УДК 373.167.1:004+004(075.3)
ББК 32.97я72
З67

Златопольский Д. М.

З67 Подготовка к ЕГЭ по информатике в компьютерной форме. – М.: ДМК Пресс, 2021. – 304 с.: ил.

ISBN 978-5-97060-896-8

Книга предназначена для самостоятельной подготовки учащихся к единому государственному экзамену по информатике и ИКТ, который начиная с 2021 года будет проходить в компьютерной форме. Согласно демонстрационному варианту ЕГЭ 2021 года, в содержании экзамена будет существенно увеличено количество заданий, связанных с алгоритмизацией и программированием. Таким задачам в книге уделяется особое внимание. Обсуждаются и методики выполнения других заданий, представленных в демонстрационном варианте. В приложениях приведены вспомогательные материалы, связанные с заданиями ЕГЭ. Кроме учащихся старших классов, готовящихся к сдаче экзамена, книгу могут использовать преподаватели информатики, а также студенты и школьники, желающие углубить общие знания по информатике и ИКТ.

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

ISBN 978-5-97060-896-8

Содержание

Предисловие

Книга, которую вы читаете, является первым и на конец 2020 года единственным пособием по подготовке к единому государственному экзамену по информатике и ИКТ, который, начиная с 2021 года, будет проходить в компьютерной форме [1].

Как показывает анализ проекта демонстрационного варианта ЕГЭ 2021 года, в содержании экзамена существенно увеличится количество заданий, связанных с алгоритмизацией и программированием. Так, в нем таких заданий 12 при общем числе заданий – 27. При этом весомость заданий 25, 26 и 27 составляет не 1, а 2 балла (общий максимальный первичный балл за весь экзамен – 30).

Это говорит о том, что от умения решать задачи по алгоритмизации и программированию в значительной степени зависит успешность сдачи ЕГЭ в целом. В то же время, как показывает опыт, такие задачи часто вызывают у школьников заметные трудности. В большой степени это связано с недостаточным числом часов, отводимых на изучение алгоритмизации и программирования в школе.

Данная книга должна восполнить этот недостаток – помочь учащимся подготовиться к экзамену самостоятельно. В ней системно, подробно и доступно описана методика выполнения заданий по программированию и алгоритмизации, которые могут встретиться на экзамене.

Содержание книги основано на материалах, представленных в [1] и [2][1], и информации, полученной автором от коллег из регионов России, в которых проводился эксперимент по проведению ЕГЭ по информатике в компьютерной форме.

Сначала в главе 1 книги обсуждены все частные, вспомогательные задачи, умение решать которые позволит успешно выполнить задания экзамена. Общая методика выполнения заданий, связанных с алгоритмизацией и программированием, а также ряда других заданий, в том числе нового типа, описана в главе 2. В заключительной главе обсуждены методики выполнения соответствующих заданий экзамена, представленных в [1].

---

[1] Пользуясь случаем, автор выражает благодарность Константину Юрьевичу Полякову и его коллегам за опубликованную информацию.

В приложениях приведены другие материалы, связанные с заданиями ЕГЭ.

При разработке программ (частных и из заданий ЕГЭ) использован школьный алгоритмический язык (система программирования КуМир [3]). Русский синтаксис этого языка и большое число комментариев делают приведенные программы максимально понятными и легко переносимыми на любой другой язык. Это означает, что книга может быть использована читателями, владеющими любым языком программирования. В большинстве случаев после разбора методики решения и программы на школьном алгоритмическом языке приводятся также соответствующие программы на популярных среди школьников языках Python и Паскаль.

Кроме учащихся, готовящихся к сдаче экзамена самостоятельно, книгу могут использовать учителя и преподаватели информатики, а также студенты и учащиеся, желающие углубить свои знания по информатике вне связи с ЕГЭ.

Глава 1

Вспомогательные задачи

В данной главе рассмотрена методика решения задач, предусмотренных Кодификатором элементов содержания и требований к уровню подготовки выпускников общеобразовательных учреждений для проведения в 2021 году единого государственного экзамена по информатике и ИКТ [4], а также задач, которые используются в демонстрационных вариантах экзамена 2021 года [1] и нескольких последних лет [5–10].

1.1. Обработка натурального числа

1.1.1. Выделение цифр

Задача формулируется так: «Дано шестизначное[1] натуральное число n. Вывести его цифры в столбик, начиная с последней. Оператор цикла не использовать».

Решение

Первую цифру шестизначного числа можно найти, разделив заданное число на 100 000:

```
цифра1 := div(n, 100000),
```

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).

Последняя цифра равна остатку от деления на 10:

```
цифра6 := mod(n, 10),
```

где mod – функция школьного алгоритмического языка, возвращающая остаток от деления своего первого аргумента на второй (в других языках программирования для этого также используется специальная операция).

Остальные цифры одним действием определить нельзя.

Приведем программу, в которой используются однотипные формулы:

```
алг
нач цел n, цифра1, цифра2, цифра3, цифра4, цифра5, цифра6
  ввод n
  цифра6 := mod(n, 10); n := div(n, 10)
  цифра5 := mod(n, 10); n := div(n, 10)
```

[1] Или любой другой заданной «значности».

```
  цифра4 := mod(n, 10); n := div(n, 10)
  цифра3 := mod(n, 10); n := div(n, 10)
  цифра2 := mod(n, 10); n := div(n, 10)
  цифра1 := mod(n, 10)
  вывод нс, цифра6
  вывод нс, цифра5
  …
  вывод нс, цифра1
кон
```

1.1.2. Выделение цифр. Вариант 2

Задача формулируется так: «Дано натуральное число n. Вывести его цифры в столбик, начиная с последней».

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?

первая	*вторая*	…	*предпоследняя*	*последняя*

цифры числа

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

```
посл = mod(n, 10) | посл - последняя цифра числа n
```

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.

Но сколько раз надо повторить указанные действия? Неизвестно (неизвестно число разрядов в заданном числе n), т. е. оператор цикла с параметром использовать нельзя. Надо применить оператор цикла с предусловием.

Соответствующий фрагмент программы:

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

Язык Паскаль	Язык Python

`while n > 0 do`
`  begin:`
`  posl := n mod 10;`
`  writeln(posl);`
`  n := n div 10`
`  end;`
	`while n > 0:`
	`  posl = n % 10`
	`  print(posl)`
	`  n = n//10`

1.1.3. Определение суммы цифр

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	Язык Python
… `sum := 0;`	… `sum = 0`

```
  while n > 0 do
    begin
      posl := n mod 10;
      sum := sum + posl;
      n := n div 10
    end;
  …
```

```
while n > 0:
  posl = n % 10
  sum = sum + posl
  n = n//10
…
```

1.1.4. Определение произведения цифр

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	Язык Python

```
  …
  proizv := 1;
  while n > 0 do
    begin
      posl := n mod 10;
      proizv := proizv * posl;
      n := n div 10
    end;
  …
```

```
…
proizv = 1
while n > 0:
  posl = n % 10
  proizv = proizv * posl
  n = n//10
…
```

1.1.5. 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	Язык Python
…	…
kol := 0;	kol = 0
while n > 0 do	while n > 0:
begin	kol = kol + 1
kol := kol + 1;	n = n//10
n := n div 10	…
end;	
…	

1.1.6. Выделение цифр шестизначного натурального числа и запись их в массив (в программе на языке Python – в список)

Решение

Здесь после выделения каждой очередной цифры следует записать ее в массив:

```
алг
нач цел n, к, посл, цел таб цифры[1:6]
  ввод n
  нц для к от 1 до 6
    | Определяем последнюю цифру
    посл := mod(n, 10)
    | и записываем ее в массив
    цифры[к] := посл
    | Отбрасываем последнюю цифру числа n
    n := div(n, 10)
  кц
  …
кон
```

Можно определение цифры и запись ее в массив провести одним оператором:

```
    цифры[к] := mod(n, 10)
```

Язык Паскаль	Язык Python

`var n: longint; posl, k: byte;`
`  tsifri: array[1..6] of byte,`
`BEGIN`
`readln(n);`
`kol := 0;`
`for k := 1 to 6 do`
`  begin`
`  posl := n mod 10;`
`  tsifri[k] := posl;`
`  n := n div 10`
`  end;`
…
	`# Инициация и начальное`
	`# заполнение списка с цифрами`
	`tsifri = [0, 0, 0, 0, 0, 0]`
	`# Можно использовать генератор списков`
	`# tsifri = [0 for k in range(6)]`
	`n = int(input())`
	`for k in range(6):`
	`  posl = n % 10`
	`  tsifri[k] = posl`
	`  n = n//10`
	…

В программе на языке Python можно также использовать метод `append()`:

```
# Инициация списка
tsifri = [] # Пустой список
n = int(input())
for k in range(6):
  # Определяем последнюю цифру
  posl = n % 10
  # Добавляем ее в конец списка
  tsifri.append(posl)
  # Отбрасываем последнюю цифру числа n
  n = n//10
```

Нетрудно убедиться, что цифры числа записываются в массив в обратном порядке – последняя цифра записана в первом элементе массива[1], предпоследняя – во втором и т. д. Если это нежелательно, то можно обеспечить «естественный» порядок записи цифр:

В программе на языке Python можно также после заполнения списка в обратном порядке перевернуть элементы с помощью метода `reverse()`:

```
…
tsifri.reverse()
```

[1] В программах на языке Python, в котором нумерация элементов списка начинается с нуля, последняя цифра будет записана в элементе с индексом 0.

1.1.7. 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 он 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, так как диапазон возможных значений цифр известен, отдельно последнюю цифру можно не выделять, а в качестве начального значения переменной макс принять 0 (см. п. 1.4.1):

```
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

```
…
max := 0;
while n > 0 do
  begin
    posl := n mod 10;
    if posl > max then
      max := posl;
    n := n div 10
  end;
…
```

Язык Python

```
…
max = 0
while n > 0:
  posl = n % 10
  if posl > max:
    max = posl
  n = n//10
…
```

1.1.8. Определение минимальной цифры числа

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

1.1.9. Определение количества делителей числа

Прежде чем представлять методику решения задачи, заметим, что установить, является ли некоторое число b делителем числа a, можно, определив остаток от деления a на b. Если этот остаток равен нулю, то ответ положительный, в противном случае – отрицательный:

```
если mod(a, b) = 0
  то
    число b является делителем числа a
  иначе
    число b не является делителем числа a
все
```

Самый простой способ определить количество делителей числа n – это проверить по очереди делимость n на каждое из чисел 1, 2, 3, …, n и в случае кратности n проверяемому числу увеличить значение некоторой переменной-счетчика.

В приведенной ниже программе, кроме переменной n, использованы следующие переменные величины:

кол_дел – искомое количество делителей (переменная-счетчик);
возм_дел – проверяемые числа.

```
алг Определение_количества_делителей
нач цел n, кол_дел, возм_дел
ввод n
кол_дел := 0
нц для возм_дел от 1 до n
  если mod(n, возм_дел) = 0
    то  | возм_дел - делитель числа n
      кол_дел := кол_дел + 1
  все
кц
вывод нс, кол_дел
кон
```

Язык Паскаль

```
…
readln(n);
kol_del := 0;
for vozm_del := 1 to n do
  if n mod vozm_del = 0 then
    kol_del := kol_del + 1;
writeln(n)
```

Язык Python

```
n = int(input())
kol_del = 0
for vozm_del in range(1, n + 1):
  if n % vozm_del == 0:
    kol_del = kol_del + 1
print(kol_del)
```

Возможен ряд усовершенствований программы, обеспечивающих ускорение ее работы. Одно их них учитывает тот факт, что в интервале от n/2 + 1 до n – 1 делителей числа n нет, т. е. количество проверок можно сократить вдвое:

```
кол_дел := 0
нц для возм_дел от 1 до div(n, 2)
  если mod(n, возм_дел) = 0
    то
      кол_дел := кол_дел + 1
  все
кц
| Учитываем также число n
кол_дел := кол_дел + 1
```

В описанном варианте можно при начальном присваивании переменной кол_дел сразу учесть делитель, равный n:

```
кол_дел := 1
…
```

а после оператора цикла не менять значение переменной кол_дел.

Язык Паскаль	Язык Python
…	n = int(input())
readln(n);	kol_del = 1
kol_del := 1;	for vozm_del in range(1, n//2 +
for vozm_del := 1 to n div 2 do	1):
if n mod vozm_del = 0 then	if n % vozm_del == 0:
kol_del := kol_del + 1;	kol_del = kol_del + 1
writeln(kol_del);	print(kol_del)
…	

Обратим внимание на то, что в общем количестве делителей учитываются также значения 1 и n.

1.1.10. Поиск делителей натурального числа и сохранение их в массиве

Пусть имя массива для записи делителей – делители. Так как количество делителей заданного числа неизвестно, то в программе на школьном алгоритмическом языке, языке Паскаль и ряде других размер этого массива придется определить с запасом (примем, что указанное количество не превышает 10 000).

Программа:

```
алг Поиск_и_сохранение_делителей
нач цел n, возм_дел, кол_дел, цел таб делители[1 : 10000]
  ввод n
  кол_дел := 0
  нц для возм_дел от 1 до n
    если mod(n, возм_дел) = 0
      то
        | Найден очередной делитель
        кол_дел := кол_дел + 1
        | Записываем его в массив
        делители[кол_дел] := возм_дел
    все
  кц
  | Все делители (их количество – кол_дел) записаны в массив
…
```

Язык Паскаль

```
var n, kol_del: longint;
   deliteli: array[1..6] of
longint;
BEGIN
readln(n);
kol_del := 0;
for vozm_del := 1 to n do
  if n mod vozm_del = 0 then
    begin
      kol_del := kol_del + 1;
      deliteli[] := vozm_del
    end;
…
```

Язык Python

```
n = int(input())
deliteli = []
for vozm_del in range(1, n + 1):
  if n % vozm_del == 0:
    deliteli.append(vozm_del)
…
```

Здесь также можно сократить количество проверяемых чисел.

1.1.11. Проверка числа на «простоту»

Как известно, *простыми* называют числа, которые имеют только два делителя: самого себя и 1. Такими являются числа 2[1], 3, 5, 7, 11, 13, 17, 19, 23, 31, 37, 41… .

Из определения простого числа вытекает методика решения указанной задачи – для получения ответа следует определить количество делителей и сравнить его с числом 2. Задача подсчета количества делителей была рассмотрена в п. 1.1.9. С учетом приведенного там простейшего варианта программы программа решения нашей задачи оформляется следующим образом:

```
алг
нач цел n, возм_дел, кол_дел
  ввод n
  | Подсчет делителей числа n
  кол_дел := 0
  нц для возм_дел от 1 до n
    если mod(n, возм_дел) = 0
      то
        кол_дел := кол_дел + 1
    все
  кц
  | Проверка
```

[1] По соглашению в теории чисел число 1 простым не является.

```
  если кол_дел = 2
    то
      вывод “Это число простое”
    иначе
      вывод “Это число простым не является”
  все
кон
```

Язык Паскаль

```
var n, kol_del: longint;
BEGIN
readln(n);
…
kol_del := 0;
for vozm_del := 1 to n do
  if n mod vozm_del = 0 then
    kol_del := kol_del + 1;
if kol_del = 2 then
  writeln('Это число простое')
else
  …
```

Язык Python

```
n = int(input())
kol_del = 0
for vozm_del in range(1, n + 1):
  if n % vozm_del == 0:
    kol_del = kol_del + 1
if kol_del == 2:
  print(«Это число простое»)
else:
  …
```

Заметим, что если в задаче происходит проверка на «простоту» большого количества чисел и/или проверяемые числа достаточно большие, то с целью ускорения работы программы целесообразно применять усовершенствованные методы подсчета количества делителей, один из которых описан в п 1.1.9.

Можно также прекратить подсчет делителей, когда их количество станет равно трем:

```
алг
нач цел n, возм_дел, кол_дел
  ввод n
  кол_дел := 0
  нц для возм_дел от 1 до n
    если mod(n, возм_дел) = 0
      то
        кол_дел := кол_дел + 1
        если кол_дел = 3
          то | Прекращаем подсчет делителей
            выход
        все
    все
```

```
  кц
  если кол_дел = 2
    то
      …
кон
```

Язык Паскаль	Язык Python
…	…
kol_del := 0;	kol_del = 0
for vozm_del := 1 to n do	for vozm_del in range(1, n + 1):
begin	if n % vozm_del == 0:
if n mod vozm_del = 0 then	kol_del = kol_del + 1
kol_del := kol_del + 1;	if kol_del == 3:
if kol_del = 3 then break;	break
if kol_del = 2 then	if kol_del == 2:
…	…

Задачи для самостоятельной работы

1. Дано натуральное число. Определить:

   а) сумму его четных делителей;
   б) количество его нечетных делителей;
   в) количество его делителей, больших числа d.

2. Определить количество простых чисел (см. выше) в заданном диапазоне натуральных чисел.

3. Натуральное число называется совершенным, если оно равно сумме своих делителей, включая 1 и, естественно, исключая это самое число. Например, число 6 – совершенное (6 = 1 + 2 + 3). Дано натуральное число. Выяснить, является ли оно совершенным.

4. Дано натуральное число. Определить медиану всех его делителей. *Медианой ряда чисел* называется число, стоящее посередине упорядоченного по возрастанию ряда (в случае, если количество чисел нечетное). Если же количество чисел в ряду четно, то медианой ряда является полусумма двух стоящих посередине чисел упорядоченного по возрастанию ряда. Например, для чисел 5, 2, 18, 8, 3 медиана равна 5, для чисел 5, 17, 3, 9, 14, 2 – 7.