Медведев Михаил Геннадиевич
Кандидат физико-математических наук, доцент факультета кибернетики Киевского национального университета имени Тараса Шевченко.

Наибольший общий делитель

Каждый из нас в школе изучал, что такое наибольший общий делитель (далее НОД) двух чисел a и b. Конечно же, это наибольшее целое число d, на которое a и b делятся без остатка. Без труда каждый ученик может сказать, например, что НОД(12, 18) = 6. Но что, если одно из чисел равно 0? А если a или b отрицательно? Над этим вопросом на школьных уроках, наверное, не каждый из нас задумывался. Для того чтобы ответить на поставленные вопросы, приведём определение – что же такое наибольший общий делитель.

Определение 1. *Наибольшим общим делителем* (далее НОД) двух целых чисел a и b, одновременно не равных нулю, называется такое наибольшее целое число d, на которое a и b делятся без остатка. Этот факт обозначается так: $d = \text{НОД}(a, b)$. Если оба числа равны нулю, то положим $\text{НОД}(0, 0) = 0$.

Исходя из определения, имеют место следующие равенства:

$$\text{НОД}(a, b) = \text{НОД}(b, a),$$
$$\text{НОД}(a, b) = \text{НОД}(-a, b),$$
$$\text{НОД}(a, 0) = |a|.$$

Почему, скажете вы, НОД(-12, 18) равен 6, а не –6? Ведь и –12, и 18 делятся нацело на 6 и на –6. Ответ прост: ведь НОД – это же *наибольший* общий делитель, а число 6 больше –6.

С понятием наибольшего общего делителя тесно связано понятие наименьшего общего кратного.

Определение 2. *Наименьшим общим кратным* (далее НОК) двух целых чисел a и b называется наименьшее положительное целое число, кратное как a, так и b.

Основная теорема арифметики утверждает, что любое натуральное число n можно представить в виде произведения простых чисел:

$$n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}.$$

Такое разложение натурального числа называется *каноническим*. Из него следует, что если

$a = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$, $b = p_1^{b_1} p_2^{b_2} \ldots p_k^{b_k}$, то

$$\text{НОД}(a, b) = p_1^{\min(a_1, b_1)} p_2^{\min(a_2, b_2)} \ldots p_k^{\min(a_k, b_k)},$$

$$\text{НОК}(a, b) = p_1^{\max(a_1, b_1)} p_2^{\max(a_2, b_2)} \ldots p_k^{\max(a_k, b_k)}.$$

Пример 1. Рассмотрим числа $a = 24$ и $b = 18$. Разложим их на простые множители: $24 = 2^3 \cdot 3$, $18 = 2 \cdot 3^2$. Следовательно,

$$\text{НОД}(24, 18) = 2^{\min(3, 1)} \cdot 3^{\min(2, 1)} =$$

$$= 2^1 \cdot 3^1 = 6,$$

$$\text{НОК}(24, 18) = 2^{\max(3, 1)} \cdot 3^{\max(1, 2)} =$$

$$= 2^3 \cdot 3^2 = 8 \cdot 9 = 72.$$

Именно такой метод, с использованием канонического разложения чисел, мы изучали в школе для нахождения НОД и НОК. Однако этот метод не эффективен для реализации алгоритмов их вычисления.

Рассмотрим следующий очевидный факт. Если НОД(a, b) = d, то a и b делятся на d. Следовательно, их разность $a - b$ также делится на d. Имеет место следующее рекуррентное соотношение для вычисления НОД:

$$\text{НОД}(a, b) = \begin{cases} a, & b = 0, \\ b, & a = 0, \\ \text{НОД}(a - b, b), & a \geq b, \\ \text{НОД}(a, b - a), & a < b. \end{cases}$$

Пример 2. Пусть $a = 32$, $b = 12$. Тогда

$$\text{НОД}(32, 12) = \text{НОД}(32 - 12, 12) =$$

$$= \text{НОД}(20, 12) = \text{НОД}(20 - 12, 12) =$$

$$= \text{НОД}(8, 12) = \text{НОД}(8, 12 - 8) =$$

$$= \text{НОД}(8, 4) = \text{НОД}(8 - 4, 4) =$$

$$= \text{НОД}(4, 4) = \text{НОД}(4 - 4, 4) =$$

$$= \text{НОД}(0, 4) = 4.$$

Приведённый метод вычисления не является оптимальным. Например, для нахождения НОД(1000000, 2) следует выполнить 500000 операций вычитания. Для ускорения вычисления НОД операцию вычитания заменим операцией взятия остатка от деления:

$$\text{НОД}(a, b) = \begin{cases} a, & b = 0, \\ b, & a = 0, \\ \text{НОД}(a \bmod b, b), & a \geq b, \\ \text{НОД}(a, b \bmod a), & a < b. \end{cases}$$

Пример 3. Пусть $a = 78$, $b = 14$. Тогда

$$\text{НОД}(78, 14) = \text{НОД}(78 \bmod 14, 14) =$$

$$= \text{НОД}(8, 14) = \text{НОД}(8, 14 \bmod 8) =$$

$$= \text{НОД}(8, 6) = \text{НОД}(8 \bmod 6, 6) =$$

$$= \text{НОД}(2, 6) = \text{НОД}(2, 6 \bmod 2) =$$

$$= \text{НОД}(2, 0) = 2.$$

Упростим приведённую выше рекуррентность, сократив количество условий до двух:

$$\text{НОД}(a, b) = \begin{cases} a, & b = 0, \\ \text{НОД}(b, a \bmod b), & b \neq 0. \end{cases}$$

Если $a < b$, то

$$\text{НОД}(a, b) = \text{НОД}(b, a \bmod b) = \text{НОД}(b, a),$$

то есть аргументы функции переставляются. При последующих вызовах функции НОД первый аргумент всегда больше второго. Нулём может стать только второй аргумент b.

Пример 4. Пусть $a = 14$, $b = 78$. Тогда

$$\text{НОД}(14, 78) = \text{НОД}(78, 14) =$$

$$= \text{НОД}(14, 8) = \text{НОД}(8, 6) =$$

$$= \text{НОД}(6, 2) = \text{НОД}(2, 0) = 2.$$

Реализуем на языке программирования Си функцию gcd (*Greatest Common Divisor)* – вычисления НОД, используя последнюю рекуррентность. Знак % в Си обозначает операцию взятия остатка от деления.

```
int gcd(int a, int b)
{
  if (b == 0) return a;
  return gcd(b, a % b);
}
```

Напомним, что *условный оператор* в Си имеет следующий синтаксис:

```
if (<условное выражение>) <выражение 1>; else <выражение 2>;
```

Если <условное выражение> истинно, то выполняется <выражение 1>, иначе выполняется <выражение 2>.

Тернарный условный оператор имеет синтаксис:

```
<условное выражение> ? <выражение 1>:<выражение 2>;
```

и семантически немного отличается от оператора **if..then..else**. Если <условное выражение> истинно, то оператор возвращает значение, которое возвращает <выражение 1>, иначе возвращается значение выражения <выражение 2>.

Используя тернарный оператор, функцию **gcd** можно записать следующим образом:

```
int gcd(int  a, int b)
{
  return (!b) ? a : gcd(b, a %  b);
}
```

Теорема. Между НОД и НОК двух чисел имеет место соотношение:

$a \cdot b = \text{НОД}(a, b) \cdot \text{НОК}(a, b)$.

Функция lcm (*Lowest Common Multiple*) вычисления НОК имеет вид:

```
int lcm(int a, int b)
{
return a / gcd(a, b) * b;
}
```

Заметим, что при вычислении выражения $a \cdot b / \gcd(a, b)$ может возникнуть переполнение, а при $a / \gcd(a, b) \cdot b$ нет. Здесь подразумевается, что значения a, b и $\text{lcm}(a, b)$ лежат в границах типа **int**.

Задача. При делении числа n на d получается частное q и остаток r. При этом q – максимально возможное целое, для которого $qd \le n$, а $r = n - qd$.

Для любого множества целых чисел

$$\{a_1, \ldots, a_k\}$$

всегда существует такое целое d, что числа $a_i \bmod d$ равны.

Вход. Каждая строка является отдельным тестом и содержит последовательность целых чисел $a_1, \ldots, a_k$, заканчивающуюся нулём. Последний ноль не принадлежит самой последовательности. Последова-

тельность содержит не менее 3 и не более 1000 чисел. Не все числа в последовательности равны между собой. Признаком конца входных данных является строка с одним нулём.

Выход. Для каждой входной последовательности $a_1, \ldots, a_k$ вывести максимальное d, для которого при делении a_i на d будут получаться равные остатки.

Пример входа	Пример выхода
701 1059 1417 2312 0	179
14 23 17 32 122 0	3
14 -22 17 -31 -124 0	3
0	

Решение. Из условия задачи следует, что

$$a_1 = d \cdot r_1 + rest,$$
$$a_2 = d \cdot r_2 + rest,$$
$$\ldots$$
$$a_k = d \cdot r_k + rest.$$

Поскольку d максимально, то

$$\text{НОД}(r_1, r_2, \ldots, r_n) = 1.$$

Далее имеем:

$$a_2 - a_1 = d \cdot (r_2 \quad r_1),$$
$$a_3 - a_2 = d \cdot (r_3 \quad r_2),$$
$$\ldots$$
$$a_k - a_{k-1} = d \cdot (r_k - r_{k-1}).$$

Откуда d – наибольшее целое, которое делит

$$a_2 - a_1,\ a_3 - a_2,\ \ldots,\ a_k - a_{k-1}.$$

То есть

$$d = \text{НОД}(|a_2 - a_1|, |a_3 - a_2|, \ldots, |a_k - a_{k-1}|).$$

Реализация. Основной цикл программы состоит в чтении последовательности чисел $\{a_1, \ldots, a_k\}$ и последовательном вычислении значения

$$\text{НОД}(|a_2 - a_1|, |a_3 - a_2|, \ldots, |a_k - a_{k-1}|).$$

```
#include <stdio.h>
#include <math.h>

int a, b, res;

int gcd(int a, int b) { ... } // код функции gcd приведён выше

void main(void)
{
  while (scanf("%d %d", &a, &b), a)
  {
    res = abs(a - b); a = b;
    while(scanf("%d", &b), b)
    {
      res = gcd(res, abs(a - b));
      a = b;
    }
    printf("%d\n", res);
  }
}
```