

# Программирование 1

## Лекция 1 (часть 2)

### Вводный пример

# РАЗРАБОТКА и ФОРМЫ ЗАПИСИ АЛГОРИТМА и ПРОГРАММЫ

# РАЗРАБОТКА И ФОРМЫ ЗАПИСИ АЛГОРИТМА

*Пример* основных этапов работы над алгоритмом

Наибольший общий делитель (НОД) двух  
натуральных чисел

Greatest Common Divisor (GCD)

**Дано** : два натуральных числа  $a$  и  $b$  ( $a, b > 0$ ).

**Требуется** : найти натуральное число  $c = \text{НОД}(a, b)$ .

Школьный способ: вычислять НОД на основе разложения чисел  $a$  и  $b$  на *простые множители*

$$a = 2^{a_2} 3^{a_3} 5^{a_5} 7^{a_7} \dots = \prod_{q \text{ простое}} q^{a_q},$$

$$b = 2^{b_2} 3^{b_3} 5^{b_5} 7^{b_7} \dots = \prod_{q \text{ простое}} q^{b_q},$$

где целые  $a_q$  и  $b_q \geq 0$ .

$$\text{НОД}(a, b) = \text{gcd}(a, b) = \prod_{q \text{ простое}} q^{\min(a_q, b_q)}.$$

## Пример

$$a = 754, \quad b = 143$$

$$a = 2 \times 13 \times 29, \quad b = 11 \times 13$$

$$\text{НОД}(a, b) = 2^0 \times 11^0 \times 13^1 \times 29^0 = 13$$

**!** Получение разложения произвольного числа на простые множители само по себе является *непростой* задачей

## Другой способ вычисления НОД

Сначала рассмотрим  
формальное (точное) определение  $\text{НОД}(a, b)$ .

Запись  $p \mid q$  для натуральных  $p$  и  $q$  далее означает, что  $q$  является делителем (делит нацело)  $p$ .

Например,  $754 \mid 13$                        $(754 : 13 = 58)$

**Определение.** Натуральное число  $c = \text{НОД}(a, b)$ , если

1)  $c$  – делитель  $a$ , т. е.  $a \mid c$ ;

2)  $c$  – делитель  $b$ , т. е.  $b \mid c$ ;

}  $c = \text{НОД}(a, b)$

3)  $c$  – наибольшее из натуральных чисел, удовлетворяющих 1) и 2).

4) для натуральных  $p$  и  $q$  запись  $p \mid q$  означает, что существует такое натуральное  $s$ , что  $p = s q$ ;

5) наибольшим из множества  $M$  натуральных чисел является такое  $p \in M$ , что не существует другого числа  $q \in M$ , такого, что  $q > p$ .

## Способ вычисления НОД на основе определения

Последовательно перебираем числа  $c = 1, 2, 3, \dots, \min(a, b)$  и находим *максимальное* среди тех из них, для которых справедливо  $a | c$  и  $b | c$ .

Улучшенный способ: числа перебираются в порядке убывания от  $\min(a, b)$  до 1, тогда первое встретившееся  $c$ , такое, что  $a | c$  и  $b | c$ , и будет НОД( $a, b$ ).

! Оказывается, существует более эффективный (по количеству операций) алгоритм.

**Алгоритм Евклида**

Полезно строить вычисления не  
непосредственно на определении  
вычисляемой величины,  
а на её свойствах.

Свойства (очевидные):

1.  $\gcd(a, b) = \gcd(b, a)$

2.  $\gcd(a, a) = a$

Можно расширить область значений входных чисел  
 $a$  и  $b$ , допуская, что одно из них может быть равно  $0$ .

Тогда

3.  $\gcd(a, 0) = a.$

Для формулировки важного свойства НОД, напомним определения операций

деления нацело **div** и нахождения остатка от деления **mod**.

Пусть  $a, b \in N$  и  $a > b > 0$ , тогда существуют, и притом единственные  $q \in N$  и  $r \in N_0$ , такие, что имеет место представление

$$a = qb + r, \quad 0 \leq r < b.$$

Например,  $25 = 3 \cdot 7 + 4$ .

Обычно используют обозначения

$$q = a \text{ div } b, \quad r = a \text{ mod } b,$$

и тогда

$$a = (a \text{ div } b) b + (a \text{ mod } b).$$

## Свойство НОД

Пусть  $a, b \in \mathbb{N}$  и  $a > b > 0$ , тогда  $\gcd(a, b) = \gcd(b, r)$ , где  $r = a \bmod b$ ,

В других обозначениях  
 $\gcd(a, b) = \gcd(b, a \bmod b)$ ,  
 $\gcd(a, b) = \gcd(b, a - q b)$ .

Доказательство см. в учеб. пособии

Пример:  $\gcd(754, 143) = \gcd(143, 39)$ ,  
 $754 = 5 * 143 + 39$

Можно сформулировать и доказать аналогичное свойство НОД, включающее операцию вычитания:

$$(a > b > 0) \rightarrow \gcd(a, b) = \gcd(a - b, b).$$

# Разработка алгоритма

В основу алгоритма положим два свойства НОД:

1.  $(a > b > 0) \rightarrow \gcd(a, b) = \gcd(b, a \bmod b)$ ;
2.  $\gcd(a, 0) = a$ .

Общая идея алгоритма:

последовательно от пары чисел  $(a, b)$  переходить к новой паре чисел  $(b, a \bmod b)$ .

При этом  $\max(b, a \bmod b) < \max(a, b)$ ,

т. е. каждый такой шаг «уменьшает» текущую пару.

Шаги продолжаютя, пока не будет получена пара  $(a, 0)$ , и тогда  $\gcd(a, 0) = a$ .

Пример 1:  $a = 754$ ,  $b = 143$

Номер шага	Текущая пара чисел	Нахождение остатка	Следующая пара чисел
1	(754, 143)	$754 = 5 \cdot 143 + 39$	(143, 39)
2	(143, 39)	$143 = 3 \cdot 39 + 26$	(39, 26)
3	(39, 26)	$39 = 1 \cdot 26 + 13$	(26, 13)
4	(26, 13)	$26 = 2 \cdot 13 + 0$	(13, 0) !

Ответ  $\gcd(754, 143) = 13$ .

Пример 2:  $a = 754$ ,  $b = 144$

Номер шага	Текущая пара чисел	Нахождение остатка	Следующая пара чисел
1	(754, 144)	$754 = 5 \cdot 144 + 34$	(144, 34)
2	(144, 34)	$144 = 4 \cdot 34 + 8$	(34, 8)
3	(34, 8)	$34 = 4 \cdot 8 + 2$	(8, 2)
4	(8, 2)	$8 = 4 \cdot 2 + 0$	(2, 0)!

Ответ  $\gcd(754, 144) = 2$ .

Пример 3:  $a = 610$ ,  $b = 144$

Номер шага	Текущая пара чисел	Нахождение остатка	Следующая пара чисел
1	(610, 144)	$610 = 4*144 + 34$	(144, 34)
2	(144, 34)	$144 = 4*34 + 8$	(34, 8)
3	(34, 8)	$34 = 4*8 + 2$	(8, 2)
4	(8, 2)	$8 = 4*2 + 0$	(2, 0)!

Ответ  $\gcd(610, 144) = 2$ .

Пример 4:  $a = 233$ ,  $b = 144$

Номер шага	Текущая пара чисел	Нахождение остатка	Следующая пара чисел
1	(233, 144)	$233 = 1 * 144 + 89$	(144, 89)
2	(144, 89)	$144 = 1 * 89 + 55$	(89, 55)
3	(89, 55)	$89 = 1 * 55 + 34$	(55, 34)
4	(55, 34)	$55 = 1 * 34 + 21$	(34, 21)

См. продолжение на следующем слайде



Номер шага	Текущая пара чисел	Нахождение остатка	Следующая пара чисел
1	(233, 144)	$233=1*144+89$	(144, 89)
2	(144, 89)	$144=1*89+55$	(89, 55)
3	(89, 55)	$89=1*55+34$	(55, 34)
4	(55, 34)	$55=1*34+21$	(34, 21)
5	(34, 21)	$34=1*21+13$	(21, 13)
6	(21, 13)	$21=1*13+8$	(13, 8)
7	(13, 8)	$13=1*8+5$	(8, 5)
8	(8, 5)	$8=1*5+3$	(5, 3)
9	(5, 3)	$5=1*3+2$	(3, 2)
10	(3, 2)	$3=1*2+1$	(2, 1)
11	(2, 1)	$2=2*1+0$	(1, 0)!

**Ответ**  $\gcd(233, 144) = 1.$

# Замечание о вычислительном процессе и алгоритме (программе)

Каждый пример содержит *последовательность шагов*.

*Шаг* определяется текущим состоянием (парой чисел) и вызывает определенное действие (нахождение остатка и замену предыдущей пары на новую).

В каждом примере набор конкретных состояний (в том числе начальное) и действий, вообще говоря, разные.

Все примеры - это один **вычислительный процесс**, но разные его *реализации* (проявления), определяемые начальным состоянием - входными данными).

# О вычислительном процессе и алгоритме

(продолжение)

*Реальные осуществления вычислительного процесса (ВП) - его реализации.*

Сам ВП - это совокупность всех своих реализаций - уже абстракция.

Что объединяет все реализации ВП?

Ответ: алгоритм (или программа), как **описание** ВП.

Программа = набор правил (инструкций), который направляет эволюцию ВП.

Иногда мы не будем различать ВП и его реализацию (из контекста будет ясно о чём речь), но всегда различаем ВП и алгоритм (программу).

# Цитата

Структура и  
Интерпретация  
Компьютерных  
Программ

Второе издание



Харольд Абельсон  
Джеральд Джей Сассман  
при участии Джули Сассман

**Вычислительные процессы** – это абстрактные существа, которые живут в компьютерах. Развиваясь, процессы манипулируют абстракциями другого типа, которые называются **данными**. Эволюция процесса направляется набором правил, называемым **программой**. В сущности, мы заколдовываем дух компьютера с помощью своих чар.

Абельсон Х., Сассман Д.Д., Сассман Д.

**Структура и интерпретация компьютерных программ –**

М.: Добросвет, КДУ, 2006

# Конец замечания об алгоритмах вычислительных процессах

Вернемся к алгоритму Евклида

# Алгоритм Евклида («Математическая запись»)

Пусть  $c_0 = a, c_1 = b$  ( $a > b > 0$ ). Тогда  $\gcd(a, b) = \gcd(c_0, c_1)$ .

№	До шага	Действия шага	После шага
1:	$\{c_0, c_1\}$ <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid black; padding: 2px;">Делимое</div> <div style="border: 1px solid black; padding: 2px;">Делитель</div> <div style="border: 1px solid black; padding: 2px;">Остаток</div> </div>	$c_0 = q_1 c_1 + c_2$	$\{c_1, c_2\}$ $\{\gcd(c_1, c_2) = \gcd(c_0, c_1)\}$
2:	$\{c_1, c_2\}$	$c_1 = q_2 c_2 + c_3$	$\{c_2, c_3\}$ $\{\gcd(c_2, c_3) = \gcd(c_1, c_2)\}$
		...	
i:	$\{c_{i-1}, c_i\}$	$c_{i-1} = q_i c_i + c_{i+1}$	$\{c_i, c_{i+1}\}$ $\{\gcd(c_i, c_{i+1}) = \gcd(c_{i-1}, c_i)\}$
		...	
n:	$\{c_{n-1}, c_n\}$	$c_{n-1} = q_n c_n + c_{n+1}$	$\{c_n, c_{n+1}\}$ $\{\gcd(c_n, c_{n+1}) = \gcd(c_{n-1}, c_n)\}$

Предполагается, что  $n$ -й шаг вычислений последний, т. е.  $c_{n+1} = 0$  и  $\gcd(c_n, 0) = c_n$ , а следовательно,  $c_n = \gcd(a, b)$ .

1. **Обоснование правильности алгоритма** (отложим)
2. Обоснование завершимости алгоритма:

$$c_0 > c_1 > c_2 > c_3 > \dots > c_{n-1} > c_n > c_{n+1} = 0$$

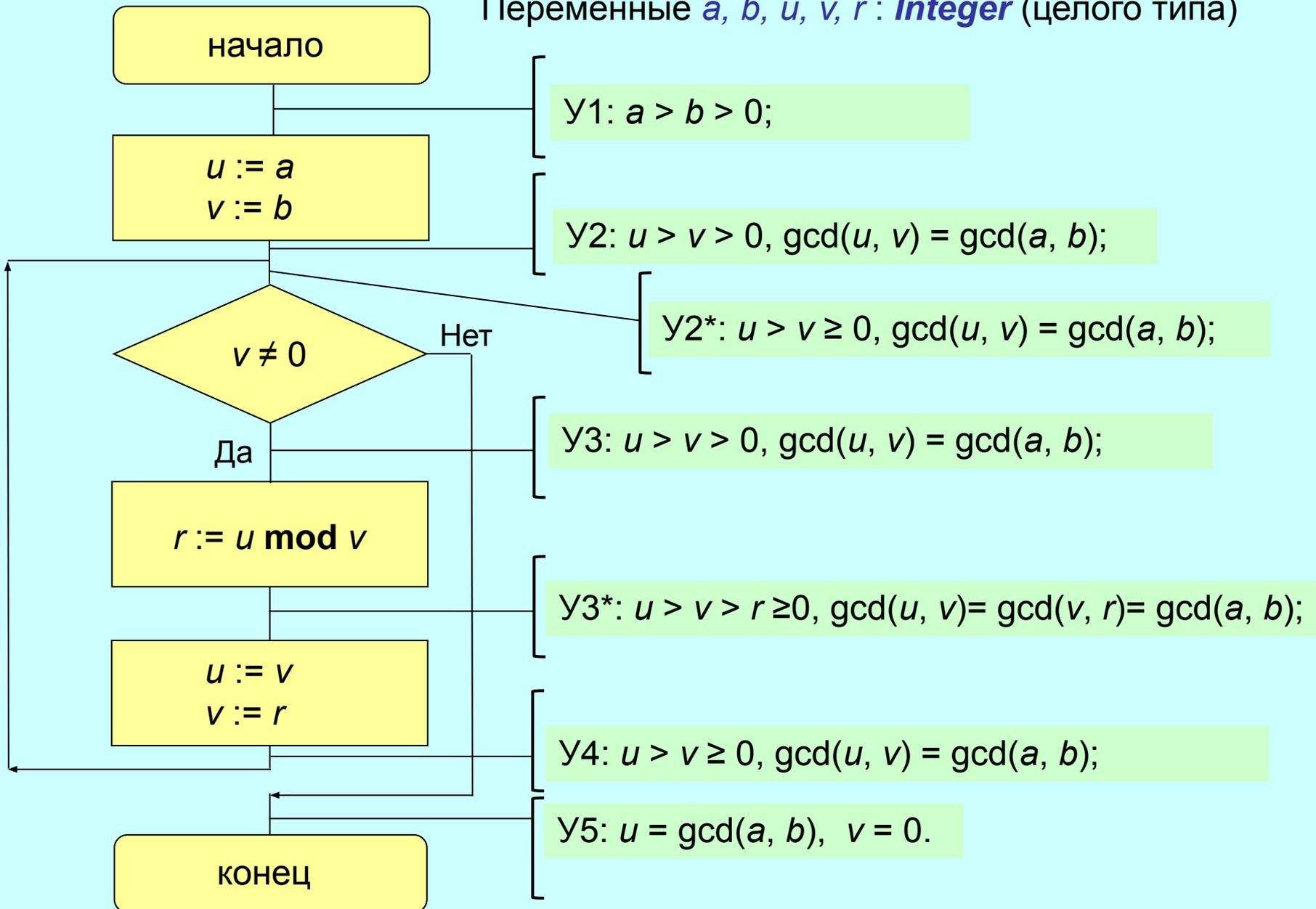
Не может существовать бесконечной строго убывающей последовательности целых неотрицательных чисел ( $c_k \geq 0$ ).

# Компьютерная запись

Отличная от «математической».

В виде блок-схемы  
(графической схемы) алгоритма

Переменные  $a, b, u, v, r$ : *Integer* (целого типа)



Задание. Ослабить ограничения на входные данные:

$a \geq b \geq 0$  и  $(a \neq 0$  или  $b \neq 0)$

$a \geq 0, b \geq 0$  (доопределить  $\text{gcd}(0, 0) = 0$ )

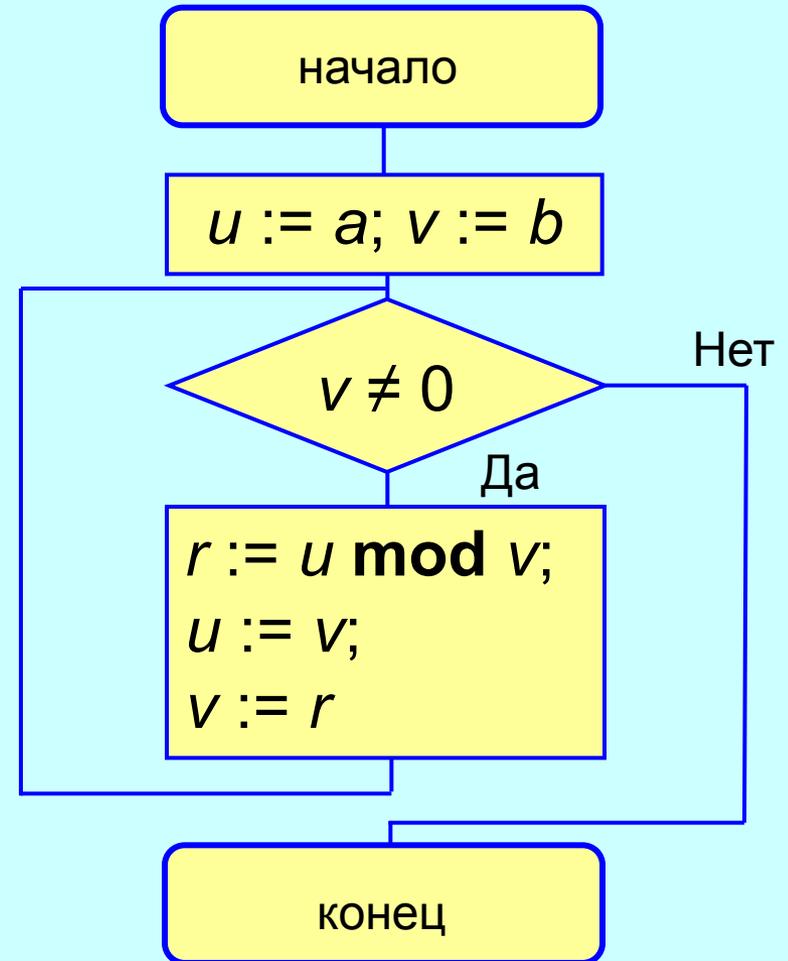
### Метод индуктивных утверждений

*Утверждение о состоянии переменных программы в некоторой её точке даётся таким образом, что оно справедливо при любом проходе вычислений через эту точку независимо от количества предыдущих проходов и от предыстории (от того, какой путь при вычислениях привёл в эту точку).*

*Правильность программы означает, что если она начала выполняться при заданном предусловии (утверждении У1) и завершилась, то после завершения будет справедливо постусловие (утверждение У5).*

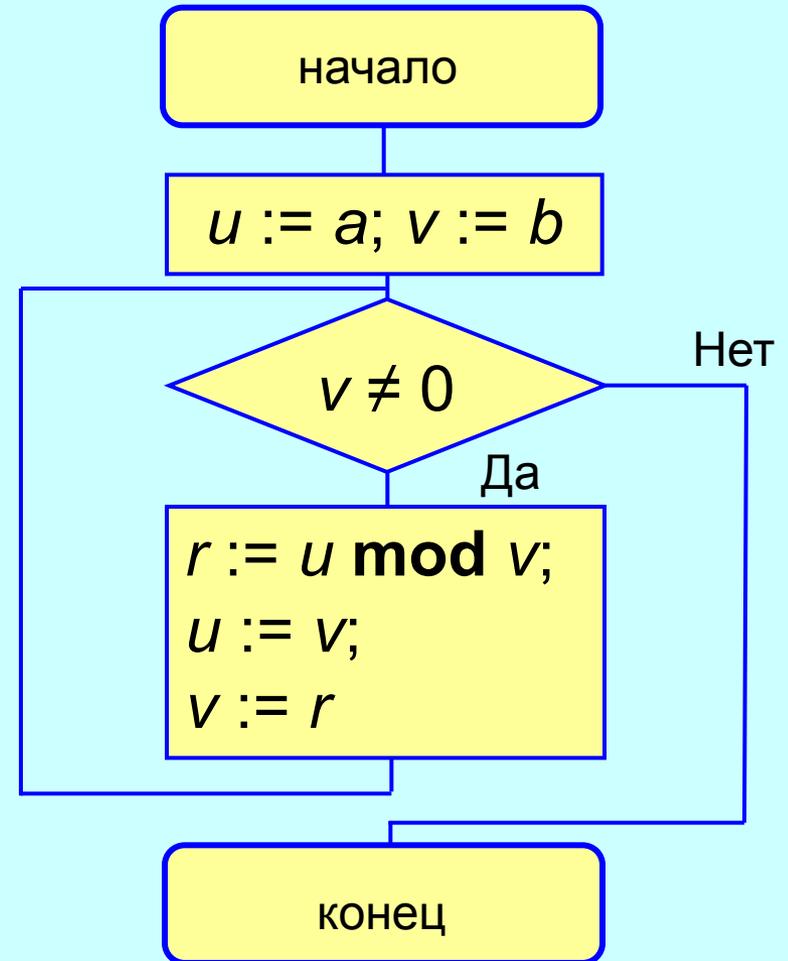
# Запись алгоритма Евклида на языке Паскаль

```
u := a; v := b;  
while v ≠ 0 do  
begin  
  r := u mod v;  
  u := v;  
  v := r;  
end
```



# Запись алгоритма Евклида на языке C++

```
u = a;  
v = b;  
while ( v != 0 )  
{  
    r = u % v;  
    u = v;  
    v = r;  
}
```



# Аннотирование программы (алгоритма)

```
// У1: Предусловие
u = a ; v = b ;
// У2: утверждение перед первым входом в цикл
while (v != 0 )
{   // У3: утверждение в точке входа в тело цикла
    r = u % v ;
    u = v ;
    v = r ;
    // У4: утверждение в точке выхода из тела цикла
}
// У5: Постусловие
```

## Утверждения У1–У5 для алгоритма Евклида

$$У1: \quad a > b > 0;$$

$$У2: \quad u > v > 0, \gcd(u, v) = \gcd(a, b);$$

$$У3: \quad u > v > 0, \gcd(u, v) = \gcd(a, b);$$

$$У4: \quad u > v \geq 0, \gcd(u, v) = \gcd(a, b);$$

$$У5: \quad u = \gcd(a, b), \quad v = 0.$$

# Аннотированный алгоритм Евклида

```
// У1:  $a > b > 0$   
 $u = a; v = b;$   
// У2:  $u > v > 0, \gcd(u, v) = \gcd(a, b)$   
while ( $v \neq 0$ )  
{  
    // У3:  $u > v > 0, \gcd(u, v) = \gcd(a, b)$   
     $r = u \% v;$   
     $u = v;$   
     $v = r;$   
    // У4:  $u > v \geq 0, \gcd(u, v) = \gcd(a, b)$   
}  
// У5:  $u = \gcd(a, b), v = 0$ 
```

```
/* Сергеев А.И., гр.8304, 7.09.2008
```

```
Лабораторная работа N 0
```

```
Greatest Common Divisor
```

```
GCD(a,b) - наибольший общий делитель натуральных a,b
```

```
примечание: пометка "Dem" в тексте указывает на  
демонстрационный фрагмент */
```

```
#include <iostream>
```

```
using namespace std ;
```

```
int main ( )
```

```
{unsigned int a,b,u,v;
```

```
unsigned int Remainder, Quotient, i; // Dem
```

```
cout << "Введите два натуральных числа: \n" ;
```

```
cin >> a >> b;
```

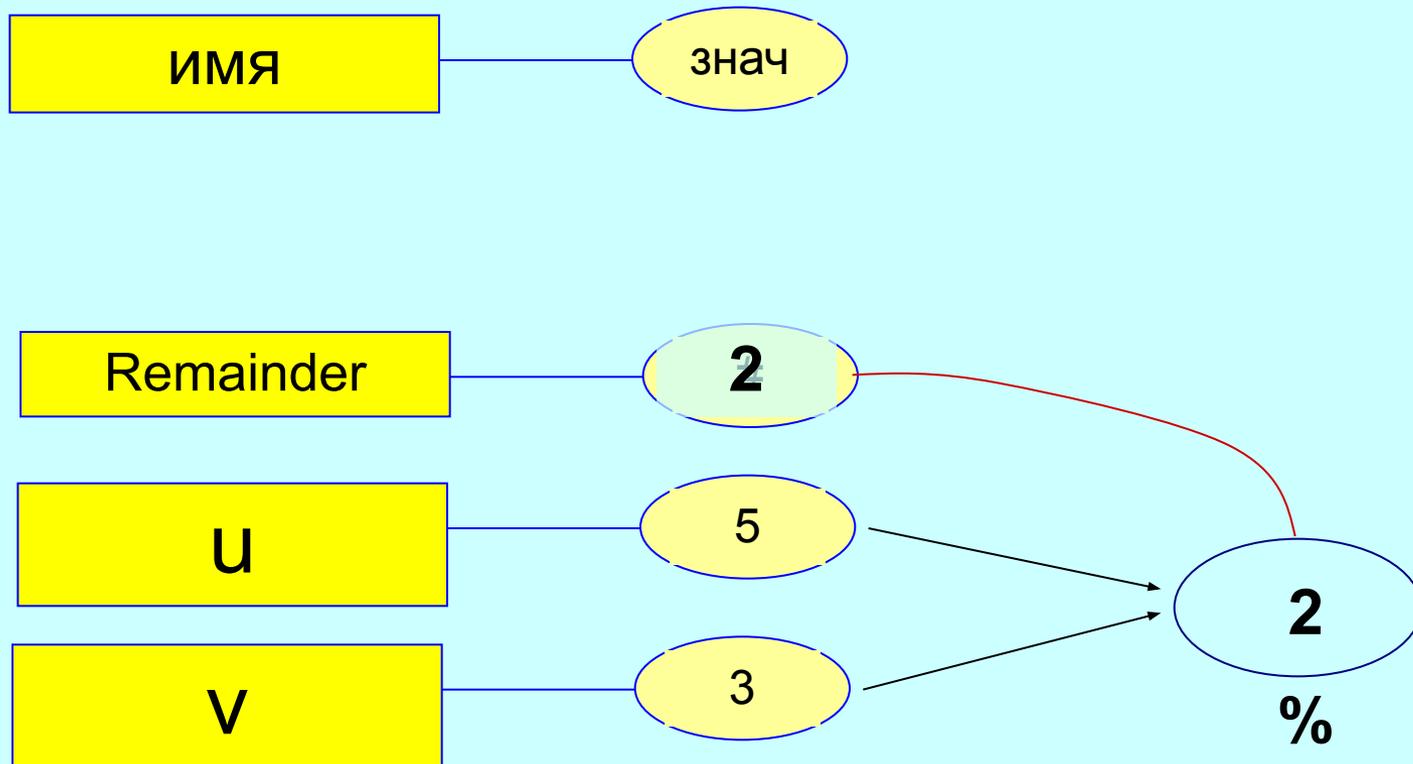
```
cout << "Находим НОД пары чисел : " << a << " , " << b << "\n";
```



```
i = 0; // Dem
u = a;
v = b;
// u>=0 & v>=0 & GCD(u,v)=GCD(a,b)
while ( v != 0 )
{ // u>=0 & v>0 & GCD(u,v)=GCD(a,b)
    i = i + 1; // Dem
    cout << "Step " << i ; // Dem
    Quotient = u / v; // Dem
    Remainder = u % v;
    cout << " :: " << u << " = " << Quotient << " * " << v << " + " <<
    Remainder << "\n"; // Dem
    u = v;
    v = Remainder;
    // u>0 & v>=0 & GCD(u,v)=GCD(a,b)
}
// u>=0 & v=0 & u=GCD(u,0)=GCD(a,b)
cout << "Результат : --> НОД(" << a << ", " << b << ") = " << u << "\n";
return 0;
}
```

## Замечание

Например,  $\text{Remainder} = u \% v$ ;



## Способ вычисления НОД на основе определения

```
// a > 0 & b > 0
if ( a < b ) c = a;
else c = b;
// c=min(a,b)}

while ( ((a % c) != 0) || ((b % c) != 0))
{c = c - 1;
};
// c = gcd(a, b)}
```

См. файл [gcd\\_w4.cpp](#)

Анализ АЕ

Отложен

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ