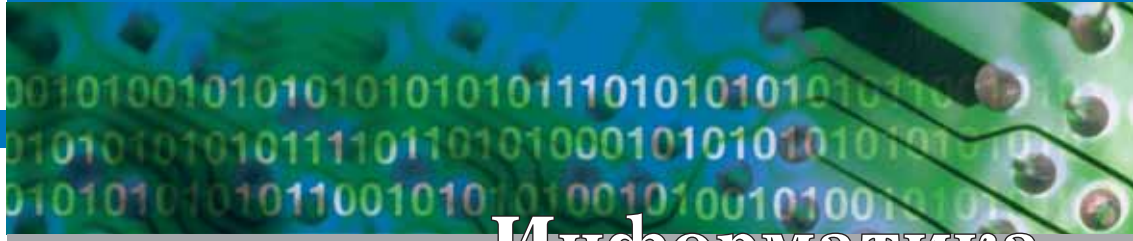




**Гончаренко Валерий Евстафиевич**
*Доцент кафедры «Информационные технологии в экономике и организация производства» (ИТЭ и ОП) ГОУ ВПО «Ивановский государственный университет», кандидат технических наук. Ответственный организатор городской олимпиады школьников по информатике, член экспертной комиссии ЕГЭ по информатике и ИКТ.*

# Определение простых чисел от века папируса до века ПК

Вот уже более 2000 лет неизменный интерес вызывают вопросы, связанные с простыми числами. В наши дни только в поисковой системе Яндекс за один месяц обрабатывается более 1 млн 300 тысяч запросов по определению простых чисел. Впору говорить о магии простых чисел. В предлагаемом материале многие могут найти ответы, начиная с истории вопроса до реализации алгоритмов и программ определения простых чисел на ПК.

## Из века в век на пике популярности

Помимо обычного человеческого любопытства к чему-либо необычному, простые числа играют важную роль в фундаментальном разделе математики – теории чисел. Прочные основы этой теории были заложены ещё в третьем веке до н. э. легендарным греческим математиком Евклидом и изложены в его книге «Начала» как итог 300-летнего развития античной математики. С именем другого великого греческого математика – Эратосфена (276–194 гг. до н. э.) – связано решение задачи определения всех простых чисел в ряду натуральных чисел от 2 до $N$. И по сей день идеи и результаты трудов великих греческих математиков востребованы. Основная теорема арифметики, определение НОД по алгоритму Евклида, решето Эратосфена и многое другое постоянно упоминается в учебных программах по математике, алгоритмизации и программированию.

Евклидом было выполнено доказательство бесконечности ряда простых чисел, а основная теорема арифметики делает простые числа незаменимыми в определении делимости чисел. Согласно теореме, каждое натуральное число $N>1$ можно представить в виде $N = p_1 \cdot \ldots \cdot p_k$, где $p_1,\ldots,p_k$ – простые числа, причём такое представление единственное с точностью до порядка следования сомножителей. Такое представление числа $N$ называется его каноническим разложением на простые сомножители, или факторизацией числа, например, $588\,000 = 2\cdot2\cdot2\cdot2\cdot2\cdot3\cdot5\cdot5\cdot7\cdot7 = 2^5\cdot3\cdot5^3\cdot7^2$.

Неоценимый вклад в определе-

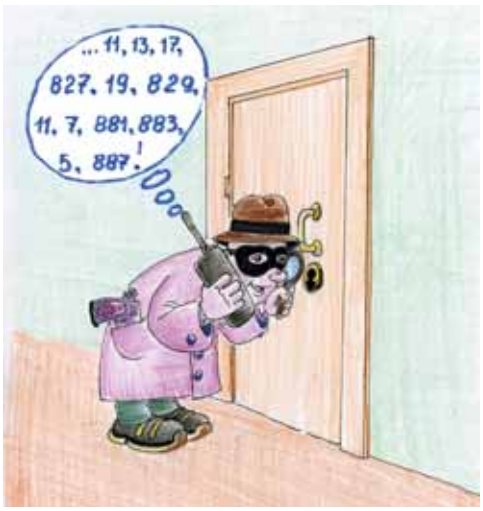
ние простых чисел внёс Эратосфен, предложив простой и очень эффективный алгоритм их определения от 2 до $N$, получивший название «решето Эратосфена». Математиков буквально пленила идея собрать самую большую коллекцию простых чисел, которые в виде таблиц издавались тысячестраничными томами. Титаническую работу проделал в XIX веке профессор чешского университета в Праге Якуб Филипп Кулик (1793–1863), составив самую большую на то время таблицу, которая содержала простые числа до 100 330 201. Читателям будет предложена программа, которая позволит создать за короткое время свою собственную коллекцию простых чисел, превышающую этот результат XIX века.

В настоящее время большие простые числа используются в криптографии при шифровке данных. Простые числа и их распределение играют важную роль в алгебре, физике, теории информации, используются в тестировании предельных возможностей вычислительных машин.

Учёные состязаются в своеобразном соревновании по определению самого большого простого числа. Полученные результаты расцениваются как достижения в теории чисел, заносятся в Книгу рекордов Гиннесса, а авторы получают солидные денежные премии.

Области исследований непосредственно простых чисел разнообразны, основными являются:

- определение принадлежности натурального числа $X$ к простым числам;
- определение простых чисел в диапазоне натуральных чисел от 2 до $N$.




Учёных в области теории чисел буквально завораживает задача, связанная с тайной распределения простых чисел. Они то собираются плотными стайками в ряду натуральных чисел, то располагаются в гордом одиночестве на большом расстоянии от своих собратьев. Премия в 1 млн долларов ждёт автора решения этой задачи.

## Немного теории

Натуральные числа являются подмножеством целых чисел. Натуральные (естественные) числа – числа, возникающие естественным образом при счёте, например при перечислении предметов: первый, второй, третий и т. д. Натуральные числа больше единицы подразделяются на простые и составные. Единица, будучи началом всех целых чисел, не является ни простым, ни составным числом (Л. Эйлер). Простое число имеет только два натуральных делителя – единицу и собственное значение, соответственно составные числа имеют более двух натуральных делителей. В соответствии с основной теоремой арифметики любое составное число можно представить в виде произведения простых чисел. Сле-

довательно, для определения делимости натурального числа $N$ достаточно использовать только простые числа.

До рассмотрения вопросов нахождения простых чисел уделим немного внимания составным числам – они в абсолютном большинстве скромно соседствуют с избалованными вниманием простыми числами.

Составное число можно представить в виде произведения первой, заранее известной пары делителей, например $12 = 1 \cdot 12$. Один из делителей – самый маленький из всех возможных, а другой – самый большой. Следующим наименьшим делителем после единицы может быть делитель 2, а в паре с ним наибольший после 12 будет 6, в итоге получим $12 = 2 \cdot 6$. Отсюда делаем вывод, что все возможные делители, кроме единицы и самого значения числа, находятся в диапазоне от 2 до целой его середины, т. е. от 2 до ($N$ div 2).

Ниже представлены фрагменты программ определения всех делителей составного числа $N$ на языке Pascal:

```
for i: = 2 to (N div 2) do
if (N mod i = 0) then
write (i, ' ');
```

или на языке C++:

```
for (i = 2; i < = N/2; i++)
if ! (N% i) cout <<i<<" ";.
```

В приведённых фрагментах программ можно существенно сократить количество итераций цикла, если проверять наличие парных делителей.

Теперь можно вернуться и к простым числам. Нет простой операции проверки неделимости числа. Для решения этой задачи лучше идти от противного. У составного числа, которое можно представить в виде произведения простых чисел, должен быть как минимум один делитель, значение которого не более корня квадратного из $N$. Приведём доказательство этого утверждения. Самыми большими будут сомножители, если у составного числа их будет только два, меньше уже быть не может:

$$N = p_1 \cdot p_2.$$

Хотя бы один из простых сомножителей должен принимать значение $p_i \le \sqrt{N}$, иначе $p_i^2$ будет больше $N$. Проверим это на простых примерах. Для $N = 9$ его каноническое разложение на простые сомножители, или факторизация числа будет $n = 3 \cdot 3$, получим $p_1 = \sqrt{9}$ и $p_2 = \sqrt{9}$, т. е. если происходит каноническое разложение на два одинаковых простых числа, то они в точности равны $\sqrt{N}$.

Для $N = 21$ его каноническое разложение будет $N = 3 \cdot 7$ и $p_1 = 3$ меньше чем $\sqrt{21}$ $\left(\sqrt{21} \approx 4{,}58\right)$. При каноническом разложении на два различных сомножителя один из них будет меньше, чем $\sqrt{N}$.

Таким образом, если мы будем испытывать число на делимость начиная с простого делителя – числа 2, то ряд потенциальных простых делителей можно заканчивать на значении $q \le \sqrt{N}$. Если в этом диапазоне нет ни одного делителя числа $N$, то оно является простым числом. В этом собственно отличается задача определения всех делителей натурального числа от проверки его на простоту.

Но оказывается, при определении простых чисел в диапазоне от 2 до $N$ можно вообще обойтись без операций проверки делимости. Если нам известна начальная последовательность простых чисел, например 2, 3, 5, 7, 11, 13, то из ряда натуральных чисел претендентов на звание простых чисел сразу можно вычеркнуть все числа, кратные представленному ряду простых чисел. Причём проверку числа на кратность можно даже и не производить, достаточно для делителя 2 обра-

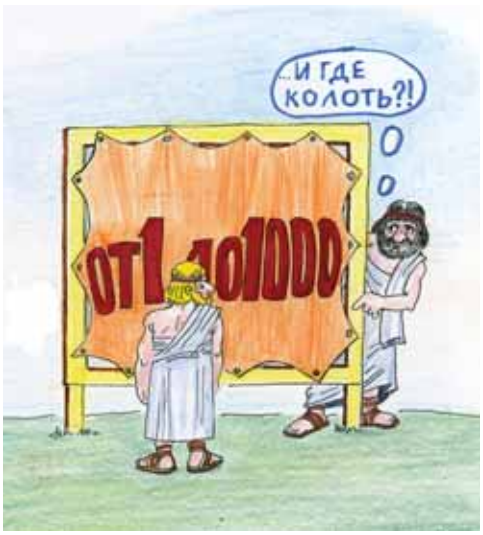
щаться к каждому второму числу на числовой оси и их вычёркивать, для 3 – к каждому третьему и т. д. Если мы выполним эти действия от 2 до верхней границы ряда натуральных чисел $N$, то невычеркнутыми останутся только простые числа в этом диапазоне значений.

Для наглядности рассмотрим последовательные этапы для ряда чисел от 2 до $N = 20$. После вычёркивания каждого второго числа после первого простого числа 2 получим: 2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~,11, ~~12~~, 13, ~~14~~, 15, ~~16~~, 17, ~~18~~, 19, ~~20~~.

Следующим после простого числа 2 невычеркнутым окажется простое число 3, и после него, вычёркивая каждое третье число, получим: 2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~.

На этом можно остановиться и отметить, что следующее невычеркнутое простое число будет 5, значение которого уже больше чем $\sqrt{N} = \sqrt{20} \approx 4,47$, или округлённо 4.

Мы, собственно, рассмотрели алгоритм, который предложил Эратосфен Киренский (276–194 гг. до н. э.) – греческий математик, астроном, географ и поэт. Эратосфен написал на папирусе, натянутом на рамке, все числа от 1 до 1000 и прокалывал составные числа. Папирус стал как решето, которое «просеивает» составные числа, а простые оставляет. Поэтому такой метод и получил название «решето Эратосфена».




## Программная реализация решета Эратосфена

В наши дни легче воспользоваться ПК, чем найти папирус или таблички, покрытые воском, поэтому для воспроизводства алгоритма решета Эратосфена ряд натуральных чисел можно представить в виде линейного массива, индексы которого будут в диапазоне до значения $N$. Именно индексы элементов массива используются в качестве значений натуральных чисел. А какие значения хранят сами элементы массива? Они хранят признак – «вычеркнули» или «не вычеркнули», принимая соответственно значения **false** или **true**.

Из источника [1] приведём фрагмент программы на языке Pascal для определения простых чисел в интервале от 2 до $n$.

```
q:=round(sqrt(n));
for i:=2 to q do
 if a[i] then
  begin
   j:=i*i;
   while j<=n do
    begin
     a[j]:=false;
     j:=j+i;
    end;
  end;
```

В приведённом фрагменте рассматривается линейный массив $a[i]$ логических значений типа **boolean**, индексы которого – в интервале от 2 до $n$. Изначально всем элементам массива должны быть присвоены значения **true**. Значение $q$ определяется как наибольшее значение в

ряду делителей $i$, где $i$ одновременно является и индексом массива $a[i]$. В переменную $j$ записывается очередной номер элемента, который будет кратен $i$, т. е. отстоять от очередного значения $a[j]$ через $i$ пунктов.

Как следует из приведённого фрагмента программы, непосредственные операции деления одного числа на другое не используются, и в итоге в диапазоне от 2 до $n$ номера элементов массива $a[i]$ со значением **true** будут простыми числами. Эти значения можно вывести на экран или записать в любую другую структуру натуральных чисел, например в линейный массив или в файл, используя цикл

```
for i: = 2  to n do
if a[i] then write (i, '  ');
```

В настоящее время имеется достаточно много дополнений и усовершенствований базового алгоритма решета Эратосфена, с ними можно познакомиться по ссылкам в [1].

Общим для этих алгоритмов остаётся то, что определяются простые числа на интервале от 2 до $N$.

Следовательно, если мы задаём новую границу ряда натуральных чисел $N$, алгоритм будет заново выполнять определение простых чисел, в том числе и ранее определённых.

Если с течением времени накапливать коллекцию простых чисел, то естественное решение: её надо хранить на устройствах постоянной памяти, т. е. в файле. Это решение очень важное во многих отношениях, в том числе и в необходимости модификации известных алгоритмов и программ. При хранении файла на жёстком диске современных ПК его ёмкость можно считать неограниченной для нашей задачи. Реальным ограничением будет ограничение на величину целого числа. Если для начала ограничиться стандартными возможностями, то для типа **longint** удовлетворимся формированием коллекции в пределах от 2 до 2 147 483 647 (в других системах программирования и в зависимости от разрядности процессора это ограничение будет гораздо выше).

Ниже представлен фрагмент программы, в которой для определения простых чисел в интервале от 2 до значения $N$ используется типизированный файл $f$ записей логического типа.

Если в файл $f$ записать $N$+1 значений **true**, то после реализации представленного фрагмента программы номера записей в файле, значение которых **true**, будут простыми числами.

```
bo:=true;
b:=false;
q:=round(sqrt(N));
for i:=0 to N do
     begin
          write(f,bo); { для всех чисел пока "истина"}
     end;
seek(f,0);
```

```
for i:=2 to q do
     begin
          seek(f,i);
          read(f,bo);
          if ( bo ) then
          begin
             j:=i*i;
             while j<=N do
                   begin
                       seek(f,j);
                       write(f,b);
                       j:=j+i;
                   end;{while}
          end;{then}
     end;{for i}
seek(f,0);
repeat
     read(f,bo);
     if bo then write(FilePos(f)-1,'  ');
until Eof(f);
```

Этот фрагмент мало чем отличается от ранее приведённого с использованием массива $a[i]$. Поскольку вместо линейного массива используется файл $f$, то для обращения к его записям используется процедура позиционирования внутреннего указателя **seek** ($f$, $i$), которая перемещает его к началу записи с номером $i$ (нумерация записей в файле начинается с нуля). Функция **FilePos**($f$) возвращает текущее значение позиции внутреннего указателя, при этом необходимо иметь в виду, что при операции чтения из файла или записи в него внутренний указатель автоматически смещается на одну позицию. Функция **Eof**($f$) возвращает истину, если указатель будет находиться перед меткой конца файла **(end of file)**. Пожалуй, это и все пояснения, которые помогут прочитать и понять работу приведённого фрагмента.

В конце фрагмента программы цикл **repeat** выводит на экран все простые числа из диапазона от 2 до $N$ в качестве номеров записей в файле, значения которых **true** (остались «невычеркнутые»).

После реализации программы на своём ПК может возникнуть вопрос: «Какое конкретное значение верхней границы натуральных чисел $N$ можно задать?». Использование в программе файлов обеспечивает надёжную сохранность полученных результатов, но существенно замедляет её работу. Как решить эту проблему при создании большой коллекции простых чисел, будет рассмотрено в последующей публикации.

## Литература

1. http://ru.wikipedia.org/wiki/Решето_Эратосфена.