

Златопольский Д. М.

# Подготовка к ЕГЭ по информатике



**Решение задач**  
по программированию

**УДК 373.167.1:004+004(075.3)**  
**ББК 32.97я72**  
**367**

Златопольский Д. М.  
367 Подготовка к ЕГЭ по информатике. Решение задач по программированию. – М.: ДМК Пресс, 2017. – 252 с.: ил.  
ISBN 978-5-97060-598-1

Книга предназначена для подготовки учащихся к Единому государственному экзамену по информатике в части решения задач по программированию. Рассмотрена методика решения основных типовых задач по программированию, а также заданий из демонстрационных вариантов ЕГЭ и из пособий, написанных разработчиками контрольно-измерительных материалов по информатике.

Издание также будет полезно студентам вузов и колледжей, преподавателям информатики и другим читателям при изучении программирования вне связи с ЕГЭ.

**УДК 373.167.1:004+004(075.3)**  
**ББК 32.97я72**

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-5-97060-598-1

© Златопольский Д.М., 2017  
© Издание, оформление, ДМК Пресс, 2018

# Содержание

<b>Предисловие</b> .....	8
<b>Глава 1. Вспомогательные задачи</b> .....	9
1.1. Обработка натурального числа .....	10
1.1.1. Выделение цифр.....	10
1.1.2. Определение суммы цифр числа .....	11
1.1.3. Определение произведения цифр числа.....	12
1.1.4. Определение количества цифр числа.....	13
1.1.5. Определение максимальной цифры числа .....	13
1.1.6. Определение минимальной цифры числа .....	15
1.2. Операции с элементами массива, отобранными по некоторому условию .....	15
1.2.1. Изменение элементов массива с заданными свойствами (удовлетворяющих некоторому условию).....	15
1.2.2. Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию) .....	16
1.2.3. Нахождение количества элементов массива с заданными свойствами .....	18
1.2.4. Нахождение среднего арифметического значения элементов массива с заданными свойствами.....	19
1.2.5. Нахождение максимального количества подряд идущих элементов массива, обладающих заданными свойствами.....	20
1.2.6. Нахождение максимальной суммы подряд идущих элементов массива, обладающих заданными свойствами.....	23
1.3. Линейный поиск элемента.....	26
1.3.1. Проверка факта наличия в массиве элемента с заданным значением.....	26
1.3.2. Проверка факта наличия в массиве элемента с заданными свойствами .....	29
1.3.3. Поиск индекса элемента массива, равного некоторому числу .....	30
1.3.4. Поиск индекса элемента массива с заданными свойствами.....	31

1.3.5. Поиск индекса первого элемента массива, равного некоторому числу .....	31
1.3.6. Поиск индекса первого элемента массива с заданными свойствами .....	33
1.4. Задачи на нахождение максимальных (минимальных) элементов массива, их индексов, количеств и т. п. ....	33
1.4.1. Определение максимального элемента массива .....	33
1.4.2. Определение минимального элемента массива .....	35
1.4.3. Определение индекса максимального элемента массива .....	35
1.4.4. Нахождение индекса минимального элемента .....	37
1.4.5. Нахождение минимального (максимального) элемента массива и количества элементов, равных ему .....	38
1.4.6. Нахождение количества минимальных элементов .....	40
1.4.7. Определение минимального значения среди тех элементов массива, которые удовлетворяют некоторому условию .....	40
1.4.8. Определение индекса минимального элемента среди элементов массива, которые удовлетворяют некоторому условию .....	44
1.4.9. Нахождение второго по величине максимального элемента .....	45
1.4.9.1. Поиск элемента массива, который стоял бы на предпоследнем месте, если бы массив был отсортирован по неубыванию .....	45
1.4.10. Нахождение второго минимума .....	49
1.5. Разные задачи .....	50
1.5.1. Обмен значениями переменных величин .....	50
1.5.2. Обмен значениями двух элементов массива .....	51
1.5.3. Перестановка всех элементов массива в обратном порядке .....	51
1.5.4. Рассмотрение всех вариантов сочетания по одному элементу из нескольких наборов .....	53
<b>Глава 2. Задания 11</b> .....	<b>55</b>
2.1. Задание из [2] .....	58
2.2. Задание из [3] .....	59

2.3. Задание из [6] .....	62
2.4. Задание из [4] .....	64
2.5. Задание из [10].....	67
2.6. Задание из [5] .....	70
2.7. Задание из [7]* .....	73
<b>Глава 3. Задания 20</b> .....	<b>78</b>
3.1. Задание из [3] .....	79
3.2. Задание из [2] .....	80
3.3. Задание из [6] .....	81
3.4. Задание из [3] .....	83
3.5. Задание из [7]* .....	84
<b>Глава 4. Задания 21</b> .....	<b>89</b>
4.1. Задание из [13].....	90
4.2. Задание из [5] .....	92
4.3. Задание из [6] .....	94
4.4. Задание из [1] .....	97
4.5. Задание из [2] .....	100
4.6. Задание из [3] .....	101
4.7. Задание из [7]* .....	102
4.8. Задание из [11].....	104
4.9. Задание из [4] .....	106
4.10. Задание из [10].....	110
<b>Глава 5. Задания 24</b> .....	<b>115</b>
5.1. Задание из [3] .....	116
5.2. Задание из [5] .....	119
5.3. Задание из [4] .....	122
5.4. Задание из [10].....	126
5.5. Задание из [7]* .....	130
5.6. Задание из [6] .....	133
<b>Глава 6. Задания 25</b> .....	<b>143</b>
6.1. Задание варианта 3 из [12] .....	144
6.2. Задание варианта 4 из [12] .....	146

6.3. Задание варианта 1 из [11] .....	146
6.4. Задание варианта 1 из [12] .....	147
6.5. Задание варианта 9 из [12] .....	148
6.6. Задание варианта 10 из [12] .....	150
6.7. Задание из [2].....	150
6.8. Задание варианта 8 [13] .....	151
6.9. Задание из [7]* .....	153
6.10. Задание варианта 5 из [12] .....	154
6.11. Задание из [1].....	155
6.12. Задание из [11].....	156
6.13. Задание из [6] .....	156
6.14. Задание из [5].....	157
6.15. Задание из [4].....	157
6.16. Задание варианта 10 из [13].....	158
6.17. Задание варианта 2 из [12].....	159
6.18. Задание варианта 7 из [12] .....	161
6.19. Задание из [3] .....	162
6.20. Задание варианта 2 из [13] .....	164
6.21. Задание варианта 6 из [13] .....	167
6.22. Задание варианта 7 из [13] .....	169
6.23. Задание варианта 5 из [13] .....	171
6.24. Задание варианта 3 из [13] .....	175
6.25. Задание варианта 6 из [12] .....	177
6.26. Задание варианта 8 из [12] .....	179
6.27. Задание варианта 4 из [13] .....	179
<b>Глава 7. Задания 27</b> .....	<b>182</b>
7.1. Задание из [1].....	183
7.1.1. Определение того факта, что некоторая решенная задача уже имеется в списке ранее введенных задач (в массиве задачи) .....	185
7.1.2. Заполнение массива задачи неповторяющимися значениями.....	186
7.1.3. Заполнение массива задачи неповторяющимися значениями и определение «встречаемости» (количества вхождений) каждой задачи.....	187

7.1.4. Сортировка массива кол_задач в порядке невозрастания (и соответственно ей – изменение массива задачи) .....	188
7.2. Задание из [2] .....	190
7.3. Задание из [3] .....	195
7.4. Задание из [4] .....	202
7.5. Задание из [5] .....	214
7.6. Задание из [6] .....	215
7.7. Задание из [7]* .....	221

**Приложение 1. Задания на определение значений переменных величин .....** 225

П1.1. Задания, связанные с линейным алгоритмом .....	226
П1.2. Задания, связанные с разветвляющимся алгоритмом .....	226
П1.3. Задания, связанные с циклическим алгоритмом .....	228
П1.4. Задания на заполнение и изменение одномерного массива .....	231
П1.5. Задания на обработку одномерного массива .....	233
П1.6. Задания на заполнение двух массивов .....	234
П1.7. Задания на заполнение и изменение двумерного массива .....	235

**Приложение 2. Сортировка массива методом обмена .....** 245

**Список литературы .....** 250

В данной главе рассмотрена методика решения задач, предусмотренных Кодификатором элементов содержания и требований к уровню подготовки выпускников общеобразовательных учреждений для проведения в 2018 году единого государственного экзамена по информатике и ИКТ [7], которые используются в демонстрационных вариантах ЕГЭ по информатике последних лет, а также в книгах от разработчиков ЕГЭ [11–13].

## 1.1. Обработка натурального числа

### 1.1.1. Выделение цифр

Задача формулируется так: «Дано натуральное число  $n$ . Вывести его цифры в “столбик”».

*Решение*

Когда количество цифр в заданном числе известно, можно выделить любую его цифру. Но в данном случае это количество неизвестно. Поэтому подумаем над вопросом – какую цифру целого числа (см. схему ниже) определить можно?

<i>первая</i>	<i>вторая</i>	<i>...</i>	<i>предпоследняя</i>	<i>последняя</i>
---------------	---------------	------------	----------------------	------------------

цифры числа

Ответ – последнюю (последняя цифра любого натурального числа равна остатку от деления этого числа на 10 – убедитесь в этом!):

$\text{посл} = \text{mod}(n, 10)$  | *посл* - последняя цифра числа  $n$

где  $\text{mod}$  – функция школьного алгоритмического языка, возвращающая остаток от деления своего первого аргумента на второй (в других языках программирования для этого используется не функция, а специальная операция).

А остальные цифры? Как, например, определить предпоследнюю цифру? Ее можно найти только так: получить число без последней цифры исходного числа (разделив исходное нацело на 10) и для него определить последнюю цифру.

Аналогично можно получить и предпредпоследнюю, и остальные цифры.

Следовательно, для решения задачи в программе надо многократно выполнить следующие действия:

- 1) определить последнюю цифру числа;
- 2) вывести ее на экран;
- 3) получить число без последней цифры.



Соответствующий фрагмент программы:

```
нц пока n > 0:
    посл := mod(n, 10)
    вывод нс, посл
    n := div(n, 10)
кц
```

где `div` – функция школьного алгоритмического языка, возвращающая целую часть частного от деления первого аргумента на второй (в других языках для расчета также используется не функция, а специальная операция).

## Язык Паскаль

```
while n > 0 do
begin
    посл := n mod 10;
    writeln(посл);
    n := n div 10
end;
```

### 1.1.2. Определение суммы цифр числа

Для решения задачи в программе надо многократно выполнить следующие действия:

- 1) определить последнюю цифру числа;
- 2) учесть ее в найденной ранее сумме;
- 3) получить число без последней цифры.

С использованием переменной `сум`, накапливающей в себе сумму уже учтенных цифр, программа решения задачи оформляется следующим образом:

```
вывод нс, "Введите натуральное число "
ввод n
сум := 0 |Начальное значение суммы цифр
нц пока n > 0
    посл := mod(n, 10)
    сум := сум + посл
    n := div(n, 10)
кц
вывод нс, "Сумма цифр этого числа равна ", сум
```

## Язык Паскаль

```
write('Введите натуральное число ');
readln(n);
sum := 0;
```

```
while n > 0 do
  begin
    posl := n mod 10;
    sum := sum + posl;
    n := n div 10
  end;
writeln('Сумма цифр этого числа равна ', sum);
```

Обратим внимание на важное обстоятельство. Хотя в языке программирования Паскаль и в ряде других языков переменной по умолчанию присваивается начальное значение, равное нулю, в программах решения заданий ЕГЭ оператор

```
sum := 0;
```

является обязательным, и его отсутствие, как показывает опыт, рассматривается проверяющими экзаменационные работы как ошибка.

### 1.1.3. Определение произведения цифр числа

Задача решается аналогично предыдущей. Отличия:

- 1) рассчитывается не сумма, а произведение цифр;
- 2) начальное значение искомой величины произв должно быть принято равным 1.

Программа:

```
Вывод нс, "Введите натуральное число "
Ввод n
произв := 1 |Начальное значение
нц пока n > 0
  посл := mod(n, 10)
  произв := произв * посл
  n := div(n, 10)
кц
Вывод нс, "Произведение цифр этого числа равно ", произв
```

## Язык Паскаль

```
write('Введите натуральное число ');
readln(n);
proizv := 1;
while n > 0 do
  begin
    posl := n mod 10;
    proizv := proizv * posl;
    n := n div 10
  end;
writeln('Произведение цифр этого числа равно ', proizv);
```

### 1.1.4. Определение количества цифр числа

Здесь следует использовать переменную-«счетчик» количества уже «обработанных» цифр:

```
вывод нс, "Введите натуральное число "  
ввод n  
кол := 0 |Начальное значение количества цифр  
нц пока n > 0  
    посл := mod(n, 10)  
    кол := кол + 1  
    n := div(n, 10)  
кц  
вывод нс, "Количество цифр этого числа равно ", кол
```

Обратим внимание на то, что последнюю цифру `посл` можно не определять.

#### Язык Паскаль

```
write('Введите натуральное число ');  
readln(n);  
kol := 0;  
while n > 0 do  
    begin  
        kol := kol + 1;  
        n := n div 10  
    end;  
writeln('Количество цифр этого числа равно ', kol);
```

Здесь также присваивание начального значения, равного нулю, является обязательным (см. п. 1.1.2).

### 1.1.5. Определение максимальной цифры числа

Алгоритм решения этой задачи аналогичен алгоритму действий человека, который определяет максимальную цифру в некотором числе:

**324086312**

– сначала он запоминает первую цифру, а затем рассматривает вторую. Если она больше того числа, которое помнил, то запоминает вторую цифру и переходит к следующей, третьей цифре, в противном случае просто переходит к следующей цифре и делает то же самое. В нашей программе единственное отличие – в том, что «просматривать» (выделять и сравнивать) цифры мы будем, начиная с последней.

В приведенной далее программе искомое максимальное значение хранит переменная макс.

```

вывод нс, "Введите натуральное число "
ввод n
посл := mod(n, 10) |Выделяем последнюю цифру
макс := посл      |Принимаем ее в качестве максимальной
n := div(n, 10)   |Отбрасываем последнюю цифру
|Рассматриваем остальные цифры
нц пока n > 0
  посл := mod(n, 10) |Выделяем
  если посл > макс  |Сравниваем
    то              |и при необходимости
      макс := посл  |меняем значение макс
  все
  n := div(n, 10)
кц
вывод нс, "Максимальная цифра заданного числа равна ", макс

```

В данном случае (см. п. 1.4.1) отдельно последнюю цифру можно не выделять, так как можно в качестве начального значения переменной макс принять 0:

```

вывод нс, "Введите натуральное число "
ввод n
макс := 0
|Рассматриваем все цифры
нц пока n > 0
  посл := mod(n, 10)
  если посл > макс
    то
      макс := посл
  все
  n := div(n, 10)
кц
вывод нс, "Максимальная цифра заданного числа равна ", макс

```

## Язык Паскаль

```

write('Введите натуральное число ');
readln(n);
max := 0;
while n > 0 do
  begin
    посл := n mod 10;
    if посл > max then
      max := посл;
    n := n div 10
  end;
writeln('Максимальная цифра заданного числа равна ', max);

```

### 1.1.6. Определение минимальной цифры числа

Задача решается аналогично предыдущей (начальное значение искомой переменной `мин` можно принять равным 9).

## 1.2. Операции с элементами массива, отобранными по некоторому условию<sup>1</sup>

Для каждой рассмотренной задачи в данном разделе приведен фрагмент программы ее решения на школьном алгоритмическом языке и на языке Паскаль. Используются следующие основные величины:

- `a` – имя массива;
- `n` – общее количество элементов массива (условно принято, что нумерация элементов массива начинается с 1).

Смысл остальных величин можно легко определить по их именам.

Принято, что элементы массива имеют целые значения.

### 1.2.1. Изменение элементов массива с заданными свойствами (удовлетворяющих некоторому условию)

Общая формулировка задач рассматриваемого типа: «Все элементы массива с заданными свойствами заменить на ...», где вместо многоточия указывается конкретное значение, на которое следует заменить указанные элементы, или правило, по которому надо провести замену.

*Примеры задач*

1. Дан массив целых чисел. Все четные элементы заменить числом 666.
2. Дан массив чисел, среди которых есть отрицательные. Все отрицательные элементы уменьшить на 10.
3. Дан массив целых чисел. Все элементы с нечетным индексом, оканчивающиеся цифрой 5, увеличить на 1.

---

<sup>1</sup> Такая формулировка приведена в Кодификаторе элементов содержания и требований к уровню подготовки выпускников общеобразовательных учреждений для проведения в 2018 году единого государственного экзамена по информатике и ИКТ [7].

### Решение

```
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
если <условие>
  то
  |Меняем его значение
  a[i] := ...
все
кц
```

где <УСЛОВИЕ> – заданное условие. Это условие может определяться значением элемента массива  $a[i]$  или/и его индексом  $i$  (см., например, выше пример 3 задачи).

### Язык Паскаль

```
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие>
  then {Меняем его значение}
  a[i] := ...;
```

#### 1.2.2. Нахождение суммы элементов массива с заданными свойствами (удовлетворяющих некоторому условию)

##### Примеры задач рассматриваемого типа

1. В массиве записана стоимость 30 предметов. Определить общую стоимость тех предметов, которые стоят менее 200 руб.
2. В массиве записаны 20 целых чисел. Определить сумму тех из них, которые оканчиваются нулем.
3. В массиве записаны данные о количестве осадков, выпавших за каждый день января. Определить общее количество осадков, выпавших второго, четвертого, шестого и т. д. числа этого месяца. В программу должны вводиться данные за каждый день месяца.

##### Решение

Обсудим сначала более простую задачу: «Найти сумму всех элементов массива».

С использованием переменной *сум*, накапливающей в себе сумму значений уже рассмотренных элементов массива, фрагмент программы решения задачи оформляется следующим образом:

```

сум := 01
нц для i от 1 до n
  сум := сум + a[i]
кц

```

Вернемся теперь к «основной» задаче. Отличие задач данного типа от только что рассмотренной – в том, что добавлять значение элемента массива к уже рассчитанной ранее сумме следует только тогда, когда элемент обладает заданными свойствами:

```

сум := 0
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
  если <условие>
    то
      |Учитываем его значение в сумме
      сум := сум + a[i]
  все
кц

```

В данном случае <условие> также может определяться значением элемента массива  $a[i]$  или/и его индексом  $i$ .

Однако так можно оформить программу только в случае, когда известно, что в массиве имеется хотя бы один элемент с заданными свойствами. Если допускается, что таких элементов может не быть, то фрагмент, связанный с выводом ответа, должен иметь вид:

```

если сум = 0
  то
    вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
  иначе
    вывод нс, "Сумма чисел, удовлетворяющих условию, равна ", сум
все

```

*Примечание.* Случай, когда элементы с заданными свойствами в массиве имеются, но их сумма равна нулю, не учитывается.

## Язык Паскаль

```

sum := 0;
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие>

```

<sup>1</sup> Еще раз напомним, что присваивание переменной начального значения, равного нулю, в программах решения заданий ЕГЭ является обязательным.

```

    then {Учитываем его значение в сумме}
        sum := sum + a[i];
if sum = 0
then write('Чисел, удовлетворяющих условию, в массиве нет')
else write('Сумма чисел, удовлетворяющих условию, равна ', sum);

```

### 1.2.3. Нахождение количества элементов массива с заданными свойствами

*Примеры задач рассматриваемого типа*

1. В массиве хранятся 40 целых чисел. Найти количество четных чисел.
2. В массиве записан рост 22 юношей. Определить, сколько из них имеют рост менее 165 см.
3. В массиве записаны оценки по информатике каждого из 25 учеников класса. Определить количество пятерок.

Особенность задач данного типа – в том, что в случае, когда элемент обладает заданными свойствами (удовлетворяет некоторому условию), искомое количество увеличивается на 1:

```

кол := 0
нц для i от 1 до n
    |Если элемент обладает заданными свойствами
    если <условие>
        то
            |Учитываем его в искомом количестве
            кол := кол + 1
    все
кц
вывод нс, "Количество чисел, удовлетворяющих условию, равно ", кол

```

В данном случае условие в команде **если** (в условном операторе) определяется значением элемента массива  $a[i]$  или одновременно значениями  $a[i]$  и  $i$ . Количество элементов, зависящих только от значения индекса  $i$ , может быть найдено без использования оператора цикла (убедитесь в этом!).

## Язык Паскаль

```

kol := 0;
for i := 1 to n do
    {Если элемент обладает заданными свойствами}
    if <условие>
        then {Учитываем его в искомом количестве}
            kol := kol + 1; {или inc(kol)}
write('Количество чисел, удовлетворяющих условию, равно ', kol);

```



### 1.2.4. Нахождение среднего арифметического значения элементов массива с заданными свойствами

Для нахождения искомого значения необходимо определить сумму элементов массива с заданными свойствами и их количество. Такие две задачи мы уже решили ранее. Здесь их можно объединить в одном операторе цикла:

```
сум := 0
кол := 0
нц для i от 1 до n
  |Если элемент обладает заданными свойствами
  если <условие>
    то
      |Учитываем его значение в сумме
      сум := сум + a[i]
      |и учитываем этот элемент в количестве
      кол:= кол + 1
  все
кц
|Подсчет результата
сред_арифм := сум/кол
```

Обратите внимание на то, что многократно определять значение `сред_арифм` в «теле» условного оператора (команды `если`) необходимости нет. Это можно сделать один раз после окончания оператора цикла. Однако может оказаться, что чисел, удовлетворяющих заданному условию, в массиве не окажется. В этом случае при расчете будет иметь место деление на ноль, что недопустимо. Правильное оформление:

```
...
|Подсчет и вывод результата
если кол > 0
  то
    средн_ариф := сум/кол
    вывод нс, "Среднее арифметическое: ", сред_арифм
  иначе
    вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
все
```

## Язык Паскаль

```
sum := 0;
kol := 0;
for i := 1 to n do
  {Если элемент обладает заданными свойствами}
  if <условие> then
```

```
begin
    {Учитываем его значение в сумме}
    sum := sum + a[i];
    {и учитываем этот элемент в количестве}
    kol := kol + 1
end;
{Подсчет и вывод результата}
if kol > 0 then
begin
    sred_arifm := sum/kol;
    write('Среднее арифметическое: ', sred_arifm:7:2)
end
else
write('Чисел, удовлетворяющих условию, в массиве нет');
```

*Примечание.* В ряде случаев в заданиях ЕГЭ указывается требование о разработке программы, эффективной с точки зрения используемой памяти. Для учета этого требования величину `сред_арифм` можно не использовать, а включить в команду **вывод** выражение для ее расчета:

```
если кол > 0
то
    вывод нс, "Среднее арифметическое: ", сумма/кол
иначе
    вывод нс, "Чисел, удовлетворяющих условию, в массиве нет"
все
```

### 1.2.5. Нахождение максимального количества подряд идущих элементов массива, обладающих заданными свойствами

*Пример задачи:* «Определить максимальное количество подряд идущих четных элементов в заданном целочисленном массиве».

В дальнейшем для краткости будем называть участок массива с указанными элементами «подмассивом». Значит, для решения обсуждаемой задачи надо определить максимальное из чисел – длин каждого подмассива (количество элементов в них). Это можно сделать по аналогии с определением максимальной цифры натурального числа (см. п. 1.1.5), только вместо цифр следует использовать длину каждого подмассива.

Используем в программе следующие основные величины:

- `кол` – число элементов в текущем подмассиве;
- `макс_кол` – искомое значение (максимальное количество подряд идущих элементов с заданными свойствами).

Можно рассуждать так. Если очередной элемент обладает заданными свойствами, то подмассив таких элементов продолжится или начался – увеличиваем его длину `кол` на 1, иначе – текущий подмассив закончился, и в этом случае:

- 1) сравниваем его длину с максимальной длиной уже рассмотренных ранее подмассивов `макс_кол`. Если длина текущего подмассива больше, то принимаем ее в качестве нового значения величины `макс_кол`;
- 2) имея в виду обработку следующего подмассива, обнуляем значение величины `кол`.

Фрагмент программы на основе сделанных рассуждений:

```
кол := 0
макс_кол := 0
нц для i от 1 до n
  если <условие>
    то
      |Подмассив продолжается или начался
      |Увеличиваем его длину на 1
      кол := кол + 1
  иначе |Встретился элемент, не обладающий заданными свойствами
      |Текущий подмассив закончился
      |Сравниваем его длину со значением макс_кол
      если кол > макс_кол
        то
          макс_кол := кол
      все
      кол := 0 /Новое значение
все
кц
```

Однако при таких рассуждениях в случае, когда последний,  $n$ -й элемент массива также обладает заданными свойствами (например, четный), последний подмассив не будет учтен. Значит, нужно дополнительно учесть и его, сравнив длину этого подмассива с максимальной длиной:

```
|Проверяем длину последнего (возможного) подмассива
если кол > макс_кол
  то
    макс_кол := кол
все
|Выводим ответ
вывод нс, макс_кол
```

Обратим внимание на то, что после окончания текущего подмассива величине `кол` присваивается нулевое значение независимо от результата сравнения величин `кол` и `макс_кол`.

**Язык Паскаль**

```
kol := 0;
max_kol := 0;
for i := 1 to n do
  if <условие>
  then {Подмассив продолжается или начался.
        Увеличиваем его длину на 1}
    kol := kol + 1
  else {Встретился элемент, не обладающий
        заданными свойствами, - текущий подмассив закончился.
        Сравниваем его длину со значением max_kol}
    begin
      if kol > max_kol then max_kol := kol;
      kol := 0 {Новое значение}
    end;
{Проверяем длину последнего (возможного) подмассива}
if kol > max_kol
then max_kol := kol;
{Выводим ответ}
write(max_kol);
```

Возможны также задачи рассмотренного типа, в которых начальное значение величины `kol` и ее значение после окончания некоторого подмассива принимаются равными 1. Это имеет место, когда первый элемент подмассива также учитывается в его длине.

*Пример:* «Определить максимальное количество подряд идущих совпадающих элементов целочисленного массива».

Ясно, что здесь первый из совпадающих элементов также должен быть учтен. Поэтому начальные значения переменных величин, хранящих длину текущего подмассива и максимальную длину подмассивов, равны 1 (если потом выяснится, что во всех парах находящихся рядом элементов массива нет одинаковых, то искомая величина будет равна 1):

```
кол_совп := 1
макс_совп := 1
нц для i от 2 до n
  если m[i] = m[i - 1]
  то
    кол_совп := кол_совп + 1
  иначе
    если кол_совп > макс_совп
    то
      макс_совп := кол_совп
  все
  кол_совп := 1 /Новое значение
все
```