

«Мир информатики»

Журнал для тех, у кого информатика – любимый школьный предмет

Выпуск № 68, июнь 2022 г.

Спасибо вам, уважаемые коллеги!

За активную работу с учащимися по материалам интернет-журнала “Мир информатики” издательство «Образование и информатика» и редакция журнала “Информатика в школе” выражают благодарность:

- Балка Ольге Олеговне, Ставропольский край, г. Зеленокумск, школа № 11;
- Брунову Александру Сергеевичу, средняя школа села Ириновка, Новобурасский р-н Саратовской обл.;
- Букиной Елене Юрьевне, г. Зеленокумск Ставропольского края, школа № 1;
- Букиной Елене Юрьевне, г. Зеленокумск Ставропольского края, школа № 2;
- Вознесенской Марине Васильевне, Иркутская обл., г. Слюдянка, школа № 2
- Волковой Татьяне Петровне, Владимирская обл., г. Струнино, школа № 11;
- Волкову Андрею Юрьевичу, Владимирская обл., г. Александров, школа № 3;
- Волкову Юрию Павловичу, Владимирская обл., г. Струнино, школа № 11;
- Дгебуадзе З.А., Ставропольский край, г. Зеленокумск, школа № 14;
- Дикову Андрею Валентиновичу, г. Пенза, школа № 73;
- Елистратовой Анжелике Александровне, Московская обл., г. Мытищи, ЧУОШ «Логос М»;
- Зудину Василию Павловичу, Ардатовский областной многопрофильный техникум, поселок Ардатов Нижегородской обл.;
- Каменецкой Татьяне Семёновне, Москва, школа № 1530 «Школа Ломоносова»;
- Козыревой Людмиле Владимировне, Ставропольский край, поселок Селивановка, основная школа № 16;
- Комбаровской Светлане Ивановне, г. Воронеж, лицей № 2;
- Косенко Елене Викторовне, Ставропольский край, г. Зеленокумск, школа № 12;
- Лукьяновой Наталии Владимировне, Барнаульский государственный педагогический колледж;
- Муравьевой Ольге Викторовне, средняя школа деревни Муравьево, Вологодская обл.;
- Пичугину Виталию Владимировичу, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н;
- Порываевой Нафисе Сабитовне, Республика Татарстан, Мамадышский р-н, совхоз “Мамадышский”, политехнический колледж;
- Савченко Елене Александровне, Ставропольский край, село Солдато-Александровское, школа № 6;
- Тощевой Татьяне Валентиновне, Владимирская обл., г. Карabanово, школа № 7 им. А.П. Чулкова;
- Фирсовой Марии Николаевне, Волгоградская обл., г. Фролово, школа № 1 имени А.М. Горького;
- Цапиной Таисии Александровне, Владимирская обл., Александровский р-н, г. Карabanово, школа № 7;
- Ярцевой Ольге Владимировне, г. Ярославль, школа № 33.

Методика выполнения варианта 15.2 задания 15 демонстрационных вариантов ОГЭ по информатике

1. Общие вопросы

В Спецификации контрольных измерительных материалов для проведения в 2022 году ОГЭ по информатике в качестве проверяемого результата обучения применительно к варианту 15.2 задания 15 указывается «Умение создавать и выполнять программы на универсальном языке программирования».

Темы в федеральном компоненте ГОС ООО – «Алгоритм, свойства алгоритмов, способы записи алгоритмов. Блок-схемы. Представление о программировании. Алгоритмические конструкции. Логические значения, операции, выражения. Разбиение задачи на подзадачи, вспомогательный алгоритм».

Уровень сложности – высокий.

Максимальный балл за задание – 2.

Примерное время выполнения задания (мин.) – 45.

В демонстрационных вариантах контрольных измерительных материалов ОГЭ по информатике нескольких последних лет и в открытом банке заданий ОГЭ представлены задания на разработку программ, связанных с определением количественных характеристик заданной последовательности натуральных чисел (нахождение их суммы, среднего арифметического, количества чисел, удовлетворяющих некоторому условию, и др.). В общем виде задания имеют вид:

«Напишите программу, которая в последовательности натуральных чисел определяет *<указывается, что следует определить>*. Программа получает на вход количество чисел в последовательности, а затем сами числа. Количество чисел не превышает *<указывается значение>*. Введённые числа не превышают *<указывается значение>* (или «Введённые числа по модулю не превышают *<указывается значение>*»). Программа должна вывести одно число – *<указывается значение-результат>*.

Пример работы программы:

Входные данные	Выходные данные
4	2
16	
7	
26	
77	

Возможен также вариант задания, в котором количество введённых чисел неизвестно, а известно, что последовательность чисел заканчивается числом 0 (0 – признак окончания ввода, не входит в последовательность). Пример работы программы решения такой задачи:

Входные данные	Выходные данные
36	2
91	
40	
0	

2. Типовые задачи и методика их решения

Опишем методику разработки программ решения ряда типовых задач с указанными особенностями. Начнем с варианта, в котором количество чисел последовательности известно (задается первым).

Для каждой рассмотренной задачи приведён фрагмент программы её решения на школьном алгоритмическом языке.

Использованы следующие основные величины:

N – общее количество чисел в последовательности;

a – очередное число последовательности;

$ном$ – порядковый номер очередного числа последовательности.

Смысл остальных величин можно легко определить по их именам.

При решении задач можно выделить четыре основных этапа:

1. Ввод количества N чисел в последовательности:

```
ввод N
```

2. Принятие начальных значений переменных, используемых при расчётах.

3. Ввод и обработка каждого отдельного числа последовательности. Так как количество обрабатываемых чисел N на этом этапе уже известно, то целесообразно использовать оператор цикла с параметром:

```
нц для ном от 1 до N
  ввод a
  | Обработка очередного числа
кц
```

4. Вывод ответа

2.1. Типовая задача 1 «Суммирование всех чисел последовательности»

Пусть требуется найти сумму всех 10 чисел последовательности: 5, 3, 1, 5, 12, 6, 3, 3, 8, 12.

Используем в программе переменную $сум$, которая будет хранить сумму чисел среди уже введенных чисел.

Составим таблицу:

Порядковый номер числа	Значение	Сумма
1	5	5
2	3	8
...		
7		$сум$
8	a	?

Как определить сумму чисел после ввода некоторого очередного числа n , если информацию о такой сумме среди уже рассмотренных чисел хранит величина $сум$? Ответ – надо к «старому» значению суммы $сум$ прибавить n :

```
сум := сум + n
```

Такая запись, невозможная в математике, в программировании возможна. С учетом этого фрагмент программы, связанный с этапом 2, имеет вид:

```
сум := 0
нц для ном от 1 до N
  ввод a
  сум := сум + a
кц
```

Внимание! В программировании Бейсик и Паскаль начальное присваивание переменной нулевого значения в данном случае не является обязательным. Вместе с тем, начальное присваивание переменной нулевого значения в программировании считается «правилом хорошего тона». В программах на школьном алгоритмическом языке, языке Python и ряде других такое присваивание обязательно.

2.2. Типовая задача 2 «Суммирование чисел последовательности, которые обладают некоторыми свойствами (удовлетворяют некоторому условию)»

Пусть требуется найти сумму чётных чисел в последовательности: 5, 3, 1, 5, 12, 6, 3, 3, 8, 12.

Отличие задач данного типа от рассмотренных в предыдущем разделе в том, что учитывать в значении переменной *сум* надо не все числа последовательности, а только чётные. Значит, должен быть использован неполный условный оператор (команда **если**):

```
ввод N
сум := 0
нц для ном от 1 до N
  ввод а
  если <число а - чётное>
    то
      сум := сум + а
  все
кц
```

Как определить, что целое число *a* является чётным, решите самостоятельно или читайте комментарий перед заданиями для самостоятельной работы.

В общем случае вид программы такой:

```
ввод N
сум := 0
нц для ном от 1 до N
  ввод а
  если <условие>
    то
      сум := сум + а
  все
кц
| Вывод результата
вывод нс, сум
```

Прежде чем идти дальше, заметим, что условие для учета очередного числа *a* может быть сложным – состоящим из двух или нескольких простых условий¹, соединённых служебными словами (логическими операторами) **and** (**и**), **or** (**или**) или **not** (**не**).

Результат проверки сложного условия определяется по известным читателю таблицам истинности. Для сложного условия с двумя простыми условиями *У1* и *У2* эти таблицы приведены ниже.

1) **и** (**and**)

У1	У2	У1 и У2
Ложь	Ложь	Ложь
Ложь	Истина	Ложь
Истина	Ложь	Ложь
Истина	Истина	Истина

¹ «Простым» условимся называть условие, к которому используется один из шести знаков сравнения («=», «>», «>=» и т. д.).

2) **или** (**or**)

Таблица 3.3

у1	у2	у1 или у2
Ложь	Ложь	Ложь
Ложь	Истина	Истина
Истина	Ложь	Истина
Истина	Истина	Истина

Результат выполнения логической операции отрицания (**not/не**) над условием *у* определяется так:

у	не у
Ложь	Истина
Истина	Ложь

2.3. Типовая задача 3 «Подсчет количества чисел последовательности, которые обладают некоторыми свойствами»

Пусть требуется найти количество чётных чисел последовательности: 5, 2, 7, 8, 3, 1, 5, 12, 6, 3, 3, 8, 12.

Используем в программе переменную *кол*, которая будет хранить количество чётных чисел среди уже введенных чисел.

Составим таблицу:

Порядковый номер числа	Значение	Количество чётных чисел
1	5	0
2	2	1
...		
...		<i>кол</i>
8	12	?

Как определить количество чётных чисел после ввода некоторого очередного чётного числа *a*, если информацию о таком количестве среди уже рассмотренных чисел хранит величина *кол_чет*? Ответ – надо к «старому» значению количества *кол* прибавить 1:

```
кол_чет := кол_чет + 1
```

Напомним, что в программировании такая запись также возможна. Это значит, что фрагмент программы для подсчета искомого количества такой:

```
кол_чет := 0
нц для ном от 1 до к
  ввод а
  если <число а - чётное>
  то
    кол_чет := кол_чет + 1
  все
кц
```

В общем случае соответствующая программа:

```
ввод N
кол := 0
нц для ном от 1 до N
  ввод а
  если <условие>
  то
    кол := кол + 1
  все
кц
вывод нс, кол
```

2.4. Типовая задача 4 «Определение среднего арифметического тех чисел последовательности, которые обладают некоторыми свойствами»

Для нахождения среднего арифметического чисел последовательности, которые удовлетворяют некоторому условию (например, положительных, чётных или т. п.), надо знать сумму и количество таких чисел. Решать соответствующие задачи мы уже умеем. Поэтому можем так оформить программу:

```
ввод N
сум := 0
кол := 0
нц для ном от 1 до N
  ввод а
  если <условие>
    то
      сум := сум + а
      кол := кол + 1
  все
кц
сред_арифм := сумма/кол
вывод нс, сред_арифм
```

Обращаем внимание на то, что многократно определять значение сред_арифм в «теле» условного оператора:

```
нц для ном от 1 до к
  ввод а
  если <условие>
    то
      сум := сум + а
      кол := кол + 1
      | Определяем результат
      сред_арифм := сум/кол
  все
кц
| Выводим окончательный результат
вывод нс, сред_арифм
```

необходимости нет. Это можно сделать один раз после окончания работы оператора цикла.

Обратим внимание на то, что приведенная программа будет работать без ошибок, если числа, удовлетворяющие заданному условию, в последовательности имеются (иначе будет иметь место деление на ноль). Именно такое уточнение, как правило, дается в демонстрационных вариантах экзамена.

2.5. Типовая задача 5 «Определение максимального значения в последовательности чисел»

Сразу же уточним, что в демонстрационных вариантах экзамена представлены задания, в которых речь идет о последовательности натуральных чисел.

Алгоритм решения задач такого типа аналогичен алгоритму действий человека, который определяет максимальное значение в некотором наборе:

4, 2, 9, 6, 3, 16, 10, 2, 7

– сначала он запоминает первое число, а затем рассматривает второе число. Если оно больше того числа, которое помнил, то запоминает новое число и переходит к следующему, третьему числу; в противном случае просто переходит к следующему числу и делает то же самое. Иными словами, каждое число последовательности, начиная со 2-го, сравнивается с максимальным значением среди чисел, рассмотренных до него.

Это значит, что в программе следует использовать переменную величину, которая будет хранить² максимальное число из уже рассмотренных чисел последовательности. Если принять имя этой переменной – макс, то сначала следует задать первое число и принять его в качестве максимального (макс). Затем задаются остальные числа *a* и каждое из них сравнивается со значением макс. Если $a > \text{макс}$, то в качестве нового значения макс принимается значение числа *a*.

Соответствующий фрагмент:

```
| Ввод первого числа a
ввод a
макс := a
нц для ном от 2 до N
  ввод a
  | Сравниваем
  если a > макс
    то
      макс := a
  все
кц
| Вывод результата или использование его в расчётах
...
```

Обсудим вопрос – можно ли не вводить отдельно первое число последовательности, а вводить и обрабатывать *все* его числа в теле оператора цикла:

```
нц для ном от 1 до N
  | Ввод очередного числа a последовательности
  ввод a
  если a > макс
    то
      макс := a
  все
кц
вывод нс, макс
```

На первый взгляд, да, можно. Однако это не так, точнее, так можно оформлять программу не на всех языках программирования и не во всех случаях.

В программе на школьном алгоритмическом языке и на языке Python при ее выполнении появится сообщение об ошибке, связанное с тем, что значение макс при первом использовании условного оператора не определено (в программе на языке Паскаль это значение будет равно нулю).

А если до оператора цикла присвоить величине макс нулевое значение?

```
макс := 0
```

В этом случае сообщения об ошибке не будет, но если, например, стоит задача определения максимальной температуры в первой декаде января и вводимые значения равны -6, -5, -4, -6, -3, -3, -2, -5, -5, -4, то в приведенном варианте программы результат окажется неправильным (он будет равен 0³).

Можно сделать вывод о том, что краткий вариант программы решения задачи о максимальной температуре в первой декаде января может быть применен, если известно, что в обрабатываемом наборе чисел не все значения температуры отрицательны:

² Напомним, что переменные величины используются в программе для хранения информации.

³ Убедитесь в этом.

```

макс := 0
нц для ном от 1 до 10
  вывод нс, "Введите температуру очередного дня "
  ввод темп
  если темп > макс
    то
      макс := темп
  все
кц
вывод нс, "Максимальная температура за декаду: ", макс, " градусов"

```

В общем случае краткий вариант программы может быть использован, если известно число X , которое заведомо не превышает минимальное число обрабатываемой последовательности:

```

макс := X
нц для ном от 1 до N
  ввод а
  если а > макс
    то
      макс := а
  все
кц
вывод нс, макс

```

В заданиях экзамена речь идет об определении максимального *натурального* числа. Это значит, что в качестве начального значения переменной *макс* можно принимать значение $0, -1, \dots$.

2.6. Типовая задача 6 «Определение порядкового номера максимального значения в последовательности чисел»

Здесь также алгоритм решения задачи аналогичен алгоритму действий человека, определяющего номер максимального значения в некотором наборе:

4, 9, 2, 6, 3, 16, 10, 2, 7

– сначала он запоминает первое число и номер 1, а затем рассматривает второе число. Если оно больше того числа, которое помнил, то запоминает новое число и номер 2 и переходит к следующему, третьему числу; в противном случае просто переходит к следующему числу и делает то же самое и т. д. Иными словами, каждое число последовательности, начиная со второго, сравнивается с максимальным числом среди рассмотренных до него.

В программе сначала следует задать первое число и принять его в качестве максимального (*макс*), а искомый порядковый номер (*номер_макс*) принять равным 1. Затем задаются остальные числа a и каждое из них сравнивается со значением *макс*. Если $a > \text{макс}$, то в качестве нового значения *макс* принимается значение числа a , а в качестве нового значения *номер_макс* – номер встреченного числа *ном*.

Соответствующая программа:

```

| Ввод первого числа а
ввод а
макс := а
номер_макс := 1
нц для ном от 2 до N
  | Ввод остальных чисел а последовательности
  ввод а
  | Сравниваем
  если а > макс

```



```

    то
      макс := а
      номер_макс := ном
  все
кц
вывод нс, номер_макс

```

Если заведомо известно число X , которое заведомо не превышает минимальное число обрабатываемой последовательности известно, то, как и в предыдущей задаче, фрагмент может быть оформлен короче:

```

макс := X
нц для ном от 1 до к
  | Ввод очередного числа а последовательности
  ввод а
  если а > макс
    то
      макс := а
      номер_макс := ном
  все
кц
вывод нс, номер_макс

```

При определении номера максимального *натурального* числа последовательности в качестве начального значения переменной *макс* можно принимать значение 0, -1, ..., а начальное значение переменной *номер_макс* – вообще не задавать.

2.7. Типовая задача 7 «Определение максимального значения с заданными свойствами в последовательности чисел»

Пример: определение максимального чётного числа, максимального отрицательного числа и т. п.

Отличие задач данного типа от рассмотренных типовой задачи 5 в том, что учитывать при поиске максимального числа надо не все числа последовательности, а только числа с заданными свойствами. Значит, как и при решении типовой задачи 2 должен быть использован неполный условный оператор (команда **если**).

Краткий вариант программы, который напомним, применим к обработке натуральных чисел, имеет вид:

```

макс := 0
нц для ном от 1 до N
  ввод а
  если <число а обладает заданными свойствами>
    то
      | Сравниваем
      если а > макс
        то
          макс := а
    все
  все
кц
вывод нс, макс

```

2.8. Типовая задача 8 «Определение порядкового номера максимального значения с заданными свойствами в последовательности чисел»

Соответствующий фрагмент программы решения задачи:

```
макс := 0
нц для ном от 1 до N
  ввод а
  если <число а обладает заданными свойствами>
    то
      | Сравниваем
      если а > макс
        то
          макс := а
          номер_макс := ном
        все
      все
кц
вывод нс, номер_макс
```

Вопрос: «Номер какого числа с заданными свойствами будет найден, если в последовательности будет несколько таких чисел?»

Прежде чем предлагать задания для самостоятельной работы, дадим ещё ряд комментариев.

В заданиях экзамена могут быть указаны максимальные значения переменных величин, например, «количество чисел не превышает 1000, введённые числа последовательности не превышают 30 000». Поэтому в программе на языке Паскаль и других языках программирования, в которых происходит объявление (описание) переменных, для соответствующих переменных должен быть указан соответствующий тип данных. Например, в программе на языке Паскаль для переменных с приведёнными значениями 1000 и 30000 это может тип *integer*.

Для проверки того факта, что некоторое целое число a является чётным следует определить остаток от его деления на 2. Если остаток равен нулю, то число a – чётное, в противном случае – нечётное. Знаки операции определения остатка показаны в таблице:

Язык программирования	Определение остатка	Пример выражения для определения остатка
Паскаль	mod	$a \bmod 2$
Python	%	$a \% 2$

В школьном алгоритмическом языке для определения остатка от деления одной величины целого типа на другую используется не знак операции, а функция `mod`:

```
mod(a, 1000)
```

Последнюю цифру любого натурального числа a можно определить, как остаток от деления этого числа на 10.

Ответы на следующие вопросы найдите самостоятельно:

- 1) как определить, что некоторое натуральное число a кратно числу b ?
- 2) как определить число, образованное двумя последними цифрами некоторого натурального числа a ?

В качестве примера выполним задание из демонстрационного варианта ОГЭ 2022 года.

Условие

Напишите программу, которая в последовательности натуральных чисел определяет количество чисел, кратных 4, но не кратных 7. Программа получает на вход количество чисел

в последовательности, а затем сами числа. В последовательности всегда имеется число, кратное 4 и не кратное 7. Количество чисел не превышает 1000. Введённые числа не превышают 30 000 Программа должна вывести одно число: количество чисел, кратных 4, но не кратных 7.

Пример работы программы:

Входные данные	Выходные данные
4	2
16	
28	
26	
24	

Программа на школьном алгоритмическом языке:

```

алг
нач цел N, а, кол, ном
  ввод N
  кол := 0
  нц для ном от 1 до N
    ввод а
    если mod(а, 4) = 0 и mod(а, 7) <> 0
      то
        кол := кол + 1
    все
  кц
  вывод кол
кон

```

3. Задания для самостоятельной работы

Во всех заданиях:

- количество чисел не превышает 1000;
- введённые натуральные числа не превышают 30 000, а при наличии в последовательности и отрицательных чисел все числа по модулю не превышают 30 000;
- в последовательности всегда имеется число с заданными свойствами.

1. Напишите программу, которая в последовательности натуральных чисел определяет количество чисел, оканчивающихся на 6. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – количество чисел, оканчивающихся на 6.

Пример работы программы:

Входные данные	Выходные данные
3	2
16	
26	
24	

2. Напишите программу, которая в последовательности натуральных чисел определяет сумму нечётных чисел. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – сумму нечётных чисел.

Пример работы программы:

Входные данные	Выходные данные
3 13 23 24	36

3. Напишите программу, которая в последовательности целых чисел определяет среднее арифметическое неотрицательных чисел. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – среднее арифметическое неотрицательных чисел.

Пример работы программы:

Входные данные	Выходные данные
4 -4 2 0 4	2.0

4. Напишите программу, которая в последовательности натуральных чисел определяет количество чисел, кратных 5, но не оканчивающих на 0. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – количество чисел, кратных 5, но не оканчивающих на 0.

Пример работы программы:

Входные данные	Выходные данные
5 20 26 25 5 100	2

5. Напишите программу, которая в последовательности натуральных чисел определяет сумму чисел, оканчивающихся цифрами 123. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – сумму чисел, оканчивающихся цифрами 123.

Пример работы программы:

Входные данные	Выходные данные
4 123 23 2123 23	2246

6. Напишите программу, которая в последовательности чисел определяет максимальное положительное число. Программа получает на вход количество чисел в последовательности, а затем сами числа. Программа должна вывести одно число – максимальное положительное число.

Пример работы программы:

Входные данные	Выходные данные
4	6
4	
-1	
6	
4	

7. Напишите программу, которая в последовательности натуральных чисел определяет номер максимального из чисел, больших 100 и оканчивающихся на 7. Программа получает на вход количество чисел в последовательности, а затем сами числа. Если в последовательности будет несколько таких чисел, должен быть выведен номер *первого* из них».

Пример работы программы:

Входные данные	Выходные данные
4	2
7	
247	
340	
247	

4. Частный случай задания

В заключение обсудим особенности выполнения варианта задания, в котором количество введённых чисел неизвестно, а известно, что последовательность чисел заканчивается числом 0 (0 – признак окончания ввода, не входит в последовательность).

Так как известно условие окончания цикла ввода и обработки чисел последовательности, то в программе можно использовать так называемый «оператор цикла с постусловием». Его общий вид такой:

```
нц
  <тело оператора цикла>
кц_при <условие окончания цикла>
```

причём в теле оператора цикла все действия должны выполняться только для введённых чисел, отличных от нуля:

```
нц
  ввод а
  если а <> 0
  то
    ...
  все
кц_при а = 0
```

Например, в этом случае определение суммы всех чисел последовательности в программе происходит так:

```
сум := 0
нц
  ввод а
  если а <> 0
  то
    сум := сум + а
  все
кц_при а = 0
```

В языке Паскаль общий вид оператора цикла с постусловием следующий:

```
repeat  
  <тело оператора цикла>  
until <условие окончания цикла>
```

а фрагмент программы определения суммы всех чисел последовательности:

```
sum := 0;  
repeat  
  readln(a);  
  if a <> 0 then  
    sum := sum + a  
until a = 0;
```

В языке Python оператор цикла с постусловием не предусмотрен, но его можно организовать с помощью оператора цикла с предусловием так:

```
while True:  
  a = int(input())  
  if a != 0:  
    sum = sum + a  
  else:  
    break
```

Цикл, который начинается с заголовка **while** True, будет выполняться бесконечно, потому что условие True (логическая константа) всегда истинно. Выйти из такого цикла можно только с помощью специальной инструкции **break** (досрочный выход из цикла). В данном случае она сработает тогда, когда станет истинным условие **a == 0**, то есть когда будет введено нулевое значение.

Можно также в программе можно применить так называемый «оператор цикла с предусловием». Его общий вид следующий:

```
нц пока <условие продолжения цикла>  
  <тело оператора цикла>  
кц
```

Условие продолжения цикла противоположно условию его окончания, то есть в данном случае общий вид оператора такой:

```
нц пока a <> 0  
  <тело оператора цикла>  
кц
```

Но при этом, чтобы оператор цикла начал «работать», нужно, чтобы условие в нем было истинным. Значит, в программе на предварительно величине *a* присвоить (условно) некоторое ненулевое значение:

```
a := 100  
нц пока a <> 0  
  ввод a  
  если a <> 0  
    то  
    ...  
  все  
кц
```

Например, в этом случае определение количества всех чётных чисел последовательности в программе происходит так:

```
кол := 0  
a := 100  
нц пока a <> 0  
  ввод a
```

```
если a <> 0
  то
    кол := кол + 1
  все
кц
```

Задания для самостоятельной работы

Напишите программы решения задач, описанных в заданиях 1–7 выше для случая, когда количество введённых чисел неизвестно, а известно, что последовательность чисел заканчивается числом 0 (0 – признак окончания ввода, не входит в последовательность).

Наше пожелание выпускникам:



Об использовании двоичной системы счисления в докомпьютерную эпоху

Златопольский Д.М.

«...эта система [двоичная] лучше всякой другой годилась бы для устройства арифметической машины...»

Эдуард Люка, французский математик, цитата из книги «Занимательная математика» [11], 1882 г.⁴

Во многих источниках, в том числе и учебных пособиях, утверждается, что впервые привычный современный вид двоичной системе придал великий немецкий ученый Готфрид Вильгельм Лейбниц (1646–1716) в 1703 г. в статье «Explication de l'Arithmétique Binaire» («Объяснение двоичной арифметики») [15]. Однако это, как можно прочитать в статьях [4, 5], не совсем так. Двоичная система была известна и до Лейбница: в качестве примера приводились и описывались работы Томаса Харриота и Хуана Карамюэля. При этом отмечалось, что в глазах учёных двоичная система не представляла практического значения (например, для производства вычислений), и виделась им лишь неким математическим курьёзом.

Первый пример практического применения двоичной системы счисления привел Лейбниц в упоминавшейся работе [15]. В ней он пишет, что двоичная запись числа может быть использована для определения минимального набора гирь для взвешивания грузов или для минимального набора монет (или купюр) для формирования некоторых сумм денег⁵. Таким набором являются гири массой 1, 2, 4, ... до степени двойки, не превышающей заданное число, а конкретный набор гирь для уравнивания на чашечных весах конкретного груза определяется по разрядам двоичной записи массы груза, в которых записана единица. Например, для уравнивания груза массой 13 кг следует использовать гири массой 8, 4 и 1 кг ($13_{10} = 1101_2$).

В 1623 г. английский философ, историк, политик Фрэнсис Бэкон (1561–1626) в трактате «De Dignitate et Augentis Scientiarum» («О достоинстве и приумножении наук») описал следующий метод шифрования текста [9]. Каждую букву алфавита в сообщении он предложил шифровать некоторой последовательностью букв двухлитерного алфавита, например, А и В. Тогда шифрующие последовательности для букв латинского алфавита будут иметь следующий вид:

Буква	Шифрующая последовательность
<i>a</i>	ААААА
<i>b</i>	ААААВ
...	...
<i>t</i>	ВААВА
...	...
<i>z</i>	ВАВВВ

Рис. 1

По сути, пользуясь современной терминологией, это был 5-битный двоичный код (если иметь в виду, что буква А соответствует нулю, а В – единице).

⁴ На русском языке цитата публикуется впервые.

⁵ Имелось в виду, что монеты (купюры) имеют достоинство 1, 2, 4,

Если код Бэкона был предназначен для шифрования текста, и неизвестно, пользовался ли им кто-либо, то так называемый «шрифт Брайля», разработанный в 1824 году французом Луи Брайлем, был первой системой записи с двоичным кодированием символов, которой пользовались (и пользуются) миллионы людей [1]. Он предназначен для письма и чтения незрячими и плохо видящим людям. В нём буквы на бумаге изображаются в виде проколов. Всего в предложенном Брайлем варианте могло быть до 6 проколов (точек), расположенных в два столбца. Каждой букве соответствовало определённое сочетание точек:

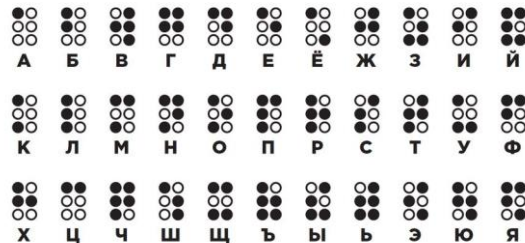


Рис. 2

Тёмные точки – это выпуклости на бумаге, светлые – плоские элементы кода символа.

Всего шрифтом Брайля можно было записать $2^6 = 64$ различных символа. При письме точки прокалываются, и поскольку читать можно только по выпуклым точкам, «писать» текст приходится с обратной стороны листа. Текст «пишется» справа налево, затем страница переворачивается, и текст читается слева направо.

Если пронумеровать места для точек так, как показано на рис. 3, и выпуклым точкам сопоставить цифру 1, а плоским элементам кода – 0, то можно сказать, что, например, для буквы Т двоичный код будет таким: 011110.



Рис. 3

По сути двоичным был так называемый «код Морзе», который предложил в 1838 г. американец Сэмюэл Морзе, хотя некоторые исследователи полагают, что её автором был Альфред Вейл – партнер Морзе по бизнесу, известный тем, что ввёл «коммерческий код» из групп по 5 символов (правда, и здесь автор/авторы системы не использовали цифры 0 и 1).

В современном варианте международного кода Морзе каждый символ кодируется на бумаге некоторой последовательностью из тире и точек, а при передаче с помощью технических средств (телеграфом, по радио и т. п.) – последовательностью длинных и коротких сигналов [7]. Если точку интерпретировать как 0, а тире – как единицу, то получим двоичный код. Отличие кода Морзе от двоичного кода с фиксированной разрядностью будет в том, что буквам с меньшей частотой использования в тексте того или иного языка, поставлена в соответствие более короткая последовательность точек и тире. Так, код Морзе для русской буквы Т состоит лишь из одного тире (в двоичном виде – 1), а для русской буквы Щ имеет вид – – · – (или 1101).

В связи с тем, что в коде Морзе символам соответствуют различные длины последовательностей точек и тире, учащимся может быть предложен следующий вопрос на смекалку: «Рассказывают, что, создавая свой код, Морзе отправился в ближайшую типографию, в которой в те времена, да и в течение длительного времени потом, текст в книгах, газетах и т. п. набирался вручную из отдельных литер. Литера – металлический, пластмассовый или деревянный брусок прямоугольного сечения с рельефным изображением буквы или знака на одной из его сторон. Зачем Морзе ходил в типографию?» [2].

(Целью визита в типографию было узнать, какие буквы наборщики текста используют чаще, а какие – реже. Такой подход уменьшает общее количество передаваемых сигналов, что ускоряет передачу информации по каналу связи).

Помимо кодирования информации, идеи использования двоичного представления чисел можно встретить в различных математических фокусах и головоломках. Так, известен фокус по отгадыванию числа (например, возраста человека или т. п.) по таблице (рис. 3 [3]).

I.	II.	III.	IV.	V.	VI.
1	2	4	8	16	32
3	3	5	9	17	33
5	6	6	10	18	34
7	7	7	11	19	35
9	10	12	12	20	36
11	11	13	13	21	37
13	14	14	14	22	38
15	15	15	15	23	39
17	18	20	24	24	40
19	19	21	25	25	41
21	22	22	26	26	42
23	23	23	27	27	43
25	26	28	28	28	44
27	27	29	29	29	45
29	30	30	30	30	46
31	31	31	31	31	47
33	34	36	40	48	48
35	35	37	41	49	49
37	38	38	42	50	50
39	39	39	43	51	51
41	42	44	44	52	52
43	43	45	45	53	53
45	46	46	46	54	54
47	47	47	47	55	55
49	50	52	56	56	56
51	51	53	57	57	57
53	54	54	58	58	58
55	55	55	59	59	59
57	58	60	60	60	60
59	59	61	61	61	61
61	62	62	62	62	62
63	63	63	63	63	63

Рис. 4

С помощью этой таблицы можно отгадать любое задуманное число от 1 до 63, если задумавший число скажет, в каких столбцах оно встречается (номера столбцов приведены в первой строке таблицы). Например, если задумано число 22, то по названным номерам столбцов (5, 3 и 2) его можно вычислить следующим образом: $2^{5-1} + 2^{3-1} + 2^{2-1} = 16 + 4 + 2$ (это можно сделать, даже не смотря на таблицу). Можно также запомнить, что с первым столбцом связано число 1, со вторым – 2, с третьим – 4, с четвертым – 8, с пятым – 16, с шестым – 32.

Секрет фокуса заключается в том, что в таблице в первом столбце справа выписаны те десятичные числа, которым в двоичной системе счисления соответствуют числа, оканчивающиеся на 1, во втором – числа, у которых в двоичном представлении вторая от конца цифра равна 1, в третьем – десятичные числа, которым в двоичной системе соответствуют числа, имеющие 1 в третьем справа разряде, и т. д. Так, уже приведенное число 22 в двоичной системе счисления имеет вид: 10110, и в таблице это число записано в пятом, третьем и втором столбцах. Если вспомнить правило перевода чисел из двоичной системы в десятичную, то по записи 10110 соответствующее десятичное число можно получить так: $2^4 + 2^2 + 2^1$ (это так называемая «развернутая запись числа»), или, «привязываясь» к нашей нумерации столбцов таблицы – $2^{5-1} + 2^{3-1} + 2^{2-1}$.

Описанный фокус упоминается в книге «История двоичной и других недесятичных систем счисления» [14], которую в 1971 году опубликовал Антон Глазер, профессор математики Пенсильванского университета, США. Этот фундаментальный труд,

выдержавший два издания, содержит подробный обзор всех известных работ по истории десятичных систем счисления с конца XVI века до момента появления компьютеров.

А.Глазер сначала пишет, что фокус на отгадывание числа в 1875 году описал немецкий математик Мориц Кантор (1829–1920), а потом уточняет: «Кантор был бы очень расстроен, если бы ему сообщили, что в издании 1814 года книги Жака Озанáма⁶ “Математические и физические развлечения” этот фокус уже был описан» [14, с. 93].

В 1872 году неизвестный автор опубликовал в Лионе, Франция, 16-страничную брошюру «Théorie du baguenaudier...» [16]. Она посвящена старинной головоломке, происходящей, предположительно, из Китая, состоящей из замкнутой проволочной «вилки», имеющей вид длинной шпильки для волос и воткнутой обоими свободными концами в рукоятку, и нескольких колец, связанных между собой довольно сложным образом (см. рис. 5). Задача состоит в том, чтобы снять всю систему колец с вилки или надеть её обратно. Французское название головоломки – *baguenaudier*. Русский вариант головоломки называется *меледа́* [6].

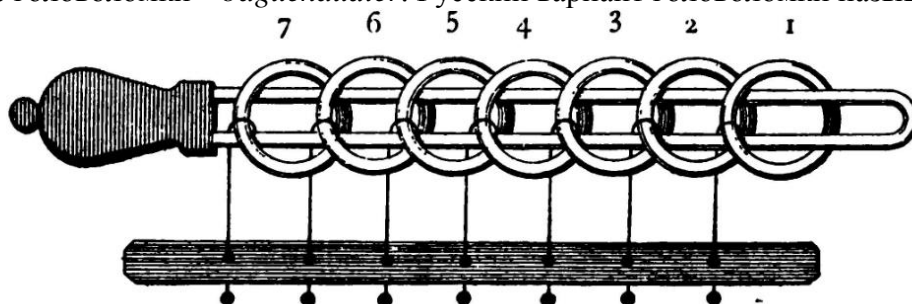


Рис. 5

Эдуард Люка в изданной в Париже в 1882 г. книге «Récréations mathématiques» («Математические развлечения») написал об анонимном авторе брошюры: «Нам недавно удалось узнать, что был Луи Гро, советник лионского апелляционного суда» [11, с. 169].

Луи Гро в своей работе, среди прочего, приводит таблицу, начальный фрагмент которой показан на рис. 6.

1 = 1	1
2 = 2	10
3 = 3	11
4 = 2 ²	100
5 = 5	101
6 = 2·3	110
7 = 7	111
8 = 2 ³	1000
9 = 3 ²	1001
10 = 2·5	1010

Рис. 6

Анализ показывает, что в таблице:

1) схемы с линией и точками выше и ниже её иллюстрируют текущую ситуацию на головоломке, когда стоит задача надеть на «вилку» некоторое количество колец. Точки

⁶ Французский математик, профессор Сорбонны (1640–1718). В 1694 году он опубликовал книгу «Récréations mathématiques et physiques» («Математические и физические развлечения»).

соответствуют кольцам; расположенные над линией – надеты, находящиеся ниже линии – сняты;

2) десятичные числа слева – порядковые номера манипуляций с кольцами («ходов»);

3) числа под схемами – двоичный эквивалент номера хода.

Такая таблица позволяет определить общее число ходов, требующихся для решения головоломки при том или ином количестве колец.

В 1899 году итальянский математик Джузеппе Пеано (1858 – 1932) опубликовал статью, предлагающую новую систему записи чисел, в основе которой лежала двоичная система счисления. Подробно он описал ее в сборнике «Основания математики» [12] и в брошюре «Применение двоичной нумерации в стенографии» [13].

Учёный предложил использовать символы "." и «!»⁷, обозначающие, соответственно, для 0 и 1 и привел ноль и несколько первых положительных целых чисел в виде:

. ! !.. !! !.. !!.. !!..

Для представления больших чисел Пеано ввел изображение, которые можно описать как «ромашку», в которой может быть до восьми лепестков. Каждый имеющийся лепесток соответствует единице, каждый отсутствующий – нулю. Таким образом, «ромашка» определяла восьмиразрядное двоичное число. Нумерация разрядов в числе показана на рис. 7, а несколько вариантов «цветка» и соответствующих десятичных значений – на рис. 8.

5	4	3
6*	*	2
7	8	1

Рис. 7

↘ = 1 - = 2 ' = 4 ♣ = 24 * = 255

Рис. 8

Примечание. Приведённые на рис. 8 изображения соответствуют двоичным числам 1, 10, 100, 11000 и 1111111.

При использовании двух изображений-«ромашек» можно было представлять и бóльшие числа (см. рис. 9).

↙♣ = 1900

Рис. 9

Свою систему Пеано предложил использовать в стенографической машине – разновидности пишущей машинки для стенографирования⁸. В брошюре [13] он упоминает использовавшуюся в то время в Сенате Италии 10-клавишную стенографическую машину профессора Антонио Микела Зукко и утверждает, что машина, созданная на основе его системы, проще и удобнее. При использовании двух 8-клавишных клавиатур можно одновременно записать две буквы или два слога, а общее число возможных вариантов букв, цифр и слогов составляет 65 536.



Для удобства запоминания «системы кодировки» (соответствия между буквами, цифрами и слогами и их числовыми кодами-«ромашками») Пеано разработал систему, учитывающую особенности итальянского языка, а также применил ряд приемов,

⁷ В [13] Пеано вместо символа «!» использует «:»:

↙ = :: = 4 + 1 = 5 ♣ = :: = 8 + 2 = 10,
 ↙♣ = :::: = 128 + 32 + 8 + 2 = 170.

(о графических изображениях — см. далее).

⁸ Пеано писал, что его система кодирования текста может быть применена и в телеграфной связи, «используя весь ее [связи] потенциал...».

облегчающих запоминание. Так, например, «глухой» согласной *f* соответствует код  (напоминающий букву *f*), при удалении из которого одного «лепестка» можно получить код «звонкой» согласной *v* (). Предложенные учёным коды 25 латинских букв (без буквы *w*) и 10 цифр приведены на рис. 10 и 11, соответственно, а код слога *pas* – на рис. 12 [12].

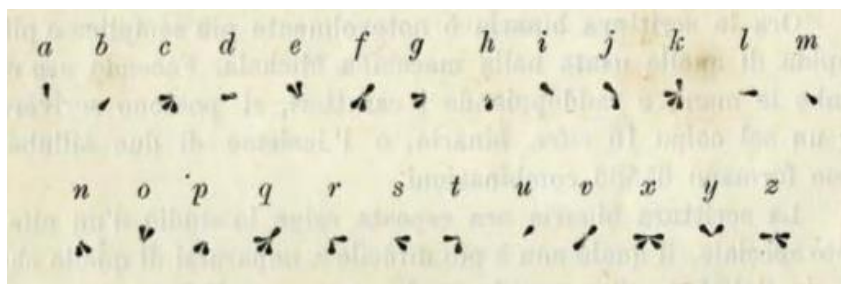


Рис. 10

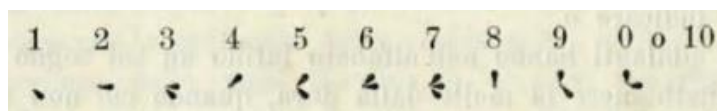


Рис. 11

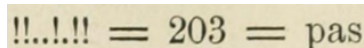


Рис. 12

В 1901 году Чарльз Бутон, профессор математики Гарвардского университета (США), опубликовал статью [10], связанную с игрой, которую он назвал «ним». Интересная информация о происхождении игры и её названии, в том числе убедительно опровергающая распространённые гипотезы и мифы, связанные с этим, приведена в книге [8].

Суть игры заключается в следующем. Имеется некоторое количество мелких предметов (камешков, монет, спичек и т. п.), разложенных в несколько кучек. Два игрока по очереди забирают по одному или несколько предметов из одной любой кучки (или все предметы из нее). Выигрывает тот, кто возьмёт последний предмет (или несколько последних предметов)⁹.

В статье Бутон провел полный анализ игры с тремя кучками предметов и представил доказательство существования оптимальной выигрышной стратегии.

Целью статьи, как пишет сам её автор, было доказательство того, что, если один из игроков *A* может каждым своим ходом оставлять некоторый набор значений предметов в кучках, другой игрок *B* не может выиграть. Такой набор значений Бутон назвал «безопасное сочетание» (*safe combination*), или «безопасная позиция». Он пишет: «Запишите количество предметов в каждой кучке в двоичной системе счисления¹⁰ и разместите эти числа одно под другим так, чтобы единицы были в одном столбце. Если при этом сумма в каждом столбце равна 2 или 0, то такой набор предметов образует безопасную позицию».

Например, Бутон приводит пример безопасной позиции для трех кучек из 9, 5 и 12 предметов:

$$\begin{array}{r} 1\ 0\ 0\ 1, \\ 1\ 0\ 1, \\ 1\ 1\ 0\ 0, \end{array}$$

Двоичная запись позволяет понять, что для двух любых различных чисел всегда однозначно определяется третье, которое образует безопасную комбинацию с двумя заданными числами, и наоборот – если числа *a*, *b*, *c* образуют безопасную комбинацию, любые

⁹ Можно играть и наоборот — считать того, кому достанется последний предмет, проигравшим.

¹⁰ И приводит при этом такой пример двоичной записи числа 9:

$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1001.$

два из них определяют оставшееся третье. Кроме того, видно, что игрок *B*, делая очередной ход в безопасной позиции, которую ему «оставил» соперник, никогда сам не сможет получить такую же – любой его ход делает позицию опасной (для себя). Выигрышная стратегия состоит в том, чтобы оставлять после каждого своего хода безопасную позицию.

В последующих доказательствах Бутон также везде использует двоичную запись чисел.

В заключение он приводит признак безопасной позиции для любого количества кучек (аналогичный признаку для трёх кучек – общее количество единиц в каждом столбце при записи двоичных значений чисел предметов в кучках одно под другим должно быть чётным), а также анализирует другой вариант игры – при котором проигравшим считается взявший последний предмет.

Через 52 года после слов Э.Люка (см. эпиграф к статье) немецкий студент Конрад Цузе (1910–1985) начал работу по созданию вычислительной машины, использующей двоичную систему счисления. В 1937 году такая механическая машина, названная Z1 (что означало «Цузе 1»), была готова и заработала!

А в конце 40-х годов XX столетия началась эра электронных вычислительных машин, в основе устройства и работы которых лежала двоичная система счисления...

Литература

1. Брайль Луи. https://ru.wikipedia.org/wiki/Брайль_Луи
2. *Златопольский Д.М.* 400 вопросов по информатике на логику и смекалку. М. ДМК Пресс, 2021. 226 с.
3. *Златопольский Д.М.* Занимательная информатика. М. Бином. Лаборатория знаний. 2010.
4. *Златопольский Д.М., Шилов В.В.* Из истории двоичной системы счисления. Томас Харриот // Информатика в школе. 2020. № 6. С. 19–23.
5. *Златопольский Д.М., Шилов В.В.* Из истории двоичной системы счисления. Хуан Карамюэль // Информатика в школе. 2020. № 8. С. 61–63.
6. Меледа. <https://ru.wikipedia.org/wiki/Меледа>
7. Морзе Сэмюэл. https://ru.wikipedia.org/wiki/Морзе_Сэмюэл
8. *Шилов В.В.* Удивительная история информатики и автоматки. М.: ЭНАС, 2011.
9. Шифр Бэкона. https://ru.wikipedia.org/wiki/Шифр_Бэкона
10. *Bouton C.L.* Nim, a game with a complete mathematical theory // The Annals of Mathematics, 2nd Ser., Vol. 3, No. 1/4. (1901–1902). P. 35–39.
11. *Édouard Lucas.* Récréations mathématiques. Paris. 1882. <https://archive.org/stream/rcrationsmathma00lucagoog#page/n26/mode/1up>
12. *Giuseppe Peano.* Formulaire de mathématiques. 3 vols. Paris. 1901. P. 77–78. <https://archive.org/details/formulairedemath00pean/page/211/mode/1up>
13. *Giuseppe Peano.* La numerazione binaria applicata alla stenografia. Torino. 1898. <https://archive.org/details/PeanoNumerazioneBinaria/page/n11/mode/2up>
14. *Glaser A.* History of binary and other nondecimal numeration. 1981.
15. *Leibnitz, Godefroy-Guillaume.* Explication de l'Arithmétique Binaire... // Mémoires de mathématique et de physique de l'Académie royale des sciences, Académie royale des sciences. 1703. P. 85–89.
16. Théorie du baguenaudier par un clerc de notaire lyonnais. <https://books.google.fr/books?id=EcoBJRekd-sC&pg=PP1#v=onepage&q&f=false>