Занятие 18. Алгоритм Евклида

В занятии 5 мы уже касались этого вопроса. Теперь рассмотрим алгоритм Евклида для нахождения наибольшего общего делителя двух натуральных чисел более подробно.

Из математики известно, что всякое целое число a представляется единственным способом через натуральное число b в форме

$$a = bq + r;\ 0 \le r < b \qquad (1)$$

В этом случае q называют неполным частным от деления a на b, а r остатком. Если $r = 0$, то говорят, что a кратно b или что b является делителем числа a.

Обратите внимание, что неполное частное q и остаток r в формуле (1) для $a < 0$ при ненулевом остатке отличаются от того, что дают операторы Паскаля $q := a\ div\ b$ и $r := a\ mod\ b$. В формуле (1) частное $q = \left\lfloor \frac{a}{b} \right\rfloor$ есть наибольшее целое, не превосходящее $\frac{a}{b}$. При этом остаток r будет неотрицательным. Например, для $a = -7$, $b = 2$ получим

$$q = \left\lfloor \frac{-7}{2} \right\rfloor = -4 \le -3{,}5$$
$$r = 1$$

В Паскале же получится $-7\ div\ 2 = -3$, $-7\ mod\ 2 = -1$. Знак остатка там всегда совпадает со знаком делимого.

Найдём наибольший общий делитель положительных целых чисел a и b. Будем обозначать его $\gcd(a,b)$[1].

Теорема 1

Если для целых чисел a, b, q, r выполняется равенство

$$a = bq + r,$$

то совокупность общих делителей чисел a и b совпадает с совокупностью общих делителей чисел b и r.

Доказательство.

Действительно, пусть k — общий делитель чисел a и b. Это значит, что существуют такие целые q_1 и q_2, для которых $a = kq_1$ и $b = kq_2$. Тогда $kq_1 = kq_2q + r$ или $r = k(q_1 - q_2q)$. Последнее равенство доказывает, что k является делителем числа r. Аналогично доказывается, что всякий общий делитель чисел b и r является также и делителем числа a. Таким образом, множество общих делителей чисел a и b совпадает с множеством общих делителей чисел b и r. В частности, совпадают их наибольшие общие делители: $\gcd(a,b)=\gcd(b,r)$.

Теперь для неотрицательных целых чисел можно записать ряд равенств

$$\begin{aligned} a &= bq_1 + r_1, && 0 < r_1 < b \\ b &= r_1q_2 + r_2, && 0 < r_2 < r_1 \\ r_1 &= r_2q_3 + r_3, && 0 < r_3 < r_2 \end{aligned} \qquad (2)$$

и т.д. Совокупность остатков получается убывающей $r_1 > r_2 > r_3 > \ldots$ Но все эти остатки положительны и меньше b. А последовательность убывающих натуральных чисел, большее из

[1] От английского greatest common divisor. В русской литературе применяют также обозначение НОД(a,b)

которых меньше b, не может состоять больше, чем из b чисел. Значит, на каком-то шаге мы получим нулевой остаток:

$$r_{n-1} = r_n q_{n+1} + 0$$

Таким образом, совокупность общих делителей чисел a и b совпадает с совокупностью общих делителей чисел b и r_1, которая совпадает с совокупностью общих делителей чисел r_1 и r_2 и т.д., пока не дойдём до пары r_n и 0. Наибольший общий делитель последней пары есть r_n.

Мы доказали, что наибольший общий делитель пары натуральных чисел a и b равен последнему отличному от нуля остатку в приведённом выше ряду (2).

Рекурсивный алгоритм

Опираясь на теорему 1, можно построить следующую рекурсивную подпрограмму.

```
Function gcd(a,b : Int64) : Int64;
Begin
  If b>0 Then gcd:=gcd(b,a mod b)
  Else gcd:=a
End;
```

Заметим, что условие выхода из рекурсии будет выполнено обязательно, так как появление нулевого остатка неизбежно.

Итеративный алгоритм

Этот же алгоритм можно представить в виде цикла. При этом будем опираться на ряд (2).

```
Function gcd(a,b : Int64) : Int64;
```

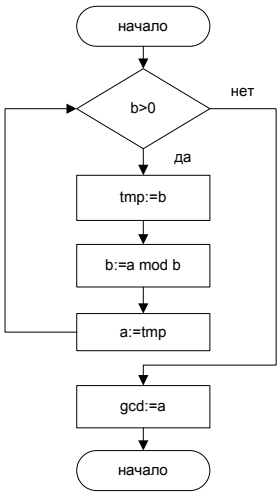

Рисунок 1. Алгоритм Евклида

Эта реализация алгоритма Евклида содержит проверку всего одного условия в отличие от алгоритма, рассмотренного в занятии 5. А это значит, что алгоритм, представленный на рис. 1, работает быстрее. Ведь на вычисление значения каждого логического выражения и на передачу управления нужной ветви в условном операторе тратится некоторое время.

Заметим также, что обе реализации алгоритма Евклида, рассмотренные выше: и рекурсивная, и итеративная — правильно работают не только для натуральных чисел, а и в том случае, если одно из чисел натуральное, а другое равно нулю. Если хотя бы одно из чисел отрицательно, для нахождения наибольшего общего делителя можно использовать следующее свойство:

$$\gcd(a, b) = \gcd(|a|, |b|)$$

Вспомните, что модуль числа *x* на языке Паскаль вычисляет функция abs(*x*).

Расширенный алгоритм Евклида

Попробуем выразить наибольший общий делитель *натуральных* чисел a и b в виде линейной комбинации этих чисел. То есть мы хотим найти такие числа x и y, что

$$x \cdot a + y \cdot b = \gcd(a, b) \qquad (3)$$

Если $b = 0$, то такое разложение получить легко. Достаточно взять $x = 1$, а значением y можно выбрать любое число, например ноль:

$$1 \cdot a + 0 \cdot 0 = \gcd(a, 0) \qquad (4)$$

Пусть теперь $b \neq 0$.
Как мы уже знаем (см. теорему 1), в этом случае $\gcd(a, b) = \gcd(b, a \bmod b)$.

Примем во внимание, что $a \bmod b = a - q \cdot b$, где $q = \left\lfloor \frac{a}{b} \right\rfloor$ — целая часть от деления a на b.

Теперь для $\gcd(b, a \bmod b)$ ищем разложение, аналогичное (3).

$$x_1 \cdot b + y_1 \cdot (a - q \cdot b) = \gcd(b, a \bmod b) = \gcd(a, b)$$

Преобразуя последнее равенство, получаем

$$y_1 \cdot a + (x_1 - q \cdot y_1) \cdot b = \gcd(a, b) \qquad (5)$$

Равенства (3) и (5) должны выполняться при любых значениях a и b. Это заведомо будет иметь место, если положить

$$\begin{aligned} x &= y_1 \\ y &= x_1 - q \cdot y_1 \end{aligned} \qquad (6)$$

Равенства (4) и (6) позволяют организовать рекурсивное вычисление коэффициентов разложения (3). Более того, мы видим, что коэффициенты x и y можно сделать целыми числами. Ведь при их вычислении используются только умножение, вычитание и *целочисленное* деление (для вычисления q), а начальные значения x и y, как следует из (4), можно выбрать целочисленными. При этом числа x и y не обязательно будут натуральными.

В вычислении наибольшего общего делителя двух натуральных чисел и его представления в виде линейной комбинации (3) исходных чисел описанным выше способом состоит суть расширенного алгоритма Евклида.

Ниже приведён пример программы, реализующей расширенный алгоритм Евклида. В ней ищется наибольший общий делитель двух натуральных чисел и его представление в виде линейной комбинации исходных чисел.

```
{$mode delphi}
Program gcde;

//Расширенный алгоритм Евклида
Procedure gcd_ext(a, b : int64; var x, y, gcd : int64);
var x1,y1 : int64;
Begin
  If b=0 Then
  Begin
    gcd:=a;
    x:=1;
    y:=0;
    Exit
  End;
  gcd_ext(b, a mod b, x1, y1, gcd);
  x := y1;
```

```
  y := x1 - (a div b) * y1
End;


Var m,n,x,y,g : int64;
Begin
  WriteLn('Введите 2 натуральных числа');
  Read(m,n);
  If (m<=0) or (n<=0) Then
  Begin
    WriteLn('Числа должны быть натуральными');
    Halt
  End;
  gcd_ext(m,n,x,y,g);
  WriteLn('gcd(',m,',',n,')=',g);
  Write(x,'*',m);
  If y>=0 Then Write('+');
  Write(y);
  WriteLn('*',n,'=',g);
End.
```

Задача

На седьмой интернет-олимпиаде для школьников сезона 2009–2010 года 22 мая 2010 года Санкт-Петербургский государственный университет ИТМО предлагал следующую задачу.

Обратные числа

Имя входного файла:	inverse.in
Имя выходного файла:	inverse.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Позавчера Рома узнал про новую для него операцию в математике — *взятие по модулю*, которая обозначается $a \bmod m$, например $5 \bmod 3 = 2$. Напомним, что $a \bmod m = b$, если $a = m \cdot k + b$ и $0 \le b < m$ (все перечисленные числа целые).

Вчера Рома начал перемножать числа, и заметил, что бывают такие случаи, что $(a \cdot b) \bmod m = 1$ (причем не обязательно оба числа равны единице). И это его очень заинтересовало, он даже придумал название этому феномену — числа a и b взаимно обратны по модулю m.

Сегодня утром Рома начал рассматривать простые числа m. Он доказал, что в таком случае для любого числа $a : 1 \le a < m$ существует ровно одно b, такое что $(a \cdot b) \bmod m = 1$. Однако он не знает, как по числам a и m найти число b. Помогите ему!

Формат входного файла

В единственной строчке входного файла заданы числа a и m ($1 \le a < m \le 2 \cdot 10^9$). Число m — простое.

Формат выходного файла

В выходной файл выведите число b, такое, что $(a \cdot b) \bmod m = 1$. Оно также должно удовлетворять неравенству $1 \le b < m$.

Примеры

inverse.in	inverse.out
3 7	5
6 23	4

Решение

Число m — простое. Это значит, что у него всего два делителя: 1 и m. Число $a < m$, следовательно, все его делители тоже меньше m. Отсюда делаем вывод, что единственный, он же

наибольший, общий делитель чисел a и m равен единице[2]. Пользуясь расширенным алгоритмом Евклида, можно найти такие целые p и k, что

$$p \cdot a + k \cdot m = 1 \qquad (7)$$

При этом выполняется соотношение

$$(p \cdot a)\ mod\ m = 1 \qquad (8)$$

Однако не гарантируется истинность неравенства $1 \le p < m$, хотя ясно, что $p \ne m$, иначе равенства (7) и (8) не могли бы выполняться.

Действительно, при $p = m$ из (7) следует $m \cdot a + k \cdot m = 1$, или $m \cdot (a + k) = 1$, что невозможно ввиду ограничения $m > a \ge 1$.

Равенство (8) выполняется для любого $p = b + n \cdot m$, где b — требуемый ответ задачи, а n — некоторое целое число, в том числе и отрицательное. Тогда, вычислив $f = p\ mod\ m$, мы получим $-m < f < m$. (Здесь операция mod понимается в смысле языка Паскаль.) Число $f + m$, для которого также истинно (8), будет удовлетворять неравенству $0 < (f + m) < 2m$. Отсюда видно, что ответом задачи будет $(f + m)\ mod\ m$, т.е.

$$b = \big((p\ mod\ m) + m\big)\ mod\ m \qquad (9)$$

Таким образом, решение задачи сводится к следующей последовательности шагов:

1. пользуясь расширенным алгоритмом Евклида, вычислить p, удовлетворяющее формуле (7);
2. вычислить b по формуле (9).

Задания для самостоятельного выполнения

1. Дано n целых чисел. Найдите их наибольший общий делитель. Сведите эту задачу к задаче нахождения наибольшего общего делителя двух чисел. Используйте алгоритм функции gcd, представленный на рис. 1. Обязательно ли при вызове функции gcd требовать, чтобы a было больше, чем b?
2. Напишите программу решения задачи «Обратные числа». Тесты для её проверки возьмите на сайте авторов задачи http://neerc.ifmo.ru/school/io/archive/20100522/archive-advanced-20100522.rar
3. В процедуре gcd_ext при $b = 0$ мы полагали $x = 1$, а $y = 0$. А можно ли было переменной y не присваивать вообще никакого значения? Ведь $1 \cdot a + 0 \cdot b = a$ для любого b. Проверьте своё предположение, подправив текст процедуры gcd_ext и выполнив пробные запуски программы при различных входных данных, в том числе достаточно больших.
4. Напишите итеративную (не рекурсивную) реализацию процедуры gcd_ext.
5. Усовершенствуйте расширенный алгоритм Евклида так, чтобы он работал с целыми, а не только натуральными, числами. Напишите программу. Учтите, что в языке Паскаль операция *mod* для отрицательных чисел может давать отрицательный остаток, что не соответствует равенствам (2).
6. Малая теорема Ферма утверждает, что для любого простого p и целого a, которое не делится на p, справедливо соотношение $a^{p-1}\ mod\ p = 1$. Используя эту теорему, попробуйте найти решение задачи «Обратные числа», не использующее алгоритм Евклида. (В теореме Ферма операцию mod будем понимать в смысле вычисления неотрицательного остатка, как в формуле (1). Хотя для данной задачи это не столь важно: все числа в ней неотрицательные.) Вам также может помочь тот факт, что остаток от произведения чисел равен остатку от произведения остатков: $(a \cdot b)\ mod\ n = \big((a\ mod\ n) \cdot (b\ mod\ n)\big)\ mod\ n$

[2] Такие числа называют взаимно простыми

Литература

1. Виноградов И.М. Основы теории чисел. Издание шестое. — Москва, Ленинград: Государственное издательство технико-теоретической литературы, 1952.
2. Д. Кнут. Искусство программирования для ЭВМ. Том 1. Основные алгоритмы. Издание 3. — Издательский дом «Вильямс», 2005.
3. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. Второе издание. — Москва, Санкт-Петербург, Киев. Издательство «Вильямс», 2010.
4. Деление с остатком. // http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BA%D0%BE%D0%BC
5. http://neerc.ifmo.ru/school/io/archive/20100522/problems-basic-20100522.pdf
6. http://neerc.ifmo.ru/school/io/archive/20100522/archive-advanced-20100522.rar
7. Michaël Van Canneyt. Reference guide for Free Pascal, version 2.4.2. Пункт 9.8.1.