Содержание

Введение

Представленное пособие включает вторую его часть в количестве 20 занятий, обычно проводимых в весеннем семестре, по материалам таких разделов школьной информатики, как, например, электронные таблицы, массивы, комбинаторика, сетевые технологии, теория алгоритмов, информационные модели и программирование. Структура каждого занятия имеет состав, аналогичный пособию его первой части: домашнее задание, подробный разбор задач по предыдущему домашнему заданию, теоретический материал по заявленной теме занятия. Кроме того, непосредственно на занятии слушателям предлагается выполнить небольшие по объему, как правило, электронные тесты, состоящие обычно из 10 практических примеров. В течение семестра обязательно проводится разбор заданий вариантов контрольных измерительных материалов (КИМ) Единого Государственного Экзамена 2017 - 18 годов по информатике и ИКТ. Домашние задания, предлагаемые слушателям курсов, охватывают широкий курс вопросов, которые рассматривались и в первом семестре, таким образом, осуществляется постоянный контроль знаний всех материалов курса.

Во второй части пособия подробно анализируются задания ЕГЭ повышенного и высокого уровней подготовки, например, задачи на чтение программ на языке программирования и исправление допущенных ошибок; запись алгоритмов на естественном языке или коротких простых программ на языке программирования; построение дерева игры и обоснование выигрышных стратегий; создание собственных программ для решения задач средней сложности**.** На каждом занятии даются подробные рекомендации по оформлению решений таких заданий с кратким и развернутым ответами.

Авторы надеются, что представленные ими материалы в пособии позволят выпускнику выработать собственную стратегию и навыки самостоятельной успешной подготовки к ЕГЭ по информатике и ИКТ.

Практическое занятие 1. Основной состав и структура заданий второго семестра

Домашние задания к занятию 2.

Примечание. При решении задач предварительно нужно указать к какому типу относится алгоритм решения и привести его подробное описание.

Задача 1. По шестнадцатеричной форме внутреннего представления числа в форме с плавающей точкой C8814000 восстановить само число.

Решение. Восстановление числа по его шестнадцатеричной форме внутреннего представления в форме с плавающей точкой.

Шаг 1. Перейдем к двоичному представлению числа в 4-х байтовой ячейке, заменив каждую шестнадцатеричную цифру 4-мя двоичными цифрами: C – 1100, 8 - 1000, 8 – 1000, 1- 0001, 4- 0100, 0000, 0000, 0000.

Шаг 2. Занесем представленные числа в таблицу:

1	100	1000	0100	0000
	1000	0001	0000	0000
31	Mp_2	23		0

Шаг 3. Анализ числа в таблице: получен код отрицательного числа, поскольку в старшем разряде с номером 31 записана 1. Получим порядок числа из уравнения:

$Mp_2 = p_2 + 100\ 0000_2$; $p_2 = 1001000_2 – 100\ 0000_2 = 1000_2 = 8_{10}$.

Шаг 4. Запишем число в форме нормализованного двоичного числа с плавающей точкой с учетом знака числа: –0,1000 0001 0100 0000 0000 0000 $\times 2^{1000}$. Таким образом, число в двоичной системе счисления имеет вид: $–10000001,01_2$.

Шаг 5. Переведем число в десятичную систему счисления:

$–10000001,01_2 = –(1 \times 2^7 + 1 \times 2^1 + 1 \times 2^{-2}) = –129,25_{10}$.

Ответ: $–129,25_{10}$.

Задача 2. Перечислите все решения уравнения: $(a \rightarrow b) \rightarrow c = 0$.

Решение. Знание законов преобразования логических уравнений и построения таблицы истинности.

Шаг 1. Преобразуем исходное логическое уравнение $(a \rightarrow b) \rightarrow c$:

а) по уравнению импликации: $(a \rightarrow b) \rightarrow c = (\neg a + b) \rightarrow c = \neg(\neg a + b) + c$; б) по закону де Моргана: $\neg(\neg a + b) + c = a * \neg b + c$

Шаг 2. Для полученного уравнения $y = a * \neg b + c$ построим таблицу истинности для трех переменных a, b, c:

a	b	c	y= a*⌐ b + c
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Шаг 3. По таблице истинности находим решения, при которых уравнение равно нулю: (0, 0, 0), (0, 1, 0), (1, 1, 0) – всего три решения.
Ответ: три решения - (0, 0, 0), (0, 1, 0), (1, 1, 0)
Задача 3. Скорость самолета измеряется с точностью до 1 км/ч. Какой объем памяти (в байтах) потребуется для хранения значения скорости в бортовом компьютере? Самолет не сверхзвуковой (скорость не более 1200 км/ч).
Решение. Определение количества информации с помощью формулы Хартли.
Шаг 1. Анализ формулы Хартли: $2^n>Q$, где Q в данном случае скорость самолета и равна 1200 км/ч, n – количество информации в битах.
Отсюда, ближайшее число $2^n = 2048 = 2^{11}$. Таким образом, количество информации составляет n =11 бит.
Шаг 2. Преобразуем полученное количество информации в байты:
n = 11/8 = 1, 375 байт
Ответ: 1, 375 байт
Задача 4. В группе имеется 23 студента; 10 из них умеют играть в шахматы, 8 — в шашки; и в шахматы, и в шашки играют 4 студента. Сколько студентов не умеет играть ни в шахматы, ни в шашки?
Решение. Задача является логической задачей, графическое решение которой использует круги Эйлера.
Шаг 1. Введем обозначения: С – общее количество студентов, ШМ – количество студентов, которые играют в шахматы, Ш - количество студентов, которые играют в шашки, ШМШ - количество студентов, которые играют в шахматы и шашки, ⌐ (ШМШ) - количество студентов, которые не играют в шахматы и шашки.
Шаг 2. По условиям задачи построим круги Эйлера (рисунок 1 – графическое изображение к задаче 4):

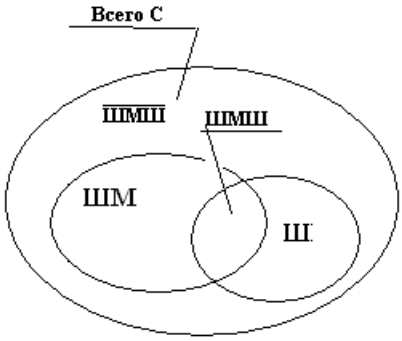


Рисунок 1 – Графическое изображение к задаче 4

Шаг 3. Согласно рисунку 1− (ШМШ)= С – (ШМ+Ш-ШМШ) = 23-(10+8-4) =9 (студентов)

Ответ: 9 студентов не умеют играть в шахматы и шашки.

Задача 5. У исполнителя Полтор две команды, которым присвоены номера: **1. прибавь один,**

2. умножь на полтора.

Первая из них увеличивает на 1 число на экране, вторая увеличивает это число в 1,5 раза, если число чётное. К нечётным числам вторая команда неприменима. Программа для Полтора - это последовательность команд.

Сколько существует программ, которые число 1 преобразуют в число 22?

Решение. Относится к алгоритмам решения задач динамического программирования.

Шаг 1. Обозначим R(n) — количество программ, которые преобразуют число 1 в число n. Составляем рекуррентные формулы, для составления которым переменим заданные программы на обратные и запомним, что для получения числа самого из себя существует ровно одна программа, которая ничего не делает.

При этом верны следующие утверждения:

1. Если n делится на 1,5 и результат деления — нечётное число, то тогда рекуррентная формула имеет вид: R(n) = R(n−1), так как существует единственный способ получения n из (n−1) — прибавлением единицы.

2. Если n делится на 1,5 и результат деления — чётное число, тогда рекуррентная формула имеет вид: R(n) = R(n / 1,5) + R(n − 1). При этом R от четного числа всегда равно нулю, R от числа, которое после вычитания 1 не кратно 2, тоже равно нулю.

Шаг 2. Заполняем следующую таблицу:

Числа R(n) **2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22**

Программы **1 2 2 2 4 4 4 8 8 8 12 12 12 20 20 20 32 32 32 44 44**

При этом R(2) = 1, R(3) = 2 (можно умножить двойку на 1,5 или прибавить 1 к единице), R(4) = R(3) = R(5) = R(4) = 2, R(6) = R(4) + R(5) = 4, R(7)= R(8)=4, R(9) = R(6) + R(8) = 4 + 4 = 8, R(10)= R(10)=8, R(12) = R(8) + R(11) = 4 + 8 = 12, R(13)= R(14)=12, R(15) = R(10) + R(14) = 8 + 12 = 20, R(16)= R(17)=20, R(18) = R(12) + R(17) = 12 + 20 = 32, R(19)= R(20)=32, R(21) = R(14) + R(7) = 12 + 32 = 44, R(22) = R(19) + R(7) = 44.

Ответ: 44

Задача 6. Все 4-буквенные слова, составленные из букв Л, М, С, Т, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. ЛЛЛЛ 2. ЛЛЛМ 3. ЛЛЛС 4. ЛЛЛТ 5. ЛЛМЛ …

Запишите слово, которое стоит под номером 156.

Решение. Задача относится к задачам на системы счисления.

Шаг 1. Введем обозначения: Л – 0, М -1, С – 2, Т – 3.

Шаг 2. Заменим буквенный список на числовой список:

1. 0000, 2. 0001, 3. 0002, 4. 0003

Шаг 3. Анализ списка: Определим систему счисления: четверичная система счисления, использует 4 буквы или цифры.

Порядковое место в списке: так как список в десятичной системе счисления будет: 1. 0, 2. 1, 3. 2, 4. 3, то есть число справа имеет обозначение на единицу меньше, чем число слева. Значит можно предположить, что на 156 месте должно быть число 155.

Шаг 4. Переводим 155 в четверичную систему:

$155_{10} = 2* 4^3 + 1*4^2 + 2*4^1 + 3*4^0 = 2123_4$

Шаг 5. Переводим полученное число в его буквенное обозначение: 2123 = **СМСТ**

Ответ: СМСТ

Задача 7. «О музыкантах». В оркестр приняли на работу трех музыкантов Брауна, Смита и Вессона, умеющих играть на скрипке, флейте, альте, кларнете, гобое и трубе. Известно, что:

1. Смит самый высокий;
2. Играющий на скрипке меньше ростом играющего на флейте;
3. Играющие на скрипке и флейте и Браун любят пиццу;
4. Когда между альтистом и трубачом возникает ссора, Смит мирит их;
5. Браун не умеет играть ни на трубе, ни на гобое.

На каких инструментах играет каждый из музыкантов, если каждый владеет двумя инструментами?

Решение. Задача относится к логическим задачам, которые обычно решаются табличным способом.

Шаг 1. Строится таблица с соответствующими строками и столбцами:

Инструменты/Фамилии	**скрипка**	**флейта**	**альт**	**гобой**	**труба**	**кларнет**
Браун						
Смит						
Вессон						

Шаг 2.

По условиям задачи заполняем таблицу, рассуждая следующим образом:

Так как музыкантов трое а инструментов шесть и каждый владеет только двумя инструментами, получается что каждый музыкант играет на инструментах, которыми остальные не владеют.

Условие 5: прочерк в строке Браун на столбцах «гобой» и «труба»;

Шаг 3. Условие 3: прочерк в строке Браун на столбцах «скрипка» и «флейта».

Отсюда делаем вывод по таблице, что Браун играет на альте и кларнете, ставим ему соответственно в таблице «+», а остальным ставим «-».

Шаг 4. Условие 4: прочерк в строке Смит на столбцах «альт» и «труба»; таким образом, из таблицы видно, что на трубе играет только Вессон (ему в столбце «+»).

Шаг 5. Условия 1и 2: Смит не играет на скрипке (ставим ему «-»), по таблице также делаем вывод, что Вессон играет на скрипке (в столбце «+»).

Шаг 6. Отсюда делаем вывод по таблице, что Смиту остается играть на флейте и гобое, ставим ему «+» в этих столбцах.

Инструменты/Фамилии	**скрипка**	**флейта**	**альт**	**гобой**	**труба**	**кларнет**
Браун	-	-	+	-	-	+
Смит	-	+	-	+	-	-
Вессон	+	-	-	-	+	-

Ответ: Браун – альт и кларнет, Смит – флейта и гобой, Вессон – скрипка и труба.

Лекция (по материалам презентации)

Тема: Основной состав и структура заданий второго семестра.

Во втором части пособия предлагается изучить следующие темы:

1. Электронные таблицы на примерах табличного процессора MS Excel.
2. Сетевые технологии: архитектура компьютерных сетей.

3. Информационные модели: моделирование и компьютерный эксперимент.
4. Массивы (заполнение, считывание, поиск, сортировка, массовые операции и др.).
5. Задачи комбинаторики.
6. Основы и элементы теории алгоритмов.
7. Программирование. Основные понятия эффективности программ.
8. Технологии поиска и хранения информации.

При этом многие задачи ЕГЭ части 1 уже рассматривались нами в первой части пособия, но задания части 2 ЕГЭ непосредственно изучаются во втором семестре и состоят из следующих 4 задач:

1. Практические задачи по чтению программ на языке программирования и исправление допущенных ошибок.
2. Задачи на решение записи алгоритмов на естественном языке или коротких простых программ на языке программирования.
3. Практические задачи на построение дерева игры и обоснование выигрышных стратегий.
4. Практические задачи на создание собственных программ для решения задач средней сложности.

Необходимо на экзамене для данных задач давать либо краткий, либо развернутый ответы. Выполнение заданий части 2 оценивается от 0 до 4 баллов. Ответы на задания части 2 проверяются и оцениваются экспертами. Максимальное количество баллов, которое можно получить за выполнение заданий части 2, равно 12.

В соответствии с Порядком проведения государственной итоговой аттестации по образовательным программам среднего общего образования

(приказ Минобрнауки России от 26.12.2013 № 1400 зарегистрирован Минюстом России 03.02.2014 № 31205) эксперты осуществляют следующие функции:

«61. По результатам первой и второй проверок эксперты независимо друг от друга выставляют баллы за каждый ответ на задания экзаменационной работы ЕГЭ с развернутым ответом...

62. В случае существенного расхождения в баллах, выставленных двумя экспертами, назначается третья проверка. Существенное расхождение в баллах определено в критериях оценивания по соответствующему учебному предмету.

Эксперту, осуществляющему третью проверку, предоставляется информация о баллах, выставленных экспертами, ранее проверявшими экзаменационную работу».

Если расхождение в отметках составляет 2 и более балла за выполнение любого из последних заданий (24-27), то третий эксперт проверяет ответы только на те задания, которые вызвали столь существенное расхождение.

Практические задачи, которые предлагаются для самостоятельного решения на занятии 1.

Решения данных задач приведены в приложении 1.

Задача 1. Сколько существует натуральных чисел x, для которых выполнено неравенство: $11011101_2 < x < DE_{16}$?

В ответе укажите только количество чисел, сами числа писать не нужно.

Задача 2. Определить информационный объем стерео аудиофайла, у которого глубина кодирования звука составляет 4 бит, частота дискретизации звука 50 КГц, длительность звучания 5 мин, количество каналов звучания равно 2.

Задача 3. Для кодирования цвета фона страницы Интернет используется атрибут bgcolor=”XXXXXX”, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB – модели. Какой цвет будет у страницы, заданной тэгом <body bgcolor=”#FFXXXX”>?

Задача 4. Для какого из приведенных чисел X логическое условие истинно?

((X<15) /\ (2·X>23)) → ((X<14) /\ (X>15))

1. 11 2. 12 3. 13 4. 14

Задача 5. «О сестрах». Три сестры Джуди, Айрис и Линда приобрели известность в разных видах искусства – пение, балет и кино. Все они живут в разных городах – Чикаго, Риме и Париже. Известно, что:

1. Джуди живет не в Париже, а Линда не в Риме;
2. Парижанка не снимается в кино;
3. Та, кто живет в Риме – певица;
4. Линда равнодушна к балету.

Какова профессия каждой и где они живут?

Практическое занятие 2. Электронные таблицы

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 1.

Домашние задания к занятию 3.

Задача 1.

Дан фрагмент электронной таблицы (таблица 1). Из ячейки A2 в ячейку B3 была скопирована формула. При копировании адреса ячеек в формуле
автоматически изменились. Запишите в ответе числовое значение формулы
в ячейке B3.

Примечание: знак $ обозначает абсолютную адресацию.

	A	B	C	D	E
1	40	4	400	80	7
2	=$C$2+D$3	3	300	70	6
3	20		200	50	5
4	10	1	100	30	4

Решение.

При копировании все абсолютные ссылки (со знаком $) не меняются.
Формула **=$C$2+D$3** в ячейке **A2** содержит одну абсолютную и одну смешанную ссылки:
— в первой **$C$2** — адрес ячейки не меняется при копировании
— во второй **D$3** — не меняется адрес строки 3 при копировании
Формула **=$C$2+D$3** была скопирована из ячейки **A2** в ячейку **B3,** поэтому первая ссылка не изменилась, а вторая ссылка сместилась на одну строку вниз (увеличилась на одну строку).
Следовательно, после копирования формула **=$C$2+D$3**, примет вид: **=$C$2+E$3**. Из таблицы видно, что значение в B3 равно: 300+5=305

Ответ: 305

Задача 2.

В ячейке F7 электронной таблицы записана формула =D$12-$D13. Какой вид приобретет формула, после того как ячейку F7 скопируют в ячейку E8?

Решение.

D$12: меняется столбец и не меняется номер строки. $D13: столбец не меняется, меняется номер строки. Номер столбца E меньше номера столбца F на 1, поэтому столбец D станет столбцом C. Номер строки E8 на 1 больше номера строки F7, значит, строка 13 станет строкой 14. Значит, формула в ячейке E8 приобретает вид: =C$12-$D14.

Ответ: =C$12-$D14

Задача 3.

В электронной таблице значение формулы =СРЗНАЧ(A3:D3) равно 5. Чему равно значение формулы =СУММ(A3:C3), если значение ячейки D3 равно 6?

Решение.

Функция СРЗНАЧ(A3:D3) считывает среднее арифметическое диапазона A3:D3, т. е. сумму значений четырёх ячеек A3, B3, C3, D3, делённую на 4. Если умножить среднее значение на число ячеек, то получим сумму значений ячеек A3 + B3 + C3 + D3 = 5 * 4 = 20.

Если вычесть значение ячейки D3 из полученного числа, то находим искомую сумму: A3 + B3 + C3 = 20 - 6 = 14.

Ответ: 14

Задача 4.

Дан фрагмент электронной таблицы. Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:C2 соответствовала рисунку? Известно, что все значения диапазона, по которым построена диаграмма, имеют один и тот же знак.

	A	B	C
1	2		=A1*4
2	=b1/A1	=C1/B1	=B2+A1

Решение.

Определим значения в тех ячейках таблицы, в которых это возможно.

	A	B	C
1	2		8
2	=B1/2	=8/B1	=8/B1+2

Из диаграммы видно, что две ячейки должны быть равны друг другу. B2 не равно C2 значит, A2 = B2, а значение в ячейке C2 в два раза больше.

Таким образом: В1/2 = 8/В1, В1 = 4.

Ответ: 4.

Задача 5.

В цехе трудятся рабочие трех специальностей: токари (Т), слесари (С), фрезеровщики (Ф). Каждый рабочий имеет разряд, не меньший второго и не большей пятого. На диаграмме I отображено распределение рабочих по специальностям, на диаграмме II – их количество по разрядам. При этом каждый рабочий имеет одну специальность с одним разрядом.

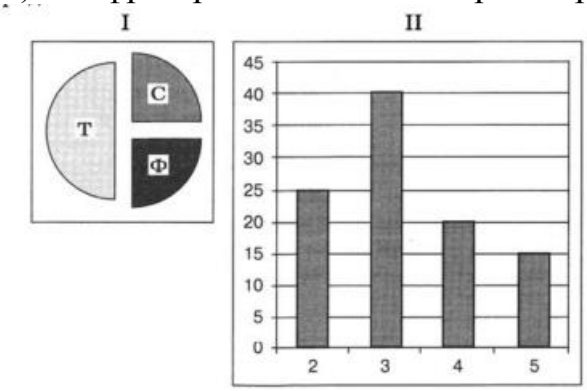
Какое из утверждений является верным:

А) среди слесарей имеется хотя бы один третьего разряда

Б) среди токарей имеется хотя бы один второго разряда

В) все токари имеют четвертый разряд

Г) все фрезеровщики имеют третий разряд



Решение.

Из диаграммы II следует, что больше всего рабочих третьего разряда – 40 человек, второго разряда 25 человек, четвертого – 20 человек, пятого – 15 человек.

Подсчитываем общее количество рабочих:

40+25+20+20+15=100 человек.

Из диаграммы I следует, что токарей 50%, то есть 50 человек, а слесарей и фрезеровщиков поровну, по 25% - 25 человек.

Рассматриваем последовательно утверждения:

А) неверно: токари и фрезеровщики составляют 25+50 =75 человек, третий разряд имеют 40 человек, поэтому только первые могут иметь третий разряд, а слесари 2-й, 4-й и 5-й разряды.

Б) неверно: слесари и фрезеровщики составляют 50 человек, а второй разряд имеют 25 человек, поэтому они могут иметь этот разряд, а не токари.

В) неверно: токарей всего 50 человек, четвертый разряд имеют 20 человек, поэтому это не токари.

Г) это утверждение верно, так как фрезеровщиков 25 человек, с третьим разрядом 40 человек, вполне возможно, что в их число входят все фрезеровщики.

Лекция (по материалам презентации).

Тема: Электронные таблицы.

Определение: Электронная таблица (ЭТ)– это работающее в диалоговом режиме приложение, хранящее и обрабатывающее данные в прямоугольных таблицах.

ЭТ иначе называют табличным процессором. Пример ЭТ **– Microsoft Excel.**

Основные понятия ЭТ – строка, столбец, ячейка, рабочий лист, рабочая книга.

Основные разделы лекции:

- **Основные типы и форматы данных.**
- **Относительные и абсолютные ссылки.**
- **Встроенные функции.**
- **Диаграммы и графики.**

Словарь терминов (глоссарий):

1. **Блок -** любая прямоугольная часть таблицы, которая обозначается именами диагонально-противоположных ячеек, разделенных двоеточием.
2. **Ячейка,** с которой производятся какие-то действия, выделяется рамкой и называется **активной.**

Основные типы и форматы данных

Основные виды информации в ЭТ:

- **Числа (числовой формат).** Выравниваются по правому краю.
- **Текст (символьный формат).** Выравниваются по левому краю.
- **Формула.** Начинается со знака равенства и включает числа, имена ячеек, функции и знаки математических операций. Не включает текст.
- **Специализированный формат** (например, денежный формат).

Выбор формата ячеек осуществляется с помощью меню **«Формат ячеек».**

Режимы отображения информации в ЭТ:

- **Режим отображения ячеек.**
- **Режим отображения формул.**
- **Режим отображения значений с невозможностью вычислений.**

Режимы отображения информации в электронных таблицах представлены в таблице 1. В случае занесения в ячейку таблицы формулы может выводиться целочисленное значение по данной формуле, сама формула или сообщение об ошибке введения формулы.

Таблица 1 – Режимы отображения информации в ЭТ

Введенная информация	**Выведенная информация**
Занесено число	Выведено число
Занесен текст	Выведен текст или его часть
Занесена формула	• Выведено вычисленное значение • Выведена формула • Выведено сообщение об ошибке

Типичные ошибки отображения информации в ЭТ:

- **#### - столбец недостаточно широк для отображения результата или время и дата имеют отрицательные числа.**
- **#ДЕЛ/0! – деление на ноль.**
- **#Н/Д – не задан аргумент функции или значение недоступно формуле.**
- **#ИМЯ! – невозможность распознавания имени.**
- **#ПУСТО! – заданные пересекающие области не имеют общих ячеек.**
- **#ЧИСЛО! – неправильные числовые значения.**
- **#ССЫЛКА! – неверная ссылка на ячейку.**
- **#ЗНАЧ! – недопустимый тип аргумента или операнда (например, вместо числа введен текст).**

Относительные и абсолютные ссылки:

Ссылки на ячейки бывают:

- **Относительные**
- **Абсолютные**
- **Смешанные**

Относительные ссылки определяют положение ячейки и изменяют свой адрес при копировании.

Абсолютные ссылки – постоянные, указывают всегда на одну ячейку и при копировании адрес ячейки не меняют.

Смешанные ссылки имеют постоянную часть адреса.

Замораживание адреса ячейки – процедура вставки знака $ в имя адреса ячейки (частичного $A1 или полного $A$1).

Встроенные функции.

Основные встроенные функции ЭТ:

- **Математические функции (SIN(), COS(), TAN(), КОРЕНЬ, СУММ())**
- **Логические (ЕСЛИ(), И(), ИЛИ())**
- **Статистические (СРЗНАЧ(), МИН(), МАКС())**
- **Финансовые**

- **Дата и время**
- **и т. д.**

Диаграммы и графики.

Диаграмма – средство наглядного графического отображения информации, предназначенное для сравнения значения величин и слежения за изменением значений величин.

Основные стандартные диаграммы:

- **Круговая (Pie) –** показывает составные части одного целого.
- **Линейчатая или ярусная (Bar) –** показывает соотношение между данными в горизонтальных столбиках**.**
- **Столбчатая или гистограмма (Column) –** показывает соотношение между данными в вертикальных столбиках.
- **Линейная или график (Line) –** отражает динамику развития для большого числа данных.
- **Областная или диаграмма площадей или кольцевая (Doughnut)** – диаграмма, в которой различные элементы показываются сегментами.

Пример работы в MS Excel - решение уравнений:

1) Решить уравнение $\mathbf{y=\sin^2(5x+1)}$
2) Решить уравнение $\mathbf{z=(x^2+tg(x+e^{-x}))/2}$

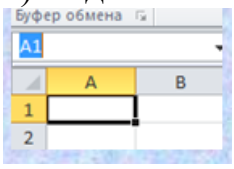
Следует отметить, что формулы в редакторе MS Excel обычно преобразуют в следующий вид:

1) **y=SIN(5*x+1)* SIN(5*x+1)**
2) **z =(x*x+TAN(x+EXP(-x))/2**

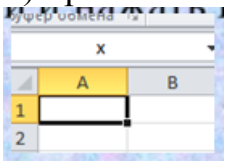
Удобно присвоить именам ячеек значения переменных. Например, ячейке A1 присвоим значение **x,** B1- **y,** C1- **z.** При этом для входной переменной x вводим значение 2.

Для присваивания имени переменной и ввода числа в ячейке A1 осуществляем следующие действия:

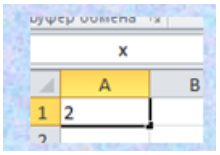
а) выделить имя в названии ячейки:



б) присвоить ей значение переменной и нажать ENTER:



в) ввести число в ячейку:



Аналогично присваиваем значения переменных другим ячейкам. В данные ячейки вводим уравнения одной строкой (название некоторых функций в редакторе отличаются от принятых обозначений), и редактор выводит результат MS Excel:

	A	B	C
1	2	0,99998	2,067668
2			
3			

Условное форматирование MS Excel.

Для заливки ячеек используют условное форматирование:

Пример: Какого цвета будет число 35, если для данной ячейки установлен формат (Красный)(<=10);(Синий)(<=50);(Зеленый) (рисунок 2)?

Ответ: **Синий**

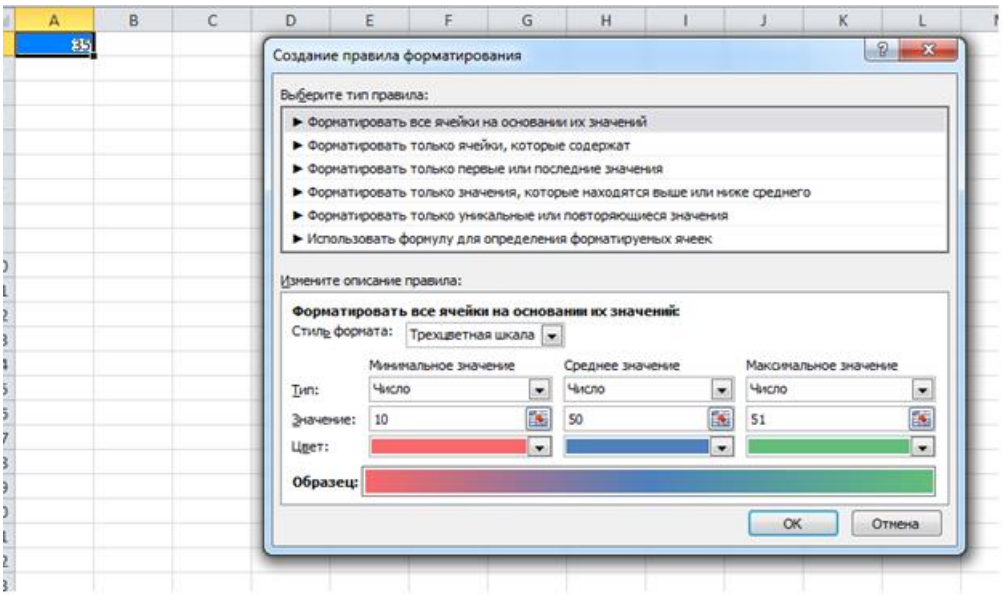


Рисунок 2 – Пример условного форматирования в MS Excel

Задача 2017 года. Дан фрагмент электронной таблицы. Из ячейки A2 в ячейку B3 была скопирована формула. При копировании адреса ячеек в формуле автоматически изменились. Запишите в ответе числовое значение формулы в ячейке B3. Примечание: знак $ обозначает абсолютную адресацию.

	A	B	C	D	E
1	40	4	400	80	7
2	=C$2+D$3	3	300	70	6
3	20		200	50	5
4	10	1	100	30	4

Решение.

Формула **=C$2+D$3** в ячейке **A2** содержит две смешанные ссылки. — в первой **C$2** — адрес строки 2 не меняется при копировании — во второй **D$3** — не меняется адрес строки 3 при копировании.

Формула **=C$2+D$3** была скопирована из ячейки **A2** в ячейку **B3:**

– сместилась на один столбец вправо (увеличилась на один столбец)

– сместилась на одну строку вниз (увеличилась на одну строку)

Следовательно, после копирования формула **=C$2+D$3**, примет вид **=D$2+E$3**.

Вычисление этого выражения дает следующий результат: **70+5=75**.

Ответ: 75

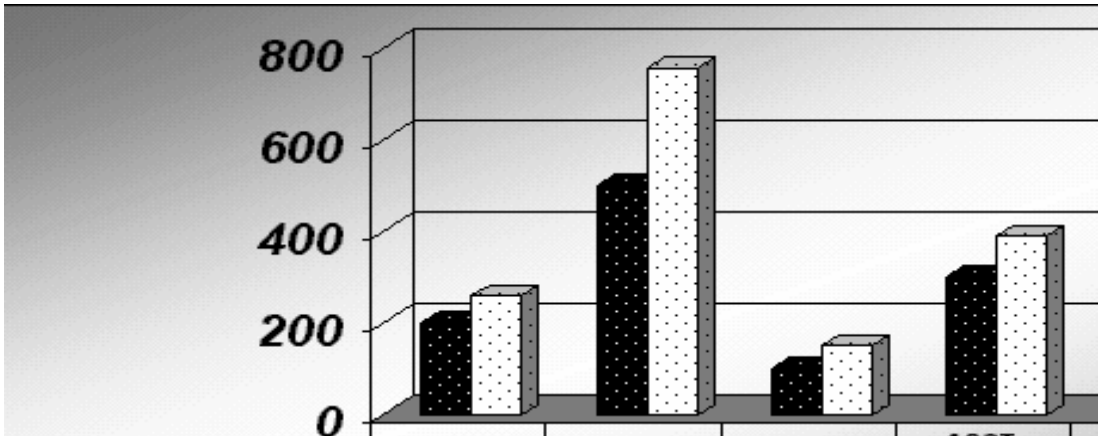
Практические задачи на занятии.

Практическая задача 1.

В электронной таблице значение формулы =СУММ(B2:D2) равно 15. Чему равно значение ячейки A2, если значение формулы = СРЗНАЧ(A2:D2) равно 4?

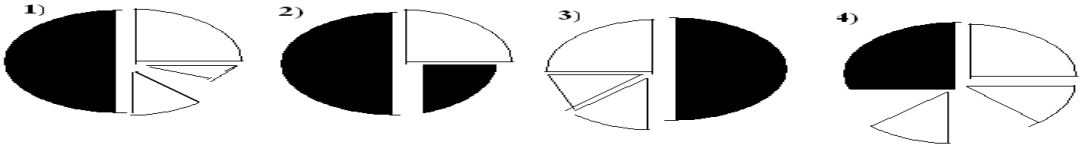
Практическая задача 2.

На диаграмме показано суммы возвратов кредита некоторых фирм. Какая из диаграмм правильно указывает суммы по этим фирмам?



	АО Родник	СП Изотоп	ООО Термо	АОЗТ Восход
Сумма кредита (млн. руб.)	200	500	100	300
Сумма возврата (млн. руб)	260	755	150	390

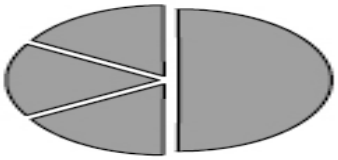
Варианты ответов:



Практическая задача 3.

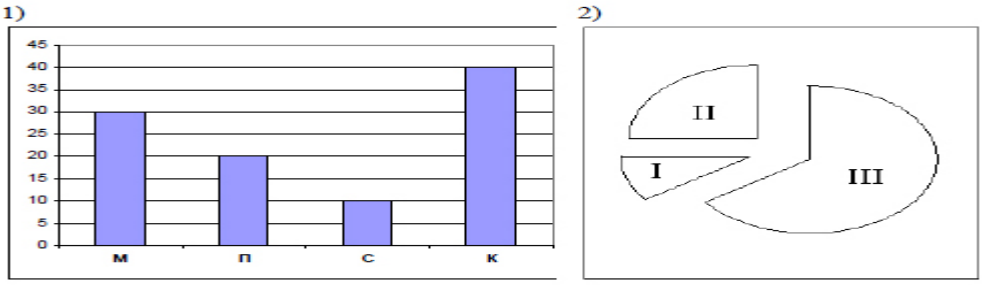
Дан фрагмент электронной таблицы. Какое число должно быть записано в ячейке B1, чтобы построенная после выполнения вычислений диаграмма по значениям диапазона ячеек A2:D2 соответствовала рисунку.

	A	B	C	D
1	3		3	2
2	=(C1+A1)/2	=C1–D1	=A1–D1	=B1/2



Практическая задача 4.

Торговое предприятие владеет тремя магазинами (I, II и III), каждый из которых реализует периферийные компьютерные устройства: мониторы (М), принтеры (П), сканеры (С) или клавиатуры (К). На диаграмме 1 показано количество проданных товаров каждого вида за месяц. На диаграмме 2 показано, как за тот же период соотносятся продажи товаров (в штуках) в трех магазинах предприятия.



Какое из приведенных ниже утверждений следует из анализа обеих диаграмм? А) Все сканеры могли быть проданы через магазин III Б) Все принтеры и сканеры могли быть проданы через магазин II В) Все мониторы могли быть проданы через магазин I

Г) Ни один принтер не был продан через магазин II

Практическое занятие 3. Сетевые технологии: задачи IP

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 3.

Домашние задания к занятию 4.

Задача 1.

В терминологии сетей TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса узла сети относится к адресу сети, а какая - к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес, - в виде четырех байтов, причём каждый байт записывается в виде десятичного числа. При этом в маске сначала (в старших разрядах) стоят единицы, а затем с некоторого разряда - нули. Адрес сети получается в результате применения поразрядной конъюнкции к заданным IP-адресу узла и маске. Например, если IP-адрес узла равен 231.32.255.131, а маска равна 255.255.240.0, то адрес сети равен 231.32.240.0.

Для узла с IP-адресом 119.83.208.27 адрес сети равен 119.83.192.0. Каково
наименьшее возможное количество единиц в разрядах маски?

Решение.

Так как адрес сети получается в результате применения поразрядной конъюнкции к заданным IP-адресу узла и маске, то маску в двоичной системы счисления вычисляют из соотношения:

01110111.01010011.11010000.00011011(119.83.208.27)*????????.??????
??. ????????. ????????=
01110111.01010011.11000000.00000000(119.83.192.0).

Таким образом, для нахождения адреса маски нужно выполнить операцию, обратную логическому умножению.

Тогда первые старших разряда маски состоят из 16 единиц, последний разряд равен 0, средние разряды получают из соотношения: (11010000)*
(????????)=(11000000), и они равны 11000000=192, то есть средним разрядам адреса сети, и имеют в своем составе 2 единицы.

Отсюда: в разрядах маски всего возможных 18 единиц.

Ответ: 18 единиц.

Задача 2.

В терминологии TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса относится к адресу сети, а какая — к адресу самого адреса узла в этой сети. Маска записывается по тем же правилам, что и IP-адрес. При этом адрес сети получается в

результате применения поразрядной конъюнкции к заданному IP-адресу сети и маске.

По заданному IP-адресу узла и маске определите адрес сети, если IP-адрес узла 217.233.232.3, маска имеет адрес 255.255.252.0.

Решение.

По условию задачи следует, что первые 16 старших разрядов имеют разряды IP-адреса, младшие равны 0, а средние определяются из выражения: 11101000(232)*11111100(252)=11101000(232), то есть адрес сети имеет вид:

217.233.232.0

Ответ: 217.233.232.0

Задача 3.

Для некоторой TCP/IP-подсети используется маска 255.255.240.0. Сколько различных адресов компьютеров допускает эта маска?

Примечание. На практике два из возможных адресов не используются для адресации узлов сети: адрес сети, в котором все биты, отсекаемые маской, равны 0, и широковещательный адрес, в котором все эти биты равны 1.

Решение.

Переводим маску 255.255.240.0 в двоичную систему счисления: 11111111.1111111.11110000.00000000. Адреса компьютеров с нулевыми битами образуют пространство, равное 2^{12}, т. е. 4096.

Поскольку 2 из них не используют (адрес сети и широковещательный адрес), то получаем 4096-2= 4094.

Ответ: 4094 компьютера.

Задача 4.

Если маска TCP/IP-подсети 255.255.255.240 и IP-адрес компьютера в сети 162.198.0.34, то чему равен порядковый номер компьютера в сети?

Решение.

Первые три числа в маске равны 255, в двоичной системе это 8 единиц, поэтому первые три числа IP-адреса компьютера целиком относятся к номеру сети. Для последнего числа (октета) маска и соответствующая ей последняя часть IP-адреса равны $240 = 11110000_2$ и $34 = 10010_2$.

Нулевые биты маски и соответствующие им биты IP-адреса определяют номер компьютера в сети: $10_2 = 2$

Ответ: 2

Задача 5.

Если маска подсети 255.255.244.0 и IP-адрес компьютера в сети 162.198.73.45, то чему равен полный номер узла в сети?

Решение.

Первые два числа в маске равны 255, в двоичной системе это 8 единиц, поэтому первые два числа IP-адреса компьютера целиком относятся к номеру сети, в этой задаче они не используются.

Последнее число в маске – 0, поэтому последнее число IP-адреса целиком относится к номеру узла.

Третье число маски – $244 = 11110100_2$, это значит, что первые 4 бита третьей части адреса (73) относятся к адресу сети, а последние 2 бита – к номеру узла:

$244 = 11110100_2$ и $73 = 01001001_2$.

Нулевые биты маски и соответствующие им биты IP-адреса, определяющие старшую часть номера компьютера в сети: $01_2 = 1$

Кроме того, нужно учесть еще и последнее число IP-адреса ($45 = 00101101_2$), таким образом, полный номер компьютера (узла) в двоичной и десятичной системах имеет вид: $01.00101101_2 = 1.45$

Для получения полного номера узла нужно перевести число 0100101101_2 в десятичную систему: $0100101101_2 = 301$

Ответ: 301.

Задача 6.

Определите, какое из указанных имен файла удовлетворяет маске ?o???*.

Варианты ответов: 1) bob9 2) glossy 3) sos11 4) onion

Решение.

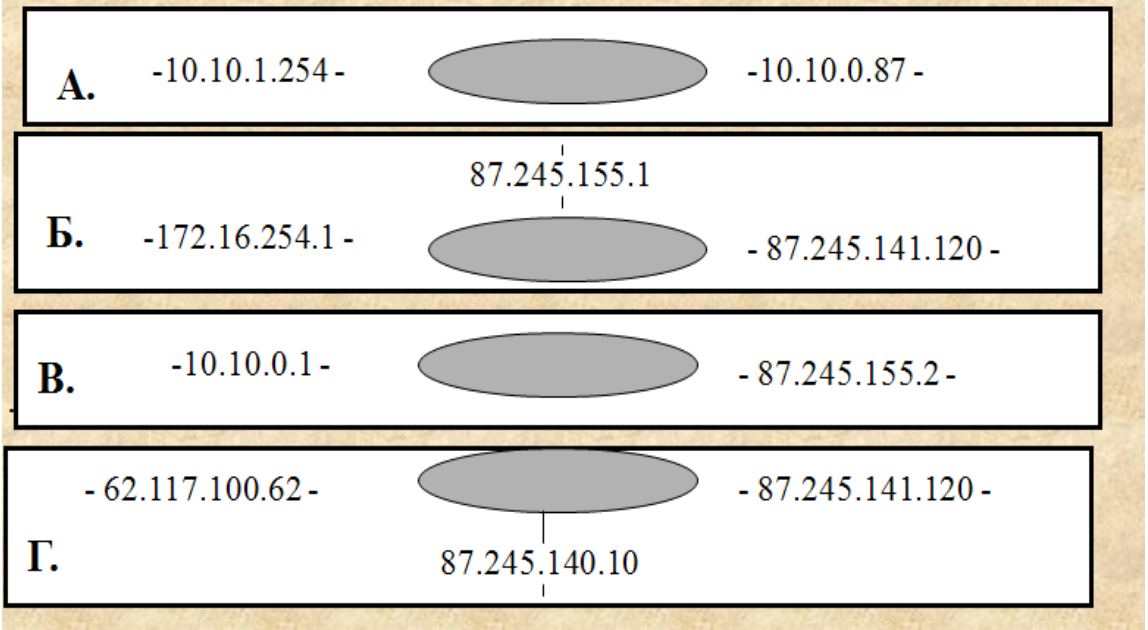
Полное имя файла состоит из расширения и собственного имени, разделяемых точкой. При этом знак «?» означает один неизвестный символ в имени, «*» - неопределенное количество символов или их отсутствие.

Таким образом, имя файла, согласно маске, состоит из любого количества символов, вторым из которых является буква «о». К этой маске подходят варианты 1) и 3).

Ответ: 1) и 3)

Задача 7.

На рисунке обозначены маршрутизаторы, каждый из которых имеет свой IP-адрес. Расставьте их так, чтобы получился маршрут от узла 10.10.1.2.



Решение.

Способ IP - адресации. В 4-ой версии IP-адрес представляет собой 32-битовое двоичное число. Удобной формой записи IP-адреса является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками, например, 192.168.0.1. (или 128.10.2.30 — традиционная десятичная форма представления адреса). Номер узла в протоколе IP назначается независимо от локального адреса узла. Маршрутизатор по определению входит сразу в несколько сетей. Поэтому каждый порт маршрутизатора имеет собственный IP-адрес. Конечный узел также может входить в несколько IP-сетей. В этом случае компьютер должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

В сети Интернет обычно первый и второй байты определяют адрес сети, третий байт определяет адрес подсети, а четвертый - адрес компьютера в подсети.

Так как узел 10.10.1.2 является вторым компьютером в сети 10.10 и подсети 1, то только маршрутизатор А подключается в данную подсеть через узел 10.10.1.254. Второй адрес А - 10.10.0.87, поэтому он подсоединится к маршрутизатору В с адресом 10.10.0.1. Второй адрес В - 87.245.155.2, поэтому он подключается к Б с адресом 87.245.155.1, который, в свою очередь, через адрес 87.245.141.120 подключается к маршрутизатору Г с тем же адресом. Таким образом, правильный ответ: А, В, Б, Г.

Ответ: А, В, Б, Г.

Задача 8.
Дан фрагмент электронной таблицы в режиме отображения формул. Содержимое ячейки B2 было скопировано в ячейку B3.

	A	B
2	7	=A1*B1+B$1
3	2	

После этого фрагмент электронной таблицы в режиме отображения результатов вычислений стал иметь вид:

	A	B
2	7	34
3	2	239

Определите числовое значение в ячейке A1.
Решение.
При копировании формулы из ячейки B2 в ячейку B3 формула = A1*B1+B$1 будет иметь следующий вид:
= A2*B2+B$1
Обозначим A1=X и B1=Y. Составим систему уравнений для нахождения X и Y:
X*Y+Y=34, 7*34+Y=239
Решим систему относительно неизвестных:
Y=239-7*34=239-238=1;
X=34- Y=34-1=33.
Таким образом, в ячейке A1 записано число 33.
Ответ: 33
Задача 9.
База данных "Микрорайон", наряду с другими, имеет поля с названиями "тип дома" и "этажность". В базе данных находятся 38 записей о панельных и кирпичных домах высотой в 9, 12 и 16 этажей. Количество записей N, удовлетворяющих различным запросам, приведено в следующей таблице:
Определите количество записей, удовлетворяющих запросу "этажность^12".

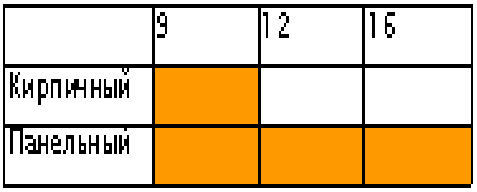
ЗАПРОС	N
этажность=9 или тип дома=панельный	23
неверно, что (этажность=12 и тип дома=панельный)	31
этажность=16 и тип дома=кирпичный	9

Решение.

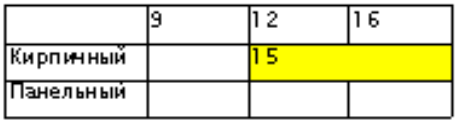
Нарисуем таблицу, в ячейки которой занесем количество домов каждого типа и этажности. Необходимо найти сумму чисел в выделенных ячейках.

1) этажность=9 или тип дома=панельный

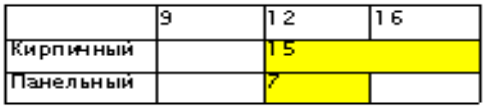
Запрос покрывает ячейки, выделенные на следующем рисунке, и его результатом является сумма значений в этих ячейках:



Найдем количество кирпичных 12-этажных и 16-этажных домов в микрорайоне: 38 - 23 = 15



2) неверно, что (этажность=12 и тип дома=панельный). Результат этого запроса - количество всех домов в микрорайоне, кроме 12-этажных панельных. Вычтем результат из общего их количества и получим количество таких домов: 38 - 31 = 7



3) этажность=16 и тип дома=кирпичный. Мы знаем, что кирпичных 12-этажных и 16-этажных домов всего 15. Благодаря результату этого запроса, мы можем определить количество кирпичных домов каждой этажности:

15 - 9 = 6

	9	12	16
Кирпичный		6	
Панельный		7	

Общий результат запроса: 38 - 6 - 7 = 25

Ответ: 25

Задача 10.

Два узла, находящиеся в одной сети, имеют IP-адреса 118.222.130.140 и 118.222.201.140. Укажите наибольшее возможное значение третьего слева байта маски сети. Ответ запишите в виде десятичного числа.

Решение.

Первые два числа обоих адресов, 118.222, одинаковые, поэтому возможно, что оба эти числа относятся к адресу сети (или третий байт маски равен нулю).

В третьем байте числа адреса различаются (130 и 201), поэтому третье число не может относиться к адресу сети целиком.

Чтобы определить возможную границу «зоны единиц» в маске, переводим числа 130 и 201 в двоичную систему счисления и представим в 8-битном коде:

$130 = 128 + 2 = \mathbf{1}0000010_2$

$201 = 128 + 64 + 8 = \mathbf{1}1001000_2$

В двоичном представлении обоих чисел выделяем одинаковые биты слева – совпадает всего один бит; поэтому в маске единичным может быть только один старший бит.

Таким образом, максимальное значение третьего байта маски – $10000000_2 = 128$

Ответ: 128.

Лекция (по материалам презентации).

Тема: Сетевые технологии: задачи IP.

Введение.

Основные понятия. Адресация в Интернете.

Необходимые условия обмена информацией:

1. Канал связи для передачи данных;
2. Аппаратные средства (модем, сетевая карта);
3. Программные средства (браузер, почта, менеджер обмена);
4. Протокол (соглашение процедуры обмена).

В Интернете используют протокол **TCP/IP (Transmission Control Protocol/Internet Protocol - Протокол управления передачей/Протокол объединения сетей), или комплект протоколов, из которых TCP/IP два основных протокола.**

Согласно TCP/IP каждый компьютер имеет свой уникальный IP-адрес, состоящий из 4 чисел (каждое от 0 до 255), разделенный точками (например, 172.16.1.95). Для хранения IP-адреса, таким образом, требуется 8 бит или 1 байт.

Формально для использования Интернета могут подключаться не более 256^4 компьютеров (около 4 миллиардов), но специальные технологии позволяют использовать групповые подключения компьютеров через общий IP-адрес.

Имеются специальные компьютера с информацией общего доступа, которые для удобства имеют доменные адреса в буквах (например, www.mail.ru). IP-адрес сервера по доменному адресу выдается службой **DNS (Domain Name System – система доменных имен).**

Доменные адреса выбираются по двум принципам:

1. По территориальной принадлежности (например, sch239.spb.ru)
2. По принадлежности к организации (например, greenpeace.org)

Для указания местоположения какого-нибудь ресурса (файла) в сети Интернет используют уникальный указатель местоположения ресурса **URL (Uniform Resource Locator**). Он состоит из трех частей:

1) Имени протокола, по которому происходит передача (http, ftp).
2) Через символы «://» адрес компьютера хранения файла (доменный или IP-адрес).
3) Через символ «/» имя файла, иногда с указанием его пути к нему. Например: http://masha.ru/risonok.jpg.

   Поиск информации в сети. Используют поисковые системы, обычно по ключевым словам или предложениям. Для уменьшения поиска страниц используют логическую связку И (знак амперсанда &), для увеличения глубины поиска – ИЛИ (знак вертикальная черта |).

   Маскирование сети Интернет.

   Маска сети - побитовый логический множитель IP-адреса, содержащий единицы в старших разрядах и нули в младших. Единичные разряды соответствуют адресу сети, нулевые - адресу компьютера. При логическом умножении IP-адреса на маску получаем адрес сети, при логическом умножении IP-адреса на логическое отрицание маски получаем адрес машины. Маска необходима для того, чтобы сообщить устройствам, в какой части адреса содержится номер сети, а в какой - номер хост-компьютера.

Требования к маске:

1. В маске сначала идет цепочка единиц, а потом до конца – цепочка нулей;

2. Число, где цепочка единиц начинается не с левого края (не со старшего, 8-ого бита) или внутри встречаются нули, не может быть маской;
3. Всего есть несколько допустимых чисел для последней части маски (все предыдущие должны быть равны 255):

10000000_2 = 128, 11000000_2 = 192
11100000_2 = 224, 11110000_2 = 240
11111000_2 = 248, 11111100_2 = 252
11111110_2 = 254, 11111111_2 = 255

Пример маски: 255.255.252.0

Для выполнения заданий в ЕГЭ в части сетевых технологий проверяемые
элементы предполагают проверку знание базовых принципов организации и функционирования компьютерных сетей, адресации в сети.

Рассмотрим задачи ЕГЭ, вызывающие наибольшие затруднения при решении.

Задача 1.

В терминологии сетей TCP/IP маской подсети называется 32-разрядное двоичное число, определяющее, какие именно разряды IP-адреса компьютера являются общими для всей подсети - в этих разрядах маски стоит 1. Обычно маски записываются в виде четверки десятичных чисел - по тем же правилам, что и IP-адреса. Для некоторой подсети используется маска 255.255.252.0. Сколько различных адресов компьютеров допускает эта маска?

Примечание. На практике два из возможных адресов не используются для адресации узлов сети: адрес сети, в котором все биты, отсекаемые маской, равны 0, и широковещательный адрес, в котором все эти биты равны 1.

Решение.

Переводим маску 255.255.252.0 в двоичную систему счисления: 11111111.111111.11111100.00000000. Адреса компьютеров с нулевыми битами образуют пространство, равное 2^{10}, т. е. 1024. Поскольку 2 из них не используют (адрес сети и широковещательный адрес), то получаем 10324-2= 1022.

Ответ: 1022 компьютера.

Задача 2.

Если маска подсети 255.255.255.240 и IP-адрес компьютера в сети 162.198.0.44, то чему равен порядковый номер компьютера в сети?

Решение.

Задача аналогична предыдущей, но требуется определить не номер сети, а номер компьютера (узла) в этой сети:

- Первые три числа в маске равны 255, в двоичной системе это 8 единиц, поэтому первые три числа IP-адреса компьютера целиком относятся к номеру сети.
- Для последнего числа (октета) маска и соответствующая ей последняя часть IP-адреса равны $240 = 11110000_2$ и $44 = 00101100_2$
- Нулевые биты маски и соответствующие им биты IP-адреса определяют номер компьютера в сети: $1100_2 = 12$

Ответ: 12

Задача 3.

Если маска подсети 255.255.240.0 и IP-адрес компьютера в сети 162.198.75.44, то чему равен полный номер узла в сети?

Решение.

Первые два числа в маске равны 255, в двоичной системе это 8 единиц, поэтому первые два числа IP-адреса компьютера целиком относятся к номеру сети, в этой задаче они не используются. Последнее число в маске – 0, поэтому последнее число IP-адреса целиком относится к номеру узла. Третье число маски – $240 = 11110000_2$, это значит, что первые 4 бита третьей части адреса (75) относятся к адресу сети, а последние 4 бита – к номеру узла:

$240 = 11110000_2$ и $75 = 01001011_2$

Нулевые биты маски и соответствующие им биты IP-адреса, определяющие старшую часть номера компьютера в сети: $1011_2 = 11$. Кроме того, нужно учесть еще и последнее число IP-адреса ($44 = 00101100_2$), таким образом, полный номер компьютера (узла) в двоичной и десятичной системах имеет вид: $1011.00101100_2 = 11.44$

Для получения полного номера узла нужно перевести число 101100101100_2 в десятичную систему: $101100101100_2 = 2860$

Ответ: 2860.

Практические задачи, выполняемые на занятии.

Решение данных задач даются в приложении 1.

Задача 1.

В терминологии TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса относится к адресу сети, а какая — к адресу самого адреса узла в этой сети. Маска записывается по тем же правилам, что и IP-адрес. При этом адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу сети и маске.

По заданным IP-адресу узла и маске определите адрес сети, если IP-адрес узла 217.233.232.3, маска имеет адрес 255.255.252.0.

Задача 2.

Даны числовые записи:

а) 164.24.45.305
б) 192.168.255.255
в) 102.255.256.254
г) 222.255.255.254
д) 127.0.0.1

Какие записи являются недопустимыми для IP — адресов и почему?

Задача 3.

Записать доменное имя компьютера, зарегистрированного в домене верхнего уровня RU, домене второго уровня schools и имеющего собственное имя www.

Задача 4.

Идентификатор некоторого ресурса сети Интернет имеет следующий вид: ftp://home.net/www.doc.

Какая часть этого идентификатора является именем сервера, на котором расположен ресурс?

Задача 5.

Идентификатор некоторого ресурса сети Интернет имеет следующий вид: http://www.ftp.ru/index.html.

Какая часть этого идентификатора указывает на протокол, используемый для передачи ресурса?

Задача 6.

На месте преступления были обнаружены четыре обрывка бумаги. Следствие установило, что на них записаны фрагменты одного IP-адреса. Криминалисты обозначили эти фрагменты буквами А, Б, В и Г. Восстановите IP-адрес.

.62	18	4.2	26.73
А	**Б**	**В**	**Г**

Задача 7.

Доступ к файлу htm.net, находящемуся на сервере com.edu, осуществляется по протоколу ftp. В таблице фрагменты адреса файла закодированы буквами от А до Ж. Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет.

А	/
Б	.com
В	.edu
Г	://
Д	.net
Е	htm
Ж	ftp

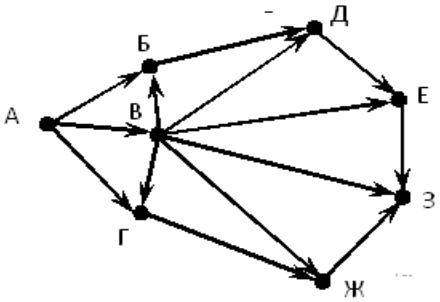
Практическое занятие 4. Информационные модели

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 4.

Домашние задания к занятию 5.

Задача 1.

На рисунке — схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город З?



Решение.

Начнем считать количество путей с конца маршрута – с города З.

Обозначим: N_X — количество различных путей из города А в город X, N — общее число путей.

В "З" можно приехать из В, Ж, или Е, поэтому составляем уравнение путей: $N = N_З = N_Е + N_В + N_Ж$

Аналогично:

$N_Е = N_Д + N_В$; $N_В = N_А$; $N_Ж = N_В + N_Г$.

Добавим еще вершины:

$N_Д = N_Б + N_В$; $N_Б = N_А + N_В = 1$; $N_Г = N_А + N_В = 1$;

Преобразуем вершины:

$N_Е = N_Д + N_В = 3 + 1 = 4$; $N_В = N_А = 1$; $N_Ж = N_В + N_Г = 1 + 2 = 3$. $N_Д = N_Б + N_В = 2 + 1 = 3$; $N_Б = N_А + N_В = 1 + 1 = 2$;

$N_Г = N_А + N_В = 1 + 1 = 2$.

Подставим в формулу уравнения путей:

$N = N_К = 4 + 1 + 3 = 8$.

Ответ: 8 путей

Задача 2.

Путешественник пришел в 08:00 на автостанцию населенного пункта ЛИСЬЕ и обнаружил там следующее расписание автобусов. Укажите самое раннее время, когда он сможете попасть в пункт ЗАЙЦЕВО по этому расписанию?

Пункт отправления	Пункт прибытия	Время отправления	Время прибытия
ЛИСЬЕ	ЗАЙЦЕВО	07:50	9:05
СОБОЛЕВО	ЛИСЬЕ	08:55	10:05
ЕЖОВО	ЛИСЬЕ	09:05	10:15
ЗАЙЦЕВО	ЕЖОВО	10:00	11:10
ЛИСЬЕ	СОБОЛЕВО	10:15	11:30
ЛИСЬЕ	ЕЖОВО	10:45	12:00
ЗАЙЦЕВО	ЛИСЬЕ	11:05	12:15
СОБОЛЕВО	ЗАЙЦЕВО	11:10	12:25
ЕЖОВО	ЗАЙЦЕВО	12:15	13:25

Решение.

Из Лисье после 8.00 отправляются следующие автобусы:

1) На Соболево в 10.15 – прибытие в 11.30.
2) На Ежово в 10.45 – прибытие в 12.00

В свою очередь, из Соболево на Зайцево отправится автобус в 11.10 и прибудет в 12.25, из Ежово на Зайцево отправится автобус в 12.15 и прибудет в 13.25.

Таким образом, путешественнику остается выбрать только вариант: Лисье - Ежово в 10.45 – прибытие в 12.00, Ежово на Зайцево - в 12.15, прибытие в 13.25

Ответ: 13.25.

Задача 3.

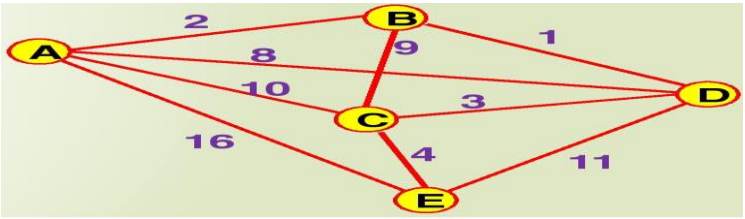
В таблице представлено расстояние между населенными пунктами. Определить кратчайшее расстояние между пунктами A и E.

	A	B	C	D	E
A		2	10	8	16
B	2		9	1	
C	10	9		3	4
D	8	1	3		11
E	16		4	11	

Решение.

Шаг 1. Анализ таблицы: данная таблица представляет собой весовую матрицу, у которой части таблицы, разделённые диагональю – симметричны, т.е. содержат одни и те же данные. Следовательно, можно рассматривать данные любой половины таблицы, разделенной диагональю.

Шаг 2. Построение графа A B C D E - A 2 10 8 16 B 2 9 1 C 10 9 3 4 D 8 1 3 11 E 16 4 11:



Шаг 3. Проверка правильности построения графа A B C E D 2 9 8 10 16 11 3 1 4 A B C D E A 2 10 8 16 B 2 9 1 C 10 9 3 4 D 8 1 3 11 E 16 4 11

Шаг 4. Определение всех путей в графе и расстояний, пройденных на этом пути (вес - расстояние в км.). A B C E D 2 9 8 10 16 11 3 1 4: осуществить обход по графу в алфавитном порядке, т.е. сначала все пути через AB, AC, AD и т.д. 1.ABCDE – 25 км 2.ABCE – 15 км 3.ABDCE – 10 км 4.ACBDE – 31 км 5.ACDE – 24 км 6.ACE – 14 км 7.ADCE – 15 км 8.ADE – 19 км 9.AE – 16 км

Шаг 5. Определяем кратчайший путь по графу: ABDCE – 10 км A B C E D 2 9 8 10 16 11 3 1 4 A B C D E A 2 10 8 16 B 2 9 1 C 10 9 3 4 D 8 1 3 11 E 16 4 11

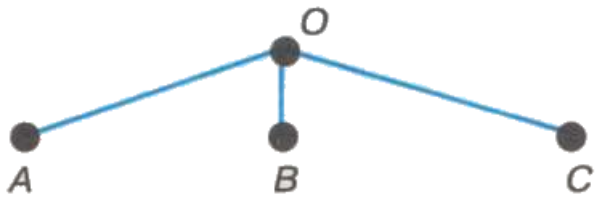
Ответ: 10 км.

Задача 4.

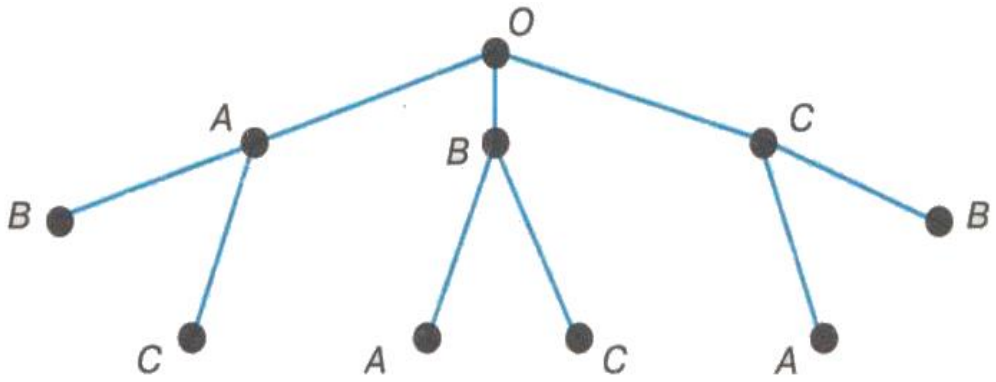
Сколькими способами можно рассадить в ряд на три стула трех учеников? Нарисовать граф решений, по которому определить количество способов.

Решение.

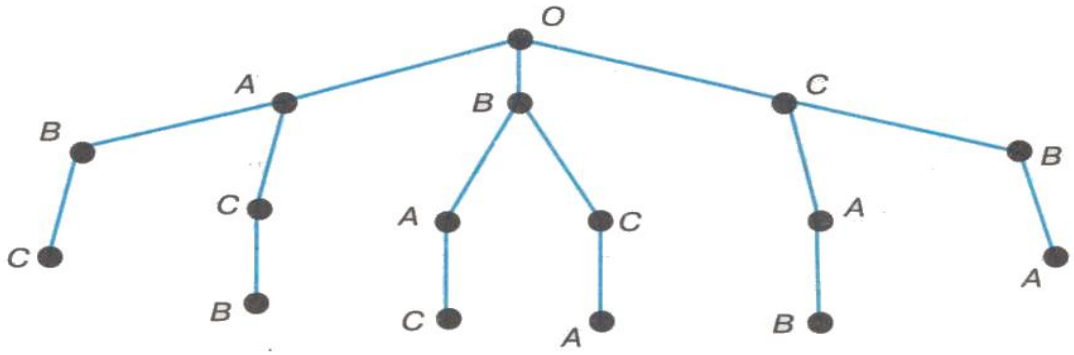
Решение этой задачи удобнее всего представить в виде дерева. За его корневую вершину возьмём произвольную точку плоскости O. На первый стул можно посадить любого из трех учеников — обозначим их A, B и C. На схеме это соответствует трём ветвям, исходящим из точки O:



Посадив на первый стул ученика A, на второй стул можно посадить ученика B или C. Если же на первый стул сядет ученик B, то на второй можно посадить A или C. Если на первый стул сядет C, то на второй можно будет посадить A или B. Это соответствует на схеме двум ветвям, исходящим из каждой вершины первого уровня:



Очевидно, что третий стул в каждом случае займёт оставшийся ученик. Это соответствует одной ветви дерева, которая «вырастает» на каждой из предыдущих ветвей:



Очевидно, что существует 6 способов посадки учеников.

Ответ: 6.

Задача 5.

Результаты тестирования учеников представлены в таблице. Сколько записей в ней должны удовлетворять условию: «Пол = “ж” ИЛИ Физика = Биология»?

Фамилия	Пол	Математика	История	Физика	Химия	Биология
Андреев	м	80	72	68	66	70
Борисов	м	75	88	69	61	69
Васильева	ж	85	77	73	79	74
Дмитриев	м	77	85	81	81	80
Егорова	ж	88	75	85	85	75
Захарова	ж	72	80	70	70	70

Решение.
Задача относится к логическому поиску информации в базах данных. Осуществляем последовательно отбор вариантов по условию логического сложения «Пол = “ж” ИЛИ Физика = Биология»: все женщины (3 варианта) + мужчины, у которых баллы по физике равны баллам по биологии (Борисов), таким образом, всего 4 варианта.
Ответ: 4.
Лекция (по материалам презентации)
Тема: Информационные модели.
Введение.
Информационная модель - система сигналов, свидетельствующих о динамике объекта управления, условиях внешней среды и состоянии самой системы управления.
Примеры информационных моделей:

- наглядные изображения (фото, кино, видео),
- знаки (текст, знаковое табло),
- графические модели (график, чертеж, блок - схема),
- комбинированные изображения (мнемосхема, карта).

Граф (в информатике) - способ определения отношений в совокупности элементов.
Отношения элементов являются основными объектами изучения **теории графов.**
Граф в информатике включает множество объектов, называемых вершинами или узлами, некоторые пары которых связаны т. н. ребрами. Например, граф на рисунке 3 состоит из четырех узлов, обозначенных A, B, C, и D, из которых B соединен с каждой из трех других вершин ребрами, а C и D также соединены. Два узла являются соседними, если они соединены ребром. На рисунке показан типичный способ того, как строить графы по информатике. Круги представляют вершины, а линии, соединяющие каждую их пару, являются ребрами.

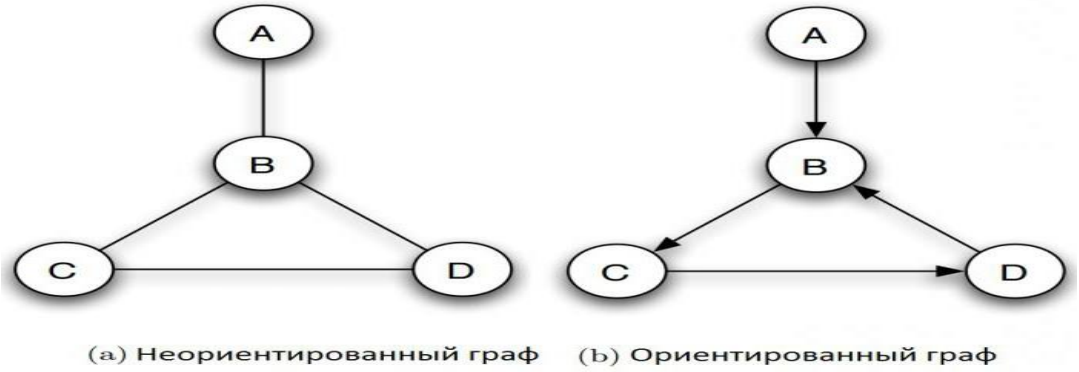


Рисунок 3 – Обозначение графов в информатике

В задачах ЕГЭ на анализ информационных моделей (ИМ) обычно используют таблицы и схемы, для которых умения понимать их построения является достаточным для решения данного типа задач. Уровень сложности таких задач является базовым, в которых проверяется, прежде всего, умение представлять и считывать данные в разных типах информационных моделей (схемах, картах, таблицах, графиках и формулах).

Принцип построения ИМ в таблице: на пересечении строки и столбца находится информация, характеризующая комбинацию данной строки и столбца.

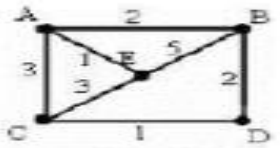
Принцип построения ИМ на схеме: если между объектами схемы имеется связь, то она отображается линией, соединяющей названия этих объектов на схеме.

Для примера рассматривается следующая задача.

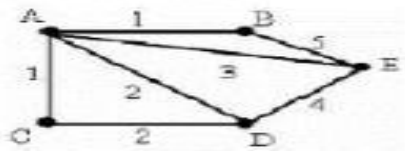
Задача 1. В таблице приведена стоимость перевозок между пятью железнодорожными станциями, обозначенными буквами A, B, C, D и E. Укажите схему, соответствующую таблице.

	A	B	C	D	E
A		1	1	2	3
B	1				5
C	1			2	
D	2		2		4
E	3	5		4	

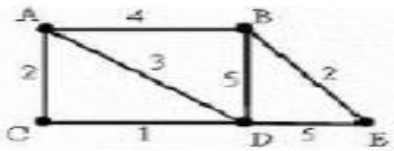
1)



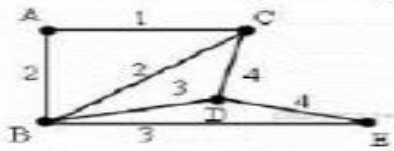
2)



3)



4)



Рекомендации к решению данных задач:

Если таблица большая и плотно заполнена числами, то для решения такого рода задач требуется довольно сложный алгоритм, например, **алгоритм Дейкстры.** Если таблица небольшая и не все ее клетки заполнены, то используют перебор возможных вариантов.

Алгоритм Дейкстры находит все кратчайшие пути из одной изначально заданной вершины графа до всех остальных. С его помощью, при наличии всей необходимой информации, можно, например, узнать какую последовательность дорог лучше

использовать, чтобы добраться из одного города до каждого из многих других, или в какие страны выгодней экспортировать нефть и тому подобное. Минусом данного метода является невозможность обработки графов, в которых имеются ребра с отрицательным весом, т. е. если, например, некоторая система предусматривает убыточные для Вашей фирмы маршруты, то для работы с ней следует воспользоваться отличным от алгоритма Дейкстры методом.

В данной задаче используют метод перебора возможных вариантов. Берем точку А, которая по таблице имеет расстояние с В, равное 1. Таким условием удовлетворяет вариант 2). Проверяем следующие соотношения расстояний в данном варианте: AC -1, AD - 2, AE -3; BE -5; CD - 2; DE – 4. Таким образом, правильный вариант 2).

Задача 2.

Таблица стоимости перевозок устроена следующим образом: числа, стоящие на пересечениях строк и столбцов таблиц, означают стоимость проезда между соответствующими соседними станциями. Если пересечение строки и столбца пусто, то станции не являются соседними. Укажите таблицу, согласно рисунку, для которой выполняется условие: “Минимальная стоимость проезда из А в В не больше 6”. Стоимость проезда по маршруту складывается из стоимостей проезда между соответствующими соседними станциями.

1)

	A	B	C	D	E
A			3	1	
B			4		2
C	3	4			2
D	1				
E		2	2		

2)

	A	B	C	D	E
A			3	1	1
B			4		
C	3	4			2
D	1				
E	1		2		

3)

	A	B	C	D	E
A			3	1	
B			4		1
C	3	4			2
D	1				
E		1	2		

4)

	A	B	C	D	E
A				1	
B			4		1
C		4		4	2
D	1		4		
E		1	2		

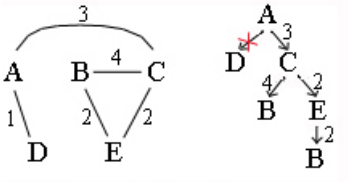
Рекомендации для эффективного решения:

1) Вариант пути, который становится больше или равен 6, но еще не достигает конечной точки (В), можно дальше не рассматривать.
2) Возвращение в точку, через которую уже прошел путь (петля) не приводит к эффективному решению, поэтому такие ветки тоже можно не рассматривать.
3) Если количество ветвей, входящих в конечную точку В меньше, чем количество путей, выходящих из стартовой точки А, то решение задачи предпочтительнее начинать с конца (из В в А).

Решение.

Шаг 1. Нарисуем схемы, соответствующие таблицам и определим по ним возможные пути из A в B.

Таблица 1: Из станции A через пункт D нельзя попасть в B, поэтому на рисунке путь A→D перечеркнут красным крестиком:



Всего, таким образом, получается 2 пути: A→C→B=3+4=7; A→C→E→B=3+2+2=7. Наименьший путь равен 7.

Таблица 2: Всего 2 пути: A→E→C→B=1+2+4=7; A→C→B=3+4=7. Наименьший путь равен 7.

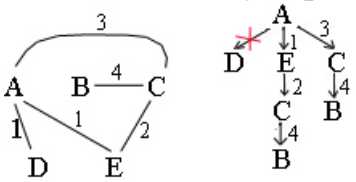


Таблица 3: Всего 2 пути: A→C→B=3+4=7; A→C→E→B=3+2+1=6. Наименьший путь равен 6.

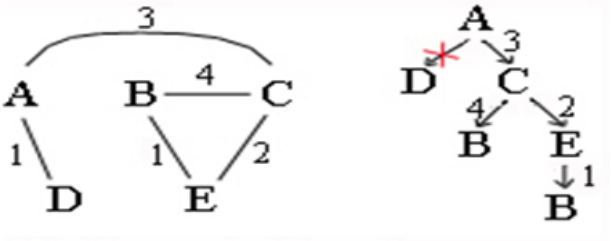
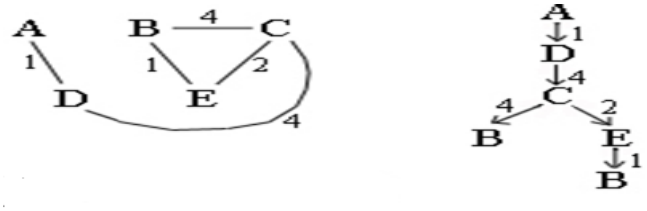


Таблица 4: Всего 2 пути: A→D→C→B=1+4+4=9; A→D→C→E→B=1+4+2+1=8.



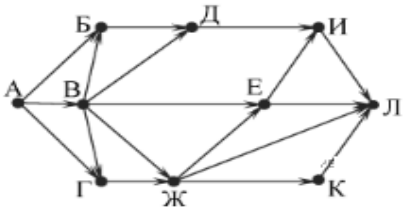
Наименьший путь равен 8.

Поэтому условие “Минимальная стоимость проезда из A в B не больше 6” выполняется только для таблицы 3 (варианта 3).

Ответ: вариант 3).

Задача 3.

На рисунке — схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, И, К, Л. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город Л?



Решение.

Алгоритм решения данного типа задач заключается в следующем:

Шаг 1. Вывод общего уравнения путей.

Начнем считать количество путей с конца маршрута – с города Л.

N_X — количество различных путей из города А в город Х, N — общее число путей.

В "Л" можно приехать из И, Е, Ж или К, поэтому:

$N = N_Л = N_И + N_Е + N_Ж + N_К$ (*)

Шаг 2.

Производим вычисления:

$N_И = N_Д + N_Е = 3 + 4 = 7$; $N_Е = N_В + N_Ж = 1 + 3 = 4$; $N_Ж = N_В + N_Г = 1 + 2 = 3$;

$N_К = N_Ж = 3$.

Шаг 3.

Добавим вершины в графе:

$N_Д = N_Б + N_В = 2 + 1 = 3$; $N_В = N_А = 1$;$N_Г = N_А + N_В = 1 + 1 = 2$;

$N_Б = N_А + N_В = 1 + 1 = 2$.

Шаг 4.

Производим вычисления по формуле (*):

$N = N_Л = 7 + 4 + 3 + 3 = 17$.

Ответ: 17 путей.

Практические задачи, выполняемые на занятии.

Решение данных задач даются в приложении 1.

Задача 1.

Грунтовая дорога проходит последовательно через населенные пункты A, B, C и D. При этом длина дороги между A и B равна 80 км, между B и C – 50 км, и между C и D – 10 км. Между A и C построили новое асфальтовое шоссе длиной 40 км. Оцените минимально возможное время движения велосипедиста из пункта A в пункт B, если его скорость по грунтовой дороге – 20 км/час, по шоссе – 40 км/час.

Задача 2.

Дан фрагмент расписания перелетов между аэропортами. Если вы оказались в аэропорту Синее в полночь (00:00), то укажите самое

раннее время, когда вы сможете попасть в аэропорт Остров? Варианты ответов: 1) 12:10 2) 14:30 3) 16:45 4) 20:45

Аэропорт вылета	Аэропорт прилета	Время вылета	Время прилета
НОЯБРЬ	СИНЕЕ	07:30	9:50
ОСТРОВ	НОЯБРЬ	08:15	10:35
ЕЛКИНО	СИНЕЕ	11:35	13:25
СИНЕЕ	НОЯБРЬ	12:10	14:30
НОЯБРЬ	ОСТРОВ	12:30	14:30
ОСТРОВ	ЕЛКИНО	14:10	16:20
НОЯБРЬ	ЕЛКИНО	15:15	16:45
СИНЕЕ	ЕЛКИНО	14:20	16:30
ЕЛКИНО	НОЯБРЬ	17:40	19:10
ЕЛКИНО	ОСТРОВ	18:40	20:45

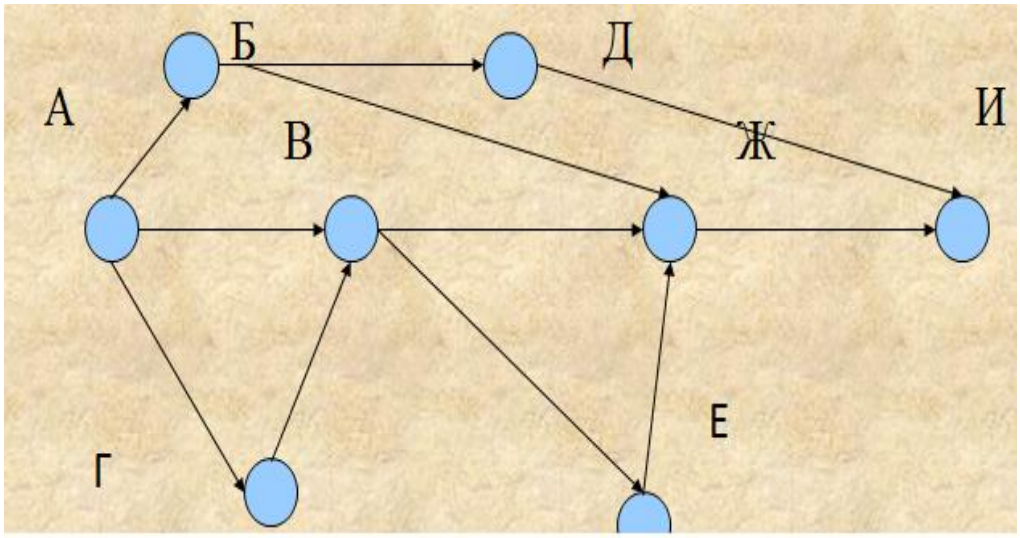
Задача 3.

Между населенными пунктами A, B, C, D, E, F построены дороги, протяженность которых приведена в таблице (Отсутствие числа в таблице означает, что прямой дороги между пунктами нет). Определите длину кратчайшего пути между пунктами A и F (при условии, что передвигаться можно только по построенным дорогам). Варианты ответов: 1) 21 2) 22 3) 23 4) 33

	A	B	C	D	E	F
A		7				
B	7		12	7	12	
C		12			10	
D		7			4	
E		12	10	4		
F					4	

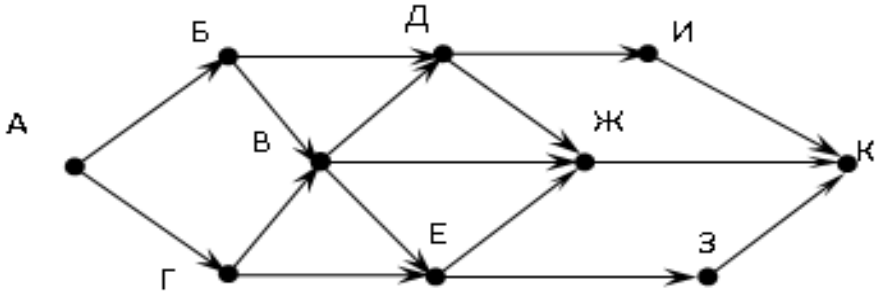
Задача 4.

На рисунке — схема дорог, связывающая города А, Б, В, Д, Е, Ж, И. По каждой дороге можно двигаться только в одном направлении, указанной стрелкой. Сколько существует различных путей из города А в город И?



Задача 5.

На рисунке схема дорог, связывающая города А, Б, В, Г, Д, Е, Ж, З, И, К. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город К?



Практическое занятие 5. Задачи по комбинаторике

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 5.

Домашние задания к занятию 6.

Задача 1.

Сколько существует пятизначных чисел, которые читаются одинаково «слева направо» и «справа налево»?

Решение.

Представим пятизначный палиндром в виде: XYZZYX.

На месте X - могут стоять все числа от 1 до 9 (без 0), так как число пятизначное. На месте Y и Z любые числа от 0 до 9. Итого: 9*10*10=900

Ответ: 900 цифр.

Задача 2.

Сколько существует различных трехзначных чисел, в записи которых все цифры нечетные и хотя бы одна из них равна 3?

Решение.

Шаг 1. Все числа, состоящие только из нечетных цифр, можно разбить на две группы: те, в которых есть 3, и те, где ее нет.

Шаг 2. Общее число чисел, состоящих только из нечетных цифр, находим по таблице:

1 разряд	2 разряд	3 разряд
1,3,5,7,9 – 5 вариантов	1,3,5,7,9 – 5 вариантов	1,3,5,7,9 – 5 вариантов

Всего: 5·5·5 = 125 вариантов

Шаг 3. Аналогично найдем количество чисел, состоящих только из цифр 1, 5, 7 и 9 (без 3); так как на каждом из 3-х мест может стоять одна из 4-х цифр, получаем 4·4·4 = 32 варианта

Шаг 4. Число вариантов трехзначных чисел, в записи которых все цифры нечетные и хотя бы одна из них равна 3, находим по разности: 125-32=93 варианта.

Ответ: 93 варианта.

Задача 3.

Задача Леонарда Эйлера. Четверо господ при входе в ресторан отдали швейцару свои шляпы, а при выходе получили их обратно. Сколько существует вариантов, при которых каждый из них получит чужую шляпу?

Решение.

Обозначим гостей цифрами 1, 2, 3, 4 и так же обозначим их шляпы. Считаем, что шляпа с данным номером принадлежит гостю с этим же номером (то есть, например, шляпа 2 принадлежит гостю 2). Тогда

каждый вариант получения шляп обозначается четырехзначным числом, составленным из цифр 1, 2, 3 и 4, в котором номер позиции цифры есть номер гостя, а сама цифра есть номер полученной им шляпы (номера позиций будем считать слева направо). Например, комбинация 4132 означает, что первый гость получил четвертую шляпу, второй — первую, третий — третью, а четвертый — вторую. Такой вариант не годится по условию, поскольку третий получил свою шляпу. Таким образом, нужно выписать по возрастанию все четырехзначные числа, содержащие по одной цифре 1, 2, 3 и 4, причем такие, что никакая цифра не стоит на позиции со своим номером. Эти числа выписаны ниже в таблице:

1 гость	2 гость	3 гость	4 гость
2	1	4	3
2	3	4	1
2	4	1	3
3	1	4	2
3	4	1	2
3	4	2	1
4	1	2	3
4	3	1	2
4	3	2	1

По таблице видно, что всего имеется 9 вариантов нужной раздачи шляп.

Ответ: 9 вариантов

Задача 4.

Автомобильные номера состоят из трех букв (всего 30 букв) и четырех цифр (используется 10 цифр). Сколько автомобилей можно занумеровать таким способом, чтобы никакие два автомобили не имели одинаковые номера?

Решение.

Автомобильный номер, состоящий из 3 букв и 4 цифр, имеет букв – 30, цифр – 10. Необходимо вычислить количество возможных способов нумерования автомобилей. Так как автомобильный номер состоит одновременно и из букв, и из цифр, то нужно воспользоваться правилом произведения способов выбора 3 букв и 4 цифр.

Так как цифры и буквы берутся сразу не все, а только часть, важен порядок их выбора, и они могут в номере повторяться, то это размещения с повторениями:

$A^3_{30}=30^3=27000$; $A^4_{10}=10^4=10000$

По правилу произведения получаем: $A^3_{30} * A^4_{10} = 27000*10000=270\ 000000$.

Ответ: 270 000 000 способов.

Задача 5.

Алфавит племени Пиджен состоит из четырех букв. Аборигены закодировали слово CADC с использованием следующей кодовой таблицы:

A	B	C	D
1	0	0	0
	0	1	

и передали его, не сделав промежутков, отделяющих одну букву от другой. Чему равно количество способов прочтения переданного слова?

Решение.

Шаг 1. Запишем слово CADC в закодированном виде: CADC = 011001.

Определим функцию количества возможных способов прочтения строки N(011001). Слева направо будем пробовать расшифровать первую букву алфавита. Если возможна неоднозначная трактовка, будем разделять такие случаи и продолжать чтение букв для каждого варианта отдельно.

Под обозначением функции будем подписывать букву, полученную в этом способе прочтения последней:

- первую букву мы можем прочитать как D (011001) или C (011001)

D C

Соответственно: N (011001) = N (11001)+ N (1001)

D C

Первое слагаемое – количество способов прочтения в случае, если первая буква была расшифрована как D. Второе слагаемое – количество способов прочтения, если первая буква была расшифрована как C. Очевидно, что общее количество способов прочтения – это сумма выделенных слагаемых. **Шаг 2.** Продолжим последовательное чтение. Рассматриваем строку символов большей длины, т.к. в результате мы можем получить функцию от меньшей строки и привести подобные слагаемые для упрощения дальнейшего подсчета.

Если первой прочитана буква D как 11001, то следующую букву можно определить однозначно: это буква A, и тогда:

N (11001)= N (1001)

D A

Это равно второму слагаемому суммы.
После приведения слагаемых получаем:
N (011001) = N (11001)+ N(1001) = N (1001)+ N (1001)= 2 N (1001)
D C A C
Шаг 4. Следующая буква также может быть расшифрована однозначно. Это буква A: 1001 = A.
Количество вариантов прочтения не изменяется: 2 N (1001) =2 N (001)
A
Следующей буквой может быть как D= 001, так и B=001. Разделим эти случаи: 001 001
D B
N (001)= 2 (N (01)+ N (1))
A D B
Количество способов прочтения увеличивается.
Шаг 5. Расшифруем строку символов 01. Возможно 2 способа. Первый –
расшифровать букву D (01), второй – расшифровать букву C (01)
D *C*
Если мы расшифровываем последнюю букву, то полагаем, что существует один способ ее получения, остальные возможные способы мы учитываем в других слагаемых.
Снова разделим эти два случая и приведем подобные элементы.
2(*N* (01) + *N* (1))= 2(*N* (1) +1+ *N* (1))
D
Единица – способ прочтения 01 как C. *N*(1) также будет равно единице. Приведем подобные, заменим *N*(1) на 1 и посчитаем количество способов расшифровки переданного слова:
2(*N*(1)+ □1□ + *N*(1)) = 4*N*(1) □ +2 = 4□+ 2= 6
Ответ: 6
Лекция (по материалам презентации)
Тема: Задачи по комбинаторике.
Введение. Комбинаторика является одним из способов измерения информации.
Структурные меры измерения информации.
Учитывают только дискретное строение информации. Элементами информационного комплекса являются кванты - неделимые части информации. Различают геометрическую, комбинаторную и аддитивную меры. В комбинаторной мере количество информации вычисляется как количество комбинаций элементов. Здесь учитываются возможные или реализованные комбинации.

Комбинаторика (определение):

1) то же, что математический комбинаторный анализ (раздел математики, изучающий дискретные объекты, множества и их отношения, включающий теорию графов)**.**

2) Раздел элементарной математики, связанный с изучением количества комбинаций, подчиненных тем или иным условиям, которые можно составить из заданного конечного множества объектов (безразлично, какой природы; это могут быть буквы, цифры, какие-либо предметы и т.п.).

Типовые комбинаторные задачи сводятся к **расчету количества возможных вариантов.**

Что нужно знать:

1. Если мы разбили все нужные нам комбинации на несколько групп **(не имеющих общих элементов)** и подсчитали количество вариантов в каждой группе, то для вычисления общего количества вариантов нужно все эти числа сложить:

например, есть 9·10=90 трехзначных чисел, оканчивающихся на 5, и 9·10=90 трехзначных чисел, оканчивающихся на 2, поэтому 90+90=180 трехзначных чисел оканчиваются на 2 или на 5.

2. Если в предыдущем случае группы имеют общие элементы, их количество нужно вычесть из полученной суммы;
   например, есть 9·10=90 трехзначных чисел, оканчивающихся на 5, и 10·10=100 трехзначных чисел, начинающихся на 5; в обе группы входят числа, которые начинаются и заканчиваются на 5, их всего 10 штук, поэтому количество чисел, которые начинаются *или* заканчиваются на 5, равно 90+100-10=180.
3. Если есть n различных элементов, число их различных перестановок равно **факториалу числа n, то есть произведению всех натуральных чисел от 1 до n: n! = 1·2·3·…·(n-1)·n:**

например, три объекта (А, Б и В) можно переставить 6 способами (3!=1·2·3=6): (А, Б, В), (А, В, Б), (Б, А, В), (Б, В, А), (В, А, Б) и (В, Б, А).

4. Если нужно выбрать m элементов из n (где n>=m) и две комбинации, состоящие из одних и тех же элементов, расположенных в разном порядке, считаются различными, число таких комбинаций (они называются **размещениями)** равно:

$$A_n^m = \frac{n!}{(n-m)!} = (n-m+1)\cdot(n-m+2)\cdot\ldots\cdot(n-1)\cdot n$$

Например, в соревновании пяти спортсменов призовые места (первые три) могут распределиться 60 способами, поскольку:

$$A_5^3 = \frac{5!}{(5-3)!} = 3 \cdot 4 \cdot 5 = 60$$

5. Если нужно выбрать m элементов из n (где n>=m) и порядок их расположения не играет роли, число таких комбинаций (они называются **сочетаниями**) равно:

$$C_n^m = \frac{n!}{m!(n-m)!}$$

Например, выбрать двух дежурных из пяти человек можно 10 способами, поскольку:

$$C_5^2 = \frac{5!}{2!(5-2)!} = \frac{120}{2 \cdot 6} = 10$$

Практические задачи, выполняемые на занятии.

Решение данных задач даются в приложении 1.

Задача 1.

Сколько существует различных четырехзначных чисел, в записи которых используются только четные цифры?

Варианты ответов: 1) 125 2) 250 3) 500 4) 625

Задача 2.

Виктор хочет купить пять разных книг, но денег у него хватает только на три (любые) книги. Сколькими способами Виктор может выбрать три книги из пяти?

Варианты ответов: 1) 10 2) 20 3) 30 4) 60

Задача 3.

Сколько существует четырехзначных чисел, которые читаются одинаково «слева направо» и «справа налево»?

Варианты ответов: 1) 50 2) 90 3) 100 4) 120

Задача 4.

У Паши есть 6 воздушных шариков разного цвета. Три из них он хочет подарить Маше. Сколькими способами он может это сделать?

Задача 5.

У людоеда в подвале томятся 25 пленников.

а) Сколькими способами он может выбрать трех из них себе на завтрак, обед и ужин?

б) А сколько есть способов выбрать троих, чтобы отпустить на свободу?

Задача 6.

На склад завезли 17 серверов с различными дефектами, которые стоят в 2 раза дешевле нормальных серверов. Директор купил в школу 14

таких серверов, а сэкономленные деньги своровал и купил дочке шубу из меха соболя за 200 000 рублей. Сколькими способами директор может выбрать бракованные серверы?

Задача 7.

В корзине лежат 9 шаров и 12 черных. Сколькими способами можно достать из этой корзины 2 белых шара и 2 черных?

Задача 8.

В ларьке продаются 15 роз и 18 тюльпанов. Ученик хочет купить 3 цветка для своей одноклассницы, причем все цветы должны быть одинаковыми. Сколькими способами он может составить такой букет?

Задача 9.

В группе из 20 студентов, среди которых 2 отличника, надо выбрать 4 человека для участия в конференции. Сколькими способами можно выбрать этих четверых, если отличники обязательно должны попасть на конференцию?

Задача 10.

Какое наименьшее число символов должно быть в алфавите, чтобы при помощи всевозможных трехбуквенных слов, состоящих из символов данного алфавита, можно было передать не менее 9 различных сообщений?

Практическое занятие 6. Массивы

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 6.

Домашние задания к занятию 7.

Задача 1.

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать целые значения от 0 до 10000 включительно. Опишите на естественном языке алгоритм, позволяющий найти и вывести максимальное значение среди двузначных элементов массива, не делящихся на 3. Если в исходном массиве нет элемента, значение которого является двузначным числом и при этом не кратно трём, то выведите сообщение «Не найдено».

Исходные данные объявлены так, как показано ниже, на примерах естественного языка. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать некоторые из описанных переменных.

Объявляем массив A из 40 элементов.
Объявляем целочисленные переменные I, J, MAX.
В цикле от 1 до 40 вводим элементы массива A с 1-го по 40-й.
…

Решение.

Записываем в переменную MAX начальное значение, равное 9. В цикле от первого элемента до сорокового находим остаток от деления элемента исходного массива на 3. Если значение данного остатка не равно 0 и значение текущего элемента массива больше 9 и меньше 100, то сравниваем значение текущего элемента массива со значением переменной MAX. Если текущий элемент массива больше MAX, то записываем в MAX значение этого элемента массива. Переходим к следующему элементу.

После завершения цикла проверяем значение переменной MAX. Если оно больше 9, то выводим его, иначе выводим сообщение «Не найдено».

Задача 2.

Определить максимальное количество подряд идущих четных элементов в заданном целочисленном массиве.

Решение.

Примечание: для краткости будем называть участок массива с элементами, обладающими некоторыми заданными свойствами, — “подмассивом”. Для решения задачи определяем длину (количество элементов) каждого подмассива.

Алгоритм на естественном языке.

Вводим следующие обозначения: **кол —** число элементов в текущем подмассиве; **макс_кол —** искомое значение (максимальное количество подряд идущих элементов с заданными свойствами).

Если очередной обладает заданными свойствами, то подмассив таких элементов продолжается или начался — увеличиваем его длину **кол** на 1, иначе — текущий подмассив кончился, и в этом случае:

1) сравниваем его длину с максимальной длиной уже рассмотренных ранее подмассивов **макс_кол**. Если длина текущего подмассива больше, то принимаем ее в нового значение величины **макс_кол;**

2) имея в виду обработку следующего подмассива, обнуляем значение величины **кол.**

Однако после завершения цикла в случае, когда последний, *n*-й, элемент массива также обладает заданными свойствами (например, будет четным), последний подмассив не будет учтен, поэтому нужно дополнительно учесть и его, сравнив длину этого подмассива с максимальной длиной: проверяем длину последнего (возможного) подмассива: если **кол** > **макс_кол,** то **макс_кол** := кол.

Задача 3.

Переставить элементы заданного массива в обратном порядке, то есть произвести реверс массива.

Решение.

Пояснение и алгоритм.

При реверсе массива первый элемент становится последним, а последний первым; второй - предпоследним, а предпоследний - вторым; третий элемент уходит на место третьего с конца, а тот на место третьего и т. д. Таких пар перестановок надо сделать в два раза меньше, чем длина массива, то есть переставлять элементы до тех пор, пока меняющиеся элементы не встретятся в центре массива.

Можно переставлять элементы, начиная с середины массива и двигаясь к его началу и концу, но вышеописанный способ немного проще.

Если в массиве нечетное количество элементов, то в середине массива находится один элемент, у которого нет пары и который обменивать не надо. Если же в массиве четное количество элементов, то в середине находится пара, которая также должна обменяться. В любом случае количество обменов будет равно количеству элементов массива, нацело деленному на 2.

Таким образом, реверс массива происходит в цикле, количество итераций (проходов) которого равно не более половины от количества элементов. В теле цикла происходит обмен элементов. Если индексация (i) массива начинается с единицы, а количество элементов N, то индекс элемента, с которым должен происходить обмен будет находиться по формуле N-i+1. Если же индексация идет с нуля, то противоположный для i элемент находится как N-i-1.

Пример программы на языке Pascal:

```
const N = 10; var  a: array[1..N] of integer;
   i: byte;  b: integer;
begin
   for i:=1 to N do read(a[i]);     for i:=1 to N div 2 do begin
      b := a[i];  a[i] := a[N-i+1];  a[N-i+1] := b;
   end;
 for i:=1 to N do  write(a[i],' ');  writeln;
end.
```

Задача 4.

Заполнить одномерный массив случайными числами. Найти и вывести на экран наибольший его элемент и порядковый номер этого элемента.

Решение.

Пояснение и алгоритм.

Заполнение массива и поиск наибольшего элемента можно выполнять в одном цикле. Так как необходимо найти не только максимальный элемент, но и его индекс, то лучше искать индекс, так как по нему всегда можно получить значение из массива. При поиске можно сохранять и индекс, и элемент в двух разных переменных, но этого делать не обязательно. До цикла присвоим переменной, в которой будет храниться индекс максимального элемента, значение 1, предполагая, что максимальный элемент находится в первой ячейке массива.

Тело цикла будет состоять из следующих действий:

1. Сгенерировать случайное число и записать его в очередную ячейку массива.
2. Вывести полученное число на экран.
3. Если это число больше, чем то, что хранится под индексом, записанным в переменную-максимум, то присвоить этой переменной текущий индекс (не само число!).

После того, как индекс наибольшего элемента будет найден, вывести его на экран. Чтобы вывести элемент по данному индексу, нужно использовать выражение извлечения элемента из массива, например,

если max - это индекс, а arr - массив, то выражение будет таким: arr[max].

Пример программы на языке Pascal:

```
const N = 10; var    arr: array[1..N] of integer;
   i, max: byte;
begin
   randomize;  max := 1;
   for i:=1 to N do begin
      arr[i] := random(100);  write(arr[i], ' ');
      if arr[max] < arr[i] then max := i;
   end;
   writeln; writeln('arr[',max,'] = ',arr[max]);
end.
```

Задача 5.

Заполнить один массив случайными числами, другой - введенными с клавиатуры числами, в ячейки третьего записать суммы соответствующих ячеек первых двух. Вывести содержимое массивов на экран.

Решение.

Пояснение и алгоритм.

Все три массива должны иметь одинаковую размерность и тип.

Заполнение массива случайными числами:

Перебирать индекс массива от начала до конца, записывать в каждую его ячейку случайно сгенерированное число.

Заполнение массива значениями с клавиатуры:

Перебирать индекс массива от начала до конца, записывать в каждую его ячейку число, введенное с клавиатуры.

Заполнение массива суммами значений из других массивов:

Перебирать индекс массива от начала до конца, записывать в каждую его ячейку сумму значений из ячеек под таким же индексом из двух первых массивов.

Вывод массива на экран:

Перебирать индекс массива от начала до конца, выводить на экран значение ячейки массива под каждым индексом.

Пример программы на языке Pascal:

```
const N = 10; var a,b,c: array[1..N] of byte;
   i: byte;
begin
   randomize;
   for i:=1 to N do  a[i] := random(100);
    write('Введите десять чисел до 100: ');
```

```
    for i:=1 to N do  read(b[i]);
    for i:=1 to N do c[i] := a[i] + b[i];
     for i:=1 to N do  write(a[i]:4);
    writeln;
    for i:=1 to N do  write(b[i]:4);
    writeln;
    for i:=1 to N do write(c[i]:4);
    writeln;
end.
```

Лекция (по материалам презентации)

Тема: Массивы.

Введение.

Определения:

- **Массивы –** это упорядоченное множество значений одного типа.
- **Массивы –** это переменные с одним именем и разными индексами.

Словарь терминов (глоссарий)

- **Массивы –** это группа переменных с одним именем и различными индексами.
- Каждая переменная в массиве называется его **элементом.**
- **Атрибуты элемента –** местоположение внутри массива и значение.
- **Тип элементов массива –** числовой или текстовой.
- **Индексы элементов массива –** число, переменные целого типа, арифметические выражения.

Обозначение массива.

Математическая форма представления массива:

$a_1, a_2, a_3, \ldots a_i, \ldots, a_n,$

где a – имя элементов массива,

n – число элементов в массиве.

Кроме представленных одномерных массивов, существуют многомерные массивы.

Двумерный массив – это таблица, которая обозначается, как:

$a_{11}, a_{12}, a_{13}, \ldots a_{1i}, \ldots, a_{1n},$

$a_{21}, a_{22}, a_{23}, \ldots a_{2i}, \ldots, a_{2n},$

…….,

$a_{m1}, a_{m2}, a_{m3}, \ldots a_{mi}, \ldots, a_{mn},$

Например: a_{11}, a_{12}, a_{13} -индексы первой строки, a_{11}, a_{21}, a_{31} – индексы первого столбца

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}
a_{51}	a_{52}	a_{54}	a_{55}

В программировании обычно элемент массива обозначается именем переменной, в скобках которой указывается индекс:

Например: **a(1), a(2), …a(i), …a(n)** или **a(11), a(12), …a(1i), …a(1n)….. a(mn).**

Обозначение массива целочисленных переменных в языке программирования Pascal: a:array[1..n] of integer

Обозначение массива целочисленных переменных в языке программирования Basic: DIM a(n) INTEGER

Использование массивов в программировании:

Массивы (как индексированные переменные) используются вместе с циклами для ввода, сохранения и обработки группы значений (см. рисунок 4).

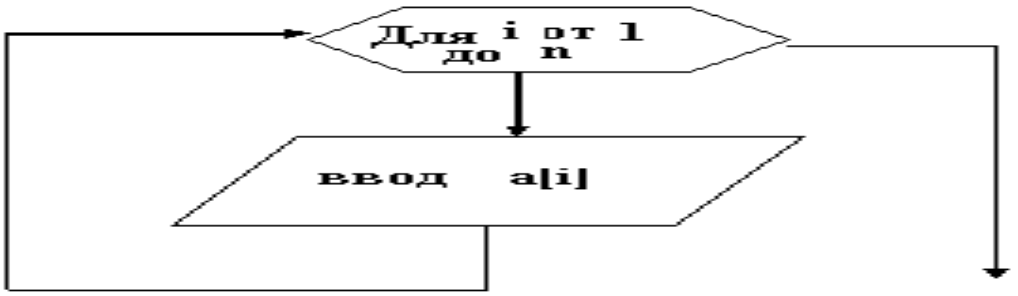


Рисунок 4 – Блок-схема цикла ввода элементов массива.

Типовые алгоритмические задачи с массивами

1. Нахождение суммы всех элементов массива.
2. Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию).
3. Нахождение количества элементов массива с заданными свойствами.
4. Нахождение среднего арифметического элементов массива с заданными свойствами.

5. Изменение значений элементов массива с заданными свойствами.
6. Вывод на экран элементов массива с заданными свойствами.
7. Нахождение номеров (индексов) элементов массива с заданными свойствами.
8. Определение индекса элемента массива, равному заданному числу.
9. Определение индекса элемента массива, равному заданному числу, для массива, отсортированному по возрастанию.
10. Определение максимального элемента в массиве.
11. Определение индекса максимального элемента в массиве.
12. Определение максимального значения среди элементов в массиве, удовлетворяющих некоторому условию.
13. Определение заданного числа в упорядоченном массиве.
14. Обмен местами двух элементов массива с заданными номерами.
15. Удаление из массива к-го элемента со сдвигом на одну позицию влево.
16. Вставка в массив к-го элемента со сдвигом на одну позицию вправо.
17. Циклические перемещения элементов в массиве.
18. Проверка массива на упорядоченность по не убыванию.
19. Проверка наличия в массиве одинаковых элементов.

Массивы и типовые задачи.

Работа с массивами - это задачи на заполнение, считывание, поиск, сортировка, массовые операции и др. Задачи этого типа вызывают наибольшее затруднение, их уровень сложности – **повышенный.**

Типы задач на массивы:

1) **Тип первый** - задание элементов одномерных массивов. Схема построения задачи: с помощью простого алгоритма задаются значения элементов первого массива. Значения элементов второго массива задаются как некоторое арифметическое выражение от значений первого массива. В задаче требуется вычислить значения элементов второго массива и ответить на вопрос. Поэтому разнообразие задач этого типа может быть большим.
2) **Тип второй** – задание элементов двумерного массива. Схема построения задачи: используется простой алгоритм задания значений элементов, чтобы получить элементы с разными знаками и нулевые. Вопрос в задаче ставится в зависимости от

значений элементов: выдать значение элемента с заданными индексами или сосчитать количество элементов с заданным знаком.

3) **Тип третий** – манипуляция с элементами двумерного массива.
4) **Тип четвертый** – оценка времени исполнения алгоритма.
5) **Тип пятый** – манипуляция с элементами одномерного массива

Задачи первого типа

Пример 1. Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы:

Бейсик	Паскаль	Алгоритмический
FOR n=1 TO 100 A(n)=n-10 NEXT n FOR n=1 TO 100 B(n)=A(n)*n NEXT n	for n:=1 to 100 do A[n]:=n-10; for n:=1 to 100 do B[n]:=A[n]*n	нц для n от 1 до 100 A[n]=n-10 кц нц для n от 1 до 100 B[n]=A[n]*n кц

Сколько элементов массива B будут иметь положительные значения?

Решение. Используем анализ алгоритма.

В каждом одномерном массиве A и B имеется по 100 элементов. Элементы массива B по знаку совпадают с элементами массива A, так как получены из элементов массива A путем умножения на целое положительное число n, которое является счетчиком цикла для задания элементов массива B. Поэтому нужно исследовать только элементы массива A. В нем только элементы с номерами от 1 до 9 принимают отрицательные значения от (- 9) до (-1), а элемент с номером 10 имеет нулевое значение. Всего 10 элементов массива A имеют отрицательные или нулевые значения. Поэтому, в массиве B количество положительных элементов (не отрицательных и не равных 0) равно 90.

Ответ: 90.

Пример 2. Значения двух массивов A[1..100] и B[1..100] задаются с помощью следующего фрагмента программы, какой элемент массива B будет наибольшим?

Бейсик	Паскаль	Алгоритмический
FOR n=1 TO 100 A(n)=(n-80)*(n-80) NEXT n FOR n=1 TO 100 B(101-n)=A(n) NEXT n	for n:=1 to 100 do A[n]:= (n-80)*(n-80); for n:=1 to 100 do B[101-n]:=A[n];	нц для n от 1 до 100 A[n]=(n-80)*(n-80) кц нц для n от 1 до 100 B[101-n]=A[n] кц

Решение. Анализ массива А.
Взять нужно характерные для этого массива точки при n=1, 80, 100. при этом получим: $A[1]=79^2$, далее значения элементов массива убывают до A[80]=0, а потом возрастают до $A[100]=20^2$. Очевидно, что в массиве А максимальный элемент находится на первом месте. Теперь в зависимости от закона формирования массива В найдем номер максимального элемента.
При n=1 получим B[100]=A[1], при n=100 получим B[1]=A[100], при n=80 получим B[21]=A[80]. Последних вычислений можно и не проводить, ответ найден, максимальный элемент стоит в массиве В на последнем месте.
Ответ: B[100]
Задачи второго типа.
Пример 1. Значения двумерного массива задаются с помощью вложенного оператора цикла в представленном фрагменте программы. Чему будет равно значение B(2,4)?

Бейсик	Паскаль	Алгоритмический

FOR n=1 TO 5
FOR k=1 TO 5
B(n, k)=n+k
NEXT k
NEXT n
	for n:=1 to 5 do
	for k:=1 to 5 do
	B[n,k]:=n+k;
		нц для n от 1 до 5
		нц для k от 1 до 5
		B[n, k]=n+k
		кц
		кц

Решение. Анализ алгоритма:
Элемент B[n, k]= n+k при n=2, k=4 равен: B[2, 4]= 2+4=6
Ответ: 6.
Пример 2. Значения двумерного массива размера 7*7 задаются с помощью вложенного оператора цикла в представленном фрагменте

программы. Сколько элементов массива будут иметь положительные значения?

Бейсик	Паскаль	Алгоритмический

FOR n=1 TO 7
FOR k=1 TO 7
B(n, k)=k–n
NEXT k
NEXT n
	for n:=1 to 7 do
	for k:=1 to 7 do
	B[n, k]:=k–n;
		нц для n от 1 до 7
		нц для k от 1 до 7
		B[n, k]=k–n
		кц
		кц

Решение. Используем вычислительную таблицы (двумерную матрицу).

n \ k	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6
2	-1	0	1	2	3	4	5
3	-2	-1	0	1	2	3	4
4	-3	-2	-1	0	1	2	3
5	-4	-3	-2	-1	0	1	2
6	-5	-4	-3	-2	-1	0	1
7	-6	-5	-4	-3	-2	-1	0

Можно результат вычислений в черновике оформить в виде таблицы, а потом подсчитать количество отрицательных элементов. Или провести анализ значений элементов массива:

На главной диагонали квадратной матрицы (таблицы) при k=n стоят нули, их 7. Всего в данном массиве 49 элементов, осталось исследовать 49-7=42 элемента. Над главной диагональю при k>n стоят положительные элементы, под главной диагональю при k<n стоят отрицательные элементы. Отрицательных и положительных элементов в данном массиве поровну, т.е. по 21.

Ответ: 21

Задачи третьего типа.

Пример 1. Дан фрагмент программы, обрабатывающей двухмерный массив A размера n*n. Представим массив в виде квадратной таблицы, в которой для элемента массива A[i,j] величина i является номером строки, а величина j – номером столбца, в котором расположен элемент.

Бейсик	Паскаль	Алгоритмический

k = 1
FOR i = 1 TO n
  c = A(i,i)
  A(i,i) = A(k,i)
  A(k,i) = c
	k:=1;
	for i:=1 to n do
	begin
	  c:=A[i,i];
	  A[i,i]:=A[k,i];
		k:=1
		нц для i от 1 до n
		  c:=A[i,i]
		  A[i,i]:=A[k,i]
		  A[k,i]:=c

NEXT i	A[k,i]:=c end	Кц

Когда данный алгоритм меняет местами:
1) два столбца в таблице,
2) две строки в таблице,
3) элементы диагонали и k-ой строки таблицы,
4) элементы диагонали и k-го столбца таблицы?

Решение. Анализ алгоритма:

В данном случае нужно посмотреть на индексы обмениваемых элементов: A[i,i] – элемент диагонали, A[k,i] – элемент k-ой строки. Ответ очевиден: данный алгоритм меняет местами элементы диагонали и k-ой строки таблицы.

Ответ: вариант 3).

Пример 2. Стандартный алгоритм вычисления среднего арифметического элементов числового массива работает на массиве из миллиона элементов 0,5 сек. Оцените время работы того же алгоритма на том же компьютере, если длина массива 3 миллиона.

Решение.

По условию задачи 10^6 элементов - 0,5 сек. Обозначим $3*10^6$ элементов - x.

Составим пропорцию: x=(3*~~10~~^{6}*0,5)\~~10~~^{6}=3*0,5=1, 5 сек

Ответ: 1, 5 сек

Пример 3. Следующий фрагмент программы записывает в переменную Max максимальный элемент в двумерном массиве Dist размера NxN, заполненном целыми неотрицательными числами:

Max:=0; for i:=1 to N do for j:=1 to N do if Dist [i,j]>Max then Max:=Dist [i,j];

На очень медленном компьютере эта программа при N=1000 работала 5 секунд. Оцените время работы этой программы на том же компьютере при N=2000.

Решение.

Чтобы найти максимальный элемент, нужно сравнить каждый очередной элемент с максимальным. Всего элементов в массиве NxN, поэтому и время работы алгоритма пропорционально NxN. Составляем пропорцию:
1000*1000 операций – 5 секунд
2000*2000 операций – x секунд.
x=(4000000*5)/1000000=20 секунд.
Ответ: 20 секунд.
Задачи пятого типа.
Пример 1. В программе описан одномерный целочисленный массив A с индексами от 0 до 10 и целочисленные переменные k и i. Ниже представлен фрагмент одной и той же программы, записанный на разных языках программирования, в котором значения элементов сначала задаются, а затем меняются. Найти элементы массива.

Бейсик	**Паскаль**

FOR i=0 TO 10
A(i)=i;
NEXT i
FOR i=0 TO 10
k=A(i)
A(i)=A(10-i)
A(10-i)=k
NEXT i
	For i:=0 to 10 do
	A[i]:=i;
	For i:=0 to 4 do
	begin
	k:=A[i];
	A[i]:=A[10-i];
	A[10-i]:=k;
	End;

Си	**Алгоритмический язык**

For (i=0; i<=10; i++)
A[i]=i;
For (i=0; i<=4; i++)
{
k=A[i];
A[i]=A[10-i];
A[10-i]=k;
}
	Нц для i от 0 до 10
	A[i]:=i
	Кц
	Нц для i от 0 до 4
	k:=A[i];
	A[i]:=A[10-i];
	A[10-i]:=k;

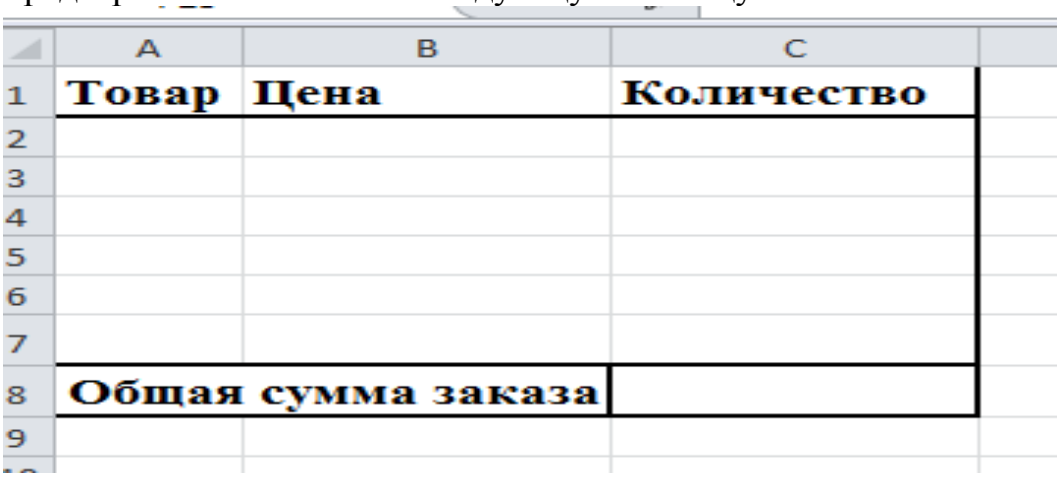
Решение.
В программе приведен стандартный алгоритм транспонирования элементов одномерного массива – заменой элементов строк на элементы столбцов. В массиве 11 элементов, после выполнения первого оператора цикла значения элементов массива равны: 0 1 2 3 4 5 6 7 8 9 10.
Во втором операторе цикла переставляются местами первые 5 элементов массива относительно среднего элемента. Значения элементов, которые меняются местами: 0 – 10, 1 – 9, 2 – 8, 3 – 7, 4 – 6. Средний элемент имеет номер 6 и остается на месте.
Ответ: 10 9 8 7 6 5 4 3 2 1 0

Использование формул массивов в Excel.

Формулы массива в Excel - это специальные формулы для обработки данных из таких массивов. **Формулы массива делятся на две категории:**

- **возвращающие одно значение,**
- **возвращающие целый набор (массив) значений.**

Пример 1. Рассчитать общую сумму заказа по товарному чеку, предварительно заполнив следующую таблицу:



	A	B	C
1	**Товар**	**Цена**	**Количество**
2			
3			
4			
5			
6			
7			
8	**Общая сумма заказа**		
9			

Выполним следующие действия:

- выделяем ячейку С8
- вводим с клавиатуры =СУММ(
- выделяем диапазон В2:В6
- вводим знак умножения (*)
- выделяем диапазон С2:С6 и закрываем скобку функции СУММ, в итоге должно получиться: =СУММ(В2:В6*С2:С6).

Чтобы Excel воспринял нашу формулу как формулу массива, жмем не Enter, как обычно, а Ctrl + Shift + Enter

Обратите внимание на фигурные скобки, появившиеся в формуле - отличительный признак формулы массива. Вводить их вручную с клавиатуры бесполезно - они автоматически появляются при нажатии Ctrl + Shift + Enter.

Excel произвел попарное умножение элементов массивов В2:В6 и С2:С6 и образовал новый массив стоимостей (в памяти компьютера), а затем сложил все элементы этого нового массива.

Пример 2. Транспонирование.

Транспонирование матрицы - это операция над матрицей, при которой ее строки и столбцы меняются местами:

$$a^{T}_{ij} = a_{ji}$$

Построить следующую таблицу:

	A	B	C
1	Список		
2	Фамилия	Имя	Отчество
3			
4			
5			
6			
7			

Выполним следующие действия:

- Выделяем диапазон ячеек для размещения транспонированной таблицы. Поскольку исходный массив ячеек был 7 строк на 3 столбца, то надо выделить диапазон пустых ячеек размером 3 строки на 7 столбцов.
- вводим функцию транспонирования =ТРАНСП(, в качестве аргумента функции выделяем наш массив ячеек A2:C7.

Пример 3. Таблица умножения.

- Вводим в первую строку (B2:K10) и столбец (A2:A10) цифры от 1 до 10
- выделяем диапазон B2:K11
- вводим формулу =A2:A11*B1:K1
- жмем Ctrl + Shift + Enter, чтобы Excel воспринял ее как формулу массива, в результате получаем таблицу:

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Пример 4. Выборочное суммирование

Построить таблицу и ввести формулу в соответствующую ячейку, оформив ее массивом:

G6 fx {=СУММ((C3:C21=G4)*(B3:B21=G5)*D3:D21)}

	A	B	C	D	E	F	G
1							
2		*Товар*	*Заказчик*	*Сумма*			
3		Alice Mutton	ANTON	$56			
4		Alice Mutton	ANTON	$967		Заказчик	ANTON
5		Aniseed Syrup	ALFKI	$592		Товар	Boston Crab Meat
6		Boston Crab Meat	BOTTM	$504		Всего заказов на сумму	1057
7		Alice Mutton	ERNSH	$447			
8		Alice Mutton	ANTON	$237			
9		Boston Crab Meat	LEHMS	$54			
10		Boston Crab Meat	BSBEV	$953			
11		Boston Crab Meat	ANTON	$659			
12		Boston Crab Meat	BONAP	$434			
13		Aniseed Syrup	BOTTM	$801			
14		Alice Mutton	GODOS	$186			
15		Boston Crab Meat	GODOS	$120			
16		Boston Crab Meat	GODOS	$39			
17		Boston Crab Meat	ANTON	$398			
18		Aniseed Syrup	ERNSH	$249			
19		Boston Crab Meat	FRANS	$65			
20		Alice Mutton	BOTTM	$321			
21		Alice Mutton	GODOS	$555			
22							

Практическое занятие 7. Исполнители и практические задания по исполнителям

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 7.

Домашние задания к занятию 8.

Задача 1.

Исполнитель Чертежник перемещается на координатной плоскости, оставляя за собой след в виде линии. Может выполнять команду сместиться на (a, b), где a, b – целые числа. Эта команда перемещает Чертежника из точки с координатами (x, y) в точку с координатами (x+a, y+b). Например, если Чертежник находится в точке с координатами (4, 2), то команда сместиться на (2, -3) переместит Чертежник в точку (6, -1). Цикл «ПОВТОРИ число РАЗ последовательность команд КОНЕЦ ПОВТОРИ» означает, что последовательность команд будет выполнена указанное число раз(число должно быть натуральным).

Чертёжнику был дан для исполнения следующий алгоритм: Повтори 3 раз Сместиться на (–3, –2) Сместиться на (2, 1) Сместиться на (3, 0) Конец.

Какую команду надо выполнить Чертёжнику, чтобы вернуться в исходную точку, из которой он начал движение? 1) Сместиться на (–3, –6) 2) Сместиться на (–6, 3) 3) 4) Сместиться на (3, 6)

Решение.

Запишем общее уравнение изменения координат исполнителя Чертежник в результате выполнения алгоритма:

$\Delta x=3*(-3+2+3)$ и $\Delta y=3*(-2+1+0)$

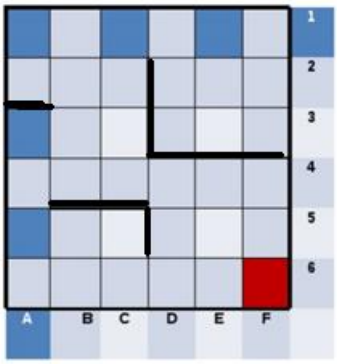
Отсюда, после решения уравнений: $\Delta x=3*(-3+2+3)=6$ и $\Delta y=3*(-2+1+0)=-3$, таким образом, Чертежник после выполнения алгоритма окажется в точке с координатами (–6, 3), а для возврата в исходную точку ему нужно выполнить команду «Сместиться на (6, –3)».

Ответ: Сместиться на (6, –3)

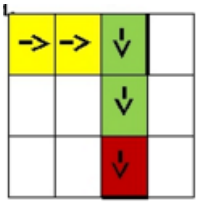
Задача 2.

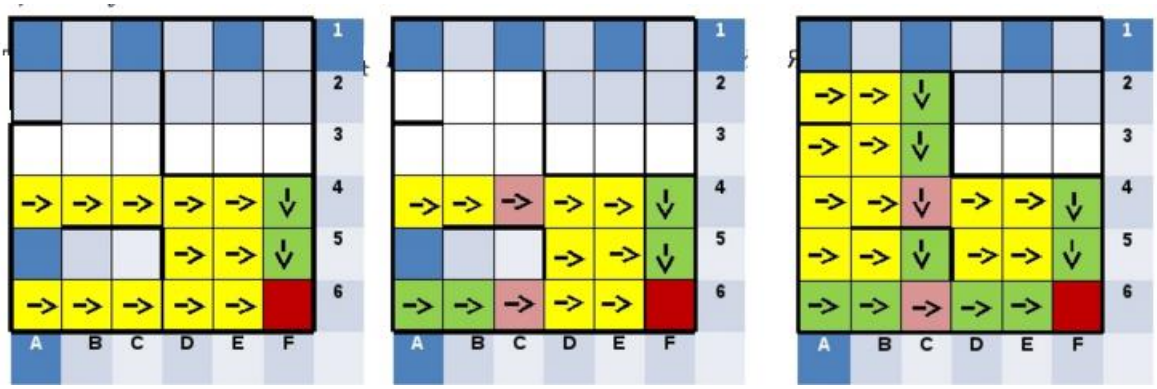
Исполнитель Робот. Система команд исполнителя (СКИ) Робота: вверх вниз влево вправо вверх ↑, вниз -↓, влево ←, вправо →. сверху свободно снизу свободно слева свободно справа свободно Цикл ПОКА <условие> команда выполняется, пока условие истинно, иначе происходит переход на следующую строку. В конструкции ЕСЛИ < условие > ТО команда 1 ИНАЧЕ команда 2 КОНЕЦ ЕСЛИ выполняется команда1 (если условие истинно) или команда 2 (если условие

ложно) Если РОБОТ начнёт движение в сторону находящейся рядом с ним стены, то он разрушится и программа прервется.
В задаче с остановкой в заданной клетке определить, сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка F6 на рисунке)?
Программа Робота: НАЧАЛО ПОКА < справа свободно ИЛИ снизу свободно > ПОКА < справа свободно > вправо КОНЕЦ ПОКА ПОКА < снизу свободно > вниз КОНЕЦ ПОКА КОНЕЦ ПОКА КОНЕЦ



Решение.
Задачи данного типа сводятся к определению таких особых точек в лабиринте, к которым робот вернется, пройдя четыре раза по прямой (пока выполняется условие цикла). При этом он, естественно, пройдет по сторонам прямоугольника. 1) необходимо, чтобы стенки были расставлены так, чтобы Робот упирался в них сначала при движении вниз, затем влево, вверх и вправо; 2) необходимо, чтобы выделенный зеленый коридор был свободен; 3) возможны еще «вырожденные» варианты.
За каждый шаг внешнего цикла Робот проходит путь в виде «сапога», двигаясь вначале вправо до упора, затем вниз до упора (см. рисунок). Красная клетка – особая, в ней заканчивается один внешний цикл и начинается другой. Возможны следующие варианты: а) Робот может попасть в нее, двигаясь вниз из клетки, когда справа – стенка; б) снизу есть стенка; в) снизу стенка есть, справа – нет, тогда выполнится еще один шаг внешнего цикла. По приведенным рисункам видно, что таких всего клеток 24.





Ответ: 24

Задача 3.

Исполнитель Черепашка (Turtle) перемещается на экране компьютера, оставляя след в виде линии. Его СКИ имеет команды: Вперед n, вызывающее изменение движения Черепашки на n шагов в направлении движения. Направо m, вызывающее изменение направления движения Черепашки на m градусов по часовой стрелке 0≤m≤359. Черепашка выполняет алгоритм: «Повтори 5[Вперед 100 Направо X]», при котором на экране появилась незамкнутая ломаная линия. Определить значение X.

Решение.

Выполняя алгоритм, Черепашка оставляет след в виде одинаковых отрезков, расположенных под некоторым углом друг к другу. Сумма внутренних углов выпуклого n-угольника равна $180°*(n-2)$, поэтому угол между его сторонами может быть найден по формуле $180°*(1-2/n)$. Если бы команды **[Вперёд 100 Направо X]** повторялись шесть раз, то такой угол составлял бы между собой стороны правильного шестиугольника и был равен 60 градусам. Так как команды **[Вперёд 100 Направо 60]** повторяются пять раз, то оставленный Черепашкой след представляет собой незамкнутую ломаную линию.

Ответ: 60.

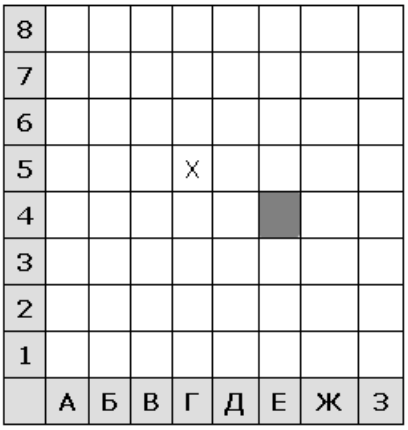
Задача 4.

Исполнитель Муравей перемещается по полю, разделенному на клетки. Размер поля 8x8, строки нумеруются числами, столбцы обозначаются буквами (см. рисунок). Муравей может выполнять команды движения:

Вверх N, Вниз N, Вправо N, Влево N (где N — целое число от 1 до 7), перемещающие исполнителя на N клеток вверх, вниз, вправо или влево соответственно.

Запись: **Повтори k раз Команда1 Команда2 КомандаЗ Конец** означает, что последовательность команд **Команда1 Команда2 КомандаЗ** повторится k раз. Если на пути Муравья встречается кубик,

то он перемещает его по ходу движения. Пусть, например, кубик находится в клетке Е4. Если Муравей выполнит команды **вправо 2 вниз 2**, то сам окажется в клетке **Е3**, а кубик в клетке **Е2.**



Пусть Муравей и кубик расположены так, как указано на рисунке. Муравью был дан для исполнения следующий алгоритм: **Повтори 2 раз Вправо 2 вниз 1 влево 2 Конец.** В какой клетке окажется кубик после выполнения этого алгоритма?

Решение.

После исполнения команд вправо 2 влево 2, согласно рисунку, Муравей окажется в той же клетке, из которой он стартовал. Изначально кубик находится в клетке Е4. Выполнив два раза команду вниз 1, Муравей передвинет кубик в клетку Е2.

Ответ: Е2.

Задача 5.

У исполнителя КАЛЬКУЛЯТОР имеется две команды, которым присвоены номера: 1 - Прибавь 3. 2 - Умножь на 4.

Запишите порядок команд в программе получения из числа 2 числа 104, содержащего лишь номера команд, не более 6.

Решение.

Подобные задачи решаются с «конца». При делении 104 на 4 (обратное действие команды 2) получается 26.При последовательном вычитании из 26 3 (обратное действие команды 1) получается 20, которое при делении на 4 (команда 2) образует 5, из которого вычитанием 3 (команда 1) получаем искомое 2. Отсюда: программа имеет вид - 12112

Ответ: 12112.

Задача 6.

Исполнитель Редактор получает на вход строку цифр и преобразовывает ее.

Редактор может выполнять две команды, в обеих командах v и w обозначают

цепочки цифр:
А) **заменить** (v, w). Эта команда заменяет в строке первое слева вхождение цепочки v на цепочку w. Например, выполнение команды **заменить (111, 27)**
преобразует строку 05111150 в строку 0527150. Если в строке нет вхождений цепочки v, то выполнение команды **заменить** (v, w) не меняет эту строку.
Б) **нашлось** (v). Эта команда проверяет, встречается ли цепочка v в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение ≪истина≫, в противном случае возвращает значение ≪ложь≫. Строка исполнителя при этом не изменяется.
Цикл ПОКА условие последовательность команд КОНЕЦ ПОКА выполняется, пока условие истинно.
В конструкции ЕСЛИ условие ТО команда1ИНАЧЕ команда2 КОНЕЦ ЕСЛИ
выполняется команда1 (если условие истинно) или команда2 (если условие
ложно).
Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 69 идущих подряд цифр 8? В ответе запишите полученную строку.
НАЧАЛО
ПОКА **нашлось** (3333) ИЛИ **нашлось** (8888)
ЕСЛИ **нашлось** (3333)
ТО **заменить** (3333, 88)
ИНАЧЕ **заменить** (8888, 33)
КОНЕЦ ЕСЛИ КОНЕЦ ПОКА КОНЕЦ
Решение.
Программа будет работать до тех пор, пока в строке есть цепочка цифр 3333 или цепочка цифр 8888. Если в строке встречается 3333, то заменяем на 88.
Если в строке нет цепочки 3333, но встречается цепочка 8888, то заменяем ее на 33.
Наша строка состоит из 69 идущих подряд цифр 8.
1) заменяем первые 8888 на 33 и получаем 33 и (65 цифр 8)
2) заменяем следующие 8888 на 33 и получаем 3333 и (61 цифру 8)
3) появилась цепочка 3333, поэтому мы должны заменить ее на 88. Получаем строку, состоящую из 63 цифр 8.

Анализ алгоритма: за 3 шага мы заменили 8 восьмерок на 2 (или удалили 6 восьмерок из нашей строки).таким образом, за каждые 3 шага мы должны удалять по 6 восьмерок.
63-6=57;57-6=51;51-6=45;45-6=39;39-6=33;33-6=27;27-6=21;21-6=15;15-6=9;
9-6=3 - останется строка, состоящая из 3 следующих подряд цифр 8 (888)

Ответ: 888

Лекция (по материалам презентации)

Тема: Исполнители и практические задания по исполнителям.

Введение.

В определении алгоритма присутствует его исполнитель. **Исполнитель алгоритма** выполняет действия по механическим правилам – **командам**. Совокупность команд представляет собой **систему команд исполнителя – СКИ**. Обстановка, в которой действует исполнитель, называется **средой исполнителя.**

Список терминов (глоссарий)

Исполнитель:

1.**Исполнитель** - человек или коллектив людей, вооруженных набором инструментов и обученный выполнению некоторой совокупности операций в заданной последовательности.

2. **Исполнитель** - автоматическое устройство (электронное, электромеханическое и т.п.), изготовленное таким образом, что, будучи включенным в работу, выполняет заданную последовательность операций над некоторым исходным продуктом, преобразуя его в заданный конечный продукт.

Исполнитель алгоритма - некоторая абстрактная или реальная система, способная выполнить действия, предписываемые алгоритмом.

Обычно исполнитель ничего не знает о цели алгоритма. Он выполняет все полученные команды формально (не обдумывая). В информатике универсальным исполнителем алгоритмов является компьютер.

На рисунке 5 представлена схема основных характеристик исполнителя. Все исполнители, согласно схеме, разделяются на две категории: формальные и неформальные. К основным характеристикам исполнителей (как формальным, так и неформальным) относятся: круг решаемых задач, среда исполнителя, система команд исполнителя СКИ, режимы работы исполнителя.

Рисунок 5 - Схема основных характеристик исполнителя.

Формально любой алгоритм можно представить как модель деятельности исполнителя алгоритмов. Вследствие этого конечной целью разработки алгоритма является программа последовательности действий исполнителя в формате его СКИ. На рисунке 6 показана модель деятельности исполнителя алгоритмов при разработке алгоритма, состоящая из следующих этапов:

1. Определение объектов, представленных в задаче,
2. Определение свойств, отношений и действий между объектами,
3. Описание исходных данных и результата,
4. Указание последовательности действий,
5. Запись данной последовательности действий в СКИ.

Рисунок 6 - Модель деятельности исполнителя алгоритмов

При реализации последовательности действий исполнителя могут выявляться ошибки в его работе, которые обычно делятся на три

основных вида. В таблице 2 представлены виды ошибок при работе Исполнителя.

Таблица 2 – Виды ошибок Исполнителя

Ответ (отклик) Исполнителя на ошибочную команду	**Описание**
«НЕ ПОНИМАЮ»	Заданной команды не существует в СКИ
«НЕ МОГУ»	Команда не может быть выполнена Исполнителем, обычно из-за выхода за пределы среды исполнителя
Логическая ошибка	После выполнения команды Исполнителем получен неверный результат. Причина: ошибка в составлении задачи или при разработке алгоритма

При выполнении ошибочной команды появляется ответ (отклик) исполнителя на нее, и возникает так называемый отказ. Как правило, отказы исполнителя возникают при вызове команды в недопустимом для данной команды состоянии среды. Иногда попытка исполнителя выполнить команду приводит к аварии.

В информатике особый класс составляют **Учебные Исполнители.**

Учебными исполнителями в информатике называют различные образы на экране компьютера, которыми можно управлять, отдавая команды. Используются Учебные Исполнители для обучения составлению управляющих алгоритмов.

Существует множество различных Учебных Исполнителей, придуманных для занятий по информатике: Черепашка, Робот, Чертежник, Кенгуренок, Пылесосик, Муравей, Кузнечик, Кукарача и др. Одни исполнители создают рисунки на экране, другие складывают слова из кубиков с буквами, третьи перетаскивают предметы из одного места в другое. Данные Учебные Исполнители имеют следующие общие свойства:

- управляются программным путем;
- имеют определенную среду деятельности, систему команд управления, режимы работы;
- предназначены для построения различных алгоритмов управления.

Некоторые Учебные Исполнители занимаются рисованием на экране компьютера, например, такие, как Черепашка, Кенгуренок, Чертежник. Их называют Графическими Исполнителями, сокращенно ГРИС (ГРафический ИСполнитель). Среда ГРИС - лист (страница

экрана) для рисования. ГРИС может перемещаться в горизонтальном и вертикальном направлениях с постоянным шагом. При этом Исполнитель может двигаться только по линиям сетки листа и не может выходить за границы. Состояние Исполнителя на поле определяется, во-первых, его местоположением (в какой точке поля он находится) и направлением (куда он смотрит). Обычно система команд ГРИС имеет следующие команды: шаг, поворот, прыжок. Шаг – перемещение ГРИС на один шаг вперед с рисованием линии; поворот – поворот на 90^0 против часовой стрелки; прыжок – перемещение на один шаг вперед без рисования линии. Такие команды называются простыми командами.

Основными ГРИС при составлении заданий по ЕГЭ являются Чертежник и Черепашка, для составления алгоритмов их работы обычно требуются знания и навыки решения задач по элементарной геометрии.

Практические задачи, выполняемые на занятии.

Решение данных задач даются в приложении 1.

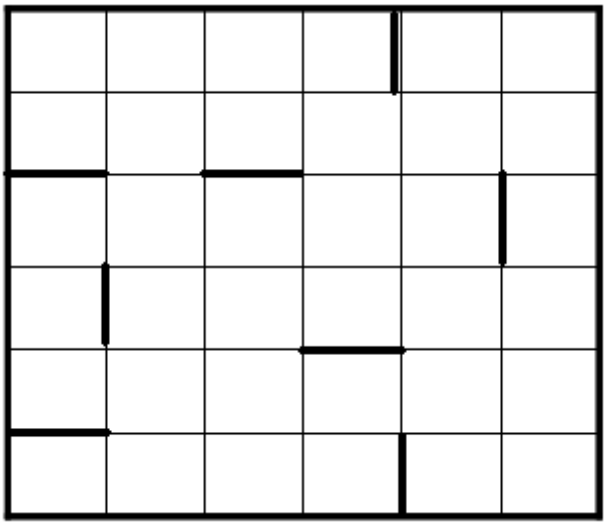
Задача 1.

СКИ РОБОТ состоит из команд: вверх, вниз, влево, вправо. 4 команды проверяют истинность условия отсутствия стены у той клетки, где находится РОБОТ ПОКА <условие>: сверху свободно, снизу свободно, слева свободно, справа свободно. В Цикле Пока <условие> команда выполняется при истинности условия, иначе осуществляется переход на следующую строку.

РОБОТ выполнил программу:

НАЧАЛО
ПОКА <справа свободно> вправо
ПОКА <снизу свободно> вниз
ПОКА <слева свободно> влево
ПОКА <сверху свободно> вверх
КОНЕЦ

Сколько клеток приведенной программы соответствует требованию, что после ее выполнения РОБОТ остановится в клетке начала своего движения?



Задача 2.

У исполнителя КАЛЬКУЛЯТОР имеется две команды, которым присвоены номера: 1 Прибавь 3. 2 Умножь на 4.

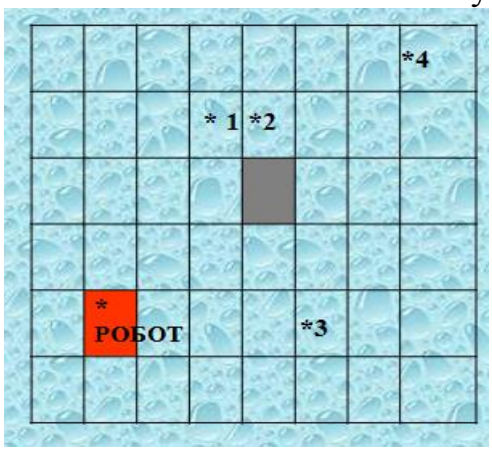
Запишите порядок команд в программе получения из числа 2 числа 104, содержащего лишь номера команд, не более 6.

Задача 3.

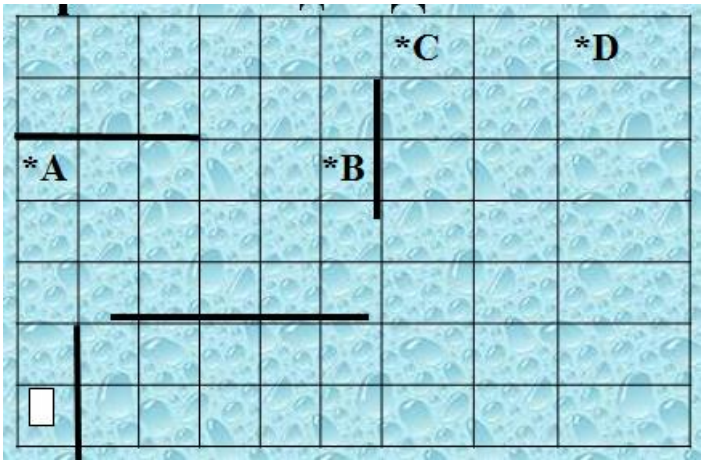
СКИ РОБОТ состоит из команд: вверх, вниз, влево, вправо. Он передвигается на одну клетку поля, на котором расположен «клад» (закрашенная клетка). В поисках «клада» РОБОТ выполняет следующий алгоритм:

НАЧАЛО ПОКА клетка не закрашена ДЕЛАТЬ
ЕСЛИ сверху свободно ТО идти вверх ИНАЧЕ
ПОКА снизу свободно ДЕЛАТЬ идти вниз
КОНЕЦ
ЕСЛИ справа свободно ТО идти вправо ИНАЧЕ
ПОКА слева свободно ДЕЛАТЬ идти влево
КОНЕЦ КОНЕЦ КОНЕЦ

Какая из точек окажется на пути Робота?



Задача 4.
Исполнитель Кузнечик находится на числовой оси. Команды Кузнечика: вперед 3, назад 5. Исходное положение Кузнечика – (+20). За какое минимальное количество команд Кузнечик окажется в точке (-4)?
Задача 5.
Исполнитель Кузнечик должен побывать в точках 1,2, 3, 4, 5 на числовой оси по одному разу, не выходя за пределы 5. Команды Кузнечика: вперед 3, назад 2. Исходное положение Кузнечика –0. Какое минимальное количество команд Кузнечика должна содержать такая программа?
Задача 6.
У исполнителя УТРОИТЕЛЬ имеется две команды, которым присвоены номера: 1 Прибавь 1. 2 Умножь на 3.
Запишите порядок команд в программе получения из числа 5 числа 49, содержащего лишь номера команд, не более 5.
Задача 7.
У исполнителя ДЕЛИТЕЛЬ имеется две команды, которым присвоены номера: 1 Вычти 1. 2 Раздели на 2.
Запишите порядок команд в программе получения из числа 57 числа 7, содержащего лишь номера команд, не более 5.
Задача 8.
Специализированный компьютер имеет СКИ из следующих команд:
A? – ввод данных в регистр A, A! - вывод данных из регистра A, A*B – выражение истинно при сохранении без изменений единичных разрядов регистра A, соответствующим нулевым разрядам регистра B, в противном случае соответствует инвертированному разряду в регистре A. Компьютер выполнил следующую программу: A? B? A*B A!. Найти содержимое регистра A, в котором находились результаты выполнения данной программы.
Задача 9.
Исполнитель Робот действует на клетчатой доске, между соседними клетками которой могут стоять стены. Робот может передвигаться по клеткам доски и выполняет следующие команды: вверх, вниз, направо, налево. выберите точку, в которой остановится РОБОТ, если он перемещается по следующему алгоритму:
ПОКА сверху свободно ИЛИ справа свободно ДЕЛАТЬ
ЕСЛИ сверху свободно ТО идти вверх
ИНАЧЕ идти вправо КОНЕЦ КОНЕЦ



Задача 10.

Исполнитель Черепашка (Turtle) перемещается на экране компьютера, оставляя след в виде линии. Его СКИ имеет команды: Вперед n, вызывающее изменение движения Черепашки на n шагов в направлении движения. Направо m, вызывающее изменение направления движения Черепашки на m градусов по часовой стрелке $0 \leq m \leq 359$. Черепашка выполняет алгоритм: Повтори 9 [Вперёд 50 Направо 60]. Какая фигура появится на экране?

Практическое занятие 8. Основы теории алгоритмов

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 8.

Домашние задания к занятию 9.

Задача 1.

На вход алгоритма подаётся натуральное число *N*. Алгоритм строит по нему
новое число *R* следующим образом:
1) Строится двоичная запись числа *N*.
2) К этой записи дописываются справа ещё два разряда по следующему правилу: а) складываются все цифры двоичной записи числа *N*, и остаток от
деления суммы на 2 дописывается в конец числа (справа). Например, запись 11100 преобразуется в запись 111001; б) над этой записью производятся те же действия – справа дописывается остаток от деления суммы её цифр на 2.
Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа *N*) является двоичной записью искомого числа *R*. Укажите минимальное число *R*, которое превышает число 83 и может являться результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

Решение.

Заметим, что после второго пункта условия задачи получаются только четные числа (т.к. если число в двоичной системе заканчивается на **0**, то это число четное). Таким образом, нас будут интересовать только четные числа.
Наименьшим возможным числом, превышающим число **83,** является число **84**. С ним и будем работать.
Переведем **84** в двоичную систему счисления: 84=**1010**00
В данном числе выделенная часть — **N**. Значит необходимое нам двоичное число —**10101**. После первого пункта задачи к данному числу должна была добавиться справа единица, так как число нечетное, однако мы имеем 0. Значит, это число не подходит.
Возьмем следующее четное число — **86**. Переведем его в двоичную систему счисления: 86=**1010**10
В данном числе выделенная часть —**N**. Значит, необходимое нам двоичное число — **10101**. После первого пункта задачи к данному числу должна была добавиться справа **единица**: 101011. Затем добавляется **0**: 1010110. Значит, данное число подходит.
Ответ: 86

Задача 2.

Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

F(1)=1 F(n)=F(n-1)*n, при n>1

Чему равно значение функции F(5)?

Решение.

Очевидно, функция F(n) вычисляет n!=1*2*3*…*n. 5!=1*2*3*4*5=120. **Ответ: 120.**

Задача 3.

Определите, сколько звездочек будет напечатано в результате вызова F(5) приведенной подпрограммы на языке Pascal:

```
procedure F(n:integer);
begin
  if n>1 then
      begin
      F(n div 2); F(n-1);
          end;      write(‘*’);
end;
```

Решение.

Определим рекуррентную формулу, обозначив Q(n) - количество звездочек, которые будут выведены на экран при вызове F(n).

Из программы видно, что

Q(n)=1, для всех n<=1,

Q(n)=1+Q(n div 2)+Q(n-1) при n>0.

Составим таблицу для нахождения количества выведенных на экран звездочек.

n	1	2	3	4	5
Q(n)	1	3	5	9	13

Q(1) = 1

Q(2) = 1 + Q(1) + Q(1) = 1 + 1 + 1 = 3

Q(3) = 1 + Q(1) + Q(2) = 1 + 1 + 3 = 5

Q(4) = 1 + Q(2) + Q(3) = 1 + 3 + 5 = 9

Q(5) = 1 + Q(2) + Q(4) = 1 + 3 + 9 = 13

Ответ: 13.

Задача 4.

Напишите в ответе число, которое будет напечатано в результате выполнения следующего алгоритма. Программа представлена на языке Pascal:

```
var a, b, t, M, R :longint;
function F(x: longint): longint;
```

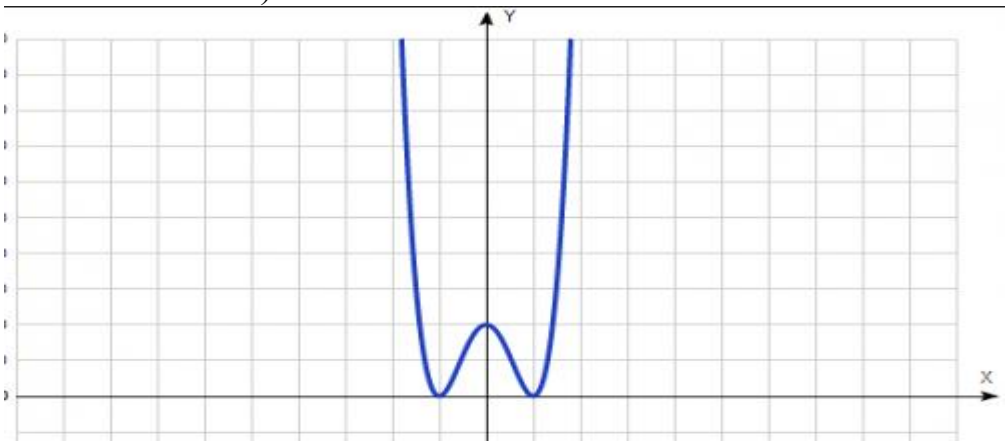
```
begin
F:= 2*(x*x-1)*(x*x-1)+27;
end;
begin
a:=-25; b:=25;
M:=a; R:=F(a);
for t:= a to b do begin
if (F(t) <= R) then begin
M:=t; R:=F(t) end;
end;
write(M-R)
end.
```

Решение.

Анализ алгоритма:

- В цикле программы ищется минимальное значение, возвращаемое функцией **F(t)**, вызываемой с параметрами от **-25** до **25**.
- После завершения работы цикла минимальное значение функции помещается в переменную **R**, а параметр, при котором функция возвратила минимальное значение, помещается в переменную **M**.
- В конце программы выводится результат (**M-R)**.
- В теле функции: **2*(x*x-1)*(x*x-1)+27**
- При раскрытии скобок получится уравнение 4-й степени, корни которого находятся через вычисление производной функции. Поэтому в дальнейшем пока не будем учитывать число **27** в выражении.
- Преобразуем выражение, раскрыв скобки:

F(x) = 2*(x*x-1)*(x*x-1)+27 = 2*x^2-2*x^2-1 = 2*x^4-2*x^2-2*x^2+2 = 2*x^4-4*x^2+2 Получаем уравнение 4-й степени, график которой на рисунке представляет собой квадратичную параболу с ветвями, направленными вверх (первый коэффициент уравнения — положительный).



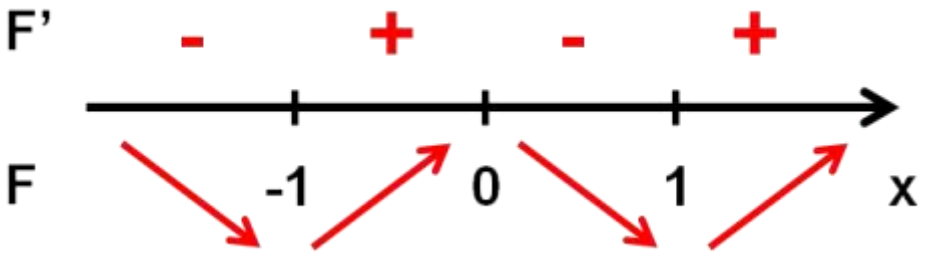
- Вычислим производную функции:

F'(x) = 8*x^3 – 8*x = 8*x*(x^2-1) = 8*x*(x-1)*(x+1)

- Находим нули производной (точки, в которых график пересекает ось ox):

y=0, x=1, x=-1

- Методом интервалов согласно рисунку находим отрезки, в которых график убывает и возрастает:



- Таким образом, функция имеет **два минимума** — в точках **-1** и **1** (оси ox).
- Так как в условии программы стоит F(t) <= R, данное то условие будет работать и при F(1). Проверим:

F(-1) = 2*0*0 = 27

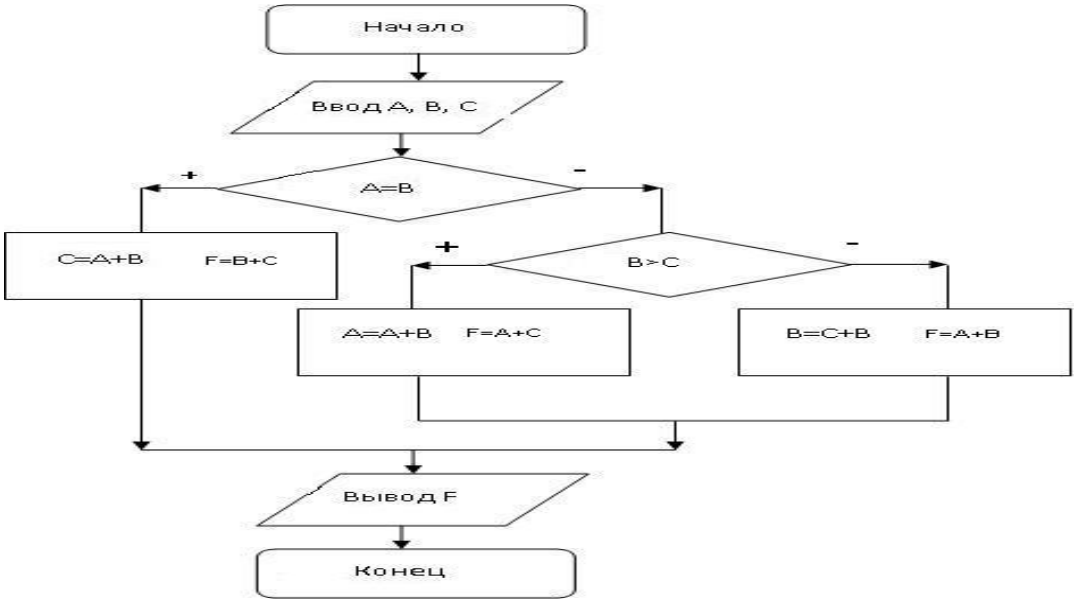
F(1) = 2*0*0 = 27

- Таким образом, минимальное значение функции: **27** (R), а параметр при минимальном значении функции: **1** (M).
- Результат: **M - R = 26**

Ответ: 26

Задача 5.

Дана блок-схема алгоритма:



Определить результат выполнения алгоритма при определённых значениях исходных данных. Например, при A=7; B=8; C=9

Решение.

Определим последовательность решения задачи по этапам:

1) Ввод: A=7; B=8; C=9

Проверка условия A=B: условие не выполняется A≠B, тогда

Проверка условия B>C: условие не выполняется B<C; определяем B=9+8=17 F=7+17=24

Вывод F=24

Ответ: 24

Задача 6.

Ниже на нескольких языках программирования записан рекурсивный алгоритм F. Запишите подряд без пробелов и разделителей все числа, которые будут напечатаны на экране при выполнении вызова F(9). Числа должны быть записаны в том же порядке, в котором они выводятся на экран.

Бейсик

```
SUB F(n)
  IF n > 0 THEN
     PRINT n
     F(n - 3)
     F(n \ 3)
   END IF
END SUB
```

Python

```
def F(n):
    if n > 0:
        print(n)
        F(n - 3)
        F(n // 3)
```

Алгоритмический язык

```
алг F(цел n)
нач
   если n > 0 то
     вывод n
     F(n - 3)
     F(div(n, 3))
   все
кон
```

Паскаль

```
procedure F(n: integer);
begin
  if n > 0 then
  begin
    write(n);
    F(n - 3);
    F(n div 3)
  end
end;
```

C++

```
void F(int n){
    if (n > 0){
        std::cout <<n;
        F(n - 3);
        F(n / 3);
    }
}
```

Решение.

Из заданной рекуррентной формулы по условию очевидно, что функция (F(n)) зависит от предыдущей функции (F(n–1)) и от предыдущей функции (F(n-2)). Так как первые два значения заданы: (F(0) = 1, F(1) = 1), то можно построить таблицу последующих значений:

n	0	1	2	3	4	5	6
F(n)= 2*F(n – 1)+F(n — 2)	1	1	2*1+ 1 =3	2*3+ 1 =7	2*7+ 3 =17	2*17+ 7 =41	2*41+1 7 =99

Таким образом, получаем, что при вызове функции F(6) результатом будет число **99**

Ответ: 99

Задача 7.

Автомат получает на вход два двузначных шестнадцатеричных числа. В этих числах все цифры не превосходят цифру 6 (если в числе есть цифра больше 6, автомат отказывается работать). По этим числам строится новое шестнадцатеричное число по следующим правилам:

1. Вычисляются два шестнадцатеричных числа – сумма старших разрядов полученных чисел и сумма младших разрядов этих чисел.
2. Полученные два шестнадцатеричных числа записываются друг за другом в порядке возрастания (без разделителей).

Пример. Исходные числа 66, 43. Поразрядные суммы: A, 9. Результат: 9A. Определите, какое из предложенных чисел может быть результатом работы автомата: 1) 9F 2) 911 3) 42 4) 7A

Решение.

Так как автомат обрабатывает числа не превосходящие 6 (6 – обрабатывается), то при выполнении пункта 1 могут получаться цифры: 0,1,3,4,5,6,7,8,9,A,B,C (6+6=12=C - максимальное значение).

Таким образом, из предложенных вариантов ответов исключаем 1-9F, 2 -911(11 в шестнадцатеричной системе - B).

Согласно пункту 2, числа должны быть записаны в порядке возрастания – исключаем вариант 3 – 42.

Все условия задачи выполнены в 4 варианте ответов -7A

Ответ: 4

Лекция (по материалам презентации)
Тема: Теория алгоритмов.

Введение.

Алгоритмы являются основным инструментом для написания компьютерных программ. В основные этапы создания компьютерной программы обычно включают следующие процессы:

- - проектирование программы,

- - написание программы,
- - тестирование и отладка программы.

Алгоритм необходим для проектирования программы.

Тема «Алгоритмизация и программирование» наиболее широко представлена в ЕГЭ по информатике. Для успешного выполнения заданий этой темы нужно знать и уметь использовать на практике основные алгоритмические конструкции, составлять, анализировать и выполнять алгоритмы, используя различные формы записи, а также уметь использовать простые структуры данных – одномерные и двумерные массивы.

Основные разделы по теории алгоритмов при подготовке к ЕГЭ:

- **Понятие алгоритма.**
- **Основные свойства алгоритмов.**
- **Способы описания алгоритмов.**
- **Типы алгоритмов.**
- **Примеры алгоритмов.**

Словарь терминов (глоссарий)

Алгоритм:

- система правил, описывающая последовательность действий, которые необходимо выполнить для решения задачи,
- точное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи.

Алгоритм решения вычислительной задачи – совокупность правил преобразования исходных данных в результатные данные.

Основные термины в теории алгоритмов: алгоритм, исполнитель алгоритма, команда исполнителя, система команд исполнителя СКИ, среда исполнителя. Данные понятия и практические задачи подробно рассматривались на занятии 7.

Основные свойства алгоритмов.

С точки зрения исполнителя:

- **Понятность,** т. к. он составляется из команд, входящих в СКИ.
- **Конечность (результативность) –** выполнение алгоритма должно приводить к результату за конечное число шагов.
- **Точность –** каждая команда алгоритма управления определяет однозначное действие исполнителя.

Правильное построение алгоритма характеризуется **полным набором данных – необходимым и достаточным набором данных для решения поставленной задачи (получения искомого результата).**

С точки зрения алгоритма:

- **детерминированность** или **определенность –** свойство алгоритма, благодаря которому каждая команда воспринимается однозначно.
- **результативность** – свойство алгоритма, которое характеризует, что при точном исполнении всех команд алгоритма, процесс решения задачи прекращается за конечное число шагов и при этом получается определенный результат.
- **массовость** - свойство алгоритма, которое характеризует пригодность алгоритма для решения задач некоторого класса.
- **дискретность** - свойство алгоритма, которое характеризует разбиение процесса решения задачи на последовательность отдельных шагов**.**

Виды алгоритмов:

- **Терминистический** (выполняющийся за конечное число шагов),
- **Детерминистический** (при строгом указании очередного шага алгоритма),
- **Детерминированный** (определение результата независимо от последовательности выполняемых шагов).

Исходными данными для построения алгоритмов, независимо от способов их написания являются:

- постановка задачи, которую необходимо решить с помощью алгоритма.

- способ (метод) решения задачи.

Способы описания алгоритмов

Алгоритм может быть описан (формализован) по некоторым правилам посредством конкретных изобразительных средств.

Основные способы записи (формализации) алгоритма:

- **словесный,**
- **графический,**
- **формульно-словесный,**
- **алгоритмический язык.**
- Наибольшее представление из-за своей наглядности получил **графический (блок-схемный)** способ записи алгоритма.

Блок-схема – графическое изображение логической структуры алгоритма, в которой каждый этап процесса обработки информации представляется в виде геометрических символов (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций.

Перечень символов, их наименование, отображаемые ими функции, форма и размер определяются правилами ГОСТ и представлен на рисунке 7.

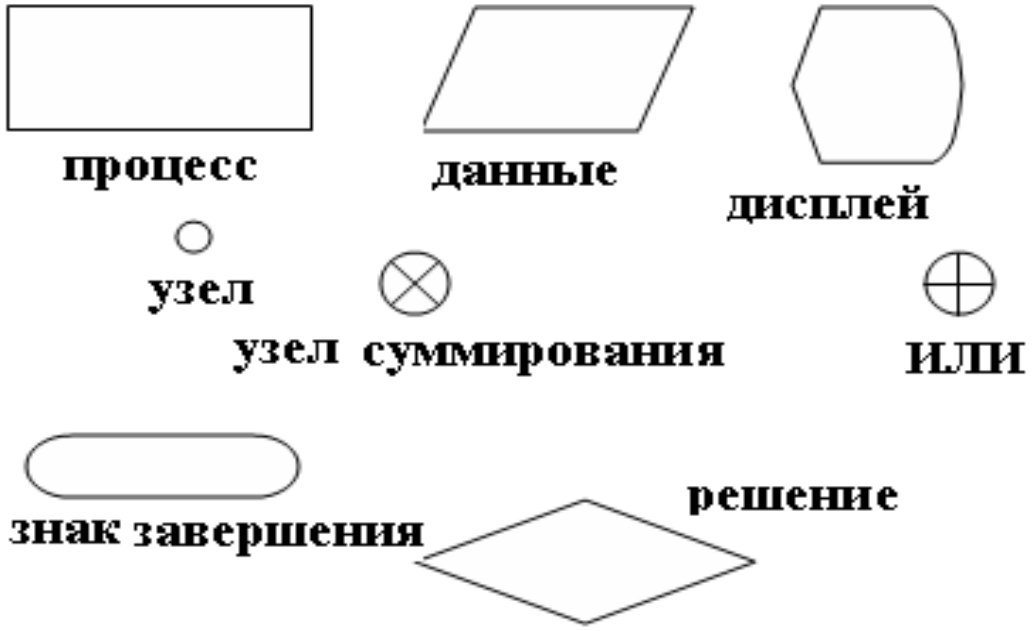


Рисунок 7 - Некоторые элементы блок-схем алгоритмов

Типы алгоритмов

Результатами выполнения алгоритмов могут быть информация и управляющие сигналы, по которым осуществляются определенные преобразования.

При всем многообразии алгоритмов решения задач выделяют **три основных вида вычислительных процессов и соответствующих типов алгоритмов:**

- **Линейный процесс,** повторяющийся последовательно шаг за шагом.
- **Ветвящийся процесс,** выполняющийся по одному из заранее определенных условий.
- **Циклический процесс,** повторяющийся более одного раза.

Цикл – это многократно повторяемый участок вычислений. Имеются циклы с определенным (заранее заданным) числом повторений и с неопределенным числом повторений.

Количество повторений зависит от соблюдения условия. Если проверка условия идет в начале цикла, то такой цикл называется с предусловием, в конце - с потусловием.

И ветвящийся и циклический алгоритм изображаются с помощью единого блока проверки по условию. Основной графический блок проверки по условию показан на рисунке 8. Блок- схемы типов алгоритмов и разновидностей алгоритмов показаны на рисунках 9 и 10.

Рисунок 8 - Графический блок проверки по условию.

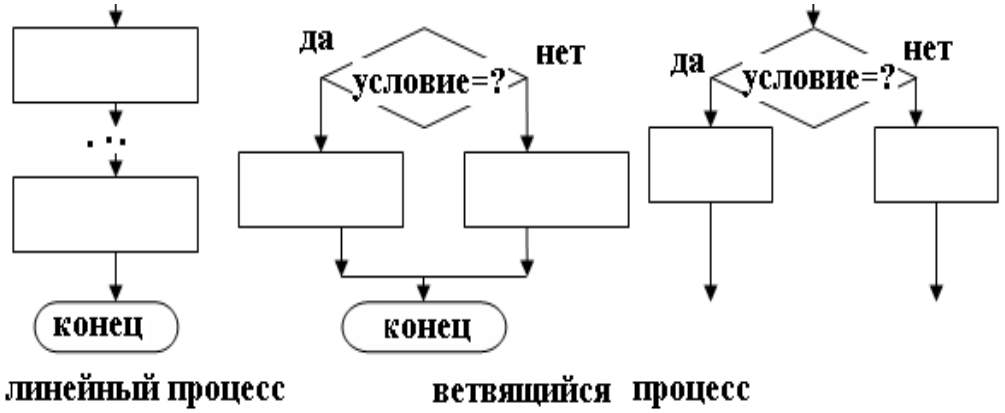


Рисунок 9 - Графические блок-схемы алгоритмов.

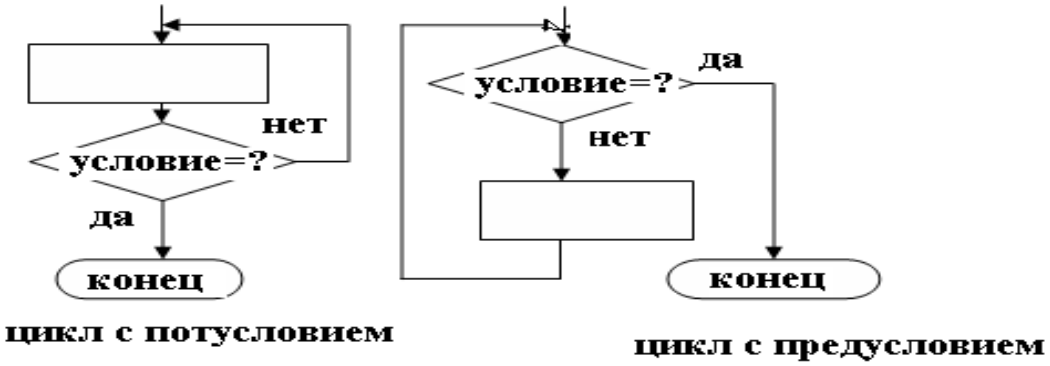


Рисунок 10 - Графические блок-схемы циклических алгоритмов.

Примеры алгоритмов

1. Алгоритм (Лотки – Вольтерра) ХИЩНИК-ЖЕРВА. Исследование динамической модели численности конкурирующих популяций (хищника и жертвы), когда исчезновение одной из них означает выигрыш для другой (дополнительные ресурсы), однако исчезновение жертвы влечет за собой исчезновение хищника, для которого в модели жертва является единственным кормом.

2. Алгоритм Муравьиные Колонии. Итерационный метод последовательного приближения к оптимальному решению. Каждая итерация – запуск «искусственного муравья», который по некоторому

правилу пытается выбрать наилучший маршрут к «пище» (оптимуму функции), используя метки своих предшественников.

Инструкция по решению задач на алгоритмы (в том числе, и в виде блок-схем):

1. Внимательно ознакомьтесь с условием исходной задачи**.** Продумайте ее решение: имеется ли цикличность в задаче. Возможно, заданы операции, выполнение которых обусловлено удовлетворением разных условий. Выпишите все известные данные и искомые величины.
2. Любой алгоритм требует формализованной записи. Если вам требуется составить блок-схему алгоритма, используйте специальные элементы для обозначения каждой операции создаваемой инструкции. Как правило, это блоки из прямоугольных и ромбических фигур, соединенные в общее дерево.
3. Составьте общий алгоритм решения задачи**.** На первом шаге введите в алгоритм переменные величины, обозначающие известные данные и результирующие значения. Присвойте переменным известные из условия задачи значения.
4. Детализируйте алгоритм. Подробно распишите условие задачи**.** Каждый шаг инструкции должен быть записан на отдельной строке. При необходимости задайте циклы или разветвления алгоритма.
5. Все действия в шагах инструкции производите с заданными переменными. Если необходимо ввести вспомогательные переменные, включите их дополнительно в самом начале алгоритма.
6. Часто из смысла исходной задачи в процессе решения вытекают условия, при удовлетворении которых над данными проводится одно действие, а без удовлетворения – другое. В этом случае речь идет о разветвлении алгоритма. Оформите его двумя ветками дерева-инструкции.
7. Если при разветвлении алгоритма после прохождения условия одну из веток необходимо вернуть назад по телу алгоритма, то образуется циклический алгоритм. Четко проследите, чтобы цикл внутри инструкции не был бесконечен и имел конечное число итераций.
8. Любая последовательность выполняемых действий должна привести к конечному результату, заданному в условии задачи**.** После получения искомой величины, завершите тело алгоритма и запишите полученный ответ.

Практические задачи, выполняемые на занятии.

Решение данных задач даются в приложении 1.

Задача 1.

Составить блок-схему алгоритма, решающего следующую задачу:

Даны три вещественных положительных числа a, b и c. Найти площадь треугольника, стороны которого равны a, b и c. При этом для нахождения площади треугольника по трем его известным сторонам воспользуемся формулой Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где p – полупериметр треугольника, равный

$$p = \frac{a+b+c}{2}.$$

Задача 2.

Составить блок-схему решения следующей задачи. Даны значения двух действительных переменных a и b. Найти наибольшее значение из a и b.

Задача 3.

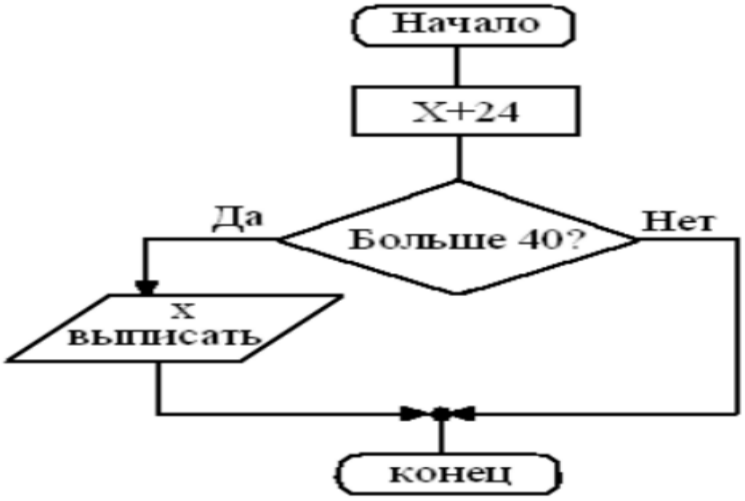
Составить блок-схему решения следующей задачи. Даны значения трех действительных переменных a, b и c. Найти наибольшее значение из a, b и c.

Задача 4.

Старый мост. Семья (папа, мама, сын и бабушка) ночью подошла к мосту, способному выдержать только двух человек одновременно. По мосту можно двигаться только с фонариком. Известно, что папа может перейти мост в одну сторону за минуту, мама - за две, сын - за пять и бабушка - за десять минут. Фонарик у них один. Светить издали нельзя. Носить друг друга на руках тоже. Если по мосту идут двое, время перехода определяется наиболее медлительным членом семьи. Как семье переправиться за 17 минут?

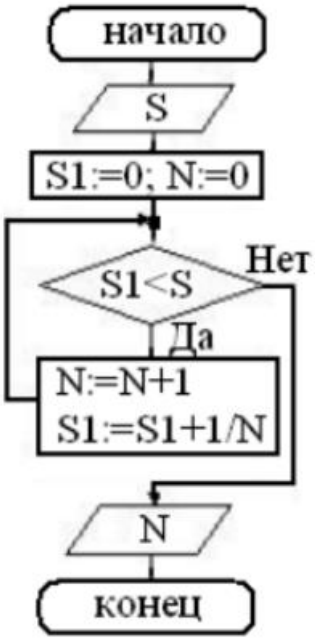
Задача 5.

Из ряда чисел 15, 16, 17, 18 выписать значения x, удовлетворяющие условию (см. блок-схему на рисунке).



Задача 6.

Дана блок-схема (см. рисунок). Какое значение будет иметь N на выходе, если: S=1,1?



Задача 7.

Определите, какое число будет напечатано в результате выполнения программы, записанной ниже на нескольких языках программирования.

Бейсик

```
DIM N, S AS INTEGER
N = 1
S = 0
WHILE N <= 150
  S = S + 30
  N = N * 5
WEND
PRINT S
```

Python

```
n = 1
s = 0
while n <= 150:
    s = s + 30
    n = n * 5
print(s)
```

Алгоритмический язык

```
алг
нач
  цел n, s
  n := 1
  s := 0
  нц пока n <= 150
    s := s + 30
    n := n * 5
  кц
  вывод s
кон
```

Паскаль

```
var n, s: integer;
begin
  n := 1;
  s := 0;
  while n <= 150 do
  begin
    s := s + 30;
    n := n * 5
  end;
  write(s)
end.
```

Задача 8.

Ниже на нескольких языках программирования записан рекурсивный алгоритм *F*. Чему равна сумма напечатанных на экране чисел при выполнении вызова *F*(10)?

Бейсик

```
DECLARE SUB F(n)
SUB F(n)
  IF n > 2 THEN
    PRINT n
    F(n - 3)
    F(n - 4)
  END IF
END SUB
```

Python

```
def F(n):
    if n > 2:
        print(n)
        F(n - 3)
        F(n - 4)
```

Алгоритмический язык

```
алг F(цел n)
нач
  если n > 2 то
    вывод n, нс
    F(n - 3)
    F(n - 4)
  все
кон
```

Паскаль

```
procedure F(n: integer);
begin
  if n > 2 then begin
    writeln(n);
    F(n - 3);
    F(n - 4)
  end
end;
```

Задача 9.

Напишите в ответе число, которое будет напечатано в результате выполнения следующего алгоритма (для Вашего удобства алгоритм представлен на нескольких языках программирования).

Бейсик

```
DECLARE SUB F(n)
SUB F(n)
  IF n > 2 THEN
    PRINT n
    F(n - 3)
    F(n - 4)
  END IF
END SUB
```

Python

```
def F(n):
    if n > 2:
        print(n)
        F(n - 3)
        F(n - 4)
```

Алгоритмический язык

```
алг F(цел n)
нач
  если n > 2 то
    вывод n, нс
    F(n - 3)
    F(n - 4)
  все
кон
```

Паскаль

```
procedure F(n: integer);
begin
  if n > 2 then begin
    writeln(n);
    F(n - 3);
    F(n - 4)
  end
end;
```

Практическое занятие 9. Программирование. Основные понятия

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 9.

Домашние задания к занятию 10.

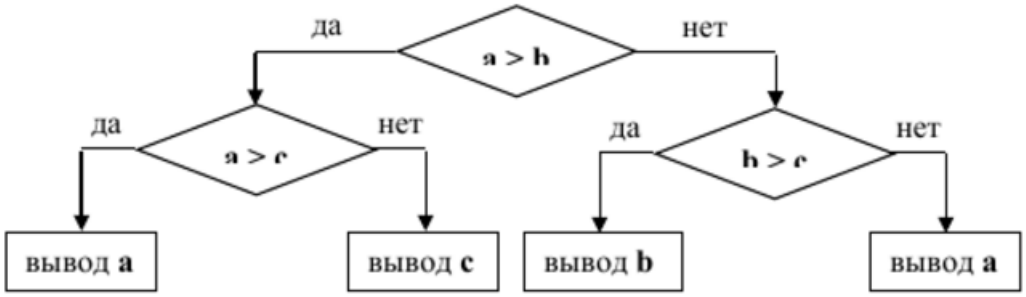
Задача 1.

Вывести на экран наибольшее из трех чисел.

Формулировка. Даны три числа. Вывести на экран то из них, которое больше.

Решение.

В решении нужно использовать условный оператор if, в данном случае для нахождения максимального числа также нужно выполнить минимум два сравнения. Алгоритм выбора в виде условного оператора с вложенными в него двумя другими условными операторами можно легко пояснить следующей блок-схеме:



Несмотря на то, что выполняется всего одна инструкция вывода, при написании кода все ветвления будем помещать в отдельный составной оператор. Это значит, что при движении от более общего уровня к частному все конструкции нужно смещать на два пробела относительно родительского блока/оператора.

Код программы:

```
program MaxOfThree;

var
   a, b, c: integer;

begin
   readln(a, b, c);
   if a > b then begin
      if a > c then begin
     writeln(a)
   end
   else begin
     writeln(c)
   end
end
else begin
   if b > c then begin
     writeln(b)
   end
   else begin
     writeln(c)
   end
end
end.
```

Задача 2.

Посчитать количество единичных битов числа.

Формулировка. Дано натуральное число меньше 16. Посчитать количество его единичных битов. Например, если дано число 9, запись которого в двоичной системе счисления равна 1001_2, то количество его единичных битов равно 2

Решение.

Необходима переменная для ввода с клавиатуры. Обозначим ее как n. Так как
нужно накапливать количество найденных битов, то возникает потребность в еще одной переменной. Обозначим ее как count («count» в переводе с англ. означает «считать», «подсчет» и т. д.).

Переменные возьмем типа byte (они могут принимать значения от 0 до 255), и пусть в данном случае такой объем избыточен, но это не принципиально важно.

Число вводится в десятичной системе счисления, однако его не нужно переводить в двоичную систему, используя правило:

Остаток от деления любого десятичного числа x на число p дает нам разряд единиц числа x (его крайний разряд справа) в системе счисления с основанием p.

То есть, разделив некоторое десятичное число, например, на 10, в остатке получаем разряд единиц этого числа в системе счисления с основанием 10. Возьмем, например, число 3468. Остаток от деления его на 10 равен 8, то есть разряду единиц этого числа.

Данные правила применимы и в арифметике в других системах счисления, и
в том числе в двоичной системе.

В двоичной записи числа нет цифр, кроме 0 и 1, поэтому сложив все разряды двоичного числа, получаем количество его единичных битов.

Это значит, что вместо проверки значений разрядов двоичного представления числа можно прибавлять к так называемому счетчику разрядов сами эти разряды – если в разряде был 0, значение счетчика не изменится, а если 1, то повысится на единицу.
В результате можно поэтапно сформировать сам алгоритм:
1) Вводим число n;
2) Обнуляем счетчик разрядов count. Это делается потому, что значения всех переменных при запуске программы считаются неопределенными, и хотя в большинстве компиляторов Pascal они обнуляются при запуске, все же считается признаком «хорошего тона» в программировании обнулить значение переменной, которая будет изменяться в процессе работы без предварительного присваивания ей какого-либо значения.
3) Прибавляем к count разряд единиц в двоичной записи числа n, то есть остаток от деления n на 2:
count := count + n mod 2;
Строго говоря, можно не прибавлять предыдущее значение переменной count к остатку от деления, так как оно все равно было нулевым. Но поступаем так для того, чтобы сделать код более однородным, далее это будет видно. Учтя разряд единиц в двоичной записи n, мы должны отбросить его, чтобы исследовать число далее. Для этого разделим n на 2. На языке Pascal это будет выглядеть так: n := n div 2;
4) Теперь нам нужно еще два раза повторить п. 3 , после чего останется единственный двоичный разряд числа n, который можно просто прибавить к счетчику без каких-либо дополнений: count := count + n;
5) В результате в переменной count будет храниться количество единичных разрядов в двоичной записи исходного числа. Осталось лишь вывести ее на экран.
Программа работает правильно на всех вариантах правильных исходных данных, в чем несложно убедиться с помощью простой проверки.
Код программы:

```
program BinaryUnits;

var
  n, count: byte;

begin
  readln(n);
  count := 0;
  count := count + n mod 2;
  n := n div 2;
  count := count + n mod 2;
  n := n div 2;
  count := count + n mod 2;
  n := n div 2;
  count := count + n;
  writeln(count)
end.
```

Задача 3.

Вывести название дня недели по его номеру.

Формулировка. Вывести название дня недели по его номеру.

Решение.

Задача простейшим образом решается с помощью оператора выбора case. Данный оператор позволяет организовать ветвления в зависимости от значений некоторой переменной, для каждого из которых можно предусмотреть выполнение различных действий. Причем, если значению переменной не соответствует ни один вариант, выполняется else-блок (если он присутствует). Кстати, после перечисления всех вариантов оператора case необходимо написать ключевое слово end (ключевое слово case является еще и открывающей операторной скобкой).

Для того чтобы воспользоваться оператором case, необходимо произвести ввод номера дня недели в некоторую переменную i типа byte, и по этому номеру вывести название текущего дня недели. Благодаря else-блоку в этой программе впервые предусматривается сообщение об ошибке, связанной с некорректно введенным номером, которому не соответствует ни один из дней недели.

Кстати, в каждом из вариантов ветвлений может быть помещен составной оператор, но при описании вариантов не стали использовать операторные скобки, так как на этот раз они испортили бы все оформление кода, которое сейчас является достаточно гармоничным.

Код программы:

```
program DaysOfTheWeek;

var
  i: byte;

begin
  readln(i);
  case i of
    1: writeln('Monday');
    2: writeln('Tuesday');
    3: writeln('Wednesday');
    4: writeln('Thursday');
    5: writeln('Friday');
    6: writeln('Saturday');
    7: writeln('Sunday')
    else writeln('This day of the week does not exist!')
  end
end.
```

Задача 4.

Ниже на двух языках программирования записан некий алгоритм. Получив на вход число x, по данному алгоритму программа напечатает два числа A и B. Укажите наибольшее из таких чисел x, при вводе которых программа печатает сначала число 4, а потом 8.

Бейсик	Паскаль

DIM X, A, B AS INTEGER
INPUT X
A = 0
B = 0
WHILE X > 0
A = A + 1
IF B < 1 + (X MOD 10) THEN
B = 1 + (X MOD 10)
END IF
X = X \ 10
WEND
PRINT A
PRINT B
	var x, A, B: integer;
	begin
	readln(x);
	A := 0;
	B := 0;
	while x > 0 do begin
	A := A + 1;
	if B < 1 + (x mod 10)
	then
	B := 1 + (x mod 10);
	x := x div 10
	end;
	writeln(A); writeln(B)
	end.

Решение.

Анализ алгоритма от конечного результата: печать A =4, B=8.

1) Так как во внешнем цикле происходит последовательное суммирование 1 с A и в конце каждого шага происходит уменьшение разрядов x целочисленным делением на 10, то число x имеет четыре разряда.
2) При этом, так как B=1+(x mod 10), то старший разряд x должен быть равен 8-1=7. Если бы не было проверки условия B меньше (1+(x mod 10)), то максимальное число x=7999.
3) Но условие проверки B означает, что B в первом (и последнем) шаге внутреннего цикла должно быть равно 8, значит максимальное число x=7777.

Ответ: 7777

Задача 5.

Ниже на двух языках программирования записан некий алгоритм. Получив на вход число x, по данному алгоритму программа напечатает два числа K и R. Укажите наибольшее из таких чисел x, при вводе которых программа печатает сначала число 3, а потом 8.

Бейсик	Паскаль

DIM X, K, R, Y AS INTEGER
INPUT X
K = 0 : R = 0
WHILE X > 0
K = K + 1
IF R < X MOD 10 THEN
R = X MOD 10
END IF
IF X < 10 THEN Y = X
X = X \ 10
WEND
R = R - Y
PRINT K
PRINT R
	var x, K, R, y: integer;
	begin
	readln(x);
	K := 0; R := 0;
	while x > 0 do begin
	K := K + 1;
	if R < x mod 10 then
	R := x mod 10;
	if x < 10 then y := x;
	x := x div 10
	end;
	R := R - y;
	writeln(K); writeln(R)
	end.

Решение.

Анализ алгоритма по конечному результату: печать K =3, R=8.

1) Так как во внешнем цикле происходит последовательное суммирование 1 с K, и в конце каждого шага происходит уменьшение разрядов x целочисленным делением на 10, то число x имеет три разряда.
2) При этом, так как итоговое R=R-y, где y –значение старшего разряда x, то старший разряд x должен быть равен (R-8)=1, отсюда R=9.
3) Во внутреннем цикле проверяется условие, чтобы значение R было меньше x mod 10, а это значит, что значению R

присваивается значение остатка от деления среднего и младшего разрядов x на 10.

4) Значение R в первом и во втором шаге проверки условия должно быть равно максимально 9, значит максимальное число x=199.

Ответ: 199

Лекция (по материалам презентации)

Тема: Программирование. Основные понятия.

Введение.

«Я думаю, это неизбежно, что люди программируют плохо, и что так будет и впредь. Обучение не приведет к дальнейшему улучшению дел. Использование специализированных языков не поможет, так как люди всегда нарушают правила. Нам придется просто привыкать к этому».

Э. Йордан. Структурное программирование

Список изучаемых разделов:

- Основные термины в программировании.
- Виды языков программирования.
- Структурный подход в программировании.
- Алфавит и словарь языка программирования.
- Основные команды алгоритмов.

Словарь терминов (глоссарий)

- **Программа вычислительной машины –** это описание алгоритма решения задачи, заданное на языке вычислительной машины.
- **Язык программирования –** строго определенный набор правил, характеризующий систему алгоритмов, лежащих в основе составляемой программы.

Основные термины в программировании

1. Система программирования – это совокупность различных программ, используемых для автоматизации процессов программирования сценариев работы ЭВМ.

2. Машинно-ориентированные (или низкого уровня) системы – это системы, зависящие от особенностей архитектуры компьютеров, их иначе называют процессорными.

3. Машинно-независимые (высокого уровня) системы – это системы, которые не зависят от аппаратного построения вычислительных систем.

Виды языков программирования

Все языки программирования до недавнего времени разделялись на два основных класса: низкого уровня и высокого уровня.

Примеры языков программирования представлены на рисунке 11.

Структурный подход в программировании.

Термин «Структурный подход к программированию» означает:

1. нисходящие методы разработки программы «сверху вниз»,
2. собственно структурное программирование означает программирование по структурам.
3. Имеется так называемый структурный контроль.

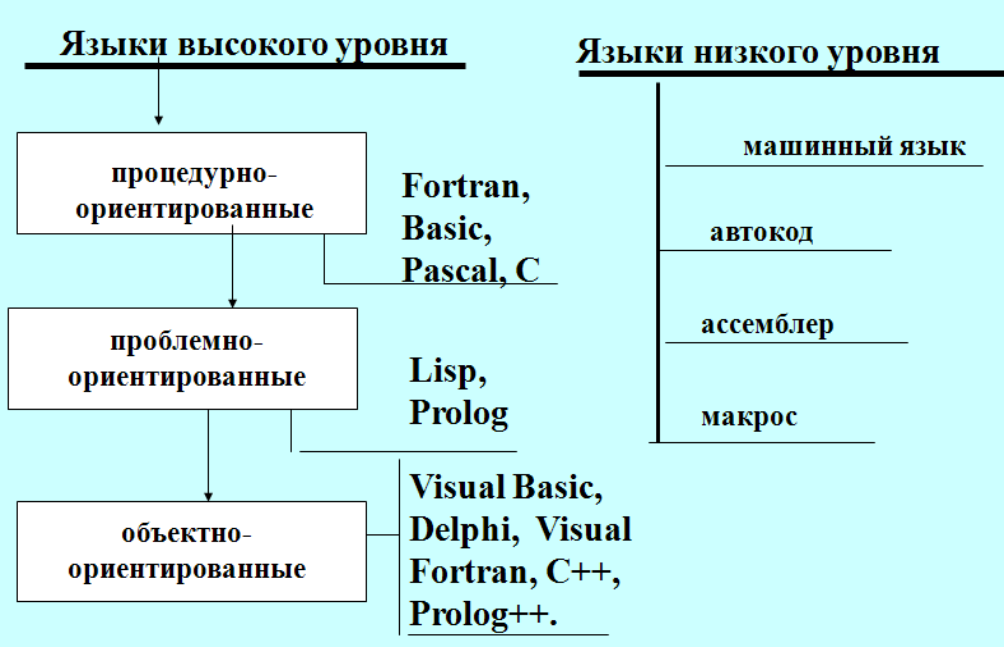


Рисунок 11 – Примеры языков программирования.

Алфавит и словарь языка программирования

Алфавит – это символы, наименованные в таблице 3:

Таблица 3 – Перечень основных символов в программировании.

Наименование символов	**Обозначение символов**
Заглавные и прописные буквы	***A - Z, a – z***
Цифры	***0 – 9***
Восклицательный и вопросительный знаки	***!, ?***
Знак вставки	^
Знаки равенства, меньше, больше	=, <, >
Левая и правая скобки	*0*
Знак подчеркивания	_
Тильда	~
Точка, запятая	. , ,
Точка с запятой	;
Двоеточие	:
Знаки плюс и минус	+, -
Знак Asterisk	*

Наклонная черта (слеш)	/
Обратная наклонная черта (обратный слеш)	\
диез	#
Знак процента	%
Знак амперсанда	&
Знак доллара	$
Двойные кавычки	«»
Апостроф (одинарная кавычка)	‘
Коммерческое at	@

Словарь языка программирования – набор сокращенных слов английского языка, которые называются ключевыми или зарезервированными словами, обозначающими выполнение определенного действия.

Многие ключевые слова вместе с дополнительными параметрами формируют операторы, из которых состоит тест программы. Ранее ключевые слова, зарезервированные в словарь языка, было нельзя использовать в программах в качестве других имен (например, переменных). В языках программирования существуют такие понятия, как например, величины и выражения и многие другие.

Величина – отдельный информационный объект, имеющий имя, значение и тип.

Выражение – запись, которая определяет последовательность действий над величинами.

Оператор (команда) присваивания – команда исполнителя, в результате которой переменная получает новое имя.

Формат команды присваивания (знаки := используются не во всех языках):

<имя переменной> := <выражение>

Напомним, что команды (операторы) алгоритма разделяют на следующие группы:

1. ввод,
2. вывод,
3. ветвление (полное и неполное),
4. цикл (с предусловием, с потусловием и с параметром).

Поэтому в языках программирования существуют аналогичные операторы, которые будем рассматривать на следующих занятиях.

Практические задачи, выполняемые на занятии.

Решения данных задач приводятся в приложении 1.

Задача 1.

Вывести на экран сообщение «Hello World!».

Формулировка. Вывести на экран сообщение «Hello World!». Некоторые учебные курсы по программированию рассматривают эту задачу как самую первую при изучении конкретного языка или основ программирования.

Задача 2.

Вывести на экран три числа в порядке, обратном вводу.

Формулировка. Вывести на экран три введенных с клавиатуры числа в порядке, обратном их. Другими словами, мы ввели с клавиатуры три числа (сначала первое, потом второе и третье), и после этого единственное, что нам нужно сделать – это вывести третье, затем второе и первое.

Задача 3.

Вывести на экран квадрат введенного числа.

Формулировка. Дано натуральное число меньше 256. Сформировать число, представляющее собой его квадрат.

Задача 4.

Получить реверсную запись трехзначного числа.

Формулировка. Сформировать число, представляющее собой реверсную (обратную в порядке следования разрядов) запись заданного трехзначного числа. Например, для числа 341 таким числом должно быть 143. В условии с клавиатуры вводится некоторое трехзначное число (трехзначными называются числа, в записи которых есть три разряда (то есть три цифры), например: 115, 263, 749 и т. д.). Необходимо получить в некоторой переменной число, которое будет представлять собой реверсную запись введенного числа. Другими словами, нужно перевернуть введенное число «задом наперед», представить результат в некоторой переменной и вывести его на экран.

Задача 5.

Вывести на экран наибольшее из двух чисел.

Формулировка. Даны два числа. Вывести на экран то из них, которое больше.

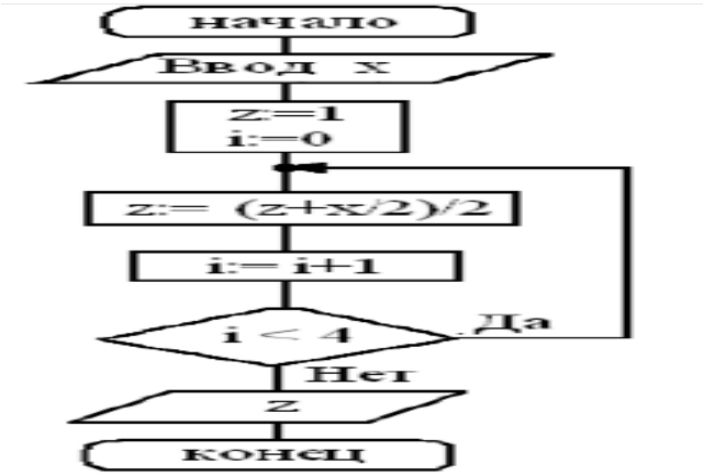
Практическое занятие 10. Операторы циклов

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 10.

Домашние задания к занятию 11.

Задача 1.

Дана блок-схема (см. рисунок). Какое значение будет иметь число z на выходе, если: x=2.



Решение.

В блок-схеме алгоритма применяется цикл с предусловием. Для решения алгоритма используем вычислительную таблицу:

Шаг	1	2	3	4
Начальное значение z	1	1	1	1
Значение i	0	1	2	3
Результат выполнения z	z = (1+2/2)/2=1	z = (1+2/2)/2=1	z = (1+2/2)/2=1	z = (1+2/2)/2=1
Результат выполнения i	i = 0+1=1	i = 1+1=2	i = 2+1=3	i = 3+1=4
Тело цикла	1 < 4 (Да)	2 < 4 (Да)	3 < 4 (Да)	4 < 4 (Нет)
Вывод z				1

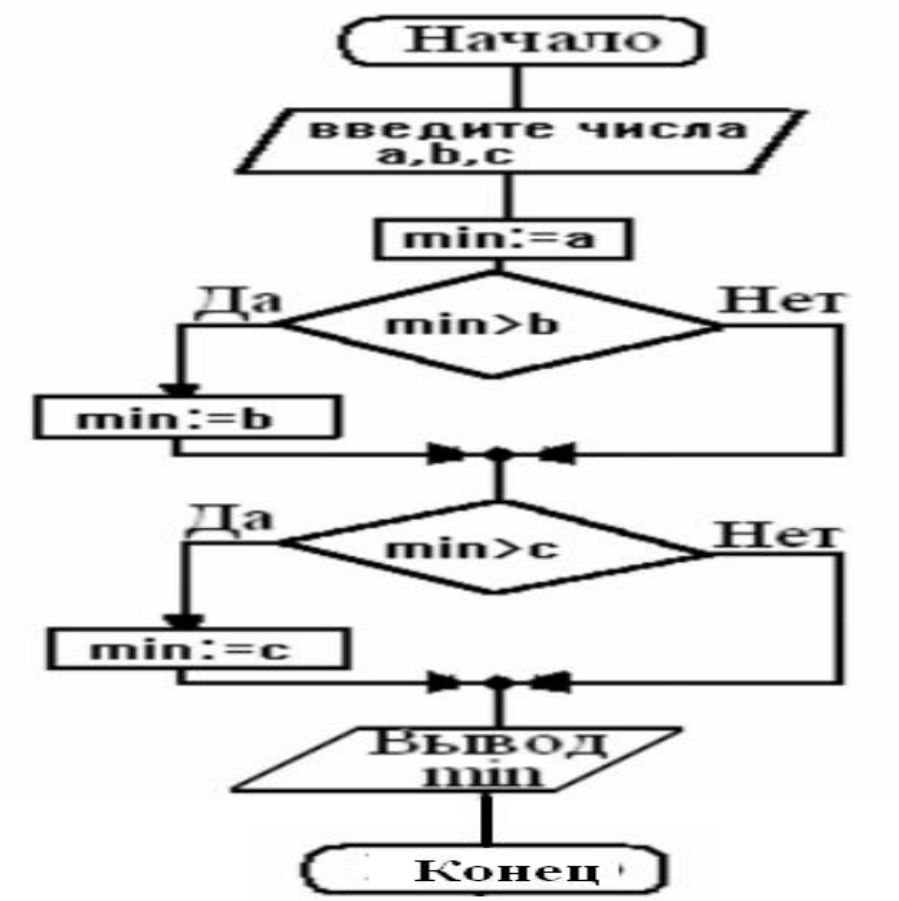
Ответ: z=1

Задача 2.

Реализовать алгоритм нахождения минимального значения min для трех переменных a, b, c в виде блок- схемы. Вывести значение переменной min, если: a=5, b=8, c=3.

Решение.

Примерная блок-схема алгоритма представлена на рисунке:



Для решения алгоритма используем вычислительную таблицу:

Шаг	**1**
Начальное значение a	**5**
Начальное значение b	**8**
Начальное значение c	**3**
Результат выполнения	**min:=5**
Проверка условия	**5>8 (Нет)**
Проверка условия	**5>3 (Да)**
Результат выполнения	**min:=3**
Результат выполнения	**Вывод min=3**

Ответ: 3

Задача 3.

Найти наибольший нетривиальный делитель натурального числа.

Формулировка. Дано натуральное число. Найти его наибольший нетривиальный делитель или вывести единицу, если такового нет.

Примечание 1: делителем натурального числа a называется натуральное число b, на которое a делится без остатка. То есть выражение «b – делитель a» означает: a / b = k, причем k – натуральное число.
Примечание 2: нетривиальным делителем называется делитель, который отличен от 1 и от самого числа (так как на единицу и само на себя делится любое натуральное число).
Решение.
Пусть ввод с клавиатуры осуществляется в переменную n. Можно решить задачу перебором чисел. Для этого возьмем число на единицу меньшее n и проверим, делится ли n на него. Если да, то выводим результат и выходим из цикла с помощью оператора break. Если нет, то снова уменьшаем число на 1 и продолжаем проверку. Если у числа нет нетривиальных делителей, то на каком-то шаге проверка дойдет до единицы, на которую число гарантированно поделится, после чего будет выдан соответствующий условию ответ.
Но следовало бы начать проверку с числа, равного n div 2 (чтобы отбросить дробную часть при делении, если n нечетно), так как ни одно натуральное число не имеет делителей больших, чем половина этого числа. В противном случае частное от деления должно быть натуральным числом между 1 и 2, которого просто не существует.
Данная задача также решается через for, но через другую его разновидность, и теперь счетчик будет убывать от n div 2 до 1 Для этого do заменится на downto, при позиции начального и конечного значений остаются теми же.
Алгоритм на естественном языке:
1) Ввод n;
2) Запуск цикла, при котором i изменяется от n div 2 до 1 В цикле:
1 Если n делится на i (то есть, остаток от деления числа n на i равен 0), то выводим i на экран и выходим из цикла с помощью break.
Код программы:

```
program GreatestDiv;

var
  i, n: word;

begin
  readln(n);
  for i := n div 2 downto 1 do begin
    if n mod i = 0 then begin
      writeln(i);
      break
    end
  end
end.
```

Кстати, у оператора ветвления if в цикле отсутствует else-блок. Такой условный оператор называется оператором ветвления с одной ветвью.

Задача 4.

Проверить, начинается ли каждый из членов последовательности с десятичной цифры, на которую оканчивается предыдущий член последовательности.

Формулировка. Дана последовательность натуральных чисел, ограниченная вводом нуля. Проверить, начинается ли каждый из ее членов (со второго) с десятичной цифры, на которую оканчивается предыдущий член. Например, таковой последовательностью будет являться 14 47 712 2179 9 9 93 0 (также сохранен ограничивающий ноль).

Решение.

Задача решается через цикл с предусловием, что характерно для задач на последовательность с ограничителем. При ее решении мы могли бы не рассматривать «вырожденные варианты», в которых на вход будет подаваться, например, пустая последовательность (то есть состоящая из единственного нуля) или последовательность из одного члена, так как на вопрос о том, удовлетворяют ли такие последовательности заданному критерию, ответить теоретически достаточно трудно. Разумнее всего было бы считать выполнение критерия для таких последовательностей неопределенным, что, однако, в формате даваемого нами ответа не представляется возможным: под «проверкой» характеристического свойства в данном случае понимается ответ на вопрос: либо «да», и последовательность отвечает заданным требованиям, либо «нет», и, соответственно, не отвечает.

Следовательно, нужно вместо ответа о «неопределенности» проверяемого свойства дать один из допустимых ответов. Наверное, разумно было бы дать при обработке вырожденных случаев

программой ответ «нет». Мы возьмем это на заметку и попытаемся сделать контроль исключений, когда уже будет готово решение задачи для общего случая, чтобы заранее не наделать ошибок. Это необходимо потому, что мы попробуем инициализировать значения переменных при запуске программы таким образом, чтобы обработку вырожденных случаев можно было выполнить с минимальным вложением дополнительного кода.

Так как нам необходимо проверять выполнение критерия на двойках (парах) элементов, то хранить в памяти нужно сразу два элемента (a и b). Первый элемент не имеет предшественника, поэтому после ввода мы не обрабатываем его и можем вводить следующий элемент в одном операторе с ним: **read(a, b);**

Имея два элемента, мы уже можем выполнить проверку нашего свойства. Однако уже на этом этапе мы можем догадаться, что в силу необходимости выполнить проверку для всех пар элементов последовательности необходимо сразу поместить ее в цикл. Мы будем считывать каждый очередной член последовательности в переменную b, и так как последнее вводимое число по условию – 0, то предусловием цикла будет b < > 0 (так как при b = 0 цикл должен прекратиться):

```
while b <> 0 do begin
...
a := b;
read(b)
end;
```

На месте троеточия будет располагаться код проверки каждой пары, полностью охватывающий определение нашего свойства. Примечание: в «шаблоне» основного цикла мы выделили также оператор a := b, который обеспечивает движение по каждым двум соседним элементам последовательности. Следует обратить внимание на то, что мы должны проверять выполнение нашего свойства для 1-го и 2-го, 2-го и 3-го, 3-го и 4-го и т. д. элементов последовательности.

После выполненной проверки для двух элементов, например, для 1-го (который хранится в переменной a) и 2-го (который хранится в переменной b) присваиваем переменной a значение переменной b (в которой у нас хранился 2-ой элемент), затем считываем в b 3-й элемент, чтобы проверить свойство для 2-го и 3-го элементов и т. п.

При этом нужно понимать, что мы не можем считывать в цикле сразу две переменные, так как при таком подходе проверка будет выполняться лишь для 1-го и 2-го, 3-го и 4-го и т. д. элементов, что неверно.

Разбор проверки: Так как на каждом шаге цикла программе требуется выяснить, начинается ли следующий член последовательности с десятичной цифры данного, то, имея данный член в переменной a и следующий член в b, мы должны сравнить последнюю цифру a (обозначим ее как last) с первой цифрой b (обозначим ее как first). Сделать это можно так:

```
last := a mod 10;
first := b;
while first > 9 do begin
first := first div 10
end;
```

Здесь мы сначала взяли последнюю цифру a (строка 1), затем скопировали в last значение b (переменную b нельзя изменять, так как ее значение понадобится нам на следующем шаге цикла) и во вложенном цикле разделили last на 10 столько раз, чтобы в ней осталась лишь одна цифра, кото
рая является его первой цифрой.

Взяв нужные цифры, мы можем выполнить их сравнение и выйти из цикла в том случае, когда они не равны, так как при этом нарушается наше характеристическое свойство, данное в условии, и после этого дальнейшая проверка бессмысленна:

```
if last <> first then break;
```

Когда цикл завершится, нам останется лишь вывести на экран результат сравнения переменных last и first: если цикл завершился, то последовательность отвечает заданному свойству (так как не было выхода через break), они будут равны и будет выведен ответ true; если же был совершен выход через break, то переменные неравны и ответ, соответственно, false. Теперь попробуем оптимизировать программу для обработки вырожденных случаев для пустой последовательности (когда вводится единственный 0) и для последовательности из одного члена (когда вводится некоторое число и 0): мы договорились выводить для них ответ false. Очевидно, в данный момент наша программа обрабатывает корректно минимальный случай из двух членов: тогда проходит одно повторение тела цикла, в котором переменные last и first получают значения, затем может произойти выход по break или завершение цикла по вводу нуля, как и должно быть.

Однако если мы введем последовательность из одного члена, то при вводе a и b в переменную a пойдет этот член, а в b окажется ограничивающий ноль, что приведет к невыполнению входа в основной цикл и программа перейдет к оператору вывода writeln(last =

first), что неверно, так как значение переменных last и first в данный момент будет не определено и выражение в операторе вывода может дать любой результат. Это значит, что для избегания подобного исхода необходимо выполнить инициализацию переменных last и first заведомо неравными значениями, чтобы получить гарантированный ответ false при вводе последовательности из одного члена. **Это можно сделать так:**

first := 1; last := 0;

Если ввести последовательность, состоящую из одного нуля, то в данной программе это невозможно, так как оператор ввода в начале содержит две переменные, и если мы сразу введем 0, то программа «зависнет» в ожидании ввода второго числа. Чтобы избежать этого, мы должны вводить одно число в переменную a, и если оно не равно 0, нужно ввести b. Вместе с этим необходимо заранее присвоить переменной b число 0, так как она была определена в случае последовательности из одного члена, чтобы не осуществился вход в основной цикл:

read(a); b := 0; if a <> 0 then read(b);

Эта конструкция заменит оператор read(a, b), который мы описывали в самом начале решения задачи.

Код программы:

```
program LastAndFirst;

var
  a, b, first, last: word;

begin
  first := 1;
  last := 0;
  read(a);
  b := 0;
  if a <> 0 then read(b);
  while b <> 0 do begin
    last := a mod 10;
    first := b;
    while first > 9 do begin
      first := first div 10
    end;
    if last <> first then break;
    a := b;
    read(b)
  end;
  writeln(last = first)
end.
```

Задача 5.

Проверить, является ли натуральное число степенью двойки

Формулировка. Дано натуральное число n. Проверить, представляет ли оно собой натуральную степень числа 2.

Решение.

В общем случае нужно ответить на вопрос: можно ли возвести число 2 в какую либо натуральную степень (или в нулевую степень, так как $2^0 = 1$), чтобы получилось число n?

Вообще, для решения этой задачи существует достаточно красивое равенство, выполняющееся для всех натуральных степеней числа 2, которое позволяет получить ответ с помощью одной единственной логической побитовой операции:

n and (n – 1) = 0.

Обозначим ее как (1).

Дело в том, что натуральная степень числа 2 с показателем p в двоичном виде всегда представляется как единица с p нулями справа. Это происходит потому, что двоичная запись этого числа в десятичном виде представляется как $1 * 2^p + 0 * 2^{p-1} + ... + 0 * 2^1 + 0 * 2^0$, где все пропущенные слагаемые имеют коэффициент 0, и из этой записи легко восстановить двоичное представление: 10...00, здесь нулей всего p. Поэтому если мы отнимем от любой степени двойки 1, то получим число 1...11, где всего p единиц (точнее говоря, это будет число 01...11). В итоге, если мы применим к этим двум числа битовую конъюнкцию, то всегда будем получать результирующее число, равное 0.

Примечание: побитовая конъюнкция – это бинарная операция, которая эквивалента обычной конъюнкции, примененной к двоичным разрядам операндов (двух исходных чисел), стоящим на одинаковых позициях в двоичных представлениях этих чисел. При этом результатом применения побитовой конъюнкции является некое результирующее число, значение соответствующих битов которого зависит от значений битов исходных чисел: в соответствующем разряде будет находиться 1 тогда и только тогда, когда на этих позициях в обоих исходных числах стояли единичные биты, и иначе нули.

Пример: выполнить поразрядную конъюнкцию двоичных чисел 0110012 и 1010112 (при этом выпишем их так, чтобы соответствующие двоичные разряды стояли друг под другом):

Первый операнд: 011001_2

Второй операнд: 101011_2

Результат: 001001_2

Так как 1-й разряд слева у первого числа равен 0, а у второго – 1, то в соответствующий первый разряд результата идет бит 0 2-е разряды, соответственно, равны 1 и 0, и в результат снова идет бит 0 3-и разряды у обоих чисел равны 1, поэтому в 3-й разряд результата

Кстати, наша формула (1) пропускает число 0 в качестве степени двойки. Так как компиляторы языка Pascal (гарантированно называются Borland Delphi 7 и Pascal ABC) реализуют числовые типы данных в виде кольцевых отрезков (то есть, например, в типе byte после числа 255 следует число 0, а перед числом 0 – число 255), то в любом таком типе выражение (0 – 1) имеет некоторое ненулевое битовое представление (так как нулевое битовое представление имеет лишь число 0), а побитовая конъюнкция числа 0 и любого другого числа дает в результате число 0

Вообще, так как нам данное нам n является натуральным числом, число 0 вводиться не будет. Однако покажем, как отсечь 0 при проверке числа по формуле (1): можно осуществить проверку введенного числа на равенство нулю, и в случае равенства его заменить на какое-либо другое число, заведомо не являющееся степенью двойки, чтобы условие формулы (1) было правильным: **if n = 0 then n := 3;**

Вообще, формула (1) требует доказательства в обе стороны: мы лишь доказали, что если n является степенью двойки, то есть $n = 2^p$ (где p – любое натуральное число или 0), то выражение n and (n – 1) гарантированно дает результат 0. Покажем это схематически еще раз:

Первый операнд:100...00
Второй операнд: 011...11
Результат: 000...00

Однако мы также должны доказать, что никакое другое число n, кроме как степень двойки, не может дать 0 в результате выполнения операции n and (n – 1). Однако мы примем это утверждение без доказательства. В итоге, тело программы может выглядеть так (для натурального n, которое также может быть нулем):

```
readln(n);
if n = 0 then n := 3;
writeln(n and (n – 1) = 0);
```

но в качестве основного решения возьмем более простую идею: пусть данное число n является степенью двойки. Следовательно, его можно представить так: $2^p = 1 * 2 * 2 * ... * 2$ (здесь ровно p двоек). Разделив это выражение на 2 определенное количество раз, в результате мы получим число 1

Если же число n не является степенью двойки, то на некотором шаге мы получим остаток при делении на 2 В связи с этим имеем алгоритм:

1) Вводим n;
2) В цикле с предусловием n > 1 работаем с n:

1.Если остаток от деления n на 2 равен 1 (n mod 2 = 1), то выходим из цикла;
2. Делим n на 2 (n := n div 2);
3) Выводим на экран значение выражения n = 1 (если цикл завершился, то это условие истинно, и n – степень двойки, а если нет – то на каком-то шаге мы получили остаток при делении на 2 и вышли через break);
Даже если ввести n, равное 0, то программа выдаст правильный ответ, так как не будет осуществлен вход в цикл (2) и на шаге (3) будет выведено значение выражения 0 = 1, равное false.

Код программы:

```
program PowerOfTwo;

var
  n: integer;

begin
  readln(n);
  while n > 1 do begin
    if n mod 2 = 1 then break;
    n := n div 2
  end;
  writeln(n = 1)
end.
```

Задача 6.

Исполнитель М17 преобразует число, записанное на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Умножить на 3

Первая из них увеличивает число на экране на 1, вторая увеличивает его на
2, третья умножает на 3.
Программа для исполнителя М17 – это последовательность команд. Сколько существует таких программ, которые преобразуют исходное число 2 в число 14, и при этом траектория вычислений программы содержит числа 8 и 10? Траектория должна содержать оба указанных числа.
Траектория вычислений программы – это последовательность результатов

выполнения всех команд программы. Например, для программы **132** при исходном числе 7 траектория будет состоять из чисел 8, 24, 26.

Решение.

Общий алгоритм решения с использованием рекуррентных формул. Для составления рекуррентных формул переменим заданные программы на обратные и запомним, что для получения числа самого из себя существует

ровно одна программа, которая ничего не делает.

Условие: ***1. Прибавить 1***, меняем на ***вычти 1*** или кратко (N-1).

Условие: 2***. Прибавить 2***, меняем на ***вычти 2*** или кратко (N-2).

Условие: 3***. Умножить на 3***, меняем на ***раздели на 3*** или кратко (N/3).

Исходя из этого, составляем следующие рекуррентные формулы:

если из заданного числа N можно вычесть только 1, то F(n) = F (n-1)

если заданное число N не делится на 3, а из него можно вычесть 1 и 2, то F(n) = F (n-1)+F(n-2),

если к тому же заданное число N еще делится на 3, то F(n) = F(n/3)+F(n-1)+ +F(n-2).

Таким образом, общая рекуррентная формула имеет следующий вид: F(n)=F(n/3)+F(n-1)+F(n-2)

Заполняем следующую таблицу:

Числа F(n)	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Программы	**1**	**1**	**2**	**3**	**6**	**9**	**15**

Напротив числа 2 сразу же ставим 1, поскольку помним правило: «число из самого себя можно получить с помощью одной программы». После чего берем число 3. Пользуемся формулой F(n) = F(n-1), где вместо n, подставив число 3, получаем F(3) = F(3-1) = F(2) = 1. Напротив 3 ставим 1. Переходим к следующему числу 4, оно не делится на 3, поэтому берем формулу: F(n) = F(n-2) + F(n-1) = F(2)+F(3) = 1+1 = 2. Первое число, которое выполняется по третьему условию – 6, поэтому: F(6) = F(6/3) + F(6-1)+ F(6-12= F(3) + F(5)+ F(4)= 1+2+3=6. Аналогично: F(7) =9, F(8) =15.

Так как траектория вычислений программы содержит число 8, то заполняем уже другую таблицу, считая, что первое число F(8) содержит 15 программ:

Числа F(n)	**8**	**9**	**10**
Программы	**15**	**15**	**30**

По общей рекуррентной формуле F(9)= F(9-1)= F(8)=15, F(10)= F(10-1)+ F(10-2)= F(9)+ F(8)=15+15=30.

Так как траектория вычислений программы содержит число 10, то заполняем уже другую таблицу, считая, что первое число F(10) содержит 30 программ:

Числа F(n)	**10**	**11**	**12**	**13**	**14**
Программы	**30**	**30**	**60**	**90**	**150**

По общей рекуррентной формуле F(11)= F(11-1)= F(10)=30, F(12)= F(12-1)+ F(12-2)= F(11)+ F(10)=30+30=60, F(13)= F(13-1)+ F(13-2)= F(12)+ F(11)=30+60=90, F(14)= F(14-1)+ F(14-2)= F(13)+ F(12)=60+90=150.

Ответ: 150

Лекция (по материалам презентации)

Тема: Операторы циклов.

Введение.

Циклические алгоритмы используют условные операторы, которые иначе называют операторами циклов.

Операторы циклов (или повторений) – это операторы, которые обеспечивают повторение одних и тех же действий.

Список изучаемых разделов:

- **Оператор цикла с параметром.**
- **Оператор цикла с предусловием.**
- **Оператор цикла с постусловием.**
- **Вложенные циклы**

Словарь терминов (глоссарий)

- **Оператор цикла с параметром** применяется в тех случаях, когда в программе некоторые действия повторяются известное количество раз с постоянным шагом.
- **Оператор цикла с предусловием (цикл – пока)** применим, когда требуемое количество повторений заранее неизвестно. Цикл выполняется до тех пор, пока заданное ложное условие не станет истинным.
- **Оператор цикла с постусловием (цикл – до)** применим, когда требуемое количество повторений заранее неизвестно. Цикл выполняется, пока истинное условие не станет ложными, проверка производится после выполнения тела цикла.

Оператор цикла с параметром

Формат применения данного оператора обычно имеет вид:

Начало цикла: для <параметр цикла> от <начальное значение> до <конечное значение> <тело оператора цикла> конец цикла,

где <параметр цикла> - имя переменной, являющейся параметром цикла, значение которой меняется при повторении действий,

<начальное значение> и <конечное значение> - соответствующие начальные (меньшее) и конечное (как правило, большее), значение данной переменной. Здесь предполагается, что изменение происходит с шагом, равным 1, а <тело оператора цикла> - это повторяющиеся операторы.

Блок-схема, в общем виде использующаяся данный оператор, представлена на рисунке 12.

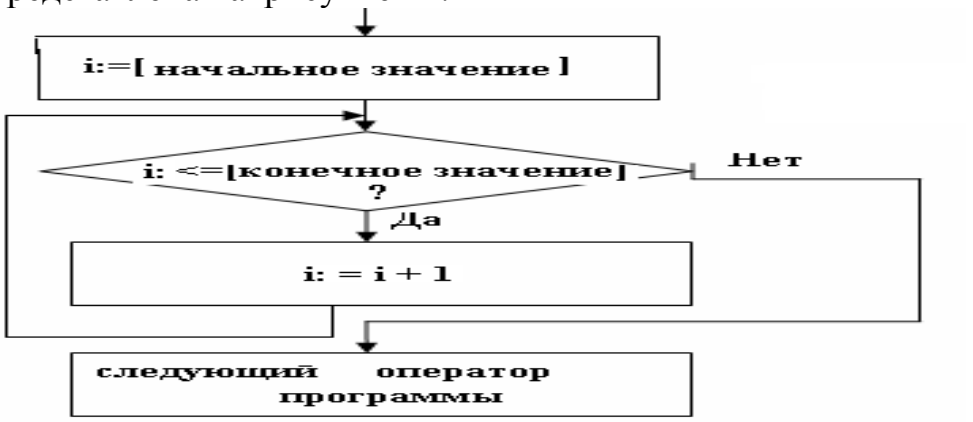


Рисунок 12 – Блок-схема цикла с оператором - параметром.

Для языка программирования PASCAL цикл с параметром I имеет следующие основные форматы:

for I:=Iн to Iк do <тело цикла> или **for I:=Iн down to Iк do <тело цикла>,**

где Iн, Iк соответствующие начальные и конечные значения параметра цикла.

Пример. Вывести целые числа от 10 до 20 на экран компьютера.

Решение. Используем оператор с параметром i, при этом начальное значение оператора i равно 10, конечное 20. Блок-схема представлена на рисунке 13, при этом считаем, что, параметр цикла i – целое число.

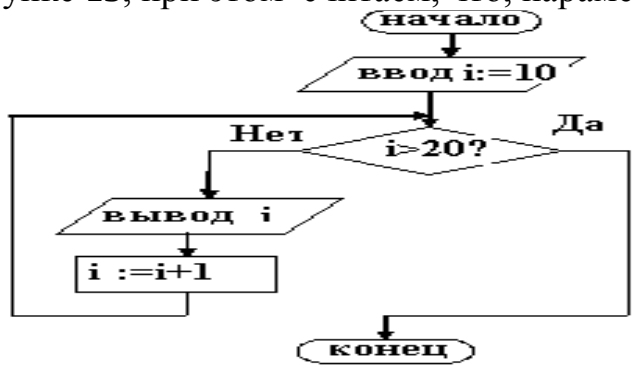


Рисунок 13 – Блок-схема алгоритма решения примера.

Оператор цикла с предусловием

Формат применения оператора обычно имеет вид:

Начало цикла: пока <условие> <тело оператора цикла> конец цикла,
где <условие> – условие, при котором выполняется <тело оператора цикла>. На рисунке 14 показана блок-схема данного цикла.

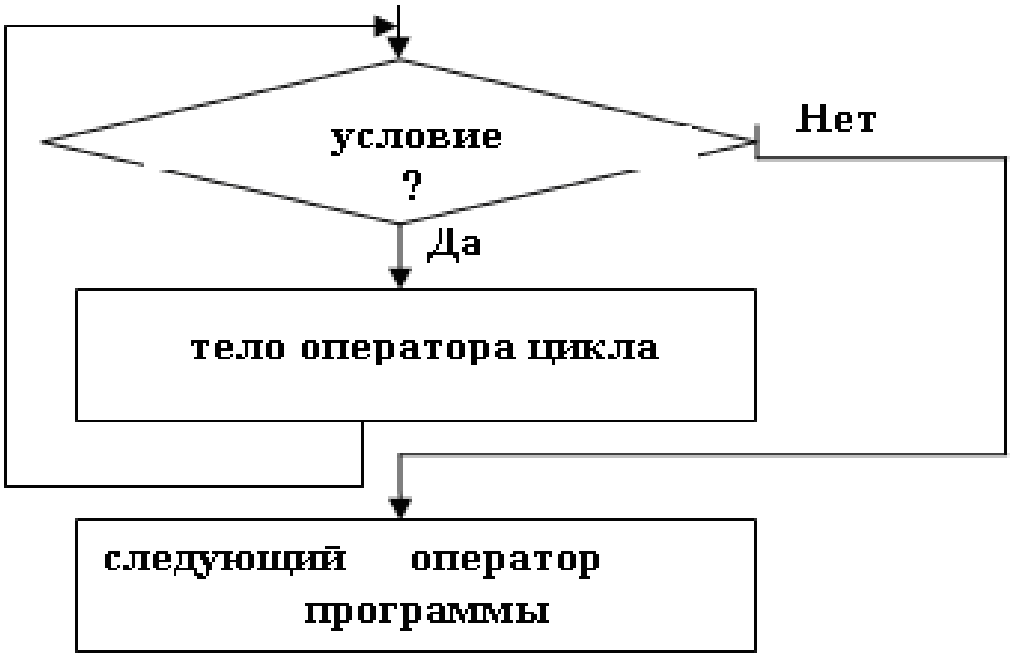


Рисунок 14 – Блок-схема цикла с оператором-предусловием.
В языках программирования данный оператор может использовать ключевое слово WHILE, например, для Basic формат такого оператора имеет вид:
DO …LOOP WHILE <условие>
Для языка PASCAL формат цикла имеет вид:
While < условие – логическое выражение> do <тело цикла>.

Правило нахождения условия, применяемого е в данном операторе:

- Определите действия, при котором нужно выполнять повторяющиеся действия;
- запишите в операторе цикла условие, противоположное найденному действию.

Оператор цикла с постусловием

Формат применения оператора обычно имеет вид:
Начало цикла: <тело оператора цикла> при <условие> конец цикла,
где <условие> – условие, при котором прекращает действие <тело оператора цикла>.
На рисунке 15 показана блок-схема данного цикла.

В языках программирования данный оператор может использовать ключевое слово **UNTIL**, например, для Basic формат такого оператора имеет вид:

DO UNTIL <условие>….LOOP.

Для языка PASCAL формат цикла имеет вид:

repeat <тело цикла> until < условие – логическое выражение>

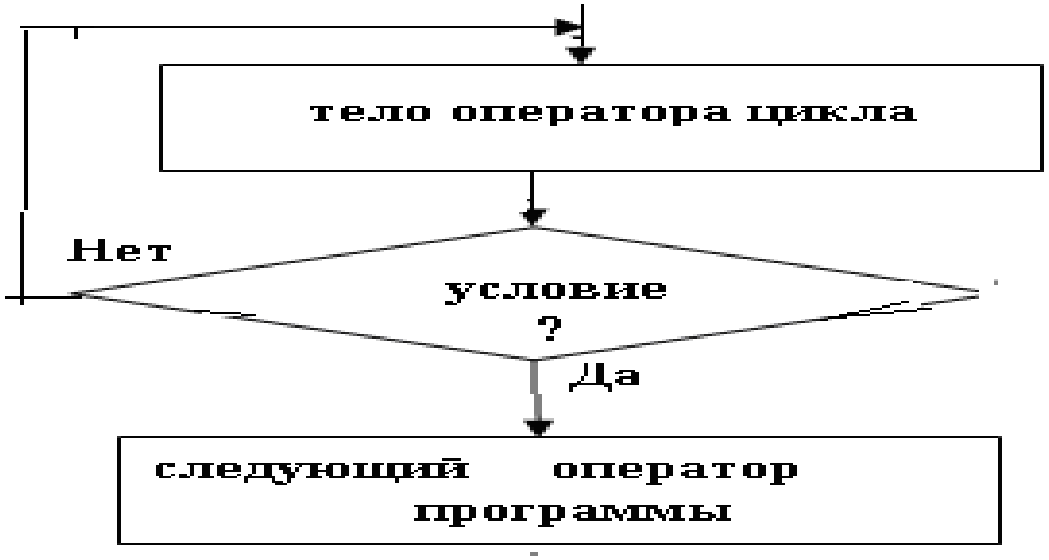


Рисунок 15 – Блок-схема цикла с оператором-постусловием.

Вложенные циклы.

Правила использования циклов:

- не переопределять переменную параметра (счетчика) внутри цикла – т. е. не присваивать ей других значений во время выполнения цикла;
- после завершения цикла значение переменной параметра должно быть равно сумме конечного значения и величины шага;
- не передавать управление внутрь цикла.

Правила использования вложенных циклов:

- внутренний цикл должен заканчиваться раньше, чем внешний;
- управление не должно передаваться сразу в середину цикла, так как в этом случае переменная параметра становится неопределенной.

Операторы циклов с предусловием и постусловием легко взаимозаменяемы. Оператор цикла с параметром заменим операторами цикла с предусловием и с постусловием, но обратная замена на такой оператор не всегда возможна.

Примеры использования циклов: организация временной паузы (пустого цикла), проверка правильности вводимых в алгоритм

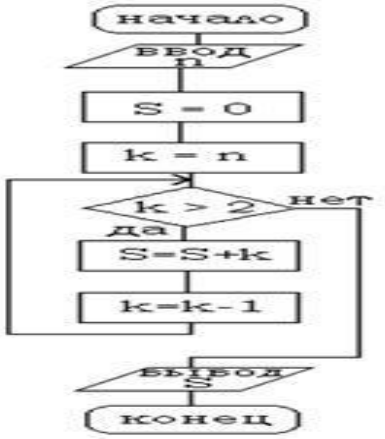
значений, получение паузы для приостановки программы до нажатия любой клавиши и т. д.

Практические задачи, выполняемые на занятии.

Решения данных задач приводятся в приложении 1.

Задача 1.

Дана блок-схема алгоритма (см. рисунок). Определить результат выполнения алгоритма при значениях исходных данных: n=4. Какой вид операторов циклов использует данный алгоритм?

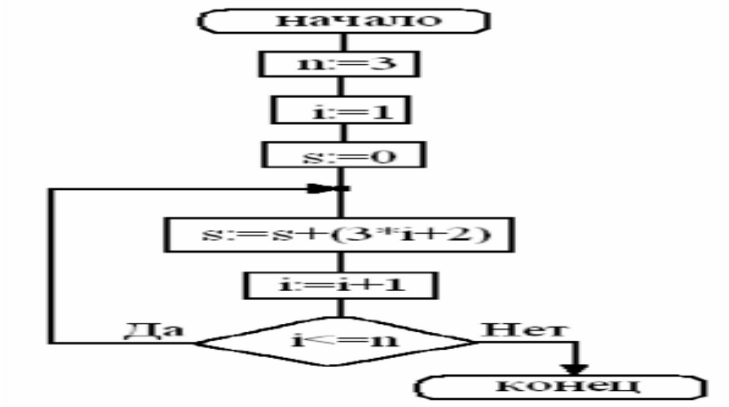


Задача 2.

Алгоритм Евклида, определяющий наибольший общий делитель (НОД) для двух натуральных чисел A и B представить в виде блок-схемы, использующий цикл с предусловием. Найти НОД = A на выходе блок- схемы, если: A=5, B=10.

Задача 3.

Дана блок-схема (см. рисунок). Тогда после исполнения алгоритма значение переменной S равно …Какой цикл используют в блок-схеме алгоритма?



Задача 4.

Вывести на экран все натуральные числа до заданного значения.

Формулировка. Дано натуральное число. Вывести на экран все натуральные числа до заданного значения.

Задача 5.

Проверить, являются ли два натуральных числа дружественными

Формулировка. Даны два натуральных числа. Проверить, являются ли они дружественными.

Примечание: дружественными числами называются два различных натуральных числа, для которых сумма всех собственных делителей первого числа равна второму числу и сумма всех собственных делителей второго числа равна первому числу.

Например, 220 и 284 – пара дружественных чисел, потому что:

Сумма собственных делителей 220: 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284

Сумма собственных делителей 284: 1 + 2 + 4 + 71 + 142 = 220

Практическое занятие 11. Анализ алгоритмов, содержащих циклы и ветвления

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 11. С данного занятия в качестве домашнего задания даются задачи демонстрационного варианта ЕГЭ за 2018 год.

Домашние задания к занятию 12.

Задача 1.

Сколько существует целых чисел x, для которых выполняется неравенство $2A_{16}<x<61_8$? В ответе укажите только количество чисел, сами числа писать не нужно.

Решение.

Шаг 1. Переводим численные значения интервала в десятичную систему счисления: $2A_{16}= 2*16^1+10*16^0=42_{10}$, $61_8=6*8^1+1*8^0=49_{10}$.

Значит, неравенство имеет вид: $42_{10}<x<49_{10}$

Шаг 2. По правилу открытого интервала таких чисел x в интервале имеется (49-43)-1=5, значит:

Ответ: 5

Задача 2.

Логическая функция F задаётся выражением $\neg x \vee y \vee (\neg z \wedge w)$. На рисунке приведён фрагмент таблицы истинности функции F, содержащий **все** наборы аргументов, при которых функция F ложна. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных w, x, y, z.

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

В ответе напишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала – буква, соответствующая первому столбцу; затем – буква, соответствующая второму столбцу, и т.д.)

Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Пример. Если бы функция была задана выражением $\neg x \vee y$, зависящим от двух переменных: x и y, и был приведён фрагмент её таблицы

истинности, содержащий все наборы аргументов, при которых функция истинна.

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

Тогда первому столбцу соответствовала бы переменная *y*, а второму столбцу – переменная *x*. В ответе следовало бы написать: *yx*.

Решение (основной алгоритм).

Шаг 1. Рассматриваем логические операции в исходном выражении $\neg x \vee y \vee (\neg z \wedge w)$.

Это дизъюнкция, значение которой ложно, если все ее члены равны 0.

Шаг 2. Отсюда все значения, согласно уравнению и значениям F=0, x =1(так как $\neg x$ =0), y = 0, а также значение $(\neg z \wedge w)$ =0, значит: *z =1 при w* =0, *z =1 при w* =1, или *z =0 при w* =0.

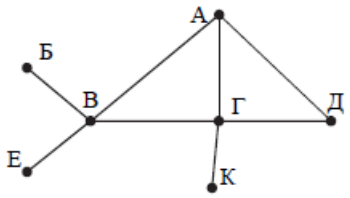
Шаг 3. Таким образом, по столбцам таблицы истинности определяем, что переменная 1 – это x, переменная 4 – y, переменная 2 – z, , переменная 3 – w.

Ответ: x z w y

Задача 3.

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

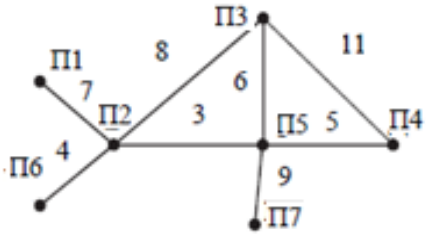
	П1	П2	П3	П4	П5	П6	П7
П1		7					
П2	7		8		3	4	
П3		8		11	6		
П4			11		5		
П5		3	6	5			9
П6		4					
П7					9		



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

Решение.

Схематически рисуем по таблице граф:



Таким образом, расстояние А – Г, это расстояние П3- П5, всего 6 км.

Ответ: 6

Задача 4.

Ниже представлены два фрагмента таблиц из базы данных о жителях микрорайона. Каждая строка таблицы 2 содержит информацию о ребёнке и об одном из его родителей. Информация представлена значением поля ID в соответствующей строке таблицы 1. Определите на основании приведенных данных, у скольких детей на момент их рождения матерям было больше 22 полных лет. При вычислении ответа учитывайте только информацию из приведенных фрагментов таблиц.

Таблица 1

ID	Фамилия_И.О.	Пол	Год_рождения
15	Петрова Н.А.	Ж	1944
22	Иваненко И.М.	М	1940
23	Иваненко М.И.	М	1968
24	Иваненко М.М.	М	1993
32	Будай А.И.	Ж	1960
33	Будай В.С.	Ж	1987
35	Будай С.С.	М	1965
42	Коладзе А.С.	Ж	1941
43	Коладзе Л.А.	М	1955
44	Родэ О.С.	М	1990
46	Родэ М.О.	М	2010
52	Ауэрман А.М.	Ж	1995
73	Антонова М.А.	Ж	1967
…	…	…	…

Таблица 2

ID_Родителя	ID_Ребёнка
22	23
42	23
23	24
73	24
22	32
42	32
32	33
35	33
15	35
32	44
35	44
23	52
73	52
…	…

Решение.

Шаг 1. По таблице 2 по ID ребенка находим ID родителя и с помощью таблицы 1 исключаем записи с полом «М» - ID отца:

ID 23 ребенка – ID 22 родителя, ID 24 ребенка – ID 23 родителя, ID 32 ребенка – ID 22 родителя, ID 23 ребенка – ID 35 родителя, ID 44 ребенка – ID 35 родителя, ID 52 ребенка – ID 23 родителя.

Шаг 2. По таблице 1 высчитываем возраст ребенка как разность года рождения ребенка и года рождения матери. Проверяем условие возраста матери на превышение ею 22 лет:

1) ID 23 ребенка – ID 42 родителя - 1968-1941=27 (подходит).
2) ID 24 ребенка – ID 73 родителя - 1993-1967=26 (подходит).

3) ID 32 ребенка – ID 42 родителя – 1960-1941=19 (не подходит).
4) ID 33 ребенка – ID 32 родителя – 1987-1960=27 (подходит).
5) ID 35 ребенка – ID 15 родителя – 1965-1944=21 (не подходит).
6) ID 44 ребенка – ID 32 родителя – 1990-1960=30 (подходит).
7) ID 52 ребенка – ID 73 родителя – 1995-1967=28 (подходит).

Шаг 3. Подсчитываем количество матерей, старших 22 лет: 7-2=5

Ответ: 5

Задача 5.

По каналу связи передаются шифрованные сообщения, содержащие только десять букв: А, Б, Е, И, К, Л, Р, С, Т, У. Для передачи используется неравномерный двоичный код. Для девяти букв используются кодовые слова.

Буква	Кодовое слово
А	00
Б	
Е	010
И	011
К	1111

Буква	Кодовое слово
Л	1101
Р	1010
С	1110
Т	1011
У	100

Укажите кратчайшее кодовое слово для буквы Б, при котором код будет удовлетворять условию Фано. Если таких кодов несколько, укажите код с наименьшим числовым значением.

Примечание. Условие Фано означает, что никакое кодовое слово не является началом другого кодового слова. Это обеспечивает возможность однозначной расшифровки закодированных сообщений.

Решение.

Шаг 1. Размерность кодов. Всего нужно закодировать 10 букв, поэтому для их кодирования избыточным двоичным кодом потребуется 4 разряда, так как $2^4=16>10$ (по формуле Хартли). Согласно таблице, число разрядов кода соответствует 4.

Шаг 2. Сравнение приведенных кодов. Так неравномерный код позволяет уменьшить число разрядов, то для решения задачи составим кодовую таблицу и сравним полученные коды с ранее принятыми неравномерными:

0	**0**	**0**	**0**
0	**0**	**0**	**1**
0	**0**	**1**	**0**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**0**	**1**
0	**1**	**1**	**0**
0	**1**	**1**	**1**

1	**0**	**0**	**0**
1	**0**	**0**	**1**
1	**0**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**1**
1	**1**	**1**	**0**
1	**1**	**1**	**1**

По исходным кодовым таблицам, очевидно, что кодирование неравномерным кодом 10 букв произведено, начиная со столбцов последних двух двоичных разрядов, постепенно прибавляя по разряду и отсекая повторяющиеся кодовые комбинации в старших разрядах, поэтому:

А – 00 (два разряда), Е – 010 (три разряда), И - 011(три разряда), У - 100 (три разряда), К -111 (три разряда), Р – 1010 (четыре разряда), Т - 1011(четыре разряда), С – 1110 (четыре разряда), Л – 1101 (четыре разряда).

Шаг 3. Определение неуказанного кода. Два разряда имеет только одна буква, три разряда – три буквы, четыре разряда должны иметь остальные шесть букв. Пять из них известны, буква Б, согласно таблице, будет иметь код: 1100.

Ответ: 1100

Практические задачи на анализ алгоритмов, содержащих циклы и ветвления.

Данные задачи позволяют значительно слушателям повысить знания на построение основных алгоритмов, применяемых в алгоритмизации и программировании типовых заданий в ЕГЭ.

Задача 1. Разветвляющий алгоритм.

Составить алгоритм решения квадратного уравнения: $a*x^2+b*x+c=0$

Решение.

Чтобы построить универсальный алгоритм, сначала требуется тщательно проанализировать математическое содержание задачи.

Решение уравнения зависит от значений коэффициентов a, b, c. Анализ задачи (по поиску вещественных корней):

если a = 0, b = 0, c = 0, то любое x — решение уравнения;

если a = 0,b = 0, c ≠ 0,то действительных решений нет;

если a = 0, b ≠ 0, то линейное уравнение имеет одно решение x=-c/b;

если a ≠ 0 и d = b2- 4ac ≥ 0, то уравнение имеет два вещественных корня (формулы приведены в блок-схеме);

если a ≠ 0 и d < 0, то уравнение не имеет вещественных корней. Блок-схема алгоритма приведена на рисунке 16:

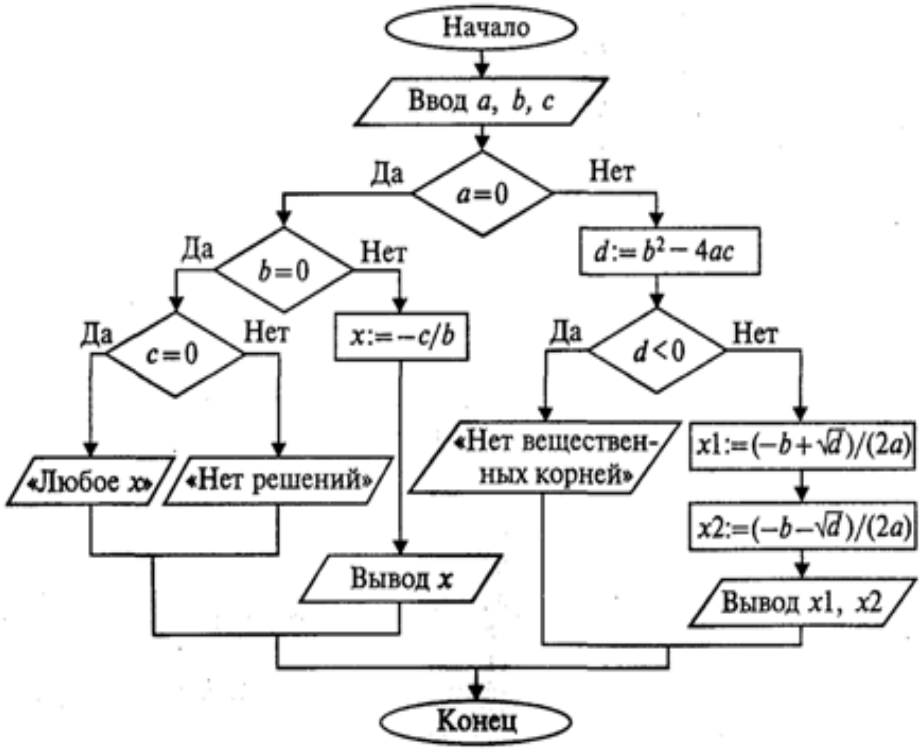


Рисунок 16- Блок-схема разветвляющегося алгоритма

В данном алгоритме многократно использована структурная команда ветвления. Общий вид команды ветвления в блок-схемах имеет вид, представленный на рисунке 17:

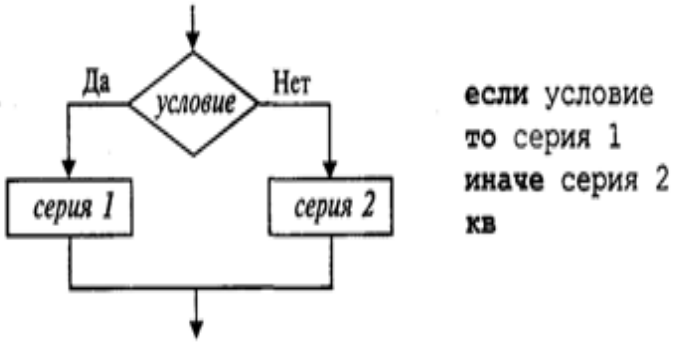


Рисунок 17 – Блок-схема структурной команды ветвления

Вначале проверяется условие (вычисляется отношение, логическое выражение). Если условие истинно, то выполняется серия 1 — последовательность команд, на которую указывает стрелка с надписью «да» (положительная ветвь). В противном случае выполняется серия 2 (отрицательная ветвь). Если на ветвях одного ветвления содержатся другие ветвления, то такой алгоритм имеет

структуру вложенных ветвлений. Именно такую структуру имеет алгоритм «Корни квадратного уравнения».

Задача 2. Полная форма структуры ветвления.

Заданы длины сторон треугольника, определить, является ли этот треугольник прямоугольным.

Полная форма структуры ветвления представления на рисунке 18:

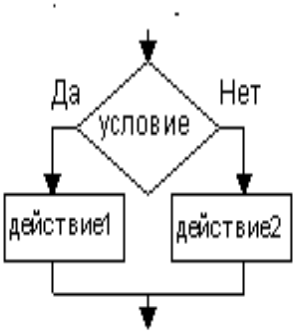


Рисунок 18 – Полная форма алгоритмической структуры ветвления.

Решение.

Код программы на языке Pascal имеет вид:

```
Program sum3;
Var a,  b, c: real;
Begin
Readln ( a, b,c );
If a*a + b*b = c*c   then writeln (‘треугольник прямоугольный’)
else writeln (‘не прямоугольный’);
Readln;
End.
```

Пример выполнения программы:

Длина первой стороны

3

Длина второй стороны

4

Длина третьей стороны

5

треугольник прямоугольный

Задача 3. Неполная форма структуры ветвления.

Найти большее число MAX из двух исходных A и B.

Неполная форма структуры ветвления представления на рисунке 19:

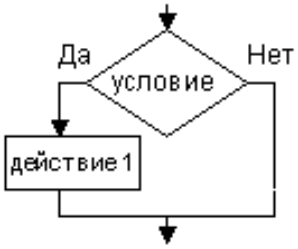


Рисунок 19 – Неполная форма алгоритмической структуры ветвления.

Неполная форма отличается от полной формы тем, что в одной из ветвей действия отсутствуют. В таком алгоритме в соответствии с условием либо будут выполнены действия, имеющиеся в ветви, либо начнут сразу выполняться действия, расположенные после ветвления**.**

Решение.

Код программы на языке Pascal имеет вид:

```
Program BID;
var   A, B,MAX: real;
begin
  readln(A, B);
MAX :=A;
   if  B>A
then  MAX:=B;

   write(MAX )
end.
```

Задача 4. Ветвление «Выбор».

Приведите введенное пользователем число от 0 до 9 к его словесному представлению.

Блок-схема алгоритмической структуры с выбором представлена на рисунке 20:

Рисунок 20 –Блок-схема алгоритмической структуры выбора.

Код программы на языке Pascal имеет вид (используем оператор if):

```
program chislo;
var n : shortint; {-128..127}
begin
    write('Введи число: ');
    readln(n);
    if n=0 then write('Нуль')
      else if n=1 then write('Один')
      else if n=2 then write('Два')
        else if n=3 then write('Три')
          else if n=4 then write('Четыре')
            else if n=5 then write('Пять')
              else if n=6 then write('Шесть')
                else if n=7 then write('Семь')
                  else if n=8 then write('Восемь')
                    else if n=9 then write('Девять')
                      else write('Это не цифра');
end;
```

Один из способов реализации множественного ветвления – проверка условий до тех пор, пока одно из них не окажется истинным, выполнение предусмотренных этим условием действий и выход из ветвления. При этом используют оператор выбора **case.**

Структура оператора выбора такова:

case <переменная> **of**<значение1>:<действия1>;<значение2>:<действия2>; <значение3>:<действия3>;……; **else** <действия 4>; **end**;

При выборе оператора в зависимости от того, какое значение принимает переменная, выполняется тот или иной блок действий. В случае если переменная не принимает ни одно из перечисленных значений, выполняется ветвь “ELSE” Но данная ветвь может и отсутствовать, в этом случае просто ничего не выполняется. У множественного ветвления есть ограничения; в роли переменной может выступать только переменная порядкового типа, однако в качестве значений можно указывать целый диапазон.

Код программы на языке Pascal имеет вид (используем оператор case):

```
program chislo;
var n : shortint;
begin
   write('Введи число: ');
   readln(n);
   case n of
    0: write('Нуль');
    1: write('Один');
    2: write('Два');
    3: write('Три');
    4: write('Четыре');
    5: write('Пять');
    6: write('Шесть');
    7: write('Семь');
    8: write('Восемь');
    9: write('Девять');
    else write('Это не цифра');
   end;
end;
```

Задача 5. Использование оператора case.

Написать программу, которая запрашивает у пользователя номер месяца и выводит соответствующее название времени года. Предусмотреть ошибку ввода.

Решение.

Код программы на языке Pascal имеет вид:

```
program nomermesaca;
var x:byte;
begin
    writeln(' введите номер месяца '); readln(x); Case x of
      1,2,12:writeln ('зима');
      3,4,5:writeln ('весна');
      6,7,8:writeln ('лето');
      9,10,11:writeln ('осень')
    else
       writeln ('ошибка');
    end;
end.
```

Задача 6. Циклические алгоритмы. Цикл с предусловием.

Вывод на печать чисел от -5 до 0 как пример циклических алгоритмов с предусловием.

Решение.

В таких циклических алгоритмах условие продолжения проверяется до обработки тела цикла, т. е. наличествует необходимость повторения обработки цикла. Блок-схема алгоритма представлена на рисунке 21:

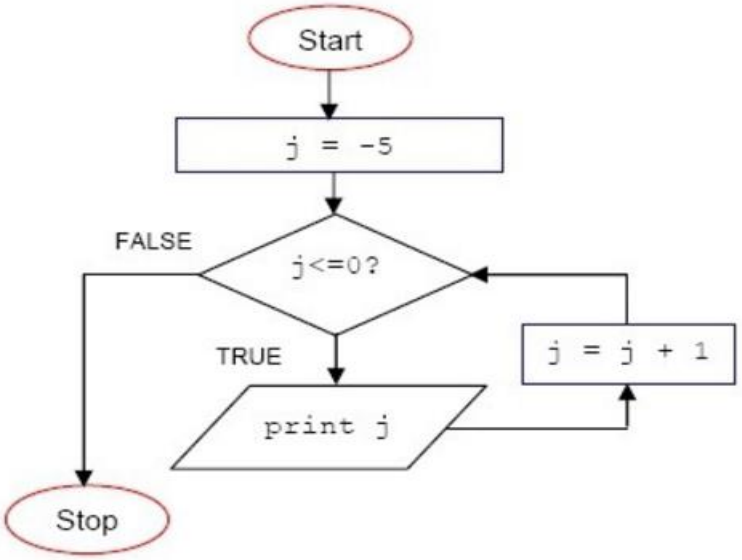


Рисунок 21 – Блок-схема циклического алгоритма с предусловием.

Элементы алгоритма: Задаем начальное значение базовой переменной j, равное -5. Проверяем условие цикла. Условие положительное, и тело цикла выполняется первый раз. Далее прибавляем к переменной j единицу, снова проверяем условие цикла. Цикл продолжает выполняться, пока значение j меньше нуля или равно ему, в противном случае выходим из цикла по ветке FALSE .

Код программы на языке Pascal имеет вид:

```
program example6;
var j: integer;
begin
j:= -5; { Присваиваем j значение -5 }
while j <= 0 do {Как только j станет равным 0, цикл прекратится
(можно было бы написать просто <, но пришлось бы добавлять 1 к
0) }
begin {Открываем операторные скобки}
write(j, ' '); {Выводим j}
inc(j); {увеличиваем j на один.}
end; { закрываем скобки }
end.
```

Задача 7. Циклические алгоритмы. Цикл с постусловием.

Расчет суммы квадратов чисел от 1 до 8 как пример циклических алгоритмов, в которых используются постусловие. Блок-схема примерного алгоритма представлена на рисунке 22:

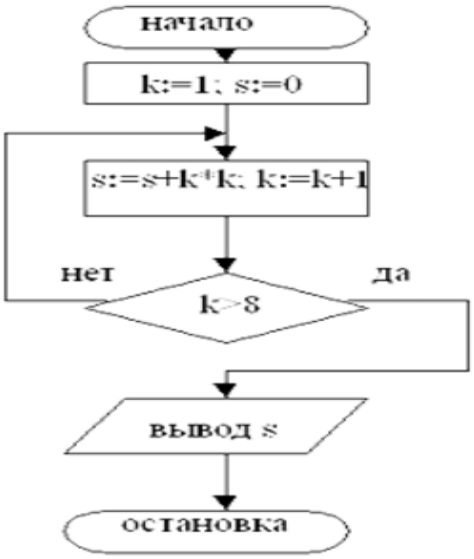


Рисунок 22 - Блок-схема алгоритма с потусловием.

Проверка условия выполняется после первой обработки тела цикла и контролирует выход из него.

Элементы алгоритма: Вводим начальное число к=1, которое используем, как счетчик цикла, и задаем нулевые начальные значения итоговой суммы s. Цикл выполняется до первой проверки условия. Проверяем условие цикла, т. е. значение счетчика к больше 8. Если результат условия отрицательный, то выполняем цикл еще раз, иначе заканчиваем цикл и выводим сумму на дисплей или печать.

Код программы на языке Pascal имеет вид:

```
PROGRAM kv_1;

VAR k , s : integer ;

BEGIN
  k:=1;
  s:=0;
  Repeat
              s : = s + k * k ;
              k : = k + 1 ;

   Until k > 8 ;
   Writeln( ' сумма равна= ' ,

Readln ;

END.
```

Задача 8. Циклические алгоритмы. Безусловный цикл (цикл с параметром).

Найти сумму элементов одномерного массива. Размер произвольный. Элементы вводятся с клавиатуры.

Обычно используется в алгоритмах, когда нужное количество выполнений цикла заранее известно, и очень часто применяется при работе с массивами. Блок-схема алгоритма приведена на рисунке 23, и данный алгоритм содержит три обязательных элемента: 1) Стартовое значение, которое называют параметром цикла, т. к. именно эта переменная изменяется после каждого выполнения цикла и определяет момент его завершения. 2) Значение, при котором цикл завершается. 3) Шаг цикла.

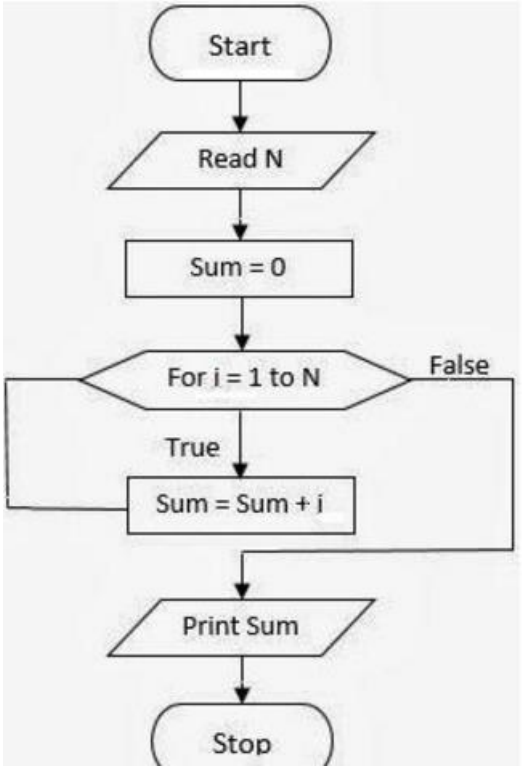


Рисунок 23 - Блок-схема алгоритма с безусловным циклом.

На каждом шаге программа проверяет, не превосходит ли стартовое значение конечное. И если да, то цикл завершается. В противном случае к стартовому значению прибавляем величину шага и цикл повторяется. Следует отметить, что любой безусловный цикл можно заменить условным с пред- или постусловием.

Код программы на языке Pascal имеет следующий вид:

```
Program summa;
Var a: array[1..100] of real; i, n: integer; s: real;
Begin
Write (‘n=’); Readln (n);  s:=0;
For i:=1 to n do
begin
```

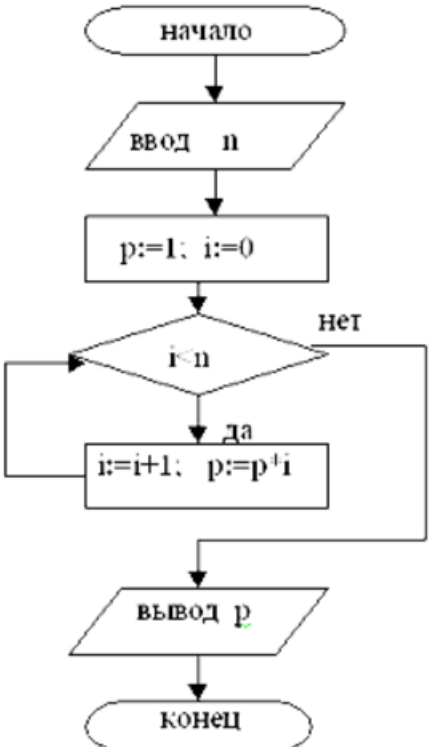
write (‘введите число’); readln (a[i]); s:=s+a[i];
end; writeln(‘сумма элементов равна ‘,s); end.

Задача 10. Циклические алгоритмы. Взаимосвязь циклов.

Вычислить n! – n-факториал(n! = 1*2*3*…* n).

Решение.

Блок – схема алгоритма и программа на языке Pascal с использованием оператора While представлена на рисунке 24:




```
PROGRAM fact_n;

VAR n , I , p : word ;

BEGIN

  Writeln('вычисление n-факто
  Write('введите n=');
  Readln(n);

  While i<n do begin
                i:=i+1;
                p:=p*i;
                end;
  Writeln('n!=',p);
  Readln;

END.
```

Рисунок 24 - Блок – схема алгоритма с оператором While.

Код программы на языке Pascal с использованием оператора for имеет следующий вид:

```
Program fact;
var n, i, s: integer;
begin
read(n); s := 1;
for i := 1 to n do s := s * i;
writeln(s);
end.
```

While — это цикл, в котором условие стоит перед телом. Причем тело цикла выполняется тогда и только тогда, когда условие истинно - true; как только условие становится ложным false, выполнение цикла прекращается. Данный цикл подходит только для одного оператора,

если же вы хотите использовать несколько операторов в своем коде, вам следует заключить их в операторные скобки —begin и end;.
Использование цикла for обычно делает код программы проще и лаконичнее. К тому же, он является не совсем обычным циклом, так как в нем нет логического условия (true и false). Поэтому цикл с параметром в программировании называют синтаксическим сахаром. Синтаксический сахар — это дополнения синтаксиса языка программирования, которые не добавляют новых возможностей, а делают использование языка более удобным.

Задача 11. Циклические алгоритмы.

Дано целое число N (> 0). Если оно является степенью числа 3, то вывести True, если не является — вывести False.

Решение.

Алгоритм: Для того чтобы решить задачу, требуется знать, что такое div и mod, и как работают логические выражения.
При этом, очевидно, что пока N делится нацело на три, делим N нацело. Затем, если N = 1 — число является степенью тройки; если N <> 1, тогда число — не степень тройки.

Код программы на языке Pascal имеет следующий вид:

```
program while4;

var
  N: integer;

begin
  readln(N);
  while N mod 3 = 0 do N := N div 3; {Пока остаток от
деления на три равен нулю, делим N нацело }
  writeln(N = 1); {логическое выражение}
end.
```

Задача 12. Циклические алгоритмы.

Даны два целых числа A и B (A < B). Вывести в порядке возрастания все целые числа, расположенные между A и B (включая сами числа A и B), а также количество N этих чисел.

Решение.

Алгоритм: так как A < B, то цикл должен будет выводить все числа от A до B. Чтобы сосчитать количество чисел, используем формулу: <конечное_значение> — <начальное_значение> + 1.

Код программы на языке Pascal имеет следующий вид:

```
program for2;

var
  A, B, i, count: integer;

begin
  read(A, B);
  for i := A to B do write(i, ' '); {выписываем числа
от меньшего к большему}
  count := B - A + 1; {считаем количество чисел}
  writeln;
  write( 'Количество чисел - ', count);
end.
```

Практическое занятие 12. Решение задач по массивам на языке программирования

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 12, которые являются задачами демонстрационного варианта ЕГЭ за 2018 год.

Домашние задания к занятию 13.

Задача 6.

На вход алгоритма подаётся натуральное число *N*. Алгоритм строит по нему
новое число *R* следующим образом.
1) Строится двоичная запись числа *N*.
2) К этой записи дописываются справа ещё два разряда по следующему
правилу:
а) складываются все цифры двоичной записи числа *N*, и остаток от деления суммы на 2 дописывается в конец числа (справа). Например, запись 11100 преобразуется в запись 111001;
б) над этой записью производятся те же действия – справа дописывается остаток от деления суммы её цифр на 2.
Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа *N*) является двоичной записью искомого числа *R*.
Укажите минимальное число *R*, которое превышает число 83 и может являться результатом работы данного алгоритма. В ответе это число запишите в десятичной системе счисления.

Решение.

Шаг 1. Переводим в двоичную систему число 83: $83_{10}=1010011_2$

Шаг 2. Согласно указанному правилу, для числа *N* результатом алгоритма будут число 10_2 при нечетном числе единиц в исходном числе, и 00_2 – при четном числе единиц. Поэтому число *N* в данном случае представляет собой число 10100.

Шаг 3. Сравниваем двоичные представления чисел, больших 83_{10} на соответствующее правило:

1) $84_{10} = 1010100_2$ – не подходит, так как для числа $N=10101_2$ с нечетным числом единиц остаток от деления не может быть равен 00_2.
2) $85_{10} = 1010101_2$ – не подходит, так как для числа $N=10101_2$ с нечетным числом единиц остаток от деления не может быть равен 01_2.

3) $86_{10} = 1010110_2$ – подходит, так как для числа $N=10101_2$ с нечетным числом единиц остаток от деления равен 00_2.

Ответ: 86.

Задача 7.

Дан фрагмент электронной таблицы. Из ячейки B3 в ячейку A4 была скопирована формула. При копировании адреса ячеек в формуле автоматически изменились. Каким стало числовое значение формулы в ячейке A4?

Примечание: знак $ обозначает абсолютную адресацию.

	A	B	C	D	E
1	1	10	100	1000	10000
2	2	20	200	2000	20000
3	3	= $C2 + D$3	300	3000	30000
4		40	400	4000	40000

Решение.

Формула в адресе ячейки B3 определяется как: =$C2+D$23. При этом, в общем случае при копировании из B3 в ячейку A4 изменяется буквенная составляющая адреса в сторону уменьшения на букву, а цифровая составляющая адреса в сторону увеличения на одну цифру. Но так как имеется абсолютная адресация в исходной ячейке, B3 то значение формулы в ячейке A4 представляет собой следующую формулу: =$C3+C$3.

Подставляем цифровые значения ячеек из таблицы:

=$C3+C$3=300+300=600.

Ответ: 600

Задача 8.

Запишите число, которое будет напечатано в результате выполнения следующей программы. Для Вашего удобства программа представлена на пяти языках программирования.

Бейсик

```
DIM S, N AS INTEGER
S = 260
N = 0
WHILE S > 0
  S = S - 15
  N = N + 2
WEND
PRINT N
```

Python

```
s = 260
n = 0
while s > 0:
    s = s - 15
    n = n + 2
print(n)
```

Алгоритмический язык

```
алг
нач
  цел n, s
  s := 260
  n := 0
  нц пока s > 0
    s := s - 15
    n := n + 2
  кц
вывод n
кон
```

Паскаль

```
var s, n: integer;
begin
  s := 260;
  n := 0;
  while s > 0 do
  begin
    s := s - 15;
    n := n + 2
  end;
  writeln(n)
end.
```

C++

```
#include <iostream>
using namespace std;


int main() {
    int s = 260, n = 0;
    while (s > 0) {
        s = s - 15;
        n = n + 2;
    }
    cout << n << endl;
    return 0;
}
```

Решение.

Шаг 1. В цикле Пока, согласно программе, при S>0 происходит последовательное накопление нулевого значения n с шагом 2 и уменьшения первоначального значения S=260 на 15.

Шаг 2. Таким образом, цикл имеет 260/15=17,333… шага. Принимаем 18 шагов.

Шаг 3. Значение n последовательно увеличивается на 2: первый шаг n равен 2, второй – 4, третий – 6, и так далее. Таким образом, n увеличивается на последовательность четных чисел. 18 шагов определяет n как 18*2=36.

Ответ: 36.

Задача 9.

Автоматическая фотокамера производит растровые изображения размером 640*480 пикселей. При этом объём файла с изображением не может превышать 320 Кбайт, упаковка данных не производится. Какое максимальное количество цветов можно использовать в палитре?

Решение.

Шаг 1. Находим битовую составляющую изображения i, согласно формуле:

i=Q/S,
где Q – объем файла (в битах), S- площадь изображения (в пикселях^2).
Отсюда: i=320*8*1024/(640*480)=8,5 (бит)
Принимаем i=8 бит
Шаг 2. Цветовая палитра по формуле Хартли определяется как: $K=2^i=2^8=256$ (цветов)
Ответ: 256

Задача 10.

Все 4-буквенные слова, составленные из букв Д, Е, К, О, Р, записаны в алфавитном порядке и пронумерованы, начиная с 1. Ниже приведено начало списка.
1. ДДДД
2. ДДДЕ
3. ДДДК
4. ДДДО
5. ДДДР
6. ДДЕД

…
Под каким номером в списке идет первое слово, которое начинается с буквы
К?
Решение.
Шаг 1. Переводим четырехбуквенные последовательности в цифровые формы по таблице, обозначая:

Д	Е	К	О	Р
0	**1**	**2**	**3**	**4**

Таким образом, имеем 5- ричною систему счисления с цифрами (0, 1, 2, 3, 4).
Шаг 2. Определим место в списке последовательности, которая начинается с буквы К, как 2001(так первое место в списке рано 1, а не 0)
Шаг 3. Переводим данную последовательность в десятичную систему счисления: $2001_5 = 2.5^3+0+0+1*5^0 = 251_{10}$
Ответ: 251
Практические задачи по массивам на языке программирования.
Задача 1.
На языке Pascal создать программы ввода (заполнения) элементов массива данными:
1. С клавиатуры.

2. По указанной формуле
3. С помощью генератора случайных чисел.

Решение.

Фрагменты кодов программы:

1. С клавиатуры для одномерного массива

```
for i:=1 to 40 do read(a[i]);
```

для двумерного массива

```
for i:=1 to 10 do for j:=1 to 10 do read(a[i,j] );
```

2. По формуле для одномерного массива

```
for i:=1 to 40 do a[i]:=i*i+5;
```

для двумерного массива

```
for i:=1 to 10 do for j:=1 to 10 do a[i,j]:=i*j-4;
```

3. С помощью генератора случайного числа

Для одномерного массива

```
for i:=1 to 40 do a[i]:=random*20;
```

для двумерного массива

```
for i:=1 to 10 do 18 for j:=1 to 10 do a[i,j]:=random*10;
```

Задача 2.

Создать программу на языке Pascal по выводу элементов массива.

Решение.

Вывод элементов массива на экран монитора для одномерного массива:

```
for i:=1 to 40 do write(a[i]);
```

для двумерного массива (в виде матрицы)

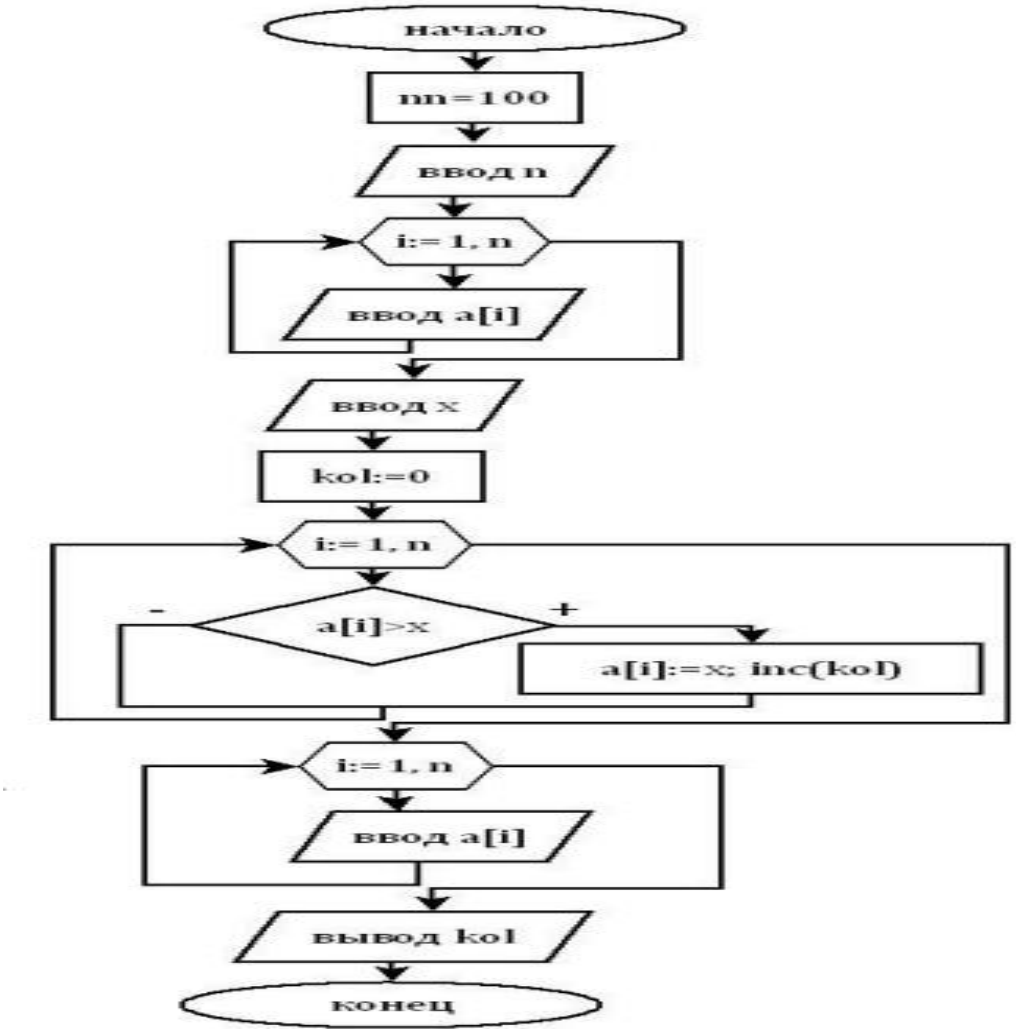
```
for i:=1 to 10 do begin for j:=1 to 10 do write(a[i,j] ); writeln; end;
```

Задача 3.

Дан массив действительных чисел A(n). Заменить все члены, большие заданного x, этим числом. Подсчитать количество замен.

Решение.

Блок-схема алгоритма представлена на рисунке:



Код программы:

```
Program zad3;
const nn=100; var a: array[1..nn] of real; i,n, kol:integer; x:real; begin writeln('Введите количество элементов массива'); readln(n); writeln('Введите элементы массива'); for i:=1 to n do read(a[i]); writeln('Введите значение x'); readln(x); kol:=0; for i:=1 to n do if a[i]>x then begin a[i]:=x; inc(kol); end; writeln('Массив после замены'); for i:=1 to n do write(a[i], ' '); writeln('Количество замен=', kol); end.
```

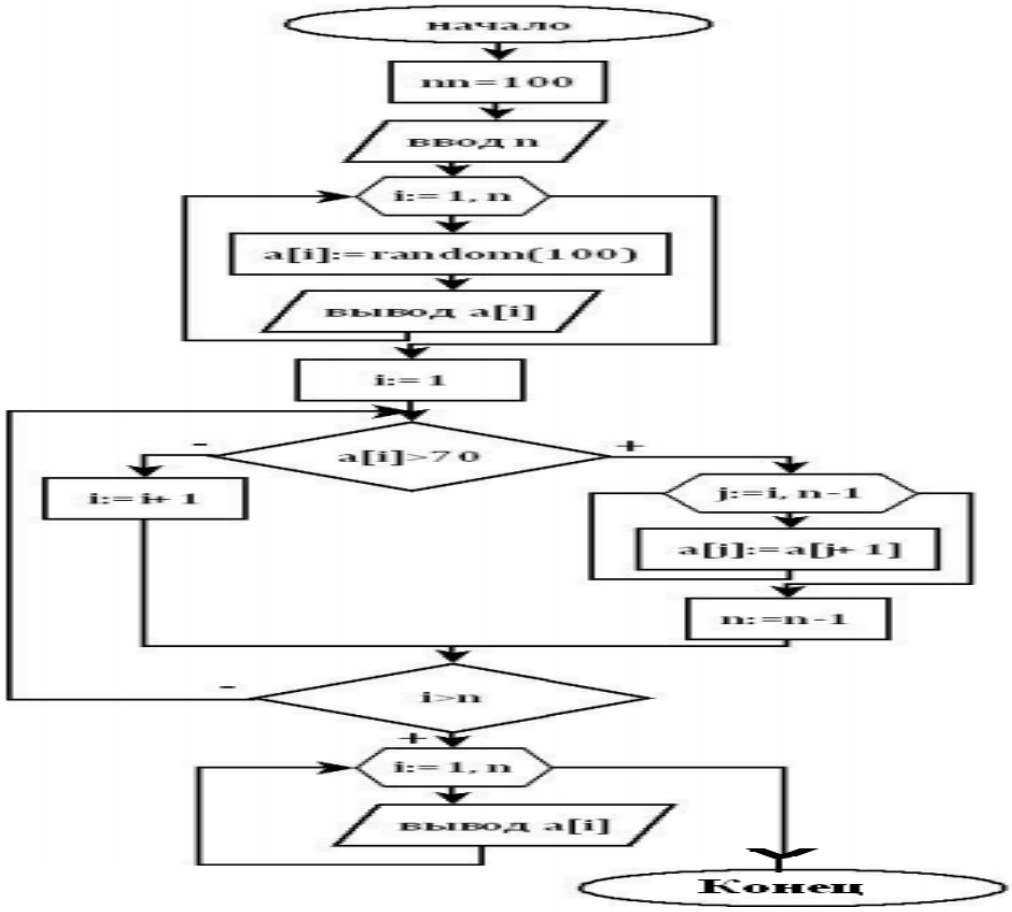
Контрольный счет: Введите количество элементов массива 8
Введите элементы массива 2 4 5 6 8 9 1 2
Введите значение x 4
Массив после замены 2 4 4 4 4 4 1 2
Количество замен=4

Задача 4.

Определить максимальный элемент массива B(50) и его порядковый номер. Предусмотреть вывод элементов массива на экран в строку.

Решение.

Поскольку количество элементов одномерного массива велико, воспользуемся генератором случайного числа для заполнения массива элементами. Блок-схема решения представлена на рисунке:



Код программы:

program zad4; const nn=50; var n,i,imax:integer; bmax:real; b:array[1..nn] of integer; begin write('Введите размер массива N<=50: '); read(n); randomize; for i:=1 to n do b[i]:=random(31); bmax:=b[1]; imax:=1; for i:=2 to n do if b[i]>bmax then begin bmax:=b[i]; imax:=i; end; writeln('**********Rezultat********* **'); write('Elementy massiva:'); for i:=1 to n do write(b[i]:4); writeln; writeln('bmax = ',bmax:4); write('imax = ',imax); end.

Контрольный счет

Введите размер массива N<=50: 8
**********Rezultat**********
Elementy massiva: 26 5 29 25 2 28 9 21
bmax = 29 imax = 3

Задача 5.

Имеется массив A(n), содержащий числа от 0 до 100 включительно. Требуется исключить из него все элементы, значения которых больше 70.

Решение.

Алгоритм: С помощью генератора случайного числа заполним массив А числами от 0 до 100 по формуле: A[i]:=random(100). Организуем просмотр всех эле- ментов массива и будем каждый из них сравнивать с числом 70: если a[i]>70, то сдвигаем элементы массива на одну позицию влево, значение n, обозначающее длину массива, уменьшаем на 1; если очередной элемент этому условию не удовлетворяет, то переходим к просмотру следующего элемента (i:=i+1) и не уменьшаем размер массива.

Код программы:

```
program zad5; const nn = 100; var a: array[1..nn] of real; i, j, n: integer; begin readln(n); for i := 1 to n do begin a[i] := random(100); write(a[i]:5:1); end; i := 1; writeln; repeat if a[i] > 70 then begin for j := i to n - 1 do a[j] := a[j + 1]; n := n - 1;
end else i := i + 1; until i > n; for i := 1 to n do write(a[i]:5:1); end.
```

Контрольный счет:

10 32.0 15.0 2.0 76.0 84.0 57.0 79.0 5.0 65.0 56.0 32.0 15.0 2.0 57.0 5.0 65.0 56.0

Задача 6.

Сформировать квадратную матрицу с помощью генератора случайных чисел. Вырезать из данной матрицы подматрицу размером [m,m] (m<n), в которой сумма диагональных элементов наименьшая. Вывести координаты этой подматрицы – номера строки и столбца левого нижнего ее угла, а также построчно в форматном виде элементы подматрицы.

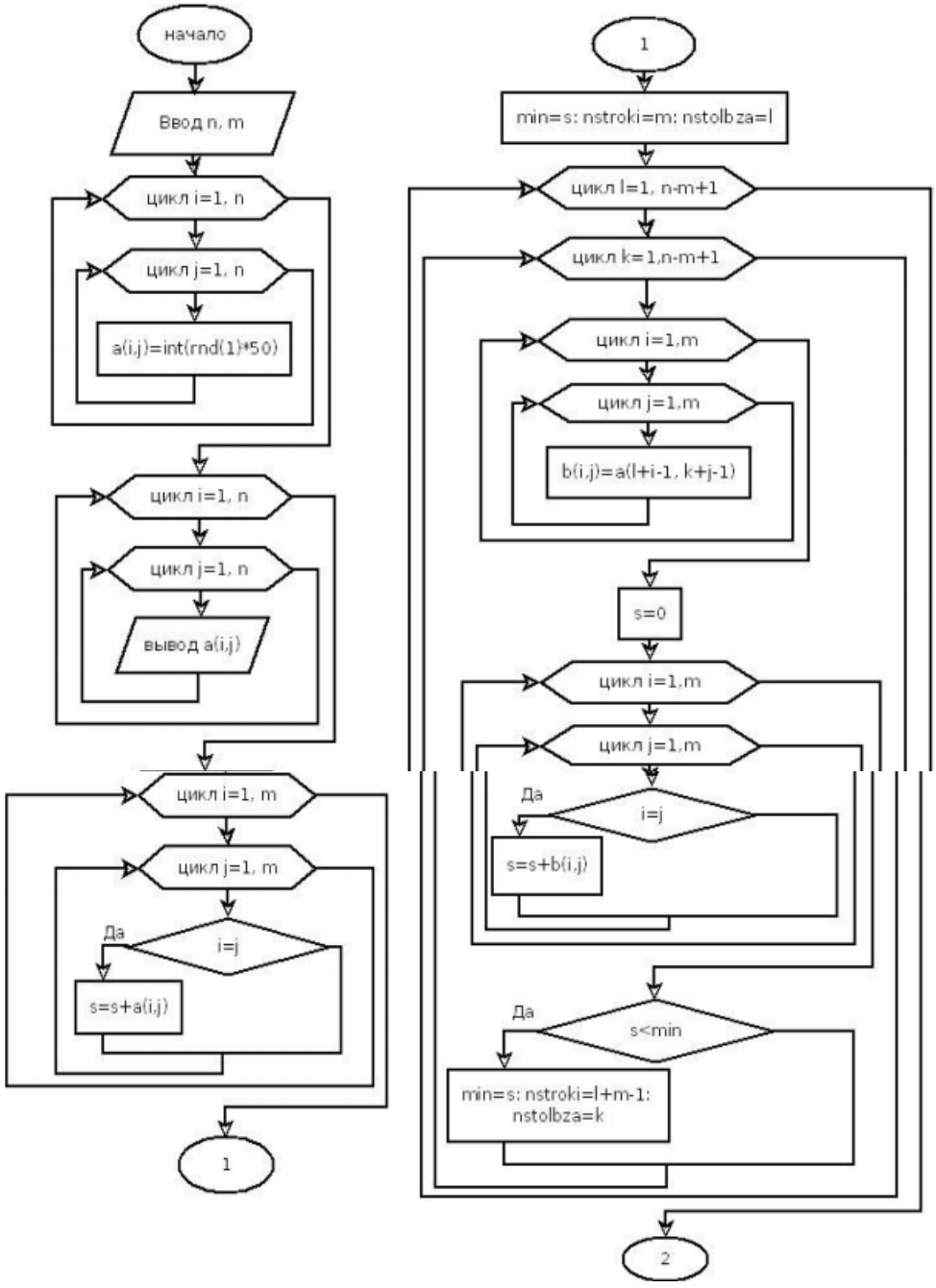
Решение.

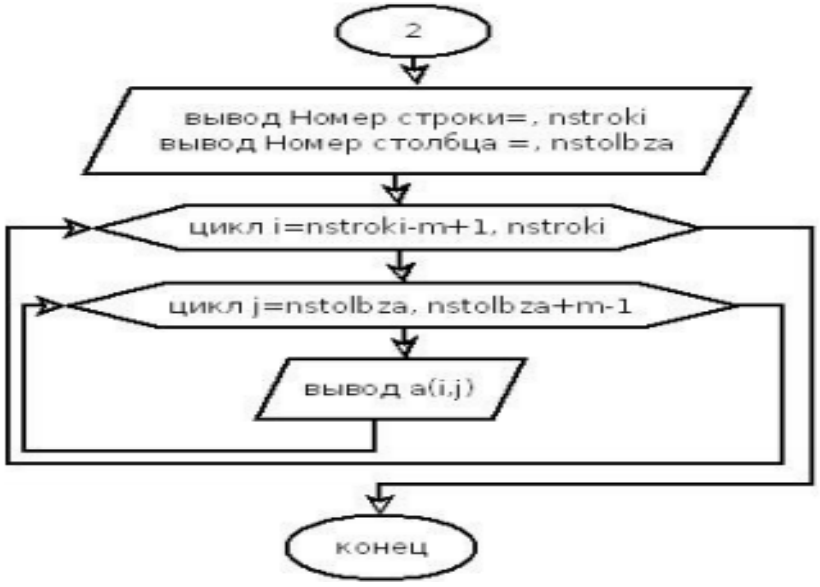
Код программы:

```
program zad6; const nn=50; mm=50; var i, j, l, k, n, m : integer; a, b:array[1..nn, 1..nn] of integer; s, min, nstroki, nstolbza :integer;
begin writeln('Введите размер матрицы'); readln(n); writeln('Введите размер подматрицы'); readln(m); randomize; for i:=1 to n do for j:=1 to n do a[i,j]:=trunc(random*50); writeln(' Сформированная матрица');
for i:=1 to n do begin for j:=1 to n do write(a[i, j]:6); writeln; end;
for i:=1 to m do for j:=1 to m do if i=j then s:=s+a[i, j]; min:=s; nstroki:=m; nstolbza:=1;
for l:=1 to n-m+1 do for k:=1 to n-m+1 do
```

```
begin for i:=1 to m do for j:=1 to m do b[i, j]:=a[l+i-1, k+j-1]; s:=0; for i:=1
to m do for j:=1 to m do if i=j then s:=s+b[i, j]; if s<min then begin min:=s;
nstroki:=l+m-1; nstolbza:=k; end; end;
writeln('Номер строки = ', nstroki, ' Номер столбца = ', nstolbza);
writeln(' Искомая подматрица');
for i:=nstroki-m+1 to nstroki do begin for j:=nstolbza to nstolbza+m-1 do
write(a[i, j]:6); writeln; end; end.
```

Блок-схема алгоритма:





Контрольный счет:

```
Введите размер матрицы
8
Введите размер подматрицы
3
    Сформированная матрица
     7    30    13    13    41    21    16    28
    37    12    21    37    18    24    23    20
    24     1    48    47     9    42    45    28
    17    47     0     9     7     7    28    28
    45    25    30    30    40    40    35     6
    31    16    25    37    15     1    43    22
    43    44    42     9    37    37     0    15
    34    35    46    44    21     6    21    30
Номер строки = 5  Номер столбца = 2
  Искомая подматрица
     1    48    47
    47     0     9
    25    30    30
```

Задача 7.

Вычислить сумму и число положительных элементов двумерного массива C(N , N) , находящихся над главной диагональю матрицы, где N принадлежит промежутку от 12 до 10017: если будет введено другое значение N, то запросить размер массива заново.

Решение.

Код программы:

program zad7; label vozvrat;

var i, j, k, n, summa : integer; имена i, j: of integer; c:array[1..100,1..100] of integer; begin

vozvrat: write('Введите размер матрицы: '); read(n); if (n100) then

```
begin writeln('Введите число 12<=x<=100'); {Вывод строки «Введите число от 12 до 100» 17 Замечание: на этапе разработки и тестирования программы рекомендуем взять размер массива 0} then {Если сформированное значение элемента массива c[i, j]>0, тогда} if j>i then {Если для его индексов выполняется условие j>i (элемент стоит над диагональю), то} begin inc(k); summa:=summa+c[i,j]; end; end;
for i:=1 to n do
begin
for j:=1 to n do write(c[i,j]:5); end;
writeln('********** **********'); writeln('Количество= ',k);
write('Сумма= ',summa); end.
```

Контрольный счет:

```
Введите размер матрицы: 12
   14   29    1   15    2  -43   -7   48   -9    3  -28    2
   -8   20  -16  -14   47    6  -45   10  -45    5    7  -29
   -7   23    6   -1  -15   31   45   16  -42   30   -6  -48
   48  -34  -25  -49    7  -11   22  -10  -13   -5  -31   36
  -49  -29  -28  -34  -33  -13    7  -42  -28  -21  -28   29
    1   44   12   10    9    7   -7   12  -24  -18  -11   31
  -20  -46  -48   17   34  -33   35   48    3   21   -2  -36
    9   28   -2   22  -27  -15   30   19   44   -3    0   14
   28  -38   31   30  -49    8   10   20   32   -8   37  -37
  -16   41  -25  -45   -6  -26  -25   31   -4   -3  -49   32
  -41   22  -11   41    2   21   37   34    4  -12   40   22
  -14  -46   47   46  -40   28  -18  -34   41  -16  -13  -18
**********
Количество= 31
Сумма= 662
```

Задача 8.

Составить программу для решения системы линейных алгебраических уравнений {2 x+y+z=7, x+2y+z=8, x+ y+2 z=9 } методом Крамера:

а) в предположении, что система совместна и имеет единственное решение; б) в предположении, что система может быть как совместной, так и несовместной.

Решение.

Математическая модель:

Для системы трех уравнений с тремя неизвестными требуется:

1. Вычислить определитель системы Δ по формуле:
   $\Delta = a_{11} * a_{22} * a_{33} + a_{12} * a_{23} * a_{31} + a_{13} * a_{21} * a_{32} - a_{31} * a_{22} * a_{13} - a_{32} * a_{23} * a_{11} - a_{33} * a_{21} * a_{12}$
2. Вычислить определители, полученные поочередной заменой каждого из трех столбцов на столбец свободных членов:

$$\Delta_x = \begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}, \Delta_y = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}, \Delta_z = \begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}.$$

3. Проанализировать полученные значения:
   a) определитель системы $\Delta \neq 0$: система совместна, единственное решение системы ищем по формулам: $x = \Delta x / \Delta$, $y = \Delta y / \Delta$, $z = \Delta z / \Delta$; b) определитель системы $\Delta = 0$, при этом один из определителей $\Delta x, \Delta y, \Delta z \neq 0$: система несовместна, решений не имеет;
   c) все полученные определители равны 0: система совместна, имеет бесчисленное множество решений.

Приведем программу для решения задачи при условии a). Коэффициенты системы уравнений будем хранить в константах — одномерном и двумерном массивах.

Код программы:

```
program zad8; const a: array[1..3, 1..3] of integer = ((2, 1, 1), (1, 2, 1), (1, 1, 2)); b: array[1..3] of integer = (7, 8, 9); var i, j: integer; delta, delta1, delta2, delta3, x1, x2, x3: real; begin delta := a[1, 1] * a[2, 2] * a[3, 3] + a[1, 2] * a[2, 3] * a[3, 1] + a[1, 3] * a[2, 1] * a[3, 2] - a[3, 1] * a[2, 2] * a[1, 3] - a[3, 2] * a[2, 3] * a[1, 1] - a[3, 3] * a[2, 1] * a[1, 2]; delta1 := b[1] * a[2, 2] * a[3, 3] + a[1, 2] * a[2, 3] * b[3] + a[1, 3] * b[2] * a[3, 2] - b[3] * a[2, 2] * a[1, 3] - a[3, 2] * a[2, 3] * b[1] - a[3, 3] * b[2] * a[1, 2]; delta2 := a[1, 1] * b[2] * a[3, 3] + b[1] * a[2, 3] * a[3, 1] + a[1, 3] * a[2, 1] * b[3] - a[3, 1] * b[2] * a[1, 3] - b[3] * a[2, 3] * a[1, 1] - a[3, 3] * a[2, 1] * b[1]; delta3 := a[1, 1] * a[2, 2] * b[3] + a[1, 2] * b[2] * a[3, 1] + b[1] * a[2, 1] * a[3, 2] - a[3, 1] * a[2, 2] * b[1] - a[3, 2] * b[2] * a[1, 1] - b[3] * a[2, 1] * a[1, 2]; x1 := delta1 / delta; x2 := delta2 / delta; x3 := delta3 / delta; writeln('Найденное решение:'); writeln('x1=', x1, ' x2=', x2, ' x3=', x3); end.
```

Найденное решение: x1=1 x2=2 x3=3

Задача 9.

Составить программу для решения системы линейных алгебраических уравнений с тремя переменными с любыми целочисленными коэффициентами. Коэффициенты системы уравнений будем вводить с клавиатуры.

Код программы:

```
program zad9; var a: array[1..3, 1..3] of integer; b: array[1..3] of integer; i, j: integer; delta, delta1, delta2, delta3, x1, x2, x3: real; begin writeln('Введите коэффициенты системы при неизвестных x1, x2, x3'); for i := 1 to 3 do for j := 1 to 3 do read(a[i, j]); writeln('Введите столбец
```

свободных членов'); for i := 1 to 3 do read(b[i]); delta := a[1, 1] * a[2, 2] * a[3, 3] + a[1, 2] * a[2, 3] * a[3, 1] + a[1, 3] * a[2, 1] * a[3, 2] - a[3, 1] * a[2, 2] * a[1, 3] - a[3, 2] * a[2, 3] * a[1, 1] - a[3, 3] * a[2, 1] * a[1, 2]; delta1 := b[1] * a[2, 2] * a[3, 3] + a[1, 2] * a[2, 3] * b[3] + a[1, 3] * b[2] * a[3, 2] - b[3] * a[2, 2] * a[1, 3] - a[3, 2] * a[2, 3] * b[1] - a[3, 3] * b[2] * a[1, 2]; delta2 := a[1, 1] * b[2] * a[3, 3] + b[1] * a[2, 3] * a[3, 1] + a[1, 3] * a[2, 1] * b[3] - a[3, 1] * b[2] * a[1, 3] - b[3] * a[2, 3] * a[1, 1] - a[3, 3] * a[2, 1] * b[1]; delta3 := a[1, 1] * a[2, 2] * b[3] + a[1, 2] * b[2] * a[3, 1] + b[1] * a[2, 1] * a[3, 2] - a[3, 1] * a[2, 2] * b[1] - a[3, 2] * b[2] * a[1, 1] - b[3] * a[2, 1] * a[1, 2]; if delta <> 0 then begin x1 := delta1 / delta; x2 := delta2 / delta; x3 := delta3 / delta; writeln('Найденное решение:'); writeln('x1=', x1, ' x2=', x2, ' x3=', x3); end else if (delta1 = 0) and (delta2 = 0) and (delta3 = 0) then writeln('Система имеет бесконечно много решений') else writeln('Система решений не имеет'); end.

Тестовый набор 1 (единственное решение)

Введите коэффициенты системы при неизвестных x1, x2, x3 3 4 2 5 -6 -4 -4 5

3

Введите столбец свободных членов 5 -3 1

Найденное решение: x1=1 x2=-2 x3=5

Тестовый набор 2 (решений нет)

Введите коэффициенты системы при неизвестных x1, x2, x3 1 1 1 1 -1 1 1 0 1 Введите столбец свободных членов 5 1 2

Система решений не имеет

Тестовый набор 3 (бесконечно много решений)

Введите коэффициенты системы при неизвестных x1, x2, x3 1 1 1 3 -1 2 1 -3 0 Введите столбец свободных членов 0 0 0

Система имеет бесконечно много решений

Задача 10.

Выполнить пояснения к программе и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работоспособность.

Код программы:

program zad11; const nn=100; var a: array[1..nn] of real; i,n:integer; s:real;
begin
writeln('Введите количество элементов массива'); readln(n);
writeln('Введите элементы массива'); for i:=1 to n do read(a[i]); s:=0; for i:=1 to n do if a[i]>0 then s:=s+a[i]; writeln('Сумма =', s);
end.

Решение.

Формулировка задачи: найти сумму положительных элементов одномерного массива A размера N.

Контрольный счет:

Введите количество элементов массива 5

Введите элементы массива -8 7 5 0 -1

Сумма =12

Задача 11.

Проанализировать программу, определить, для решения какой задачи написана программа. Составить таблицу трассировки.

Код программы:

```
program massiv;
const n=30; var a:array[1..n] of real; i, j:integer; t:real;
begin
for i:=1 to n do a[i]:=random(50); writeln('Исходный массив');
for i:=1 to n do write(a[i]:8:1); writeln; for i:=1 to n-1 do for j:=i+1 to n do
begin if a[i]>a[j] then begin t:=a[i]; a[i]:=a[j]; a[j]:=t; end;
end;
writeln('Получен массив'); for i:=1 to n do write(a[i]:8:1); end.
```

Решение.

Пусть массив A состоит из следующих чисел: 45, 17, 5, 14, 39, 6, 37, 6, 13. Составим таблицу трассировки для следующего основного блока программы: for i:=1 to n-1 do for j:=i+1 to n do begin if a[i]>a[j] then begin t:=a[i]; a[i]:=a[j]; a[j]:=t; end; end;

end;

	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
i=0, j=0	**45**	**17**	5	14	39	6	37	6	13

i=1, j=2, a[1]>a[2] - верно	**17**	**45**	5	14	39	6	37	6	13
j=3, a[1]>a[3] - неверно	**5**	45	**17**	14	39	6	37	6	13
j=4, a[1]>a[4] - неверно	**5**	45	17	**14**	39	6	37	6	13
j=5, a[1]>a[5] - неверно	**5**	45	17	14	**39**	6	37	6	13
j=6, a[1]>a[6] - неверно	**5**	45	17	14	39	**6**	37	6	13
j=7, a[1]>a[7] - неверно	**5**	45	17	14	39	6	**37**	6	13
j=8, a[1]>a[8] - неверно	**5**	45	17	14	39	6	37	**6**	13
j=9, a[1]>a[9] - неверно	**5**	45	17	14	39	6	37	6	**13**
i=2, j=3, a[2]>a[3] - верно	5	**17**	45	14	39	6	37	6	13
j=4, a[2]>a[4] - верно	5	**14**	**45**	**17**	39	6	37	6	13
j=5, a[2]>a[5] - неверно	5	**14**	45	17	**39**	6	37	6	13
j=6, a[2]>a[6] - верно	5	**6**	45	17	39	**14**	37	6	13
j=7, a[2]>a[7] - неверно	5	**6**	45	17	39	14	**37**	6	13
j=8, a[2]>a[8] - неверно	5	**6**	45	17	39	14	37	**6**	13
j=9, a[2]>a[9] - неверно	5	**6**	45	17	39	14	37	6	**13**
i=3, j=4, a[3]>a[4] - верно	5	6	**17**	**45**	39	14	37	6	13
j=5, a[3]>a[5] - неверно	5	6	**17**	45	**39**	14	37	6	13
j=6, a[3]>a[6] - верно	5	6	**14**	45	39	**17**	37	6	13
j=7, a[3]>a[7] - неверно	5	6	**14**	45	39	17	**37**	6	13
j=8, a[3]>a[8] - верно	5	6	**6**	45	39	17	37	**14**	13
j=9, a[3]>a[9] - неверно	5	6	**6**	45	39	17	37	14	**13**
i=4, j=5, a[4]>a[5] - верно	5	6	6	**39**	**45**	17	37	14	13
j=6, a[4]>a[6] - верно	5	6	6	**17**	45	**39**	37	14	13
j=7, a[4]>a[7] - неверно	5	6	6	**17**	45	39	**37**	14	13
j=8, a[4]>a[8] - неверно	5	6	6	**17**	45	39	37	**17**	13
j=9, a[4]>a[9] - верно	5	6	6	**13**	45	39	37	17	**14**
i=5, j=6, a[5]>a[6] - верно	5	6	6	13	**39**	**45**	37	17	14
j=7, a[5]>a[7] - верно	5	6	6	13	**37**	45	**39**	17	14
j=8, a[5]>a[8] - верно	5	6	6	13	**17**	45	39	**37**	14
j=9, a[5]>a[9] - верно	5	6	6	13	**14**	45	39	37	**17**

	A[1]	**A[2]**	**A[3]**	**A[4]**	**A[5]**	**A[6]**	**A[7]**	**A[8]**	**A[9]**
i=6, j=7, a[6]>a[7] - верно	5	6	6	13	14	**39**	**45**	37	17
j=8, a[6]>a[8] - верно	5	6	6	13	14	**37**	45	**39**	17
j=9, a[6]>a[9] - верно	5	6	6	13	14	**17**	45	39	**37**
i=7, j=8, a[7]>a[8] - верно	5	6	6	13	14	17	**39**	**45**	37
j=9, a[7]>a[9] - верно	5	6	6	13	14	17	**37**	45	**39**
i=8, j=9, a[8]>a[9] - неверно	5	6	6	13	14	17	37	**39**	**45**

Из таблицы видно, что элементы в массиве расположились по возрастанию, т.е. программа предназначена для сортировки элементов одномерного массива (методом обмена) по возрастанию.

Практическое занятие 13. Практические задачи по чтению программ на языке программирования и исправление допущенных ошибок

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 13, которые являются задачами демонстрационного варианта ЕГЭ за 2018 год.

Домашние задания к занятию 14.

Задача 11.

Ниже на пяти языках программирования записан рекурсивный алгоритм F. Запишите подряд без пробелов и разделителей все числа, которые будут напечатаны на экране при выполнении вызова F(9). Числа должны быть записаны в том же порядке, в котором они выводятся на экран.

Бейсик

```
SUB F(n)
  IF n > 0 THEN
     PRINT n
     F(n - 3)
     F(n \ 3)
   END IF
END SUB
```

Python

```
def F(n):
    if n > 0:
        print(n)
        F(n - 3)
        F(n // 3)
```

Алгоритмический язык

```
алг F(цел n)
нач
   если n > 0 то
     вывод n
     F(n - 3)
     F(div(n, 3))
   все
кон
```

Паскаль

```
procedure F(n: integer);
begin
  if n > 0 then
  begin
    write(n);
    F(n - 3);
    F(n div 3)
  end
end;
```

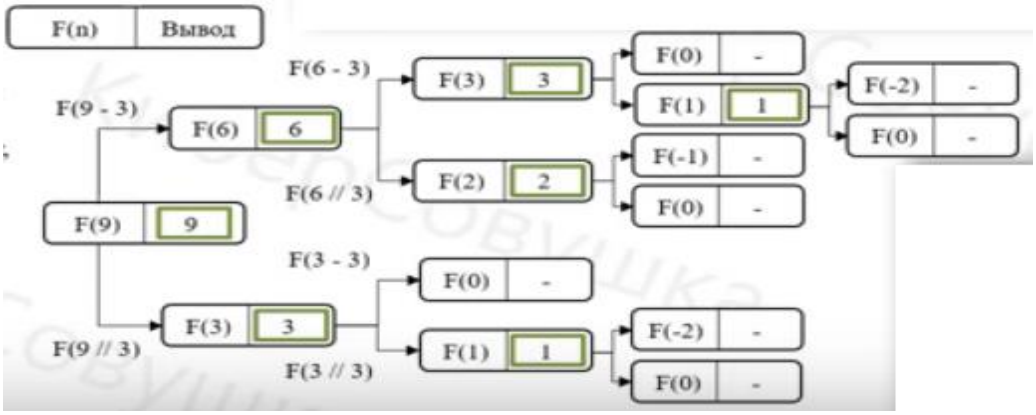
C++

```
void F(int n){
    if (n > 0){
        std::cout <<n;
        F(n - 3);
        F(n / 3);
    }
}
```

Решение.

Алгоритм: вывести последовательно на экран значение n при выполнении вызова рекурсивной функции F(n), где первоначальное значение n равно 9, и при выполнении цикла n>0 осуществляется вызов функции F при (n-3)и функции F (целое от деления n на 3). При этом команды внутри функции выполняются последовательно.

Шаг 1. Построить дерево вызова функции F(n), где пробел обозначает отсутствие вывода на экран (см. рисунок).



Шаг 2. Согласно рисунку записать значения вывода функции на экран (по часовой стрелке по верхней ветви к нижней) в символьной и цифровой форме:

F(9) - F(6) - F(3) - F(1) -F(2) - F(3) - F(1),

9631231

Ответ: 9631231

Задача 12.

В терминологии сетей TCP/IP маской сети называется двоичное число, определяющее, какая часть IP-адреса узла сети относится к адресу сети, а какая – к адресу самого узла в этой сети. Обычно маска записывается по тем же правилам, что и IP-адрес, – в виде четырёх байтов, причём каждый байт записывается в виде десятичного числа. При этом в маске сначала (в старших разрядах) стоят единицы, а затем с некоторого разряда – нули.

Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске. Например, если IP-адрес узла равен 231.32.255.131, а маска равна 255.255.240.0, то адрес сети равен 231.32.240.0.

Для узла с IP-адресом 57.179.208.27 адрес сети равен 57.179.192.0. Каково наибольшее возможное количество единиц в разрядах маски?

Решение.

Алгоритм: определение разрядов маски по с IP-адресам узла и сети.

Шаг 1. Так как три байта с IP-адресов совпадают, то адрес маски представляет собой последовательность: 255.255.???.0. переведем третьи байты адресов в двоичную систему:

Сеть: $208_{10}=1101000_2$,

Узел: $192_{10}=11000000_2$.

Шаг 2. Определим третий байт маски по таблице, учитывая, что адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске:

Байт сети 208	1	1	0	1	0	0	0	0
Байт маски	? 1	? 1	? 1	? 0	? 0	? 0	? 0	? 0
Байт IP-адреса сети 192	1	1	0	0	0	0	0	0

Таким образом, адрес маски представляет собой: 255.255.224.00.

Шаг 3. Подсчитаем количество единиц в разрядах маски: 8+8+3=19

Ответ: 19

Задача 13.

При регистрации в компьютерной системе каждому пользователю выдается пароль, состоящий из 10 символов. В качестве символов используют прописные буквы латинского алфавита, т.е. 26 различных символов. В базе данных для хранения каждого пароля отведено одинаковое и минимально возможное целое число байт. При этом используют посимвольное кодирование паролей, все символы кодируют одинаковым и минимально возможным количеством бит.

Определите объем памяти (в байтах), необходимый для хранения данных о 50 пользователях. В ответе запишите только целое число – количество байт.

Решение.

Алгоритм: двоичное кодирование информации.

Шаг 1. Определяем по формуле Хартли мощность двоичного кода:

2^i>=26=32.

Отсюда: i= 5 бит.

Шаг 2. Определяем количество информации для одного пользователя в битах:

N=5*10=50 бит

Округляем до байтов: 50/8=6,25 = 7байт

Шаг 3. Определяем объем информации для всех пользователей в байтах:

Q=50*7=350 байт.

Ответ: 350 байт.

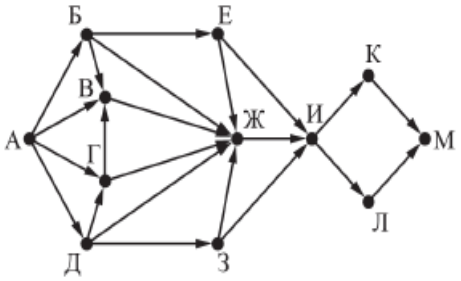
Задача 14.

Исполнитель Чертежник перемещается на координатной плоскости, оставляя след в виде линии. Чертежник может выполнять команду сместиться на (*a*, *b*), где *a*, *b* – целые числа. Эта команда перемещает Чертежника из точки с координатами (*x*,*y*) в точку с координатами (*x* + *a*, *y* + *b*). Например, если Чертёжник находится в точке с

координатами (4, 2), то команда сместиться на (2, −3) переместит Чертежника в точку (6, −1).
Цикл
ПОВТОРИ *число* РАЗ
последовательность команд
КОНЕЦ ПОВТОРИ
означает, что последовательность команд будет выполнена указанное число раз (число должно быть натуральным). Чертёжнику был дан для исполнения следующий алгоритм (число повторений и величины смещения в первой из повторяемых команд неизвестны):
НАЧАЛО
сместиться на (4, 6)
ПОВТОРИ …РАЗ
сместиться на (…, …)
сместиться на (4, -6)
КОНЕЦ ПОВТОРИ
сместиться на (-28, -22)
КОНЕЦ
В результате выполнения этого алгоритма Чертежник возвращается в исходную точку. Какое наибольшее число повторений могло быть указано в конструкции «ПОВТОРИ … РАЗ»?
Решение.
Алгоритм: обратная задача на исполнители.
Шаг 1. Обозначим количество повторений в конструкции как «ПОВТОРИ … РАЗ» как n. Тогда в результате выполнения программы Чертежник должен очутиться в точке с начальными координатами (x=0,y=0): 4+ n*(a+4)-28=0 и 6+ n*(b -6)-22=0.
Шаг 2. Преобразуем уравнения относительного свободного члена: n*(a+4)=24 и n*(b -6)=16.
Таким образом, чтобы определить максимальное число повторений в цикле, необходимо найти n, как наибольший общий делитель свободных членов уравнений координат (x,y):
НОД(24, 16)=8
Ответ: 8

Задача 15.

На рисунке представлена схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З, И, К, Л, М.
По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Сколько существует различных путей из города А в город М, проходящих через город Ж?



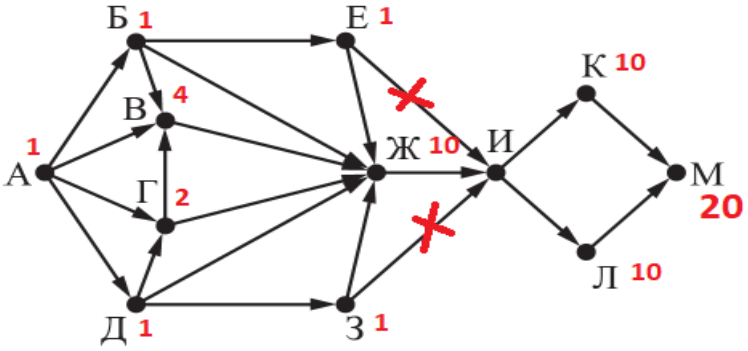
Решение.

Алгоритм: информационная модель транспортной задачи – графическое решение.

Шаг 1. Согласно рисунку, прежде всего, преобразуем граф, отбрасывая пути, на проходящие через город Ж: это участки Е-И и З-К.

Шаг 2. Производим последовательно подсчет путей с начальной точки А (имеющей наибольшее число ветвей) до точки Ж.

Шаг 3. Производим окончательный подсчет путей от точки Ж до точки М. Согласно рисунку, от А до М имеется 20 путей.



Ответ: 20

Практические задачи по чтению программ на языке программирования и исправление допущенных ошибок.

Типы ошибок при программировании.

Ошибки программирования (баги - на жаргоне) имеют место в работе любого разработчика программного обеспечения.

В таблице 4 представлены основные типы ошибок.

Ошибки программирования являются частью обучения, и их никогда нельзя полностью избежать. Большинство ошибок можно исправить в процессе практической разработки программного обеспечения со строгими процедурами отладки. Ошибки программирования можно ликвидировать с помощью предварительного планирования во время стадии кодирования, в процессе интенсивного тестирования в фазе отладке цикла разработки программного обеспечения.

Таблица 4 - Основные типы ошибок программирования.

Тип ошибок программирования	Описание
1	2
Логическая ошибка	Самая серьезная ошибка алгоритмического уровня. Если написанная программа на любом языке компилируется и работает правильно, но выдает неправильный вывод, то данный недостаток заключается в самой логике программирования в базовом алгоритме. Для устранения ошибки необходимы фундаментальные изменения алгоритма.
Синтаксическая ошибка	Каждый компьютерный язык, имеет специфический синтаксис, в котором будет написан код. Если программист не придерживаться "грамматики" спецификациями компьютерного языка, то возникнет ошибка синтаксиса. Обычно данные ошибки легко устраняются на этапе компиляции.
Ошибка компиляции	В процессе компиляции программа, написанная на языке высокого уровня, преобразуется в машиночитаемую форму. Иногда, если синтаксис исходного кода может быть безупречным, ошибка компиляции все же может произойти, что связано с проблемами в самом компиляторе. Данные ошибки исправляются на стадии разработки.
Ошибки среды выполнения (RunTime)	Программный код успешно скомпилирован, и исполняемый файл был создан. Вы можете вздохнуть с облегчением и запустить программу, чтобы проверить ее работу. Данные ошибки при выполнении программы и запуске исполняемого файла могут возникнуть в результате аварии или нехватки ресурсов носителя в реальных условиях развертывания программы. Данные ошибки ликвидируются на стадии кодирования.
Арифметическая ошибка	Многие программы используют числовые переменные, и алгоритм может включать несколько математических вычислений. Арифметические ошибки возникают, если компьютер не может справиться с проблемами, такими как "Деление на ноль", или ведущие к бесконечному результату. Данная ошибка является логической ошибкой и может быть исправлена только путем изменения алгоритма.

1	2
Ошибки ресурса	Ошибка ресурса возникает, если значение переменной переполняет максимально допустимое значение. Переполнение буфера, использование неинициализированной переменной, нарушение прав доступа и переполнение стека являются примерами некоторых распространенных ошибок данного типа.
Ошибка взаимодействия	Возникают в связи с несоответствием программного обеспечения с аппаратным интерфейсом или интерфейсом прикладного программирования. В случае веб-приложений ошибка интерфейса может быть результатом неправильного использования веб-протокола.

Задания ЕГЭ по информатике представлены в его второй части и проверяют у экзаменующего его умения прочесть фрагмент программы на языке программирования и исправить допущенные ошибки как логические, так и синтаксические. При этом необходимо из описания алгоритма и приведенного листинга программы определить наиболее эффективные методы исправления ошибок без переписывания самой программы. Следует учесть, что данные задачи относятся к повышенному уровню сложности задания.

Практическая задача 1.

Дано целое положительное число N, не превосходящее 1000. Необходимо определить, является ли это число степенью числа 3. То есть требуется определить, существует ли такое целое число K, что $3^K = N$, и вывести это число либо сообщение, что такого числа не существует.

Для решения этой задачи ученик написал программу, но, к сожалению, его программа оказалась неверной. Ниже эта написанная им программа для Вашего удобства приведена на пяти языках программирования.

Бейсик

```
DIM N, K AS INTEGER
INPUT N
K = 0
WHILE K MOD 3 = 0
   K = K + 1
   N = N \ 3
WEND
IF N > 0 THEN
   PRINT K
ELSE
   PRINT "Не существует"
END IF
END
```

Python

```
n = int(input())
k = 0
while k%3 == 0:
     k = k + 1
     n = n // 3
if n > 0:
     print(k)
else:
     print("Не существует")
```

Алгоритмический язык

```
алг
нач
   цел n, k
   ввод n
   k := 0
   нц пока mod(k, 3)=0
      k := k + 1
      n := div(n,3)
   кц
   если n > 0
      то вывод k
      иначе вывод "Не существует"
   все
кон
```

Паскаль

```
var n, k: integer;
begin
   read(n);
   k := 0;
   while k mod 3 = 0 do begin
      k := k + 1;
      n := n div 3;
   end;
   if n > 0 then
      writeln(k)
   else
      writeln('Не существует')
end.
```

Си

```
#include <stdio.h>
int main(){
    int n, k;
    scanf("%d",&n);
    k = 0;
    while (k%3 == 0) {
        k = k + 1;
        n = n / 3;
    }
    if (n > 0)
        printf("%d", k);
    else
        printf("Не существует");
    return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 9.
2. Приведите пример числа, при вводе которого приведенная программа напечатает то, что требуется.
3. Найдите в программе все ошибки (их может быть одна или несколько).

   Для каждой ошибки выпишите строку, в которой она допущена, и приведите эту же строку в исправленном виде. Достаточно указать ошибки и способ их исправления для одного языка программирования.

Решение.

1. При вводе числа 9 программа выведет число 1.

Представим следующую вычислительную таблицу:

Исходные данные	**Шаг 1**
n:=9 k:=0 цикл при k mod 3=0?	**n:=n div 3=3 k:=k+1=1**
Выход из цикла и печать	**k mod 3=1 - выход из цикла, при n>0 печать k:=1**

Таким образом, программа работает неверно, так как при вводе 9, как числа с основанием 3, k должно быть равно 2.

2. Примеры чисел, при вводе которых программа выводит корректный ответ: 2, 3. Других чисел нет.

Согласно таблице после выполнения программы при любом введённом n значение k будет равно 1 (тело цикла выполнится ровно 1 раз).

В результате программа напечатает либо 1 (если $n \geq 3$), либо «Не существует» (в противном случае). Таким образом, программа

выводит корректный ответ, только если введено 2 («Не существует») или 3 (k=1). 3. Программа содержит две ошибки:

1) неверное условие цикла;
2) неверное условие при печати результата.

Пример исправления для языка Паскаль:
Первая ошибка:
while k mod 3 = 0 do begin
Исправленная строка:
while n mod 3 = 0 do begin
Вторая ошибка:
if n>0 then
Исправленная строка:
if n=1 then
После исправления первой ошибки в результате выполнения цикла значение переменной n будет равно $n_0/(3^k)$, где n_0 – введённое пользователем значение;
k – максимальный показатель степени, при котором 3^k является делителем числа n_0. Число n_0 является степенью числа 3, если $n_0 = 3^k$, т.е. $n_0/(3^k) = 1$.

Практическая задача 2.

На обработку поступает положительное целое число, не превышающее 10^9. Нужно написать программу, которая выводит на экран количество разрядов числа, имеющих четные значения. Программист написал программу неправильно. Ниже эта программа приведена на языке Pascal:

```
var N, digit, sum: longint;
begin
  readln(N);    sum := 1;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit mod 2 = 0 then
      sum := sum + digit;
    N := N div 10;   end;
  writeln(sum)
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 128.
2. Приведите пример такого трёхзначного числа, при вводе которого программа выдаёт верный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Решение.

Разбор алгоритма и программу: Вначале вводится число N, затем проверяется каждый разряд числа, и если он кратен 2, то значение переменной sum должно увеличиваться на 1.

Выполняем задания последовательно.

1. При вводе числа 128:

Переменная sum первоначально равна 1. Цикл Пока:

1 повтор цикла:

digit := n mod 10 = 128 mod 10 = 8

условие digit mod 2 = 0 выполняется, к переменной sum прибавляется значение digit, то есть sum := 1 + 8 = 9

N := N div 10 = 128 div 10 = 12

2 повтор цикла:

digit := n mod 10 = 12 mod 10 = 2

условие digit mod 2 = 0 выполняется, к переменной sum прибавляется значение digit, то есть sum := 9+2 = 11

N := N div 10 = 12 div 10 = 1

3 повтор цикла:

digit := n mod 10 = 1 mod 10 = 1

условие digit mod 2 = 0 не выполняется

N := N div 10 = 1 div 10 = 0

Цикл завершен, на экран вывелось значение sum, то есть 11.

Ответ: 11

2. Приведенная программа и алгоритм:

Переменная sum вначале равна 1, к sum прибавляются все четные цифры числа, значит, мы должны найти такое число, в котором количество четных разрядов равно сумме четных разрядов +1. При этом цифра 0 четна, и мы можем ее использовать в числе.

Нам нужно трехзначное число, поэтому максимальный результат, который может быть выведен при его вводе — 3, то есть все разряды должны быть четные. Для этого подходит число 200, так как 1+2+0+0 = 3, и число 200 содержит ровно три четных разряда.

Ответ: 200

3. Ошибки в этой программе:

Переменная суммы sum первоначально должна быть равна 0, а не 1.
То есть строку sum := 1; нужно заменить на sum := 0;
Каждый раз, когда выполняется условие, переменная должна увеличиваться на 1, а не на разряд числа. То есть строку sum := sum + digit;
нужно заменить на sum := sum + 1;

Ответ:

sum := 1 заменяется на **sum := 0;**
sum := sum + digit; заменяется на **sum := sum + 1;**

Практическая задача 3.

На обработку поступает положительное целое число, не превышающее 10^9. Нужно написать программу, которая выводит на экран сумму цифр этого числа, меньших 7. Если в числе нет цифр, меньших 7, требуется на экран вывести 0. Программист написал программу неправильно (на языке Pascal):

```
var N, digit, sum: longint;
begin
  readln(N); sum := 0;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit < 7 then
      sum := sum + 1;  N := N div 10;
  end;
  writeln(digit)
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 456.
2. Приведите пример такого трёхзначного числа, при вводе которого программа выдаёт верный ответ.
3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Решение.

При правильном алгоритме, приведенном в условии задачи, программа должна выводить сумму цифр числа, меньших 7.
Рассмотрим последовательно решение в приведенной программе.

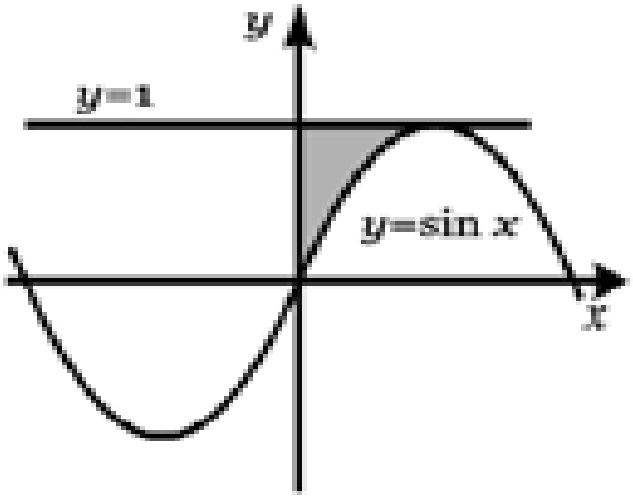
1. Что выведет программа при вводе числа 456.
Заявлено, что N = 456, sum = 0
Начало цикла Пока:
1-я итерация:
digit := 456 mod 10 = 6, 6 < 7 — условие
выполняется, следовательно
sum := sum + 1 = 0 + 1 = 1, N := N div 10 = 456 div 10 = 45
2-я итерация:
digit := 45 mod 10 = 5, 5 < 7 — условие выполняется, следовательно
sum := sum + 1 = 1 + 1 = 2, N := N div 10 = 45 div 10 = 4
3-я итерация:
digit := 4 mod 10 = 4, 4 < 7 — условие выполняется, следовательно sum := sum + 1 = 2 + 1 = 3, N := N div 10 = 4 div 10 = 0.
цикл завершен при N=0, digit в последней итерации равна 4, то есть выводится 4. Но так как сумма цифр 456 равна 15 (большая, чем 7), то алгоритм и программа неверны
Ответ: 4
2. Приведите пример такого трехзначного числа, при вводе которого программа выдает верный ответ.
Сначала определим, что вообще выводит данная неверно написанная программа. Выводится значение digit. В цикле Пока операция digit := N mod 10; выполняется каждый повтор цикла, в последнем повторе переменная digit равна первому (старшему) разряду числа.
Поэтому при вводе 456 — программа выводит 4, при вводе 389 — программа выведет 3. Поэтому мы должны подобрать такое трехзначное число, чтобы старший разряд этого числа был равен сумме цифр числа, которые меньше 7. Примером может быть, к примеру, число 936 — сумма разрядов, меньших 7 равна 9-ти, на экран выводится старший разряд, то есть 9.
Ответ: 936
3. Найдите все ошибки в этой программе.
Ошибка 1. Каждый повтор цикла при выполнении условия к переменной sum прибавляется единица. Программа должна искать сумму разрядов, а не их количество, то есть к переменной sum должна прибавляться переменная digit:
строку sum := sum + 1; нужно заменить на sum := sum + digit;
Ошибка 2. Программа выводит значение переменной digit, должна же выводить значение переменной sum. Заменим строку writeln(digit) на writeln(sum)
Ответ: sum:= sum + 1;заменяем на **sum:= sum + digit; writeln(digit)** заменяем на **writeln(sum)**

Практическая задача 4.

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x, y – действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы (см. рисунок). Программист торопился и написал программу неправильно:

Код программы на языке Pascal:

```
var x,y: real;
begin
readln(x,y);
if y <= 1 then
  if x >= 0 then
    if y >= sin(x) then
      write('принадлежит')
else  write('не принадлежит')
end.
```



Последовательно выполните следующее:

1) Приведите пример таких чисел x, y, при которых программа неверно решает поставленную задачу.
2) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

Решение.

Алгоритм:

1. Нужно написать условие, которым должны отвечать точки, попавшие в выделенную область.
2. Разбираем, как должны быть записаны условия в программе.

3. Отмечаем точки, при которых выполняется оператор write('принадлежит‘).
4. Выясняем, когда программа выдает сообщение «не принадлежит» (с помощью блок-схемы).
5. Приводим примеры входных данных, на которых программа работает неверно.
6. Исправляем исходную программу.

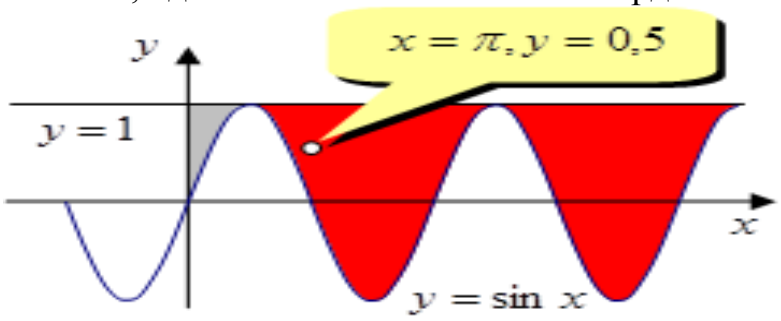
Условия нахождения заштрихованной области (см. рисунок):

1) ограничена по x: справа от оси y, что равносильно условию $x \geq 0$ (с учетом границы здесь и далее получаем нестрогие неравенства); слева от первого максимума функции y=sinx; так как функция sin достигает максимума при $\pi/2$, поэтому получаем второе условие $x \leq \pi/2$.

2) ограничена с двух сторон по координате y: она находится ниже линии y, откуда следует третье условие $y \leq 1$, выше линии y=sinx, что дает четвертое условие $y \geq \sin x$.

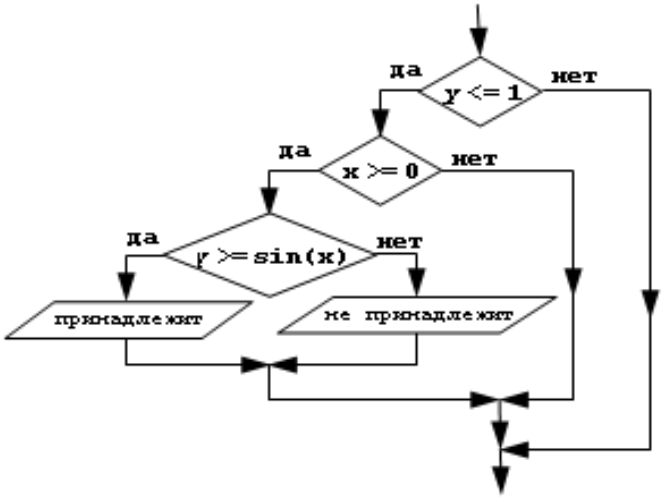
2. Но в программе имеется только три оператора, отсутствует условие $x \leq \pi/2$. Значит нужно добавить это условие в программу.

3. Выполняется в программе write('принадлежит‘), хотя по приведенному рисунку данные точки не принадлежит заданной области; одна из таких точек имеет координаты ($x = \pi$, y=0,5).



4. Сообщение «не принадлежит» анализируем по фрагменту и соответствующей блок-схеме:

```
if y <= 1 then
  if x >= 0 then
    if y >= sin(x) then
        write('принадлежит')
    else write('не принадлежит')
```



Блок-схема показывает, что координаты любой точки, для которой y > 1 или x < 0, являются примером набора входных данных, при которых программа работает неправильно. Значит, примеры входных данных, на которых программа работает неверно: (x=3.14, y=0.5) (неправильно определяет принадлежность точки области) (x=0, y=2) или (x=-1, y=0) (не выдает вообще никакого сообщения).

Корректировка программы:

```
if x <= pi/2 then   if y <= 1 then     if x >= 0 then
     if y >= sin(x) then write('принадлежит')
     else write('не принадлежит')
   else write('не принадлежит')
 else write('не принадлежит')
else write('не принадлежит');
```

Более правильное написание программы с использованием логических связок:

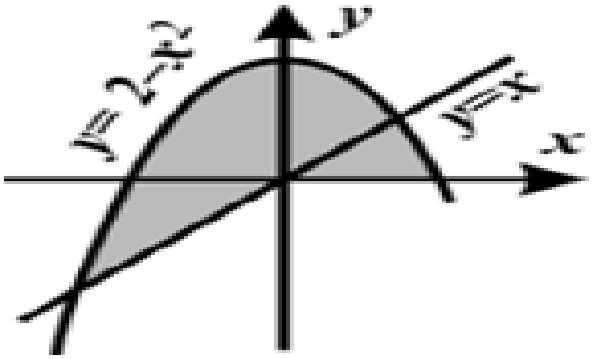
```
var x,y: real;
begin
readln(x,y);
if (x >= 0) and (x <= pi/2) and
   (y <= 1) and (y >= sin(x)) then
    write('принадлежит')
else write('не принадлежит');
end.
```

Практическая задача 5.

Требовалось написать программу, которая вводит с клавиатуры координаты точки на плоскости (x, y – действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы (см. рисунок). Программист торопился и написал программу неправильно.

Код программы на языке Pascal:

```
var x,y: real;
begin
readln(x,y);
if y>=x then
  if y>=0 then
    if y<=2-x*x then
      write('принадлежит')
    else
      write('не принадлежит')
end.
```



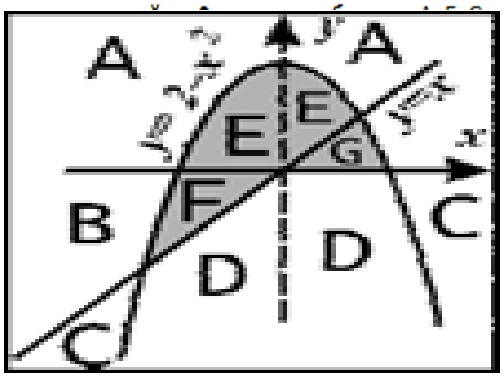
Последовательно выполните следующее:

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям (A, B, C, D, E, F и G). Точки, лежащие на границах областей, отдельно не рассматривать. В столбцах условий укажите "да", если условие выполнится, "нет" если условие не выполнится, "—" (прочерк), если условие не будет проверяться, «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области. В столбце "Программа выведет" укажите, что программа выведет на экран. Если программа ничего не выводит, напишите "—" (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите "да" или "нет".
2. Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы).

Решение.

Алгоритм: используем указанную таблицу и график с выделенными областями.

Область	y>=x?	y>=0?	y<=2-x*x?	вывод	верно?
A	да				
B	да				
C	нет				
D	нет				
E	да				
F	да				
G	нет				



1. Заполняем таблицу, выписывая истинность каждого из трех условий:

1) условие y>=x истинно выше прямой y=x, то есть в областях A, B, E, F
2) условие **y>=0** истинно выше прямой **y=0** (в областях A, E, G), однако это условие проверяется только тогда, когда первое условие, y>=x, истинно; поэтому для всех областей, где первое условие неверно (это области C, D, G), сразу в столбце второго условия ставим прочерк (условие не будет проверяться).
3) третье условие выполняется «внутри» параболы, то есть для E, F, G, D; однако оно проверяется только тогда, когда первые два истинны (для A и E), в остальных строках ставим прочерк.
4) как следует из текста программы, она выведет какое-то сообщение на экран только в том случае, когда выполняются первые два условия, и программа выходит на третье: для области A будет выведено «не принадлежит», для области E – «принадлежит», именно в этих двух случаях программа работает правильно, в остальных – нет. В результате разбора всех условий таблица будет иметь следующий вид:

Область	y>=x?	y>=0?	y<=2-x*x?	вывод	верно?
A	да	да	нет	не принадлежит	да
B	да	нет	-	-	нет
C	нет	-	-	-	нет
D	нет	-	-	-	нет
E	да	да	да	принадлежит	да
F	да	нет	-	-	нет
G	нет	-	-	-	нет

5) Для того, чтобы доработать программу, нужно составить одно сложное условие, описывающее всю заштрихованную область, то есть удобно представить данную область в виде объединения областей, первая из которых включает области E+G, а вторая – области E+F:

- область E+G соответствует условию **(y>=0) and (y <=2-x*x)**
- область E+F соответствует условию **(y>=x) and (y <=2-x*x)**

6) объединение областей выполняется с помощью операции ИЛИ (**or**), так что полное условие принимает вид:

(y>=0) and (y <=2-x*x) or (y>=x) and (y <=2-x*x)

Операция И (**and**) имеет более высокий приоритет, чем ИЛИ (**or**), порядок выполнения операций тут правильный; в случае сомнений можно поставить дополнительные скобки:

((y>=0) and (y <=2-x*x)) or ((y>=x) and (y <=2-x*x))

В обоих условиях есть условие **y <=2-x*x,** поэтому запись можно сократить: **(y <=2-x*x) and ((y>=x) or (y>=0)).**

7) В результате доработанная программа выглядит следующим образом:

```
var x,y: real;
begin
readln(x,y);
if (y <=2-x*x) and ((y>=x) or (y>=0)) then
  write('принадлежит')
else
  write('не принадлежит')
end.
```

Рекомендации по решению данных задач на ЕГЭ.

Возможные проблемы:

1. прежде всего, необходимо иметь знания из курса математики (решение уравнений, графики функций, область допустимых значений, составление уравнений прямой по приведенному графику).

2. Нужно уметь разбираться в серии вложенных условных операторов в полной и неполной форме.
3. Чтобы разобраться в программе, лучше предварительно построить блок-схему алгоритма и сверить по ней запись программы.
4. Нужно внимательно проверять, всегда ли программа выдает сообщение, если заданное условие не выполняется.
5. Полезно построить детальную блок-схему алгоритма, которая позволяет увидеть ход выполнения программы при всех возможных вариантах.
6. Следует учесть, что при оценке работы можно (при абсолютно правильном решении) потерять баллы из-за синтаксических ошибок в программе (скобки, точки с запятой, неправильное написание оператора и т.п.)

За что снимают баллы:

1. Неправильно определены входные данные, при которых исходная программа работает неверно.
2. Исправлены не все ошибки в программе, например, в данной задаче легко «просмотреть», что необходимо еще условие $x \le \pi/2$.
3. Программа работает правильно в большем количестве случаев, чем исходная, но не для всех возможных исходных данных.
4. Перепутаны знаки < и >, логические операции or и and.
5. Неверно расставлены операторные скобки begin-end.
6. Синтаксические ошибки (знаки пунктуации – запятые, точки, точки с запятой; неверное написание ключевых слов).

Чтобы получить максимальные 3 балла за данное задание ЕГЭ, нужно при абсолютно правильном решении сделать не более одной синтаксической ошибки; на 2 балла – до двух ошибок, на 1 балл – до трех ошибок.

Возможные ошибки (при использовании языка Pascal).

1. Чтобы не запутаться, к какому оператору относится else, используйте следующее правило: если перед else нет слова end, нужно искать ближайший сверху условный оператор if, если перед else стоит end (конец блока), нужно искать парный ему begin (начало блока) и соответствующий условный оператор if ... then begin.
2. Проверяйте, все ли необходимые условия учтены в программе, это особенно актуально для немонотонных функций типа синуса или косинуса (немонотонные функции на некоторых участках возрастают при увеличении аргумента, на некоторых – убывают); например, в некоторых задачах можно пропустить необходимость выполнения условия $x \le \pi/2$

3. Не перепутайте в решениях, где нужно использовать операцию and («И», одновременное выполнение условий), где – or («ИЛИ», хотя бы одно условие).
4. Нужно внимательно проверять, всегда ли программа выдает сообщение, если заданное условие не выполняется, поэтому часто бывает полезно построить блок-схему алгоритма, которая позволяет увидеть ход выполнения программы при всех возможных вариантах.
6. Проверяйте, включает ли заданная область свои границы; если включает –будут нестрогие неравенства (<=, >=), если не включает – строгие (<, >).
7. Не забывайте, что в сложном условии все простые условия (отношения) нужно брать в скобки, так как в языке Pascal отношения при вычислении логического выражения имеют самый низкий приоритет.
8. помните, что перед else точка с запятой никогда не ставится, в конце программы после последнего end ставится точка.

Практическое занятие 14. Задачи на решение записи алгоритмов на естественном языке или коротких простых программ на языке программирования

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 14, которые являются задачами демонстрационного варианта ЕГЭ за 2018 год.

Домашние задания к занятию 15.

Задача 16.

Значение арифметического выражения: $\mathbf{49^{10} + 7^{30} - 49}$ – записали в системе счисления с основанием 7. Сколько цифр «6» содержится в этой записи?

Решение.

Алгоритм: задачи на системы счисления.

Шаг 1. Исходную запись уравнения приводим к стандартной форме показательного уравнения с основанием 7:

$49^{10}+7^{30}-49=7^{30}+7^{20}-7^{2}$.

Шаг 2. В таблице решаем полученное уравнение, учитывая, что максимальный весовой коэффициент в системе счисления с основанием 7 равен 6.

10000…0000	730 – 30 нулей	Первое слагаемое
100…000	720– 20 нулей	Второе слагаемое
100..100…000	9 нулей+ «1»+20 нулей	Сумма (уменьшаемое)
100	72– 2 нулей	Вычитаемое
…06666…66600	Содержится: 20-2= 18 «6»	Полученная разность

Ответ: 18

Задача 17.

В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для обозначения логической операции «И» – символ «&». В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет. Какое количество страниц (в сотнях тысяч) будет найдено по запросу Бабочка & Гусеница?

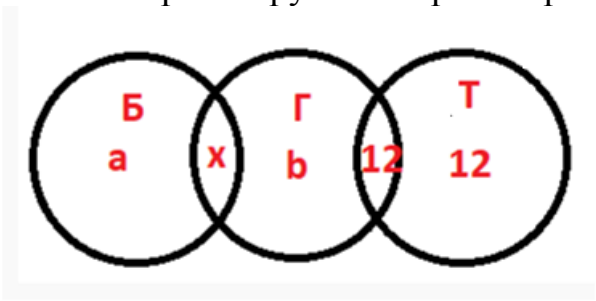
Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Запрос	Найдено страниц (в сотнях тысяч)
Бабочка	22
Гусеница	40
Трактор	24
Трактор | Бабочка | Гусеница	66
Трактор & Гусеница	12
Трактор & Бабочка	0

Решение.

Алгоритм с использованием кругов Эйлера.

Шаг 1. Строим круги Эйлера по приведенной таблице:



Шаг 2. Анализ рисунка:

Обозначим выражение Бабочка&Гусеница= х, из рисунка значит:

Бабочка=a+x=22.

Трактор&Гусеница=12, значит:

Бабочка&Гусеница|Трактор&Гусеница=

Гусеница&(Бабочка|Трактор)=x+12, Гусеница= b+x+12=40, отсюда:

b+x=28.

Шаг 3. По таблице и рисунку: Трактор|Бабочка|Гусеница =

a+x+b+12+12=66

Отсюда: a+x+b=42, значит: a=42-28=14

Шаг 4. Так как Бабочка =a+x=22, отсюда: x =22-14 = 8

Ответ: 8

Задача 18.

Для какого наибольшего целого числа A формула:

$((x \le 9) \rightarrow (x*x \le A)) \wedge ((y*y \le A) \rightarrow (y \le 9))$ тождественно истинна, то есть принимает значение 1 при любых целых неотрицательных x и y?

Решение.

Алгоритм: решение логических неравенств.

Шаг 1. Анализ неравенства:

Так как неравенство является конъюнкцией, то оно истинно, если обе из его составляющих (импликат) будут истинны.

Шаг 2. Преобразование импликат и решение неравенств относительно А:

1) $((x \le 9) \rightarrow (x*x \le A))$ = x>9 + x*x≤A, решаем относительно А, отсюда: A≥81

2) $((y*y \le A) \rightarrow (y \le 9)$=y*y>A + y≤9, решаем относительно А, отсюда: при y=10 100>A.

Шаг 3. Выбираем наибольшее число A=99, так как в этом случае A≥81 и A<100

Ответ: 99

Задача 19.

В программе используется одномерный целочисленный массив A с индексами от 0 до 9. Значения элементов равны 3, 0, 4, 6, 5, 1, 8, 2, 9, 7 соответственно, т.е. A[0] = 3, A[1] = 0 и т.д.

Определите значение переменной с после выполнения следующего фрагмента этой программы (записанного ниже на разных языках программирования).

Бейсик

```
c = 0
FOR i = 1 TO 9
  IF A(i-1) > A(i) THEN
    c = c + 1
    t = A(i)
    A(i) = A(i-1)
    A(i-1) = t
  END IF
NEXT i
```

Python

```
c = 0
for i in range(1,10):
  if A[i-1] > A[i]:
    c = c + 1
    t = A[i]
    A[i] = A[i-1]
    A[i-1] = t
```

Алгоритмический язык

```
c := 0
нц для i от 1 до 9
  если A[i-1] > A[i] то
    c := c + 1
    t := A[i]
    A[i] := A[i-1]
    A[i-1] := t
  все
кц
```

Паскаль

```
c := 0;
for i := 1 to 9 do
  if A[i-1] > A[i] then
  begin
    c := c + 1;
    t := A[i];
    A[i] := A[i-1];
    A[i-1] := t;
  end;
```

C++

```
c = 0;
for (int i = 1; i < 10; i++)
    if (A[i-1] > A[i]){
        c++;
        t = A[i];
        A[i] = A[i-1];
        A[i-1] = t;
    }
```

Решение.

Шаг 1. Анализ алгоритма:

В построенной нами вычислительной таблице приводим подробное решение по данному алгоритму. Согласно условию значения элементов одномерного массива A[0] до A[9] последовательно сравниваются в цикле. На первом шаге i=1 значение элемента A[i-1]= A[0]=3 сравнивается со значением элемента A[1]=0. Так как цикл выполняется при значениях A[i-1] >A[i1], то значение переменной с

увеличивается на 1 (c=0+1=1), далее в некоторую переменную t записывается значение второго элемента A[1], а затем значение первого элемента массива A[0] записывается во второй элемент массива A[1], наконец, значение первого элемента переписывается в переменную t. Данная последовательность действий позволяет осуществить обмен значениями первого и второго элементов массива со сохранением значения первого элемента в переменной t. На втором шаге значение второго элемента A[1] соответственно равно 3, третий же элемент A[2] имеет значение 4, значит, цикл обмена не выполняется, и значение c остается равным 1. Далее, согласно таблице, осуществляется обмен значениями A[3] и A[4] с увеличением с до 2, A[5] и A[6] с увеличением с до 3, обмен A[6] и A[7] с итоговым c=4, и наконец, обмен значениями элементов A[8] и A[9] в результате которого c=5. Таким образом, переменная с показывает количество обменов значениями элементов с большего на меньшее значение в одномерном массиве.

	i	0	1	2	3	4	5	6	7	8	9	
Цикл	Исх.A[i]	3	0	4	6	5	1	8	2	9	7	c=0
Шаг 1	Пол.A[i]	0	3									c=1
Шаг 2	Пол.A[i]				5	6						c=2
Шаг 3	Пол.A[i]					1	6					c=3
Шаг 4	Пол.A[i]							2	8			c=4
Шаг 5	Пол.A[i]									7	9	c=5

Ответ: 5

Задача 20.

Ниже на пяти языках программирования записан алгоритм. Получив на вход число *x*, этот алгоритм печатает два числа: *L* и *M*. Укажите наименьшее число *x*, при вводе которого алгоритм печатает сначала 5, а потом 7.

C++

```
#include <iostream>
using namespace std;

int main(){
    int x, L, M;
    cin >> x;
    L = 0;
    M = 0;
    while (x > 0) {
        M = M + 1;
        if(x % 2 != 0) {
            L = L + 1;
        }
        x = x / 2;
    }
    cout << L << endl << M << endl;
    return 0;
}
```

Бейсик

```
DIM X, L, M AS INTEGER
INPUT X
L = 0
M = 0
WHILE X > 0
  M = M + 1
  IF X MOD 2 <> 0 THEN
    L = L + 1
  END IF
X = X \ 2
WEND
PRINT L
PRINT M
```

Python

```
x = int(input())
L = 0
M = 0
while x > 0:
    M = M + 1
    if x % 2 != 0:
        L = L + 1
    x = x // 2
print(L)
print(M)
```

Алгоритмический язык

```
алг
нач
  цел x, L, M
  ввод x
  L := 0
  M := 0
  нц пока x > 0
    M := M + 1
    если mod(x,2) <> 0
      то
      L := L + 1
    все
    x := div(x,2)
  кц
  вывод L, нс, M
кон
```

Паскаль

```
var x, L, M: integer;
begin
  readln(x);
  L := 0;
  M := 0;
  while x>0 do
  begin
    M := M + 1;
    if x mod 2 <> 0 then
      L := L + 1;
    x := x div 2;
  end;
  writeln(L);
  writeln(M);
end.
```

Решение.

Алгоритм с циклами и ветвлениями.

Шаг 1. Анализ алгоритма:

1) Возьмем произвольное число x (например, 19) и выполним программу, построив соответствующую вычислительную таблицу:

Обозначение	**x**	**Условие x mod 2 <>0**	**L**	**M**
Начальные значения	19	-	0	0
Итерация 1	9	1 (Да)	1	1
Итерация 2	4	1 (Да)	2	2
Итерация 3	2	0 (Нет)	2	3
0				

Таким образом, M подсчитывает количество итераций цикла, L подсчитывает количество раз, которое при делении текущего числа x на 2 дает остаток 1.

Все это означает, что обработка текущего числа x в результате выполнения программы преобразует его десятичный формат в двоичный формат.

Шаг 2. Определение искомого значения **x.**

Число x должно удовлетворять следующим условиям:

1) Двоичное представление содержит всего M(Сумма цифр)=7 и L(Сумма единиц)=5.

2) Двоичное число должно быть наименьшим.
   Очевидно, таким числом является 1001111_2. Переводим данное число в десятичное: $1001111_2=79_{10}$.

Ответ: 79

Практические задачи данного занятия относятся к заданию части 2 ЕГЭ (25 задание) с высоким уровнем сложности. На практике предполагаемый процент выполнения заданий высокого уровня составляет менее 40 % экзаменующих. Как правило, проверяемые элементы содержания относятся к умениям написать короткую (10–15 строк) простую программу на языке программирования. Обычно в данных задачах используют задания на обработку элементов одномерного и двумерного массивов по заданным правилам. Можно указать следующие алгоритмы работы с массивами:

− Операции с элементами массива (линейный поиск элемента. Вставка и удаление элементов в массиве. Перестановка элементов данного массива в обратном порядке. Суммирование элементов массива. Проверка соответствия элементов массива некоторому условию).

− Нахождение второго по величине (второго максимального или второго минимального) значения в данном массиве за однократный просмотр массива.

− Нахождение минимального (максимального) значения в данном массиве и количества элементов, равных ему, за однократный просмотр массива.

− Операции с элементами массива, отобранных по некоторому условию (например, нахождение минимального четного элемента в массиве, нахождение количества и суммы всех четных элементов в массиве).

− Сортировка массива.

− Слияние двух упорядоченных массивов в один без использования сортировки.

Хотя в 2018 году в задании 25 по стандарту была убрана возможность написания алгоритма на естественном языке в связи с не востребованностью этой возможности участниками экзамена, на занятии данный вариант решения мы оставляем.

Практическая задача 1.

Дан массив, содержащий неотрицательные целые числа. Если сумма всех элементов массива чётная, нужно вывести количество нечётных (по значению) элементов массива, если нечётная – количество чётных.

Например, для массива из 6 элементов, равных соответственно 2, 6, 12, 17, 3, 8, ответом будет 2 – количество нечётных элементов, так как общая сумма всех элементов чётна.

Напишите на одном из языков программирования программу для решения этой задачи. Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из описанных.

Код программы на языке Pascal:

```
const N=2000;
var a: array [1..N] of integer; i, k: integer;
begin
   for i:=1 to N do
      readln(a[i]);
   …
end.
```

В качестве ответа Вам необходимо привести фрагмент программы, который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Free Pascal 2.6). В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в приведённых фрагментах.

Решение.

Алгоритм: подсчет количества нечетных элементов массива.

Хотя для стандартного решения задачи на языке Pascal необходимо использовать 4 переменные: счетчик для перебора элементов, счетчики четных и нечетных элементов, сумма всех элементов, в условии разрешено использовать только 2 переменные.

Чтобы выполнить данное условие значение суммы можно не вычислять, так как требуется только четность суммы, которая однозначно определяется количеством нечетных элементов. Количество четных элементов, если оно потребуется, можно вычислить, зная общее число элементов и количество нечетных.

С учетом данного алгоритма записываем недостающий фрагмент программы:

```
k:=0;
for i:=1 to N do begin
   if a[i] mod 2 = 1 then k:=k+1;
end;
if k mod 2 = 0  then writeln(k)
else writeln(N-k);
```

Также можно применить такой алгоритм: на первом проходе определяется общая сумма, на втором количество элементов нужной четности.

Недостающий фрагмент программы:

```
k:=0;
begin
 for i:=1 to N do   k:=k+a[i]
end;
   if  k mod 2 = 1 then k:=0;
      begin
  for i:=1 to N do
   if a[i] mod 2 = 1 then k:=k+1;
end;
else       k:=0;
       for i:=1 to N do
    if a[i] mod 2 = 1 then k:=k+1;
end;
writeln(k);
```

Практическая задача 2.

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать целые значения от 0 до 10000 включительно. Опишите на естественном языке или на одном из языков программирования алгоритм, позволяющий найти и вывести максимальное значение среди двузначных элементов массива, не делящихся на 3. Если в исходном массиве нет элемента, значение которого является двузначным числом и при этом не кратно трем, то выведите сообщение «Не найдено».

Исходные данные объявлены так, как показано ниже на примерах для некоторых языков программирования и естественного языка. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать некоторые из описанных переменных.

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Free Pascal 2.4) или в виде блок-схемы. В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Естественный язык:

Объявляем массив A из 40 элементов.

Объявляем целочисленные переменные I, J, MAX.
В цикле от 1 до 40 вводим элементы массива A с 1-го по 40-й.
…

Код программы на языке Pascal:

```
const   N = 40;
var      a: array [1..N] of integer;     i, j, max: integer;
begin
  for i := 1 to N do
    readln(a[i]);
  ...
end.
```

Решение.

На естественном языке:

Записываем в переменную MAX начальное значение, равное 9. В цикле от первого элемента до 40-го находим остаток от деления элемента исходного массива на 3. Если значение данного остатка не равно 0, и значение текущего элемента массива больше 9 и меньше 100, то сравниваем значение текущего элемента массива со значением переменной MAX. Если текущий элемент массива больше MAX, то записываем в MAX значение этого элемента массива. Переходим к следующему элементу.

После завершения цикла проверяем значение переменной MAX. Если оно больше 9, то выводим его, иначе выводим сообщение «Не найдено».

Код фрагмента программы на языке Pascal:

```
max := 9;
for i := 1 to N do
  if (a[i]>=10) and (a[i]<=98) and (a[i] mod 3<>0) and (a[i]>max) then
    max := a[i];
if max > 9 then writeln(max)
else writeln('Не найдено');
```

Практическая задача 3.

Дан массив, содержащий 2015 неотрицательных целых чисел. Пиком называется не крайний элемент массива, который больше обоих своих соседей. Необходимо найти в массиве самый высокий пик, то есть пик, значение которого максимально. Если в массиве нет ни одного пика, ответ считается равным 0.

Например, в массиве из шести элементов, равных соответственно 4, 9, 2, 17, 3, 8, есть два пика – 9 и 17, максимальный пик равен 17.

Напишите на одном из языков программирования программу для решения этой задачи. Исходные данные объявлены так, как показано

ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из описанных.

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Free Pascal 2.4) или в виде блок-схемы. В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Код программы на языке Pascal:

```
const  N=2015;
var a: array [1..N] of integer; i, j, k: integer;
begin
for i:=1 to N do readln(a[i]);
…
end.
```

Решение.

Алгоритм: Необходимо перебрать все элементы массива, кроме первого и последнего (они не могут быть пиками, так как имеют всего по одному соседу), выделить пики и выбрать из них максимальный элемент.При этом недостаточно просто выбрать максимальный элемент массива, так как он может не быть пиком (например, если несколько одинаковых элементов идут подряд). Решение, основанное на поиске максимума без проверки пиков, не считается верным и оценивается 0 баллов.

Код фрагмента программы на языке Pascal:

```
k:=0;
for i:=2 to N-1 do begin
if (a[i]>a[i-1]) and (a[i]>a[i+1]) and (a[i]>k)
then k:=a[i];
end;
writeln(k);
```

Практическая задача 4.

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет подсчитать и вывести среднее арифметическое элементов массива, имеющих нечетное значение. Гарантируется, что в исходном массиве хотя бы один элемент имеет нечетное значение.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

В качестве ответа необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Естественный язык:

Объявляем массив A из 30 элементов.
Объявляем целочисленные переменные I, X, Y.
Объявляем вещественную переменную S.
В цикле от 1 до 30 вводим элементы
массива A с 1-го по 30-й.
…

Код программы на языке Pascal:

```
const N=30;
var a: array [1..N] of integer; i, x, y: integer; s: real;
begin
for i:=1 to N do readln(a[i]);
…
end.
```

Решение.

Естественный язык:

Записываем в переменные x и y начальное значение, равное нулю. В цикле от первого элемента до тридцатого находим остаток от деления элемента исходного массива на два.
Если этот остаток равен единице, то увеличиваем счетчик суммы x на значение текущего элемента массива, а счетчик количества y на 1. Переходим к следующему элементу.
После цикла производим деление счетчика суммы x на счетчик количества y и записываем результат в переменную s. Выводим значение переменной s.

Код фрагмента программы на языке Pascal:

```
x:=0; y:=0;
for i:=1 to N do
if (a[i] mod 2=1) then begin
```

```
x:=x+a[i];
y:=y+1;
end;
s:=x/y; writeln(s);
```

Практическая задача 5.

Дан массив, содержащий 2015 неотрицательных целых чисел. Ямой называется не крайний элемент массива, который меньше обоих своих соседей. Необходимо найти в массиве самую глубокую яму, то есть яму, значение которой минимально. Если в массиве нет ни одной ямы, ответ считается равным 0.

Например, в массиве из шести элементов, равных соответственно 4, 9, 2, 17, 3, 8, есть две ямы – 2 и 3, самая глубокая яма – 2.

Напишите на одном из языков программирования программу для решения этой задачи. Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из описанных переменных.

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например, Free Pascal 2.4) или в виде блок-схемы. В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Код программы на языке Pascal:

```
const  N=2015;
var a: array [1..N] of integer; i, j, k: integer;
begin
for i:=1 to N do readln(a[i]);
…
end.
```

Решение.

Алгоритм: Необходимо перебрать все элементы массива, кроме первого и последнего (они не могут быть ямами, так как имеют всего по одному соседу), выделить ямы и выбрать из них минимальную яму. При этом недостаточно просто выбрать минимальный элемент массива, так как он может не быть ямой (например, если несколько одинаковых элементов идут подряд).

Код фрагмента программы на языке Pascal:

```
k:=-1;
for i:=2 to N-1 do begin
```

```
if ((k=-1) or (a[i] < k)) and (a[i] < a[i-1]) and (a[i] < a[i+1]) then
k:=a[i];
end;
if (k=-1) then writeln('0') else writeln(k);
```

Практическая задача 6.

Дан целочисленный массив из 40 элементов. Элементы массива могут принимать произвольные значения. Опишите на русском языке или на одном из языков программирования алгоритм, который находит и выводит сумму элементов наибольшей возрастающей последовательности подряд идущих элементов массива.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать часть из них.

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

Естественный язык:

Объявляем массив A из 40 элементов.
Объявляем целочисленные переменные I, L, Lmax, S, Smax.
В цикле от 1 до 40 вводим элементы массива A с 1-го по 40-й.
...

Код программы на языке Pascal:

```
const  N = 40;
var a: array [1..N] of integer; i, l, lmax, s, smax:
integer;
begin
for i : = 1 to N do readln(a[i]);
...
end.
```

Решение.

Естественный язык (алгоритм):

Записываем в переменную Lmax начальное значение, равное нулю, в переменную L — начальное значение, равное единице, в переменную S — начальное значение, равное первому элементу массива.

В цикле перебираем все элементы со 2-го до 40-го. Если значение текущего элемента массива оказывается больше значения предыдущего элемента: увеличиваем значение переменной L на 1;
увеличиваем значение переменной S на значение текущего элемента. Иначе: если значение переменной L больше значения переменной Lmax, то переменной Lmax присваиваем значение переменной L, а переменной Smax присваиваем значение переменной S; переменной L присваиваем значение 1; переменной S присваиваем значение текущего элемента массива.

После окончания цикла, если значение переменной L оказалось больше значения переменной Lmax, присваиваем переменной Smax значение переменной S. Выводим значение переменной Smax.

Код фрагмента программы на языке Pascal:

```
lmax : = 0 ; l := 1; s := a[1];
for i : = 2 to N do if a[i] > a[i - 1] then
begin
l := l + 1;
s:=s+a[i];
end
else
begin
if l>lmax, then
begin
lmax:=l;
smax : = s
end;
l := 1; s := a[i]
end ;
if l > lmax then smax : = s;
writeln(smax);
```

Указания авторов: на данном занятии при решении практических задач проводится обязательная проверка рекомендуемых алгоритмов и решений с компиляцией программ в среде программирования Pascal. Таким образом, должна полностью исключаться возможность ошибок в данном пособии.

Практическое занятие 15. Практические задачи на построение дерева игры и обоснование выигрышных стратегий

Введение. Проводятся разборы алгоритмов и решений слушателями домашнего задания занятия № 15, которые являются задачами демонстрационного варианта ЕГЭ за 2018 год.

Домашние задания к занятию 16.

Задача 21.

Напишите в ответе число, которое будет напечатано в результате выполнения следующего алгоритма. Для Вашего удобства алгоритм представлен на нескольких языках программирования.

Алгоритмический язык

```
алг
нач
цел a, b, t, M, R
  a:=-20; b:=20
  M:=a; R:=F(a)
  нц для t от a до b
    если F(t) <=  R то
        M:=t; R:=F(t)
    все
  кц
  вывод M+R
кон
алг цел F(цел x)
нач
  знач:=2*(x*x-1)*(x*x-1)+27
кон
```

Паскаль

```
var a, b, t, M, R :longint;
function F(x: longint) :
longint;
  begin
    F:= 2*(x*x-1)*(x*x-1)+27;
  end;
begin
  a:=-20; b:=20;
  M:=a; R:=F(a);
  for t:= a to b do begin
    if (F(t) <= R) then begin
      M:=t;
      R:=F(t)
    end
  end;
  write(M+R)
end.
```

C++

```
#include <iostream>
using namespace std;


long F(long x) {
    return 2*(x*x-1)*(x*x-1)+27;
}


int main() {
    long a = -20, b = 20, M = a, R = F(a);
    for (int t = a; t <= b; ++t) {
        if (F(t)<= R) {
            M = t; R = F(t);
        }
    }
    cout << M + R;
    return 0;
}
```

Решение.

Шаг 1. Анализ алгоритма:

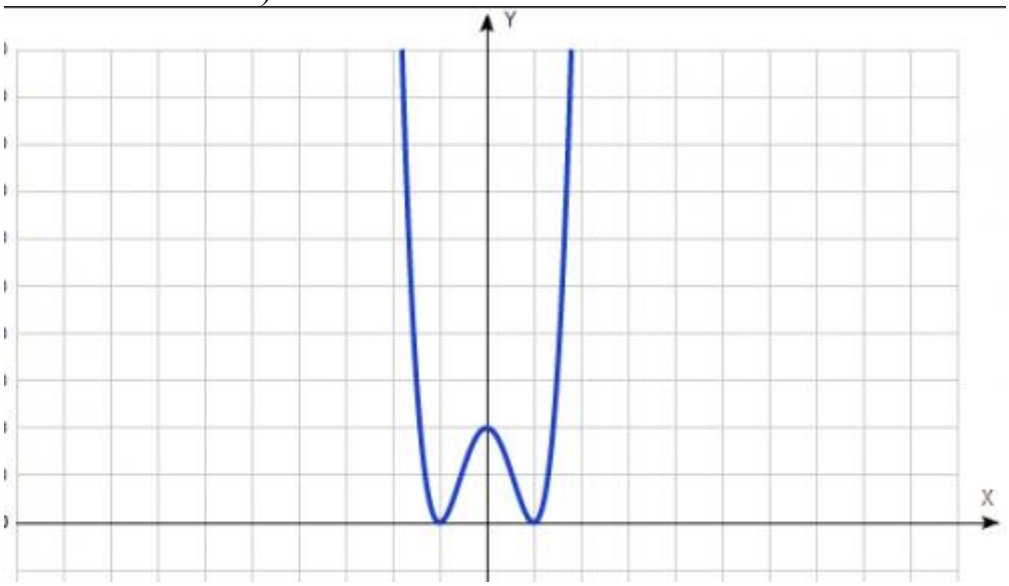
- В цикле программы ищется минимальное значение, возвращаемое функцией **F(t)**, вызываемой с параметрами от **-20** до **20**.

- После завершения работы цикла минимальное значение функции помещается в переменную **R**, а параметр, при котором функция возвратила минимальное значение, помещается в переменную **M**.
- В конце программы выводится результат **M+R**.
- В теле функция равна: **2*(x*x-1)*(x*x-1)+27**
- При раскрытии скобок получится уравнение 4-й степени, корни которого находятся через вычисление производной функции. Поэтому в дальнейшем пока не будем учитывать число **27** в выражении.

Шаг 1. Нахождение искомого значения:

- Преобразуем выражение, раскрыв скобки:

F(x) = 2*(x*x-1)*(x*x-1)+27 = $2*x^2-2*x^2-1$ = $2*x^4-2*x^2-2*x^2+2$ = $2*x^4-4*x^2+2$ Получаем уравнение 4-й степени, график которой на рисунке представляет собой квадратичную параболу с ветвями, направленными вверх (первый коэффициент уравнения — положительный).



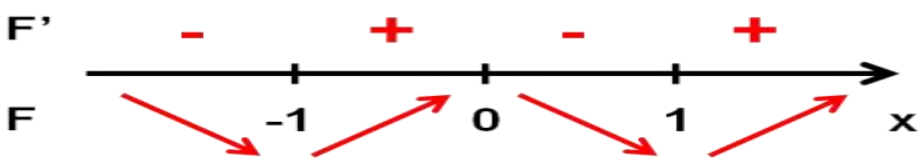
- Вычислим производную функции:

F'(x) = 8x3 - 8x = 8x(x2-1) = 8x(x-1)(x+1)

- Находим нули производной (точки, в которых график пересекает ось ox):

y=0, x=1, x=-1

- Методом интервалов согласно рисунку находим отрезки, в которых график убывает и возрастает:



- Таким образом, функция имеет **два минимума** — в точках **-1** и **1** (оси ox).
- Так как в условии программы стоит F(t) <= R, данное то условие будет работать и при F(1). Проверим:

F(-1) = 2*0*0 = 27

F(1) = 2*0*0 = 27

Таким образом, минимальное значение функции: **27** (R), а параметр при минимальном значении функции: **1** (M).

Результат: **M + R = 28**

Ответ: 28

Задача 22.

Исполнитель M17 преобразует число, записанное на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1

2. Прибавить 2

3. Умножить на 3

Первая из них увеличивает число на экране на 1, вторая увеличивает его на 2, третья умножает на 3.

Программа для исполнителя M17 – это последовательность команд. Сколько существует таких программ, которые преобразуют исходное число 2 в число 12 и при этом траектория вычислений программы содержит числа 8 и 10? Траектория должна содержать оба указанных числа. Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы **132** при исходном числе 7 траектория будет состоять из чисел 8, 24, 26.

Решение.

Шаг 1. Общий алгоритм решения с использованием рекуррентных формул. Для составления рекуррентных формул переменим заданные программы на обратные и запомним, что для получения числа самого из себя существует ровно одна программа, которая ничего не делает.

Условие: ***1. Прибавить 1***, меняем на ***вычти 1*** или кратко (N-1).

Условие: 2***. Прибавить 2***, меняем на ***вычти 2*** или кратко (N-2).

Условие: 3***. Умножить на 3***, меняем на ***раздели на 3*** или кратко (N/3).

Исходя из этого, составляем следующие рекуррентные формулы:

если из заданного числа N можно вычесть только 1, то F(n) = F (n-1),

если заданное число N не делится на 3, а из него можно вычесть 1 и 2, то F(n) = F (n-1)+F(n-2),

если к тому же заданное число N еще делится на 3, то F(n) = F(n/3)+F(n-1)+ +F(n-2).

Таким образом, общая рекуррентная формула имеет следующий вид: F(n)=F(n/3)+F(n-1)+F(n-2).

Шаг 2. Находим искомые значения числа программ согласно приведенной формуле.

Заполняем следующую таблицу, учитывая, что траектория вычислений программы содержит числа 8 и 10:

n= 2	3	4	5	6	7	8	9	10	11	12
Число программ: 1	1	2	3	6	9	15	15	30	30	60

Ответ: 60.

Задача 23.

Сколько существует различных наборов значений логических переменных

$x1, x2, ...x7, y1, y2, ...y7$, которые удовлетворяют всем перечисленным ниже

условиям?

$(\neg x1 \vee y1) \rightarrow (\neg x2 \wedge y2) = 1$

$(\neg x2 \vee y2) \rightarrow (\neg x3 \wedge y3) = 1$

…

$(\neg x6 \vee y6) \rightarrow (\neg x7 \wedge y7) = 1$

В ответе **не нужно** перечислять все различные наборы значений переменных $x1, x2, ...x7, y1, y2, ...y7$, при которых выполнена данная система равенств.

В качестве ответа Вам нужно указать количество таких наборов.

Решение.

Шаг 1. Существенных упрощений в системе уравнений нет.

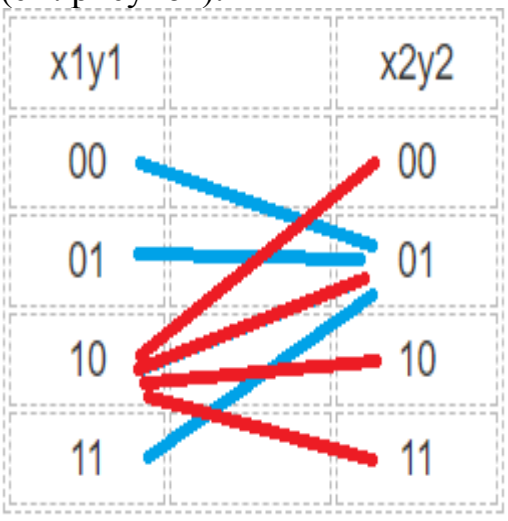
Преобразуем первое уравнение в системе:

$(\neg x1+y1) \rightarrow (\neg x2*y2) = \neg(\neg x1+y1) + (\neg x2*y2) = (x1*\neg y1) + (\neg x2*y2) = 1$

Шаг 2. Данное уравнение представляет собой дизъюнкцию, которое истинно при истинности каждой ее составляющей. Наборы значения переменных (x1, y1, x2, y2) представлены в соответствующей таблице:

x1	y1	x2	y2
0	0	0	1
		1	
	1	0	1
		1	
1	0	0	0
			1
		1	0
			1
	1	0	1
		1	

Шаг 3. Представим двоичное дерево решений для первого уравнения (см. рисунок):



Таким образом, первое уравнение имеет 7 решений.

Шаг 4. Решения для системы уравнений представлены в таблице:

Разрешенные наборы	x1y1	x2y2	x3y3	x4y4	x5y5	x6y6	x7y7
00	1	1	1	1	1	1	1
01	1	4	7	10	13	16	19
10	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1

Итого: 1+19+1+1+1=22

Ответ: 22.

Задача 24.

На обработку поступает натуральное число, не превышающее 10^9. Нужно написать программу, которая выводит на экран максимальную цифру числа, кратную 5. Если в числе нет цифр, кратных 5, требуется на экран вывести «NO». Программист написал программу неправильно. Ниже эта программа для Вашего удобства приведена на нескольких языках программирования.

Напоминание: 0 делится на любое натуральное число.

Алгоритмический язык

```
алг
нач
  цел N, digit, maxDigit
  ввод N
  maxDigit := mod(N,10)
  нц пока N > 0
    digit := mod(N,10)
    если mod(digit, 5) = 0 то
      если digit > maxDigit то
        maxDigit := digit
      все
    все
    N := div(N,10)
  кц
  если maxDigit = 0 то
    вывод "NO"
  иначе
    вывод maxDigit
  все
кон
```

Паскаль

```
var N,digit,maxDigit: longint;
begin
  readln(N);
  maxDigit := N mod 10;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit mod 5 = 0 then
      if digit > maxDigit then
        maxDigit := digit;
    N := N div 10;
  end;
  if maxDigit = 0 then
    writeln('NO')
  else
    writeln(maxDigit)
end.
```

C++

```
#include <iostream>
using namespace std;

int main() {
    long N, digit, maxDigit;
    cin >> N;
    maxDigit = N % 10;
    while (N > 0) {
        digit = N % 10;
        if (digit % 5 == 0)
            if (digit > maxDigit)
                maxDigit = digit;
        N = N / 10;
    }
    if (maxDigit == 0)
        cout << "NO" << endl;
    else
        cout << maxDigit << endl;
    return 0;
}
```

Последовательно выполните следующее.
1. Напишите, что выведет эта программа при вводе числа 132.
2. Приведите пример такого трехзначного числа, при вводе которого программа выдаёт верный ответ
3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
1) выпишите строку, в которой сделана ошибка;
2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.
Достаточно указать ошибки и способ их исправления для одного языка программирования.
Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Решение.

Выполним задания последовательно.

1. Вывод программы при вводе числа 132. В maxDigit изначально запишется максимальная цифра числа 3. При этом в числе нет ни одной цифры, кратной 5, поэтому ответ обновлен не будет и результатом работы программы будет **3.**

2. **555**. Изначально в maxDigit запишется 5. Чисел, делящихся на 5 и больших, чем 5, нет, поэтому ответ не обновится ни разу за время работы программы, и итоговым результатом будет 5.

3. Рассматриваем программу для языка Pascal:

Первая ошибка: Неверная инициализация переменной maxDigit:
maxDigit := N mod 10;

Следует проинициализировать ее числом, меньшим нуля, например, -1, что будет означать, что пока в числе не найдена цифра, которая делится на 5:
maxDigit := -1;

Вторая ошибка: 0 может быть ответом, поэтому ошибка в строчках:
if maxDigit = 0 then writeln('NO')

Так как решено изначально проинициализировать maxDigit как -1, то именно это значение нужно использовать для обнаружения отсутствия ответа:
if maxDigit = -1 then writeln('NO')

Задача 25.

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит количество элементов массива, больших 100 и при этом кратных 5, а затем заменяет каждый такой элемент на число, равное найденному количеству. Гарантируется, что хотя бы один такой элемент в массиве есть. В качестве результата необходимо вывести измененный массив, каждый элемент массива выводится с новой строчки. Например, для массива из шести элементов: 4 115 7 195 25 106 программа должна вывести числа 4 2 7 2 25 106

Исходные данные объявлены так, как показано ниже на примерах для некоторых языков программирования. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать некоторые из описанных переменных.

В качестве ответа Вам необходимо привести фрагмент программы, который

должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Free Pascal 2.6). В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на Алгоритмическом языке).

Алгоритмический язык

```
алг
нач
  цел N = 30
  цел таб a[1:N]
  цел i, j, k
  нц для i от 1 до N
    ввод a[i]
  кц
  ...


кон
```

Паскаль

```
const
N = 30;
var
a: array [1..N] of longint;
i, j, k: longint;
begin
    for i := 1 to N do
        readln(a[i]);
    ...


end.
```

C++

```
#include <iostream>
using namespace std;
const int N = 30;
int main() {
long a[N];
long i, j, k;
    for (i = 0; i<N; i++)
    cin >>  a[i];
    ...
    return 0;
}
```

Решение.

Алгоритм:

1. Первый повтор: Найти количество элементов массива, больших 100 и делящихся на 5. Обозначим его, например, как k.
2. Второй повтор: заменить значения таких элементов массива на количество найденных элементов.
3. Вывести значения элементов обработанного массива на экран.

Код программы на языке Pascal:

```
{инициализация и ввод элементов исходного массива}
const  N = 30;
var a: array [1..N] of longint;
i, j, k: longint;
begin
   for i := 1 to N do
      readln(a[i]);
   {первый повтор: нахождение количество элементов массива, удовлетворяющих данному условию}
k := 0;
for i := 1 to N do
   if (a[i] > 100) and (a[i] mod 5 = 0) then
         k:=k+1;
   {второй повтор: замена элементов массива, удовлетворяющих данному условию, на найденное количество элементов}
for i := 1 to N do begin
   if (a[i] > 100) and (a[i] mod 5 = 0) then
      a[i] := k;
   {вывод элементов найденного массива на экран}
   writeln(a[i]);
end.
```

Примечание: в данной программе не используем объявленную целочисленную переменную j.

Дерево игры и обоснование выигрышных стратегий.

Теория игр (определение) - раздел прикладной математики (точнее —исследования операций). При этом под теорией игр понимают математический метод изучения оптимальных стратегий в играх.

Довольно часто в практической деятельности человеку приходится сталкиваться с задачами, в которых необходимо принимать решение в условиях, когда две или более стороны преследуют различные цели, а результаты любого действия каждой из сторон зависят от мероприятий партнера. Такие ситуации, возникающие, например, при

игре в шахматы, шашки или домино, относят к конфликтным ситуациям, потому что результат каждого хода игрока зависит от ответного хода противника. **Ситуация называется конфликтной, если в ней участвуют стороны, интересы которых полностью или частично противоположны.**

Так как, как правило, ответный ход заранее неизвестен, поэтому считается, что решение приходится принимать в условиях неопределенности. Целью игры всегда является выигрыш одного из участников.

Для рационального решения задач с конфликтными ситуациями существуют научно обоснованные методы, которые разработаны математической теорией конфликтных ситуаций, называемой теорией игр.

Игра в данной теории – это действительный или формальный конфликт, в котором имеется, по крайней мере, два участника (игрока), каждый из которых стремится к достижению собственных целей.

Допустимые действия каждого из игроков, направленные на достижение некоторой цели, называются правилами игры.

Игра называется парной, если в ней участвуют два игрока, и множественной, если число игроков больше двух.

В парной игре участвуют два игрока (А и В), интересы которых противоположны. Под игрой (или процессом игры) понимается ряд действий со стороны игроков А и В. **Количественная оценка результатов игры называется платежом.**

Парная игра бывает игрой с нулевой суммой, или антагонистической, если сумма платежей равна нулю, тогда выигрыш одного игрока равен проигрышу другого. В этом случае для полного задания игры достаточно указать одну из величин. Если, например, a – выигрыш одного из игроков, b - выигрыш другого, то для игры с нулевой суммой $b = -a$, поэтому достаточно рассматривать, например, a.

Выбор и осуществление одного из действий, предусмотренных правилами, называется ходом игрока.

Ходы могут быть личными и случайными. Личный ход – это сознательный выбор игроком одного из возможных действий (например, ход в шахматной игре). Случайный ход – это случайно выбранное действие (например, выбор карты из перетасованной колоды).

Стратегией игрока называется совокупность правил, определяющих выбор его действия при каждом личном ходе в зависимости от сложившейся ситуации.

Обычно в процессе игры при каждом личном ходе игрок делает выбор в зависимости от конкретной ситуации. Однако, в принципе, возможно, что решения приняты игроком заранее (в ответ на любую сложившуюся ситуацию). Это означает, что игрок выбрал определенную стратегию, которая может быть задана в виде списка правил или программы.

Игра называется конечной, если у каждого игрока есть конечное число стратегий, и бесконечной – в противном случае.

Стратегия игрока называется оптимальной (выигрышной), если она обеспечивает игроку максимальный выигрыш (или, что то же самое, минимальный проигрыш), при условии, что второй игрок придерживается своей стратегии.

Для того чтобы решить игру, или найти решение игры, необходимо для каждого из игроков выбрать оптимальную стратегию.

Таким образом, предмет теории игр составляют методы отыскания оптимальных стратегий игроков.

При выборе оптимальной стратегии предполагается, что оба игрока ведут себя разумно с точки зрения своих интересов. Важнейшее ограничение теории игр - единственность выигрыша как показателя эффективности.

В задании ЕГЭ по информатике рассматриваются дискретные детерминированные игры двух лиц с полной информацией. Это означает, что в ходе игры может возникать определенный набор позиций, при этом, все возможные позиции можно описать. Хотя допустимых позиций может быть и очень много, их – конечное число.

В предложенную заданием игру играют двое («игра двух лиц»), и они ходят по очереди. Каждый ход игрока состоит в том, что позиция заменяется новой позицией, правила игры определяют, какие ходы допускаются в каждой позиции. Если в некоторых позициях у игрока допустимых ходов может не быть, то тогда партия заканчивается (и позиции *называются заключительными*). Правила игры для каждой заключительной позиции определяют, кто победил в этой партии - игрок, который должен был сделать ход, его противник или партия закончилась вничью.

При описании дискретных игр двух лиц (их будем называть Первый и Второй; Первый всегда ходит первым) указываются следующие правила игры:

1) Конечное множество возможных позиций. Для удобства в описание позиции включают указание, кто будет ходить.

2) Для каждой позиции предлагают сделать список возможных ходов.
3) Для каждой заключительной позиции указывают, каков итог партии (победа того, чей ход, победа его противника, возможно, и ничья).

Партия – это последовательность ходов, которая приводит к заключительной позиции. Удобно представлять партию как последовательность позиций (от начальной позиции до ее заключительной позиции). По последовательности позиций, как правило, можно восстановить последовательность ходов. Дискретная игра двух лиц является детерминированной игрой с полной информацией, если при каждом ходе игрок имеет всю информацию о своей позиции и полностью свободен в выборе своего хода.

В информатике деревом называют обычно одну из наиболее широко распространённых структур данных, эмулирующая древовидную структуру в виде набора связанных узлов (в виде связного графа, не содержащим циклы).

Дерево игры определяет понятие возможного хода игрока как совокупность всех переходов между узлами двух соседних уровней, конкретный же ход игрока — один переход, выбранный из всех возможных.

Задание ЕГЭ по информатике, связанное с построением дерева игры и его анализом, в отличие от других заданий с развернутым ответом, не требует знаний в области программирования, однако учащийся должен представить в виде решения алгоритм действий для достижения поставленной в условии цели. Само решение и ответ могут быть оформлены произвольно, например, простым словесным описанием, но решение должно содержать математическую модель (алгоритм) предложенного в условии задачи процесса. В данной ситуации используют дерево как графический способ записи алгоритма игры. Если построить дерево игры, из которого видны все действия, возможные к исполнению по условию задачи, то с его помощью можно проследить сначала тактики, а затем и стратегию достижения поставленной цели.

В общем случае решение задачи производится по следующему плану:

1. Построение конечного дерева процесса (игры), маркирование (с помощью условных обозначений) узлов дерева, не соответствующих условию задачи (например, относящихся к проигрышной стратегии).

2. Анализ дерева "снизу вверх", включающий:

2а: первоначальную постановку гипотезы о достижении цели игры одним из ее участников;

2б: доказательство или опровержение гипотезы п. 2а путем выделения узлов дерева особыми условными обозначениями по определённым правилам;

2в: если гипотеза п. 2а опровергнута, то п. 2б выполняется для другого участника игры.

3. Запись ответа на основе выполненного в п. 2 анализа.

Следует отметить, что данное задание ЕГЭ относится к высокому уровню сложности и проверяет умение экзаменующихся на построение дерева игры по заданному алгоритму и обоснование выигрышной стратегии игры.

Практические задачи.

Задача 1.

Два игрока, Паша и Валя, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Паша. За один ход игрок может добавить в кучу один камень или увеличить количество камней в куче в два раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 20. Если при этом в куче оказалось не более 30 камней, то победителем считается игрок, сделавший последний ход. В противном случае победителем становится его противник. Например, если в куче было 17 камней и Паша удвоит количество камней в куче, то игра закончится, и победителем будет Валя. В начальный момент в куче было S камней, $1 \le S \le 19$.

Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Выполните следующие задания.

1. а) При каких значениях числа S Паша может выиграть в один ход? Укажите все такие значения и соответствующие ходы Паши.

   б) У кого из игроков есть выигрышная стратегия при S = 18, 17, 16? Опишите выигрышные стратегии для этих случаев.

   2. У кого из игроков есть выигрышная стратегия при S = 9, 8? Опишите соответствующие выигрышные стратегии.

   3. У кого из игроков есть выигрышная стратегия при S = 7? Постройте дерево всех партий, возможных при этой выигрышной

стратегии (в виде рисунка или таблицы). На ребрах дерева указывайте, кто делает ход; в узлах – количество камней в позиции.

Решение.

1. а) **Паша может выиграть, если S = 19 или S = 10, 11, 12, 13, 14, 15**. При S = 19 первым ходом нужно добавить в кучу один камень, при остальных указанных значениях S нужно удвоить количество камней.

б) При S = 16, 17 или 18 удваивать количество камней не имеет смысла, так как после такого хода выигрывает противник. Поэтому можно считать, что единственный возможный ход – это добавление в кучу одного камня.

При S = 18 после такого хода Паши в куче станет 19 камней. В этой позиции ходящий (т.е. Валя) выигрывает (смотрите пункт 1а): при S = 18 Паша (игрок, который должен ходить первым) проигрывает. **Выигрышная стратегия есть у Вали.**

При S = 17, после того как Паша своим первым ходом добавит один камень, в куче станет 18 камней. В этой позиции ходящий (т.е. Валя) проигрывает (смотрите выше): при S = 17 Паша (игрок, который должен ходить первым) выигрывает. **Выигрышная стратегия есть у Паши.**

При S = 16 **выигрышная стратегия есть у Вали**. Действительно, если Паша первым ходом удваивает количество камней, то в куче становится 32 камня, и игра сразу заканчивается выигрышем Вали. Если Паша добавляет один камень, то в куче становится 17 камней. Как мы уже знаем, в этой позиции игрок, который должен ходить (т.е. Валя), выигрывает.

Во всех случаях выигрыш достигается тем, что при своем ходе игрок, имеющий выигрышную стратегию, должен добавить в кучу один камень.

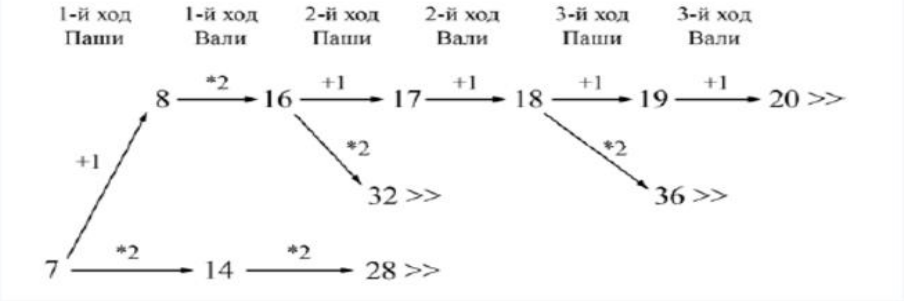
2. При S = 9 или 8 **выигрышная стратегия есть у Паши**. Она состоит в том, чтобы удвоить количество камней в куче и получить кучу, в которой будет соответственно 18 или 16 камней. В обоих случаях игрок, который будет делать ход (теперь это Валя), проигрывает (смотрите пункт 1б).

3. При S = 7 **выигрышная стратегия есть у Вали**. После первого хода Паши в куче может стать либо 8, либо 14 камней. В обеих этих позициях выигрывает игрок, который будет делать ход (теперь это Валя). Случай S = 8 рассмотрен в пункте 2, случай S = 14 рассмотрен в пункте 1а.

В таблице изображено дерево возможных партий при описанной стратегии Вали. Заключительные позиции (в них выигрывает Валя)

подчёркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

	Положения после очередных ходов					
И.п.	1-й ход Паши (все ходы)	1-й ход Вали (только ход по стратегии)	2-й ход Паши (все ходы)	2-й ход Вали (только ход по стратегии)	3-й ход Паши (все ходы)	3-й ход Вали (только ход по стратегии)
7	7+1 = **8**	8*2=**16**	16+1=**17**	17+1=**18**	18+1=**19**	**19+1=20**
					18*2=36	
			16*2=32			
	7*2=**14**	**14*2=28**				



Дерево всех партий, возможных при Валиной стратегии. Знаком >> обозначены позиции, в которых партия заканчивается.

Задача 2.

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один камень или увеличить количество камней в куче в два раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 22.

Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 22 или больше камней. В начальный момент в куче было S камней, $1 \le S \le 21$.

Пояснение. Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Выполните следующие задания. Во всех случаях обосновывайте свой ответ.

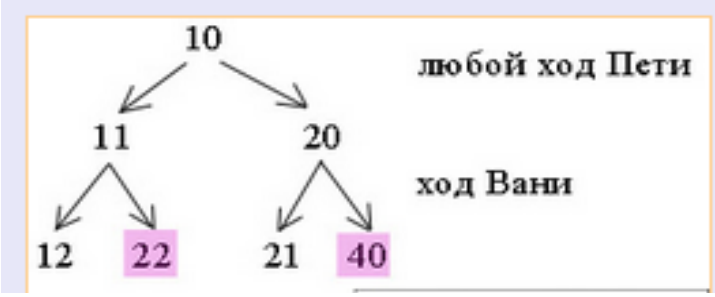
1. а) Укажите все такие значения числа S, при которых Петя может

выиграть в один ход. Обоснуйте, что найдены все нужные значения S, и укажите выигрывающий ход для каждого указанного значения S.
б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.
2. Укажите два таких значения S, при которых у Пети есть выигрышная стратегия, причем Петя не может выиграть за один ход, и Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня. Для каждого указанного значения S опишите выигрышную стратегию Пети.
3. Укажите значение S, при котором: у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, и у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.
Для указанного значения S опишите выигрышную стратегию Вани. Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы). На ребрах дерева указывайте, кто делает ход, в узлах – количество камней в куче.

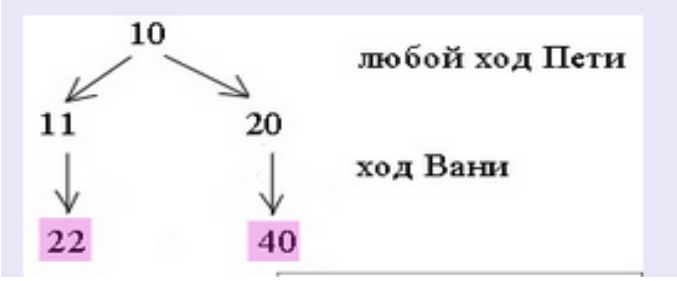
Решение.

1. а) Петя может выиграть за один ход.
   При $S \geq 22$ игра завершается. Петя может: добавить в кучу один камень (+1), увеличить количество камней в куче в два раза (*2).
   Рассмотрим каждый вариант:
   (+1):S= 22−1=21
(*2): S=22/2=11 =>$S \in [11;21]$
   11*2=22
   12*2=24 (>22)
   ...
   21*2=42 (>22)
   Получим следующие стратегии:
   при $S \in [11;20]$ надо удвоить количество камней, при S=21 надо добавить один камень или удвоить количество камней.

б) Ваня может выиграть за один ход при любом ходе Пети. Используем решение предыдущей задачи.
Ваня может выиграть при $S \geq 11$. Но Ваня ходит 2-м, а Петя 1-м. Нам нужно подобрать S.
S=11 можно получить следующим образом:
10+1
Построим дерево решений для S=10:

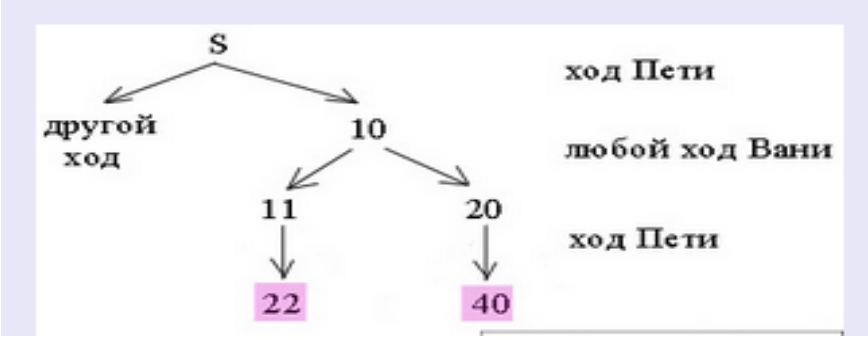


Получили, S=10. Стратегия Вани будет такой:



2.Петя не может выиграть 1-м ходом, он может выиграть за 2 хода при любом ходе Вани: используем решение задачи части 1(б). Дерево решений имеет вид:

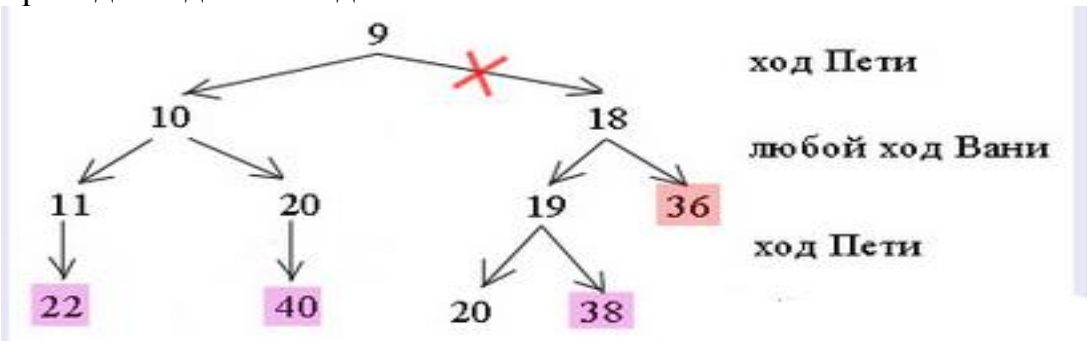
Нам нужно подобрать S. S =10 можно получить следующим образом:



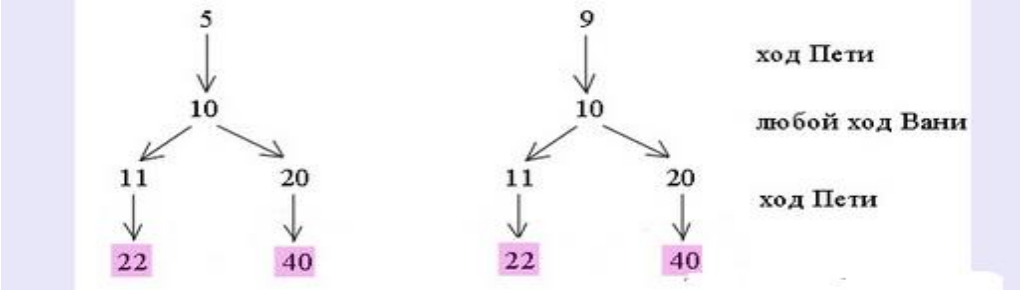
9+1 и 5*2.

Получаем следующие деревья решений при S=5 и S=9:

Красным крестом обозначена проигрышная ветка для Пети. При S=5 такая ветка приведет к тому, что никто не выиграет. При s=9 такая ветка приведет к тому, что выиграет Ваня первым ходом, получая 36. В решении проигрышные стратегии указывать не нужно. Здесь они приведены для наглядности.



Получим следующие выигрышные стратегии: S=5 и S=9

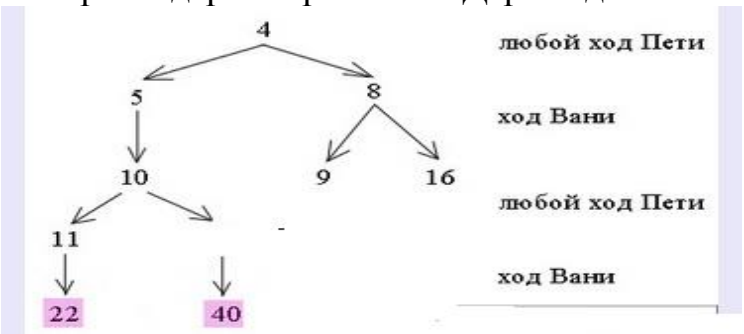


3. Ваня может выиграть за 1 или 2 хода при любых ходах Пети. Ваня не может гарантированно выиграть 1-м ходом. Используем решение части 2. Стратегии, приведенные выше, гарантируют выигрыш 2-м ходом. Нам нужно подобрать S.

S =5 можно получить следующим образом: 4+1

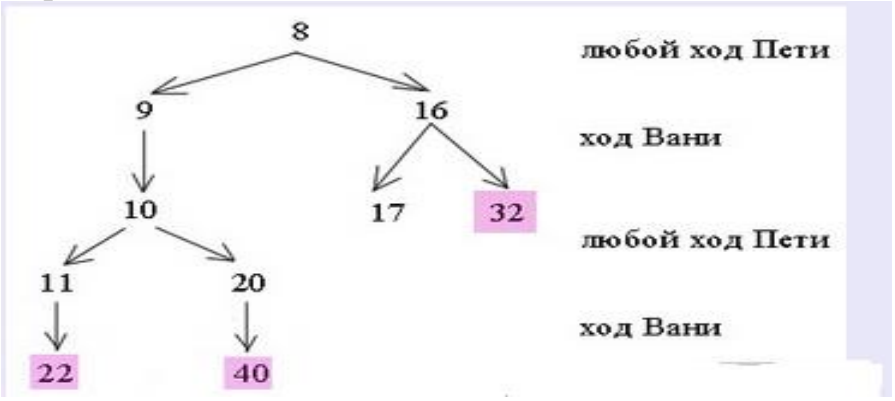
S =9 можно получить следующим образом: 8+1

Построим деревья решений. Дерево для S=4:

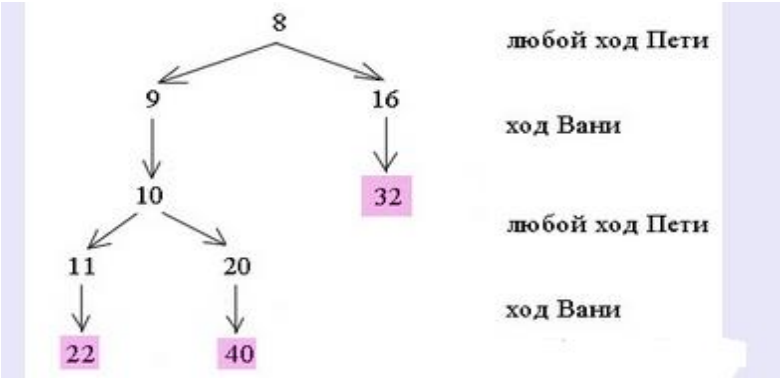


Из дерева видно, что Ваня не сможет выиграть 1-м ходом, т.к. 9<22 и 16<22 (правая ветка дерева, ветки 4→8→9 и 4→8→16), поэтому s=4 не подходит.

Дерево для S=8:



Из дерева видно, что Ваня сможет выиграть 1-м ходом, 32>22 (ветка 8→16→32). Получили, S=8. Стратегия будет такой:



Задача 3.

Два игрока, Петя и Ваня играют в следующую игру. На столе в кучке лежат фишки. На лицевой стороне каждой фишки написано двузначное натуральное число, обе цифры которого находятся в диапазоне от 1 до 4. Никакие две фишки не повторяются. Игра состоит в том, что игроки поочередно берут из кучки по одной фишке и выкладывают в цепочку на стол лицевой стороной вверх таким образом, что каждая новая фишка ставится **правее** предыдущей и ближайшие цифры соседних фишек совпадают. Верхняя часть всех выложенных фишек направлена в одну сторону, то есть переворачивать фишки нельзя. Например, из фишки, на которой написано 23, нельзя сделать фишку, на которой написано 32.

Первый ход делает Петя, выкладывая на стол любую фишку из кучки. Игра заканчивается, когда в кучке нет ни одной фишки, которую можно добавить в цепочку. Тот, кто добавил в цепочку последнюю фишку, **выигрывает**, а его противник проигрывает.

Будем называть **партией** любую допустимую правилами последовательность ходов игроков, приводящую к завершению игры. Будем говорить, что игрок имеет **выигрышную стратегию**, если он может

выиграть при любых ходах противника. Описать стратегию игрока – значит указать, какую фишку он должен выставить в любой ситуации, которая ему может встретиться при различной игре противника.

Пример партии.

Пусть на столе в кучке лежат фишки: 11, 12, 13, 21, 22, 23

Пусть первый ход Пети 12. Ваня может поставить 21, 22 или 23. Предположим, он ставит 21. Получим цепочку 12-21. Петя может поставить 11 или 13. Предположим, он ставит 11. Получим цепочку 12-21-11. Ваня может поставить только фишку со значением 13. Получим цепочку 12-21-11-13. Перед Петей в кучке остались только

фишки 22 и 23, то есть нет фишек, которые он мог бы добавить в цепочку. Таким образом, партия закончена, Ваня выиграл.

Выполните следующие три задания при исходном наборе фишек в кучке {12, 14, 21, 22, 24, 41, 42, 44}.

Задание 1.

а) Приведите пример самой короткой партии, возможной при данном наборе фишек. Если таких партий несколько, достаточно привести одну.

б) Пусть Петя первым ходом пошел 42. У кого из игроков есть выигрышная стратегия в этой ситуации? Укажите первый ход, который должен сделать выигрывающий игрок, играющий по этой стратегии. Приведите пример одной из партий, возможных при реализации выигрывающим игроком этой стратегии.

Задание 2. Пусть Петя первым ходом пошел 44. У кого из игроков есть выигрышная стратегия, позволяющая в этой ситуации выиграть своим четвертым ходом? Постройте в виде рисунка или таблицы дерево всех партий, возможных при реализации выигрывающим игроком этой стратегии.

На ребрах дерева указывайте ход, в узлах – цепочку фишек, получившуюся после этого хода.

Задание 3. Укажите хотя бы один способ убрать 2 фишки из исходного набора так, чтобы всегда выигрывал **не** тот игрок, который имеет выигрышную стратегию в задании 2. Приведите пример партии для набора из 6 оставшихся фишек.

Решение.

Задание 1.

а) Все кратчайшие партии: 12-21-14-41, 14-41-12-21.

б) Выигрышная стратегия есть у Вани.

Своим первым ходом он ставит фишку 22. И далее исход игры не зависит от выбора игроков.

Возможная партия:

42-22-21-12-24-41-14-44

Существенный элемент выигрышной стратегии Вани – поставить после фишки 42 фишку 22, а не 21 или 24.

Существует 8 вариантов возможных партий:

42-22-21-12-24-41-14-44

42-22-21-12-24-44-41-14

42-22-21-14-41-12-24-44

42-22-21-14-44-41-12-24

42-22-24-41-12-21-14-44

42-22-24-41-14-44

42-22-24-44-41-12-21-14
42-22-24-44-41-14

Задание 2.

Выигрышная стратегия, позволяющая выиграть своим четвертым ходом, есть у Пети. Дерево всех партий для этой стратегии показано в таблице и на рисунке 25.

Таблица. Дерево всех партий при выигрышной стратегии Пети, позволяющей ему выиграть своим четвертым ходом. Над пунктирной чертой указаны ходы игроков. Завершающие игру ходы подчеркнуты. Под пунктирной чертой в скобках приведены цепочки фишек после очередного хода.

	Начальная позиция [пусто]		
1-й ход Пети	44		
	44		
1-й ход Вани	41	42	
	44-41	*44-42*	
2-й ход Пети	14	21	
	44-41-14	*44-42-21*	
2-й ход Вани	42	12	14
	44-41-14-42	*44-42-21-12*	*44-42-21-14*
3-й ход Пети	21	24	41
	44-41-14-42-21	*44-42-21-12-24*	*44-42-21-14-41*
3-й ход Вани	12	41	12
	44-41-14-42-21-12	*44-42-21-12-24-41*	*44-42-21-14-41-12*
4-й ход Пети	**24**	**14**	**24**
	44-41-14-42-21-12-24	*44-42-21-12-24-41-14*	*44-42-21-14-41-12-24*

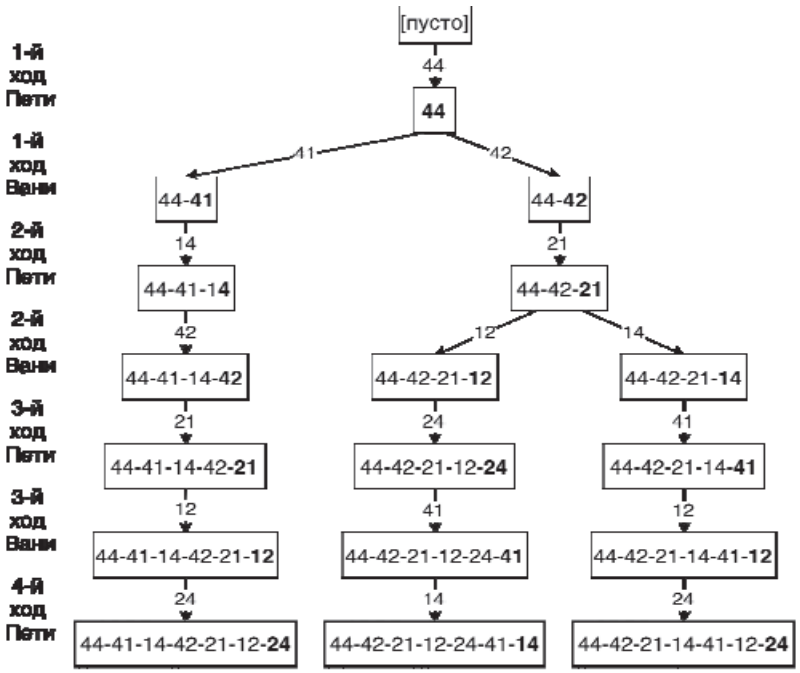


Рисунок 25 - Дерево всех партий при выигрышной стратегии Пети, позволяющей ему выиграть своим четвертым ходом. В заключительных цепочках подчеркнут последний ход.

Заключение

Представленные во второй части пособия материалы содержат решения, связанные с такими разделами информатики, как, например, комбинаторика, теория и элементы алгоритмов, теория программирования, технологии компьютерных сетей, поисковые информационные запросы. Для того, чтобы успешно освоить задания части 2 с развернутыми ответами, на занятиях даются базовые знания по теории массивов, теории игр, понятий эффективного программирования. Как правило, особую трудность у слушателей вызывают задания по навыкам программирования в случае, если слушатель не был предварительно знаком с каким-нибудь базовым языком программирования. Поэтому в процессе их подготовки используется, как наиболее применяемая в школах на уроках информатики, компьютерная среда Pascal ABC. Также в данной части пособии даются соответствующие материалы по соответствующим разделам прикладной математики, связанные с навыками решений задач ЕГЭ по информатике.

Приложение 2

Решения итоговой контрольной работы.
Таблица правильных ответов

Варианты/№ задач	Вариант 1	Вариант 2	Вариант 3
Задача 1	4	64	8
Задача 2	251	501	376
Задача 3	5	3	4
Задача 4	22	18	192
Задача 5	15	20	10
Задача 6	2	2	3
Задача 7	wzyx	zwyx	zwxy
Задача 8	28	54	11
Задача 9	135	67	33
Задача 10	24	15	21
Задача 11	34	54	32