Департамент образования Вологодской области
Вологодский институт развития образования

Серия «На пути к эффективной школе»

РАБОТА НАД ЗАДАЧАМИ ПО ТЕМАМ «ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ» И «ПРОГРАММИРОВАНИЕ» ПРИ ПОДГОТОВКЕ ОБУЧАЮЩИХСЯ К ГИА ПО ИНФОРМАТИКЕ

Учебное пособие для подготовки к итоговой государственной аттестации выпускников основной и старшей школы

Вологда
2020

УДК 004(075)
ББК 32.97я72
Р13

Печатается по решению рабочей группы
по реализации проекта «Поддержка школ с низкими
результатами и школ, функционирующих
в неблагоприятных социальных условиях».
Протокол заседания № 2 от 22 апреля 2020 года

*Издается в рамках реализации проекта «Поддержка школ
с низкими результатами и школ, функционирующих
в неблагоприятных социальных условиях»*

Авторы-составители:
Ганичева Е.М., к.п.н., доцент кафедры математики и информатики
Института математики, естественнонаучных и компьютерных наук
ФГБОУ ВО «Вологодский государственный университет»,
Голубев О.Б., к.п.н., доцент, директор Института математики, естественнонаучных
и компьютерных наук ФГБОУ ВО «Вологодский государственный университет»,
Никифоров О.Ю., старший преподаватель, начальник управления информатизации
ФГБОУ ВО «Вологодский государственный университет»

Рецензенты:
Горохова Ю.А., к.п.н., доцент кафедры математики и информатики
Института математики, естественнонаучных и компьютерных наук
ФГБОУ ВО «Вологодский государственный университет»;
Пустохина И.Н., учитель информатики МАОУ «Центр образования № 42»
г. Вологды

Р13 **Работа** над задачами по темам «Элементы теории алгоритмов» и «Программирование» при подготовке обучающихся к ГИА по информатике : учебное пособие для подготовки к итоговой государственной аттестации выпускников основной и старшей школы / Департамент образования Вологодской области, Вологодский институт развития образования ; [авторы-составители: Ганичева Е.М., Голубев О.Б., Никифоров О.Ю.]. – Вологда: ВИРО, 2020. – 80 с. : ил., табл.

ISBN 978-5-87590-513-1

Учебное пособие содержит материалы для подготовки к итоговой государственной аттестации по информатике за курс основной и старшей школы по темам «Элементы теории алгоритмов» и «Программирование».

Учебное пособие предназначено для учителей информатики, обучающихся основной и старшей школы, студентов, обучающихся по направлению «Педагогическое образование», профиль «Математическое образование и информатика».

УДК 004(075)
ББК 32.97я72

ISBN 978-5-87590-513-1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информатика относится к предметам, необязательным для сдачи единого государственного экзамена, однако перечень вузов, требующих наличия свидетельства об успешной сдаче ЕГЭ по информатике, постоянно растет. Возрастает и сложность заданий, предлагаемых на ЕГЭ по информатике. В 2021 году экзаменационная работа будет состоять из 27 заданий с кратким ответом, выполненных с помощью компьютера. Это создает дополнительные сложности, связанные с тем, что для выполнения заданий выпускнику будет необходимо владеть различным программным обеспечением, уметь составлять программы на одном из языков программирования. Существенным будет и аспект, связанный с неоднородностью программного обеспечения в пунктах проведения экзамена.

Подготовка к ОГЭ и ЕГЭ является актуальной задачей как для самих обучающихся основной и старшей школы, так и для учителей информатики.

Наилучшей стратегией такой подготовки является системное и целенаправленное формирование основных информационных компетенций школьников, отработка решения разнообразных заданий и выработка навыков работы с основными средствами ИКТ по всем без исключения изучаемым темам курса. На этапе подготовки к экзамену работа с обучающимися должна носить дифференцированный характер. Учителю следует ставить перед каждым учащимся ту цель, которую он может реализовать в соответствии с уровнем его подготовки, при этом возможно опираться на самооценку и устремления каждого учащегося. Уровень сложности заданий в данном пособии позволяет освоить базовые умения обучающимся с недостаточной подготовкой по информатике.

Тренировочные материалы предназначены для подготовки к государственному выпускному экзамену (ОГЭ, ЕГЭ) в письменной форме и включают задания из открытого банка заданий ФИПИ.

ТЕМА «ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ»

ЗАДАНИЕ № 5.
ВЫПОЛНЕНИЕ И АНАЛИЗ ПРОСТЫХ АЛГОРИТМОВ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ, задание № 5 проверяет умение формально исполнить алгоритм, записанный на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд.

Коды проверяемых элементов содержания (по кодификатору):

1.6.3. Построение алгоритмов и практические вычисления.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к базовому уровню сложности, не требует для решения специального программного обеспечения и рассчитано на 4 минуты.

В демонстрационном варианте 2021 года задание № 5 имеет следующий вид:

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

1. Строится двоичная запись числа N.

2. К этой записи дописываются справа еще 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 ее 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 такое наименьшее число N, для которого результат работы данного алгоритма больше числа 77. В ответе это число запишите в десятичной системе счисления.

Что нужно знать для выполнения задания:

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

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

• бит четности – это дополнительный контрольный бит, который добавляется к двоичному коду так, чтобы количество единиц в полученном двоичном коде стало четным; если в исходном коде уже было четное количество единиц, дописывается 0, если нечетное – дописывается 1;

• при добавлении к двоичной записи числа нуля справа число увеличивается в 2 раза;

• чтобы отбросить последнюю цифру в двоичной записи, нужно разделить число на 2 нацело (остаток отбрасывается).

Базовые задачи

1. Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам:

Вычисляются три числа – сумма старших разрядов заданных трехзначных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.

Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Пример. Исходные трехзначные числа: 835, 196. Поразрядные суммы: 9, 12, 11. Результат: 12119.

Определите, какое из следующих чисел может быть результатом работы автомата.

1) 151303 2) 161410 3) 191615 4) 121613

Решение:

В данной задаче речь идет о сложении цифр чисел, которые неизвестны. Обозначим эти числа как X и Y и запишем в виде:

$X = x_1x_2x_3$; $Y = y_1y_2y_3$

Суммы цифр, которые вычисляет автомат, есть:

$S_1 = x_1 + y_1$, $S_2 = x_2 + y_2$ и $S_3 = x_3 + y_3$

Оценим возможные значения этих сумм. Старшая цифра в двузначном числе может иметь значение от 1 до 9 (если она равна нулю, то число уже будет трехзначным!), а средняя и младшая – значение от 0 до 9. Тогда сумма $S_1 = x_1 + y_1$ может иметь значение от 2 до 18 (исходных чисел – два), а суммы $S_2 = x_2 + y_2$ и $S_3 = x_3 + y_3$ – значение от 0 до 18.

Проанализируем предлагаемые варианты ответов.

1. Число 151303. Оно может быть составлено из сумм 15, 13 и 03, но третья сумма (03) имеет незначащий нуль, тогда как в условии не указано, что после сложения цифр исходных чисел автомат должен дополнять полученную сумму нулем до двузначного числа. Поэтому данный вариант ответа не может быть правильным.

2. Число 161410. Его можно представить как сочетание сумм 16. 14 и 10. Это вполне удовлетворяет условиям задачи (все три суммы входят в допустимый интервал и записаны по убыванию).

3. Число 191615. Может быть представлено как сочетание сумм 19, 16 и 15. Однако значение суммы 19 выходит за пределы допустимого диапазона (должно быть не более 18), поэтому данный вариант числа нам не подходит.

4. Число 121613. Может быть представлено как сочетание сумм 12, 16 и 13. Эти значения соответствуют допустимому интервалу, но записаны не по убыванию. Значит, данный вариант ответа тоже ложный.

Следовательно, правильным является ответ 161410.

Ответ: число 161410.

2. Устройство считывает четырехзначное восьмеричное число и строит по нему новое число по следующему алгоритму:

a. Вычисляется сумма первой и второй цифр.

b. Вычисляется сумма третьей и четвертой цифр.

c. Эти суммы записываются друг за другом без разделителей по убыванию.

d. Пример. Задано число 7145. Суммы:

e. 7+1=10; 4+5=11. Результат: 1110.

f. Какое из чисел может быть результатом работы такого устройства:

1) 119 2) 1213 3) 1411 4) 1715

Ответ: 1411.

3. Некоторый алгоритм из одной цепочки символов получает новую цепочку следующим образом. Если цепочка символов начинается с буквы, то в начало и в конец цепочки добавляется 1. В противном

случае первый символ цепочки переставляется в конец цепочки символов. Затем в полученной цепочке символов каждая цифра заменяется следующей (1 заменяется на 2, 2 – на 3, и т.д., а 9 заменяется на 0).

Получившаяся таким образом цепочка является результатом работы алгоритма. Например, если исходной цепочкой была цепочка А2, то результатом работы алгоритма будет цепочка 2А32, а если исходной цепочкой была 3Б, то результатом работы алгоритма будет цепочка Б4.

Дана цепочка символов В54Д. Какая цепочка символов получится, если к данной цепочке применить описанный алгоритм дважды (т.е. применить алгоритм к данной цепочке, а затем к результату вновь применить алгоритм)?

Решение:

Внимательно читаем условие задачи: в первой части дано описание алгоритма, во второй – пример работы алгоритма, в третьей – цепочка символов, к которой нужно применить алгоритм дважды.

Исходная цепочка начинается с символа, значит, добавляем 1 в начало и конец. Получаем цепочку 1В54Д1, меняем все цифры на следующие за ними. Получаем цепочку 2В65Д2. Эта цепочка и будет являться результатом работы алгоритма. В задаче требуется применить алгоритм дважды, поэтому применяем тот же алгоритм уже к полученной цепочке. Теперь цепочка начинается с цифры, а в этом случае первый символ переставляется в конец: 2В65Д2 → В65Д22 → В76Д33.

Ответ: В76Д33.

4 (демо-2021). *На вход алгоритма подается натуральное число N. Алгоритм строит по нему новое число R следующим образом.*

1. Строится двоичная запись числа N.

2. К этой записи дописываются справа еще 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 ее 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 такое наименьшее число N, для которого результат работы данного алгоритма больше числа 77. В ответе это число запишите в десятичной системе счисления.

Решение:

1) На шаге 2а добавляется бит четности так, чтобы количество единиц в двоичной записи нового числа стало четным.

2) На шаге 2б всегда дописывается 0, поскольку после шага 2а число единиц уже четно.

3) Если двоичная запись числа оканчивается на 0, то число четно, поэтому имеет смысл искать число – результат R среди четных чисел.

4) Возьмем первое четное число, большее, чем 77, и переведем его в двоичную систему: $78 = 1001110_2$.

5) Видим, что все условия выполняются: в двоичной записи числа 78 четное число единиц (четыре), поэтому оно могло быть получено в результате работы приведенного алгоритма.

6) Во время работы алгоритма к двоичной записи приписали сзади две цифры, их нужно отбросить, получается $10011_2 = 19$.

Ответ: 19.

Задачи для самостоятельного решения

1. (ОГЭ) Некоторый алгоритм из одной цепочки символов получает новую цепочку следующим образом. Сначала вычисляется длина исходной цепочки символов; если она четна, то в начало цепочки символов добавляется последний символ, а если нечетна, то в конец цепочки добавляется средний символ. В полученной цепочке символов каждая буква заменяется буквой, следующей за ней в русском алфавите (*А* – на *Б*, *Б* – на *В* и т.д., а *Я* – на *А*). Получившаяся таким образом цепочка является результатом работы алгоритма.

Например, если исходной цепочкой была цепочка *АБС*, то результатом работы алгоритма будет цепочка *БВТВ*, а если исходной цепочкой была *РИ*, то результатом работы алгоритма будет цепочка *ЙСЙ*.

Ответ: *ЕЦНВЕН*.

2. (ОГЭ) Цепочка из четырех бусин, помеченных латинскими буквами, формируется по следующему правилу:

a. На третьем месте цепочки стоит одна из бусин *A*, *E*.

b. На втором месте – одна из бусин *H*, *E*, *D*, которой нет на третьем месте.

c. В начале стоит одна из бусин *H*, *A*, *C*, которой нет на втором месте.

d. В конце – одна из бусин *H*, *E*, *D*, не стоящая на первом месте.

Определите, сколько из перечисленных цепочек созданы по этому правилу.

HDEE HHAE HEAE AHAH AEAD AEED CAEH EHAD CDEA.

В ответе запишите только количество цепочек.

Ответ: 4.

3. (ОГЭ) Автомат получает на вход пятизначное десятичное число. По полученному числу строится новое десятичное число по следующим правилам:

a. Вычисляются два числа – сумма первых трех цифр и сумма последних трех цифр.

b. Полученные два числа записываются друг за другом в порядке неубывания (без разделителей).

Пример. Исходное число: 15177. Поразрядные суммы: 7,15. Результат: 715.

Определите, сколько из приведенных ниже чисел может получиться в результате работы автомата.

2727 277 2715 2730 3027 1527 727 512

В ответе запишите только количество чисел.

Ответ: 3.

4. (ЕГЭ) Автомат получает на вход четырехзначное число. По этому числу строится новое число по следующим правилам:

a. Складываются первая и вторая, а также третья и четвертая цифры исходного числа.

b. Полученные два числа записываются друг за другом в порядке возрастания (без разделителей).

Пример. Исходное число: 6531. Суммы: 6 + 5 = 11, 3 + 1 = 4. Результат: 411.

Укажите наибольшее число, в результате обработки которого автомат выдает число 1113.

Ответ: 9492.

5. (ЕГЭ) Автомат получает на вход трехзначное число. По этому числу формируется новое число по следующим правилам:

a. Складываются первая и вторая, а затем – вторая и третья цифры исходного числа.

b. Полученные два числа записываются друг за другом подряд в порядке возрастания.

Пример. Исходное число: 176. Полученные числа: 1 + 7 = 8, 7 + 6 = 13. Результат: 813.

Найдите наибольшее исходное число, для которого автомат выдаст результат 815.

Ответ: 962.

6. (ЕГЭ) Процессор получает пятизначное число и по нему вычисляет новое, используя следующие правила:

a. Отдельно суммируются первая, третья и пятая цифры, а также вторая и четвертая цифры.

b. Два полученных числа записываются подряд по неубыванию.

Требуется определить наименьшее возможное число, при обработке которого будет получен результат 621.

Ответ: 30969.

ЗАДАНИЕ № 12.
ВЫПОЛНЕНИЕ АЛГОРИТМОВ ДЛЯ ИСПОЛНИТЕЛЯ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание №12 проверяет умение анализировать результат исполнения алгоритма.

Коды проверяемых элементов содержания (по кодификатору):

1.6.3. Вычислимость. Эквивалентность алгоритмических моделей.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к повышенному уровню сложности, не требует для решения специального программного обеспечения и рассчитано на 4 минуты.

В демонстрационном варианте 2021 года задание № 12 имеет следующий вид:

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.

Какая строка получится в результате применения приведенной ниже программы к строке, состоящей из 70 идущих подряд цифр 8? В ответе запишите полученную строку.

```
НАЧАЛО
ПОКА нашлось (2222) ИЛИ нашлось (8888)
        ЕСЛИ нашлось (2222)
           ТО заменить (2222, 88)
              ИНАЧЕ заменить (8888, 22)
        КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Что нужно знать для выполнения задания:

• правила выполнения линейных, разветвляющихся и циклических алгоритмов;

• основные операции с символьными строками (определение длины, выделение подстроки, удаление и вставка символов, «сцепка» двух строк в одну);

• *исполнитель* – это человек, группа людей, животное, машина или другой объект, который может понимать и выполнять некоторые команды;

• в школьном алгоритмическом языке **нц** обозначает «начало цикла», а **кц** – «конец цикла»; все команды между **нц** и **кц** – это тело цикла, они выполняются несколько раз;

• запись **нц для i от 1 до n** обозначает начало цикла, в котором переменная **i** (она называется переменной цикла) принимает последовательно все значения от 1 до **n** с шагом 1.

Исполнитель Робот – перемещается по клеткам лабиринта.

Типичная система команд:

Перемещение: вверх, вниз, влево, вправо;

Проверка наличия препятствия (стенки): сверху свободно, слева свободно, снизу свободно, справа свободно – играют роль условия;

Команда ветвления:

ЕСЛИ <условие>

ТО команда 1

ИНАЧЕ команда 2

КОНЕЦ ЕСЛИ

– выполняется команда 1 (если условие истинно) или команда 2 (если условие ложно), где условие – это команда проверки наличия препятствия;

До начала движения робота проверяется наличие препятствия с указанной стороны. Если оно есть, то Робот остается на месте. Если его нет, то начинается движение Робота в заданном направлении, даже если по направлению его движения имеется стенка (в последнем случае Робот разбивается)

Команда цикла:

ПОКА <условие> команда

Или ПОКА <условие> последовательность команд

КОНЕЦ ПОКА

– выполняется, пока условие истинно (условие – это команда проверки наличия препятствия).

– Если в цикле ПОКА размещено несколько операторов ЕСЛИ, то:

– проверяется условие выполнения цикла; если оно ложно, то цикл не выполняется вообще;

– выполняется первый оператор ЕСЛИ (согласно заданному в нем условию);

– выполняется второй оператор ЕСЛИ (согласно заданному в нем условию);

– выполняется последний оператор ЕСЛИ (согласно заданному в нем условию);

– только после завершения выполнения всех операторов ЕСЛИ, записанных в теле цикла, выполняется новая проверка условия в цикле ПОКА для выяснения, должен ли цикл быть повторен.

Исполнитель Чертежник – перемещается по координатной плоскости.

Его основная команда – **сместиться на (*a*, *b*)**, где *a*, *b* – некоторые целые числа. Она перемещает Чертежника из исходной точки с координатами (*x*, *y*) в точку с координатами (*x* + *a*, *y* + *b*).

Цикл

ПОВТОРИ число РАЗ

Последовательность команд

КОНЕЦ ПОВТОРИ

– означает, что заданная последовательность команд выполняется указанное количество раз (значение должно быть натуральным числом).

Исполнитель Редактор – обрабатывает текстовые строки.

Он получает на вход некоторую текстовую строку и преобразует ее по заданным правилам.

Исполнитель Редактор может выполнять две команды (в них обозначения *v* и *w* обозначают некоторые последовательности – цепочки символов);

Заменить (*v*, *w*) – заменяет в строке первое встреченное при просмотре слева направо вхождение цепочки символов *v* на цепочку *w*; если же цепочка символов *v* в строке не найдена, выполнение данной команды не меняет строку;

Нашлось (*v*) – проверяет, встречается ли цепочка символов *v* в обрабатываемой текстовой строке, и возвращает логическое значение «истина» или «ложь»; сама текстовая строка при этом не изменяется.

Базовые задачи

1. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости: вверх, вниз, влево, вправо.

Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится **РОБОТ: сверху свободно, снизу свободно, слева свободно, справа свободно**

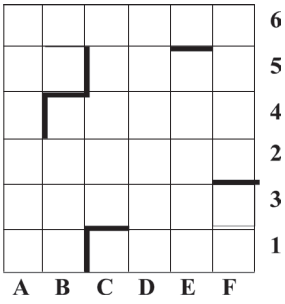
Цикл ПОКА <условие> команда

Выполняется, пока условие истинно, иначе происходит переход на следующую строку.

Сколько клеток лабиринта соответствуют требованию, что, выполнив предложенную программу, РОБОТ остановится в той же клетке, с которой он начал движение?

НАЧАЛО
ПОКА <сверху свободно> вправо
ПОКА <справа свободно> вниз
ПОКА <снизу свободно> влево

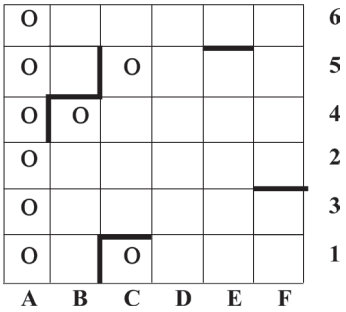
ПОКА <слева свободно> вверх
КОНЕЦ

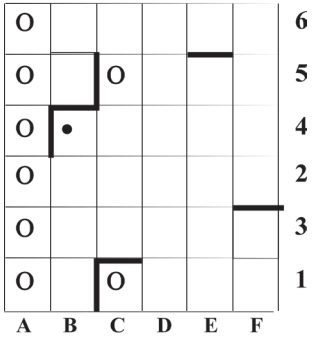


Решение:

Проще всего решить эту задачу, поочередно перебирая каждую клетку лабиринта и «проводя» робота из нее по маршруту согласно заданной программе. Однако такой способ требует слишком много времени.

Прежде всего, обратим внимание на то, что направления движения и условия остановки Робота в каждой команде ПОКА различны. Робот либо может остановиться, когда стенка (как условие остановки движения) появится в очередной клетке, в которую он перейдет, с соответствующего бока, либо может разрушиться, если первой встретится стенка, перекрывающая ему путь (тогда с соответствующего бока стенки, которая бы его остановила, нет). В условии таких задач обычно особо оговаривается факт разрушения Робота, натолкнувшегося на стенку.





Далее нужно обратить внимание на тот факт, что Робот, с какой бы клеточки он ни начинал движение, однозначно останавливается (согласно программе) в клетке, где с требуемой стороны имеется стенка. Поэтому проще всего сначала определить клеточки, в которых Робот должен, согласно его программе, завершать движение. Таких клеточек будет не так много, а дальше можно проверять каждую из них на соответствие условию: мог ли Робот прийти в эту клетку, начав движение с нее же? В данном случае Робот завершает программу, если обнаруживает стенку слева.

На схеме лабиринта отмечаются все такие точки – их оказывается всего 9.

Теперь они поочередно перебираются и проверяются, сможет ли Робот, начав путь из какой-либо точки, вернуться в нее. При этом следует учесть, что, если движение в каком-то направлении невозможно, так как оно изначально «запрещено» стенкой, то соответствующая строка программы пропускается, и необходимо сразу перейти к следующей, а если Робот при движении сталкивается со стенкой, то выполнение программы прерывается, так как Робот разрушился. Итого, есть только одна клеточка, удовлетворяющая условию.

Ответ: 1.

Примечание. В подобных задачах в командах ПОКА проверяется наличие соответствующей стенки по отношению к текущей ячейке, в которой находится Робот, а не по отношению к самому Роботу с учетом направления его движения.

2. Исполнитель Чертежник перемещается по координатной плоскости. Основная команда Чертежника – сместиться на (a, b), где a, b – целые числа. Она перемещает Чертежника из точки с координатами (x, y) в точку с координатами ($x + a$, $y + b$). Например, из точки с координатами (3, 5) команда **сместиться на** (-2, 1) переместит Чертежника в точку (1, 6).

Чертежник выполнял алгоритм, в котором количество повторений и оба смещения в первой из повторяемых команд неизвестны:

НАЧАЛО
сместиться на (-2, 3)
ПОВТОРИ … РАЗ
сместиться на (…, …)
сместиться на (-2, -1)
КОНЕЦ ПОВТОРИ
сместиться на (-19, -18)
КОНЕЦ

При этом, выполнив алгоритм, Чертежник вернулся в исходную точку. Какое наибольшее количество повторений могло быть указано в конструкции «ПОВТОРИ … РАЗ»?

Решение:

Обозначим неизвестные нам значения переменными: k – количество повторений, x и y – неизвестные смещения в первой повторяемой команде. При этом каждое уравнение приравнивается нулю, так как Чертежник после всех перемещений вернулся в исходную точку.

Перемещение Чертежника по координате x: $-2 + k \cdot (x - 2) - 19 = 0$

Перемещение Чертежника по координате y: $3 + k \cdot (y - 1) - 18 = 0$

Объединяем оба уравнения в систему:

$$\begin{cases} -2 + k \cdot (x - 2) - 19 = 0 \\ 3 + k \cdot (y - 1) - 18 = 0 \end{cases} \Rightarrow \begin{cases} k \cdot (x - 2) = 21 \\ k \cdot (y - 1) = 15 \end{cases}$$

Разложим числа справа от знаков равенства на простые сомножители: $21 = 3 \cdot 7$; $15 = 3 \cdot 5$. Заметим, что в обоих получившихся произведениях повторяется один и тот же сомножитель – 3. Значит, это и есть k.

Ответ: 3.

3. Исполнитель Редактор получает на вход строку цифр и преобразует ее. Редактор может выполнять две команды: **заменить (*v*, *w*) и нашлось (*v*)**.

Дана программа для исполнителя Редактор:

```
НАЧАЛО
ПОКА нашлось (111) ИЛИ нашлось (999)
        ЕСЛИ нашлось (111)
            ТО заменить (111, 9)
            ИНАЧЕ заменить  (999, 1)
        КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Какая строка получится из строки, состоящей из 74 идущих подряд цифр «9»?

Решение:

Начинаем анализировать алгоритм Редактора.

Вначале имеется строка из 74 цифр «9». Поэтому условие ПОКА истинно (три подряд девятки в этой строке есть). Условие ЕСЛИ – ложно (единиц в строке нет). Поэтому выполняется ветвь программы ИНАЧЕ и производится замена первых трех девяток на одну единицу:

$$\underbrace{99999999999\ldots\ldots..9999}_{\text{74 цифры «9»}} \quad \rightarrow \quad \underbrace{19999999999\ldots\ldots9999}_{\text{71 цифра «9»}}$$

На втором проходе цикла условие ПОКА все еще истинно (после полученной единицы имеется три девятки подряд). Условие ЕСЛИ попрежнему ложно (трех единиц подряд в строке пока нет). Поэтому вновь выполняется ветвь программы ИНАЧЕ и производится замена очередных трех девяток на одну единицу:

$\underbrace{99999999999\ldots\ldots..9999}_{\text{71 цифра «9»}} \rightarrow \underbrace{19999999999\ldots\ldots.9999}_{\text{68 цифр «9»}}$

На третьем проходе цикла условие ПОКА по-прежнему истинно (после двух единиц есть девятки, идущие подряд). Условие же ЕСЛИ все еще ложно (трех единиц подряд в строке нет). Поэтому выполняется ветвь программы ИНАЧЕ и снова производится замена очередных трех девяток на одну единицу:

$\underbrace{99999999999\ldots\ldots..9999}_{\text{68 цифр «9»}} \rightarrow \underbrace{19999999999\ldots\ldots.9999}_{\text{65 цифр «9»}}$

На четвертом проходе цикла условие ПОКА тоже истинно (хотя бы по первым трем единицам, образовавшимся в начале строки). А вот условие ЕСЛИ теперь истинно – у нас в начале строки имеется три подряд единицы. Поэтому теперь выполняется уже ветвь программы ТО и производится замена этих трех единиц снова на девятку – но только на одну:

$\underbrace{99999999999\ldots\ldots..9999}_{\text{65 цифр «9»}} \rightarrow \underbrace{19999999999\ldots\ldots.9999}_{\text{Новая цифра «9» + 65 цифр «9»}}$

Делаем вывод: в работе программы для исполнителя Редактор можно выделить цикл – каждые четыре шага цикла ПОКА приводят к тому, что исходная строка из девяток просто уменьшает свою длину на восемь символов. Поэтому дальнейшую работу программы можно подробно не рассматривать, а просто «отсекать» от строки по восемь девяток, пока не останется фрагмент строки длиной восемь цифр или меньше. Поэтому начинаем вычитать из количества девяток число восемь до тех пор, пока их не останется восемь или меньше: $66 \rightarrow 58 \rightarrow 50 \rightarrow 42 \rightarrow 34 \rightarrow 26 \rightarrow 18 \rightarrow 10 \rightarrow 2$.

Снова проанализируем, что будет происходить с фрагментом. В нашем случае осталась строка из двух девяток, для которой условие ПОКА ложно, и выполнение программы прекращается.

Ответ: 99.

Примечание. Если получившаяся строка имеет длину три или больше, то потребуется подробно рассмотреть, какие преобразования происходят в этом фрагменте строки.

Например, если бы осталось пять девяток:

Первые три девятки заменяются на одну единицу:
99999 → 199

Теперь в полученной строке больше нет ни трех подряд единиц, ни трех подряд девяток. Следовательно, условие ПОКА становится ложным и выполнение программы прекращается. В результате остается строка «199».

Задачи для самостоятельного решения

1. Редактор получает на вход строку цифр и преобразовывает ее. Редактор может выполнять две команды, в обеих командах v и w обозначают цепочки цифр.

Заменить (*v*, *w*)

Эта команда заменяет в строке первое слева вхождение цепочки *v* на цепочку *w*. Если цепочки *v* в строке нет, эта команда не изменяет строку.

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 при этом не изменяется.

Дана программа для исполнителя Редактор:

```
НАЧАЛО
ПОКА нашлось (111)
   заменить (111, 2)
   заменить (22, 1)
КОНЕЦ ПОКА
КОНЕЦ
```

Какая строка получится в результате применения приведенной выше программы к строке вида 1…12…2, состоящей из 44 единиц и 21 двойки? В ответе запишите полученную строку.

Ответ: 212.

2. Редактор получает на вход строку цифр и преобразовывает ее. Редактор может выполнять две команды, в обеих командах v и w обозначают цепочки цифр.

Дана программа для исполнителя Редактор:

```
НАЧАЛО
ПОКА нашлось (222) ИЛИ нашлось (888)
```

```
        ЕСЛИ нашлось (222)
            ТО заменить (222, 8)
            ИНАЧЕ заменить (888, 2)
        КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Какая строка получится в результате применения приведенной выше программы к строке, состоящей из 68 идущих подряд цифр 8? В ответе запишите полученную строку.

Ответ: 28.

3. 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 Чертежник находится в точке с координатами (4, 2), то команда **сместиться на (2, -3)** переместит Чертежника в точку (6, -1).

Чертежнику был дан для исполнения следующий алгоритм (буквами *n*, *a*, *b* обозначены неизвестные числа):

```
НАЧАЛО
сместиться на (-1, -2)
ПОВТОРИ n РАЗ
  сместиться на (a, b)
  сместиться на (-1, -2)
КОНЕЦ ПОВТОРИ
сместиться на (-24, -12)
КОНЕЦ
```

Укажите наибольшее возможное значение числа n, для которого найдутся такие значения чисел a и b, что после выполнения программы Чертежник возвратится в исходную точку.

Ответ: -1.

4. 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 координ-

натами ($x + a$; $y + b$). Например, если Чертежник находится в точке с координатами (4, 2), то команда **сместиться на (2, -3)** переместит Чертежника в точку (6, -1).

Чертежнику был дан для исполнения следующий алгоритм (буквами n, a, b обозначены неизвестные числа, при этом $n > 1$):

```
НАЧАЛО
сместиться на (-3, -3)
ПОВТОРИ n РАЗ
   сместиться на (a, b)
   сместиться на (27, 12)
КОНЕЦ ПОВТОРИ
сместиться на (-22, -7)
КОНЕЦ
```

Укажите наименьшее возможное значение числа n ($n > 1$), для которого найдутся такие значения чисел a и b, что после выполнения программы Чертежник возвратится в исходную точку.

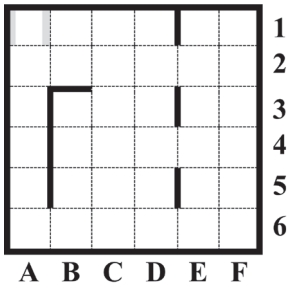
Ответ: 5.

6. Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

вверх вниз влево вправо.

При выполнении любой из этих команд РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →. Четыре команды проверяют истинность условия отсутствия стены у каждой стороны той клетки, где находится РОБОТ:

сверху свободно **снизу свободно**
слева свободно **справа свободно**



Сколько клеток лабиринта соответствуют требованию, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка $A1$)?

```
ПОКА слева свободно ИЛИ сверху свободно
  ЕСЛИ слева свободно
    ТО влево
    ИНАЧЕ вверх
```

```
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
```

Ответ: 4.

ЗАДАНИЕ № 16.
РЕКУРСИЯ. РЕКУРСИВНЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 16 проверяет умение вычислять значения рекуррентных выражений.

Коды проверяемых элементов содержания (по кодификатору):

1.5.3. Индуктивное определение объектов.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к повышенному уровню сложности, не требует для решения специального программного обеспечения и рассчитано на 9 минут.

В демонстрационном варианте 2021 года задание № 16 имеет следующий вид:

Алгоритм вычисления значения функции $F(n)$, где n – натуральное число, задан следующими соотношениями:
$F(n) = 1$ при $n = 1$;
$F(n) = n + F(n - 1)$, если n – четно,
$F(n) = 2*F(n - 2)$, если $n > 1$ и при этом n – нечетно.
Чему равно значение функции $F(26)$?

Что нужно знать для выполнения задания:

Рекурсия – это способ описания функций или процессов через самих себя.

Определение, которое задает некоторый объект в терминах более простого случая этого же объекта, называется рекурсивным.

Рекурсивные определения, позволяющие описать объекты через самих себя, встречаются в математике. К таким определениям относится, например, определение натурального числа:

1) Единица есть натуральное число.

2) Число, следующее за натуральным (т.е. больше его на единицу), есть натуральное число.

Для того, чтобы задать рекурсивную функцию, нужно определить:

• условие окончания рекурсии, то есть значения параметров функции, для которых значение функции известно или вычисляется без рекурсивных вызовов;

• рекуррентную формулу (или формулы), с помощью которых значение функции для заданных значений параметров вычисляется через значение (или значения) функции для других значений параметров (то есть, с помощью рекурсивных вызовов).

Процесс может быть описан алгоритмом, называемым рекурсивным. В таких алгоритмах выделяется два этапа выполнения:

1. «Погружение» алгоритма в себя, т.е. применение алгоритма «в обратную сторону», пока не будет найдено начальное определение, не являющееся рекурсивным.

2. Последовательное построение от начального определения до определения с введенным в алгоритм значением.

Пример 1. Наиболее распространенным рекурсивным определением является определение факториала:

(а) 1! = 1,

(б) $n > 1$, $n! = n*(n-1)!$

На основе этого определения можно записать программу вычисления факториала, использующую рекурсивную функцию.

```
Program 1;
var n, y: integer;
function F(x:integer): integer; {описание рекурсивной функции}
begin
    if x = 1
        then F: = 1 {вызов для начального определения}
        else F: = x*F(x – 1) {вызов для предыдущего определения}
end; {конец описания функции}
begin {начало главной программы}
    readln (n);
    Y: = F(n); {вызов функции в главной программе}
    write (n,'!=',Y)
end. {конец главной программы}
```

Выполним программу для $n = 4$. Рекурсивная функция будет работать следующим образом (при вызове функции значение n присваивается переменной x). Сначала осуществляется «погружение», работает оператор ветви else условного оператора:

1-й шаг: $x = 4$, $x - 1 = 3$, выполняется промежуточное вычисление 4!=4*3!

2-й шаг: $x = 3$, $x - 1 = 2$, выполняется промежуточное вычисление 3!=3*2!

3-й шаг: $x = 2$, $x - 1 = 1$, выполняется промежуточное вычисление 2! = 2*1!

4-й шаг (последний): 1! = 1 по начальному определению, работает оператор *F:* = 1 ветви then условного оператора.

Следующий этап выполнения рекурсивного алгоритма – построение «прямого» определения, от начального до получения результата с исходными для алгоритма данными (числом 4). При этом осуществляется подстановка предыдущих вычислений в более ранние: 5-й шаг: 2! = 2*1 = 2

6-й шаг: 3! = 3*2 = 6

7-й шаг: 4! = 4*6 = 24 – получен результат, он возвращается в главную программу и присваивается переменной *Y*.

Базовые задачи

1. Дан рекурсивный алгоритм:

```
procedure F(n: integer);
begin
   writeln(n);
   if n < 6 then begin
      F(n + 2);
     F(n*3)
   end
end;
```

Найдите сумму чисел, которые будут выведены при вызове $F(1)$.

Решение:

1) Сначала определим рекуррентную формулу; обозначим через $G(n)$ сумму чисел, которая выводится при вызове $F(n)$.

2) При $n >= 6$ процедура выводит число n и заканчивает работу без рекурсивных вызовов.

$G(n) = n$ при $n >= 6$

3) при $n < 6$ процедура выводит число n и дважды вызывает сама себя:

$G(n) = n + G(n + 2) + G(3n)$ при $n < 6$

4) в результате вызова $F(1)$ получаем

$G(1) = 1 + G(3) + G(3)$

$G(3) = 3 + G(5) + G(9) = 3 + G(5) + 9$

$G(5) = 5 + G(7) + G(15) = 5 + 7 + 15 = 27$

5) используем обратную подстановку:

$G(3) = 3 + G(5) + 9 = 3 + 27 + 9 = 39$

$G(1) = 1 + 2*G(3) = 79$

Ответ: 79.

2. Дан рекурсивный алгоритм:

```
procedure F(n: integer);
begin
   writeln('*');
   if n > 0 then begin
      F(n – 2);
      F(n div 2)
   end
end;
```

Сколько символов «звездочка» будет напечатано на экране при выполнении вызова $F(7)$?

Решение:

Сначала определим рекуррентную формулу; обозначим через $G(n)$ количество звездочек, которые выводит программа при вызове $F(n)$.

Из программы видим, что

$G(n) = 1$ при всех $n <= 0$

$G(n) = 1 + G(n – 2) + G(n \text{ div } 2)$ при $n > 0$

вспомним, что ***n* div 2** – это частное от деления n на 2

По этим формулам заполняем таблицу, начиная с нуля:

$G(0) = 1$

$G(1) = 1 + G(-1) + G(0) = 1 + 1 + 1 = 3$

$G(2) = 1 + G(0) + G(1) = 1 + 1 + 3 = 5$

$G(3) = 1 + G(1) + G(1) = 1 + 3 + 3 = 7$

$G(4) = 1 + G(2) + G(2) = 1 + 5 + 5 = 11$

$G(5) = 1 + G(3) + G(2) = 1 + 7 + 5 = 13$

$G(6) = 1 + G(4) + G(3) = 1 + 11 + 7 = 19$

$G(7) = 1 + G(5) + G(3) = 1 + 13 + 7 = 21$

n	0	1	2	3	4	5	6	7
$G(n)$	1	3	5	7	11	13	19	21

Ответ: 21.

3. Алгоритм вычисления значений функций $F(n)$ и $G(n)$, где n – натуральное число, задан следующими соотношениями:

$F(1) = 1$; $G(1) = 1$;
$F(n) = F(n - 1) - G(n - 1)$,
$G(n) = F(n - 1) + G(n - 1)$, при $n >= 2$

Чему равно значение величины $F(5) / G(5)$?
В ответе запишите только целое число.
Решение:
Рекуррентная формула задана для пары $(F(n); G(n))$.
Заметим, что $F(n)$ – это разность предыдущей пары, а $G(n)$ – сумма тех же значений. Заполняем таблицу, начиная с известной первой пары.

n	1	2	3	4	5
$F(n)$	1	0	-2	-4	-4
$G(n)$	1	2	2	0	-4

Искомое значение $F(5) / G(5)$ равно 1.
Ответ: 1.

Задачи для самостоятельного решения

1. Ниже записана рекурсивная процедура:

```
procedure F(n: integer);
begin
   if n > 1 then Begin
      F(n – 4);
      write(n);
      F(n div 2);
   end;
end;
```

Что будет напечатано на экране при выполнении вызова $F(11)$?
Ответ: 3731152.

2. Ниже записаны две рекурсивные процедуры: *F* и *G*:

```
procedure F(n: integer); forward;
procedure G(n: integer); forward;
procedure F(n: integer);
begin
if n > 0 then
   G(n – 1);
end;
procedure G(n: integer);
begin
writeln('*');
if n > 1 then
   F(n – 2);
end;
```

Сколько символов «звездочка» будет напечатано на экране при выполнении вызова *F*(11)?

Ответ: 4.

3. Дан рекурсивный алгоритм:

```
procedure F(n: integer);
begin
   writeln(n);
   if n < 5 then begin
      F(n + 1);
      F(n + 3)
   end
end;
```

Найдите сумму чисел, которые будут выведены при вызове *F*(1).

Ответ: 49.

4. Дан рекурсивный алгоритм:

```
procedure F(n: integer);
begin
  writeln(n);
  if n < 6 then begin
```

```
    F(n + 2);
    F(n*3)
  end
end;
```

Найдите сумму чисел, которые будут выведены при вызове F(1).

Ответ: 79.

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

$F(1) = 0$
$F(n) = F(n - 1) + n$, при $n > 1$
$G(1) = 1$
$G(n) = G(n - 1) * n$, при $n > 1$

Чему равно значение функции $F(5) + G(5)$?

В ответе запишите только натуральное число.

Ответ: 134.

ЗАДАНИЕ № 18. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ, задание № 18 проверяет умение обрабатывать вещественные выражения в электронной таблице.

Коды проверяемых элементов содержания (по кодификатору):

3.4.3. Использование инструментов решения статистических и расчетно-графических задач.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.2. Умение представлять и анализировать табличную информацию в виде графиков и диаграмм.

Задание относится к повышенному уровню сложности, требует для решения программное обеспечение – электронную таблицу и рассчитано на 6 минут.

В демонстрационном варианте 2021 года задание № 18 имеет следующий вид:

Квадрат разлинован на $N \times N$ клеток ($1 < N < 17$). Исполнитель Робот может перемещаться по клеткам, выполняя за одно перемещение одну из двух команд: вправо или вниз. По команде вправо Робот перемещается в соседнюю правую клетку, по команде вниз – в соседнюю нижнюю. При попытке выхода за границу квадрата Робот разрушается. Перед каждым запуском Робота в каждой клетке квадрата лежит монета достоинством от 1 до 100. Посетив клетку, Робот забирает монету с собой; это также относится к начальной и конечной клетке маршрута Робота. Определите максимальную и минимальную денежную сумму, которую может собрать Робот, пройдя из левой верхней клетки в правую нижнюю. В ответе укажите два числа – сначала максимальную сумму, затем минимальную.

Исходные данные записаны в файле **18-0.xls** в виде электронной таблице размером $N \times N$, каждая ячейка которой соответствует клетке квадрата.

Решение (электронные таблицы):

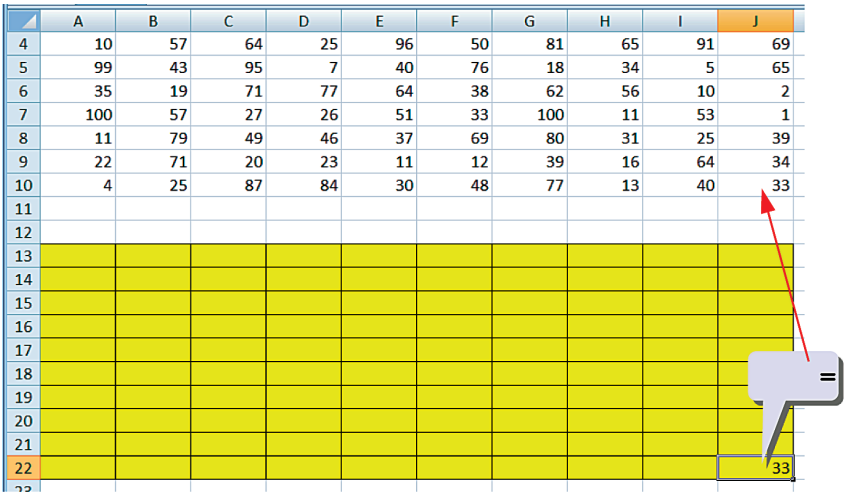
1) Откроем электронную таблицу и увидим следующую таблицу размером 10 на 10:

	A	B	C	D	E	F	G	H	I	J	K
1	51	21	93	48	45	100	67	39	18	29	
2	57	43	97	51	92	10	93	32	19	58	
3	63	16	31	16	78	88	90	72	37	67	
4	10	57	64	25	96	50	81	65	91	69	
5	99	43	95	7	40	76	18	34	5	65	
6	35	19	71	77	64	38	62	56	10	2	
7	100	57	27	26	51	33	100	11	53	1	
8	11	79	49	46	37	69	80	31	25	39	
9	22	71	20	23	11	12	39	16	64	34	
10	4	25	87	84	30	48	77	13	40	33	
11											

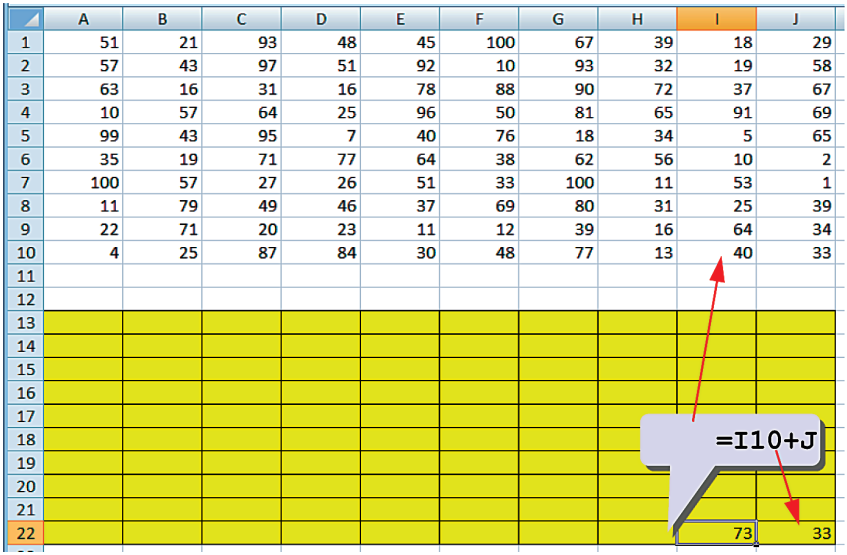
Робот начинает движение из верхнего левого угла (из ячейки $A1$) и перемещается в правый нижний (то есть в $J10$).

2) При использовании метода динамического программирования требуется выделить для вычислений дополнительную таблицу такого же размера и разместить ее под исходной.

3) Предположим, что Робот уже находится в правом нижнем углу; в этом случае он может получить только сумму в этой ячейке, то есть величину $J10$; записываем в $J22$ формулу **=*J*10**; после ввода формулы видим в ячейке $J22$ значение 33, как и ожидалось:



4) Рассмотрим нижний ряд. Если Робот находится в одной из ячеек последней строки 10, то он может идти только вправо, собирая монеты в последней строке; например, начав движение из ячейки ***I*10** он соберет монету в этой ячейке и во всех (в данном случае – в одной) следующих, то есть формула в ячейке ***I*22** должна быть **= *I*10 + *J*22.**



5) Эту формулу копируем по всей строке 22.

6) Аналогично, если Робот находится в последнем столбце, он может двигаться только вниз, собирая по пути все монеты; вводим в ячейку ***J*21** формулу = *J*9 + *J*22 и копируем ее вверх на весь столбец J вспомогательной таблицы.

7) Рассмотрим центральные ячейки. Пусть Робот находится в ячейке ***I*9**; тогда для того, чтобы получить максимальную сумму, ему нужно выбрать лучший из двух путей – пойти в ***I*10** или в ***J*9**; из второй таблицы видим, что в первом случае дополнительно к значению ***I*9** он получит сумму 73, а во втором – только 67. поэтому выгоднее первый вариант; в формуле нужно выбрать максимум из значений ячеек ***I*10** и ***J*9**, получаем **= *I*9 + МАКС(*I*22;*J*21)**. Обратите внимание, что оба аргумента функции **МАКС** находятся в рабочей таблице.

8) Для всех оставшихся ячеек принцип вычисления максимальной суммы тот же самый: нужно добавить к значению этой ячейки в исходной таблице максимум из накопленных сумм, которые Робот собирает в случае двух возможных шагов; копируем формулу из ***I*21** на весь диапазон ***A*13:*I*21**. Первый ответ к задаче – это максимальная сумма, накопленная при движении из левого верхнего угла; она записана в левом верхнем углу рабочей таблицы, то есть в ячейке ***A*13** (она выделена зеленым фоном):

13	**1204**	1153	1132	990	938	893	793	542	415	397
14	1139	1082	1039	942	891	736	726	503	390	368
15	1004	926	884	815	799	721	633	471	371	310
16	941	910	853	729	704	608	543	399	334	243
17	931	832	789	630	598	558	462	316	230	174
18	779	713	694	623	546	482	444	282	225	109
19	744	644	519	492	466	415	382	226	215	107
20	598	587	483	434	388	351	282	193	162	106
21	530	508	432	348	252	223	202	153	137	67
22	441	437	412	325	241	211	163	86	73	33

9) Чтобы найти наименьшую возможную сумму, нужно в формуле в п. 9 заменить функцию МАКС на МИН: **= *I*9 + МИН(*I*22;*J*21)**.

Ответ: 1204 502.

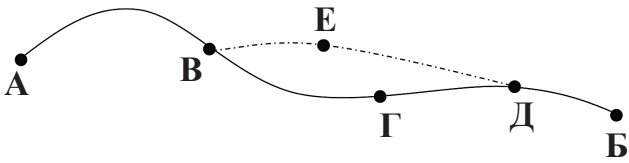
Что нужно знать для выполнения задания:

• в задачах, которые предлагаются в этом задании КИМ, нужно найти оптимальный путь для Робота, который перемещается на клетчатом поле. Робот может на каждом шаге выбирать одно из двух направлений движения (например, только вправо и вниз);

• в каждой клетке Робот получает некоторую награду («берет монету»), и нужно найти такой путь, при котором общая награда будет наибольшая (или наименьшая, если это не награда, а штраф);

• теоретически можно решить такую задачу полным перебором вариантов: рассмотреть все возможные пути и выбрать лучший. Однако количество возможных путей для полей даже не очень большого размера слишком велико для того, чтобы решить эту задачу за время проведения ЕГЭ;

• эта задача успешно и быстро решается с помощью динамического программирования – метода оптимизации, который предложил американский математик Ричард Беллман. Он сформулировал очень простой принцип оптимальности пути: любая часть оптимального пути оптимальна. Например, пусть мы нашли оптимальный путь из точки *А* в точку *Б*, который проходит через точки *В*, *Г* и *Д*:



Принцип Беллмана утверждает, что, например, путь *ВГД* – это оптимальный путь из *В* в *Д*. Если бы это было не так и существовал бы другой, лучший путь между *В* и *Д* (например, *ВЕД* на рисунке), то и путь *АВГДБ* не был бы оптимальным;

• рассмотрим применение динамического программирования на простом примере: Робот идет по клетчатому полю из левого верхнего угла в правый нижний; на каждом шаге он может переместиться на одну клетку вправо или на одну клетку вниз. В каждой клетке лежит заданное количество монет:

	A	*B*	*C*
1	63	78	58
2	10	1	42
3	25	29	87

Нужно найти такой путь из клетки *A*1 в клетку *C*3, пройдя по которому, Робот соберет наибольшее количество монет;

• задачи этого типа решают с конца, то есть, сначала предполагают, что Робот стоит в клетке *C*3, в конечной точке. У него есть един-

ственный вариант – собрать 87 монет, никуда не двигаясь. Построим дополнительную таблицу, в которой для каждой клетки будем записывать наибольшее число монет, которое может собрать Робот, пройдя из этой клетки в конечную. В правом нижнем углу пишем 87:

	A	*B*	*C*
1			
2			
3			87

• рассмотрим клетку *B*3: стартовав из нее, Робот может сделать только один шаг в *C*3, собрав всего 29 + 87 = 116 монет; аналогично при старте из *C*2 робот собирает монеты из ячеек *C*2 и *C*3, всего 42 + 87 = 129:

	A	*B*	*C*
1			
2			129
3		116	87

• заметим, что из *A*3 есть только один возможный путь – через *B*3, Робот соберет 25 + 116 = 141 монету; аналогично из *C*1 есть только путь через *C*2, Робот соберет 58 + 129 = 187 монет:

	A	*B*	*C*
1			187
2			129
3	141	116	87

• если Робот стартует из клетки *B*2, у него есть выбор: пойти вправо (набрав 1 + 129 = 130 монет) или вниз (1 + 116 = 117 монет); очевидно, что первый вариант лучше, так что в эту ячейку записываем 130 (как лучшую из двух сумм);

	A	*B*	*C*
1			187
2		130	129
3	141	116	87

• аналогично в каждую из остальных ячеек записываем лучшую из двух сумм, которые можно получить при движении вправо и вниз:

	A	*B*	*C*
1	328	265	187
2	151	130	129
3	141	116	87

• наибольшая сумма, которую может собрать Робот, двигаясь из левого верхнего угла в правый нижний, находится в ячейке *A*1 рабочей таблицы, она равна 328;

• по рабочей таблице можно восстановить путь Робота. Например, выясним, куда пойдет Робот на первом шаге. Мы знаем, что 328 получено как сумма количества монет в *A*1 (63) и одной из соседних ячеек *B*1 или *A*2. В ячейке, куда должен перейти Робот, должно быть число 328 – 63 = 265, это ячейка *B*1.

Задачи для самостоятельного решения

1. Исходные данные записаны в файле в виде электронной таблицы прямоугольной формы. Определите максимальную и минимальную денежную сумму, которую может собрать Робот, пройдя из левой верхней клетки в правую нижнюю. В ответе укажите два числа – сначала максимальную сумму, затем минимальную.

43	99	79	21	70	56	54	24	42	38
90	38	7	99	94	34	24	40	52	4
9	82	98	20	88	92	76	42	41	98
78	32	9	27	57	20	86	86	7	58
84	37	70	31	26	85	25	100	79	83
76	32	2	48	52	74	28	33	27	27
13	75	34	73	25	78	33	58	71	3
42	93	66	22	54	24	79	98	56	46
97	75	26	12	92	2	91	44	66	40
42	86	31	86	6	22	96	62	41	42

2. Исходные данные записаны в файле в виде электронной таблицы прямоугольной формы. Определите максимальную и минималь-

ную денежную сумму, которую может собрать Робот, пройдя из левой НИЖНЕЙ клетки в правую ВЕРХНЮЮ. В ответе укажите два числа – сначала максимальную сумму, затем минимальную.

43	99	79	21	70	56	54	24	42	38
90	38	7	99	94	34	24	40	52	4
9	82	98	20	88	92	76	42	41	98
78	32	9	27	57	20	86	86	7	58
84	37	70	31	26	85	25	100	79	83
76	32	2	48	52	74	28	33	27	27
13	75	34	73	25	78	33	58	71	3
42	93	66	22	54	24	79	98	56	46
97	75	26	12	92	2	91	44	66	40
42	86	31	86	6	22	96	62	41	42

ЗАДАНИЕ № 23.
ВЫПОЛНЕНИЕ И АНАЛИЗ ПРОСТЫХ АЛГОРИТМОВ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 23 проверяет умение анализировать результат исполнения алгоритма.

Коды проверяемых элементов содержания (по кодификатору):

1.6.2. Вычислимость. Эквивалентность алгоритмических моделей.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к повышенному уровню сложности, не требует для решения специального программного обеспечения и рассчитано на 8 минут.

В демонстрационном варианте 2021 года задание № 23 имеет следующий вид:

Исполнитель преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя – это последовательность команд. Сколько существует программ, для которых при исходном числе 1 результатом является число 20, и при этом траектория вычислений содержит число 10?

Что нужно знать для выполнения задания:

• динамическое программирование – это способ решения сложных задач путем сведения их к более простым задачам того же типа;

• с помощью динамического программирования решаются задачи, которые требуют полного перебор вариантов:

– «подсчитайте количество вариантов…»

– «как оптимально распределить…»

– «найдите оптимальный маршрут…»

• динамическое программирование позволяет ускорить выполнение программы за счет использования дополнительной памяти; полный перебор не требуется, поскольку запоминаются решения всех задач с меньшими значениями параметров.

Базовые задачи

1. Исполнитель *М*17 преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

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

Первая команда увеличивает число на экране на 1, вторая – увеличивает его на 2, а третья – умножает его на 3. Программа для исполнителя *М*17 – это последовательность команд. Сколько существует программ, для которых при исходном числе 2 результатом является число 12 и при этом траектория вычислений содержит числа 8 и 10?

Решение:

1) Запишем рекуррентную формулу для вычисления K_N – количества возможных программ для получения числа N из некоторого начального числа:

$$K_N = K_{N-1} + K_{N-2}, \text{ если } N \text{ не делится на 3}$$
$$K_N = K_{N-1} + K_{N-2} + K_{N-3}, \text{ если } N \text{ делится на 3}$$

2) Все допустимые программы можно разбить на 3 части:
– переход от 2 до 8.
– переход от 8 до 10.
– переход от 10 до 12.

3) Обозначим через $K_{a\to b}$ количеств возможных программ получения числа b из числа a.

4) Очевидно, что если траектория проходит через c, то $K_{a\to b} = K_{a\to c} \cdot K_{c\to b}$ для любого c, такого что $a < c < b$, поэтому $K_{2\to 12} = K_{2\to 8} \cdot K_{8\to 10} \cdot K_{10\to 11}$.

Вычисляем эти значения отдельно стандартным способом по рекуррентным формулам п. 1:

N	2	3	4	5	6	7	8
K_N	1	1	2	3	6	9	15

N	8	9	10
K_N	1	1	2

10	11	12
1	1	2

и перемножаем: $15 \cdot 2 \cdot 2 = 60$

Ответ: 60.

2. Исполнитель преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1

2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя – это последовательность команд. Сколько существует программ, для которых при исходном числе 1 результатом является число 20, и при этом траектория вычислений содержит число 10?

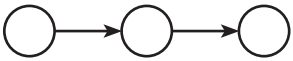
Решение:

Запишем рекуррентную формулу для вычисления K_N – количества возможных программ для получения числа N из некоторого начального числа:

$$K_N = K_{N\text{-}1}, \text{ если } N \text{ не делится на 2,}$$

$$K_N = K_{N\text{-}1} + K_{N/2} + K_{N\text{-}3}, \text{ если } N \text{ делится на } 2B.$$

Все допустимые программы можно разбить на 2 части:
– переход от 1 до 10,
– переход от 10 до 20.



Обозначим через $K_{a\to b}$ количеств возможных программ получения числа b из числа a. Очевидно, что если траектория проходит через c, то $K_{a\to b} = K_{a\to c} \cdot K_{c\to b}$ для любого c, такого что $a < c < b$, поэтому $K_{1\to 20} = K_{1\to 10} \cdot K_{10\to 20}$.

Вычисляем эти значения отдельно стандартным способом по рекуррентным формулам п. 1:

N	1	2	3	4	5	6	7	8	9	10
K_N	1	2	2	4	4	6	6	10	10	14

N	10	11	12	13	14	15	16	17	18	19	20
K_N	1	1	1	1	1	1	1	1	1	1	2

и перемножаем: $14 \cdot 2 = 28$.

Ответ: 28.

Задачи для самостоятельного решения

1. Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1.

2. Умножить на 2.

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 2 результатом является число 29 и при этом траектория вычислений содержит число 14 и не содержит числа 25?

Ответ: 13.

2. У исполнителя Удвоитель две команды, которым присвоены номера:

1. Прибавить 1

2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Удвоитель – это последовательность команд. Сколько существует программ, преобразующих число 4 в число 24, предпоследней командой которых является команда «1»?

Ответ: 18.

3. Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1

2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 1 результатом является число 21 и при этом траектория вычислений содержит число 10? Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы 121 при исходном числе 7 траектория будет состоять из чисел 8, 16, 17.

Ответ: 28.

ТЕМА «ПРОГРАММИРОВАНИЕ»

ЗАДАНИЕ № 6. АНАЛИЗ ПРОГРАММЫ С ЦИКЛОМ

Согласно спецификации контрольно-измерительных материалов, для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 6 проверяет знание основных конструкций языка программирования, понятия переменной, оператора присваивания.

Коды проверяемых элементов содержания (по кодификатору):

1.7.2. Основные конструкции языка программирования. Системы программирования.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.4. Читать и отлаживать программы на языке программирования.

Задание относится к базовому уровню сложности, не требует для решения специального программного обеспечения и рассчитано на 4 минуты.

В демонстрационном варианте 2021 года задание № 6 имеет следующий вид:

Определите, при каком наименьшем введенном значении переменной s программа выведет число 64. Для удобства программа представлена на четырех языках программирования.

Паскаль	Python	Алгоритмический язык

var s, n: integer;
begin
readln (s);
n := 1;
while s < 51 do
begin
s := s + 5;
n := n * 2
end;
writeln(n)
end.
	s = int(input())
	n = 1
	while s < 51:
	s = s + 5
	n = n * 2
	print(n)
		алг
		нач
		цел n, s
		ввод s
		n := 1
		нц пока s < 51
		s := s + 5
		n := n * 2
		кц
		вывод n
		кон

C++

```
#include <iostream>
using namespace std;
int main()
{  int s, n;
   cin >> s;
   n = 1 ;
   while (s < 51) { s = s + 5; n = n * 2; }
   cout << n << endl;
   return 0;
}
```

Решение:

1. Из программы видно, что в конце программы выводится значение переменной ***n***.

2. Из программы видно, что начальное значение переменной ***n*** равно 1.

3. На каждой итерации цикла значение переменной ***n*** увеличивается в 2 раза.

4. Поскольку $64 = 2^6$, для того чтобы получить ***n* = 64**, необходимо выполнить тело цикла 6 раз.

5. При каждой итерации цикла значение переменной ***s*** увеличивается на 5, то есть после 6 итераций мы получим ***s* = *s*0 + 6*5 = *s*0 + 30**, где ***s*0** – введенное начальное значение s.

6. Цикл останавливается при условии ***s* >= 51**, то есть при ***s*0 + 30 >= 51** или ***s*0 >= 21**.

Ответ: 21.

ЗАДАНИЕ № 17. ПРОВЕРКА ДЕЛИМОСТИ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ Задание №17 проверяет умение создавать собственные программы (20–40 строк) для обработки целочисленной информации.

Коды проверяемых элементов содержания (по кодификатору):

1.7.2. Основные конструкции языка программирования. Системы программирования. Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.5. создавать программы на языке программирования по их описанию.

Задание относится к повышенному уровню сложности, требует для решения специального программного обеспечения и рассчитано на 15 минут.

В демонстрационном варианте 2021 года задание № 17 имеет следующий вид:

Рассматривается множество целых чисел, принадлежащих числовому отрезку [1016; 7937], которые делятся на 3 и не делятся на 7, 17, 19, 27.
Найдите количество таких чисел и максимальное из них.
В ответе запишите два целых числа: сначала количество, затем максимальное число. Для выполнения этого задания можно написать программу или воспользоваться редактором электронных таблиц.

Решение:

1. Описываем переменные.

K – количество целых чисел, принадлежащих числовому отрезку [1016; 7937], которые делятся на 3 и не делятся на 7, 17, 19, 27.

Max – максимальное число из числового отрезка [1016; 7937], которое делятся на 3 и не делятся на 7, 17, 19, 27.

I – число из отрезка [1016; 7937].

```
Var
  max, k, i :integer;
```

2. Инициализация переменных *max*, *k*

```
max: = 0;
k: = 0;
```

3. Поиск количества целых чисел, принадлежащих числовому отрезку [1016; 7937], которые делятся на 3 и не делятся на 7, 17, 19, 27. Поиск максимального из таких чисел.

```
For i:=1016 to 7937 do
Begin
 If ((i mod 3)=0) and ((i mod 7)< >0) and ((i mod 17)< >0) and
    ((i mod 19) < > 0)   and ((i mod 27) < > 0) then
```

```
   Begin
     k:=k+1;
     max:=i;
    end;
end;
```

4. Вывод результата (два целых числа: сначала количество, затем максимальное число).

writeln (max, k);

Полный текст программы для решения задания № 17 на языке программирования Pascal.

```
Program m17;
Var
   max, k, i :integer;
Begin
 Max:=0;
K:=0;
For i:=1016 to 7937 do
Begin
 If ((i mod 3)=0) and ((i mod 7)< >0) and ((i mod 17)< >0) and ((i mod
19) < > 0)   and ((i mod 27) < > 0) then
   Begin
     k:=k+1;
     max:=i;
    end;
end;
writeln (max, k);
end.
```

Ответ: 1568 7935.

ЗАДАНИЕ № 22.
АНАЛИЗ ПРОГРАММЫ С ЦИКЛАМИ И ВЕТВЛЕНИЯМИ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание №22 проверяет умение анализировать алгоритм, содержащий ветвление и цикл.

Коды проверяемых элементов содержания (по кодификатору):

1.6.1. Формализация понятия алгоритм.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.4. Читать и отлаживать программы на языке программирования

Задание относится к повышенному уровню сложности, не требует для решения специальное программное обеспечение и рассчитано на 7 минут.

В демонстрационном варианте 2021 года задание №22 имеет следующий вид:

Ниже на четырех языках программирования записан алгоритм. Получив на вход число x, этот алгоритм печатает два числа: L и M. Укажите наибольшее число x, при вводе которого алгоритм печатает сначала 4, а потом 5.

C++	Python

#include <iostream>
usin*G* namespace std;
 	
int main()
{
int x, L, M, Q;
cin >> x;
Q = 9;
L = 0;
while (x >= Q){
L = L + 1;
x = x - Q;
}
M = x;
if (M < L){
M = L;
L = x;
}
cout <<L <<endl <<M <<endl;
return 0;
}
	x = int(input())
	Q = 9
	L = 0
	while x >= Q:
	L = L + 1
	x = x – Q
	M = x
	if M < L:
	M = L
	L = x
	print(L)
	print(M)

Алгоритмический язык	Паскаль

алг
нач
  цел x, L, M, Q
  ввод x
  Q := 9
  L := 0
  нц пока x >= Q
  L := L + 1
  x := x – Q
  кц
  M := x
  если M < L
  то
  M := L
  L := x
  все
  вывод L, нс, M
кон
	var x, L, M, Q: integer;
	Begin
	  readln(x);
	  Q := 9;
	  L := 0;
	  while x >= Q do
	  begin
	  L := L + 1;
	  x := x - Q;
	  end;
	  M := x;
	  if M < L then
	  begin
	  M := L;
	  L := x;
	  end;
	  writeln(L);
	  writeln(M);
	end.

Решение:

1) Видим, что в конце программы на экран выводятся переменные *L* и *M*.

2) Узнаем алгоритм в первой части программы:

```
readln(x);
Q := 9;
L := 0;
while x >= Q do
  begin
    L := L + 1;
    x := x - Q;
end;
```

Этот алгоритм вычисляет (с помощью последовательных вычитаний) частное *L* и остаток *x* от деления исходного значения *x* на *Q*.

3) Узнаем второй алгоритм:

```
M := x;
if M < L then begin
  M := L;
  L := x;
end;
```

Сначала в M просто записывается значение x – остаток от деления исходного значения x на Q. Если $x < L$, то остаток и частное меняются местами так, чтобы в переменной L оказалось меньшее из двух значений, а в M – большее.

4) Таким образом, условию задачи при $Q = 9$ соответствуют два числа – 9 * 4 + 5 = 41 и 9 * 5 + 4 = 49; наибольшее из них – 49.

Ответ: 49.

ЗАДАНИЕ № 24. ОБРАБОТКА СИМВОЛЬНЫХ СТРОК

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 24 проверяет умение создавать собственные программы (10–20 строк) для обработки символьной информации.

Коды проверяемых элементов содержания (по кодификатору):

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к высокому уровню сложности, требует для решения специального программного обеспечения и рассчитано на 18 минут.

В демонстрационном варианте 2021 года задание № 24 имеет следующий вид:

Текстовый файл состоит не более чем из 10^6 символов X, Y и Z. Определите максимальное количество идущих подряд символов, среди которых каждые два соседних различны. Для выполнения этого задания следует написать программу.

Решение:

1. Описываем переменные для обработки данных из файла.

s – массив из символов для обработки данных, *f* – файловая переменная, *a* – символьная переменная, *k* – текущее количество различных идущих подряд символов, *max* – максимальное количество различных идущих подряд символов, *i* – количество символов в текстовом файле 24.txt, *j* – параметр цикла (номер очередного символа в массиве).

```
Var
   s:array [1..1000000] of char;
   i, j, k, max: longint;
   f: text;  a: char;
```

2. Заполняем массив *s* символами из входного файла 24.txt.

```
  assign (f, ‘ 24.txt’); // связываем файловую переменную с именем файла
  reset (f); // открываем файл для чтения
     while not eof(f) do // пока не обработан последний символ файла
       begin
           read (f, a); // считываем очередной символ в файле
           s[i]:=a; // записываем считанный символ из файла в массив символов
           i:=i+1; // переходим в массиве к следующему символу
       end;
```

3. В массиве символов ищем максимальное количество идущих подряд символов, среди которых каждые два соседних различны.

```
     k:=1;
     for j:=2 to i do
       if (s[j-1]<>s[j]) then
            k:=k+1
       else
            begin
               if k>max then
                    max:=k;
                k:=1;
            end;
    writeln (max);
```

Полный текст программы для решения задания №24 на языке программирования Pascal:

```
Var
   s:array [1..1000000] of char;
   i, j, k, max: longint;
   f: text;  a: char;
Begin
     i:=1;
     max:=0;
     assign (f, ‘ 24.txt’);
     reset (f);
     while not eof(f) do
       begin
            read (f,a);
            s[i]:=a;
            i:=i+1;
       end;
     k:=1;
     for j:=2 to i do
       if (s[j-1]<>s[j]) then
            k:=k+1
       else
            begin
                 if k>max then
                      max:=k;
                 k:=1;
            end;
    writeln (max);
end.
```

Ответ: 35.

ЗАДАНИЕ № 25. ОБРАБОТКА ЦЕЛЫХ ЧИСЕЛ. ПРОВЕРКА ДЕЛИМОСТИ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 25 проверяет умение создавать собственные программы (10–20 строк) для обработки целочисленной информации.

Коды проверяемых элементов содержания (по кодификатору):

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к высокому уровню сложности, требует для решения специальнго программного обеспечения и рассчитано на 20 минут.

В демонстрационном варианте 2021 года задание № 25 имеет следующий вид:

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [174457; 174505], числа, имеющие ровно два различных натуральных делителя, не считая единицы и самого числа. Для каждого найденного числа запишите эти два делителя в таблицу на экране с новой строки в порядке возрастания произведения этих двух делителей. Делители в строке таблицы также должны следовать в порядке возрастания.

Например, в диапазоне [5; 9] ровно два целых различных натуральных делителя имеют числа 6 и 8, поэтому для этого диапазона таблица на экране должна содержать следующие значения:

2	*3*
2	*4*

Решение:

Рассмотрим вариант решения данной задачи с использованием языка программирования Pascal.

1. Описываем переменные необходимые для написания кода программы: целочисленные переменные *k* – количество делителей числа, *s* – число из числового отрезка [174457; 174505], *p* – делитель числа *s*.

```
Var
  k, s, p : integer;
```

2. Ищем для каждого из чисел из числового отрезка [174457; 174505] количество натуральных делителей, не считая единицы и самого числа.

```
For s:=174457 to 174505 do
    For p:=2 to (s div 2) do
        If ((s mod p)=0 then
            k:=k+1;
```

3. Если количество найденных натуральных делителей (не считая единицы и самого числа) для числа из числового отрезка [174457; 174505] равно двум, то найденные делители выводим через пробел на экран и переводим курсор на строчку вниз.

```
if k=2 then
    begin
            for p:=2 to (s div 2) do
                if ((s mod p)=0) then
                    write (p, ‘  ‘);
                writeln;
        end;
```

Полный текст программы для решения задания № 25 на языке программирования Pascal:

```
Var
  k, s, p : integer;
Begin
 For s:=174457 to 174505 do
   Begin
     K:=0;
     For p:=2 to (s div 2) do
         If ((s mod p)=0 then
             k:=k+1;
          if k=2 then
             begin
               for p:=2 to (s div 2) do
                   if ((s mod p)=0) then
                       write (p, ‘  ‘);
                   writeln;
               end;
    end;
end.
```

Ответ:

3	58153
7	24923
59	13421
13	13421
149	1171
5	34897
211	827
2	87251

Базовые задачи

1. Программа выводит на экран содержимое указанного текстового файла:

```
Program TextType;
Var
  F:text; s:strinG;
Begin
  Assign (F,'1.txt');
  Reset (F); // открыть файл для чтения
  While not eof(F) do // пока не конец файла
    Begin
      Readln (F,s); // читать строку из файла
      Writeln(S);  // выводить строку
    End;
   Close (F); // закрыть файл
End.
```

2. Программа выполняет копирование текста из файла 1.txt в файл 2.txt:

```
Program CopyText;
Var
  F, G : text; S:strinG;
Begin
 Assign (F,'1.txt');
Reset (F); // открыть файл для чтения
```

```
Assign (G,'2.txt');
Rewrite (G); // открыть файл для записи
While not eof (F) do
 Begin
    Readln(F,S); // читать строку из файла 1.txt
    Writeln (G,S); // записывать строку в файл 2.txt
End;
Close (F); // закрыть файл 1.txt
Close (G); // закрыть файл 2.txt
End.
```

3. Дан целочисленный массив из 30 элементов. Элементы массива могут принимать натуральные значения от 1 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит количество элементов массива, не меньших 1500 и при этом имеющих четное значение, 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 выводится с новой строчки.

Например, для исходного массива из семи элементов:

```
48
1603
2002
1502
1707
706
1906
```

программа должна вывести следующий массив:

```
48
1603
3
3
1707
706
3
```

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.

Алгоритмический язык	Паскаль

алг
нач
цел N = 30
целтаб a[1:N]
цел i, j , k
нц для i от 1 до N
  ввод a [i]
кц
. . .
кон
	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.

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).

Решение:

На языке Паскаль

```
k : = 0;
for i := 1 to N do
if (a[i] >= 1500) and (a [i] mod 2 = 0) then
              k := k + 1;
for i := 1 to N do Begin
if (a[i]>= 1500) and (a[i] mod 2 = 0) then
a [ i ]: = k ;
          writeln (a[i]);
end;
```

4. В программе используется одномерный целочисленный массив A с индексами от 0 до 11. Значения элементов массива $A[i]$ приведены в таблице.

i	0	1	2	3	4	5	6	7	8	9	10	11
$A[i]$	14	13	15	8	4	12	30	21	22	16	5	9

Определите значение переменной s после выполнения следующего фрагмента этой программы (записанного ниже на языках программирования).

Python

```
s = 0
n = 2
for i in ranGe(0, 12):
if A[i] > A[n]:
s += A[i]
else:
A[n] = A[i]
```

Паскаль

```
s := 0;
n := 2;
for i := 0 to 11 do
  if A[i] > A[n] then
s := s + A[i]
  else
     A[n] := A[i];
```

C++

```
s = 0;
n = 2;
for (int i = 0; i < 12; i++)
      {
if (A[i] > A[n])
s += A[i];
else
A[n] = A[i];
}
```

Решение:

В цикле каждый элемент массива сравнивается с $A[2]$, если текущий элемент больше $A[2]$, то текущий элемент прибавляется к сумме s, в противном случае элементу массива с индексом 2 присваивается значение текущего элемента. Для элементов массива с 0 до 4 условие ($A[i] > A[n]$) не выполняется. Для элементов массива с 5 по 11 условие ($A[i] > A[n]$)выполняется, поэтому $s = 12 + 30 + 21 + 22 + 16 + 5 + 9$.

Ответ: 115.

4. Ниже на языках программирования записан алгоритм. Получив на вход натуральное десятичное число x, этот алгоритм печатает два числа: L и M. Укажите наибольшее число x, при вводе которого алгоритм выводит сначала 2, а потом 3.

Python

```
x = int(input())
L = 0
M = 0
while x > 0:
M=M+1
if x % 2 == 0:
L = L + x % 8
x = x // 8
print(L)
print(M)
```

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 + x mod 8;
x := x div 8
end;
```

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 + x % 8;
}
x = x / 8;
}
cout << L << endl << M << endl;
return 0;
}
```

Решение:

1) Для решения задачи необходимо понять, что делает эта программа. Пока в числе x, переведенном в восьмеричную систему счисления, есть хотя бы одна цифра, в цикле выполняются следующие действия: если число x четное ($x \bmod 2 = 0$), то переменная «L» увеличивается на последнюю цифру восьмеричного представления числа x ($L := L + x \bmod 8$). После этого отбрасывается последняя цифра восьмеричного представления числа x ($x := x \operatorname{div} 8$) и, если цифры в числе x еще остались, то все повторяется.

2) Сначала алгоритм выводит $L = 2$, затем $M = 3$. Т.к. $M = 3$ значит в восьмеричном представлении числа x 3 цифры. В результате L будет записана сумма всех четных цифр восьмеричной записи введенного числа, по условию эта сумма равна 2.

3) Так как число должно быть наибольшим, то две цифры берем максимальные в восьмеричной системе счисления – 7 и ставим их на первое и второе место. Выстраиваем цифры в порядке возрастания: 772_8.

4) Теперь переведем число в десятичную систему, ответ нужно дать в десятичной.

$772_8 = 7 \cdot 8^2 + 7 \cdot 8^1 + 2 \cdot 80 = 506_{10}$

Ответ: 506.

5. На обработку поступает последовательность из четырех неотрицательных целых чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран количество четных чисел в исходной последовательности и максимальное четное число. Если четных чисел нет, требуется на экран вывести «NO». Известно, что вводимые числа не превышают 1000. Программист написал программу неправильно. Ниже написанная им программа для Вашего удобства приведена на пяти языках программирования.

Паскаль

```
const n = 4;
var i, x: integer;
var maximum, count: integer; Begin
count := 0;
maximum := 1000;
for i := 1 to n do
begin
read(x);
if x mod 2 = 0 then
Begin
count := count + 1;
if x > maximum then
maximum := i
end
end;
if count > 0 then
begin
writeln(count);
writeln(maximum)
end
else
writeln('NO')
end.
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе последовательности: 2 15 44 15.

2. Приведите пример такой последовательности, содержащей хотя бы одно четное число, что при ее вводе приведенная программа, несмотря на ошибки, выведет правильный ответ.

3. Найдите допущенные программистом ошибки и исправьте их. Исправление ошибки должно затрагивать только строку, в которой находится ошибка. Для каждой ошибки:

1) Выпишите строку, в которой сделана ошибка.

2) Укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Известно, что в тексте программы нужно исправить не более двух строк так, чтобы она стала работать правильно.

Достаточно указать ошибки и способ их исправления для одного языка программирования.

Обратите внимание на то, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения.

Примечание. 0 делится на любое натуральное число.

Решение:

Решение использует запись программы на Паскале. Допускается использование программы на других языках.

1. Программа выведет два числа: 2 и 1000.

2. Пример последовательности, содержащей четные числа, для которой программа работает правильно: 1 2 3 1000.

Замечание для проверяющего. В конце работы программы значение переменной maximum равно 1000. Соответственно, программа будет работать верно, если в последовательности есть 1000. Выведенное количество четных чисел будет правильным в любом случае.

3. В программе есть две ошибки.

Первая ошибка: неверная инициализация maximum.

Строка с ошибкой:

maximum : = 1000;

Верное исправление:

maximum : = 0;

Вместо 0 может быть использовано любое отрицательное число.

Вторая ошибка: неверное присваивание при вычислении максимума.

Строка с ошибкой:

maximum : = i

Верное исправление:

maximum : = x

6. В программе используется одномерный целочисленный массив A с индексами от 0 до 11. Значение элементов равны 20, 19, 17, 41, 23, 12, 24, 16, 4, 13, 6, 15 соответственно, т.е. A[0] =20; A[1] =19 и т.д. Определите значение переменной s после выполнения следующего фрагмента этой программы.

```
s:=0; n:=5;
for i:=0 to 11 do
    if A[i] <= A[n] then
    begin
      s:=s+i;
      t:=A[i];
      A[i]:=a[n];
      A[n]:=t;
     end;
```

Решение:

1) Сначала попытаемся понять, что же делает эта программа:

• в цикле рассматриваются пары элементов, начиная с пары (***A*[0], *A*[5]**) и заканчивая парой (***A*[11], *A*[5]**);

• если элемент ***A*[*i*]** меньше или равен элементу (***A*[5]**), они меняются местами через вспомогательную переменную ***t***.

2) Начальное значение переменной s, которая нас интересует, равно нулю; при каждой перестановке оно увеличивается на *i* (индекс элемента, который меньше или равен ***A*[5]**), то есть, *s* – сумма индексов элементов, которые меньше или равны ***A*[5]**.

3) условие ***A*[*i*] <= *A*[5]**, выполняется для *i* = 5, а также для *i* = 8.

4) поэтому *s* = 5 + 8 после выполнения этого фрагмента значение переменной *s* будет равно 13.

Ответ: 13.

7. Напишите в ответе число, которое будет выведено в результате выполнения следующего алгоритма.

```
var a, b, t, M, R: longint;
function F(x: longint): longint;
 begin
     F := 2 * (x * x - 36) * (x * X - 36) + 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;
write (M + R)
end.
```

Решение:

1) Заметим, что в программе есть цикл, в котором переменная ***t*** принимает последовательно все целые значения в интервале от ***a*** до ***b***:

```
for t:=a to b do begin
  ...
end;
```

2) До начала цикла в переменную ***M*** записывается значение ***a***, а в переменную ***R*** – значение функции в точке ***a***:

```
M:=a; R:=F(a);
```

3) Внутри цикла есть условный оператор, в котором вычисляется значение функции ***F(t)*** и сравнивается со значением переменной ***R***:

```
if (F(t) <= R) then
        begin
          M := t;
          R := F(t);
        end;
```

Если новое значение функции меньше или равно значению ***R***, в ***R*** записывается значение функции в точке ***t***, а в переменной ***M*** запоминается само значение ***t*** (аргумент функции, соответствующий значению в ***R***).

4) В результате анализа пп. 1–3 можно сделать вывод, что цикл ищет минимум функции ***F(t)*** на интервале от ***a*** до ***b***, и после выполнения цикла в переменной *M* оказывается значение аргумента *t*, при котором функция достигает минимума на заданном интервале (здесь это интервал [-20, 20]).

5) Функция, которая используется в программе, – это квадратичная парабола: $y = (x^2 - 36)^2 + 27$, ее ветви направлены вверх (коэффициент при x^4 положительный, равен 1); она имеет два минимума в точках $x = -6$ и $x = 6$.

6) Обе точки минимума находятся на отрезке [-20; 20], поэтому программа найдет одну из этих точек; вопрос: какую именно?

7) Для квадратичной параболы обе точки минимума имеют одинаковую y-координату, а запоминание новой точки минимума происходит, когда вычисленное значение $F(t)$ станет **меньше или равно *R***:

```
if (F(t) <= R) then
        begin
          M := t;
          R := F(t);
        end;
```

8) Таким образом, в переменной M останется значение «6»;

9) Обратим внимание, что на экран выводится не M, а $M + R$, $R = 27$ поэтому результат будет равен 6 + 27 = 33.

Ответ: 33.

Задачи для самостоятельной работы

1. При каком наибольшем введенном числе d после выполнения программы будет напечатано 55?

```
var n, s, d: integer;
begin
  readln(d);
  n := 0;
  s := 0;
  while s <= 365 do begin
    s := s + d;
    n := n + 5
  end;
  write(n)
end.
```

Ответ: 36.

2. Определите, что будет напечатано в результате работы следующего фрагмента программы:

```
var k, s: integer;
begin
  s:=0;
  k:=0;
  while s < 1024 do begin
    s:=s+10;
    k:=k+1;
  end;
  write(k);
end.
```

Ответ: 103.

3. Рассматривается множество целых чисел, принадлежащих отрезку [1033; 7737], которые делятся на 5 и не делятся на 11, 17, 19 и 23. Найдите количество таких чисел и максимальное из них. В ответе запишите два числа через пробел: сначала количество, затем максимальное число.

Ответ: 1040 7730.

4. Рассматривается множество целых чисел, принадлежащих отрезку [3201; 12876], которые делятся на 4 и не делятся на 7, 11, 13 и 19. Найдите количество таких чисел и максимальное из них. В ответе запишите два числа через пробел: сначала количество, затем максимальное число.

Ответ: 1649 12876.

5. Ниже приведен алгоритм. Укажите наименьшее число x, при вводе которого алгоритм напечатает сначала 2, потом – 5.

```
var x, a, b: longint;
begin
readln(x);
  a := 0; b := 1;
  while x > 0 do begin
    if x mod 2 > 0 then
      a := a + 1
    else
      b := b + (x mod 7);
```

```
    b := b + (x mod 7);
   x := x div 7;
  end;
  writeln(a); write(b);
end.
```

Ответ: 203.

6. Ниже записана программа. Получив на вход число x, эта программа печатает два числа, L и M. Укажите наибольшее из таких чисел x, при вводе которых алгоритм печатает сначала 3, а потом 8.

```
var x, L, M: longint;
begin
  readln(x);
  L:=0; M:=0;
  while x > 0 do begin
    L:= L + 1;
    if x mod 2 = 0 then
      M:= M + x mod 10;
    x:= x div 10;
  end;
  writeln(L); write(M);
end.
```

Ответ: 998.

7. Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [154026; 154043], числа, имеющие ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.

Ответ:

1	5	20543	102715
1	59	1741	102719
1	139	739	102721
1	2	51361	102722

8. Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [102714; 102725], числа, имеющие

ровно 4 различных делителя. Выведите эти четыре делителя для каждого найденного числа в порядке возрастания.

Ответ:

1	3	51343	154029
1	2	77017	154034
1	31	4969	154039
1	3	51347	154041

ЗАДАНИЕ № 26. ОБРАБОТКА МАССИВА ЦЕЛЫХ ЧИСЕЛ ИЗ ФАЙЛА. СОРТИРОВКА

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 26 проверяет умение обрабатывать целочисленную информацию с использованием сортировки.

Коды проверяемых элементов содержания (по кодификатору):

1.6.3. Построение алгоритмов и практические вычисления.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к высокому уровню сложности, требует для решения специальное программное обеспечение и рассчитано на 35 минут.

В демонстрационном варианте 2021 года задание №26 имеет следующий вид:

Системный администратор раз в неделю создает архив пользовательских файлов. Однако объем диска, куда он помещает архив, может быть меньше, чем суммарный объем архивируемых файлов.

Известно, какой объем занимает файл каждого пользователя.

По заданной информации об объеме файлов пользователей и свободном объеме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранен в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные.

В первой строке входного файла находятся два числа: S – размер свободного места на диске (натуральное число, не превышающее 10 000) и N – количество пользователей (натуральное число, не превышающее 1000). В следующих N строках находятся значения объемов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое в отдельной строке.

Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранен в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Пример входного файла:

```
100 4
80
30
50
40
```

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объемы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объем файла из перечисленных пар – 50, поэтому ответ для приведенного примера:

```
2 50
```

Входной файл для данной задачи имеет следующий вид (фрагмент)[1]:

```
8200 970
34
35
4
30
18
16
```

[1] Материалы взяты с официального сайта ФИПИ (http://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2021/inf-ege-2021.zip)

```
26
5
5
4
39
…
3
38
33
```

Входной файл для данного примера содержит 971 строку. В первой строке находится служебная информация: первое число – размер свободного места на диске (8200), второе число – количество пользователей (970), в следующих 970-ти строках находятся значения объемов файлов каждого пользователя.

Нам необходимо найти две величины:

1. Наибольшее число пользователей, чьи файлы могут быть помещены в архив, размер которого не превышает 8200.

2. Максимальный размер файла в этом архиве.

Независимо от способа решения, общий алгоритм будет следующий:

1. Сортируем по возрастанию размеры файлов пользователей.

2. Просматриваем отсортированную последовательность в прямом направлении и накапливаем сумму просмотренных элементов до тех пор, пока эта сумма будет меньше или равна максимальному размеру архива (8200).

3. Ответом на первый вопрос будет являться количество слагаемых в последней сумме, которая не превышает максимальный размер архива (8200), а на второй – последнее слагаемое в такой сумме (или одно из следующих потенциальных слагаемых, которым можно заменить последнее слагаемое без превышения допустимого размера архива).

Задание № 26 можно решить с помощью двух основных подходов: обработка целочисленных данных с помощью электронных таблиц или реализация алгоритма на одном из языков программирования.

Рассмотрим вариант решения данной задачи с использованием MS Excel.

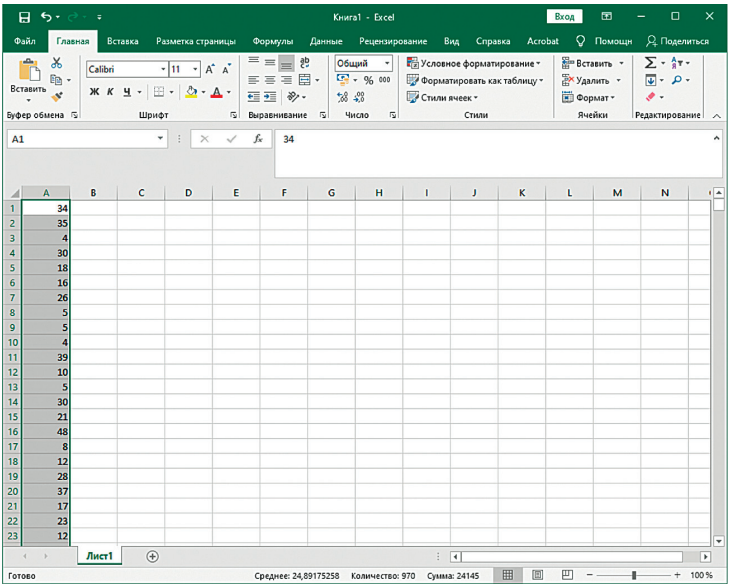
1. Открываем файл с исходными данными.

2. Выписываем максимальный размер архива (это число 8200) и удаляем из файла первую строку.

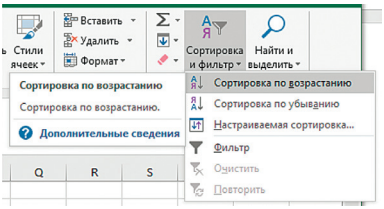
3. Выделяем все строки файла (Ctrl+A) и копируем их в буфер обмена (Ctrl+C).

4. Открываем MS Excel и создаем в нем новый документ.

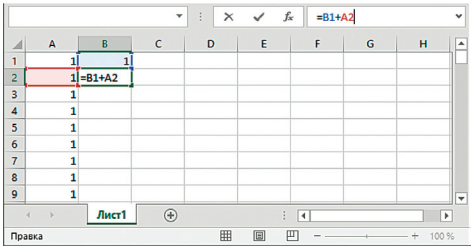
5. Устанавливаем курсор в первую ячейку и вставляем данные из буфера обмена (Ctrl+V).



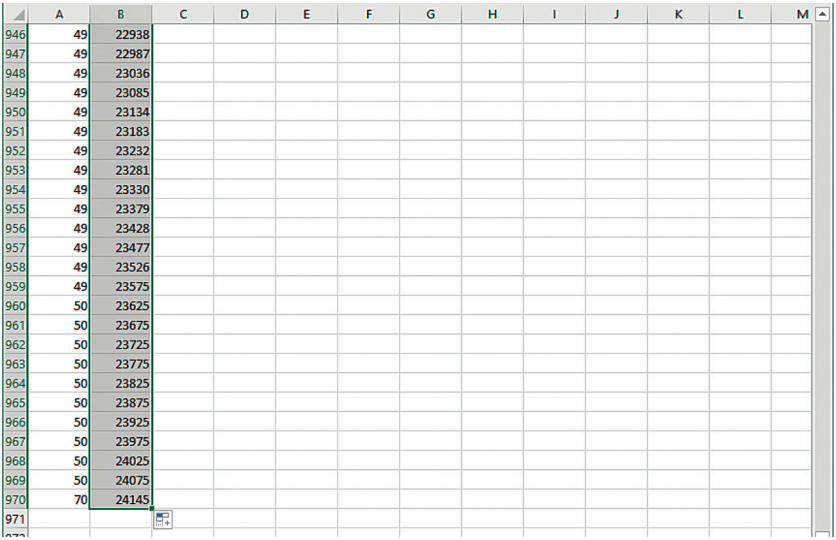
6. Выделим столбец A и отсортируем его по возрастанию.



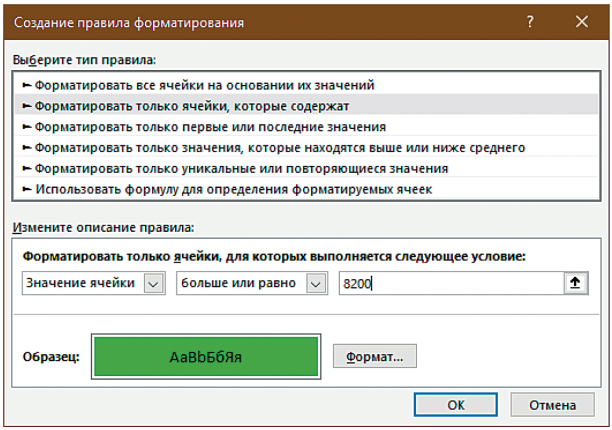
7. В столбце *B* мы будем накапливать сумму предыдущих (включая текущий) элементов отсортированной последовательность. В ячейку *B*1 просто копируем элемент из *A*1. А в ячейке *B*2 запишем формулу = *B*1 + *A*2.



8. Протягиваем эту формулу до конца последовательности данных.



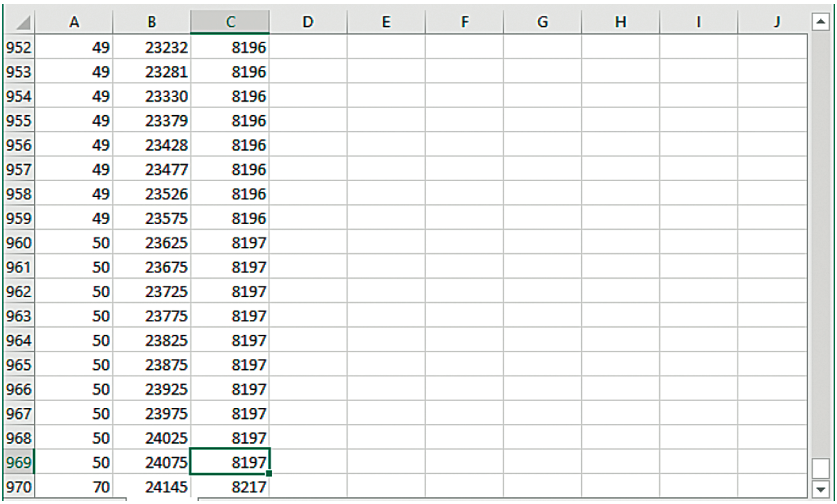
9. Можно просмотреть столбец *B* и найти первое число, которое превышает 8200 (это число мы выписали из исходного файла), а можно воспользоваться условным форматированием. Создадим правило для столбца *B*.



10. Все ячейки, значения в которых не превышают 8200, окрасятся в зеленый цвет. И мы легко найдем последнюю зеленую ячейку. Номер строки, который соответствует последней зеленой ячейке, будет ответом на первый вопрос – наибольшее число пользователей, чьи файлы могут быть помещены в архив. Для данного примера **568**.

	A	B	C	D	E	F	G	H	I	J
552	29	7712								
553	29	7741								
554	29	7770								
555	29	7799								
556	29	7828								
557	29	7857								
558	29	7886								
559	29	7915								
560	29	7944								
561	29	7973								
562	29	8002								
563	29	8031								
564	29	8060								
565	29	8089								
566	29	8118								
567	29	8147								
568	29	8176								
569	30	8206								
570	30	8236								
571	30	8266								
572	30	8296								

11. Мы видим, что последнее слагаемое 29, а последняя сумма, которая не превышает 8200, равна 8176. Нам необходимо найти одно из следующих потенциальных слагаемых (если это возможно), которым можно заменить последнее слагаемое (29) без превышения допустимого размера архива (8200). В ячейке *C*568 запишем формулу = *B*568 – *A*568, а в следующей ячейке *C*569 запишем формулу = C568 + A569 и протащим эту формулу вниз до конца последовательности данных.



	A	B	C	D	E	F	G	H	I	J
952	49	23232	8196							
953	49	23281	8196							
954	49	23330	8196							
955	49	23379	8196							
956	49	23428	8196							
957	49	23477	8196							
958	49	23526	8196							
959	49	23575	8196							
960	50	23625	8197							
961	50	23675	8197							
962	50	23725	8197							
963	50	23775	8197							
964	50	23825	8197							
965	50	23875	8197							
966	50	23925	8197							
967	50	23975	8197							
968	50	24025	8197							
969	50	24075	8197							
970	70	24145	8217							

12. Ищем последнее значение столбца *C*, которое не превышает 8200. Это значение 8197. Ответом на второй вопрос будет содержимое ячейки столбца *A*, которая расположена с строке, соответствующей значению суммы 8197. Это значение **50**.

13. Записываем ответ **568 50**.

Рассмотрим вариант решения данной задачи с использованием языка программирования Pascal.

1. Описываем переменные для обработки данных из файла. Всю последовательность мы сохраняем в массив **data**.

```
var
  s, n: integer; //размер свободного места и количество пользователей
  f: text;
```

```
  i, j, k: integer;
  c, sum: integer; //количество файлов в архиве и суммарный объем
  p: integer; //максимальный размер файла в архиве
  data: array [1..1000] of integer; //массив для обработки данных
```

2. Заполняем массив **data** числами из входного файла.

```
Assign( f, '26.txt' );
Reset( f );
Readln( f, s, n );
for i := 1 to n do
Readln( f, data[i] );
```

3. Выполняем сортировку массива по возрастанию. В данном примере использован метод «пузырька».

```
for i := 1 to n-1 do
  for j := 1 to n-i-1 do
    if data[j] > data[j+1] then begin
      k := data[j];
      data[j] := data[j+1];
      data[j+1] := k;
end;
```

4. Находим количество файлов в архиве и суммарный объем.

```
sum := 0;
for i := 1 to n do
  if sum + data[i] > s then break
    else begin
          sum := sum + data[i];
          c := i;
  end;
```

5. Ищем размер самого большого файла в архиве и выводим ответ.

```
    p := data[c];
    for i := c + 1 to n do
      if sum - data[c] + data[i] > s then break
        else p := data[i];

    Writeln(c, ' ', p);
```

Полный текст программы для решения задания № 26 на языке программирования Pascal (в данном примере используются только стандартные средства языка):

```
Program task26;
var
   s, n: integer; //размер свободного места и количество пользователей
    f: text;
    i, j, k: integer;
    c, sum: integer; //количество файлов в архиве и суммарный объем
    p: integer; //максимальный размер файла в архиве
    data: array [1..1000] of integer; //массив для обработки данных
begin
    Assign( f, '26.txt' );
    Reset( f );
    Readln( f, s, n );
    for i := 1 to n do
      Keadln( f, data[i] );
        for i := 1 to n-1 do
      for j := 1 to n-i-1 do
        if data[j] > data[j+1] then begin
           k := data[j];
           data[j] := data[j+1];
           data[j+1] := k;
        end;
    sum := 0;
    for i := 1 to n do
      if sum + data[i] > s then break
         else begin
               sum := sum + data[i];
               c := i;
         end;
    p := data[c];
    for i := c + 1 to n do
      if sum - data[c] + data[i] > s then break
         else p := data[i];
  Writeln(c, ' ', p);
end.
```

ЗАДАНИЕ № 27.
ОБРАБОТКА ДАННЫХ, ВВОДИМЫХ ИЗ ФАЙЛА В ВИДЕ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ

Согласно спецификации контрольно-измерительных материалов для проведения в 2021 году единого государственного экзамена по информатике и ИКТ задание № 27 проверяет умение создавать собственные программы (20–40 строк) для анализа числовых последовательностей.

Коды проверяемых элементов содержания (по кодификатору):

1.6.3. Построение алгоритмов и практические вычисления.

Коды проверяемых требований к уровню подготовки (по кодификатору):

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Задание относится к высокому уровню сложности, требует для решения специального программного обеспечения и рассчитано на 35 минут.

В демонстрационном варианте 2021 года задание № 27 имеет следующий вид:

Имеется набор данных, состоящий из пар положительных целых чисел. Необходимо выбрать из каждой пары ровно одно число так, чтобы сумма всех выбранных чисел не делилась на 3 и при этом была максимально возможной. Гарантируется, что искомую сумму получить можно.

Программа должна напечатать одно число – максимально возможную сумму, соответствующую условиям задачи.

Входные данные.

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество пар *N* ($1 \le N \le 100000$). Каждая из следующих *N* строк содержит два натуральных числа, не превышающих 10000.

Пример организации исходных данных во входном файле:

6
1 3
5 12
6 9

5 4
3 3
1 1

Для указанных входных данных значением искомой суммы должно быть число 32.

В ответе укажите два числа: сначала значение искомой суммы для файла *A*, затем для файла *B*.

Предупреждение: для обработки файла *B* не следует использовать переборный алгоритм, вычисляющий сумму для всех возможных вариантов, поскольку написанная по такому алгоритму программа будет выполняться слишком долго.

Входной файл A для данной задачи имеет следующий вид[2]:

```
20
5627  5841
5544  6520
1449  3580
2984  5984
6164  2583
9588  3467
1440  8636
7706  8023
6847  6023
577  1545
7361  5893
4221  5994
3118  5054
1546  4062
780  3433
6926  2390
3702  6714
2278  7180
9156  3466
2294  8733
```

[2] Материалы взяты с официального сайта ФИПИ (http://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2021/inf-ege-2021.zip)

Входной файл B для данной задачи имеет следующий вид (фрагмент)[3]:

```
60000
7722 7518
906 1474
859 1688
425 3358
2312 8232
5322 1618
4438 1697
1205 5119
2043 6171
…
4313 8124
7669 170
43 9752
```

В данной задаче мы видим, что файл *B* содержит 60000 пар и применить для поиска нужной суммы переборный алгоритм с предварительным сохранением всех пар в массив не представляется возможным, поэтому построим универсальный эффективный алгоритм, который подойдет для обоих файлов.

В общем виде алгоритм будет следующим:

• считываем число, соответствующее количеству пар;

• при считывании очередной пары чисел выполняем следующие действия: находим максимум, прибавляем его к общей сумме, находим разность между этими элементами, если она не делится на 3 и минимальна среди всех просмотренных пар, то записываем значение этой разности в соответствующую переменную;

• проверяем сумму после просмотра всех пар, если она не делится 3, то выводим ее значение, если делится на 3, то уменьшаем ее значение, на величину равную минимальной разности (которая не делится на 3) между элементами одной пары и выводим полученное значение.

Рассмотрим вариант решения данной задачи с использованием языка программирования Pascal.

---

[3] Материалы взяты с официального сайта ФИПИ (http://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2021/inf-ege-2021.zip)

1. Описываем переменные для обработки данных из файла.

```
var
   n: integer; //количество пар
   f: text;
   i, a: integer;
   mina, maxa: integer; //минимальный и максимальный элемент в паре
   sum: longint; //максимально возможная сумма
   d: integer; //минимальный разность между элементами пары, которая не делится на 3
```

2. Считываем данные из файла и просматриваем каждую пару.

```
   Assign( f, '27-B.txt' );
   Reset( f );
   Readln( f, n );

   d := 10001;
   sum := 0;
   for i := 1 to n do begin
    readln( f, mina, maxa );
```

3. В каждой паре определяем минимальный и максимальный элемент.

```
    if mina > maxa then begin
      a := mina;
      mina := maxa;
      maxa := a;
end;
```

4. Находим минимальную разность между элементами одной пары, которая не делится на 3.

```
    if ((maxa - mina) mod 3 <> 0) and (maxa - mina < d) then
      d := maxa - mina;
```

5. Прибавляем максимальный элемент пары к общей сумме.

```
     sum := sum + maxa;
```

6. Производим корректировку суммы, если она кратна 3. Выводим ответ.

```
    if sum mod 3 = 0 then sum := sum - d;
Writeln(sum);
```

Полный текст программы для решения задания № 27 на языке программирования Pascal (в данном примере используются только стандартные средства языка):

```
Program task27;
var
   n: integer; //количество пар
   f: text;
   i, a: integer;
   mina, maxa: integer; //минимальный и максимальный элемент в паре
   sum: longint; //максимально возможная сумма
    d: integer; //минимальный разность между элементами пары, которая не делится на 3
begin
   Assign( f, '27-B.txt' );
   Reset( f );
   Readln( f, n );

   d := 10001;
   sum := 0;
   for i := 1 to n do begin
     readln( f, mina, maxa );
     if mina > maxa then begin
       a := mina;
       mina := maxa;
       maxa := a;
     end;
     if ((maxa - mina) mod 3 <> 0) and (maxa - mina < d) then
       d := maxa - mina;
     sum := sum + maxa;
   end;
    if sum mod 3 = 0 then sum := sum - d;
   Writeln(sum);
end.
```

Для того, чтобы обработать файл A, необходимо в строке связывания файловой переменой с именем файла указать Assign(f, '27-A.txt'). Для файла A из демонстрационного варианта 2021 года[4] данная программы выдаст ответ: **127127**.

Для того, чтобы обработать файл B необходимо в строке связывания файловой переменой с именем файла указать Assign(f, '27-B.txt'). Для файла B из демонстрационного варианта 2021 года[5] данная программы выдаст ответ: **399762080**.

За верный ответ на задания 26 и 27 ставится по 2 балла; если значения в ответе перепутаны местами или в ответе присутствует только одно верное значение (второе неверно или отсутствует) – ставится 1 балл. В остальных случаях – 0 баллов.

[4] Материалы взяты с официального сайта ФИПИ (http://doc.fipi.ru/ege/demoversii-specifikacii-kodifikatory/2021/inf-ege-2021.zip)

[5] Там же.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Голубев О.Б., Морозова И.В. Анализ выполнения заданий в содержательных разделах ЕГЭ по информатике и ИКТ // Вестник Вологодского государственного университета. Серия: гуманитарные, общественные, педагогические науки. – 2016. – №3. – С.73–76.

2. Зорина Е.М., Зорин М.В., ЕГЭ 2020. Информатика. Сборник заданий: 350 заданий с ответами. — М.: Эксмо, 2019.

3. Крылов С.С. ЕГЭ. Информатика и ИКТ: типовые экзаменационные варианты: 20 вариантов / С.С. Крылов, Т.Е. Чуркина. – М.: Издательство «Национальное образование», 2020. – 448 с.

4. Лещинер В.Р. ЕГЭ 2020. Информатика. 14 вариантов / В.Р. Лещинер. — М.: Экзамен, 2019.

5. Поляков К.Ю., Еремин Е.А. Как нам реорганизовать ЕГЭ по информатике? // Информатика в школе. – № 3. – 2019. – С. 2–7.

6. Самылкина Н.Н. ЕГЭ 2020. Информатика: тематические тренировочные задания / Н.Н. Самылкина, И.В. Синицкая, В.В. Соболева. – М.: Эксмо, 2019. – 176 с.

7. Ушаков Д.М. ЕГЭ-2020. Информатика. 20 тренировочных вариантов экзаменационных работ для подготовки к ЕГЭ. — М.: АСТ, 2019.

8. http://www.fipi.ru – открытый банк заданий ЕГЭ.

9. https://inf-e*G*e.sdam*G*ia.ru – образовательный портал для подготовки к экзаменам «решу ЕГЭ».

10. http://kpolyakov.spb.ru/school/e*G*e.htm – сайт подготовки к ЕГЭ по информатике К.Ю. Полякова.

Учебное издание

Работа над задачами по темам «Элементы теории алгоритмов» и «Программирование» при подготовке обучающихся к ГИА по информатике

Учебное пособие для подготовки к итоговой государственной аттестации выпускников основной и старшей школы

Подписано в печать 03.11.2020. Формат 60×84/16.
Печать офсетная. Гарнитура Times.
Усл. печ. л. 4,65. Тираж 600 экз. Заказ 1765

Вологодский институт развития образования
160011, г. Вологда, ул. Козленская, 57
E-mail: izdat@viro.edu.ru