_points}")

# Закрытие окна после нажатия клавиши
window.exitonclick()

```

Ответ: 411

Анализ ошибок

Эта задача вызывает трудности у школьников чаще всего из-за ряда распространенных ошибок. Вот наиболее часто встречающиеся проблемы и советы по их устранению:

1. Неправильно определены координаты фигур:

Некоторые ученики неправильно определяют верхние/нижние, левые/правые границы фигур. Например, считают, что прямоугольник расположен иначе, чем он фактически размещен.

Решение: Всегда внимательно следуйте инструкциям, отмечайте конечные точки, обращая внимание на повороты черепахи и направление движений.

2. Ошибочный расчет пересечения:

Многие ошибаются в вычислении площади пересечения. Часто проблема возникает из-за неправильного понимания границ пересечения или отсутствия учета частичных клеток.

Решение: Используйте аккуратный геометрический подход, выделяйте чётко левую верхнюю и правую нижнюю точки пересечения и применяйте правильную формулу для подсчета целочисленных точек.

3. Непонимание метода включений и исключений:

Иногда учащиеся забывают применять принцип включений и исключений, считая либо лишнюю область, либо наоборот пропускают общую часть.

Решение: Помните простую формулу: общая площадь равна сумме площадей отдельных частей минус пересечение. Это гарантирует отсутствие дублирования или пропуска точек.

4. Недопонимание условий задачи или невнимательное прочтение:

Часто ученики начинают считать точки сразу, не разобравшись детально в порядке прохождения черепахи и формировании фигур. Часто вместо объединения ищут пересечение фигур.

Решение: Обязательно необходимо внимательно прочесть текст условия до конца, даже если на первый взгляд условие уже видели и решали такую задачу. Прежде чем приступать к решению, обязательно сделайте набросок маршрута черепахи карандашом на бумаге. Проверьте каждую команду и убедитесь, что правильно поняли траекторию движения. Это важно для моделирования алгоритма вычислений, причем важно при любом выбранном способе решения.

5. Ошибка в анализе направления движения:

Важно учитывать повороты черепахи (налево-направо) и влияние на последующие передвижения вперед и назад.

Решение: Постоянно контролируйте направление черепахи после каждого поворота, особенно при движении назад.

6. Сложности с отсечением дробных частей:

Иногда возникают сложности с округлением и определением количества целочисленных точек на границах фигур.

Решение: Пользуйтесь функциями округления (`ceil` и `floor`) для точности расчетов в случае выбора программного способа решения. Округляйте минимальное значение вверх, а максимальное вниз, чтобы включить все необходимые клетки.

Способы избежать таких ошибок:

Если не выйти на проверку заданных условий, то неважно, какой инструмент решения задачи будет выбран, так как в случае правильной проверки необходимых условий задание легко выполняется разными способами при хорошем владении методами рассуждения или представленным инструментарием в виде доступного ПО.

При подготовке учеников важно сформировать комплекс умений: читать и выполнять алгоритм, логически рассуждать, моделировать, находить объединение, пересечение, площадь, периметр фигур.

В п. 3.1.1. отмечено, что тема знакома даже слабо подготовленным ученикам и задание кажется доступным. Из всех участников экзамена к выполнению задания не приступили лишь 5,7%, остальные приступили к заданию, но они

допускали ошибки в решении. Таким образом, выявлена общая низкая способность выпускников школ моделировать и анализировать несложные алгоритмы с циклами, условными операторами, связанные с построением чертежей. Значит, необходимо проводить целенаправленную работу над вышеперечисленными проблемами при подготовке в школе, использовать разные способы решения. Причем, эти проблемы не должны выявляться и исправляться в 10-11 классе, целенаправленное формирование необходимых умений должно начинаться с 7 класса.

В 7 классе в разделе «Компьютерная графика» при изучении темы «Графический редактор» стоит использовать задания на построение чертежей с выделением областей пересечения, объединения фигур, заданных текстовым описанием с указанием координат вершин, длин сторон и пр.

В 8 классе в разделе «Алгоритмы и программирование» при изучении тем «Исполнители и алгоритмы», «Анализ алгоритмов» стоит использовать задания не только на разработку алгоритмов для построения чертежей, но и на анализ заданного алгоритма с выделением областей пересечения, объединения фигур и пр.

В 9 классе в теме «Моделирование как метод познания» и в разделе «Алгоритмы и программирование» так же стоит применять задачи на моделирование для расчета количества точек в областях пересечения, объединения фигур и пр.

В 10-11 классах при изучении раздела «Алгоритмизация и программирование» следует включать в программу отработку тех же навыков, но с применением языка программирования.

Таким образом выполняется преемственность программ обучения и формируются умения, проверяемые на ГИА.

Задание № 8

Тема: Кодирование данных, комбинаторика, системы счисления.

Уровень сложности: базовый.

Рекомендуемое время выполнения: 4 минуты.

Проверяется умение применять знания основных понятий и методов, используемых при измерении количества информации.

Проверяемые элементы содержания: Теоретические подходы к оценке количества информации. Единицы измерения количества информации. Алфавитный подход к оценке количества информации. Закон аддитивности информации. Формула Хартли. Информация и вероятность. Формула Шеннона.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Понимание основных принципов дискретизации

различных видов информации.

Проверяемые метапредметные результаты: Базовые логические действия (Устанавливать существенный признак или основания для сравнения, классификации и обобщения). Базовые исследовательские действия (Владеть навыками учебно-исследовательской и проектной деятельности, навыками разрешения проблем).

Что нужно знать:

в русском языке 33 буквы: 10 гласных букв (а, у, о, ы, и, э, я, ю, ё, е), 21 согласная буква (б, в, г, д, ж, з, й, к, л, м, н, п, р, с, т, ф, х, ц, ч, ш, щ) и два знака (ь, ъ);

алфавит английского языка по написанию совпадает с латинским алфавитом и состоит из 26 букв;

принципы работы с числами, записанными в позиционных системах счисления;

если слово состоит из L букв, причем есть n_1 вариантов выбора первой буквы, n_2 вариантов выбора второй буквы и т.д., то число возможных слов вычисляется как произведение $N = n_1 \cdot n_2 \cdot \dots \cdot n_L$

если слово состоит из L букв, причем каждая буква может быть выбрана n способами, то число возможных слов вычисляется как $N = n^L$

если в программе L вложенных циклов и внешний цикл выполняется n_1 раз, следующий (вложенный) n_2 раз и т.д., то команды самого внутреннего цикла будут выполняться N раз, где $N = n_1 \cdot n_2 \cdot \dots \cdot n_L$. Если $n_1 = n_2 = \dots = n_L = n$, то $N = n^L$. при увеличении n или L значение N сильно возрастает, что приводит к существенному увеличению времени выполнения программы.

при решении с помощью программы на языке Python удобно использовать функции из модуля `itertools`:

```
combinations – комбинации, например,  
from itertools import combinations  
cmb = list(combinations('ABC', 2))  
print( cmb )  
// Результат: [('A', 'B'), ('A', 'C'), ('B', 'C')]  
permutations – перестановки, например,  
from itertools import permutations  
cmb = list(permutations('ABC'))  
print( cmb )  
// Результат: [('A', 'B', 'C'), ('A', 'C', 'B'),  
// ('B', 'A', 'C'), ('B', 'C', 'A'), ('C', 'A', 'B'),
```

```
// ('C', 'B', 'A')]
```

product – декартово произведение (все возможные слова заданной длины, составленные из данного алфавита),
например:

```
from itertools import product
cmb = list(product('ABC',repeat=2))
print( cmb )
// Результат: [('A', 'A'), ('A', 'B'), ('A', 'C'), ('B', 'A'),
// ('B', 'B'), ('B', 'C'), ('C', 'A'), ('C', 'B'), ('C', 'C')]
```

Пример формулировки задания

Все шестибуквенные слова, составленные из букв Т, Е, О, Р, И, Я, записаны в алфавитном порядке и пронумерованы.

Вот начало списка:

1. EEEEEЕ
2. EEEEEИ
3. EEEEEО
4. EEEEEР
5. EEEEEТ
6. EEEEEЯ

.....

Определите, под каким номером в этом списке стоит первое слово с чётным номером, которое не начинается с букв Е, И или О и при этом содержит в своей записи ровно одну букву Я.

Примечание. Слово – последовательность идущих подряд букв, не обязательно осмысленная.

Решение (Вариант 1) Используем методы комбинаторики и логического рассуждения.

Выделим все условия задачи:

Алфавит: Е, И, О, Р, Т, Я.

Каждое слово шестибуквенное.

Слово должно содержать ровно одну букву «Я».

Слово не должно начинаться с букв «Е», «И», «О».

Нужен номер первого слова, удовлетворяющего данным условиям, при этом номер должен быть чётным.

Шаг 1: Посчитаем количество слов, начинающихся с букв «Е», «И», «О»

Каждое слово состоит из 6 букв, причём первая буква зафиксирована (например, «Е»), а остальные 5 букв могут быть любой из оставшихся 6 букв данного алфавита. Такие слова образуют блоки по $6^5=7776$ штук. Всего таких слов: $3 \times 7776=23328$.

Шаг 2: Остались слова, начинающиеся с букв «Р», «Т», «Я»

После слов, начинающихся с «Е», «И», «О», идут слова, начинающиеся с «Р». Соответственно, первый блок слов, начинающихся с «Р»:

23329 — РЕЕЕЕЕ

23330 — РЕЕЕЕИ

23331 — РЕЕЕЕО

23332 — РЕЕЕЕР

23333 — РЕЕЕЕТ

23334 — РЕЕЕЕЯ

Видим, что слово «РЕЕЕЕЯ» первое, которое удовлетворяет условиям. Оно стоит в списке под номером 23334.

Ответ: 23334

Решение (Вариант 2)

Этот вариант для программирующих учеников.

Пример программы на языке Python, включающей построение отсортированного по алфавиту списка заданных слов и выбора из него слова, удовлетворяющего условиям:

```
from itertools import product
```

```
# Исходные буквы
```

```
letters = ['E', 'I', 'O', 'P', 'T', 'Y']
```

```
# Генерация всех возможных шестибуквенных слов
```

```
words = sorted([''.join(p) for p in product(letters, repeat=6)])
```

```
for _ in range(len(words)):
```

```
    if words[_][0]!="E" and words[_][0]!="I" and words[_][0]!="O":
```

```
if words[_].count("Я")==1:
    print(_+1,words[_])
    break
```

Ответ: 23334

Можно составить разные алгоритмы:

```
from itertools import product
```

```
# Исходные буквы
```

```
letters = ['E', 'И', 'O', 'P', 'T', 'Я']
```

```
# Генерация всех возможных шестибуквенных слов
```

```
words = sorted([".join(p) for p in product(letters, repeat=6)])
```

```
# Функция фильтрации слов
```

```
def is_valid(word):
```

```
    # Проверяем, что слово не начинается с букв 'E', 'И', 'O'
```

```
    if word.startswith(('E', 'И', 'O')):
```

```
        return False
```

```
    # Проверяем, что слово содержит ровно одну букву 'Я'
```

```
    if word.count('Я') != 1:
```

```
        return False
```

```
    return True
```

```
# Просматриваем слова и находим первое подходящее с чётным номером
```

```
for index, word in enumerate(words, start=1):
```

```
    if is_valid(word) and index % 2 == 0:
```

```
        print(f"Первый номер: {index}, Слово: {word}")
```

```
        break
```

Ответ: 23334

Анализ ошибок

В задаче подобного формата школьники нередко допускают типичные ошибки, которые приводят к неверному ответу. Вот наиболее распространённые ошибки и рекомендации, как их избежать:

1. Неправильное понимание порядка слов:

Ошибка: Многие ученики предполагают, что слова идут в алфавитном порядке только по первой букве, игнорируя внутренние различия или ориентируются на заданный список букв в начале условия, а не на их алфавитный порядок.

Совет: Обязательно осознайте, что порядок слов зависит от полного состава букв, и даже небольшая разница в составе букв меняет позицию слова существенно, важно ориентироваться на порядок изменения букв в заданном списке слов.

2. Недооценивание влияния повтора букв:

Ошибка: Учащиеся часто пренебрегают возможностью повторения букв, думая, что каждая буква уникальна.

Совет: Внимательно прочитайте условие задачи: буквы могут повторяться неограниченно много раз, что значительно увеличивает количество возможных комбинаций.

3. Игнорирование ограничения на количество букв «Я»:

Ошибка: Иногда школьники полагают, что требуется ровно одна буква «Я».

Совет: Читайте условия задачи внимательно.

4. Неверный подсчёт позиций:

Ошибка: Некоторые ученики пытаются рассчитать позицию слова вручную, основываясь на интуиции, что неизбежно ведёт к ошибкам.

Совет: Лучше разработать стратегию: сначала оценить количество слов, исключаемых по определенным признакам (например, начинающимся с «Е», «И», «О»), затем просчитать точное положение интересующего слова.

5. Проблемы с чётностью номера:

Ошибка: Частая ошибка – забыть, что нужен чётный номер. Многие находят слово с нечётным номером и делают выводы преждевременно.

Совет: Никогда не забывайте проверять чётность найденного номера, если это указано в условии задачи, опять же, прежде нужно внимательно прочесть условие задачи.

6. Переусложнение задачи:

Ошибка: Нередко школьники усложняют задачу чрезмерно сложными формулами или вычислениями, пытаются

показать глубину знания математики. На деле при подсчете допускаются арифметические ошибки или ошибки в модели расчета.

Совет: Будьте проще. Эта задача решается базовыми методами комбинаторики и здравым смыслом. Нет нужды придумывать сложные конструкции.

Способы избежать таких ошибок:

Самая важная рекомендация – при подготовке школьников акцентировать внимание на разборе условия, внимательном его прочтении. Прежде, чем приступить к решению, нужно выделить все ограничения и требования к ответу. Большинство ошибок происходят из-за невнимания к деталям или поспешных выводов. Стоит включить в учебный материал задания, нацеленные осознанно на работу с условием: «Расставь данные буквы по алфавиту», «Продолжи данный алфавитный порядок слов до 20», «Сколько слов, начинающихся на букву «Е» будет в этом списке?», «Сколько всего слов в списке?», «Под каким номером в полном списке будет стоять слово «ИЕЕЕОЯ»?», «Выдели все требования к искомому слову». Такие приемы позволят учащимся сконцентрироваться на тексте задания, повторить комбинаторику и выбрать правильный способ и алгоритм решения.

Пропедевтически стоит включать такие задания при изучении темы «Равномерные коды» в разделе «Представление информации» в 7 классе, а также, включать задания по типу задания №8 из КИМ ЕГЭ при изучении раздела «Алгоритмизация и программирование» в 9-11 классах, усложняя формулировки условий постепенно.

Задание № 9

Тема: Встроенные функции в электронных таблицах.

Уровень сложности: базовый.

Рекомендуемое время выполнения: 6 минут.

Проверяется умение обрабатывать числовую информацию в электронных таблицах.

Проверяемые элементы содержания: Анализ данных с помощью электронных таблиц. Вычисление суммы, среднего арифметического, наибольшего (наименьшего) значения диапазона. Вычисление коэффициента корреляции двух рядов.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение использовать электронные таблицы для анализа, представления и обработки данных (включая выбор оптимального решения, подбор линии тренда, решение задач прогнозирования); умение использовать табличные (реляционные) базы данных и справочные системы.

Проверяемые метапредметные результаты: Работа с информацией

Что нужно знать:

для вычисления максимального, минимального и среднего арифметического значений диапазона (например, A1:G20) используются соответственно функции

MAX(A1:G20) МАКС(A1:G20)

MIN(A1:G20) МИН(A1:G20)

AVERAGE(A1:G20) СРЗНАЧ(A1:G20)

Слева записаны английские названия, справа – русские (выбор зависит от программы и версии операционной системы).

в списке аргументов этих функций можно указывать несколько диапазонов и адресов ячеек, разделив их точкой с запятой, например:

МАКС(A1:G20;H15;K12:Y90)

МИН(A1:G20;H15;K12:Y90)

СРЗНАЧ(A1:G20;H15;K12:Y90)

все три функции игнорируют (не учитывают) пустые ячейки и ячейки, содержащие нечисловые (например, текстовые) данные;

кроме перечисленных функций могут понадобиться и облегчить выполнение заданий следующие функции: СЧЁТ или СЧЁТЕСЛИ; СУММ или СУММЕСЛИ; НАИБОЛЬШИЙ (НАИМЕНЬШИЙ); ОСТАТ; ЦЕЛ; И; ИЛИ; ЕСЛИ.

Пример формулировки задания

Откройте файл электронной таблицы, содержащей в каждой строке шесть натуральных чисел. Определите наибольший номер строки таблицы, для чисел которой выполнены оба условия:

– в строке есть только одно число, которое повторяется дважды, остальные четыре числа различны;

– повторяющееся число строки больше, чем среднее арифметическое четырёх её неповторяющихся чисел.

В ответе запишите только число.

Пример исходных данных в файле:

	A	B	C	D	E	F	G	H
1	23	16	12	47	89	84		
2	65	61	90	11	95	79		
3	100	59	64	98	81	80		
4	67	34	45	75	29	52		
5	47	91	19	48	31	70		
6	66	80	35	70	35	72		
7	81	72	19	13	59	18		
8	53	98	46	21	88	72		
9	19	99	17	100	39	46		
10	14	50	94	60	33	11		
11	100	19	14	61	74	51		

Решение (Excel)

Подготовительные шаги:

- Скачать файл с табличными данными и открыть его в Excel.
- Пусть таблица находится на листе с именем «Данные», а сами данные размещены в столбцах А-F.

Алгоритм:

Создаём вспомогательные колонки:

Столбец G (номер строки): заполним последовательными номерами строк (используя формулу автозаполнения).

Столбцы Н-М (частоты чисел в строках): найдем частоту повторений числа в строке с помощью формулы:

Для ячейки Н1 введём следующую формулу: =СЧЁТЕСЛИ(\$A1:\$F1;A1)

Затем скопируем эту формулу в ячейки Н1:М1 и протянем на всю таблицу.

В ячейке N1 вводим формулу, проверяющую 1 условие (только одно число повторяется дважды):
=ЕСЛИ(И(СЧЁТЕСЛИ(Н1:М1;2)=2;СЧЁТЕСЛИ(Н1:М1;1)=4);1;0)

Скопируем формулу на весь столбец N.

Аналогично заполним столбец O с помощью формулы =ЕСЛИ(N1=1;СУММЕСЛИ(Н1:М1;"=2";A1:F1)/2;0). Так мы по строкам, удовлетворяющим 1 условию, выделим повторяющееся ровно 2 раза число.

Столбец P заполним с помощью формулы =ЕСЛИ(N1=1;СРЗНАЧЕСЛИ(Н1:М1;"=1";A1:F1);0). Так мы по строкам, удовлетворяющим 1 условию, вычислим среднее арифметическое неповторяющихся чисел.

Теперь остается проверить выполнение второго условия. Столбец Q заполним значениями: =ЕСЛИ(O1>P1;1;0).

В ячейке R1 поместим максимальный номер строки таблицы, для чисел которой выполнены оба условия: =МАКСЕСЛИ(G:G;Q:Q;1). Получаем результат для данных, предоставленных к заданию: 11990.

Ответ: 11990

Анализ ошибок

При решении подобной задачи учащиеся зачастую сталкиваются с рядом типичных трудностей и совершают распространенные ошибки:

1. Невнимательность к условию задачи:

Самая распространённая ошибка связана с непониманием формулировки задачи. Например, ученик может пропустить фразу "есть только одно число, которое повторяется дважды", и посчитать любую строку с повторениями, даже если повторяются разные числа. Или находить не номер максимальной строки, а количество строк с указанными числами.

Совет: читать условие задачи медленно и внимательно, делая акцент на ключевых словах ("одно число", "среднее арифметическое"), выделять (желательно, выписывая) все условия заданные в формулировке.

2. Проблемы с неправильным выполнением функции СРЗНАЧ:

Ученики часто используют функцию СРЗНАЧ для всей строки, включая повторяющееся число, что противоречит условию задачи.

Совет: научиться грамотно выбирать аргументы для функции СРЗНАЧ, исключая повторяющееся число из расчёта.

3. Неграмотное использование вспомогательных столбцов:

Зачастую ученики запутываются в организации вспомогательных столбцов для промежуточных вычислений, что приводит к путанице в формулах.

Совет: организовать рабочий лист аккуратно, создав отдельные столбцы для меток повторяющихся чисел, расчёта средних значений и прочего, желательно выделять столбцы цветами.

4. Недостаточное тестирование примеров:

Без тестирования решения на конкретных простых примерах можно допустить ошибку, даже не подозревая о ней.

Совет: перед сдачей решения проверьте алгоритм на небольшом тестовом файле, проверив несколько строк вручную, убедившись, что все числа в них обрабатываются формулами верно.

5. Паника при работе с большими объёмами данных:

Вид большого объёма данных пугает многих учеников, и они теряют концентрацию, совершая глупые ошибки.

Совет: действовать спокойно и структурированно, работая маленькими частями, разбивая задачу на подзадачи: найдите первое повторяющееся число, затем вычислите среднее, и только потом принимайтесь за сравнение.

Итоги: Большинство ошибок связано с поверхностным прочтением условий задачи, небрежностью при работе с данными и недостаточной подготовкой промежуточных этапов вычислений. Чтобы справиться с подобными задачами эффективно, старайтесь соблюдать спокойствие, планировать шаги заранее и проводить предварительное тестирование решений.

Способы избежать таких ошибок:

Итак, подытожим.

Самая важная рекомендация – при подготовке школьников акцентировать внимание на разборе условия, внимательном его прочтении, 80% правильности решения зависит от проработки и понимания формулировки. Прежде, чем приступать к решению, нужно выделить все ограничения и требования к ответу. Стоит привить у школьников привычку выполнять постановку задачи, отвечая на вопросы «Что дано?», «Что нужно найти?», «Каковы условия поиска? Сколько их?», выписывая это на бумагу.

Далее нужно приучить школьников разбивать задачу на подзадачи, это поможет выделить этапы построения алгоритма.

На каждом этапе решения необходимо моделирование: чаще ошибки здесь возникают при составлении формул: незнание функций, неверный порядок операций, пропуск отдельных условий, неверное составление формулы для проверки конкретного условия... Именно этап моделирования (составления формул для вычисления) является непреодолимой преградой для школьника. При анализе статистических результатов мы видели, что задание № 3 (базового уровня) и даже задание № 18 (повышенного уровня) экзаменуемые выполняют лучше, чем задание № 9. То есть школьники умеют составлять и применять в электронных таблицах основные встроенные функции, но составить и последовательно применить каскад формул для проверки данных на выполнение конкретных заданных условий – не могут. А именно эти этапы: формализация, моделирование, разбиение задачи на подзадачи очень важны в алгоритмизации и программировании. По сути, при выполнении таких заданий, как задание № 9, школьники учатся алгоритмизации и программированию!

Процент выполнения задания крайне низок во всех группах участников ЕГЭ, но задание проверяет базовый уровень подготовки к работе с табличными данными. Больше внимания работе с электронными таблицами уделяется в

рабочей программе углубленного курса обучения информатике в школе, однако, эта тема представлена и в программе базового курса обучения информатике как в основной, так и в старшей школе. Поэтому учить составлять формулы по выделенным в задании условиям необходимо начинать в 9 классе в разделе «Моделирование как метод познания» (название из федеральной рабочей программы) и вернуться к отработке перечисленных выше навыков в теме в 11 классе в теме «Анализ данных с помощью электронных таблиц».

Важно применять задания на работу с формулировками, условиями, учить разбивать задачи на подзадачи, моделировать, тестировать промежуточные и итоговый результат. Это важные начальные этапы при разработке алгоритмов решения задач на ЭВМ любыми средствами.

Задание № 17

Тема: Перебор последовательности целых чисел. Проверка делимости.

Уровень сложности: повышенный.

Рекомендуемое время выполнения: 14 минут.

Умение составить алгоритм обработки числовой последовательности и записать его в виде простой программы (10–15 строк) на языке программирования.

Проверяемые элементы содержания: Массивы и последовательности чисел. Вычисление обобщённых характеристик элементов массива или числовой последовательности (суммы, произведения, среднего арифметического, минимального и максимального элементов, количества элементов, удовлетворяющих заданному условию). Линейный поиск заданного значения в массиве. Алгоритмы работы с элементами массива с однократным просмотром массива. Сортировка одномерного массива. Простые методы сортировки (метод пузырька, метод выбора, сортировка вставками). Сортировка слиянием. Быстрая сортировка массива (алгоритм QuickSort). Двоичный поиск в отсортированном массиве.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение реализовывать на выбранном для изучения языке программирования высокого уровня (Паскаль, Python, Java, C++, C#) типовые алгоритмы обработки чисел, числовых последовательностей и массивов: представление числа в виде набора простых сомножителей; нахождение максимальной (минимальной) цифры натурального числа, записанного в системе счисления с основанием, не превышающим 10; вычисление обобщённых характеристик элементов массива или числовой последовательности (суммы, произведения среднего арифметического, минимального и максимального элементов, количества элементов, удовле-

творяющих заданному условию); сортировку элементов массива; умение использовать в программах данные различных типов с учётом ограничений на диапазон их возможных значений, применять при решении задач структуры данных (списки, словари, стеки, очереди, деревья); применять стандартные и собственные подпрограммы для обработки числовых данных и символьных строк; использовать при разработке программ библиотеки подпрограмм; умение использовать средства отладки программ в среде программирования.

Проверяемые метапредметные результаты: базовые исследовательские действия, самоорганизация, самоконтроль.

Что нужно знать:

как организовать простой перебор данных при проверке условий без оптимизации вычислений;

как обрабатывать данные с помощью электронных таблиц или собственной программы, как правило, написать правильную программу значительно проще;

как реализовать на языке программирования общую структуру цикла перебора для подсчета значений, удовлетворяющих заданным условиям.

Пример общей структуры цикла (Python):

```
count = 0
for n in range(a, b+1):
    if условие выполнено:
        count += 1
print( count )
```

Pascal:

```
count := 0;
for n:=a to b do
    if условие выполнено then
        count := count + 1;
writeln(count);
```

C++:

```
int count = 0;
```

```
for(int n = a; n <= b; n++)  
    if( условие выполнено )  
        count += 1;  
std::cout << count;
```

как проверить делимость числа n на число d с помощью операции взятия остатка от деления n на d : если остаток равен 0, число n делится на d нацело

проверка на языке Python выглядит так:

```
if n % d == 0:  
    print("Делится")  
else: print("Не делится")
```

тоже самое на Pascal

```
if n mod d = 0 then  
    writeln('Делится')  
else writeln('Не делится')
```

то же самое на C++

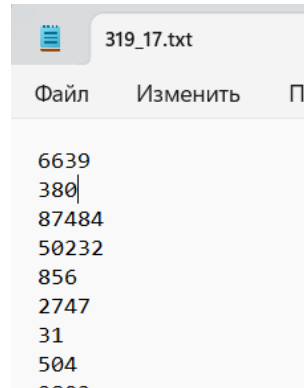
```
if( n % d == 0 )  
    std::cout << "Делится";  
else std::cout << "Не делится";
```

Пример формулировки задания

В файле содержится последовательность натуральных чисел. Её элементы могут принимать целые значения от 1 до 100 000 включительно.

Определите количество троек элементов последовательности, в которых только одно из чисел является двузначным, а сумма элементов тройки не больше максимального элемента последовательности, оканчивающегося на 13. В ответе запишите количество найденных троек чисел, затем максимальную из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

Пример исходных данных в файле:



Решение (Python)

Алгоритм:

Прочитаем файл с числовой последовательностью.

Найдём максимальное число в последовательности, оканчивающееся на 13, и сохраним его в переменную `max_value`.

Переберём все тройки подряд идущих элементов и проверим условия:

- ровно одно число в тройке должно быть двузначным;

- сумма элементов тройки должна быть не больше, чем найденное максимальное число (`max_value`).

Подсчитаем количество подходящих троек и найдём максимальную сумму среди подходящих троек.

Выведем количество подходящих троек и максимальную сумму.

Реализация (Python):

```
def find_triplets(filename):
    with open(filename, 'r') as file:
        numbers = list(map(int, file.read().split()))

    # Находим максимальное число, оканчивающееся на 13
    max_value = max(num for num in numbers if str(num).endswith('13'))

    # можно условие str(num).endswith('13') заменить str(num).endswith('13')
```

```

triplets_count = 0
max_sum = float('-inf')

# Перебираем все тройки подряд идущих элементов
for i in range(len(numbers) - 2):
    current_triplet = numbers[i:i+3]
    two_digit_numbers = sum(1 for num in current_triplet if 10 <= num < 100)
    triplet_sum = sum(current_triplet)

    # Проверяем условия:
    # 1. Ровно одно число двузначное
    # 2. Сумма элементов тройки не больше максимального числа, оканчивающегося на 13
    if two_digit_numbers == 1 and triplet_sum <= max_value:
        triplets_count += 1
        max_sum = max(max_sum, triplet_sum)

return triplets_count, max_sum

filename = input("Введите название файла: ")
result = find_triplets(filename)
print(*result)

```

Можно выполнить задание средствами электронных таблиц. Алгоритм работы тот же, расчеты выполняются с помощью формул, по примеру выполнения задания № 9.

Анализ ошибок

Типичные ошибки при выполнении данного задания аналогичны ошибкам, описанным в задании № 9, кроме этого добавились ошибки, связанные с владением языком программирования (если решение выполнено с помощью

программирования).

1. Неправильное определение максимальной суммы (ошибка в понимании условия или невнимательном прочтении):

Школьники могут сравнивать каждую тройку с каждым числом, оканчивающимся на 13, вместо того, чтобы предварительно найти одно максимальное число, оканчивающееся на 13, и сравнить с ним.

2. Некорректная обработка условий для элементов тройки:

Школьники могут включать в рассмотрение пары, где есть больше одного двузначного числа, а не ровно одно.

3. Ошибка в понимании понятия "подряд идущие элементы":

Ученики могут рассматривать не тройки подряд идущих элементов, а произвольные наборы из трёх элементов, нарушая условие задачи.

Все эти ошибки связаны с невнимательностью при работе с формулировкой задания.

Следующая группа ошибок связана с построением алгоритма и написанием программы.

4. Ошибки в механизме перебора троек:

При переборе элементов ученики допускают выход за пределы массива или неправильное формирование текущих троек.

5. Использование неподходящего способа хранения данных:

Если файл большой, школьник может столкнуться с проблемами памяти, читая все данные разом. Оптимальной стратегией будет обрабатывать данные построчно или небольшими порциями.

6. Несоблюдение ограничений на числа:

Некоторые школьники могут пренебречь ограничением на диапазон чисел (от 1 до 100 000), что приведет к проблемам при чтении и обработке файлов с большим количеством данных.

7. Трудности при работе с файлами:

Могут возникать ошибки чтения из файла, вызванные отсутствием необходимых навыков работы с операторами ввода-вывода. На языке Python часто возникает ошибка при считывании данных построчно и отбрасывании «пробельных» символов типа «\n». Здесь важно применять не вырезку (`string[:-1]`), а метод `string.strip([<chars>])` и т.п.

8. Логическая ошибка при вычислении суммы и сравнении с пределом:

Легко спутать оператор сравнения "`<=`" с "`>=`" или совершить ошибку при суммировании элементов тройки.

Способы избежать таких ошибок:

При изучении раздела «Алгоритмизация и программирование» как в базовом курсе информатики, так на

углубленном уровне обучения раздел «Алгоритмизация и программирование» является одним из основных, начиная с 8 класса.

Кроме перечисленных рекомендаций к заданию № 9, которые полностью подходят и к заданию № 17, учителям при обучению основам программирования необходимо концентрировать внимание на работу с файлами (многие ученики не владеют такими знаниями), на разбиение задачи на подзадачи, на тестирование решений. Необходимо обращать внимание на обработку крайних случаев, таких как пустые файлы или короткие последовательности.

Таким образом, данная задача требует внимания к деталям и хорошего владения основами работы с файлами и циклическими конструкциями в Python

Задание № 24

Тема: Обработка символьных строк.

Уровень сложности: высокий.

Рекомендуемое время выполнения: 18 минут.

Умение создавать собственные программы (10–20 строк) для обработки символьной информации.

Проверяемые элементы содержания: Обработка символьных данных. Встроенные функции языка программирования для обработки символьных строк. Алгоритмы обработки символьных строк: подсчёт количества появлений символа в строке, разбиение строки на слова по пробельным символам, поиск подстроки внутри данной строки, замена найденной подстроки на другую строку. Генерация всех слов в некотором алфавите, удовлетворяющих заданным ограничениям. Преобразование числа в символьную строку и обратно.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Владение универсальным языком программирования высокого уровня (Паскаль, Python, Java, C++, C#), представлениями о базовых типах данных и структурах данных; умение использовать основные управляющие конструкции; умение осуществлять анализ предложенной программы: определять результаты работы программы при заданных исходных данных; определять, при каких исходных данных возможно получение указанных результатов; выявлять данные, которые могут привести к ошибке в работе программы; формулировать предложения по улучшению программного кода.

Проверяемые метапредметные результаты: базовые исследовательские действия, самоорганизация, самоконтроль.

Что нужно знать:

сначала нужно прочитать строку из файла; эта задача в разных языках программирования решается по-разному в языке Python удобнее всего использовать такую конструкцию:

```
with open("k7.txt", "r") as F:  
s = F.readline()
```

конструкция with-as – это контекстный менеджер, в данном случае он открывает указанный файл в режиме чтения (второй аргумент «r» при вызове функции open), записывает ссылку на него в файловую переменную F, выполняет тело блока (читает первую строку файла в переменную s) и закрывает (освобождает) файл

в языке PascalABC.NET можно выполнить перенаправление потока ввода:

```
assign( input, 'k7.txt' );  
readln(s);
```

программа будет «думать», что читает данные, введённые с клавиатуры (с консоли), а на самом деле эти данные будут прочитаны из файла k7.txt

в языке FreePascal также можно выполнить перенаправление потока ввода, но нужно дополнительно открывать входной поток:

```
assign( input, 'k7.txt' );  
reset( input );  
readln(s);
```

при работе в среде FreePascal нужно убедиться, что в параметрах компилятора включена поддержка длинных символьных строк; на всякий случай стоит добавить в первой строке программы директиву

```
{ $H+ }
```

– Среда PascalABC НЕ ПОДДЕРЖИВАЕТ работу с длинными символьными строками, поэтому для решения задачи использовать версию PascalABC.NET.

В языке C++ используем потоки:

```
#include <fstream>  
#include <string>  
using namespace std;  
int main()  
{  
    ifstream F(«k7.txt»);
```

```

string s;
getline( F, s );
...
}

```

Для выполнения заданий требуется уметь определять длину строки, обращаться к символу по индексу, копировать и вставлять строки и подстроки.

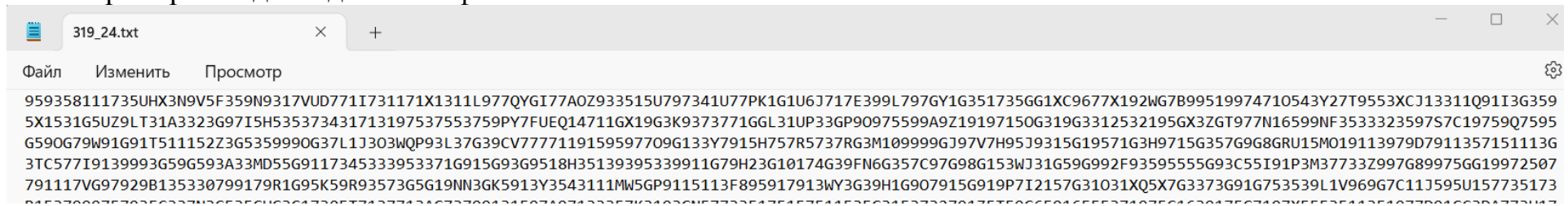
Пример формулировки задания

Текстовый файл состоит из десятичных цифр и заглавных букв латинского алфавита. Определите в прилагаемом файле последовательность из максимального количества идущих подряд символов, среди которых ровно 45 нечётных цифр и при этом начинающуюся с буквы G, не содержащую других букв G, кроме первой.

В ответе запишите число – количество символов в найденной последовательности.

Для выполнения этого задания следует написать программу.

Пример исходных данных в файле:



Решение (Python)

Приведем пример решения с использованием левой L и правой границы подстроки R и проверки вырезки между позициями L и R.

```
s = open('319_24.txt').readline()
```

```
COUNTS = 45
```

```
def valid( p ):
```

```
    return p[0]=="G" and p.count("G")==1 and sum(1 for x in p if x in "13579")==COUNTS
```

```
def cantBeValid( p ):
```

```
return p[0]!="G" or p.count("G")!=1 or sum(1 for x in p if x in "13579")>COUNTS
```

```
N = len(s)
maxLen = 0
for L in range(N):
    for R in range(L+maxLen+1,N+1):
        sub = s[L:R]
        if valid( sub ):
            maxLen = R - L
            sMax = sub
        elif cantBeValid( sub ):
            break
```

```
print( maxLen, sMax )
```

В примере печатается и сама строка (для проверки), в качестве ответа нужно вывести значение длины – maxLen.

Анализ ошибок

Типичные ошибки учеников:

1. Неправильная работа с условием "ровно одна буква G":

Ошибка: Учащиеся иногда пропускают условие наличия единственной буквы G. Например, могут учитывать последовательности с несколькими буквами G или вовсе без неё.

2. Неверная оценка количества нечётных цифр:

Ошибка: Нередко ученик учитывает все цифры вообще, а не только нечётные.

3. Недостаточный контроль длины окна:

В некоторых случаях школьники неверно определяют границы рассматриваемого окна. Они начинают искать решение в слишком узких границах или наоборот, включают лишние элементы.

4. Отсутствие раннего завершения при невозможных вариантах:

Ошибка: Ученики продолжают обработку окон, несмотря на очевидную невозможность удовлетворить требования (например, превышение лимита нечётных цифр), тем самым замедляют работу программы и не дожидаются ответа на экзамене, считая, что программа не работает.

5. Неправильное обновление результата:

Ошибка: Возможно неправильное сохранение лучшей найденной подпоследовательности. Она может заменяться короткой версией при первом попадании подходящей строки.

6. Работа с индексами вне диапазона:

Ошибка: Возможны ситуации, когда правая граница выходит за пределы строки, вызывая ошибку `IndexError`.

7. Забывание об инициализации переменных:

Ошибка: Переменные, такие как счётчики и максимальный результат, могут остаться неопределёнными или иметь некорректные начальные значения.

Способы избежать таких ошибок:

Задание 24 проверяет не базовые, а продвинутое владение программированием и сильна только хорошо подготовленным школьникам, изучавшим углубленный курс информатики. Поэтому, тренировать решение таких задач в классе стоит с сильными учениками. Важно при освоении навыков решения таких задач показать ученикам более эффективные способы разработки алгоритма, иначе программа может выполняться долго: метод динамического программирования (применение левых и правых границ); в некоторых случаях хорошо работает разбиение строки (в Python это метод `split()`). Важно учить выделять все условия в формулировке задания и тестировать промежуточные результаты.

Задание № 25

Тема: Обработка целых чисел. Проверка делимости.

Уровень сложности: высокий.

Рекомендуемое время выполнения: 20 минут.

Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации.

Проверяемые элементы содержания: Алгоритмы обработки натуральных чисел, записанных в позиционных системах счисления:

разбиение записи числа на отдельные цифры, нахождение суммы и произведения цифр, нахождение максимальной (минимальной) цифры. Представление числа в виде набора простых сомножителей. Алгоритм быстрого возведения в степень. Поиск простых чисел в заданном диапазоне с помощью алгоритма «решето Эратосфена».

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение реализовывать на выбранном для изуче-

ния языке программирования высокого уровня (Паскаль, Python, Java, C++, C#) типовые алгоритмы обработки чисел, числовых последовательностей и массивов: представление числа в виде набора простых сомножителей; нахождение максимальной (минимальной) цифры натурального числа, записанного в системе счисления с основанием, не превышающим 10; вычисление обобщённых характеристик элементов массива или числовой последовательности (суммы, произведения сред него арифметического, минимального и максимального элементов, количества элементов, удовлетворяющих заданному условию); сортировку элементов массива; умение использовать в программах данные различных типов с учётом ограничений на диапазон их возможных значений, применять при решении задач структуры данных (списки, словари, стеки, очереди, деревья); применять стандартные и собственные подпрограммы для обработки числовых данных и символьных строк; использовать при разработке программ библиотеки подпрограмм; умение использовать средства отладки программ в среде программирования

Проверяемые метапредметные результаты: базовые исследовательские действия, самоорганизация, самоконтроль.

Что нужно знать:

задачи этого типа можно решать в электронных таблицах или с помощью собственной программы, как правило, написать правильную программу значительно проще;

пусть необходимо перебрать все целые числа на отрезке $[a; b]$ и подсчитать, для скольких из них выполняется некоторое условие, общая структура цикла перебора записывается так (Python):

```
count = 0
for n in range(a, b+1):
    if условие выполнено:
        count += 1
print( count )
```

Pascal:

```
count := 0;
for n:=a to b do
    if условие выполнено then
        count := count + 1;
writeln(count);
```

C++:

```
int count = 0;
for(int n = a; n <= b; n++)
    if( условие выполнено )
        count += 1;
std::cout << count;
```

- проверку условия удобно оформить в виде функции, возвращающей логическое значение (True/False), но можно этого и не делать;

- проверить делимость числа n на число d можно с помощью операции взятия остатка от деления n на x : если остаток равен 0, число n делится на x нацело;

- проверка делимости на языке Python выглядит так:

```
if n % d == 0:
    print("Делится")
else: print("Не делится")
```

- тоже самое на Pascal

```
if n mod d = 0 then
    writeln('Делится')
else writeln('Не делится')
```

- то же самое на C++

```
if( n % d == 0 )
    std::cout << "Делится";
else std::cout << "Не делится";
```

Количество делителей

для определения числа делителей натурального числа n можно использовать цикл, в котором перебираются все возможные делители d от 1 до n , при обнаружении делителя увеличивается счётчик делителей:

```
count = 0
```

```
for d in range(1, n+1):
    if n % d == 0:
        count += 1
print( count ) # вывести количество делителей
```

тоже самое на Pascal

```
count := 0;
for d:=1 to n do
    if n mod d = 0 then
        count := count + 1;
writeln( count );
```

• то же самое на C++

```
int count = 0;
for(int d = 1; d <= n; d++)
    if( n % d == 0 ) count ++;
std::cout << count; // вывести количество делителей
```

если требуется определить не только количество делителей, но и сами делители, нужно сохранять их в массиве; в языке Python удобно использовать динамический массив: сначала он пуст, а при обнаружении очередного делителя этот делитель добавляется в массив:

```
divs = []
for d in range(1,n+1): # перебор всех возможных делителей
    if n % d == 0:     # если нашли делитель d
        divs.append(d) # то добавили его в массив
```

в языках Pascal и C++ проще обойтись без динамического массива; здесь есть два варианта:

- 1) выделить массив достаточного размера для хранения всех делителей; например, количество делителей числа n явно не превышает n ;
- 2) хранить только нужное количество делителей, например, если нас интересуют числа, имеющие 4 делите-

ля, достаточно выделить массив из 4-х элементов, а остальные делители в массив не записывать;

перебор делителей можно оптимизировать, учитывая, что наименьший из пары делителей, таких что $a \times b = n$, не превышает квадратного корня из n ; нужно только аккуратно обработать случай, когда число n представляет собой квадрат другого целого числа;

отметим, что для чисел, которые предлагаются в вариантах заданий, такая оптимизация не обязательна; более того, усложнение программы может привести к дополнительным ошибкам...

Простые числа

простое число n делится только на 1 и само на себя, причём единица не считается простым числом; таким образом, любое простое число имеет только два делителя

для определения простоты числа можно считать общее количество его делителей; если их ровно два, то число простое, если не два – не простое:

```
nDel = 0      # количество делителей числа
for d in range(1, n+1): # все возможные делители
    if n % d == 0:
        nDel += 1      # нашли ещё делитель
if nDel == 2:
    print( "Число простое" )
else:
    print( "Число составное" )
```

работу программы можно ускорить: если уже найдено больше двух делителей, то число не простое и можно досрочно закончит работу цикла с помощью оператора break:

```
nDel = 0      # количество делителей числа
for d in range(1, n+1): # все возможные делители
    if n % d == 0:
        nDel += 1      # нашли ещё делитель
        if nDel > 2:    # уже не простое число
            break      # досрочный выход из цикла
if nDel == 2:
    print( "Число простое" )
```

```
else:  
    print( "Число составное" )
```

другой вариант – считать количество делителей числа на отрезке $[2; n-1]$; как только хотя бы один такой делитель будет найден, можно завершить цикл, потому что число явно не простое:

```
nDel = 0          # количество делителей на отрезке [2; n-1]  
for d in range(2, n):  
    if n % d == 0:  
        nDel += 1    # нашли делитель  
        break       # досрочный выход из цикла  
if nDel == 0:  
    print( "Число простое" )  
else:  
    print( "Число составное" )
```

можно сделать то же самое с помощью логической переменной:

```
prime = True      # сначала считаем число простым  
for d in range(2, n):  
    if n % d == 0:  
        prime = False    # уже не простое  
        break           # досрочный выход из цикла  
if prime:  
    print( "Число простое" )  
else:  
    print( "Число составное" )
```

тоже самое на Pascal

```
prime := True;      { сначала считаем число простым }  
for d:=2 to n-1 do
```

```

if n mod d = 0 then begin
  prime := False; { уже не простое }
  break          { досрочный выход из цикла }
end;
if prime then
  writeln( 'Число простое' )
else
  writeln( 'Число составное' );

```

```

то же самое на C++
bool prime = true;          // сначала считаем число простым
for( int d = 2; d <= n-1; d++ )
  if( n % d == 0 ) {
    prime = false; // уже не простое
    break;        // досрочный выход из цикла
  }
if( count == 2 )
  std::cout << "Число простое";
else
  std::cout << "Число составное";

```

в этом задании обычно предлагаются большие числа, поэтому проверка делимости на все числа от 2 до $n-1$ выполняется очень долго, и на устаревших компьютерах время работы приведённого выше алгоритма может быть слишком велико

программу можно оптимизировать, если вспомнить, что наименьший из пары делителей, таких что $a \times b = n$, не превышает квадратного корня из n ; поэтому можно закончить перебор значением, округлив его до ближайшего целого числа; если на отрезке $[2; \sqrt{n}]$ не найден ни один делитель, их нет и на отрезке $[\sqrt{n} + 1, n - 1]$

следовательно, можно существенно ускорить перебор, изменив конечное значение переменной цикла:
 for d in range(2, round(sqrt(n))+1):

на языке Pascal:

```
for d:=2 to round(sqrt(n)) do
```

на языке C++:

```
for( int d = 2; d <= round(sqrt(n)); d++ )
```

Пример формулировки задания

Напишите программу, которая перебирает целые числа, большие 1481011, в порядке возрастания и ищет среди них представленные в произведения ровно двух простых множителей, не обязательно различных, каждый из которых содержит в своей записи ровно одну цифру 7. В ответе в первом столбце таблицы запишите первые 5 найденных чисел в порядке возрастания, а во втором столбце – для каждого из чисел наибольший из соответствующих им найденных множителей. Количество строк в таблице для ответа избыточно.

Решение (Python)

```
import math
```

```
# Проверка простоты числа
```

```
def is_prime(n):
```

```
    if n < 2:
```

```
        return False
```

```
    if n == 2:
```

```
        return True
```

```
    if n % 2 == 0:
```

```
        return False
```

```
    sqrt_n = int(math.isqrt(n)) + 1
```

```
    for i in range(3, sqrt_n, 2):
```

```
        if n % i == 0:
```

```
            return False
```

```
    return True
```

```
# Проверка, содержит ли число ровно одну цифру '7'
```

```

def one_digit_7(n):
    return str(n).count('7') == 1

# Главная функция для поиска нужных чисел
def search_numbers(limit=5):
    result = []
    number = 1481012 # начинаем с числа, следующего за указанным пределом

    while len(result) < limit:
        # Попробуем факторизацию числа
        root = int(math.isqrt(number)) + 1
        found = False

        for d in range(2, root):
            if number % d == 0:
                q = number // d

                # Оба множителя должны быть простыми и содержать ровно одну цифру '7'
                if is_prime(d) and is_prime(q) and one_digit_7(d) and one_digit_7(q):
                    result.append((number, max(d, q)))
                    found = True
                    break
            number += 1

    return result

# Выполняем поиск и выводим результат
result = search_numbers()
for pair in result:

```

```
print(pair[0], "\t", pair[1])
```

Ключевые части программы:

1. Факторизация: Мы пытаемся разделить число на два множителя. Один из них выбирается в диапазоне от 2 до корня из числа, второй множитель автоматически получается делением числа на выбранный делитель.
2. Проверка условий: Проверяется, что оба множителя простые и содержат ровно одну цифру 7.
3. Результат: Программа останавливается, как только найдено 5 подходящих чисел, и выводит результат в нужном формате.

Анализ ошибок

Типичные ошибки учеников:

1. Недопонимание условия задачи:

Ошибочное понимание требований: некоторые ученики считают, что число должно быть представлено произведением любого количества простых множителей, хотя в условии строго оговорено, что множителей должно быть ровно два.

Неточность в понимании термина "простое число": некоторые ошибочно полагают, что любое число делится на единицу и само себя, но не осознают, что единица не считается простым числом.

2. Проблемы с обработкой случаев:

Использование одного критерия: многие решают задачу, полагая, что нужно просто находить любые простые множители, а потом вручную подбирать те, что содержат цифру 7, забывая, что оба множителя должны содержать ровно одну цифру 7.

3. Логические ошибки:

Искажённое представление о составе числа: некоторые пытаются использовать формулу разложения на множители (включая степени), забывая, что задача требует именно произведения двух простых чисел.

Игнорирование порядка: многие ученики ищут пару множителей и не задумываются о порядке следования множителей, что увеличивает вероятность пропуска правильного варианта.

4. Ошибки программирования:

Программирование процесса факторизации: ошибка возникает при неправильном определении способа проверки деления числа на потенциальные множители.

Выбор подходящего метода поиска простых чисел: использование неэффективных методов поиска простых

чисел замедляет выполнение программы, что усложняет процесс тестирования больших чисел.

Работа с большим количеством данных: некоторые выбирают неэффективные способы хранения промежуточных результатов, что сильно замедляет скорость выполнения программы.

5. Технические аспекты:

Испытание большого объема чисел: многие студенты забывают установить разумные верхние границы для поиска чисел, что затрудняет тестирование больших наборов данных.

Преждевременное завершение программы: ошибки могут возникать, если цикл завершён преждевременно или выполняется дольше необходимого срока.

Способы избежать таких ошибок:

Задание 25 так же, как и задание 24 проверяет не базовые, а продвинутые навыки владения программированием и сильна только хорошо подготовленным школьникам, изучавшим углубленный курс информатики. Поэтому, тренировать решение таких задач в классе стоит с сильными учениками. Однако, поиск делителей числа и нахождение простых чисел, это базовые алгоритмы, которые можно разбирать при изучении циклов, и овладение ими можно реализовать со школьниками в начале изучения основ программирования.

Обычно задание 25 выполняло большое количество участников экзамена (до 35% в разные годы), но в 2025 году крайне низкий результат выполнения – 9,43%.

Важно при освоении навыков решения таких задач отрабатывать с учениками:

эффективные приемы поиска делителей и простых чисел;

тщательное тестирование алгоритма на небольших примерах;

эффективные методы факторизации и поиска простых чисел, чтобы минимизировать временные затраты;

организацию решения таким образом, чтобы легко было проследить ход рассуждений и выявить возможные ошибки.

Задание № 26

Тема: Обработка массива целых чисел из файла. Сортировка.

Уровень сложности: высокий.

Рекомендуемое время выполнения: 35 минут.

Умение обрабатывать целочисленную информацию с использованием сортировки.

Проверяемые элементы содержания: Массивы и последовательности чисел. Вычисление обобщённых характе-

ристик элементов массива или числовой последовательности (суммы, произведения, среднего арифметического, минимального и максимального элементов, количества элементов, удовлетворяющих заданному условию). Лине́йный поиск заданного значения в массиве. Алгоритмы работы с элементами массива с однократным просмотром массива. Сортировка одномерного массива. Простые методы сортировки (метод пузырька, метод выбора, сортировка вставками). Сортировка слиянием. Быстрая сортировка массива (алгоритм QuickSort). Двоичный поиск в отсортированном массиве.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение реализовывать на выбранном для изучения языке программирования высокого уровня (Паскаль, Python, Java, C++, C#) типовые алгоритмы обработки чисел, числовых последовательностей и массивов: представление числа в виде набора простых сомножителей; нахождение максимальной (минимальной) цифры натурального числа, записанного в системе счисления с основанием, не превышающим 10; вычисление обобщённых характеристик элементов массива или числовой последовательности (суммы, произведения среднего арифметического, минимального и максимального элементов, количества элементов, удовлетворяющих заданному условию); сортировку элементов массива; умение использовать в программах данные различных типов с учётом ограничений на диапазон их возможных значений, применять при решении задач структуры данных (списки, словари, стеки, очереди, деревья); применять стандартные и собственные подпрограммы для обработки числовых данных и символьных строк; использовать при разработке программ библиотеки подпрограмм; умение использовать средства отладки программ в среде программирования.

Проверяемые метапредметные результаты: базовые исследовательские действия, самоорганизация, самоконтроль.

Что нужно знать (К.Ю. Поляков):

Чтение данных из файла

в языке Python для чтения данных удобно использовать менеджер контекста (**with ... as**), который открывает файл и закрывает его; например, код

```
with open("26.txt") as Fin: # программа и файл в одной папке
```

```
... # какие-то операции с файлом
```

```
# при завершении работы менеджера контекста
```

```
# файл автоматически закрывается
```

равносилен такому

```
Fin = open("26.txt") # открытие файла
```

```
... # какие-то операции с файлом
```

```
Fin.close() # закрытие файла
```

– если в текущей строке файла находится целое число, то прочитать его в переменную **x** можно так:

```
x = int( Fin.readline() )
```

если в строке записаны два числа, после чтения (**Fin.readline()**) строку нужно разбить на отдельные части по пробелам между числами (каждая часть – символьная запись числа) и затем каждую часть преобразовать в целое число; например, чтение двух чисел:

```
s = Fin.readline()
```

```
symData = s.split()
```

```
x, y = map( int, symData )
```

или в компактной форме

```
x, y = map( int, Fin.readline().split() )
```

в языке PascalABC.NET для чтения данных проще всего просто перенаправить входной поток на файл:

```
Assign( input, '26.txt' );
```

после этого можно использовать операторы **read** и **readln**, так же, как при вводе с клавиатуры в языке C++ можно читать данные с помощью входного потока (**fstream**):

```
#include <fstream>
```

```
...
```

```
ifstream Fin("26.txt");
```

```
Fin >> x;
```

```
Fin >> y >> z;
```

Хранение массива данных

в языке Python для хранения массива данных используется список; следующая программа показывает чтение массива данных размера **N** в список **data** из файла «26.txt» (данные записаны в столбик, по одному числу в строке):

```
data = [0]*N  
with open("26.txt") as Fin:  
  for i in range(N):  
    data[i] = int( Fin.readline() )
```

или с помощью генератора списка

```
with open("26.txt") as Fin:  
  data = [ int( Fin.readline() )  
    for i in range(N) ]
```

в языке PascalABC.NET используем динамический массив; когда станет известен его размер, выделяем место в памяти и читаем из входного потока:

```
var data: array of integer;  
SetLength( data, N );  
for var i:=0 to N-1 do  
  read( data[i] );
```

в языке C++ аналогично используется коллекция **vector**:

```
#include <vector>
```

```
...
```

```
vector <int> data(N);  
for( int i = 0; i < N; i++ )  
  Fin >> data[i];
```

Сортировка массива

Для сортировки имеет смысл использовать встроенные функции языков программирования. Категорически НЕ рекомендуется писать собственные реализации алгоритмов сортировки.

В языке Python для сортировки массива (списка) «на месте» вызывается метод **sort**:
data.sort()

при этом числа сортируются по возрастанию. Для сортировки по убыванию в вызов метода добавляем именованный аргумент **reverse** со значением **True**:

```
data.sort( reverse = True )
```

Для сортировки по другому критерию (например, по последней цифре числа) добавляют именованный аргумент **key**, который указывает на функцию, вычисляющую нужное значение, например:

```
def lastDigit( n ):  
    return n % 10  
... # заполнение массива data  
data.sort( key = lastDigit )
```

Простую функцию можно не оформлять как отдельную подпрограмму, а записать как неименованную функцию (лямбда-функцию) :

```
data.sort( key = lambda x: x % 10 )
```

Иногда данные в массиве **data** представляют собой пары или тройки чисел, объединённые в кортежи. В этом случае при стандартной сортировке сначала сравниваются первые элементы кортежей, если они равны – вторые и т.д. Чтобы задать свой порядок сортировки, нужно использовать аргумент **key** с обычной функцией или лямбда-функцией. Например,

```
data.sort( key = lambda x: (-x[1], x[0]%10) )
```

В этом примере происходит сортировка по убыванию (знак «минус») второго числа в кортеже, **x[1]**, а если вторые элементы равны - по возрастанию последней цифры первого элемента кортежа, **x[0]**.

Если нужно создать новый массив, не изменяя исходные данные, используется функция **sorted**. Её первый аргумент – массив, а остальные совпадают с аргументами метода **sort**. Например,

```
data1 = sorted( data, key = lambda x: (-x[1], x[0]%10) )
```

(Е. Джобс) В языке Python возможна сортировка строк двумерного массива. При этом сначала выполняется сортировка по первому элементу в каждой строке, потом – по второму и т.д. Например, что при сортировке двумерного (точно работает) списка, сортировка идет по значениям второй размерности слева направо. Например,

результатом вызова

```
sorted([ [3,5], [1,6], [2,8], [3,4], [5,8] ])
```

будет

```
[ [1,6], [2,8], [3,4], [3, 5], [5,8] ]
```

В языке PascalABC.NET для динамических массивов используется метод **Sort**:

```
data.Sort;
```

По умолчанию сортировка выполняется в порядке возрастания.

Для нестандартной сортировки лучше использовать метод **OrderBy**, в качестве аргумента можно указать лямбда-функцию. При этом строится новый массив. Вот пример с сортировкой по возрастанию последней цифры числа:

```
var data1 := data.OrderBy( x->x mod 10).ToArray;
```

Если нужна сортировка по убыванию, вместо **OrderBy** применяется метод **OrderByDescending**.

в языке C++ для сортировки коллекции **vector** вызывается процедура **sort** (сортировка «на месте»):

```
#include <vector>
```

```
...
```

```
vector <int> data(N);
```

```
...
```

```
sort( data.begin(), data.end() );
```

Для сортировки по убыванию третьим аргументом указывается функция **greater<int>**:

```
sort( data.begin(), data.end(), greater<int>() );
```

Нестандартную формировку можно выполнить с помощью собственной функции сравнения. Вот, например, сортировка вектора по возрастанию последней цифры:

```
bool cmpLastDigit( int i1, int i2)
```

```
{
```

```

    return (i1 % 10 < i2 % 10);
}
...
sort( data.begin(), data.end(), cmpLastDigit );
Можно использовать и аналогичную лямбда-функцию:
sort( data.begin(), data.end(),
    []( int i1, int i2) { return (i1 % 10 < i2 % 10); } );

```

Пример формулировки задания

Входной файл содержит сведения о заявках на проведение мероприятий в конференц-зале. В каждой заявке указаны время начала и время окончания мероприятия (в минутах от начала суток). Если время начала одного мероприятия меньше времени окончания другого, то провести можно только одно из них. Если время окончания одного мероприятия совпадает со временем начала другого, то провести можно оба. Определите максимальное количество мероприятий, которые можно провести в конференц-зале, и самое позднее время окончания последнего мероприятия в этом случае.

Входные данные

В первой строке входного файла находится натуральное число N ($N \leq 1000$) – количество заявок на проведение мероприятий. Следующие N строк содержат пары чисел, обозначающих время начала и время окончания мероприятий (в минутах от начала суток). Каждое из чисел натуральное, не превосходящее 1440.

Запишите в ответе два числа: максимальное количество мероприятий и самое позднее время окончания последнего мероприятия (в минутах от начала суток).

Типовой пример организации данных во входном файле

```

5
10 150
100 110
131 170
131 180
120 130

```

При таких исходных данных можно провести максимум три мероприятия, например, по заявкам 2, 3 и 5. Конфе-

ренц-зал освободится самое позднее на 180-й минуте, если состоятся мероприятия по заявкам 2, 4 и 5. Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемых файлов.

Решение (Python)

```
a = sorted([list(map(int,s.split())) for s in open('319_26.txt')][1:],key=lambda x: x[1])
b = [a[0]]
while True:
    c = [x for x in a if x[0]>=b[-1][1]]
    if len(c)==0: break
    else: b.append(c[0])
freeLast = 0
for start, end in a:
    if start >= b[-2][1]:
        freeLast = max( end, freeLast )
print(len(b),freeLast)
```

Анализ ошибок

Напомним, что в текущем году к заданию приступили только сильные ученики, написавшие экзамен на 80 и более баллов. Для этих учеников основные приемы программирования не составляют сложности. Однако в этом задании не все так просто. Ответить правильно только на 1-й вопрос удалось 10,16%, на оба вопроса 22,99%, получили 0 баллов 33,16%. Остальные не приступали к заданию. Типичные ошибки учеников:

1. Неправильная сортировка данных

Некоторые ученики сортировали мероприятия по времени начала, а не по времени окончания. Это приводит к некорректному выбору оптимального набора мероприятий.

2. Выбор неподходящего правила обработки

Ученики иногда пытались решить задачу жадным методом, основываясь лишь на первом пришедшем на ум условии (например, выбрали мероприятия с наименьшей продолжительностью или наибольшей выгодностью). Однако правильная стратегия — рассматривать самый ранний срок окончания и двигаться дальше, выбирая следующие совместимые мероприятия. НО! Для ответа на второй вопрос важно брать не ближайшее возможное последнее мероприятие, а мероприятие с самым поздним временем окончания из всех возможных. Именно этот момент многие школьники забывают. Лучше всего с ним справляются учащиеся, имеющие опыт решения

олимпиадных задач, привыкшие искать другие возможные альтернативы.

Способы избежать таких ошибок:

Задание 26 проверяет не базовые, а продвинутые навыки владения программированием и полезна только хорошо подготовленным школьникам, изучавшим углубленный курс информатики. Поэтому, тренировать решение таких задач в классе стоит с сильными учениками.

При подготовке школьников к решению таких задач, важно акцентировать внимание на том, что важна сортировка для лучшего поиска, научить использовать четкий критерий сортировки по времени окончания с использованием lambda переменной: `events.sort(key=lambda x: x[1])` на Python. Необходимо обратить внимание на то, что нужно учитывать главное условие: нельзя совмещать мероприятия, которые накладываются друг на друга по времени.

Для правильного ответа на второй вопрос нужно выбирать не первое же подходящее мероприятие по времени начала после предпоследнего, а самое позднее из оставшихся. Например, если мы получаем последовательность значений времени окончания 15 идущих друг за другом мероприятий [600, 643, 680, 703, 738, 780, 795, 820, 833, 870, 883, 921, 968, 986, 991, ?], то последнее, 16-е мероприятие из оставшихся может начаться с 991-й минуты, или с 992-й, или позднее. Например, пусть известны оставшиеся мероприятия (начало, конец), отсортированные по времени окончания: [[845, 1240], [213, 1241], [994, 1241], [901, 1272], [813, 1273], [879, 1273], [52, 1273], [5, 1273], [991, 1273], [466, 1275], [589, 1275], [403, 1275], [526, 1275], [967, 1336], [992, 1345], [148, 1345], [622, 1347], [533, 1348], [74, 1348], [864, 1349], [283, 1349], [847, 1395], [391, 1399], [444, 1399], [763, 1399], [189, 1399], [179, 1399]]. Из них только одно еще можно провести, причем подходят: [994, 1241], [991, 1273], [992, 1345]. Мы должны по условию задания выбрать не первое подходящее по списку [994, 1241], а то, которое позднее всего закончится: [992, 1345].

Задание наглядно выполняется с помощью электронных таблиц. Возможно, учащиеся, не очень уверенно программирующие, лучше поймут такой способ решения. Важно показать, что возможен и такой прием. Тут нужно так же выполнить сортировку данных по второму столбцу, отметить пары где начало больше или равно концу предыдущей подходящей пары (начиная с пары в первой строке). Это можно сделать с помощью формул: `=ЕСЛИ(A2>=C1;B2;C1)` и `=ЕСЛИ(C2=C1;"";1)`. Так легко посчитать полученные «1» и получить ответ на 1 вопрос. А вот для ответа на второй вопрос нужно все пары, начиная с последней пары, отмеченной «1», отфильтровать по времени начала \geq времени окончания предпоследнего выбранного мероприятия, и отсортировать их по времени окончания. Ответом будем максимальное время окончания.

Задание № 27

Тема: Анализ данных. Кластеризация.

Уровень сложности: высокий.

Рекомендуемое время выполнения: 40 минут.

Умение выполнять последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и построение модели, преобразование данных, визуализация данных, интерпретация результатов.

Проверяемые элементы содержания: Анализ данных. Основные задачи анализа данных: прогнозирование, классификация, кластеризация, анализ отклонений. Последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и/или построение модели, преобразование данных, визуализация данных, интерпретация результатов. Программные средства и интернет-сервисы для обработки и представления данных. Большие данные. Машинное обучение.

Проверяемые требования к предметным результатам базового уровня освоения основной образовательной программы основного общего образования на основе ФГОС 2021 г.: Умение классифицировать основные задачи анализа данных (прогнозирование, классификация, кластеризация, анализ отклонений); понимать последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и/или построение модели, преобразование данных, визуализация данных, интерпретация результатов.

Проверяемые метапредметные результаты: базовые исследовательские действия.

Что нужно знать:

Чтение данных из файла

в языке Python для чтения данных удобно использовать менеджер контекста (**with ... as**), который открывает файл и закрывает его; например, код

```
with open("26.txt") as Fin: # программа и файл в одной папке
```

```
... # какие-то операции с файлом
```

```
# при завершении работы менеджера контекста
```

```
# файл автоматически закрывается
```

равносилен такому

```
Fin = open("26.txt") # открытие файла
```

```
... # какие-то операции с файлом
```

Fin.close() # закрытие файла

если в текущей строке файла находится целое число, то прочитать его в переменную **x** можно так:

```
x = int( Fin.readline() )
```

если в строке записаны два числа, после чтения (**Fin.readline()**) строку нужно разбить на отдельные части по пробелам между числами (каждая часть – символьная запись числа) и затем каждую часть преобразовать в целое число; например, чтение двух чисел:

```
s = Fin.readline()  
symData = s.split()  
x, y = map( int, symData )  
или в компактной форме  
x, y = map( int, Fin.readline().split() )
```

в языке PascalABC.NET для чтения данных проще всего просто перенаправить входной поток на файл:

```
Assign( input, '26.txt' );
```

после этого можно использовать операторы **read** и **readln**, так же, как при вводе с клавиатуры в языке C++ можно читать данные с помощью входного потока (**fstream**):

```
#include <fstream>  
...  
ifstream Fin("26.txt");  
Fin >> x;  
Fin >> y >> z;
```

Принципы кластеризации.

Способы визуализации (диаграммы в электронных таблицах, работа с графикой в программировании).

Пример формулировки задания

Фрагмент звёздного неба спроецирован на плоскость с декартовой системой координат. Учёный решил провести

кластеризацию полученных точек, являющихся изображениями звёзд, то есть разбить их множество на N непересекающихся непустых подмножеств (кластеров), таких, что точки каждого подмножества лежат внутри прямоугольника со сторонами длиной H и W , причём эти прямоугольники между собой не пересекаются. Стороны прямоугольников не обязательно параллельны координатным осям. Гарантируется, что такое разбиение существует и единственно для заданных размеров прямоугольников.

Будем называть центром кластера точку этого кластера, сумма расстояний от которой до всех остальных точек кластера минимальна. Для каждого кластера гарантируется единственность его центра. Расстояние между двумя точками на плоскости $A(x_1, y_1)$ и $B(x_2, y_2)$ вычисляется по формуле:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

В файле А хранятся координаты точек двух кластеров, где $H = 6$ и $W = 4,5$ для каждого кластера. В каждой строке записана информация о расположении на карте одной звезды: сначала координата x , затем координата y . Известно, что количество точек не превышает 1000.

В файле Б хранятся координаты точек трёх кластеров, где $H = 6$, $W = 5$ для каждого кластера. Известно, что количество точек не превышает 10 000. Структура хранения информации в файле Б аналогична структуре в файле А.

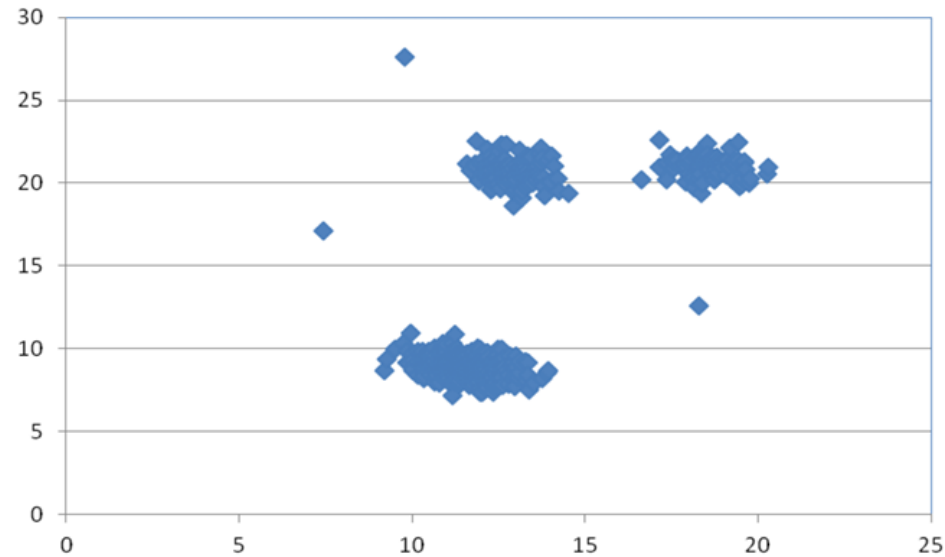
Известно, что в файле Б имеются координаты ровно трёх «лишних» точек, являющихся аномалиями, возникшими в результате помех при передаче данных. Эти три точки не относятся ни к одному из кластеров, их учитывать не нужно.

Для файла А определите координаты центра каждого кластера, затем найдите два числа: P_x – среднее арифметическое абсцисс центров кластеров, и P_y – среднее арифметическое ординат центров кластеров.

Для файла Б определите координаты центра каждого кластера, затем найдите два числа: Q_x – сумму абсцисс центров кластеров с минимальным и максимальным количеством точек, и Q_y – сумму ординат центров кластеров с минимальным и максимальным количеством точек.

Гарантируется, что во всех кластерах количество точек различно. В ответе запишите четыре числа: в первой строке – сначала абсолютную величину целой части произведения $P_x \times 10\,000$, затем абсолютную величину целой части произведения $P_y \times 10\,000$; во второй строке – сначала абсолютную величину целой части произведения $Q_x \times 10\,000$, затем абсолютную величину целой части произведения $Q_y \times 10\,000$.

Возможные данные одного из файлов проиллюстрированы графиком.



Решение (Python)

Решение А

#Считываем данные из файла А (заранее удалили первую строку и заменили "," на ".")

```
data=[]
```

```
for s in open("319_27_A.txt"):
```

```
    x,y=[float(d) for d in s.split()]
```

```
    data.append([x,y])
```

#находим кластеры точек, для которых соседями являются точкам на расстоянии <0.5

```
from math import dist
```

```
clusters=[]
```

```
while data:
```

```
    cl=[data.pop()]
```

```
    for p in cl:
```

```
        sosed=[p1 for p1 in data if dist(p,p1)<0.5]
```

```

    for p1 in sosed:
        cl.append(p1)
        data.remove(p1)
    clusters.append(cl)
#выводим количество точек в каждом полученном кластере, если кластеров больше или меньше 2-х,
#то подгоняем расстояние между соседями (>0.5 или <0.5)
print([len(cl) for cl in clusters])

#С помощью модуля turtle визуализируем полученную картину,
#используя для точек разных кластеров разные случайные цвета
from turtle import *
from random import *
tracer(0)
up()
for cl in clusters:
    color=random(), random(), random()
    for x,y in cl:
        goto(x*10, y*10)
        dot(4, color)
update()
#Функция для определения центров кластеров
def centr(cl):
    m=[]
    for p in cl:
        s = 0
        for p1 in cl:
            s+=dist(p,p1)
        m.append([s,p])
    return min(m)[1]

```

#Финальные ответы

```
cen = [centr(cl) for cl in clusters]
px = abs(sum(x for x, y in cen))/len(cen)
py = abs(sum(y for x, y in cen))/len(cen)
print(int(px*10000),int(py*10000))
```

Пример изображения кластеров:



Решение тем красиво, что для задания В изменения в нем минимальны.

Решение В

#Считываем данные из файла В (как и для А, заранее удалили первую строку и заменили "," на ".")

```
data=[]
```

```
for s in open("319_27_B.txt"):
```

```
    x,y=[float(d) for d in s.split()]
```

```
    data.append([x,y])
```

#находим кластеры точек, для которых соседями являются точкам на расстоянии <0.5

```
from math import dist
```

```
clusters=[]
```

```
while data:
```

```
    cl=[data.pop()]
```

```

for p in cl:
    sosed=[p1 for p1 in data if dist(p,p1)<0.9]
    for p1 in sosed:
        cl.append(p1)
        data.remove(p1)
clusters.append(cl)
#выводим количество точек в каждом полученном кластере, причем кластеров должно
#быть ровно 6 и в последних трех ровно по 1 точке, иначе подгоняем расстояние
#между соседями (>0.9 или <0.9)
len_cl=sorted([len(cl) for cl in clusters])[:-1] #сортируем по убыванию количества точек
print(len_cl)

#С помощью модуля turtle визуализируем полученную картину, как и в задании А,
#используя для точек разных кластеров разные случайные цвета
from turtle import *
from random import *
tracer(0)
up()
for cl in clusters:
    color=random(), random(), random()
    for x,y in cl:
        goto(x*15, y*15)
        dot(2.5, color)
update()

#Функция для определения центров кластеров как и в задании А
def centr(cl):
    m=[]
    for p in cl:

```

```
s = 0
for p1 in cl:
    s+=dist(p,p1)
m.append([s,p])
return min(m)[1]
```

```
#Финальные ответы для задания В
cen = [centr(cl) for cl in clusters if len(cl)==len_cl[0] or len(cl)==len_cl[2]]
Qx = abs(sum(x for x, y in cen))
Qy = abs(sum(y for x, y in cen))
print(int(Qx*10000),int(Qy*10000))
```

Пример изображения кластеров:



Анализ ошибок

Напомним, что это новое задание в КИМ ОГЭ, при этом 92% высокобалльников приступили к его выполнению и не выполнили только 8%. Ответить правильно только на 1-й вопрос удалось трети из этой группы учеников (34,76%), на оба вопроса правильно ответили 50% высокобалльников.

Типичные ошибки учеников:

1. Неправильная кластеризация

Некоторые ученики выбирают способ кластеризации «на глаз», используя точечные диаграммы в электронных

таблицах и неверно относят к кластерам отдельные точки. Особенно эта ошибка актуальна для файла В при отделении аномальных точек. При отделении кластеров с помощью программирования важно правильно построить модели (уравнения разделяющих прямых, расстояние между соседями и пр.)

2. Неверное нахождение центров кластеров

Эта ошибка встречается реже, но бывают при подсчете арифметические ошибки (неверно построена математическая модель) или перебор не всех точек кластера.

3. Неверный подсчет результата

Школьники часто для файла В выводят ответ, как и для файла А: считая среднее арифметическое абсцисс и ординат центров. В этом случае виновата невнимательность при прочтении задания. Если все же задание прочитано правильно, то часто для файла В ученики считают суммы абсцисс и ординат всех кластеров, а не кластеров с наибольшим и наименьшим количеством точек. Причем, аномальные точки следует отбросить.

Бывает, что участники экзамена забывают при выводе ответа умножить результат на 10000 и/или находить целую часть.

4. Нехватка времени на выполнение задания

Способы избежать таких ошибок:

В целом, задание 27 нетрудное, при подготовке школьников к ЕГЭ по информатике важно рассматривать подобные прототипы заданий; обращать внимание на детальное прочтение формулировки задания, на разные подзадания для файлов А и В, на фиксацию важных моментов; разбирать разные способы кластеризации; приучать тестировать промежуточные решения, визуализировать результаты; составлять модели для подсчета необходимых значений; правильно распределять время работы над заданиями.

3.1.3. Анализ метапредметных результатов обучения, повлиявших на выполнение заданий КИМ

Согласно ФГОС в кодификаторе ОГЭ по информатике перечислен перечень проверяемых метапредметных результатов обучения:

1. Овладение универсальными учебными познавательными действиями:

1.1 Базовые логические действия:

1.1.1. Устанавливать существенный признак или основания для сравнения, классификации и обобщения;

1.1.2. Выявлять закономерности и противоречия в рассматриваемых явлениях;

1.1.3. Самостоятельно формулировать и актуализировать проблему, рассматривать её всесторонне; определять цели деятельности, задавать параметры и критерии их достижения;

1.1.4. Вносить коррективы в деятельность, оценивать соответствие результатов целям, оценивать риски последствий деятельности;

1.1.5. Развивать креативное мышление при решении жизненных проблем;

1.2 Базовые исследовательские действия:

1.2.1. Владеть навыками учебно-исследовательской и проектной деятельности, навыками разрешения проблем;

1.2.2. Овладение видами деятельности по получению нового знания, его интерпретации, преобразованию и применению в различных учебных ситуациях, в том числе при создании учебных и социальных проектов;

1.2.3. Формирование научного типа мышления, владение научной терминологией, ключевыми понятиями и методами;

1.2.4. Выявлять причинно-следственные связи и актуализировать задачу, выдвигать гипотезу её решения, находить аргументы для доказательства своих утверждений, задавать параметры и критерии решения;

1.2.5. Анализировать полученные в ходе решения задачи результаты, критически оценивать их достоверность, прогнозировать изменение в новых условиях;

1.2.6. Уметь переносить знания в познавательную и практическую области жизнедеятельности; уметь интегрировать знания из разных предметных областей; осуществлять целенаправленный поиск переноса средств и способов действия в профессиональную среду;

1.2.7. Способность и готовность к самостоятельному поиску методов решения практических задач, применению различных методов познания; ставить и формулировать собственные задачи в образовательной деятельности и жизненных ситуациях; ставить проблемы и задачи, допускающие альтернативные решения; выдвигать новые идеи, предлагать оригинальные подходы и решения; разрабатывать план решения проблемы с учётом анализа имеющихся материальных и нематериальных ресурсов;

1.3. Работа с информацией:

1.3.1. Владеть навыками получения информации из источников разных типов, самостоятельно осуществлять поиск, анализ, систематизацию и интерпретацию информации различных видов и форм представления;

1.3.2. Создавать тексты в различных форматах с учётом назначения информации и целевой аудитории, выбирая оптимальную форму представления и визуализации;

1.3.3. Оценивать достоверность, легитимность информации, её соответствие правовым и морально-этическим

нормам;

1.3.4. Использовать средства информационных и коммуникационных технологий в решении когнитивных, коммуникативных и организационных задач с соблюдением требований эргономики, техники безопасности, гигиены, ресурсосбережения, правовых и этических норм, норм информационной безопасности;

1.3.5. Владеть навыками распознавания и защиты информации, информационной безопасности личности.

Овладение системой универсальных учебных познавательных действий обеспечивает сформированность когнитивных навыков у обучающихся.

2. Овладение универсальными учебными коммуникативными действиями:

2.1. Общение:

2.1.1. Осуществлять коммуникации во всех сферах жизни; владеть различными способами общения и взаимодействия;

2.1.2. Развёрнуто и логично излагать свою точку зрения с использованием языковых средств;

2.1.3. Аргументированно вести диалог.

3. Овладение универсальными учебными регулятивными действиями:

3.1. Самоорганизация:

3.1.1. Самостоятельно осуществлять познавательную деятельность, выявлять проблемы, ставить и формулировать собственные задачи в образовательной деятельности и жизненных ситуациях; давать оценку новым ситуациям;

3.1.2. Самостоятельно составлять план решения проблемы с учётом имеющихся ресурсов, собственных возможностей и предпочтений; делать осознанный выбор, аргументировать его, брать ответственность за решение; оценивать приобретённый опыт; способствовать формированию и проявлению широкой эрудиции в разных областях знаний;

3.2. Самоконтроль:

3.2.1. Давать оценку новым ситуациям, вносить коррективы в деятельность, оценивать соответствие результатов целям;

3.2.2. Владеть навыками познавательной рефлексии как осознания совершаемых действий и мыслительных процессов, их результатов и оснований; использовать приёмы рефлексии для оценки ситуации, выбора верного

решения; уметь оценивать риски и своевременно принимать решения по их снижению.

3.3. Эмоциональный интеллект, предполагающий сформированность: саморегулирования, включающего самоконтроль, умение принимать ответственность за своё поведение, способность адаптироваться к эмоциональным изменениям и проявлять гибкость, быть открытым новому; внутренней мотивации, включающей стремление к достижению цели и успеху, оптимизм, инициативность, умение действовать, исходя из своих возможностей.

Овладение системой универсальных учебных регулятивных действий обеспечивает формирование смысловых установок личности (внутренняя позиция личности) и жизненных навыков личности (управления собой, самодисциплины, устойчивого поведения).

Безусловно уровень сформированности этих умений влияет на результаты выполнения заданий ЕГЭ.

Необходимо отметить, что ключевым фактором выполнения заданий ЕГЭ по информатике является сформированность метапредметных навыков относящихся прежде всего к *универсальными учебными познавательными действиями и учебными регулятивными действиями.*

Во первых, это базовые логические действия, таких как способность самостоятельно выбирать способ решения учебной задачи (сравнивать несколько вариантов решения, выбрать наиболее подходящий с учетом самостоятельно выделенных критериев).

Во-вторых, это навыки работы с информацией, такие как: выбирать, анализировать, систематизировать и интерпретировать информацию различных видов и форм представления; самостоятельно выбирать оптимальную форму представления информации и иллюстрировать решаемые задачи несложными схемами, диаграммами, иной графикой и их комбинациями.

В-третьих, это навыки самоорганизации и самоконтроля, такие как самостоятельное планирование и осуществление целенаправленной деятельности, включая умения анализировать поставленную задачу и те условия, в которых она должна быть реализована; находить эффективные пути достижения результата; выявлять альтернативные, нестандартные способы решения познавательных задач; оценивать правильность выполнения поставленной познавательной задачи.

Эти навыки особенно важны для выполнения компьютерных заданий всех уровней сложности, поскольку они, как правило, предполагают разбиение процесса выполнения заданий на несколько этапов, в каждом из которых требуется продемонстрировать владение как теоретическими, так и практико-ориентированными элементами содержания курса. При этом невнимательное прочтение формулировки задания, неверное выделение всех условий и неверное планирование своих действий может привести к неверному ответу и (или) неэффективному выполнению

задания с точки зрения временных затрат.

В анализе по данному пункту приведем задания с низким процентом выполнения, на успешность выполнения которых могла повлиять слабая сформированность тех или иных метапредметных умений, для каждого приведенного задания:

Задание № 5 и задание № 6

Кроме описанных выше ошибок при выполнении этих заданий, демонстрирующих слабые предметные результаты обучения по выделенным содержательным элементам, на успешность их выполнения могла повлиять прежде всего слабая сформированность следующих метапредметных умений:

Овладение универсальными учебными познавательными действиями: базовые логические действия (1.1.3., 1.1.4.), базовые исследовательские действия (1.2.4, 1.2.5, 1.2.7).

Примеры ошибок по этой причине: неудачно выбранный способ рассуждения, неправильно построенная модель, неверный анализ информации – многие экзаменуемые не увидели части условий; отсутствие проверки и оценки промежуточных результатов, пропуск недостоверно полученных значений, не подходящих под условия; неверно выделены шаги и план решения.

Задание № 8

На успешность выполнения могла повлиять прежде всего слабая сформированность следующих метапредметных умений:

Овладение универсальными учебными познавательными действиями: базовые логические действия (1.1.1., 1.1.4.), базовые исследовательские действия (1.2.4, 1.2.5, 1.2.7).

Примеры ошибок по этой причине: неверно выбран алфавит и упорядочивание слов; отсутствие достоверности проверки промежуточных и итоговых результатов; неверный подбор критериев для проверки условий, выделенных в формулировке задания; отсутствие навыков выполнения задания разными методами для проверки результата; неумение выбрать наиболее эффективный способ решения (комбинаторика, нумерация и применение систем счисления, программирование и пр.)

Задание № 9

Низкий процент выполнения задания прежде всего связан с несформированностью навыков формализации и моделирования, относящимися к навыкам работы с информацией (1.3.2., 1.3.4.).

Примеры ошибок по этой причине: неумение разбивать задачу на подзадачи, сортировать, выделять данные ведет к путанице порядка выполнения и выделению ненужных ячеек; отсутствие навыка проверки рассчитанных значений для конкретного диапазона влечет накопление ошибок.

Задание №17, задания №№ 24-27

На успешность выполнения могла повлиять прежде всего слабая сформированность следующих метапредметных умений:

Овладение универсальными учебными познавательными действиями: базовые исследовательские действия (1.2.4, 1.2.5, 1.2.7).

Овладение универсальными учебными регулятивными действиями: самоорганизация (3.1.1., 3.1.2.); самоконтроль (3.2.1., 3.2.2.).

Процесс решения задачи на ЭВМ требует выполнения следующих этапов:

1. Постановка задачи
2. Формализация и моделирование
3. Разработка алгоритма и теста
4. Выбор языка программирования и разработка программы.
5. Тестирование и отладка

Базовые исследовательские действия 1.2.5. необходимы на каждом из этих этапов и влияют в итоге на результат.

При выполнении всех этих этапов требуется проверка результатов, анализ и коррекция, особенно на последнем этапе, здесь важную роль играют регулятивные действия самоорганизация; самоконтроль. А разработка тестов и тестирование – это самый уязвимый момент в подготовке программированию. Школьники усложняют решения записывая сложные модели, не разбивают задание на подзадачи; не тестируют решения.

3.1.4. Выводы об итогах анализа выполнения заданий, групп заданий:

Перечень элементов содержания / умений и видов деятельности, усвоение которых всеми школьниками региона в целом можно считать достаточным

Достаточный уровень подготовки выпускников школ наблюдается по следующим элементам содержания / уме-

ний и видам деятельности:

умение представлять и считывать данные в разных типах информационных моделей (схемы, карты, таблицы, графики и формулы);

умение строить таблицы истинности и логические схемы;

умение находить информацию в реляционных базах данных;

умение кодировать и декодировать информацию при работе с неравномерными кодами;

умение определять объём памяти, необходимый для хранения графической и звуковой информации;

умение осуществлять информационный поиск средствами операционной системы или текстового процессора;

умение подсчитывать информационный объём сообщения;

умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд при выполнении поиска и замены в строке символов;

умение использовать маску подсети;

умения работать с позиционными системам счисления;

знание основных понятий и законов математической логики, умение их применять в анализе значений сложных логических выражений;

умение вычислять рекуррентные выражения;

умение использовать электронные таблицы для обработки целочисленных данных с применением динамического программирования;

умение анализировать алгоритм логической игры, умение построить дерево игры по заданному алгоритму и найти выигрышную стратегию;

строить математические модели для решения практических задач с многопроцессорными системами.

умение применять динамическое программирование и анализировать ход исполнения алгоритма.

Перечень элементов содержания / умений и видов деятельности, усвоение которых всеми школьниками региона в целом, школьниками с разным уровнем подготовки нельзя считать достаточным

Недостаточный уровень подготовки выпускников школ наблюдается по следующим элементам содержания / умений и видам деятельности:

формальное исполнение простого алгоритма, записанного на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные

данные линейного алгоритма при обработке цифр числа;

определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов при работе с координатной плоскостью;

применение равномерных кодов, комбинаторики, знание основных понятий и методов, используемых при измерении количества информации;

умение обрабатывать числовую информацию в электронных таблицах;

умение создавать собственные программы (10–20 строк) для обработки символьной информации;

умение создавать собственные программы (10–20 строк) для обработки целочисленной информации, для поиска делителей числа;

умение создавать собственные программы для анализа числовых последовательностей, обработки символьной информации, целочисленных данных с использованием сортировки и эффективных методов решения;

умение выполнять последовательность решения задач анализа данных: сбор первичных данных, очистка и оценка качества данных, выбор и построение модели, преобразование данных, визуализация данных, интерпретация результатов.

Выводы об изменении успешности выполнения заданий разных лет по одной теме / проверяемому умению, виду деятельности (если это возможно сделать)

На протяжении 2023-2025 г.г. стабильно ухудшаются результаты выполнения заданий: №4 (декодирование неравномерных кодов), №12 (Умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд для обработки строк), №14 (системы счисления) и №20 (анализ игровой стратегии до 3 хода), хотя они остаются в пределах допустимых значений.

Значительно ухудшились за три года результаты выполнения задания №25 (умение создавать собственные программы (10–20 строк) для обработки целочисленной информации), при этом в 2025 году результаты ниже допустимого значения.

Значительно улучшились результаты выполнения заданий: №7 (умение определять объём памяти, необходимый для хранения графической и звуковой информации), №10 (информационный поиск средствами текстового процессора), №15 (знание основных понятий и законов математической логики) и №22 (построение моделей для анализа работы многопроцессорных систем).

Выводы о связи динамики результатов проведения ЕГЭ с использованием рекомендаций для системы образования Алтайского края и системы мероприятий, включенных с статистико-аналитические отчеты о результатах ЕГЭ по учебному предмету в предыдущие 2-3 года.

Публикация аналитического отчета и рекомендаций по изучению тем, недостаточно усвоенных школьниками в предыдущие годы, повлияло на качественную отработку учителями на уроках фундаментальных тем «Основы математической логики», «Кодирование звука и графики», «Поиск информации средствами текстового редактора», «Законы логики. Логические уравнения», «Многопроцессорные системы». Повысился уровень выполнения отдельных заданий с этими содержательными элементами. Тема «Кодирование звука и графики» перестала быть для школьников проблемной. Задания на работу с логическими уравнениями с параметрами выполняются на достаточном статистическом уровне для заданий повышенной сложности. Однако остаются проблемы с выполнением заданий по теме «Равномерные коды. Комбинаторика», «Электронные таблицы», «Алгоритмизация и программирование».

Раздел 4. РЕКОМЕНДАЦИИ ДЛЯ СИСТЕМЫ ОБРАЗОВАНИЯ АЛТАЙСКОГО КРАЯ

Проведённый анализ и выявленные недостатки позволяют дать рекомендации по совершенствованию процесса преподавания информатики в общеобразовательных организациях Алтайского края.

4.1. Рекомендации по совершенствованию организации и методики преподавания предмета Алтайском крае на основе выявленных типичных затруднений и ошибок

4.1.1. ...по совершенствованию преподавания учебного предмета всем обучающимся

Учителям

1. При организации образовательного процесса по подготовке к ГИА как в рамках изучения предмета по программе, так и на дополнительных курсах подготовки школьников необходимо руководствоваться нормативными документами, регулирующими проведение итоговой аттестации по Информатике и ИКТ, и методическими материалами, которые находятся на официальных сайтах ФИПИ (<http://fipi.ru/>) и Министерства просвещения Российской Федерации (<https://edu.gov.ru/>). Рекомендациями, размещенными на информационном сайте Алтайского края <https://gia.22edu.ru/>.

2. Обучение по информатике и ИКТ в 7-11-м классах необходимо целенаправленно проводить на основе использования заданий, построенных по аналогии с заданиями текущей демоверсии ГИА-11, учить внимательно работать с текстом заданий в КИМ, развивая метапредметные результаты, связанные с навыками работы с информацией и саморегулированием. Например, стоит разрабатывать задания на работу с текстом условия: «Какие данные известны?», «Что является параметром в задаче?», «Сколько значений параметра требуется найти?» и пр. Такие приемы позволят сориентировать обучающихся по содержанию КИМ и типам заданий, а так же сформирует навыки работы с условием задания, самоконтроля.

3. Обучение информатике нужно строить на задачном подходе, для формирования ИКТ компетенций учащихся применять практико-ориентированные задания.

4. Нельзя специально готовить к типовым заданиям ЕГЭ, нужно регулярно, с 7 по 11 класс, на уроках учить информатике и развивать инструментарий школьника, тогда он будет готов к решению любой задачи. Нельзя начать заниматься подготовкой к ЕГЭ исключительно в 11 классе.

Особое внимание стоит уделить темам со стабильно низкими результатами.

Для ликвидации проблем с выполнением задания 5 (формальное исполнение алгоритма, записанного на есте-

ственном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные данные линейного алгоритма по результатам его работы) при изучении понятий «алгоритм и его свойства», «исполнитель» целесообразно подбирать задания, проводить уроки-практикумы, применять на уроках приемы обучения «одна задача – несколько решений», «урок одной задачи». Применять само- и взаимооценивание по критериям, для развития регулятивных УУД.

Для ликвидации проблем с выполнением задания 6 (Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных Алгоритмов) при изучении понятий «алгоритм и его свойства», «исполнитель» использовать систему КуМир, формировать навык преобразования полужформального алгоритма на формальный язык программирования.

Усилить на занятиях отработку навыков формализации и моделирования.

5. При изучении КЭС необходимо целесообразно акцентировать внимание на разных приемах решения задач и работать с разными типами заданий, тогда приобретается полный комплекс средств для инструментария ученика.

Например, в 9 классе по программе формируются навыки вычислений в электронных таблицах и здесь не нужно останавливаться на применении только простейших статистических функций СУММ, МАКС, МИН, СРЗНАЧ..., а применять на уроках задачи, связанные с вычисление логических выражений и применением функций И, ИЛИ, НЕ. Такой подход позволит выстроить преемственность в изучении разных разделов курса информатики, сформировать комплекс навыков применения различного инструментария для решения задач, сформирует опыт их решения в разных ситуациях и позволит учащимся проверять результаты решения задач, применяя разные способы решения, что развивает необходимые метапредметные результаты, связанные с самоорганизацией и самоконтролем. Далее эти навыки актуализируются, углубляются и расширяются в 11 классе, при изучении темы «Анализ данных с помощью электронных таблиц».

6. Сосредоточить внимание педагогов на выявлении текущих трудностей обучающихся и их оперативной коррекции во время учебного процесса (а не на оценивании конечных достижений обучающихся).

КАУ ДПО АИРО им. А.М. Топорова, иным организациям, реализующим программы профессионального развития учителей

Организовать курсы повышения квалификации по программам «Коррекция знаний школьников по информатике на основе анализа типовых ошибок при выполнении заданий ЕГЭ», «Обучение решению задач средствами электронных таблиц».

4.1.2. ...по организации дифференцированного обучения школьников с разными уровнями предметной подготовки

Учителям

В учебном процессе применять следующие подходы и технологии:

системно-деятельностный подход, позволяющий реализовать обучение в разных формах: индивидуальной, парной, групповой;

осуществлять самоуправление и самоуправление учебно-познавательной деятельностью;

реализовывать индивидуализированное обучение различных групп учащихся при планировании содержательной части урока и его структуры;

активнее применять групповую и индивидуальную формы работы на уроке.

Для учащихся, осваивающих информатику на базовом уровне обратить особое внимание на изучение тем «Базовые алгоритмические конструкции», выполнять анализ алгоритмов, записанных на естественном или полужформальном языке, и записывать их с помощью языка программирования; «Электронные таблицы» и «Анализ данных с помощью электронных таблиц». При изучении тем не останавливаться на изучении теории, а сосредоточиться на формировании навыков решения задач, развитии навыков анализа и рассуждений при решении задач. Для учащихся, демонстрирующих успехи в изучении информатики включить в обучение рассмотрение различных вариантов решения задач. Необходимо уделить особое внимание для этой группы учащихся практическим навыкам программирования, применению эффективных методов разработки программ, разработке и применению тестов.

Администрациям образовательных организаций

Обеспечить реализацию дифференцированного подхода, возможности обучающихся учиться в своём темпе в зоне ближайшего развития:

организовать факультативные и индивидуально-групповые занятия, элективные курсы, кружки, секции и т.д. по выбору самих учащихся.

обеспечить возможность применения на уроках групповую и индивидуальную формы работы.

КАУ ДПО АИРО им. А.М. Топорова, иным организациям, реализующим программы профессионального развития

учителей

Разработать и реализовать программы повышения квалификации учителей по темам «Решение задач с применением однопараметрической и многопараметрических сортировок данных», «Обучение школьников работе приемам программирования при анализе символьных цепочек».

4.2. Рекомендуемые темы для обсуждения / обмена опытом на методических объединениях учителей-предметников, в том числе по трансляции эффективных педагогических практик ОО с наиболее высокими результатами

Обеспечение навыков самоорганизации и самоконтроля в обучении информатике и подготовке к ЕГЭ;
Обучение алгоритмизации и программированию в старшей школе; «Как избежать ошибок на ЕГЭ»;
Практика обучения обработке числовых данных средствами электронных таблиц.

4.3. Рекомендуемые направления повышения квалификации работников образования

Работа с работниками образования средствами Портала Министерства образования и науки Алтайского края и Алтайского института цифровых технологий и оценки качества образования «Сдадим экзамены вместе»: Серия вебинаров и онлайн-консультаций по наиболее труднорешаемым заданиям ЕГЭ и доступ к ним через портал Rutube.

Консультации для учителей и учащихся Алтайского края по подготовке к ЕГЭ, пробные тестирования с разбором заданий, дистанционные консультации в течение года средствами платформ Яндекс.Телемост или ВК-звонки (председатель и эксперты ПК, преподаватели ИИТиФМО АлтГПУ).

Разработка и реализация программ повышения квалификации: «Коррекция знаний школьников по информатике на основе анализа типовых ошибок при выполнении заданий ЕГЭ», «Обучение решению задач средствами электронных таблиц», Решение задач с применением однопараметрической и многопараметрических сортировок данных», «Обучение школьников работе приемам программирования при анализе символьных цепочек».

4.4. Рекомендации по другим направлениям

Организация пробных тестирований в школах.

СОСТАВИТЕЛИ ОТЧЕТА ПО УЧЕБНОМУ ПРЕДМЕТУ:

Специалисты, привлекаемые к анализу результатов ЕГЭ по учебному предмету

<i>Фамилия, имя, отчество</i>	<i>Место работы, должность, ученая степень, ученое звание, принадлежность специалиста (к региональным организациям развития образования, к региональным организациям повышения квалификации работников образования, к региональной ПК по учебному предмету, пр.)</i>
Афонины Марина Викторовна	ФГБОУ ВО «АлтГПУ», доцент кафедры теоретических основ информатики, кандидат педагогических наук, председатель предметной комиссии ОГЭ по информатики в Алтайском крае

Специалисты, привлекаемые к подготовке методических рекомендаций на основе результатов ЕГЭ по учебному предмету

<i>Фамилия, имя, отчество</i>	<i>Место работы, должность, ученая степень, ученое звание, принадлежность специалиста (к региональным организациям развития образования, к региональным организациям повышения квалификации работников образования, к региональной ПК по учебному предмету, пр.)</i>
Афонины Марина Викторовна	ФГБОУ ВО «АлтГПУ», доцент кафедры теоретических основ информатики, кандидат педагогических наук, председатель предметной комиссии ОГЭ по информатики в Алтайском крае
Чеверда Ирина Викторовна	КАУ ДПО «Алтайский институт развития образования имени Адриана Митрофановича Топорова», заместитель директора по учебно-методической работе

Ответственный специалист в Алтайском крае по вопросам организации проведения анализа результатов ЕГЭ по учебным предметам

<i>Фамилия, имя, отчество</i>	<i>Место работы, должность, ученая степень, ученое звание</i>
Чибрякова Татьяна Евгеньевна	консультант отдела организации общего образования и оценочных процедур Министерства образования и науки Алтайского края