Целочисленные алгоритмы (язык Паскаль)

Вычисление НОД

НОД = наибольший общий делитель двух натуральных чисел – это наибольшее число, на которое оба исходных числа делятся без остатка.

Перебор:

```
k := a; { или k := b; }
while (a mod k <> 0) or
      (b mod k <> 0) do
  k := k - 1;
writeln ('НОД(', a, ',', b, ')=', k);
```

или

много операций для больших чисел

Алгоритм Евклида

```
НОД(a,b)=НОД(a-b,b)
        =НОД(a,b-a)
```

Евклид
(365-300 до. н. э.)

Заменяем большее из двух чисел разностью большего и меньшего до тех пор, пока они не станут равны. Это и есть НОД.

НОД вычисляется через НОД. Как это называется?

Пример:

```
НОД (14, 21) = НОД (14, 21-14) = НОД (14,
7)           = НОД (7, 7) = 7
```

много шагов при большой разнице чисел:

```
НОД (1998, 2) = НОД (1996, 2) = … =
2
```

Модифицированный алгоритм Евклида

```
НОД(a,b)=НОД(a mod b,b)
        =НОД(a, b mod a)
```

Заменяем большее из двух чисел остатком от деления большего на меньшее до тех пор, пока меньшее не станет равно нулю. Тогда большее — это НОД.

Пример:

```
НОД (14, 21) = НОД (14, 7) = НОД (0, 7) =
7
```

Еще один вариант:

```
НОД(2·a,2·b)= 2·НОД(a,b)
НОД(2·a,b)= НОД(a,b)  // при нечетном b
```

Реализация алгоритма Евклида

Рекурсивный вариант:

```
function NOD (a, b: integer): integer;
begin
 if a = b then NOD := a
 else
   if a < b then
        NOD := NOD(a, b-a)
   else NOD := NOD(a-b, b);
end;
```

```
function NOD (a, b: integer): integer;
begin
 if a*b = 0 then NOD := a+b
 else
   if a < b then
        NOD := NOD(a, b mod a)
   else NOD := NOD(a mod b, b);
end;
```

Почему a+b?

Реализация алгоритма Евклида

Без рекурсии:

```
function NOD (a, b: integer): integer;
begin
 while a <> b  do
   if a > b then a := a - b
   else          b := b - a;
 NOD := a;
end;
```

```
function NOD (a, b: integer): integer;
begin
 while a*b <> 0 do
   if a > b then a := a mod b
   else          b := b mod a;
 NOD := a + b;
end;
```

Поиск простых чисел

Простые числа – это числа, которые делятся только на себя и на 1.

Применение:

1) криптография;
2) генераторы псевдослучайных чисел.

Наибольшее известное (сентябрь 2008):

$2^{43112609} - 1$ (содержит 12 978 189 цифр).

Задача. Найти все простые числа в интервале от 1 до заданного N.

Простое решение:

```
for i:=1 to N do begin
  isPrime := True;
  for k:=2 to i-1 do
    if i mod k = 0 then
      isPrime := False;
  if isPrime then
    writeln(i);
end;
```

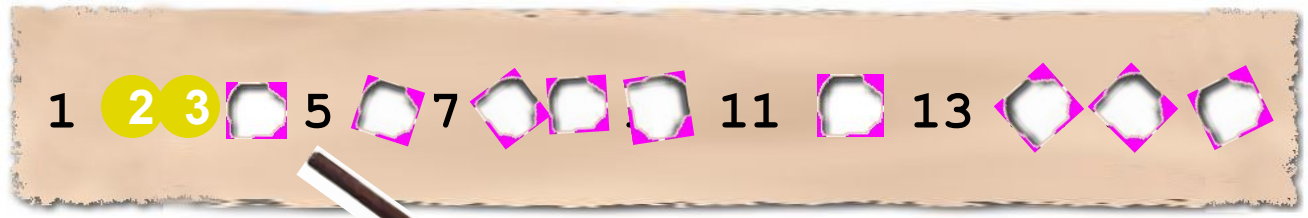
Как уменьшить число шагов внутреннего цикла?

$k <= \sqrt{i}$ ⟺ `k*k <= i`

Как оценить число операций?

`O(N²)` растет не быстрее N^2

Решето Эратосфена



Эратосфен Киренский
(Eratosthenes, Ερατοσθδνη)
(ок. 275-194 до н.э.)

Алгоритм:

1) начать с **k = 2**;
2) «выколоть» все числа через **k**, начиная с **2·k**;
3) перейти к следующему «невыколотому» **k**;
4) если **k·k <= N**, то перейти к шагу 2;
5) напечатать все числа, оставшиеся «невыколотыми».

Новая версия – решето Аткина .

высокая скорость, количество операций

$$O((N \cdot \log N) \cdot \log \log N)$$

нужно хранить в памяти все числа от **1** до **N**

Реализация

Логический массив **A[N]**, где

A[i] = True, если число **i** не «выколото»,

A[i] = False, если число **i** «выколото».

```
{ сначала все числа не выколоты }
for i:=1 to N do A[i] := True;

{ основной цикл
  «выкалывание» составных чисел }

{ выводим оставшиеся числа }
for i:=1 to N do
  if A[i] then writeln(i);
```

Реализация

Основной цикл:

```
k := 2;
while k*k <= N do begin
  if A[k] then begin
    i := k*k;
    while i <= N do begin
      A[i] := False;
      i := i + k;
    end;
  end;
  k := k + 1;
end;
```

«выкалываем» все числа, кратные **k**

Задания

«4»: Реализовать «решето Эратосфена», число *N* вводить с клавиатуры.

«5»: То же самое, но сравнить число шагов алгоритма для различных значений *N*. Построить график в *Excel*, сравнить сложность с линейной. Заполнить таблицу:

N	1000	5000	10000	20000	50000
Количество простых чисел					
Число шагов внутреннего цикла					

Что такое длинные числа?

Задача. Вычислить (точно)

$$100! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 99 \cdot 100$$

Проблема:

это число содержит более 100 цифр…

Сколько нулей в конце этого числа?

Какая последняя ненулевая цифра?

Решение:

хранить цифры в виде массива, по группам (например, 6 цифр в ячейке).

Сколько ячеек нужно?

$$100! < \underbrace{100^{100}}_{\text{201 цифра}}$$

$201/6 \approx 34$ ячейки

Хранение длинных чисел

$$1234\ 568901\ 734567 =$$
$$= 1234 \cdot 1000000^2 + 568901 \cdot 1000000^1 + 734567 \cdot 1000000^0$$

На что это похоже?

Хранить число по группам из 6 цифр – это значит представить его в системе счисления с основанием **d** = 1000000.

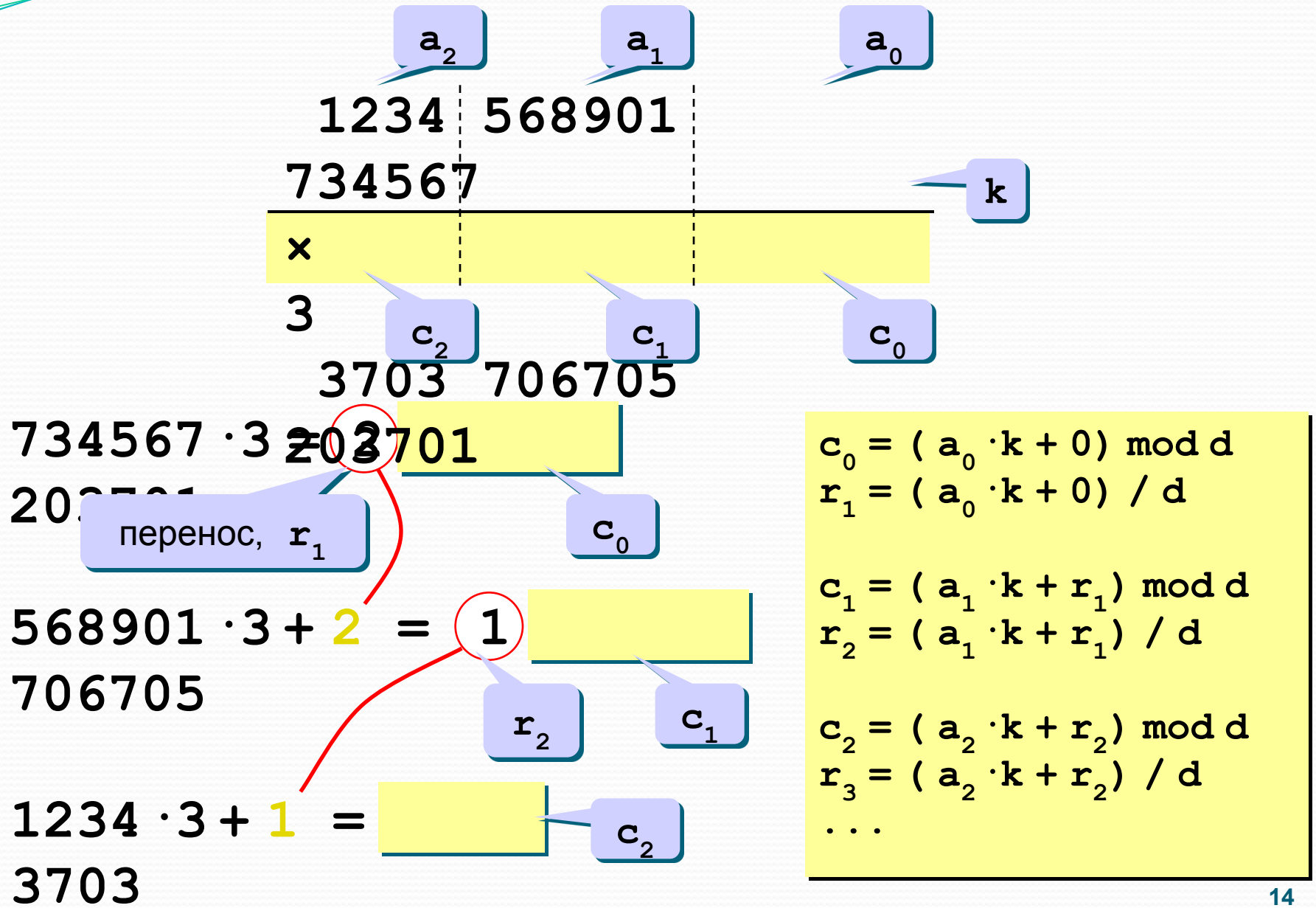
Алгоритм:

```
[A] = 1;
for k:=2 to 100 do
   [A] = [A]*k;
{ вывести [A] }
```

[A] – длинное число, хранящееся как массив

умножение длинного числа на «короткое»

Умножение длинного числа на короткое



$c_0 = (a_0 \cdot k + 0) \bmod d$
$r_1 = (a_0 \cdot k + 0) / d$

$c_1 = (a_1 \cdot k + r_1) \bmod d$
$r_2 = (a_1 \cdot k + r_1) / d$

$c_2 = (a_2 \cdot k + r_2) \bmod d$
$r_3 = (a_2 \cdot k + r_2) / d$
...

Вычисление 100!

```
const d = 1000000; { основание системы }
var A: array[0..40] of integer;
    s, r,              { произведение, остаток }
    i, k,              { вспомогательные }
    len: integer;      { длина числа }
begin
  { присвоить [A] = 1  }
  { последовательно умножать
    [A] на 2, 3, ..., 100 }
  { вывести [A] }
end.
```

Вычисление 100!

```
len := 1;                { записать [A]=1 }
A[0] := 1;
for i:=1 to 40 do
  A[i] := 0;

for k:=2 to 100 do begin
  i := 0; { с младшего разряда}
  r := 0; { пока нет переноса}
  while (i < len) or (r > 0) do begin
    s := A[i]*k + r;
    A[i] := s mod d; { в этом разряде }
    r := s div d;      { перенос }
    i := i + 1; { к
  end;
  len := i;        { новая длина числа }
end;
```

пока не кончились цифры числа [A] или есть перенос

? Где результат?

? Можно ли брать другое d?

Как вывести длинное число?

«Первая мысль»:

```
for i:=len-1 downto 0 do
  write(A[i]);
```

Что плохо?

Проблема:

как не потерять первые нули при выводе чисел, длина которых менее 6 знаков?

123 ⇨ 000123

Решение:

составить свою процедуру, а при выводе старшего разряда (**len-1**) убирать лидирующие нули:

```
write(A[len-1]); { старший разряд }
for i:=len-2 downto 0 do
  Write6(A[i]);
```

Как вывести длинное число?

Процедура:

```
procedure Write6(N: integer);
var x, d: integer;
begin
  x := 100000;
  while x > 0 do begin
    d := N div x;
    N := N mod x;
    x := x div 10;
    write(d);
  end;
end;
```

N	x	d

Задания

«4»: Составить программу для вычисления

$$99!! = 1 \cdot 3 \cdot \ldots \cdot 97 \cdot 99$$

«5»: То же самое, но написать свою процедуру для вывода, использующую символьные строки.

«6»: Написать программу для умножения двух длинных чисел (ввод из файла).

«7»: Написать программу для извлечения квадратного корня из длинного числа (ввод из файла).