ФЕДЕРАЛЬНАЯ СЛУЖБА ПО НАДЗОРУ В СФЕРЕ ОБРАЗОВАНИЯ И НАУКИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ НАУЧНОЕ УЧРЕЖДЕНИЕ
«ФЕДЕРАЛЬНЫЙ ИНСТИТУТ ПЕДАГОГИЧЕСКИХ ИЗМЕРЕНИЙ»

Методические материалы для председателей и членов предметных комиссий субъектов Российской Федерации по проверке выполнения заданий с развернутым ответом экзаменационных работ ЕГЭ 2020 года

ИНФОРМАТИКА И ИКТ

Москва
2020

Автор-составитель: Крылов С.С.

Методические материалы для председателей и членов предметных комиссий субъектов Российской Федерации по проверке выполнения заданий с развернутым ответом экзаменационных работ ЕГЭ 2020 г. по информатике и ИКТ подготовлены в соответствии с Тематическим планом работ Федерального государственного бюджетного научного учреждения «Федеральный институт педагогических измерений» на 2020 г. Пособие предназначено для подготовки экспертов по оцениванию выполнения заданий с развернутым ответом, которые являются частью контрольных измерительных материалов (КИМ) для сдачи единого государственного экзамена (ЕГЭ) по информатике и ИКТ.

В методических материалах дается краткое описание структуры контрольных измерительных материалов 2020 г. по информатике и ИКТ, характеризуются типы заданий с развернутым ответом, используемые в КИМ ЕГЭ по информатике и ИКТ, и критерии оценки выполнения заданий с развернутым ответом, приводятся примеры оценивания выполнения заданий и даются комментарии, объясняющие выставленную оценку.

Авторы будут благодарны за замечания и предложения по совершенствованию пособия.

© Крылов С.С. 2020.
© Федеральный институт педагогических измерений, 2020

ОГЛАВЛЕНИЕ

1. Общие подходы к разработке контрольных измерительных материалов ЕГЭ по информатике и ИКТ

Разработка системы единого государственного экзамена (ЕГЭ) включает в себя создание большого количества взаимосвязанных подсистем. Одной из них является формирование комплекса стандартизированной подготовки экспертов-предметников, включающей эффективное обучение проверке заданий с развернутыми ответами контрольных измерительных материалов (в частности, по информатике) с точным соблюдением централизованно разработанных критериев оценивания выполнения учащимися заданий с развернутыми ответами. Решение этой задачи – одно из условий обеспечения объективности и надежности результатов, полученных в ходе единого государственного экзамена.

Предлагаемые методические материалы (ММ) для подготовки экспертов, привлекаемых для проверки заданий с развернутыми ответами по информатике и ИКТ в рамках ЕГЭ, разработаны на основе открытых вариантов КИМ ЕГЭ и анализа опыта подготовки экспертов последних лет. Предлагаемые материалы учитывают спецификацию экзаменационной работы 2020 года.

Экзаменационная работа состоит из двух частей. Часть 1 содержит 23 задания с кратким ответом по всем основным разделам курса информатики. Задания части 2 направлены на проверку сформированности важнейших умений записи и анализа алгоритмов, предусмотренных требованиями к обязательному уровню подготовки по информатике учащихся общеобразовательных учреждений. В этой части также проверяются умения выпускников решать задачи на повышенном и высоком уровне сложности по теме «Технология программирования». Решения заданий части 2 работы записываются в развернутой форме и проверяются экспертами региональных предметных комиссий. За выполнение каждого задания дается определенное количество баллов, в зависимости от полноты и качества выполнения. Так, часть 2 включает 4 задания, что составляет почти 15% от общего количества заданий. При успешном их выполнении экзаменуемый может получить максимально 12 первичных баллов (т. е. примерно треть общего количества первичных баллов за всю работу). С другой стороны, эти задания являются самыми сложными и самыми трудоемкими: рекомендованное время их выполнения в 1,6 раза превосходит время, отводимое на выполнение первой части работы.

Извлечения из Методических рекомендаций Рособрнадзора по формированию и организации работы предметных комиссий субъекта Российской Федерации при проведении государственной итоговой аттестации по образовательным программам среднего общего образования

Во время работы экспертам запрещается:

- иметь при себе средства связи, фото-, аудио- и видеоаппаратуру;
- копировать и выносить из помещений, в которых работает ПК, экзаменационные работы, критерии оценивания, протоколы проверки экзаменационных работ;
- разглашать информацию, содержащуюся в указанных материалах.

Также запрещается:

- без уважительной причины покидать аудиторию;
- переговариваться с другими экспертами ПК, если речь не идет о консультировании с председателем ПК или с экспертом ПК, назначенным по решению председателя ПК, консультантом.

Если у эксперта возникают вопросы или проблемы, он должен обратиться к председателю ПК или лицу, назначенному председателем ПК консультантом.

2. Характеристика разных типов заданий с развернутым ответом и рекомендации по их оцениванию

2.1. Задания с развернутым ответом в структуре контрольных измерительных материалов для ЕГЭ по информатике и ИКТ

Фрагменты спецификации экзаменационной работы по информатике и ИКТ 2020 года, относящиеся к заданиям части 2

Задания части 2 направлены на проверку сформированности важнейших умений записи и анализа алгоритмов, предусмотренных требованиями к обязательному уровню подготовки по информатике учащихся средних общеобразовательных учреждений. Эти умения проверяются на повышенном и высоком уровне сложности. Также на высоком уровне сложности проверяются умения по теме «Технология программирования».

Распределение заданий с развернутым ответом по уровню сложности

В части 2 всего четыре задания, относящиеся к повышенному и высокому уровню сложности.

Если для заданий базового уровня предполагаемый процент выполнения 60%–80%, то для заданий повышенного и высокого уровня сложности требования более высокие. Для задания 24 повышенного уровня предполагаемый процент выполнения от 40 % до 60%, а для остальных заданий части 2 предполагаемый процент выполнения от 10% до 30%.

Система оценивания выполнения заданий с развернутым ответом и экзаменационной работы в целом

Ответы на задания части 2 проверяются и оцениваются экспертами (устанавливается соответствие ответов определенному перечню критериев).

Максимальное количество баллов, которое можно получить за выполнение заданий части 2, – 12 баллов.

Фрагмент обобщенного плана экзаменационной работы по информатике и ИКТ 2020 г.

№ п/п	Обозначение задания	Проверяемые элементы содержания и виды деятельности	Коды проверяемых элементов содержания по кодификатору	Коды требований к уровню подготовки выпускников по кодификатору	Уровень сложности задания	Максимальный балл за задание	Примерное время выполнения (мин)
	…	…	…	…	…	...	...
		Часть 3					
24	24	Умение прочесть фрагмент программы на языке программирования и исправить допущенные ошибки	1.7.2	1.1.4	П	3	30
25	25	Умения написать короткую (10–15 строк) простую программу обработки массива на языке программирования	1.6.3	1.1.5	В	2	30
26	26	Умение построить дерево игры по заданному алгоритму и обосновать выигрышную стратегию	1.5.2	1.1.3	В	3	30
27	27	Умения создавать собственные программы (30–50 строк) для решения задач средней сложности	1.7.3	1.1.5	В	4	55

2.2. Варианты заданий части 2 и критерии оценивания

Варианты задания 24 и критерии оценивания

Задание 24. Вариант 1

На обработку поступает натуральное число, не превышающее 10^9. Нужно написать программу, которая выводит на экран количество цифр этого числа, делящихся на 3. Если в числе нет цифр, делящихся на 3, на экран требуется вывести «NO». Программист написал программу неправильно. Ниже эта программа для Вашего удобства приведена на пяти языках программирования.

***Напоминание*:** 0 делится на любое натуральное число.

Бейсик

```
DIM N, DIGIT, COUNT AS LONG
INPUT N
COUNT = 1
WHILE N > 0
  DIGIT = N MOD 10
  IF DIGIT MOD 3 = 0 THEN
    COUNT = COUNT + DIGIT
  END IF
  N = N \ 10
WEND
IF COUNT = 0 THEN
  PRINT "NO"
ELSE
  PRINT COUNT
END IF
```

Python

```
N = int(input())
count = 1
while N > 0:
  digit = N % 10
  if digit % 3 == 0:
    count = count + digit
  N = N // 10
if count == 0:
  print("NO")
else:
  print(count)
```

Алгоритмический язык

```
алг
нач
  цел N, digit, count
  ввод N
  count := 1
  нц пока N > 0
    digit := mod(N,10)
    если mod(digit,3) = 0 то
      count := count + digit
    все
    N := div(N,10)
  кц
  если count = 0 то
    вывод "NO"
  иначе
    вывод count
  все
кон
```

Паскаль

```
var N, digit, count: longint;
begin
  readln(N);
  count := 1;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit mod 3 = 0  then
      count := count + digit;
    N := N div 10;
  end;
  if count = 0 then
    writeln('NO')
  else
    writeln(count)
end.
```

C++

```
#include <iostream>
using namespace std;

int main()
{
  int N, digit, count;
  cin >> N;
  count = 1;
  while (N > 0)
  {
    digit = N % 10;
    if (digit % 3 == 0)
      count = count + digit;
    N = N / 10;
  }
  if (count == 0)
    cout << "NO" << endl;
  else
    cout << count << endl;
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 145.

2. Приведите пример такого трёхзначного числа, при вводе которого программа выдаёт верный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

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

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. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Решение использует запись программы на Паскале. Допускается использование программы на любом из других четырёх языков.

1. Программа выведет число 1.

2. Программа выдаёт правильный ответ, например, для числа 140.

Замечание для проверяющего. Программа работает неправильно из-за неверного задания начального значения счётчика и неверного увеличения счётчика. Соответственно, программа будет работать верно, если в числе есть ровно один значащий 0 и нет других цифр, делящихся на 3.

3. В программе есть две ошибки.

Первая ошибка: неверное начальное значение счётчика.

Строка с ошибкой:

```
count := 1;
```

Верное исправление:

```
count := 0;
```

Вторая ошибка: неверное изменение счётчика.

Строка с ошибкой:

```
count := count + digit;
```

Верное исправление:

```
count := count + 1;
```

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **четыре** действия: 1) указать, что выведет программа при конкретном входном числе; 2) указать пример входного числа, при котором программа выдаёт верный ответ; 3) исправить первую ошибку; 4) исправить вторую ошибку. Для проверки правильности выполнения п. 2) нужно формально выполнить исходную (ошибочную) программу с входными данными, которые указал экзаменуемый, и убедиться в том, что результат, выданный программой, будет таким же, как и для правильной программы. Для действий 3) и 4) ошибка считается исправленной, если выполнены оба следующих условия: а) правильно указана строка с ошибкой; б) указан такой новый вариант строки, что при исправлении другой ошибки получается правильная программа	
Выполнены все четыре необходимых действия, и ни одна верная строка не указана в качестве ошибочной	3
Не выполнены условия, позволяющие поставить 3 балла. Имеет место одна из следующих ситуаций: а) выполнены три из четырёх необходимых действий. Ни одна верная строка не указана в качестве ошибочной; б) выполнены все четыре необходимых действия. Указано в качестве ошибочной не более одной верной строки	2
Не выполнены условия, позволяющие поставить 2 или 3 балла. Выполнены два из четырёх необходимых действий	1
Не выполнены условия, позволяющие поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

Задание 24. Вариант 2

На обработку поступает натуральное число, не превышающее 10^9. Нужно написать программу, которая выводит на экран сумму цифр числа, больших 5. Если в числе нет цифр, больших 5, требуется вывести на экран «NO». Программист написал программу неправильно. Ниже эта программа для Вашего удобства приведена на пяти языках программирования.

Бейсик

```
DIM N, DIGIT, SUM AS LONG
INPUT N
SUM = N MOD 10
WHILE N > 0
  DIGIT = N MOD 10
  IF DIGIT > 5 THEN
    SUM = DIGIT
  END IF
  N = N \ 10
WEND
IF SUM > 0 THEN
  PRINT SUM
ELSE
  PRINT "NO"
END IF
```

Python

```
N = int(input())
sum = N % 10
while N > 0:
  digit = N % 10
  if digit > 5:
    sum = digit
  N = N // 10
if sum > 0:
  print(sum)
else:
  print("NO")
```

Алгоритмический язык

```
алг
нач
  цел N, digit, sum
  ввод N
  sum := mod(N,10)
  нц пока N > 0
    digit := mod(N,10)
    если digit > 5 то
      sum := digit
    все
    N := div(N,10)
  кц
  если sum > 0 то
    вывод sum
  иначе
    вывод "NO"
  все
кон
```

Паскаль

```
var N, digit, sum: longint;
begin
  readln(N);
  sum := N mod 10;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit > 5 then
      sum := digit;
    N := N div 10;
  end;
  if sum > 0 then
    writeln(sum)
  else
    writeln('NO')
end.
```

C++

```
#include <iostream>
using namespace std;

int main()
{
  int N, digit, sum;
  cin >> N;
  sum = N % 10;
  while (N > 0)
  {
    digit = N % 10;
    if (digit > 5)
      sum = digit;
    N = N / 10;
  }
  if (sum > 0)
    cout << sum << endl;
  else
    cout << "NO" << endl;
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 748.

2. Приведите пример такого трёхзначного числа, при вводе которого программа выдаёт верный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

1) выпишите строку, в которой сделана ошибка;

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

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. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Решение использует запись программы на Паскале. Допускается использование программы на любом из других четырёх языков.

1. Программа выведет число 7.
2. Программа выдаёт правильный ответ, например, для числа 546.

Замечание для проверяющего. Программа работает неправильно из-за неверной начальной инициализации суммы и неверного увеличения суммы. Соответственно, программа будет работать верно, если в числе ровно одна цифра, большая 5, или таких цифр вообще нет и при этом число заканчивается на 0.

3. В программе есть две ошибки.

Первая ошибка: неверная инициализация суммы (переменная `sum`).

Строка с ошибкой:

```
sum := N mod 10;
```

Верное исправление:

```
sum := 0;
```

Вторая ошибка: неверное увеличение суммы.

Строка с ошибкой:

```
sum := digit;
```

Верное исправление:

```
sum := sum + digit;
```

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **четыре** действия: 1) указать, что выведет программа при конкретном входном числе; 2) указать пример входного числа, при котором программа выдаёт верный ответ; 3) исправить первую ошибку; 4) исправить вторую ошибку. Для проверки правильности выполнения п. 2) нужно формально выполнить исходную (ошибочную) программу с входными данными, которые указал экзаменуемый, и убедиться в том, что результат, выданный программой, будет таким же, как и для правильной программы. Для действий 3) и 4) ошибка считается исправленной, если выполнены оба следующих условия: а) правильно указана строка с ошибкой; б) указан такой новый вариант строки, что при исправлении другой ошибки получается правильная программа	
Выполнены все четыре необходимых действия, и ни одна верная строка не указана в качестве ошибочной	3
Не выполнены условия, позволяющие поставить 3 балла. Имеет место одна из следующих ситуаций: а) выполнены три из четырёх необходимых действий. Ни одна верная строка не указана в качестве ошибочной; б) выполнены все четыре необходимых действия. Указано в качестве ошибочной не более одной верной строки	2
Не выполнены условия, позволяющие поставить 2 или 3 балла. Выполнены два из четырёх необходимых действий	1
Не выполнены условия, позволяющие поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

Вариант задания 25 и критерии оценивания

Задание 25. Вариант 1

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от –10 000 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит максимальный элемент среди элементов массива, имеющих чётное значение, а затем заменяет каждый элемент с чётным значением на число, равное найденному максимуму. Гарантируется, что хотя бы один такой элемент в массиве есть. В качестве результата необходимо вывести изменённый массив, каждый элемент выводится с новой строчки.
Например, для исходного массива из шести элементов:
8
3
4
5
13
10
программа должна вывести следующий массив
10
3
10
5
13
10

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.

Бейсик	**Python**

`CONST N AS INTEGER = 30`
`DIM A (1 TO N) AS LONG`
`DIM I AS LONG,`
`    J AS LONG,`
`    K AS LONG`
 	
`FOR I = 1 TO N`
`    INPUT A(I)`
`NEXT I`
`...`
 	
`END`
	`# допускается также`
	`# использовать две`
	`# целочисленные переменные j и k`
	`a = []`
	`n = 30`
	`for i in range(0, n):`
	`    a.append(int(input()))`
	`...`

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.

C++

```
#include <iostream>
using namespace std;
const int N = 30;
int main() {
    long a[N];
    long i, j, k;
    for (i = 0; i < N; i++)
        cin >>  a[i];
    ...
    return 0;
}
```

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).

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

На языке Паскаль

```
k := -10000;
for i := 1 to N do
  if (a[i] mod 2 = 0) and (a[i] > k) then
    k := a[i];
for i := 1 to N do begin
  if (a[i] mod 2 = 0) then
    a[i] := k;
  writeln(a[i]);
end;
```

На Алгоритмическом языке

```
k := -10000
нц для i от 1 до N
  если mod(a[i], 2) = 0 и a[i] > k
  то
    k := a[i]
  все
кц
нц для i от 1 до N
  если mod(a[i], 2) = 0
  то
    a[i] := k
  все
  вывод a[i], нс
кц
```

На языке Бейсик

```
K = -10000
FOR I = 1 TO N
  IF A(I) MOD 2 = 0 AND A(I) > K THEN
    K = A(I)
  END IF
NEXT I
FOR I = 1 TO N
  IF A(I) MOD 2 = 0 THEN
    A(I) = K
  END IF
  PRINT A(I)
NEXT I
```

На языке C++

```
k = -10000;
for (i = 0; i < N; i++)
  if (a[i] % 2 == 0 && a[i] > k)
    k = a[i];
for (i = 0; i < N; i++) {
  if (a[i] % 2 == 0)
    a[i] = k;
  cout << a[i] << endl;
}
```

На языке Python

```
k = -10000
for i in range(0, n):
    if (a[i] % 2 == 0 and a[i] > k):
        k = a[i]
for i in range(0, n):
    if (a[i] % 2 == 0):
        a[i] = k
    print(a[i])
```

Указания по оцениванию	**Баллы**
Общие указания. 1. В алгоритме, записанном на языке программирования, допускается наличие отдельных синтаксических ошибок, не искажающих замысла автора программы. 2. Эффективность алгоритма не имеет значения и не оценивается. 3. Допускается запись алгоритма на языке программирования, отличном от языков, приведённых в условии. В этом случае должны использоваться переменные, аналогичные описанным в условии. Если язык программирования использует типизированные переменные, описания переменных должны быть аналогичны описаниям переменных на Алгоритмическом языке. Использование нетипизированных или необъявленных переменных возможно только в случае, если это допускается языком программирования; при этом количество переменных и их идентификаторы должны соответствовать условию задачи. 4. Допускается формат вывода массива, отличный от указанного, например в строчку	
Предложен правильный алгоритм, который изменяет исходный массив и выводит в качестве результата изменённый массив	2
Не выполнены условия, позволяющие поставить 2 балла. При этом предложено в целом верное решение, содержащее не более одной ошибки из числа следующих: 1) в цикле происходит выход за границу массива; 2) не инициализируется или неверно инициализируется максимум; 3) неверно осуществляется проверка чётности; 4) проверяется чётность не элемента массива, а его индекса; 5) в сравнении с максимумом перепутаны знаки «меньше» и «больше»; 6) сравнение с максимумом производится для индекса элемента массива, а не для его значения; 7) неверно составлено логическое условие (например, используется or вместо and); 8) исходный массив не изменяется; 9) изменяются не все требуемые элементы (например, только первый или последний из них); 10) отсутствует вывод ответа, или ответ выводится не полностью (например, только один элемент массива ввиду пропущенного цикла вывода элементов или операторных скобок); 11) используется переменная, не объявленная в разделе описания переменных; 12) не указано или неверно указано условие завершения цикла; 13) индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно	1
Ошибок, перечисленных в п. 1–13, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии в явном или неявном виде цикла поиска нужного элемента)	0
Максимальный балл	*2*

Задание 25. Вариант 2

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать целые значения от 0 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит сумму элементов массива, не меньших 99 и при этом не кратных 4, а затем заменяет каждый такой элемент на число, равное найденной сумме. Гарантируется, что хотя бы один такой элемент в массиве есть. В качестве результата необходимо вывести изменённый массив, каждый элемент выводится с новой строчки.

Например, для исходного массива из шести элементов:

101
128
6
105
4
18

программа должна вывести следующий массив:

206
128
6
206
4
18

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.

Бейсик

```
CONST N AS INTEGER = 30
DIM A (1 TO N) AS LONG
DIM I AS LONG,
    J AS LONG,
    K AS LONG

FOR I = 1 TO N
    INPUT A(I)
NEXT I
...

END
```

Python

```
# допускается также
# использовать две
# целочисленные переменные j и k
a = []
n = 30
for i in range(0, n):
    a.append(int(input()))
...
```

Алгоритмический язык

```
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
```

Паскаль

```
const
    N = 30;
var
    a: array [1..N] of longint;
    i, j, k: longint;
begin
    for i := 1 to N do
        readln(a[i]);
    ...

end.
```

C++

```
#include <iostream>
using namespace std;
const int N = 30;
int main() {
    long a[N];
    long i, j, k;
    for (i = 0; i < N; i++)
        cin >>  a[i];
    ...
    return 0;
}
```

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).

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

На языке Паскаль

```
k := 0;
for i := 1 to N do
  if (a[i] >= 99) and (a[i] mod 4 <> 0) then
    k := k + a[i];
for i := 1 to N do begin
  if (a[i] >= 99) and (a[i] mod 4 <> 0) then
    a[i] := k;
  writeln(a[i]);
end;
```

На Алгоритмическом языке

```
k := 0
нц для i от 1 до N
  если a[i] >= 99 и mod(a[i], 4) <> 0
  то
    k := k + a[i]
  все
кц
нц для i от 1 до N
  если a[i] >= 99 и mod(a[i], 4) <> 0
  то
    a[i] := k
  все
  вывод a[i], нс
кц
```

На языке Бейсик

```
K = 0
FOR I = 1 TO N
  IF A(I) >= 99 AND A(I) MOD 4 <> 0 THEN
    K = K + A(I)
  END IF
NEXT I
FOR I = 1 TO N
  IF A(I) >= 99 AND A(I) MOD 4 <> 0 THEN
    A(I) = K
  END IF
  PRINT A(I)
NEXT I
```

На языке C++

```
k = 0;
for (i = 0; i < N; i++)
  if (a[i] >= 99 && a[i] % 4 != 0)
    k = k + a[i];
for (i = 0; i < N; i++) {
  if (a[i] >= 99 && a[i] % 4 != 0)
    a[i] = k;
  cout << a[i] << endl;
}
```

На языке Python

```
k = 0
for i in range(0, n):
    if (a[i] >= 99 and a[i] % 4 != 0):
        k = k + a[i]
for i in range(0, n):
    if (a[i] >= 99 and a[i] % 4 != 0):
        a[i] = k
    print(a[i])
```

Указания по оцениванию	**Баллы**
Общие указания. 1. В алгоритме, записанном на языке программирования, допускается наличие отдельных синтаксических ошибок, не искажающих замысла автора программы. 2. Эффективность алгоритма не имеет значения и не оценивается. 3. Допускается запись алгоритма на языке программирования, отличном от языков, приведённых в условии. В этом случае должны использоваться переменные, аналогичные описанным в условии. Если язык программирования использует типизированные переменные, описания переменных должны быть аналогичны описаниям переменных на Алгоритмическом языке. Использование нетипизированных или необъявленных переменных возможно только в случае, если это допускается языком программирования; при этом количество переменных и их идентификаторы должны соответствовать условию задачи. 4. Допускается формат вывода массива, отличный от указанного, например в строчку	
Предложен правильный алгоритм, который изменяет исходный массив и выводит в качестве результата изменённый массив	2
Не выполнены условия, позволяющие поставить 2 балла. При этом предложено в целом верное решение, содержащее не более одной ошибки из числа следующих: 1) в цикле происходит выход за границу массива; 2) не инициализируется или неверно инициализируется сумма найденных элементов; 3) неверно осуществляется проверка делимости на 4; 4) проверяется делимость на 4 не элемента массива, а его индекса; 5) неверно осуществляется сравнение с 99 (например, используется знак «больше»); 6) сравнение с 99 производится для индекса элемента массива, а не для его значения; 7) неверно составлено логическое условие (например, используется or вместо and); 8) не вычисляется или неверно накапливается сумма найденных элементов; 9) исходный массив не изменяется или изменяется неверным образом; 10) отсутствует вывод ответа, или ответ выводится не полностью (например, только один элемент массива ввиду пропущенного цикла вывода элементов или операторных скобок); 11) используется переменная, не объявленная в разделе описания переменных; 12) не указано или неверно указано условие завершения цикла; 13) индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно	1
Ошибок, перечисленных в п. 1–13, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии в явном или неявном виде цикла подсчёта суммы нужных элементов)	0
Максимальный балл	*2*

Варианты задания 26 и критерии оценивания

Задание 26. Вариант 1.

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один камень или увеличить количество камней в куче в два раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 22. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 22 или больше камней.

В начальный момент в куче было S камней, $1 \leq S \leq 21$.

Говорят, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Выполните следующие задания. Во всех случаях обосновывайте свой ответ.

1. а) При каких значениях числа S Петя может выиграть первым ходом? Укажите все такие значения.

   б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

2. Укажите два значения S, при которых у Пети есть выигрышная стратегия, причем (а) Петя не может выиграть первым ходом, но (б) Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня.

   Для указанных значений S опишите выигрышную стратегию Пети.

3. Укажите такое значение S, при котором у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, но при этом у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

   Для указанного значения S опишите выигрышную стратегию Вани. Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы). На ребрах дерева указывайте, кто делает ход, в узлах – количество камней в позиции.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

(допускаются иные формулировки ответа, не искажающие его смысла)

1. а) Петя может выиграть первым ходом, если S =11, …, 21. Во всех случаях нужно удвоить количество камней в куче. При меньших значениях S за один ход нельзя получить кучу, в которой больше 21 камня.
   б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче будет S =10 камней. Тогда после первого хода Пети в куче будет 11 камней или 20 камней. В обоих случаях Ваня удваивает количество камней и выигрывает первым ходом.

2. Возможные значения S: 5 и 9. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 10 камней. Эта позиция разобрана в п. 1б. В ней игрок, который будет ходить (теперь это Ваня), выиграть не может, а его противник (то есть, Петя) следующим ходом выиграет.
3. Возможное значение S: 8. После первого хода Пети в куче будет 9 или 16 камней. Если в куче станет 16 камней, Ваня удвоит количество камней и выиграет первым ходом. Ситуация, когда в куче 9 камней, разобрана в п. 2. В этой ситуации игрок, который будет ходить (теперь это Ваня), выигрывает своим вторым ходом.
   В таблице изображено дерево возможных партий при описанной стратегии Вани. Заключительные позиции (в них выигрывает Ваня) подчеркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

	Положения после очередных ходов			
И.п.	1-й ход Пети (разобраны все ходы)	1-й ход Вани (только ход по стратегии)	2-й ход Пети (разобраны все ходы)	2-й ход Вани (только ход по стратегии)
8	8+1 =**9**	9+1=**10**	10+1=**11**	11*2=**22**
			10*2=**20**	20*2=**40**
	8*2=**16**	16*2=**32**		

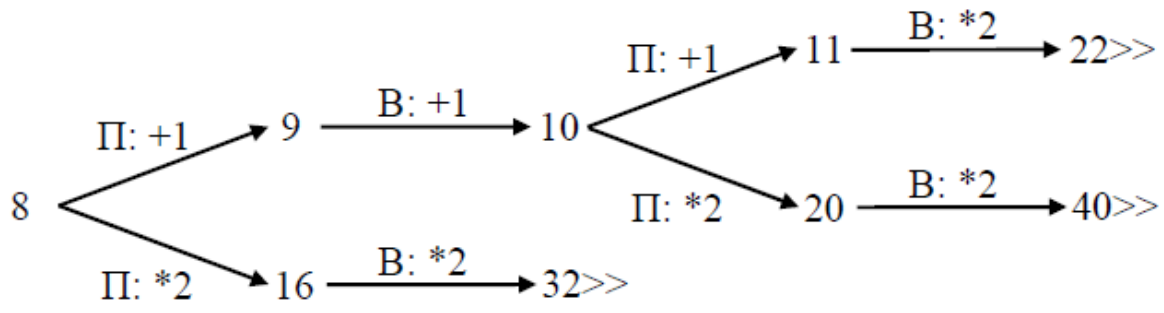


Рис.1. Дерево всех партий, возможных при Ваниной стратегии. Знаком >>обозначены позиции, в которых партия заканчивается.

Указания по оцениванию	Баллы
В задаче от ученика требуется выполнить 3 задания. Их трудность возрастает. Количество баллов в целом соответствует количеству выполненных заданий (подробнее см. ниже). Ошибка в решении, не искажающая основного замысла, например, арифметическая ошибка при вычислении количества камней в заключительной позиции, при оценке решения не учитывается. Первое задание считается выполненным полностью, если выполнены полностью оба пункта а) и б). Пункт а) считается выполненным полностью, если правильно указаны все позиции, в которых Петя выигрывает первым ходом и указано, каким должен быть первый ход. Пункт б) считается выполненным, если правильно указана позиция, в которой Ваня выигрывает первым ходом и описана стратегия Вани, т.е. показано, как Ваня может получить кучу, в которой содержится нужное количество камней при любом ходе Пети. Второе задание выполнено, если правильно указаны обе позиции, выигрышная для Пети и описана соответствующая стратегия Пети – так, как это написано в примере решения или другим способом, например, с помощью дерева всех возможных партий. Третье задание выполнено, если правильно указана позиция, выигрышная для Вани и построено дерево всех партий, возможных при Ваниной стратегии. Должно быть явно сказано, что в этом дереве в каждой позиции, где должен ходить Петя, разобраны все возможные ходы, а для позиций, где должен ходить Ваня – только ход, соответствующий стратегии, которую выбрал Ваня. Во всех случаях стратегии могут быть описаны так, как это сделано в примере решения или другим способом.	
Выполнены второе и третье задания. Первое задание выполнено полностью или частично. Здесь и далее допускаются арифметические ошибки, которые не искажают сути решения и не приводят к неправильному ответу (см. выше).	3
Не выполнены условия, позволяющие поставить 3 балла, и выполнено одно из следующих условий. 1. Задание 3 выполнено полностью. 2. Первое и второе задания выполнены полностью.	2
Не выполнены условия, позволяющие поставить 3 или 2 балла, и выполнено одно из следующих условий. 1. Первое задание выполнено полностью. 2. Во втором задании правильно указано одно из двух возможных значений S, и для этого значения указана и обоснована выигрышная стратегия Пети.	1
Не выполнено ни одно из условий, позволяющих поставить 3, 2 или 1 балл	0
Максимальный балл	*3*

Задание 26. Вариант 2.

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) **один** камень либо увеличить количество камней в куче в **два раза**. Например, пусть в одной куче 10 камней, а в другой 7 камней; такую позицию в игре будем обозначать (10, 7). Тогда за один ход можно получить любую из четырёх позиций: (11, 7), (20, 7), (10, 8), (10, 14). Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней.

Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 59.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, что в кучах всего будет 59 или больше камней.

Будем говорить, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника. Например, при начальных позициях (4, 28), (5, 27), (7, 26) выигрышная стратегия есть у Пети. Чтобы выиграть, ему достаточно удвоить количество камней во второй куче. В описание выигрышной стратегии **не следует** включать ходы играющего по этой стратегии игрока, не являющиеся для него безусловно выигрышными, т.е. не являющиеся выигрышными независимо от дальнейшей игры противника.

Задание 1. Для каждой из начальных позиций (4, 27), (6, 26) укажите, кто из игроков имеет выигрышную стратегию. В каждом случае опишите выигрышную стратегию.

Задание 2. Для каждой из начальных позиций (4, 26), (5, 26), (6, 25) укажите, кто из игроков имеет выигрышную стратегию. В каждом случае опишите выигрышную стратегию.

Задание 3. Для начальной позиции (5, 25) укажите, кто из игроков имеет выигрышную стратегию. Опишите выигрышную стратегию. Постройте дерево всех партий, возможных при указанной Вами выигрышной стратегии. Представьте дерево в виде рисунка или таблицы. Дерево не должно содержать партии, невозможные при реализации выигрывающим игроком своей выигрышной стратегии. Например, полное дерево игры не является верным ответом на это задание.

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Задание 1. В начальных позициях (4, 27), (6, 26) выигрышная стратегия есть у Вани. При начальной позиции (4, 27) после первого хода Пети может получиться одна из следующих четырёх позиций: (5, 27), (8, 27), (4, 28), (4, 54). Каждая из этих позиций содержит менее 59 камней. При этом из любой из этих позиций Ваня может получить позицию, содержащую не менее 59 камней, удвоив количество камней во второй куче. Для позиции (6, 26) после первого хода Пети может получиться одна из следующих четырёх позиций: (7, 26), (12, 26), (6, 27), (6, 52). Каждая из этих позиций содержит менее 59 камней. При этом из любой из этих позиций Ваня может получить позицию, содержащую не менее 59 камней, удвоив количество камней во второй куче. **Задание 2.** В начальных позициях (4, 26), (5, 26) и (6, 25) выигрышная стратегия есть у Пети. При начальной позиции (4, 26) он должен первым ходом получить позицию (4, 27), из начальных позиций (5, 26) и (6, 25) Петя после первого хода должен получить позицию (6, 26). Позиции (4, 27) и (6, 26) рассмотрены при разборе задания 1. В этих позициях выигрышная стратегия есть у игрока, который будет ходить вторым (теперь это Петя). Эта стратегия описана при разборе

задания 1.

Задание 3. В начальной позиции (5, 25) выигрышная стратегия есть у Вани. После первого хода Пети может возникнуть одна из четырёх позиций: (6, 25), (5, 26), (10, 25) и (5, 50). В позициях (10, 25) и (5, 50) Ваня может выиграть одним ходом, удвоив количество камней во второй куче. Позиции (6, 25) и (5, 26) были рассмотрены при разборе задания 2. В этих позициях у игрока, который должен сделать ход (теперь это Ваня), есть выигрышная стратегия. Эта стратегия описана при разборе задания 2.

В таблице изображено дерево возможных партий (и только их) при описанной стратегии Вани. Заключительные позиции (в них выигрывает Ваня) выделены жирным шрифтом.

<table>
<tr><td></td><td colspan="4">Положения после очередных ходов</td></tr>
<tr><td>Исходное положение</td><td>1-й ход Пети (разобраны все ходы, указана полученная позиция)</td><td>1-й ход Вани (только ход по стратегии, указана полученная позиция)</td><td>2-й ход Пети (разобраны все ходы, указана полученная позиция)</td><td>2-й ход Вани (только ход по стратегии, указана полученная позиция)</td></tr>
<tr><td rowspan="14">(5, 25)
Всего: 30</td><td rowspan="4">(5, 25+1) =
(5, 26)
Всего: 31</td><td rowspan="4">(5+1, 26) =
(6, 26)
Всего: 32</td><td>(6+1, 26) =
(7, 26)
Всего: 33</td><td>(7, 26*2) =
(7, 52)
Всего: 59</td></tr>
<tr><td>(6, 26+1) =
(6, 27)
Всего: 33</td><td>(6, 27*2) =
(6, 54)
Всего: 60</td></tr>
<tr><td>(6*2, 26) =
(12, 26)
Всего: 38</td><td>(12, 26*2) = (12, 52)
Всего: 64</td></tr>
<tr><td>(6, 26*2) =
(6, 52)
Всего: 58</td><td>(6, 52*2) =
(6, 104)
Всего: 110</td></tr>
<tr><td rowspan="4">(5+1, 25) =
(6, 25)
Всего: 31</td><td rowspan="4">(6, 25+1) =
(6, 26)
Всего: 32</td><td>(6+1, 26) =
(7, 26)
Всего: 33</td><td>(7, 26*2) =
(7, 52)
Всего: 59</td></tr>
<tr><td>(6, 26+1) =
(6, 27)
Всего: 33</td><td>(6, 27*2) =
(6, 54)
Всего: 60</td></tr>
<tr><td>(6*2, 26) =
(12, 26)
Всего: 38</td><td>(12, 26*2) = (12, 52)
Всего: 64</td></tr>
<tr><td>(6, 26*2) =
(6, 52)
Всего: 58</td><td>(6, 52*2) =
(6, 104)
Всего: 110</td></tr>
<tr><td>(5*2, 25) =
(10, 25)
Всего: 35</td><td>(10, 25*2) =
(10, 50)
Всего: 60</td><td></td><td></td></tr>
<tr><td>(5, 25*2) =
(5, 50) Всего:
55</td><td>(5, 50*2) =
(5, 100)
Всего: 105</td><td></td><td></td></tr>
</table>

Примечание для эксперта. Дерево всех партий может быть также изображено в виде ориентированного графа – так, как показано на рисунке, или другим способом. Например, вершины дерева, соответствующие одной и той же позиции, на рисунке могут быть «склеены». Важно, чтобы множество полных путей в графе находилось во взаимно однозначном соответствии со множеством партий, возможных при описанной в решении стратегии.

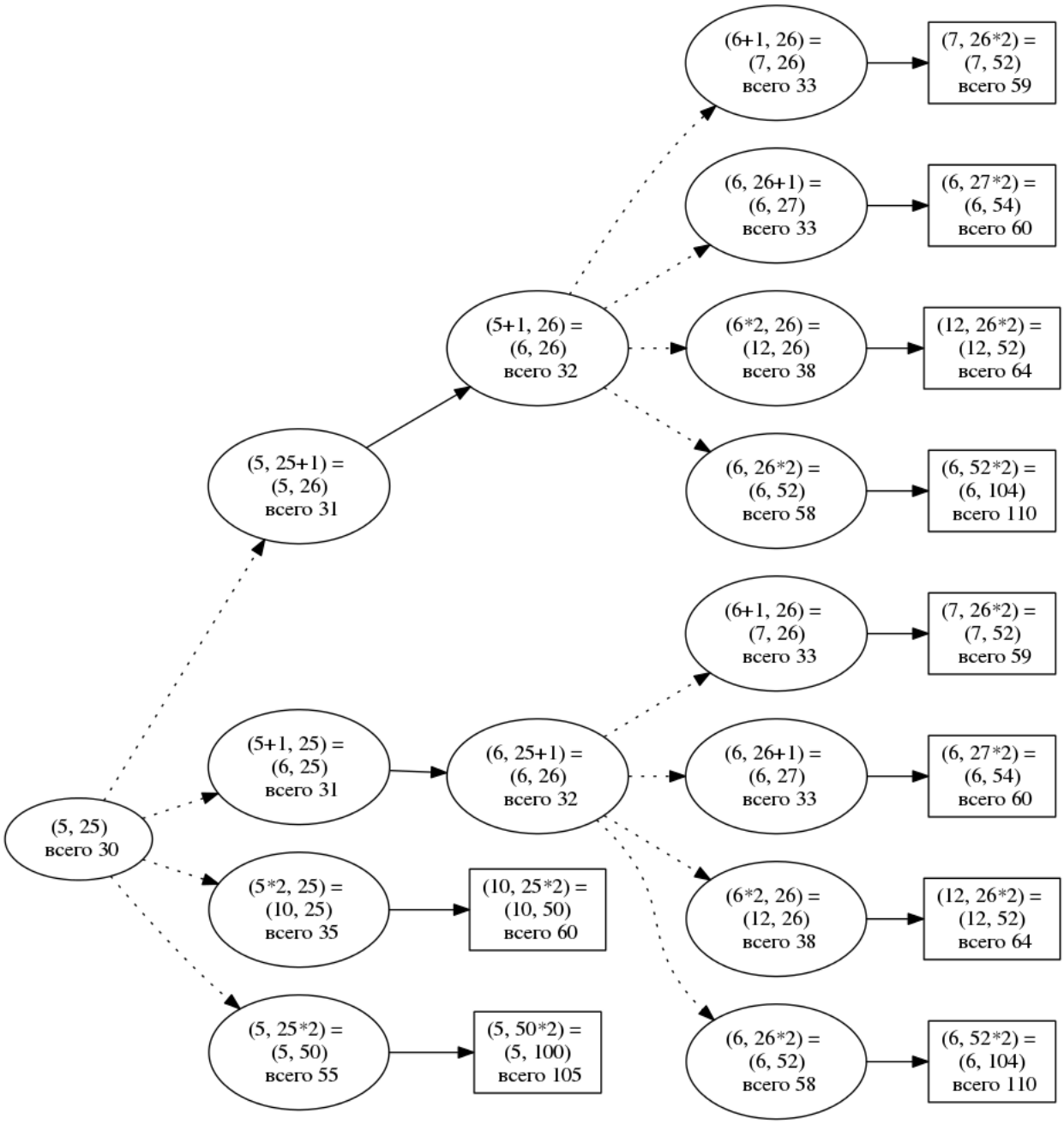


Рис. 1. Дерево всех партий, возможных при описанной стратегии Вани. Ходы Пети показаны пунктирными стрелками, ходы Вани показаны сплошными стрелками. Заключительные позиции обозначены прямоугольником

Примечание для эксперта. В некоторых позициях у Вани есть и другой способ выигрыша: например, в позиции (6, 52) можно добавить один камень в любую кучу. То, что это не указано, не является ошибкой. Экзаменуемый не должен указывать все возможные выигрышные стратегии

Указания по оцениванию	**Баллы**

В задаче требуется выполнить три задания. Количество баллов в целом соответствует количеству выполненных заданий (подробнее см. ниже). Ошибка в решении, не искажающая основного замысла и не приведшая к неверному ответу, например арифметическая ошибка при вычислении количества камней в заключительной позиции, при оценке решения не учитывается. Задание 1 выполнено, если для обоих начальных позиций верно указано, кто из игроков имеет выигрышную стратегию, и описана соответствующая выигрышная стратегия. Задание 2 выполнено, если для всех трёх начальных позиций верно указано, кто из игроков имеет выигрышную стратегию, и описана соответствующая выигрышная стратегия. Задание 3 выполнено, если верно указано, кто из игроков имеет выигрышную стратегию для данной начальной позиции, и построено дерево всех возможных при реализации данной стратегии выигрывающим игроком партий (и только их). Во всех случаях стратегии могут быть описаны так, как это сделано в примере решения, или другим способом	
Выполнены задания 1, 2 и 3	3
Не выполнены условия, позволяющие поставить 3 балла, и выполнено одно из следующих условий. 1. Выполнено задание 3. 2. Выполнены задания 1 и 2	2
Не выполнены условия, позволяющие поставить 3 или 2 балла, и выполнено одно из следующих условий. 1. Выполнено задание 1. 2. Выполнено задание 2	1
Не выполнено ни одно из условий, позволяющих поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

Вариант задания 27 и критерии оценивания

Задание 27. Вариант 1.

На вход программы поступает последовательность из N целых положительных чисел, все числа в последовательности различны. Рассматриваются все пары различных элементов последовательности, находящихся на расстоянии не меньше чем 3 (разница в индексах элементов пары должна быть 3 или более, порядок элементов в паре неважен). Необходимо определить количество таких пар, для которых произведение элементов делится на 13.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел N ($3 \le N \le 1000$). В каждой из последующих N строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна вывести одно число: количество пар элементов, находящихся в последовательности на расстоянии не меньше чем 3, в которых произведение элементов кратно 13.

Пример входных данных:

```
6
26
2
3
5
4
13
```

Пример выходных данных для приведённого выше примера входных данных:

```
5
```

Пояснение. Из шести заданных элементов с учётом допустимых расстояний между ними можно составить 6 произведений: 26·5, 26·4, 26·13, 2·4, 2·13, 3·13. Из них на 13 делятся 5 произведений.

Требуется написать эффективную по времени и памяти программу для решения описанной задачи.

Программа считается эффективной по времени, если при увеличении количества исходных чисел N в k раз время работы программы увеличивается не более чем в k раз.

Программа считается эффективной по памяти, если память, необходимая для хранения всех переменных программы, не превышает 1 килобайта и не увеличивается с ростом N.

Максимальная оценка за правильную (не содержащую синтаксических ошибок и дающую правильный ответ при любых допустимых входных данных) программу, эффективную по времени и памяти, – 4 балла.

Максимальная оценка за правильную программу, эффективную только по времени, – 3 балла.

Максимальная оценка за правильную программу, не удовлетворяющую требованиям эффективности, – 2 балла.

Вы можете сдать **одну** программу или **две** программы решения задачи (например, одна из программ может быть менее эффективна). Если Вы сдадите две программы, то каждая из них будет оцениваться независимо от другой, итоговой станет **бóльшая** из двух оценок.

Перед текстом программы обязательно кратко опишите алгоритм решения. Укажите использованный язык программирования и его версию.

Содержание верного ответа
(допускаются иные формулировки ответа, не искажающие его смысла)

Произведение двух чисел делится на 13, если хотя бы один из сомножителей делится на 13.

При вводе чисел можно подсчитывать количество чисел, кратных 13, не считая 3 последних. Обозначим их *n*13.
Примечание для проверяющего. Сами числа, кроме 3 последних, при этом можно не хранить.
Очередное считанное число будем рассматривать как возможный правый элемент искомой пары.
Если очередное считанное число делится на 13, то к ответу следует прибавить количество чисел до него, не считая 3 последних (включая считанное).
Если очередное считанное число на 13 не делится, то к ответу следует прибавить *n*13.
Чтобы построить программу, эффективную по памяти, заметим, что, поскольку при обработке очередного элемента входных данных используются значения, находящиеся на 3 элемента ранее, достаточно хранить только 3 последних элемента или информацию о них.
Ниже приведена реализующая описанный алгоритм программа на языке Паскаль (использована версия PascalABC)

Пример 1. Программа на языке Паскаль. Программа эффективна по времени и памяти

```
const s = 3; {требуемое расстояние между элементами}
var
  n: longint;
  a: array[1..s] of longint; {хранение последних s значений}
  a_: longint; {очередное значение}
  n13: longint; {количество делящихся на 13 элементов,
            не считая s последних}
  cnt: longint; {количество искомых пар}
  i, j: longint;
begin
  readln(n);
  {Ввод первых s чисел}
  for i:=1 to s do
    readln(a[i]);
  {Ввод остальных значений, подсчет искомых пар}
  cnt := 0;
  n13 := 0;
  for i := s + 1 to n do
  begin
    if a[1] mod 13 = 0 then
      n13 := n13 + 1;
    readln(a_);
    if a_ mod 13 = 0 then
      cnt := cnt + i - s
    else
      cnt := cnt + n13;
    {сдвигаем элементы вспомогательного массива влево}
    for j := 1 to s - 1 do
      a[j] := a[j + 1];
    a[s] := a_ {записываем текущий элемент в конец массива}
  end;
  writeln(cnt)
```

```
end.
```

Комментарии для проверяющего

1. При таком решении хранятся только последние 3 прочитанных элемента. Таким образом, используемая память не зависит от длины последовательности. Время обработки очередного числа фиксировано, т.е. не зависит от длины последовательности. Поэтому при увеличении длины последовательности в *k* раз время работы программы увеличивается не более чем в *k* раз. Таким образом, приведённая выше программа эффективна как по времени, так и по используемой памяти. Это решение оценивается в 4 балла.

В таких версиях Паскаля, как PascalABC или Delphi, тип longint может быть заменён на тип integer. В большинстве версий языков C\C++ также можно использовать тип int.

Программа может быть и ещё более эффективной, если на каждом шаге не сдвигать элементы вспомогательного массива, а записывать *i*-й считанный элемент в элемент с индексом `i mod 3` (Паскаль) или `i % 3` (Python), ведя нумерацию обоих индексов с нуля. Учёту подлежит элемент с этим же индексом (именно он находится на расстоянии *s* от *i*-го и будет заменён на него). Кроме того, при нумерации индексов элементов с нуля меняется одна из формул для подсчёта.

Такая программа на языке Python приведена ниже (пример 2).

Все подобные программы оцениваются, исходя из максимального балла – 4 (см. критерии).

Вместо последних 3 элементов можно хранить и 3 счётчика: количество делящихся на 13 среди всех считанных чисел, всех считанных чисел без последнего, всех считанных чисел без 2 последних – и также сдвигать их после очередного шага. Такая программа приведена на языке C++ (пример 3). В этом же примере вместо вспомогательного массива длиной 3 используются 3 переменные.

2. Возможно решение, основанное на описанных идеях, однако предварительно сохраняющее элементы последовательности в массив. Такое решение эффективно по времени, но неэффективно по памяти. Оно оценивается, исходя из максимального балла – 3 (см. критерии).

3. Решение, неэффективное ни по времени, ни по памяти, запоминает входную последовательность в массиве, после чего явно перебирает все возможные пары. Такое решение оценивается, исходя из максимального балла – 2 (см. критерии).

Пример 2. Программа на языке Python. Программа эффективна по времени и памяти

```
s = 3
a = [0]*s
n = int(input())
for i in range(s):
  a[i] = int(input())
cnt = 0
n13 = 0
for i in range(s, n):
    k = i % s
    if a[k] % 13 == 0:
        n13 = n13 + 1
    a_ = int(input())
    if   a_ % 13 == 0:
        cnt = cnt + i - s + 1
    else:
        cnt = cnt + n13
    a[i % s] = a_
print(cnt)
```

Пример 3. Программа на языке C++. Программа эффективна по времени и памяти

```
#include <iostream>
using namespace std;
int main()
{
  int s = 3; //требуемое расстояние между элементами
  int n;
  int n1 = 0, n2 = 0, n3 = 0; //хранение последних s счетчиков
  int a_;  // очередное значение
  int cnt; // количество искомых пар
  cin >> n;
  cnt = 0;
  for (int i = 0; i < n; ++i)
  {
    cin >> a_; // считано очередное значение
    if (i >= s)
    {
        if (a_ % 13 == 0)
           cnt += i - s + 1;
        else
           cnt += n3;
    }
    //сдвигаем элементы счетчиков
    n3 = n2;
    n2 = n1;
    //обновляем счетчик кратных 13
    if (a_ % 13 == 0)
        n1 += 1;
  }
  cout << cnt;
  return 0;
}
```

Указания по оцениванию	Баллы
Если в работе представлены две программы решения задачи, то каждая из них независимо оценивается по указанным ниже критериям, итоговой считается бóльшая из двух оценок. Описание алгоритма решения без программы оценивается в 0 баллов.	
Программа правильно работает для любых входных данных произвольного размера при условии исправления в ней не более трёх синтаксических ошибок из приведённого ниже списка допустимых ошибок. Используемая память не зависит от количества прочитанных чисел, а время работы пропорционально этому количеству. Допускается наличие в тексте программы до трёх синтаксических ошибок одного из следующих видов: 1) пропущен или неверно указан знак пунктуации; 2) неверно написано, пропущено или написано лишнее зарезервированное слово языка программирования; 3) не описана или неверно описана переменная; 4) применяется операция, не допустимая для соответствующего типа данных. Если одна и та же ошибка встречается несколько раз, это считается за одну ошибку	4
Не выполнены условия, позволяющие поставить 4 балла. Программа работает правильно для любых входных данных произвольного размера при условии исправления в ней не более пяти синтаксических ошибок из приведённого в критериях на 4 балла списка и не более одной ошибки из приведённого ниже списка содержательных ошибок. Время работы пропорционально количеству введённых чисел. Допускается наличие не более **одной** содержательной (не являющейся синтаксической) ошибки следующих видов: 1) допущена ошибка при вводе данных, например не считывается значение *N*, или числа могут быть считаны, только если будут записаны в одной строке через пробел; 2) неверная инициализация или её отсутствие там, где она необходима; 3) используется неверный тип данных; 4) использована одна переменная (или константа) вместо другой; 5) используется один знак операции вместо другого; 6) используется одно зарезервированное слово языка программирования вместо другого; 7) неверно используется условный оператор, например else относится не к тому условию; 8) отсутствует вывод ответа, или выводится значение не той переменной; 9) выход за границу массива; 10) неверно расставлены операторные скобки. 3 балла также ставится за программу, в которой нет содержательных ошибок, но используемая память зависит от количества прочитанных чисел (например, входные данные запоминаются в массиве, контейнере STL в C++ или другой аналогичной структуре данных)	3

Пример 4. Программа на языке Паскаль. Программа эффективна по времени и неэффективна по памяти	

```
const s = 3; {требуемое расстояние между элементами}
var
  n: longint;
  a: array[1..1000] of longint;
  n13: longint;
{количество делящихся на 13 элементов,  не считая s
последних}
  cnt: longint; {количество искомых пар}
  i, j: longint;
begin
  readln(n);
  {Ввод первых s чисел}
  for i:=1 to s do
    readln(a[i]);
  {Ввод остальных значений, подсчет искомых пар}
  cnt := 0;
  n13 := 0;
  for i := s + 1 to n do
  begin
    readln(a[i]);
    if a[i - s] mod 13 = 0 then
      n13 := n13 + 1;
    if a[i] mod 13 = 0 then
      cnt := cnt + i - s
    else
      cnt := cnt + n13;
  end;
  writeln(cnt)
end.
```

Не выполнены условия, позволяющие поставить 3 или 4 балла.
 	
Программа работает верно, эффективно по времени при условии исправления не более трёх содержательных ошибок, описанных в критериях на 3 балла, и не более девяти синтаксических ошибок, указанных в критериях на 4 балла.
 	
2 балла также ставится за корректное переборное решение, в котором все числа сохраняются в массиве (или другой аналогичной структуре), рассматриваются все возможные пары и подсчитывается количество подходящих произведений с учётом допустимого расстояния между ними. Пример фрагмента соответствующей программы на языке Паскаль:
`cnt := 0;`
`for i := 1 to N - s do`
`  for j := i + s to N do`
`    if a[i] * a[j] mod 13 = 0 then`
`      cnt := cnt + 1;`
`writeln(cnt)`
 	
Не допускается выставление 2 баллов за реализацию переборного алгоритма, содержащего любую логическую ошибку, например ошибку, приводящую к выходу индексов за границы массива, или ошибку, когда учитываются произведения вида `a[i]*a[i]`, или пары считаются дважды, или неверно учитывается расстояние между индексами элементов пары
	2

Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. При этом в программе должны присутствовать два обязательных элемента, возможно, реализованных с ошибками: 1) проверка делимости (в явной или неявной форме) элементов входной последовательности на заданное число; 2) проверка или учёт того, что расстояние между элементами искомой пары должно быть не меньше заданного	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл	*4*

Задание 27. Вариант 2

На вход программы поступает последовательность из n целых положительных чисел. Рассматриваются все пары элементов последовательности a_i и a_j, такие что $i < j$ и $a_i > a_j$ (первый элемент пары больше второго, i и j – порядковые номера чисел в последовательности входных данных). Среди пар, удовлетворяющих этому условию, необходимо найти и напечатать пару с максимальной суммой элементов, которая делится на m = 120. Если среди найденных пар максимальную сумму имеют несколько, то можно напечатать любую из них.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел n ($2 \le n \le 12\,000$). В каждой из последующих n строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна напечатать элементы искомой пары. Если таких пар несколько, можно вывести любую из них. Гарантируется, что хотя бы одна такая пара в последовательности есть.

Пример входных данных:

```
6
60
140
61
100
300
59
```

Пример выходных данных для приведённого выше примера входных данных:

```
140 100
```

Пояснение. Из шести заданных чисел можно составить 3 пары, сумма элементов которых делится на m=120: 60+300, 140+100 и 61+59. Во второй и третьей из этих пар первый элемент больше второго, но во второй паре сумма больше.

Требуется написать эффективную по времени и памяти программу для решения описанной задачи.

Программа считается эффективной по времени, если при одновременном увеличении количества элементов последовательности n и параметра m в k раз, время работы программы увеличивается не более чем в k раз. Программа считается эффективной по памяти, если память, необходимая для хранения всех переменных программы, не превышает 4 килобайта и не увеличивается с ростом n.

Максимальная оценка за правильную (не содержащую синтаксических ошибок и дающую правильный ответ при любых допустимых входных данных) программу, эффективную по времени и памяти, – 4 балла.

Максимальная оценка за правильную программу, возможно, неэффективную по памяти или время выполнения которой существенно зависит от величины m, – 3 балла.

Максимальная оценка за правильную программу, не удовлетворяющую требованиям эффективности, – 2 балла.

Вы можете сдать **одну** программу или **две** программы решения задачи (например, одна из программ может быть менее эффективна). Если Вы сдадите две программы, то каждая из них будет оцениваться независимо от другой, итоговой станет **бóльшая** из двух оценок.

Перед текстом программы обязательно кратко опишите алгоритм решения. Укажите использованный язык программирования и его версию.

Содержание верного ответа (допускаются иные формулировки ответа, не искажающие его смысла)
Сумма a_i и a_j делится на m, если сумма остатков этих чисел от деления на m равна 0 или m. Для каждого из остатков от деления на m среди уже просмотренных элементов будем хранить максимальное число, имеющее соответствующий остаток от деления на m. Для этого будем использовать массив r длиной m, изначально с элементами, равными 0. Все считанные значения при этом можно не хранить. Очередное считанное число a будем рассматривать как возможный правый элемент искомой пары. Пусть остаток от деления a на m равен p. Тогда если $r[m–p] > 0$, то сумма a и $r[m–p]$ делится на m, и, при условии $r[m–p] > a$, эта пара – кандидат для ответа. Если их сумма больше предыдущего ответа, то заменим его. При этом если остаток от деления a на m равен 0, то рассматривать надо пару a и $r[0]$. По окончании обработки элемента a необходимо обновить элемент $r[p]$ значением a, если $a > r[p]$. Ниже приведена реализующая описанный алгоритм программа на языке Паскаль (использована версия PascalABC)

Пример 1. Программа на языке Паскаль. Программа эффективна по времени и памяти

```
const m = 120; {количество различных остатков}
var
  {хранение максимального значения для каждого из остатков}
  r: array[0..m-1] of integer;
  n, a, i, p, left, right: integer;
begin
  readln(n);
  {обнуление массива r}
  for i := 0 to m - 1 do
    r[i] := 0;
  {обнуление переменных для записи ответа}
  left := 0; right := 0;
  {ввод значений, поиск искомой пары}
  for i := 1 to n do
  begin
    readln(a); {считываем очередное значение}
    p := a mod m;
    if p = 0 then
    begin
      if (r[0] > a) and (r[0] + a > left + right) then
      begin
        left := r[0]; right := a {обновление ответа}
      end
    end
    else
    begin
      if (r[m - p] > a) and (r[m - p] + a > left + right) then
      begin
        left := r[m - p]; right := a {обновление ответа}
      end
    end;
    {обновление элемента r для соответствующего остатка}
    if a > r[p] then r[p] := a
  end;
  writeln(left, ' ',right)
end.
```

Комментарии для проверяющего

1. При таком решении хранится только очередной прочитанный элемент и информация о максимальных значениях, имеющих различные остатки от деления на m (на их хранение будет потрачено не более $4m$ байт памяти, а на все переменные в целом – менее 4 килобайт). Таким образом, используемая память не зависит от длины последовательности. Время обработки очередного числа фиксированно, т.е. не зависит от длины последовательности и даже от величины m. Поэтому при увеличении длины последовательности в k раз время работы программы увеличивается не более чем в k раз. Таким образом, приведённая выше программа эффективна как по времени, так и по используемой памяти. Это решение оценивается 4 баллами.

Программа может не рассматривать отдельно случай $p = 0$, а учесть оба случая с помощью одной формулы: `(m - p) mod m`. Такой вариант реализации показан в примере 2 программы на языке Python. Может быть реализовано решение с заменой $p = 0$

на $p = m$. Такая программа на языке C++ приведена ниже (пример 3).

Все подобные программы оцениваются, исходя из максимального балла – 4 (см. критерии).

2. Возможно решение, основанное на описанных идеях, однако предварительно сохраняющее элементы последовательности в массив или другие структуры данных. Такое решение эффективно по времени, но неэффективно по памяти. Оно оценивается, исходя из максимального балла – 3 (см. критерии). Кроме того, возможен неэффективный способ определения, какой именно остаток от деления нас интересует, например с помощью цикла, выполняющегося до m раз, или с помощью m условных операторов:

```
  if p = 0 and a > r[0] then r[0] = a;
  if p = 1 and a > r[1] then r[1] = a;
```

и т.д.

Такое решение работает в m раз дольше и оценивается, исходя из максимального балла – 3 (см. критерии).

3. Решение, неэффективное ни по времени, ни по памяти, запоминает входную последовательность в массиве, после чего явно перебирает все возможные пары. Такое решение оценивается, исходя из максимального балла – 2 (см. критерии)

Пример 2. Программа на языке Python 3. Программа эффективна по времени и памяти

```
m = 120
# создание массива для максимальных значений
# для каждого из остатков
r = [0] * m
# обнуление переменных для записи ответа
left = 0
right = 0
# ввод количества элементов
n = int(input())
# ввод значений, поиск искомой пары
for i in range(n):
    a = int(input())
    p = a % m;
    if r[(m - p) % m] > a and r[(m - p) % m] + a > left + right:
        #обновление ответа
        left = r[(m - p) % m]
        right = a;
    # обновление элемента r для соответствующего остатка
    if a > r[p]:
        r[p] = a
print(left, right)
```

Пример 3. Программа на языке C++. Программа эффективна по времени и памяти

```
#include <iostream>
using namespace std;

int main()
{
  int n, a, p, left, right;
  int r[120];
  int m = 120;
  cin >> n;
  //обнуление массива r
  for (int i = 0; i < m; ++i)
    r[i] = 0;
  //обнуление переменных для записи ответа
  left = 0; right = 0;
  // ввод значений, поиск искомой пары
  for (int i = 0; i < n; ++i)
  {
    cin >> a; //считываем очередное значение
    p = a % m;
    if (p == 0) p = m;
    if (r[m - p] > a && r[m - p] + a > left + right)
    {
        left = r[m - p]; right = a; //обновление ответа
    }
    // обновление элемента r для соответствующего остатка
    if (p < m)
    {
    if (a > r[p]) r[p] = a;
    }
    else if (a > r[0]) r[0] = a;
  }
  cout << left << ' ' << right;
}
```

Указания по оцениванию	Баллы
Если в работе представлены две программы решения задачи, то каждая из них независимо оценивается по указанным ниже критериям, итоговой считается бóльшая из двух оценок. Описание алгоритма решения без программы оценивается в 0 баллов	
Программа правильно работает для любых входных данных произвольного размера при условии исправления в ней не более трёх синтаксических ошибок из приведённого ниже списка допустимых ошибок. Используемая память не зависит от количества прочитанных чисел, а время работы пропорционально этому количеству. Допускается наличие в тексте программы до трёх синтаксических ошибок одного из следующих видов: 1) пропущен или неверно указан знак пунктуации; 2) неверно написано, пропущено или написано лишнее зарезервированное слово языка программирования; 3) не описана или неверно описана переменная; 4) применяется операция, не допустимая для соответствующего типа данных. Если одна и та же ошибка встречается несколько раз, это считается за одну ошибку	4
Не выполнены условия, позволяющие поставить 4 балла. Программа работает правильно для любых входных данных произвольного размера при условии исправления в ней не более пяти синтаксических ошибок из приведённого в критериях на 4 балла списка и не более одной ошибки из приведённого ниже списка содержательных ошибок. Время работы пропорционально количеству введённых чисел, но может существенно зависеть от m (см. комментарий к эффективному решению задачи). Допускается наличие не более **одной** содержательной ошибки следующих видов: 1) допущена ошибка при вводе данных (например, не считывается значение N, или числа могут быть считаны, только если будут записаны в одной строке через пробел); 2) неверная инициализация или её отсутствие там, где она необходима; 3) используется неверный тип данных, при этом ошибка не является синтаксической; 4) не более одного раза использована одна переменная (или константа) вместо другой, или не более одного раза используется один знак операции вместо другого, или не более одного раза используется одно зарезервированное слово языка программирования вместо другого, при этом ошибка не является синтаксической; 5) служебное слово else относится не к тому if, к какому следует; 6) отсутствует вывод ответа, или выводится значение не тех переменных; 7) выход за границу массива (в частности, при обращении к m-му элементу массива с индексами от 0 до m–1, даже если он существует, но не заполнен нужным значением); 8) не выполнен или неверно выполнен учёт элементов, остаток от деления которых на m равен 0. 3 балла также ставится за программу, в которой нет содержательных ошибок, но используемая память зависит от количества прочитанных чисел (например, входные данные запоминаются в массиве, контейнере STL в C++ или другой	3

аналогичной структуре данных)	

Не выполнены условия, позволяющие поставить 3 или 4 балла.
Программа работает верно, эффективно по времени при условии исправления не более трёх содержательных ошибок, описанных в критериях на 3 балла и аналогичных им, и не более девяти синтаксических ошибок, указанных в критериях на 4 балла. При этом в программе могут быть опущены с помощью многоточия однотипные действия, связанные с рассмотрением каждого из остатков от деления на *m*.
 	
Не допускается выставление 2 баллов за программу, если в ней учитываются суммы вида a[i]+a[i] (в том числе в алгоритме без хранения элементов последовательности).
2 балла также ставится за корректное переборное решение, в котором все числа сохраняются в массиве (или другой аналогичной структуре) и рассматриваются все возможные пары. Пример фрагмента соответствующей программы на языке Паскаль:
`left := 0; right := 0;`
`for i := 1 to N - 1 do`
`  for j := i + 1 to N do`
`    if (a[i] > a[j]) and ((a[i] + a[j]) mod m = 0) then`
`      if a[i] + a[j] > left + right then`
`      begin`
`        left := a[i];`
`        right := a[j]`
`      end;`
В цикле реализации переборного алгоритма не допускаются выход индексов за границы массива, а также любые логические ошибки
	2

Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. При этом программа описывает в целом правильный алгоритм (эффективный или нет) и в ней присутствует не менее двух элементов решения из перечисленных ниже, возможно, реализованных с ошибками: • учитывается условие `a[i] > a[j]`; • проверяется делимость суммы на *m*; • ищется пара с максимальной суммой. В случае, если в любом решении содержится не более одного из указанных элементов, программа оценивается в 0 баллов	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл	*4*

Инструкции и памятки по проверке работ

Задание 24

1. При проверке правильности выполнения пунктов задания, связанных с указанием входных и выходных данных, т.е. ответов на вопросы «Что выведет программа программа?», «Для какого числа программа работает верно?», «Для какого числа программа работает неверно?», следует учитывать, что в задании не требуется каких-либо обоснований ответов на эти вопросы.
2. В случае возникновения сомнений, будет ли исходная программа с ошибками при предложенном участником экзамена исходном значении давать правильный ответ, следует провести на бумаге пошаговую трассировку программы для этого значения.
3. При проверке заданий, для которых критериями предусмотрено снижение оценки за указание верной строки в качестве строки с ошибкой нужно подсчитать количество таких строк.

Задание 25

1. Определите, на каком языке записан алгоритм, и при необходимости наведите справки о синтаксисе избранного экзаменуемым языка программирования.
2. Сравните описание алгоритма с имеющимися образцами. В случае совпадения – оцените в соответствии с рекомендациями.
3. Если описание алгоритма не совпадает с образцами, а ошибки в описании алгоритма с первого взгляда не видны, осуществите формальное исполнение алгоритма с тестовыми примерами исходных данных. Длину тестового массива следует сократить до 4–6 элементов. При тестах необходимо особенно тщательно проверять «критические» случаи, например, когда элементы массива одинаковы или изначально упорядочены. Оцените правильность полученных результатов.
4. При оценке алгоритма отметьте все ошибки, упомянутые в критериях оценивания. В случае, если таких ошибок более двух, сразу снижайте оценку до 0 баллов.
5. Не допускайте произвольного ужесточения критериев оценивания. Не вводите дополнительных ограничений. Не оценивайте синтаксические ошибки, «стиль» программирования, аккуратность записи, наличие комментариев, отступов и прочие важные, но данным заданием не проверяемые вещи.
6. Не забывайте, что эффективность алгоритмов в данной задаче **не оценивается, поэтому не следует снижать оценку за решение, в котором используются лишние просмотры массива**.

Задание 26

1. Сравните последовательно ответы экзаменуемого на вопросы 1а), 1б), 2 и 3 с эталонными.
2. Проверьте наличие и корректность обоснования решений. Стратегия правильного решения в работе экзаменуемого может быть представлена в варианте, отличном от «образцового», но основные положения должны совпадать с предложенными. Графическое оформление дерева игры также может быть различным. Обратите внимание, что для выигрывающего игрока должны быть рассмотрены только выигрышные ходы (стратегия), и только они, для проигрывающего – должны быть рассмотрены все варианты его ответов.

Задание 27

1. Определите язык программирования, на котором написана программа. При необходимости эксперт может воспользоваться справочной литературой.
2. Подсчитайте число синтаксических ошибок в данной программе. Систематически встречающаяся ошибка считается за одну. Так, например, если вместо круглых скобок ученик в записи условий использовал везде квадратные, то это считается за одну ошибку.
3. Рассмотрите реализацию каждой части алгоритма. Определите количество алгоритмических ошибок и количество ошибок в реализации алгоритмов.
4. Если ошибок мало и программа оценивается из 3–4 баллов, то оцените эффективность предложенного решения.

Описание ситуаций сложных для оценивания

Задание 24

Ситуация	Пояснение
Учащийся привел верные примеры входных данных, когда программа работает правильно (неправильно), но не обосновал свой ответ.	Поскольку в задании обоснования не требовалось, за его отсутствие оценка не снижается.
Учащийся привел верные примеры входных данных, когда программа работает правильно (неправильно), обосновал свой ответ, но допустил ошибки в обосновании.	Оценка не снижается.
Доработка программы предложена верно, но в виде фраз на естественном языке, а не на языке программирования.	Если описание содержит чёткие и недвусмысленные указания на необходимые исправления, такое описание следует рассматривать наравне с написанием исправленной строки.

Задание 25

Сложностью при проверке этого задания является разнообразие языков программирования, на которых могут записываться решения: от классического Бейсика с метками и оператором GOTO до C++, PHP и Perl. Это обстоятельство не должно смущать, так как простота формулировки задания предполагает и простую его реализацию на языке программирования, использование небольшого набора базовых операторов, достаточно инвариантных. Задачей экзаменатора при анализе решения **задания** не является проверка знания экзаменуемым синтаксиса языка, а только знания им алгоритмов и умения записать их с определенной степенью формализации.

Таким образом, за синтаксические ошибки в решениях задания оценка не снижается, если только они не ведут к невозможности анализа алгорима экспертом.

Многие экзаменуемые пишут программы, предусматривающие избыточное количество просмотров массива. За избыточные просмотры оценка не снижается

Задание 26

При проверке решений задания 26 следует учитывать, что правильная стратегия может быть представлена экзаменуемым в самых различных формах (схема, таблица, словесное описание и т.д.). Если решение представлено в виде изображения дерева, то оно может быть схематически изображено способом, отличным от схемы в критериях. Например, корень (исходная позиция) может быть расположен не только слева (как в эталонном ответе), но и сверху и вообще произвольно.

За арифметические ошибки, не приведшие к неверному ответу, оценка не снижается.

Задание 27

Программа может быть написана на языке программирования, отличном от Бейсика, Си или Паскаля, например, на языках C#, Java, Perl, PHP и т.д. Такая программа, тем не менее, должна быть оценена. Это же относится и к особенностям современных версий языков Паскаль, Си и Бейсик. При необходимости эксперт может воспользоваться справочной литературой.

Указания по оцениванию развернутых ответов участников ЕГЭ для эксперта, проверяющего ответы на задания с развернутым ответом по ИНФОРМАТИКЕ и ИКТ в 2020 г.

(документ предоставляется эксперту при проведении оценивания экзаменационных работ)

1. Общие рекомендации

При проверке правильности решений следует учитывать, что образцы решений, приведенные в указаниях по проверке отдельных задач, не являются единственно возможными. Таким образом, то, что решение ученика не совпадает с решением в образце, само по себе не означает, что решение экзаменуемого неверное. При оценке следует руководствоваться указаниями по оцениванию, примеры решений являются лишь подспорьем для проверяющего.

В указаниях по оцениванию для каждой задачи подробно описаны критерии выставления того или иного балла. Ниже даны рекомендации по проверке отдельных задач и рассмотрены некоторые сложные случаи.

Результаты оценивания переносятся в «Протокол проверки развернутых ответов», при этом баллы по каждому заданию переносятся в колонку, название которой соответствует номеру задания (см. Рисунок 1):

– баллы по заданию ***24*** переносятся в колонку ***24*** протокола;
– баллы по заданию ***25*** переносятся в колонку ***25*** протокола;
– баллы по заданию ***26*** переносятся в колонку ***26*** протокола;
– баллы по заданию ***27*** переносятся в колонку ***27*** протокола.

Рисунок 1. Протокол проверки развернутых ответов. Образец

Протокол проверки развернутых ответов

Регион 99 | Код предмета 5 | Название предмета Информатика и ИКТ (дата экзамена) | Номер протокола 1000005

ФИО эксперта Фамилия И.О. | Код эксперта 000002

Примечание

Образец заполнения 1 2 3 4 5 6 7 8 9 0 X

№	Код бланка	Позиции оценивания																			
		24	25	26	27																
1	2820500339593																				
2																					
3																					
4																					
5																					
6																					
7																					
8																					
9																					
10																					

Дата проверки ☐☐ - ☐☐ - ☐☐ Подпись эксперта

Баллы выставляются в бланк оценивания гелевой черной ручкой.

Внимание! При выставлении баллов за выполнение задания в Протокол проверки развернутых ответов следует иметь в виду, что **если ответ отсутствует** (нет никаких записей, свидетельствующих о том, что экзаменуемый приступал к выполнению задания), то в протокол проставляется «**X**», а не «0».

Если экзаменуемый написал хотя бы номер выполняемого задания, то считается, что он приступил к выполнению задания, и символ «**X**» не может быть проставлен в протоколе.

2. Рекомендации по отдельным заданиям

Задание 24

При проверке ответа на задание 24 следует последовательно проверить, насколько выполнены следующие **четыре** действия:

1) указано, что выведет программа при конкретной входной последовательности;
2) указан пример последовательности, при которой программа работает правильно;
3) исправлена первая ошибка;
4) исправлена вторая ошибка.

ВНИМАНИЕ! Экзаменуемому достаточно указать один пример таких входных данных. Указывать полное описание всех таких данных не требуется. Описания, приведенные в критериях по оцениванию (см. «*Замечания для проверяющего*»), предназначены только для проверяющего. Если при выполнении второго задания экзаменуемый указал несколько вариантов входных данных, то задание считается выполненным, если для ВСЕХ указанных входных данных программа дает нужный результат.

При проверке каждого из заданий на исправление ошибок (третье и четвертое действия) следует убедиться, что:

а) каждое внесенное исправление затрагивает только одну строку в программе;
б) указан такой новый вариант строки, что при исправлении другой ошибки получается правильная программа.

Каждое из этих действий оценивается отдельно. **Обратите внимание**: исправления, указанные экзаменуемым, могут отличаться от приведенных в критериях, но тем не менее быть верными.

Ситуации, сложные для оценивания

№	Типичная проблемная ситуация	Решение
1	Участником экзамена указаны в качестве ошибочных верные строки, содержащие, по его мнению, синтаксические ошибки или неверный тип переменных	Констатируется, что верная строка указана учащимся как ошибочная, и этот факт учитывается при оценивании по критериям
2	При выполнении первого и/или второго элемента задания экзаменуемый привел несколько числовых ответов, часть из которых верна, а часть нет	Соответствующий пункт задания признается выполненным неверно
3	В исправленной строке допущены синтаксические ошибки, не искажающие замысла автора	Соответствующий пункт задания признается выполненным верно
4	Строка с ошибкой не выписана, как требуется в задании, а указана иным образом, например номером	Если способ, которым обучающийся указал строку, позволяет ее однозначно идентифицировать, приравниваем эту ситуацию к выписыванию строки.

5	Обучающийся заново переписал текст программы или ее фрагменты из нескольких строк, внеся исправления	Если исправления касались только строк с ошибками, засчитываем этот элемент как выполненный. В противном случае считаем, что элемент не выполнен
6	Обучающийся сумел верно исправить ошибку альтернативным образом, изменив одну или две строки, но не те, которые указаны в критериях оценивания.	Соответствующий пункт задания признается выполненным верно.

Задание 25

1. Определите, на каком языке записан алгоритм в проверяемой работе, и при необходимости наведите справки (у эксперта-консультанта или с использованием рабочего места с выходом в интернет) о синтаксисе избранного экзаменуемым языка программирования.

2. Сравните описание алгоритма с имеющимися образцами и в случае совпадения оцените его в соответствии с рекомендациями.

3. Если описание алгоритма не совпадает с образцами, а ошибки в описании алгоритма с первого взгляда не видны, осуществите формальное исполнение алгоритма с тестовыми примерами исходных данных. Длину тестового массива следует сократить до четырех–шести элементов. При составлении тестов необходимо особенно тщательно проверять «критические» случаи, например когда элементы массива одинаковы или изначально упорядочены. Оцените правильность полученных результатов.

4. Обратите внимание на правильность инициализации переменных. Учтите, что приведенное в решении значение может не быть единственно возможным.

5. Обязательно проверьте, присутствует ли вывод ответа.

6. Отдельно проверьте, обрабатывается ли случай, что искомого числа (группы чисел) в данном массиве может и не быть (если иное не указано в условии задачи).

7. При оценке алгоритма отметьте все ошибки, упомянутые в критериях оценивания. В случае, если таких ошибок две или более, сразу снижайте оценку до 0 баллов.

8. Не допускайте произвольного ужесточения критериев оценивания. Не вводите дополнительных ограничений. Не оценивайте синтаксические ошибки, «стиль» программирования, аккуратность записи, наличие комментариев и прочие важные, но не проверяемые данной задачей элементы.

9. Не забывайте, что эффективность алгоритмов в данной задаче не оценивается, поэтому не следует снижать оценку за решение, в котором, например, для поиска максимума используется сортировка.

Ситуации, сложные для оценивания

№	Типичная проблемная ситуация	Решение
1	Отсутствует в явном виде инициализация счетчика или сумматора найденных элементов (пар, троек и т.д.) массива. При этом могут быть комментарии, указывающие на возможность автоматической инициализации счетчика нулевым значением.	В соответствии с критериями оценивания отсутствие инициализации в тексте программы считается ошибкой.
2	Вместо того, чтобы написать фрагмент программы соответствующий многоточию в условию, обучающийся пишет программу целиком.	В случае верной программы оценка не снижается. При наличии алгоритмических ошибок (в том числе допущенных при переписывании части программы из условия) оценка снижается в соответствии с критериями оценивания.

3	Формат вывода результата несколько отличается от предписанного в задании, например, добавляется слово «ответ:»; вместо печати результата «в строчку» он печатается «в столбик» или наоборот.	Оценка не снижается
4	Печатается верный ответ, но значения элементов массива в памяти не изменяются.	Оценка снижается в соответствии с критериями оценивания.

Задание 26

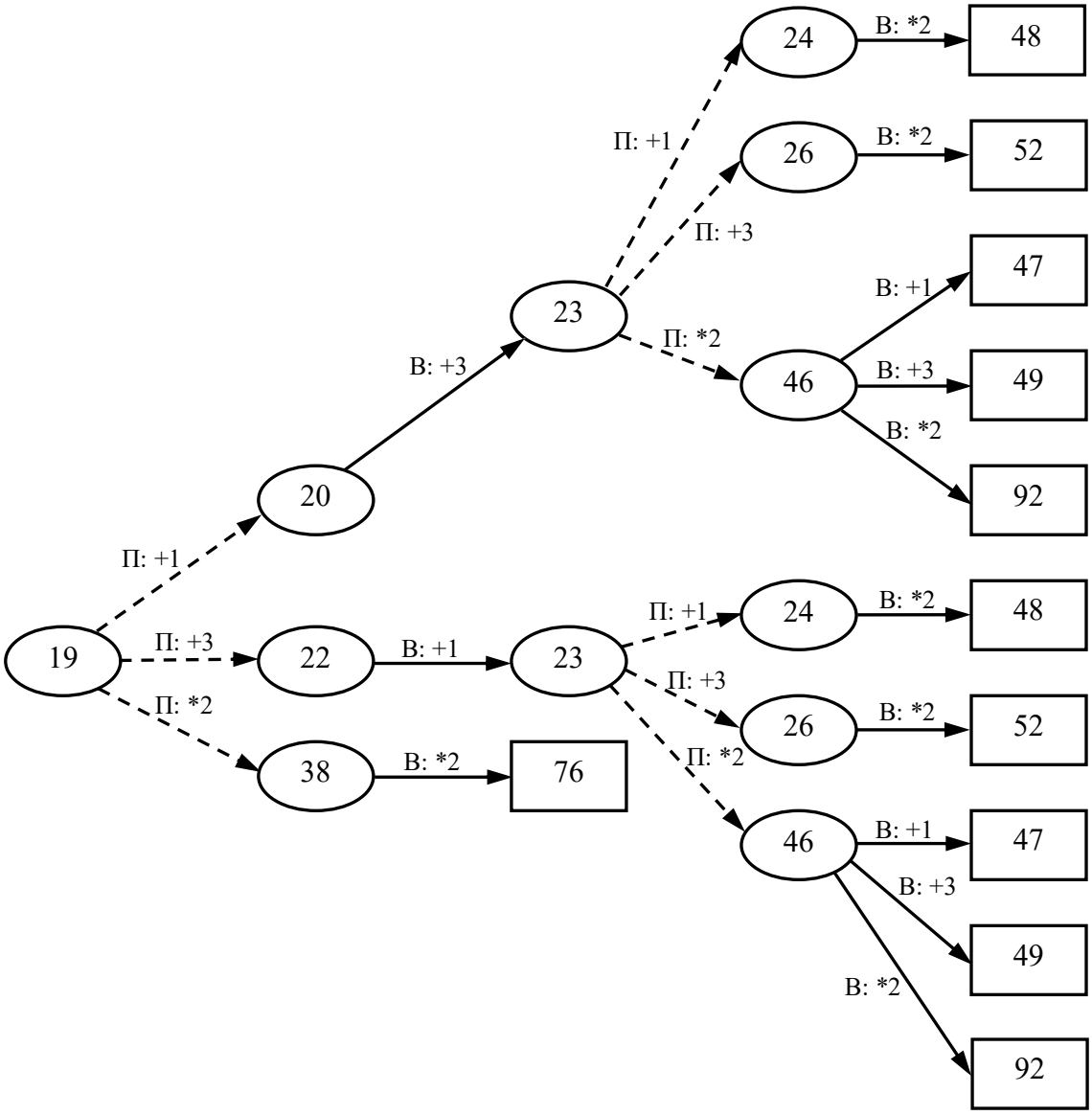
1. При описании выигрышной стратегии для определенной позиции может быть указан один из возможных выигрывающих ходов, но не указаны другие возможные выигрывающие ходы. Это не является ошибкой.

2. Обозначения на рисунке могут отличаться от использованных в примере решения, например, не обязательно использовать пунктирные линии или указывать, чей ход, над дугами графа.

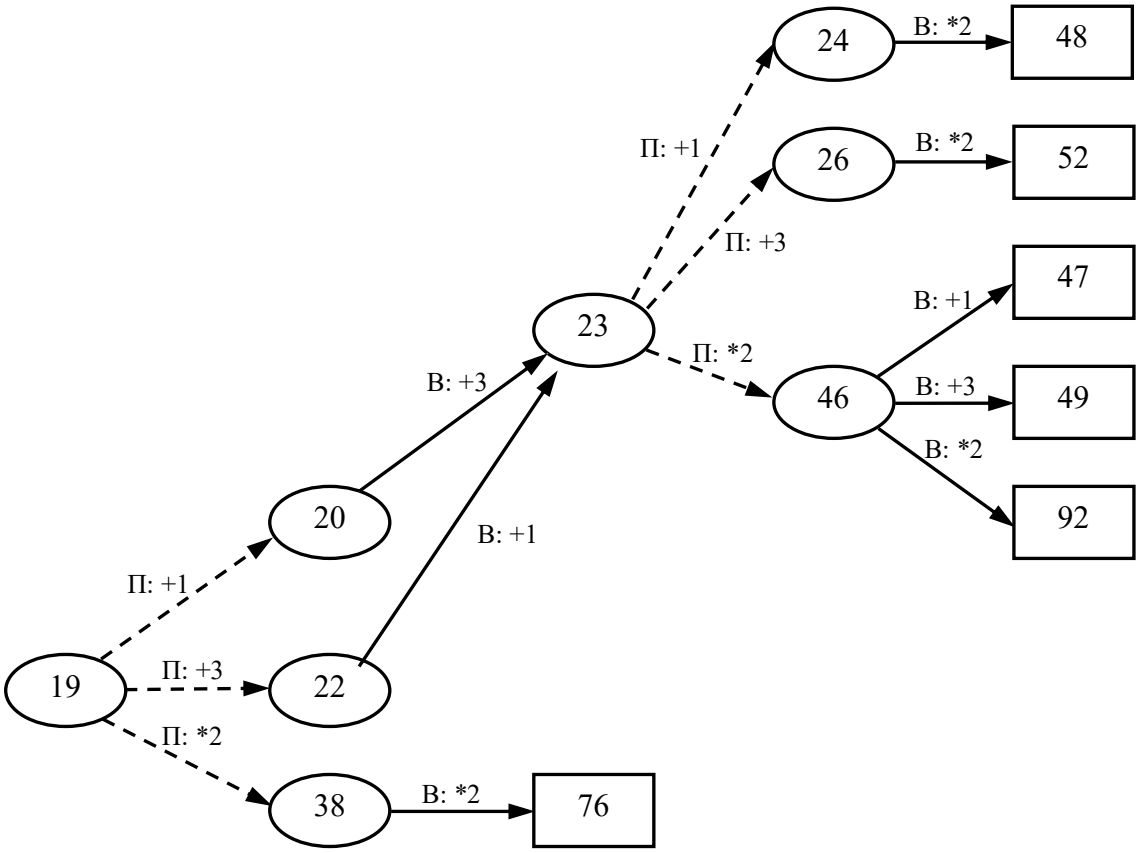
3. При выполнении задания 3 в построенном дереве должны содержаться возможные при реализации выигрывающим игроком своей выигрышной стратегии позиции, и только они. Так, например, полное дерево игры не является верным ответом на задание 3. Если построено полное дерево игры, но все соответствующие лишним позициям поддеревья зачеркнуты, это допустимо.

4. В случае, когда поддеревья дерева игры, начиная с некоторой позиции, идентичны, допускается приводить в ответе только одно из совпадающих поддеревьев, соединяя его дугами со всеми такими позициями. При этом граф игры с формальной точки зрения перестает быть деревом, но в данной ситуации оценку снижать не следует.

Пример:



Поддеревья позиций 23 идентичны, поэтому экзаменуемый может заменить их на одно, как показано на рисунке ниже, без снижения оценки за ответ.



Задание 27

Рекомендуем выполнять проверку программы в следующей последовательности.

1. Определите язык программирования, на котором написана программа. Программы, написанные на языках программирования, отличных от Паскаля, Алгоритмического языка, Бейсика, Питона и C/C++, тоже должны быть оценены. При необходимости эксперт может воспользоваться справочной литературой, а также обратиться к консультанту или председателю предметной комиссии. Подсчитайте количество таких синтаксических ошибок в программе, которые не мешают понять ее логику. Систематически встречающаяся ошибка считается за одну. Так, например, если вместо круглых скобок ученик в записи условий использовал везде квадратные, то это считается за одну ошибку.

2. Рассмотрите реализацию алгоритма и определите, верна ли она в целом. Если верна, определите, есть ли в ней легко диагностируемые и исправляемые алгоритмические ошибки (см. критерии для выставления 3 баллов). Определите количество таких алгоритмических ошибок (если они есть).

3. Если ошибок мало, то оцените эффективность предложенного решения по времени выполнения и использованию памяти. Далее руководствуйтесь указаниями по оцениванию.

Ситуации, сложные для оценивания

Программа может быть написана на языке программирования, отличном от тех, которые используются в приводимых в КИМах примерах программ, например на языках C#, Java, Perl, PHP и т.д. При этом допускается использование библиотек, описанных в стандартах соответствующего языка. Например, при использовании языка C++ допускается использование средств стандартных библиотек (библиотеки STL). Могут использоваться,

например, версии языков Паскаль и Бейсик, отличные от использованных в примерах решений. Во всех таких случаях экзаменуемый обязан явно указать, какой именно версией какого языка он пользуется. Если, например, в решении не указана версия языка, а использованное экзаменуемым средство доступно не во всех версиях языка, считается, что экзаменуемый допустил ошибку.

Подключение стандартных библиотек должно быть правильно описано в программе (если это требуется правилами языка). Вызовы библиотечных подпрограмм также должны быть корректно оформлены, т.е. не должно быть ошибок в названии подпрограммы, количестве, порядке и типе аргументов.

Чтобы разобраться в подобных ситуациях, эксперт может воспользоваться справочной литературой, а также обратиться к консультанту или председателю предметной комиссии.

В тексте задания есть фраза «Перед текстом программы кратко опишите используемый Вами алгоритм решения задачи». Отсутствие такого описания (при наличии правильной программы) не является основанием для снижения оценки, задание НЕ контролирует умение экзаменующегося составлять описания. Назначение описания – облегчить работу проверяющего эксперта.

№	Типичная проблемная ситуация	Решение
1	Обучающийся допускает в программе алгоритмическую ошибку, которая не входит в приведенный в критериях оценивания перечень допустимых ошибок на 2 или 3 балла	В соответствии с критериями оценивания программа оценивается не более чем в 1 балл независимо от её возможных остальных достоинств
2	Обучающийся использует библиотечные процедуры поиска, сортировки, обработки списков или приводит рекурсивное решение	Особое внимание при проверке должно быть уделено эффективности программы по памяти и времени, поскольку рекурсивные вызовы в решении данного задания обычно приводят к неэффективности по памяти, а процедуры сортировки– по памяти и по времени.