**Система оценивания экзаменационной работы по информатике и ИКТ**

**Ответы к заданиям с кратким ответом**

**C1**

Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (*x, y* – действительные числа) и определяется принадлежность этой точки заданной закрашенной области (включая границы).

Программист торопился и написал программу неправильно

|  |  |
| --- | --- |
| **Бейсик** | **Паскаль** |
| INPUT x, y  IF y>=x THEN  IF y>=0 THEN  IF y<=2-x\*x THEN  PRINT "принадлежит"  ELSE  PRINT "не принадлежит"  END IF  END IF  END IF  END | var x,y: real;  begin  readln(x,y);  if y>=x then  if y>=0 then  if y<=2-x\*x then  write('принадлежит')  else  write('не принадлежит')  end. |
| **Си** | **Алгоритмический** |
| void main(void){  float x,y;  scanf("%f %f",&x,&y);  if (y>=x)  if (y>=0)  if (y<=2-x\*x)  printf("принадлежит");  else  printf("не принадлежит");  } | алг  нач  вещ *x,y*  ввод x,y  если y>=x то  если y>=0 то  если y<=2-x\*x то  вывод 'принадлежит'  иначе  вывод 'не принадлежит'  все  все  все  кон |

Последовательно выполните следующее.

1. Перерисуйте и заполните таблицу, которая показывает, как работает программа при аргументах, принадлежащих различным областям

(A, B, C, D, E, F и G). Точки, лежащие на границах областей, отдельно не рассматривать.



В столбцах условий укажите "да", если условие выполнится, "нет" если условие не выполнится, "—" (прочерк), если условие не будет проверяться, «не изв.», если программа ведет себя по-разному для разных значений, принадлежащих данной области. В столбце "Программа выведет" укажите, что программа выведет на экран. Если программа ничего не выводит, напишите "—" (прочерк). Если для разных значений, принадлежащих области, будут выведены разные тексты, напишите «не изв». В последнем столбце укажите "да" или "нет".

2. Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы.)



2. Возможная доработка (Паскаль):

if (x<0) and (y<=2-x\*x) and (y>=x) or (x>=0) and (y>=0) and (y<=2-x\*x) then

write('принадлежит')

else

write('не принадлежит')

Возможны и другие способы доработки.

Пример:

if (y<=2-x\*x) and ((y>=0) or (y>=x)) then

write('принадлежит')

else

write('не принадлежит')

|  |  |
| --- | --- |
| **Указания по оцениванию** | **Баллы** |
| Обратите внимание! В задаче требовалось выполнить три действия: указать для каждой области, как будет работать программа, что она выведет на экран и правильно ли это (в виде таблицы), и исправить две ошибки.  Баллы за данное задание начисляются как сумма баллов за верное выполнение каждого действия.  1. Верное заполнение предложенной таблицы.  2. Неправильное использование условного оператора, в результате чего при невыполнении первого или второго условия программа не выдавала ничего (отсутствуют случаи ELSE). Исправлением этой ошибки может быть либо добавление случая ELSE к каждому условию IF, либо объединение всех условий IF в одно при помощи конъюнкции.  В сложных случаях это действие считается выполненным, если программа выдает одно из двух сообщений «принадлежит» или «не принадлежит» для любых чисел *x* и *y*, при этом программа не стала работать хуже, чем раньше, то есть для всех точек, для которых программа ранее выдавала верный ответ, доработанная программа также должна выдавать верный ответ.  3. Приведенным трем ограничениям не удовлетворяют точки плоскости, у которых *y* ≤ 2 − *x*2 , *y* ≥ *x* и *y* < 0 , а также точки, для которых *y* ≤ 2 − *x*2 , *y* < *x* и *y* ≥ 0 . Исправлением этой ошибки может быть разбиение области на две части и использование дизъюнкции либо отбрасывание от большей области ее части.  В сложных случаях это действие считается выполненным, если верно определена закрашенная область, то есть программа выводит сообщение «принадлежит» для всех точек закрашенной области и только для них, для точек вне закрашенной области программа выводит «не принадлежит» или не выводит ничего. |  |
| Правильно выполнены оба пункта задания. Исправлены две ошибки. Программа для всех пар чисел *x*, *y* верно определяет принадлежность точки закрашенной области.  В работе (во фрагментах программ) допускается наличие отдельных синтаксических ошибок, не искажающих замысла автора решения. | 3 |
| 1. Правильно выполнены два действия из трех (исправлены обе ошибки, но в первом пункте задания не приведена таблица (либо таблица содержит ошибки более чем в одной строке), либо приведена таблица (которая содержит ошибки не более чем в одной строке), но исправлена только одна ошибка программы).  При написании операций сравнения допускается одно неправильное использование строгих/нестрогих неравенств (считается несущественной ошибкой, погрешностью записи).  Например, вместо «y>=-x» используется «y>-x».  2. Или выполнены все три действия, но при этом в логическом выражении неверно учтены приоритеты логических операций (не расставлены или неправильно расставлены скобки в выражениях). | 2 |
| Правильно выполнено только одно действие из трех, то есть, либо только приведена таблица, которая содержит ошибки в не более чем двух строках, либо таблица не приведена (или приведена и содержит ошибки более чем в двух строках), но исправлена одна ошибка программы. При оценивании этого задания на 1 балл допускается не учитывать корректность работы программ на точках границ областей (вместо нестрогих неравенств в решении были использованы строгие неравенства или наоборот). | 1 |
| Все пункты задания выполнены неверно (таблица анализа правильности алгоритма не приведена, либо содержит ошибки более чем в двух строках, программа не приведена, либо ни одна из двух ошибок не исправлена). | 0 |
| *Максимальный балл* | 3 |

**C2**

Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, позволяющий найти и вывести минимальное значение среди элементов массива, которые имеют чётное значение и не делятся на три. Гарантируется, что в исходном массиве есть хотя бы один элемент, значение которого чётно и не кратно трем.

Исходные данные объявлены так, как показано ниже. Запрещается использовать переменные, не описанные ниже, но использовать все описанные переменные не обязательно.

|  |  |
| --- | --- |
| **Паскаль** | **Бейсик** |
| const  N=20;  var  a: array [1..N] of integer;  i, j, min: integer;  begin  for i:=1 to N do  readln(a[i]);  …  end. | N=20  DIM A(N) AS INTEGER  DIM I, J, MIN AS INTEGER  FOR I = 1 TO N  INPUT A(I)  NEXT I  …  END |
| **СИ** | **Алгоритмический язык** |
| #include <stdio.h>  #define N 20  void main(void){  int a[N];  int i, j, min;  for (i=0; i<N; i++)  scanf("% d", &a[i]);  …  } | алг  нач  цел N=20  целтаб a[1:N]  цел i, j, MIN  нц для i от 1 до N  ввод a[i]  *кц*  ...  кон |
| **Естественный язык** | |
| Объявляем массив A из 20 элементов.  Объявляем целочисленные переменные I, J, MIN.  В цикле от 1 до 20 вводим элементы массива A с 1-го по 20-й.  … | |

В качестве ответа вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Borland Pascal 7.0) или в виде блок-схемы. В этом случае вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на естественном языке).

|  |
| --- |
| **Содержание верного ответа и указания по оцениванию**  (допускаются иные формулировки ответа, не искажающие его смысла) |
| **На языке Паскаль** |
| min:=1000;  for i:=1 to N do  if (a[i] mod 2=0) and (a[i] mod 3<>0) and (a[i]<min) then  min:=a[i];  writeln(min); |
| **На Алгоритмическом языке** |
| MIN:=1000;  нц для i от 1 до N  если mod(a[i], 2) =0 и mod(a[i], 3) <>0 и a[i]< MIN  то  MIN := a[i]  все  кц  вывод MIN |
| **На языке Бейсик** |
| MIN = 1000  FOR I = 1 TO N  IF (A(I) MOD 2 = 0) AND (A(I) MOD 3 <> 0) AND (A(I) < MIN) THEN  MIN = A(I)  END IF  NEXT I  PRINT MIN |
| **На языке СИ** |
| min=1000;  for (i=0; i<N; i++)  if (a[i]%2==0 && a[i]%3!=0 && a[i]<min)  min=a[i];  printf("%d", min); |
| **На естественном языке** |
| Записываем в переменную MIN начальное значение, равное 1000. В цикле от первого элемента до двадцатого находим остаток от деления элемента исходного массива на два и на три. Если остаток от деления на два равен нулю и остаток от деления на три не равен нулю, то сравниваем значение текущего элемента массива со значением переменной MIN. Если текущий элемент массива меньше MIN, то записываем в MIN значение этого элемента массива. Переходим к следующему элементу.  После завершения цикла выводим значение переменной MIN. |

|  |  |
| --- | --- |
| **Указания по оцениванию** | **Баллы** |
| Предложен правильный алгоритм, выдающий верное значение. Допускается запись алгоритма на другом языке, использующая аналогичные переменные. В случае, если язык программирования использует типизированные переменные, описания переменных должны быть аналогичны описаниям переменных на естественном 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.  В алгоритме, записанном на языке программирования, допускается наличие отдельных синтаксических ошибок, не искажающих замысла автора программы. | 2 |
| В любом варианте решения может присутствовать не более одной ошибки из числа следующих.  1. Не инициализируется или неверно инициализируется переменная MIN (например, ей присваивается значение a[1] или число, меньшее 1000).  2. Неверно осуществляется проверка чётности элемента массива.  3. На чётность проверяется не значение элемента, а его индекс.  4. Неверно осуществляется проверка делимости на три.  5. На делимость на три проверяется не значение элемента, а его индекс.  6. В условии проверки на делимость вместо логической операции «И» используется логическая операция «ИЛИ».  7. Отсутствует вывод ответа.  8. Используется переменная, не объявленная в разделе описания переменных.  9. Не указано или неверно указано условие завершения цикла.  10.Индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно.  11.Неверно расставлены операторные скобки. | 1 |
| Ошибок, перечисленных в п. 1–11, две или больше, или алгоритм сформулирован неверно. | 0 |
| *Максимальный балл* | 2 |

**C3**

У исполнителя Утроитель две команды, которым присвоены номера:

**1. прибавь 1,**

**2. умножь на 3.**

Первая из них увеличивает число на экране на 1, вторая – утраивает его. Программа для Утроителя – это последовательность команд. Сколько есть программ, которые число 1 преобразуют в число 29?

Ответ обоснуйте.

|  |
| --- |
| **Содержание верного ответа и указания к оцениванию**  (допускаются иные формулировки ответа, не искажающие его смысла) |
| Обозначим *R*(*n*) *–* количество программ, которые преобразуют число 1 в число *n*. Обозначим *t*(*n*) наибольшее кратное трем, не превосходящее *n.* Обе команды исполнителя увеличивают исходное число, поэтому общее количество команд в программе не может превосходить 28.  Верны следующие соотношения:  1. Если *n* не делится на 3, то тогда *R*(*n*) *= R*(*t*(*n*))*,* так как существует единственный способ получения *n* из *t*(*n*) –прибавлением единиц*.*  2. Пусть *n* делится на 3. Тогда *R*(*n*) *= R*(*n/*3)*+R*(*n-*1)*= R*(*n/*3)*+R*(*n-*3) (*если n>*3)*.* При *n*=3  *R*(*n*) = 2 (два способа: прибавлением двух единиц или однократным умножением на 3).  Поэтому достаточно по индукции вычислить значения *R*(*n*) для всех чисел, кратных трем и не превосходящих 29.  Имеем:  *R*(2)=1  *R*(3) = 2 = *R*(4)=*R*(5)  *R*(6) = *R*(2)+*R*(3) =1+2 = 3 = *R*(7)=*R*(8)  *R*(9) = *R*(3)+*R*(6) =2+3 =5 = *R*(10)=*R*(11)  *R*(12) = *R*(4)+*R*(9) = 2+5 = 7 = *R*(13)=*R*(14)  *R*(15) = *R*(5)+*R*(12) =2+7 =9 = *R*(16)=*R*(17)  *R*(18) = *R*(6)+*R*(15) = 3+9 = 12 = *R*(19)=*R*(20)  *R*(21) = *R*(7)+*R*(18) = 3+12 = 15 = *R*(22)=*R*(23)  *R*(24) = *R*(8)+*R*(21) = 3+ 15 = 18 = *R*(25)=*R*(26)  *R*(27) = *R*(9)+*R*(24) = 5 + 18 = 23 = *R*(28)=*R*(29)  **Ответ: 23**  *Другой способ решения*  Будем решать поставленную задачу последовательно для чисел 1, 2, 3, …, 29 (то есть для каждого из чисел определим, сколько программ исполнителя существует для его получения). Количество программ, которые преобразуют число 1 в число *n,* будем обозначать через *R*(*n*)*.* Число 1 у нас уже есть, значит, его можно получить с помощью “пустой” программы. Любая непустая программа увеличит исходное число, т.е. даст число, больше 1. Значит, *R*(1) = 1. Для каждого следующего числа рассмотрим, из какого числа оно может быть получено за одну команду исполнителя. Если число не делится на три, то оно может быть получено только из предыдущего с помощью команды **прибавь 1**. Значит, количество искомых программ для такого числа равно количеству программ для предыдущего числа: *R*(*i*) = *R*(*i*-1). Если число на 3 делится, то вариантов последней команды два: **прибавь 1** и **умножь на 3**, тогда *R*(*i*) = *R*(*i*-1) + *R*(*i*/3). Заполним соответствующую таблицу по приведенным формулам слева направо:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 5 | 5 | 5 | 7 | 7 | 7 | 9 | | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** |  | | 9 | 9 | 12 | 12 | 12 | 15 | 15 | 15 | 18 | 18 | 18 | 23 | 23 | 23 |  |   При этом ячейки, относящиеся к числам, которые не делятся на 3, можно в решении и опустить (за исключением первого и последнего чисел):   |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | **3** | **6** | **9** | **12** | **15** | **18** | **21** | **24** | **27** | **30** | **31** | | 1 | 2 | 3 | 5 | 7 | 9 | 12 | 15 | 18 | 23 | 28 | 28 | |

|  |  |
| --- | --- |
| **Указания по оцениванию** | **Баллы** |
| Правильное указание количества возможных программ со строгим доказательством правильности (одним из приведенных выше способов или любым другим). | 3 |
| Два балла ставятся в одном из двух случаев:  1. Правильное указание количества возможных программ, основанное на верных рассуждениях, но доказательство правильности неполно. В частности, оценка в 2 балла выставляется в случае, если просто перечислены все правильные программы и не доказано отсутствие других программ, кроме приведенных.  2. Приведены правильные и строгие рассуждения, доведенные до конца, но в вычислениях допущена арифметическая ошибка, в результате чего получен неверный ответ. | 2 |
| Представленное решение обладает одним из свойств  1. Указано, что нужно рассматривать значения *n,* меньшие, чем 29, и приведены правильные рекуррентные соотношения (см. выше), возможно, неполные.  2. Правильно выписаны и обоснованы значения *R*(*n*) для небольших *n.*  3. Правильно написан ответ, но нет его обоснования. | 1 |
| Не выполнено ни одно из перечисленных выше условий | 0 |
| *Максимальный балл* | 3 |

**C4**

В командных олимпиадах по программированию для решения предлагается не больше 11 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систему соревнований. Вам предлагается написать эффективную, в том числе по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить наиболее популярные задачи. Следует учитывать, что количество запросов в списке может быть очень велико, так как многие соревнования проходят с использованием Интернет.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке подаётся количество пришедших запросов N. В каждой из последующих N строк записано название задачи в виде текстовой строки. Длина строки не превосходит 100 символов, название может содержать буквы, цифры, пробелы и знаки препинания.

*Пример входных данных:*

6

А+B

Крестики-Нолики

Прямоугольник

Простой делитель

А+В

Простой делитель

Программа должна вывести список из трёх наиболее популярных задач с указанием количества запросов по ним. Если в запросах упоминаются менее трех задач, то выведите информацию об имеющихся задачах. Если несколько задач имеют ту же частоту встречаемости, что и третья по частоте встречаемости задача, их тоже нужно вывести.

*Пример выходных данных для приведённого выше примера входных данных:*

А+В 2

Простой делитель 2

Крестики-Нолики 1

Прямоугольник 1

|  |
| --- |
| **Содержание верного ответа и указания по оцениванию**  (допускаются иные формулировки ответа, не искажающие его смысла) |
| Программа читает все входные данные один раз, не запоминая их в массиве, размер которого равен N, а составляя только список встретившихся задач и количества запросов по каждой из них. Во время чтения данных об очередной задаче просматривается список ранее сохраненных задач; если она уже есть в списке, то количество запросов по ней увеличивается на 1, иначе задача добавляется в массив упомянутых в запросах задач (при корректных данных он не может быть больше 11). После окончания ввода производится сортировка массивов задач и количества запросов, отданных за них, в порядке убывания количества запросов, затем выводится список из трёх первых задач с указанием частоты встречаемости (или весь список, если его длина меньше трёх). Вместо сортировки можно применить и алгоритм поиска трёх максимальных элементов в массиве. Затем выводятся задачи, частота встречаемости которых не ниже, чем у третьей задачи. Баллы начисляются только за программу, которая решает задачу хотя бы для одного частного случая. Ниже приведены примеры решения задания на Алгоритмическом языке, а также на языках Паскаль и Бейсик. Допускаются  решения, записанные на других языках программирования. При оценивании решений на других языках программирования необходимо учитывать особенности этих языков программирования. Так, на языке C++ при считывании строковой переменной будет считано не все название задачи, а только его первое слово, поэтому следует использовать функцию getline(cin,s), аналогичная проблема возникает и в языке Си.\_\_ |
| **Пример правильной и эффективной программы на языке Паскаль:** |
| Var N, Num, i, j, t: integer; Count: array[1..11] of integer;  s: string; Names: array[1..11] of string;  Begin  Num:=0; {Число различных задач в списке запросов}  ReadLn(N); {Считываем количество запросов}  for i:=1 to N do  begin  ReadLn(S); {считали очередную задачу}{Осуществляем ее поиск в списке уже встретившихся}  j:=1;  while (j<=Num) and (s<>Names[j]) do j:=j+1; {Если она найдена}  if j<=Num then {Увеличиваем счетчик числа запросов}  Count[j]:=Count[j]+1  else begin {Иначе добавляем задачу в конец списка}  Names[j]:=s;  Count[j]:=1;  Num:=Num+1  end  end;  {Сортируем массивы Names и Count в порядке убывания значений массива Count}  for i:=Num downto 2 do  for j:=2 to i do if Count[j-1]<Count[j] then  begin  t:=Count[j]; Count[j]:=Count[j-1]; Count[j-1]:=t;  s:=Names[j]; Names[j]:=Names[j-1]; Names[j-1]:=s;  end;  if Num >= 3 then j := 3 else j := Num;  i := 1;  while (i <= Num) and (Count[i] >= Count[j]) do  begin  WriteLn(Names[i], ' ', Count[i]);  i := i + 1;  end  end. |
| **Пример правильной и эффективной программы на Алгоритмическом языке:** |
| литтаб Names[1:11] | названия задач  целтаб Count[1:11] | счетчики числа запросов по каждой задаче  цел i, j, t  лит s  | 1. Чтение списка запросов  | 1.1. Инициализация количества запросов и счетчика задач  Num:=0 |Число различных задач в списке запросов  ввод N |Считываем количество запросов  | 1.2. Цикл чтения  нц для i от 1 до N  ввод s |Считали очередную задачу. Осуществляем ее поиск в списке уже встретившихся  j:=1  нц пока(j<=Num) и (s<> Names[j])  j:=j+1  кц | Обрабатываем очередную задачу  если j<=Num | Если задача найдена в списке то Увеличиваем счетчик числа запросов  Count[j]:=Count[j]+1  иначе | Добавляем задачу в конец списка  Names[j]:=s;  Count[j]:=1;  Num:=Num+1  все  кц  | 2. Совместно сортируем массивы Names и Count| в порядке убывания значений массива Count  нц для i от Num до 2 шаг -1  нц для j от 2 до i  если Count[j-1]<Count[j] то  t:=Count[j]; Count[j]:=Count[j-1]; Count[j-1]:=t;  s:=Names[j]; Names[j]:=Names[j-1]; Names[j-1]:=s;  все  кц  кц | 3. Вывод задач-"призеров"  | 3.1. Определение порога для количества запросов по задаче Порог равен Count[j]  если Num >= 3  то j := 3  иначе j := Num;  все | 3.2. Цикл вывода  i := 1;  нц пока (i <= Num) и (Count[i] >= Count[j])  вывод нс, Names[i], ' ', Count[i], нс  i := i+ 1;  кц  кон |
| **Пример правильной и эффективной программы на языке Бейсик:** |
| DIM n, Num, i, j, t AS INTEGER  DIM Count(11) AS INTEGER  DIM s AS STRING  DIM Names(11) AS STRING  REM Число различных задач в списке запросов  Num = 0  REM Считываем количество запросов  INPUT (n)  FOR i = 1 TO n  REM Считываем очередную задачу  INPUT (s)  REM Осуществляем ее поиск в списке уже встретившихся  j = 1  WHILE j <= Num AND s <> Names(j)  j = j + 1  WEND  IF j <= Num THEN  REM Если она найдена, увеличиваем счетчик числа запросов  Count(j) = Count(j)+1  ELSE  REM Иначе добавляем задачу в конец списка  Names(j) = s: Count(j) = 1  Num = Num + 1  ENDIF  NEXT i  REM Сортируем массивы Names и Count в порядке убывания значений массива Count  FOR i = Num TO 2 Step -1  FOR j =2 TO i  IF Count(j-1) < Count(j) THEN  t = Count(j)  Count(j) = Count(j-1)  Count(j - 1)=t  s = Names(j)  Names(j) = Names(j-1)  Names(j - 1)=s  END IF  NEXT j  NEXT i  REM определение порога для количества появлений задач из списка вывода; порог равен Count(j)  IF Num >= 3 THEN  j = 3  ELSE  j = Num  END IF  i = 1  REM Вывод наиболее популярных задач  WHILE i <= Num AND Count(i) >= Count(j)  PRINT Names(i), Count(i)  i = i + 1  WEND |

|  |  |
| --- | --- |
| **Указания по оцениванию** | **Баллы** |
| Программа работает для любых входных данных произвольного размера и находит ответ, не сохраняя входные данные в массиве, размер которого соответствует числу N (количеству запросов). Программа просматривает входные данные один раз, сохраняя в массиве размером 11 данные о количестве решений, поданных для каждой из встретившихся в списке задач (и учитывает, что в списке их может быть и меньше 11). Допускается наличие в тексте программы одной синтаксической ошибки: пропущен или неверно указан знак пунктуации, неверно написано или пропущено зарезервированное слово языка программирования, не описана или неверно описана переменная, применяется операция, недопустимая для соответствующего типа данных (если одна и та же ошибка встречается несколько раз, то это считается за одну ошибку). | 4 |
| Программа работает верно, но входные данные запоминаются в массиве, размер которого соответствует числу N. Этот массив, возможно, потом сортируется. Допускается наличие от одной до трех синтаксических ошибок. Возможно, в принципиально верно организованном вводе данных есть одна ошибка (например, использование read вместо readln в Паскале или неверное считывание строки в C++). Три балла также выставляется, если в эффективной программе, удовлетворяющей критериям выставления 4 баллов, есть одна ошибка, в результате которой программа работает неверно на некоторых наборах нетипичных входных данных (например, все запросы относятся к одной и той же задаче) | 3 |
| Программа работает в целом верно, эффективно или нет, но в реализации алгоритма содержится до двух ошибок (неверная инициализация счётчиков – хотя в предложенных выше решениях обнулять их не требуется; возможно, программа неверно работает, если в списке упомянуто меньше 11 задач, выход за границу  массива, допущена ошибка в принципиально верно организованной сортировке или алгоритме поиска минимальных элементов, используется знак “<” вместо “<=”, “or” вместо “and” и т. п.).  Возможно, некорректно организовано считывание входных данных. Допускается наличие от одной до пяти синтаксических ошибок, описанных выше. | 2 |
| Программа, возможно, неверно работает при некоторых входных данных, но по приведённому тексту решения ясно, что экзаменуемый понимает, из каких этапов должно состоять решение задачи. При использовании сортировки она может быть реализована принципиально неверно (например, вместо двух циклов используется один), или допущена принципиальная ошибка в поиске трёх максимальных элементов. Всего допускается до 4 различных ошибок в реализации алгоритма, в том числе описанных в критериях присвоения двух баллов. Допускается наличие от одной до семи синтаксических ошибок, описанных выше. | 1 |
| Не выполнено ни одно из перечисленных выше условий | 0 |
| *Максимальный балл* | 4 |