Исправление ошибок в простой программе с условными операторами.

Задание 24

Что нужно знать:

- правила построения программы на Паскале, Python или Си;
- правила работы с переменными (объявление, ввод, вывод, оператор присваивания);
- ветвление – это выбор одного из двух возможных вариантов действий в зависимости от того, выполняется ли некоторое условие;
- условный оператор if–else служит для организации ветвления в программе на языке Паскаль;

Что нужно знать:

- условный оператор может иметь полную или неполную форму; вот фрагменты программы, реализующие ветвления, показанные на рисунках 1 и 2:

полная форма:

```
if a = b then begin
  { блок-1 }
end
else begin
  { блок-2 }
end;
```

неполная форма:

```
if a = b then begin
  { блок-1 }
end;
```

здесь вместо комментариев в фигурных скобках (они выделены синим цветом) могут стоять любые операторы языка программирования (в том числе операторы присваивания, другие условные операторы, циклы, вызовы процедур и т.п.)

Что нужно знать:

- обычно при записи программы операторы, находящиеся внутри обоих блоков, сдвигают вправо на 2-3 символа (запись «лесенкой»), это позволяет сразу видеть начало и конец блока (конечно, если «лесенка» сделана правильно)
- после else не надо (нельзя!) ставить какое-то условие, эта часть выполняется тогда, когда условие после if неверно (частая ошибка – после else пытаются написать условие, обратное тому, которое стоит после соответствующего ему if)
- в Паскале перед else не ставится точка с запятой, поскольку это ключевое слово обозначает не начало нового оператора, а вторую часть условного оператора if–else

Что нужно знать:

- слова begin и end (их называют также «операторные скобки») ограничивают блок-1 и блок-2; если внутри блока всего один оператор, эти «скобки» можно не писать, например,

- допустимы такие операторы

```
if a = b then
      c:=1
else c:=0;
```

```
if a = b then begin
   c:=1;
end
else c:=0;
```

```
if a = b then c:=1;
```

Что нужно знать:

- а вот такие операторы недопустимы

```
if a = b then begin
      c:=1
 else c:=0;
```

```
if a = b then
   c:=1;
end
else c:=0;
```

```
if a = b then
   c:=1;
   d:=1;
else x:=1;
```

- ✔ в первом случае есть begin, но забыли про соответствующий ему end;
- ✔ во втором фрагменте наоборот, есть end, а begin отсутствует;

Что нужно знать:

- третий случай более сложный: судя по записи «лесенкой», здесь внутри блока-1 находятся 2 оператора, а операторных скобок begin-end нет; в результате получилось, что оператор c:=1 находится внутри блока-1, он выполняется только при условии a=b;

оператор d:=1 выполняется всегда, после того, как условный оператор закончил работу; а else вообще «висит» непонятно как, тут транслятор выдаст ошибку; исправить эту программу можно так, как показано снизу (добавив пару begin-end):

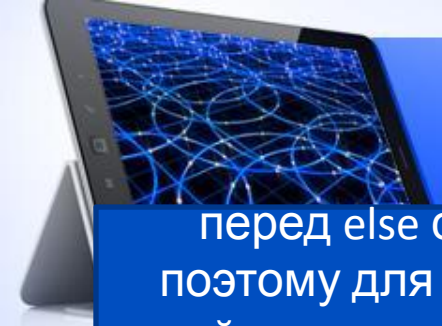
```
if a = b then begin
  c:=1;
  d:=1;
end
else x:=1;
```

Что нужно знать:

- ключевая тема этого задания ЕГЭ – использование вложенных условных операторов, причем в тексте задания фрагмент программы обычно записан без отступов «лесенкой» или с неправильными отступами, например, так:

```
if a = b then begin
  if a = c then
     c:=1;
  end
  else c:=0;
```

```
if a = b then
   if a = c then
      c:=1
else c:=0;
```



Что нужно знать:

- …о задания ЕГ… операторов, при… …ы обычно зап… …правильными о…

так:

перед else стоит end, поэтому для него нужно найти соответствующий ему begin; таким образом определяем, что else относится к первому (внешнему) условному оператору.

```
if a = b then begin
  if a = c then
     c:=1;
  end
  else c:=0;
```

перед else нет end, поэтому он относится к ближайшему по тексту внутреннему условному оператору.

```
if a = b then
  if a = c then
     c:=1
else c:=0;
```

Чтобы разобраться с работой этих программ, нужно определить, к какому из условных операторов if относится часть else; для этого используют такое правило: «любой else относится к ближайшему if».

Что нужно знать:

- 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 or («ИЛИ», выполнение хотя бы одного из двух условий)
- в сложном условии сначала выполняются действия в скобках, потом – not, затем – and, затем – or и, наконец, отношения;
- операции равного уровня (приоритета) выполняются последовательно слева направо

Что нужно знать:

- поскольку отношения в Паскале имеют низший приоритет, в сложном условии их приходится брать в скобки:

```
if (a = b) or (b < c) and (c <> d) then begin
  ...
end;
```

- в приведенном выше примере сначала определяются результаты сравнения (выражения в скобках), затем выполняется операция and («И»), а затем – or («ИЛИ»)

Пример задания:

На обработку поступает последовательность из четырёх неотрицательных целых чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран количество нечётных чисел в исходной последовательности и максимальное нечётное число. Если нечётных чисел нет, требуется на экран вывести «NO». Известно, что вводимые числа не превышают 1000. Программист написал программу неправильно. Вот она:

Пример задания:

```
const n = 4;
var i, x: integer;
var maximum, count: integer;
begin
  count := 0;
  maximum := 999;
  for i := 1 to n do begin
    read(x);
    if x mod 2 <> 0 then begin
      count := count + 1;
      if x > maximum then maximum := i
    end
  end;
  if count > 0 then begin
    writeln(count);
    writeln(maximum)
  end
  else writeln('NO')
end.
```

Пример задания:

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе последовательности: 2 9 4 3

2. Приведите пример такой последовательности, содержащей хотя бы одно нечётное число, что, несмотря на ошибки, программа печатает правильный 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. Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Решение:

1) обратим внимание на две строки в начале программы, которые начинаются с ключевого слова var: это не ошибка, такое повторение, действительно, допустимо в языке Паскаль; возможно, это была одна из ловушек разработчиков КИМ, которую они применили на реальном ЕГЭ-2014
2) теперь выполним программу для заданной последовательности 2 9 4 3, записывая все изменения переменных в таблицу:

Решение:

	условие	i	x	maximum	count	вывод
count:=0;					0	
maximum:=999;				999		
for i := 1 to n do		1				
read(x);			2			
if x mod 2 <> 0 then	нет					
end		2				
read(x);			9			
if x mod 2 <> 0 then	да					
count:=count+1;					1	
if x > maximum then	нет					
end		3				
read(x);			4			
if x mod 2 <> 0 then	нет					
end		4				
read(x);			3			
if x mod 2 <> 0 then	да					
count:=count+1;					2	
if x > maximum then	нет					
end		5				
if count > 0 then	да					
writeln(count);						2
writeln(maximum)						999

Решение:

3) при ручной прокрутке программы мы увидели, что она правильно подсчитала количество нечётных чисел во входной последовательности, но неверно определила максимум: значение переменной maximum, которое было выведено на экран, осталось равным начальному значению 999, так как все остальные нечётные числа были меньше этого начального значения; поэтому ответ на п. 1 задания должен быть таким:

1) **Программа выведет числа 2 и 999.**

Решение:

4) поскольку все числа по условию неотрицательны и не превышают 1000, программа всегда будет выдавать 999 вместо максимального нечётного числа; в то же время мы выяснили, что количество нечётных чисел в последовательности считается правильно; поэтому любая последовательность, содержащая 999, будет обрабатываться правильно
5) таким образом, правильный ответ на п. 2 должен быть таким:

2) Программа работает правильно для последовательности: 2 9 3 999.

Решение:

6) теперь будем искать ошибки; как уже отмечалось, повторное использование ключевого слова var допустимо и указывать это в качестве ошибки нельзя!

7) как следует из результатов ручной прокрутки программы, во многих случаях она выдаёт неверный результат из-за того, что неверно задано начальное значение переменной maximum: оно должно быть меньше, чем любой возможный результат;

8) наименьшее нечётное неотрицательное число – это 1, поэтому можно принять в качестве начального значения maximum любое число, меньшее единицы (на самом деле, программа будет правильно работать и для 1), например:

3) **Ошибка 1. maximum:=999;**
   Исправление: maximum:=0;

Решение:

9) если теперь (с исправленной первой ошибкой) сделать ручную прокрутку программы, то мы увидим, что на последовательности 2 9 4 3 она выдает сначала 2, а потом – 4, то есть, значение максимума вычисляется опять неверно;

)откуда появится число 4 в переменной maximum? оно будет записана в результате выполнения оператора if x > maximum then maximum:=i, который записывает в переменную maximum не значение полученного числа (x), а его номер (i); таким образом, мы нашли вторую ошибку:

Ошибка 2. if x > maximum then maximum:=i
Исправление: if x > maximum then maximum:=x

Итого:

Если Вы хотите ещё поработать над подобными задачами имея возможность свериться с решением, то откройте текстовый документ «24.docx», прикреплённый к этому уроку.

А если считаете, что вы готовы, то решите тест.

ссылка на тест