**Материалы для подготовки к ЕГЭ. Задание С1**

Выписка из «Спецификации контрольных измерительных материалов единого государственного экзамена 2010 года по информатике и ИКТ», подготовленного Федеральным государственным научным учреждением «ФЕДЕРАЛЬНЫЙ ИНСТИТУТ ПЕДАГОГИЧЕСКИХ ИЗМЕРЕНИЙ»

**Общая характеристика заданий**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Задание в работе, количество баллов** | **Элемент содержания** | **Требования к уровню подготовки** | **Соответствие стандарту** | **Код вида деятельности** |
| **С1**  Умение прочесть фрагмент программы на языке программирования и исправить допущенные ошибки  **3 балла** | 1. Основные конструкции языка программирования.  2. Система программирования. | Читать и отлаживать программы на языке программирования | СП | Применение знаний и умений в **новой** ситуации.  Повышенный уровень. |

**Примечание**

СП – стандарт среднего (полного) общего образования, профильный уровень

**Время выполнения работы**

|  |  |  |
| --- | --- | --- |
| **Раздел** | **Рекомендуемое время** | |
| **В часах** | **В минутах** |
| Часть 1 (А) и Часть 2 (В) | 1,5 | 90 |
| Часть 3 (С) | 2,5 | 150 |
| **Всего** | **4** | **240** |

**Дополнительные материалы и оборудование**

Работа выполняется учащимися **без использования компьютеров** и других технических средств. Вычислительная сложность заданий не требует использования калькуляторов, поэтому в целях обеспечения равенства всех участников

**Возможные алгоритмические задачи, 2009 год**

1. **Проверка упорядоченности массива.**
2. **Слияние двух упорядоченных массивов.**
3. **Сортировка (например, вставками).**
4. **Поиск заданной подстроки (скажем, "abc") в последовательности символов.**
5. Поиск корня делением пополам.
6. **Поиск наименьшего делителя целого числа.**
7. **Разложение целого числа на множители (простейший алгоритм).**
8. Умножение двух многочленов.
9. **Суммирование массива.**

**Возможные алгоритмические задачи, 2010 год**

1. Нахождение минимума и максимума двух, трех, четырех данных чисел без использования массивов и циклов.
2. Нахождение всех корней заданного квадратного уравнения.
3. Запись натурального числа в позиционной системе с основанием меньшим или равным 10. Обработка и преобразование такой записи числа.
4. Нахождение сумм, произведений элементов данной конечной числовой последовательности (или массива). **(2009 – 9)**
5. Использование цикла для решения простых переборных задач (поиск наименьшего простого делителя данного натурального числа, проверка числа на простоту, и т.д.)  
   **(2009 – 6, 7)**
6. Заполнение элементов одномерного и двумерного массива по заданным правилам.
7. Операции с элементами массива. Линейный поиск элемента. Вставка и удаление элементов в массиве. Перестановка элементов данного массива в обратном порядке. Суммирование элементов массива. Проверка соответствия элементов массива некоторому условию. **(2009 -1)**
8. Нахождение второго по величине (второго максимального или второго минимального) значения в данном массиве за однократный просмотр массива.
9. Нахождение минимального (максимального) значения в данном массиве и количества элементов, равных ему за однократный просмотр массива.
10. Операции с элементами массива, отобранных по некоторому условию (например, нахождение минимального четного элемента в массиве, нахождение количества и суммы всех четных элементов в массиве).
11. Сортировка массива. **(2009 – 3)**
12. Слияние двух упорядоченных массивов в один без использования сортировки.   
    **(2009 – 2)**
13. Обработка отдельных символов данной строки. Подсчет частоты появления символа в строке.
14. Работа с подстроками данной строки с разбиением на слова по пробельным символам. Поиск подстроки внутри данной строки, замена найденной подстроки на другую строку. **(2009 – 4)**

### Задание С1

Умение прочесть фрагмент программы на языке программирования и исправить допущенные ошибки.

Максимальный балл – 3

**Что нужно знать:**

1. правила построения программы на алгоритмическом языке,
2. правила работы с переменными (объявление, ввод, вывод, оператор присваивания),
3. ветвление – выбор одного из двух возможных вариантов действий в зависимости от значения некоторого логического выражения,
4. на блок-схеме алгоритма ветвление изображается:

|  |  |
| --- | --- |
| **Полная форма** | **Неполная форма** |
| **Логическое выражение**  **Оператор-1**  **Оператор-2**  **True**  **False** | **Логическое выражение**  **Оператор-1**  **True**  **False** |
| **If <Логическое выражение>**  **Then <Оператор-1>**  **Else <Опрератор-2>;** | **If <Логическое выражение>**  **Then <Оператор-1>;** |
| Если логическое выражение принимает значение *True*, то выполняется *Оператор-1*, в противном случае – *Оператор-2*. | Если логическое выражение принимает значение *True*, то выполняется *Оператор-1*, в противном случае – выполняется оператор идущий за условным оператором.. |
| **Пример** | |
| **If a = b**  **Then a:=5**  **Else b:= a+10;** | **If a = b**  **Then a:=5;** |

**Примечание**

Теоретически можно сделать, чтобы *Оператор-1* отсутствовал, но это неграмотное решение только осложняет понимание алгоритма и запутывает его.

1. в качестве *Оператор-1* или *Оператор-2* может стоять оператор ветвления:

**Логическое выражение**

**Оператор-1**

****

**Оператор-2**

**True**

**False**

|  |
| --- |
| **If <Логическое выражение>**  **Then {Оператор-1}**  **If <Логическое выражение>**  **Then <Оператор-3>**  **Else <Опрератор-4>**  **Else <Опрератор-2>;** |

|  |
| --- |
| **Пример**  **If a = b**  **Then**  **If a>5**  **Then a:=a-5**  **Else a:=a+5**  **Else b:= a+10;** |

**Примечание**

При записи операторов для наглядности целесообразно использовать сдвиг.

1. в качестве *Оператор-1* или *Оператор-2* может стоять несколько операторов:

**Логическое выражение**

**Оператор-1**

****

**Оператор-2**

**True**

**False**

Использование составного оператора (операторных скобок): **begin-end**

|  |
| --- |
| **If <Логическое выражение>**  **Then {Оператор-1}**  **begin**  **<Оператор-3>;**  **<Опрератор-4>**  **end**  **Else <Опрератор-2>;** |

|  |
| --- |
| **Пример**  **If a = b**  **Then**  **begin**  **a:=a-5;**  **b:=a+5**  **end**  **Else b:= a+10;** |

1. основным вопрос этого задания – использование вложенных условных операторов; в тексте задания фрагмент программы записан без отступов или с неправильными отступами.

Чтобы понять, как работает программа, нужно определить, к какому из условных операторов **if** относится часть **else**. Для этого используется правило: «**else** относится к ближайшему **if**»

|  |
| --- |
| **Пример**  **if a = b then begin**  **a = c**  **c:= 1**  **c:=0**  **True**  **False**  **a = b**  **True**  **False**  **if a = c then**  **c:=1;**  **end**  **else c:=0;** |

|  |
| --- |
| **Пример**  **if a = b then**  **a = c**  **c:= 1**  **c:=0**  **True**  **False**  **a = b**  **True**  **False**  **if a = c then**  **c:=1**  **else c:=0;** |

1. в условных операторах можно использовать сложные условия, которые строятся из простых отношений (**<, <=, >, >=, =, <>**) с помощью логических операций **not** («НЕ», отрицание), **and** («И», одновременное выполнение двух условий) и **o**r («ИЛИ», выполнение хотя бы одного из двух условий),
2. в сложном условии существует приоритет выполнения действий:
   1. выражение в скобках,
   2. not,
   3. and,
   4. or,
   5. отношения.
3. операции равного приоритета выполняются последовательно слева направо.

**Демонстрационный пример С1. 2010**

|  |  |
| --- | --- |
| Безимени-1 | Требовалось написать программу, при выполнении которой с клавиатуры считываются координаты точки на плоскости (*x, y* – действительные числа) и определяется принадлежность этой точки заданной заштрихованной области (включая границы). Программист торопился и написал программу неправильно. |

|  |  |
| --- | --- |
| **var x,y: real;**  **begin**  **readln(x,y);**  **if x\*x+y\*y>=4 then**  **if x>= –2 then**  **if y<= –x then**  **write('принадлежит')**  **else**  **write('не принадлежит')**  **end.** | Последовательно выполните следующее:  1) Приведите пример таких чисел x, y, при которых программа неверно решает поставленную задачу.  2) Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, достаточно указать любой способ доработки исходной программы). |

**Решение**

1. Определим условие, которому должны отвечать точки, принадлежащие заштрихованной области.

|  |  |
| --- | --- |
| **Область** | **Условие** |
| **X**  **Y** | **If x\*x+y\*y>=4**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит');** |
| **X**  **Y**  **X= ─2** | **If x\*x+y\*y>=4**  **then**  **If x>= –2**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит');** |
| **Y= ─X**  **X= ─2**  **X**  **Y** | **If x\*x+y\*y>=4**  **then**  **If x>= –2**  **then**  **If y<= –x**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит');** |

|  |  |
| --- | --- |
| **Y**  **Y= ─X**  **X**  **X= ─2** | **If x\*x+y\*y>=4**  **then**  **If x>= –2**  **then**  **If y<= –x**  **then**  **If y>= 0**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит')**  **else**  **write('не принадлежит');** |

Полученную запись можно упростить, если использовать сложное логическое выражение:

|  |
| --- |
| **if (x\*x+y\*y>=4) and (x>= –2) and (y<= –x) and (y>=0)**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит');** |

Мы получили ответ на второй вопрос.

Для того, чтобы ответить на первый вопрос, представим исходный текст программы в структурированном виде:

|  |
| --- |
| **var x,y: real;**  **begin**  **readln(x,y);**  **if x\*x+y\*y>=4**  **then**  **if x>= –2**  **then**  **if y<= –x**  **then**  **write('принадлежит')**  **else**  **write('не принадлежит')**  **end.** |

В тексте программы три условных оператора, которые содержат три ветви **Then** и одну ветвь **Else** (принадлежит последнему условному оператору).

Для наглядности представим текст программы в виде блок-схемы.

**X\*X+Y\*Y>=4**

**X>= ─2**

**Y<=─X**

Принадлежит

Не принадлежит

**True**

**True**

**True**

**False**

**False**

**False**

Примеры входных данных, при которых программа работает неверно:

X=1, Y=1 (не выдается никакого сообщения)

X=0, Y= ─5 (неправильно определяется принадлежность точки допустимой области)

**Обратите внимание!**

В задаче требовалось выполнить **три** действия: указать пример входных данных, при которых программа работает неверно и исправить две ошибки:

1. Неправильное использование условного оператора, в результате чего при невыполнении первого или второго условия программа не выдавала ничего (отсутствуют случаи ELSE).

2. Приведенным трем ограничениям удовлетворяют также те точки плоскости, у которых (x2+y2>=4) и (y<= –x) и (x>= –2) и (y<0).

**Критерии оценивания**

|  |  |
| --- | --- |
| **Указания по оцениванию** | **Баллы** |
| Правильно выполнены все действия из трех | **3** |
| Правильно выполнены 2 действия из трех | **2** |
| Правильно выполнено только одно действие из трёх. | **1** |
| Все пункты задания выполнены неверно (пример входных данных не указан или указан неверно, программа не приведена, либо приведенная программа корректно работает в не большем количестве случаев, чем исходная) | **0** |
| **Максимальный балл** | **3** |

**Пример С1. Тренировочный тест 2009.**

Требовалось написать программу, которая считает во введенной с клавиатуры строке, содержащей только буквы и пробелы (' '), количество слов, начинающихся и заканчивающихся одной и той же буквой. Слово, состоящее из одной буквы, также считается. Программист написал программу неправильно.

**Var s:string;**

**a:char;**

**i, k: integer;**

**begin**

**readln(s); k:=0;**

**for i:=1 to length(s) do**

**begin**

**if (s[i] = ' ') and (s[i+1] <> ' ') then a:=s[i+1];**

**if (s[i] <> ' ') and (s[i+1] = ' ') then if s[i]=a**

**then k:=k+1;**

**if (s[i - 1] = ' ') and (s[i] <> ' ') and (s[i+1] = ' ')**

**then k:=k+1;**

**end;**

**if k = 0 then**

**write ('таких слов нет')**

**else**

**write('k =',k)**

**end.**

Последовательно выполните три задания:

1. Приведите пример таких строк, при которых программа неверно решает поставленную задачу.
2. Укажите, какая часть программы является лишней.
3. Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому можно указать любой способ доработки исходной программы)

**Решение**

Внесем изменения в запись программы, структурируя ее.

|  |
| --- |
| **Var s:string;**  **a:char;**  **i, k: integer;**  **begin**  **readln(s); k:=0;**  **for i:=1 to length(s) do**  **begin**  **if (s[i] = ' ') and (s[i+1]<>' ')**  **then a:=s[i+1];**  **if (s[i]<>' ') and (s[i+1]=' ')**  **then**  **if s[i]=a**  **then k:=k+1;**  **if (s[i-1]=' ') and (s[i]<>' ') and (s[i+1]=' ')**  **then k:=k+1;**  **end;**  **if k = 0**  **then**  **write ('таких слов нет')**  **else**  **write('k =',k)**  **end.** |

Необходимо внимательно разобрать текст программы. Для этого можно представить текст программы в виде блок-схемы:

**Ввод строки**

**K := 0**

{количество слов}

**Обработка строки символов**

**K=0**

**таких слов нет**

**K**

**True**

**False**

Из блок-схемы можно сделать вывод, что необходимо проанализировать блок «Обработки строки символов», чтобы определить правильность подсчета количества слов.

|  |
| --- |
| **for i:=1 to length(s) do**  **begin**    **if (s[i] = ' ') and (s[i+1] <> ' ')**  **then a:=s[i+1];**    **if (s[i] <> ' ') and (s[i+1] = ' ')**  **then**  **if s[i]=a**  **then k:=k+1;**    **if (s[i - 1] =' ') and (s[i] <> ' ') and (s[i+1] =' ')**  **then k:=k+1;**    **end;** |

Пусть дана строка:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **F** | **Z** | **B** |  | **A** | **B** | **B** | **A** |  | **C** |  | **F** | **T** | **K** | **N** | **D** |  |

В результате обработки мы должны получить K=2.

**Основной блок** (оператор цикла от 1 до длины строки) последовательно выполняет три блока:

1. Анализирует символ – является ли началом нового слова, если является, то переменной **a** присваивает значение первого символа.
2. Анализирует символ – является ли он окончанием слова, если является, то сравнивает его с первым символом (переменная a). Если текущий символ равен переменной **a**, то увеличивает значение переменной **k** – количество слов.
3. Анализирует – состоит ли слово из одной буквы. Если состоит, то увеличивает значение переменной **k** – количество слов. **Этот блок является лишним.**

**Требования к строке** (для правильной работы программы):

Начинаться и оканчиваться строка должна обязательно **пробелом**.

Программа будет **неправильно работать**, если будут введены строки:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **F** | **Z** | **B** |  | **A** | **B** | **B** | **A** |  | **C** |  | **F** | **T** | **K** | **N** | **D** |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **F** | **Z** | **B** |  | **A** | **B** | **B** | **A** |  | **C** |  | **F** | **T** | **K** | **N** | **F** |

**Какие возможны ошибки в программе:**

|  |  |  |
| --- | --- | --- |
| **Номер блока** | **Ошибка** | **Исправить** |
| 1 | Если обрабатывается последний символ строки, то оператор **a:=s[i+1];** выдаст сообщение об ошибке | **for i:=1 to length(s)-1 do** |
| 3 | Если обрабатывается первый символ строки, то оператор  **if (s[i-1]=' ') and (s[i]<>' ') and (s[i+1] =' ')**  **then k:=k+1;**  выдаст сообщение об ошибке | **for i:=2 to length(s)-1 do**  Так как этот блок является лишним, то эту ошибку можно не исправлять, а просто убрать из текста этот блок. |
|  | Строка должна начинаться с пробела и заканчиваться пробелом | **s:=' '+s+' ';** |

**Исправленный текст программы:**

|  |
| --- |
| **Var s:string;**  **a:char;**  **i, k: integer;**  **begin**  **readln(s); k:=0;**  **s:=' '+s+' ';**  **for i:=1 to length(s)-1 do**  **begin**  **if (s[i]=' ') and (s[i+1]<>' ')**  **then a:=s[i+1];**  **if (s[i]<>' ') and (s[i+1]=' ')**  **then**  **if s[i]=a**  **then k:=k+1;**  **end;**  **if k = 0**  **then**  **write ('таких слов нет')**  **else**  **write('k =',k)**  **end.** |

**Рекомендуемые ссылки**

**Демонстрационный вариант контрольных измерительных материалов 2010 года по ИНФОРМАТИКЕ и ИКТ**

**http://www.fipi.ru/view/sections/211/docs/449.html**

**Сайт Полякова Константина Юрьевича**

**http://kpolyakov.narod.ru/school/ege.htm**

**Сайт Давыдовой Елены Владимировны**

**http://informat444.narod.ru/11-6.html**

**Сайт творческих учителей информатики**

**http://www.it-n.ru/communities.aspx?cat\_no=85737&tmpl=com**