Тема 11. Использование основных алгоритмических конструкций: следование, ветвление, цикл.

Что нужно знать:

- переменная – это величина, которая имеет имя, тип и значение; переменная может изменяться во время выполнения программы
- оператор присваивания (в Паскале обозначается сочетанием символов «**:=**») служит для записи нового значения в переменную (для изменения ее значения)
- если в переменную записывают новое значение, старое стирается
- знаки +, -, *, / используются для обозначения операций сложения, вычитания, умножения и деления
- запись вида **a := a + 2;** – это не уравнение, а команда «прочитать текущее значение переменной **a**, добавить к нему 2 и записать результат обратно в переменную **a**»;
- для наглядной записи небольших алгоритмов используют блок-схемы; они состоят из блоков разного назначения и соединительных линий со стрелками, которые показывают порядок выполнения блоков
- в задачах ЕГЭ встречаются два блока: *процесс* (выполнение некоторых действий) и *ветвление* (условие, в зависимости от которого выполнение алгоритма продолжается по одной или другой «ветке»)

a:=1;

b:=1;

процесс

a = b?

да

нет

ветвление

- с помощью ветвления можно организовать *цикл* (многократное выполнение одинаковых действий), в этом случае в блок-схеме будет соединительная линия, идущая «в обратном направлении» (петля, замкнутый контур)
- цикл на рисунке (выделен зеленым фоном) закончится только тогда, когда выполнится условие **a = 256**

a:=a*2;

a = 256?

да

нет

Пример задания:

Запишите значение переменной **b** *после выполнения фрагмента алгоритма:*

```
a:=a*2;
b:=b+a;
a:=1;
b:=1;
```

a = 256?

да

нет

Решение (вариант 1, ручная прокрутка):

1. по схеме видим, что алгоритм содержит цикл (есть петля, контур)
2. ручную прокрутку удобнее всего выполнять в виде таблицы, в первом столбце будем записывать выполняемые команды, во втором и третьем – изменение значений переменных **a** и **b**
3. после выполнения первого блока получаем

	a	b
a:=1;	1	?
b:=1;		1

знак вопроса означает, что после выполнения первого оператора значение **b** не определено

4. затем выполняется проверка условия; поскольку **a** не равно 256, ответ на вопрос «**a = 256**?» будет «нет»:

	a	b
a:=1;	1	?
b:=1;		1
a = 256?	нет	

5. далее алгоритм уходит на выполнение тела цикла; здесь сначала меняется переменная **a**, а потом – **b**, причем нужно помнить, что для вычисления **b** используется новое значение a, равное 2, поэтому новое значение **b** равно 1 + 2 = 3:

	a	b
a:=1;	1	?
b:=1;		1
a = 256?	нет	
a:=a*2;	2	
b:=b+a;		3

6. после этого по стрелке переходим на проверку условия; поскольку **a = 2**, ответ на вопрос «**a = 256**?» снова будет «нет», и выполняется очередной шаг цикла:

	a	b
a:=1;	1	?
b:=1;		1
a = 256?	нет	
a:=a*2;	2	
b:=b+a;		3
a = 256?	нет	
a:=a*2;	4	
b:=b+a;		7

7. аналогично можно выполнить вручную все шаги цикла, результаты последнего из них

выглядят так:

	a	b
a:=a*2;	256	
b:=b+a;		511
a = 256?	да	

как только значение **a** стало равно 256, цикл завершает работу

8. таким образом, верный ответ – 511 .

Возможные проблемы:

- таблица получается длинной, много вычислений, можно запутаться
- нужно не забыть, что при выполнении двух операторов в теле цикла к значению **b** добавляется уже новое значение **a**, полученное в предыдущей строке
- не перепутайте переменную, значение которой нужно определить (можно по ошибке вписать в ответ полученное значение **a**)

Решение (вариант 2, анализ алгоритма):

1. «прокрутив» начало алгоритма, можно заметить, что последовательные значения **a** – это степени двойки

   a = 1, 2, 4, 8, … 256

2. поскольку оператор **b:=b+a** означает «взять текущее значение **b**, прибавить к нему текущее значение **a** и результат записать обратно в **b**», изменение **b** сводится к тому, что эти степени двойки складываются:

   b = 1 + 2 + 4 + 8 + … + 256

3. теперь можно, конечно, сложить эти числа вручную (их всего 9), но можно заметить (или вспомнить), что сумма всех последовательных степеней двойки, начиная с 1, на единицу меньше, чем следующая степень двойки (первая, не вошедшая в сумму, здесь – 512); это легко проверяется по начальной части таблицы
4. таким образом, верный ответ 512 – 1 = 511 .

Возможные проблемы:

- для такого анализа требуется некоторое напряжение ума, здесь не обойтись формальным выполнением каких-то заученных действий
- не всегда удается найти короткое решение, «свернув» алгоритм таким образом (в этом случае поможет ручная прокрутка)