



ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!



Меня хорошо слышно
&& видно?



Напишите в чат, если есть проблемы!

Ставьте ☐ если все
хорошо

Добрый вечер!

Алгебраические алгоритмы



- Алгоритм Евклида
- Быстрое возведение в степень
- Решето Эратосфена
- Быстрое вычисление чисел Фибоначчи




Алгоритм нахождения НОД (наименьшего общего делителя)

- Один из старейших алгоритмов, используемых до сих пор
- III век до нашей эры



- Является основой для криптографического алгоритма с открытым ключом RSA
- Используется при решении линейных диофантовых уравнений
- Используется при построении непрерывных дробей
- Используется в методе Штурма

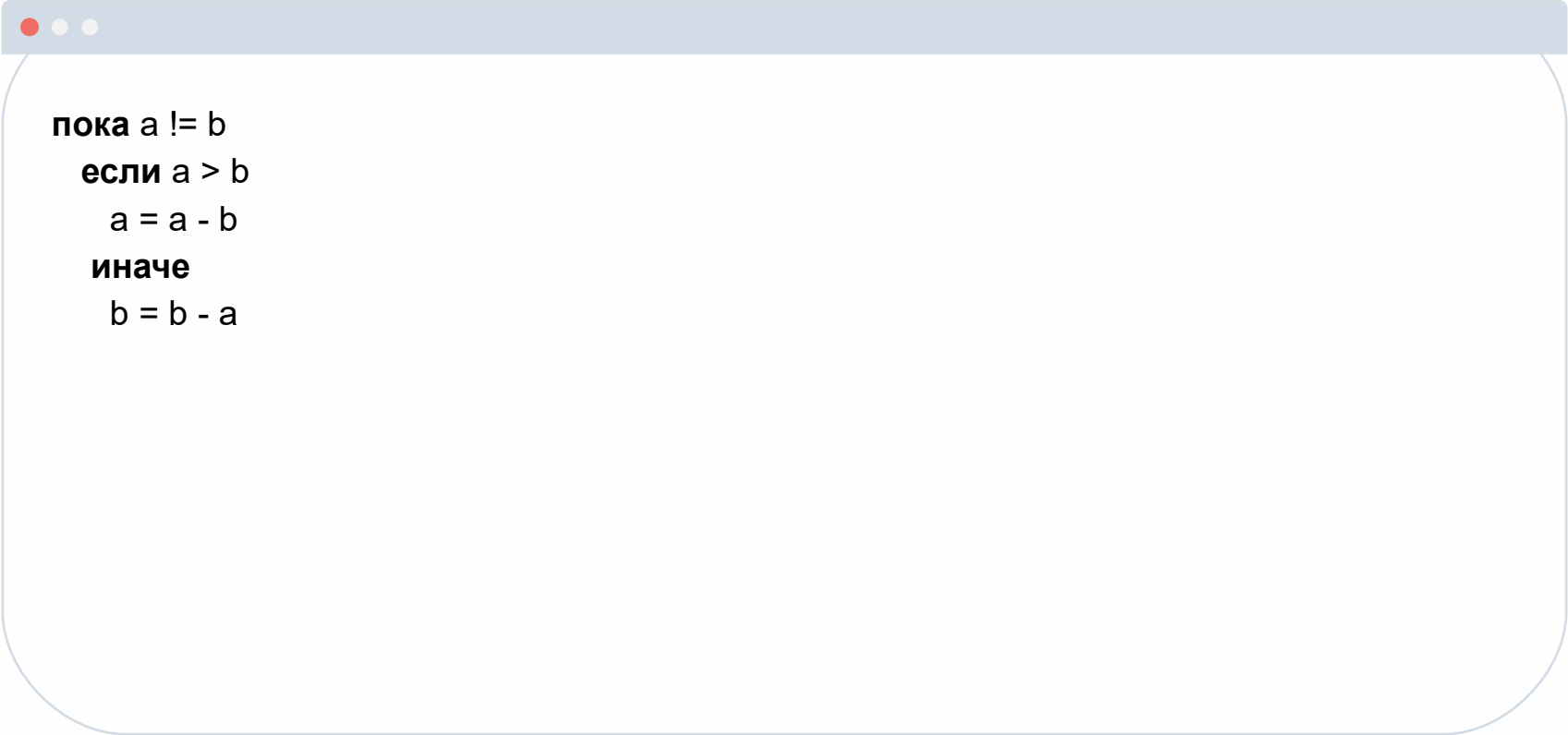
Алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары



```
пока a != b
  если a > b
    a = a - b
  иначе
    b = b - a
```

Задание - написать код за 5 минут

написать в чат НОД для 1234567890 и 12

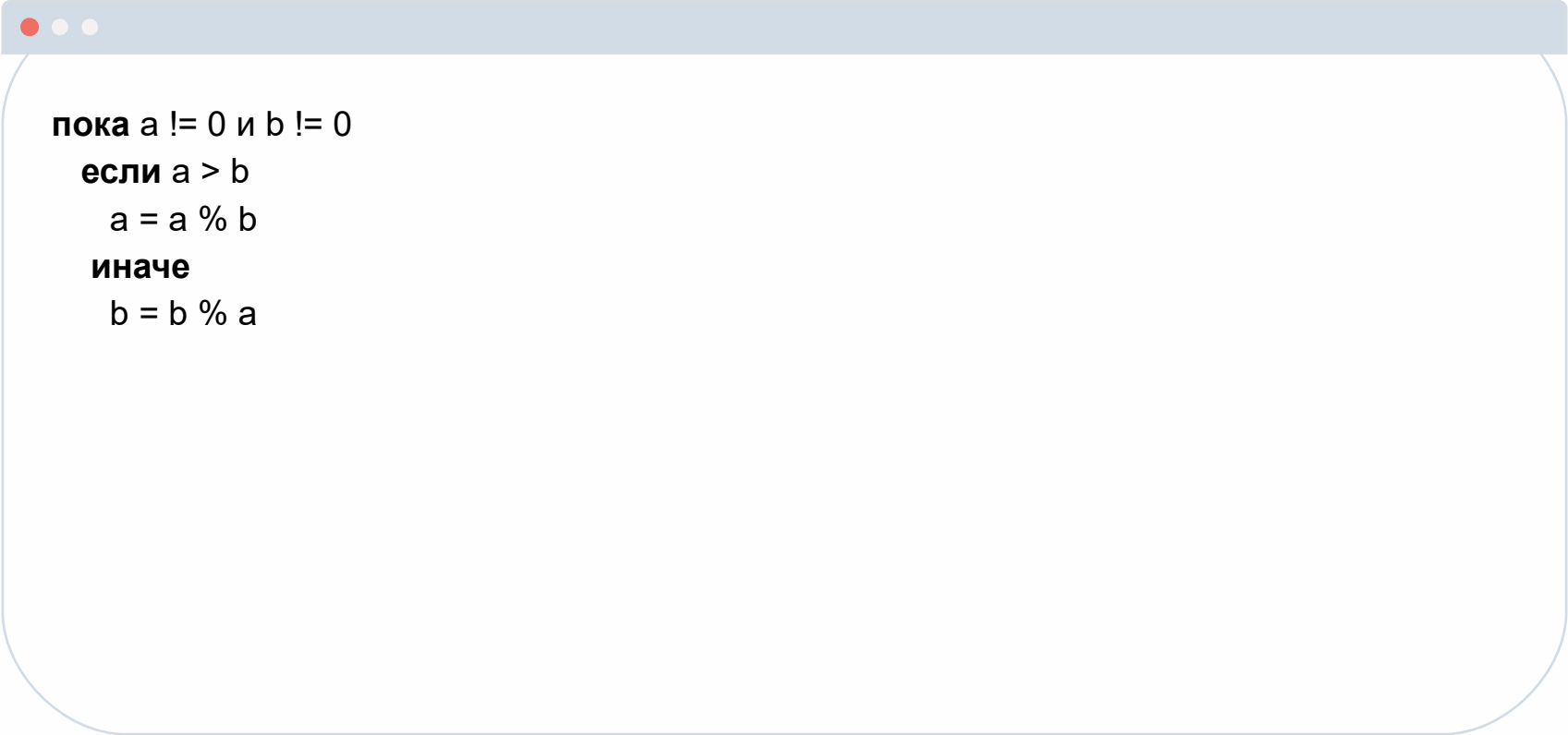


```
пока a != b  
  если a > b  
    a = a - b  
  иначе  
    b = b - a
```

Есть идеи?

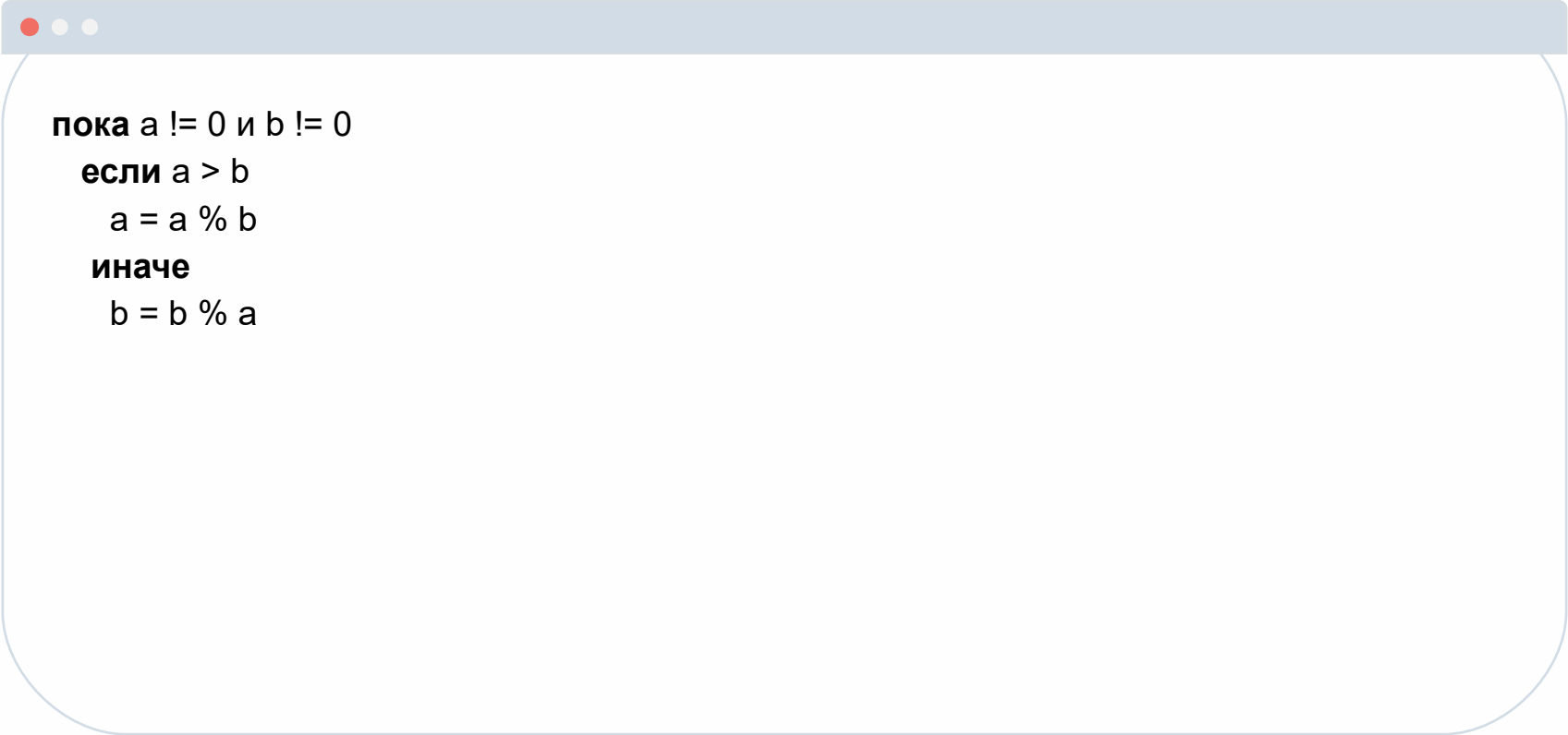


Вместо вычитания использовать остаток от деления



```
пока a != 0 и b != 0
  если a > b
    a = a % b
  иначе
    b = b % a
```

Задание - написать новую версию
написать в чат разницу в скорости



```
пока a != 0 и b != 0
  если a > b
    a = a % b
  иначе
    b = b % a
```

- Иранский математик Аль-Каши, XV век
- Как вычислить x степени n быстрее чем перемножить n раз?



- Для ускорения использовать уже вычисленные степени
- В частности, если $n = 2^k$, то вместо $n-1$ умножения, нужно сделать k

Вариант 1

Сохраняем все промежуточные вычисления, то что нужно, используем

Вариант 2

Заранее вычисляем что нам нужно, и по ходу используем

Пример

$$n = 11$$

$$11 = 8 + 2 + 1$$

$$x * x \Rightarrow 2, [2] * [2] \Rightarrow 4, [4] * [4] \Rightarrow 8, [8] * [2] \Rightarrow 10, [10] * x \Rightarrow 11$$

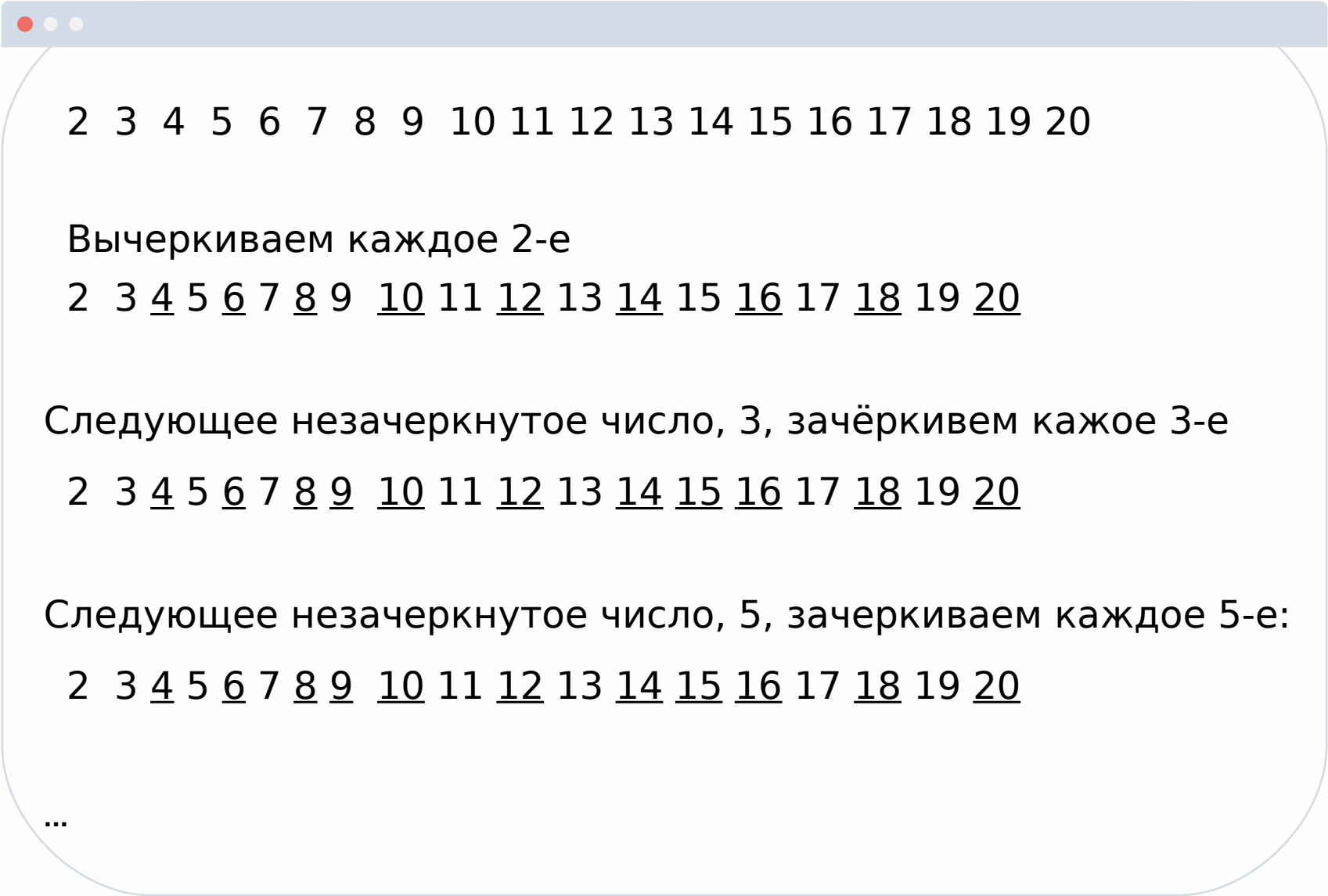
Есть вопросы?

Все понятно?



Алгоритм нахождения всех простых чисел до N





2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Вычеркиваем каждое 2-е

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Следующее незачеркнутое число, 3, зачёркиваем каждое 3-е

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Следующее незачеркнутое число, 5, зачеркиваем каждое 5-е:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

...

- Сложность алгоритма $n \log(\log(n))$

	2	3	4	5	6	7	8	9	10	Prime numbers
11	12	13	14	15	16	17	18	19	20	
21	22	23	24	25	26	27	28	29	30	
31	32	33	34	35	36	37	38	39	40	
41	42	43	44	45	46	47	48	49	50	
51	52	53	54	55	56	57	58	59	60	
61	62	63	64	65	66	67	68	69	70	
71	72	73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	89	90	
91	92	93	94	95	96	97	98	99	100	
101	102	103	104	105	106	107	108	109	110	
111	112	113	114	115	116	117	118	119	120	

- Задание - написать код за 10 минут
- Что самое важное?

- Сразу обрабатывать только нечетные числа
- Начинать просеивать с n^2 , потому что
- Заканчивать когда $n^2 > N$

● Число фибоначчи

$$F(n) = F(n-1) + F(n-2)$$

$$\text{и } F(1) = F(2) = 1.$$

$$1: 1 + 1 = 2$$

$$2: 1 + 2 = 3$$

$$3: 2 + 3 = 5$$

$$4: 3 + 5 = 8$$

$$5: 5 + 8 = 13$$

$$6: 8 + 13 = 21$$

$$7: 13 + 21 = 34$$

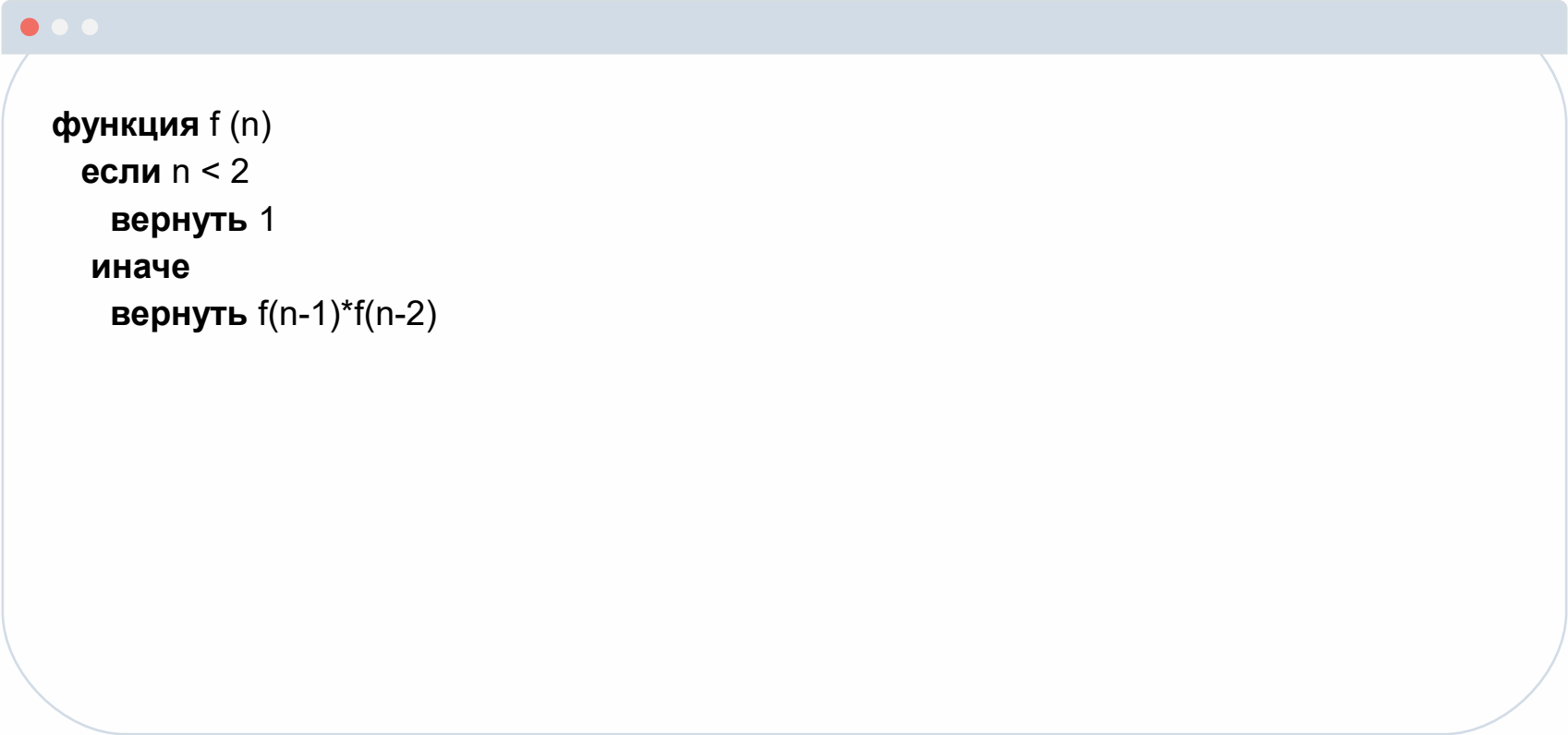
$$8: 21 + 34 = 55$$

$$9: 34 + 55 = 89$$

... и т.

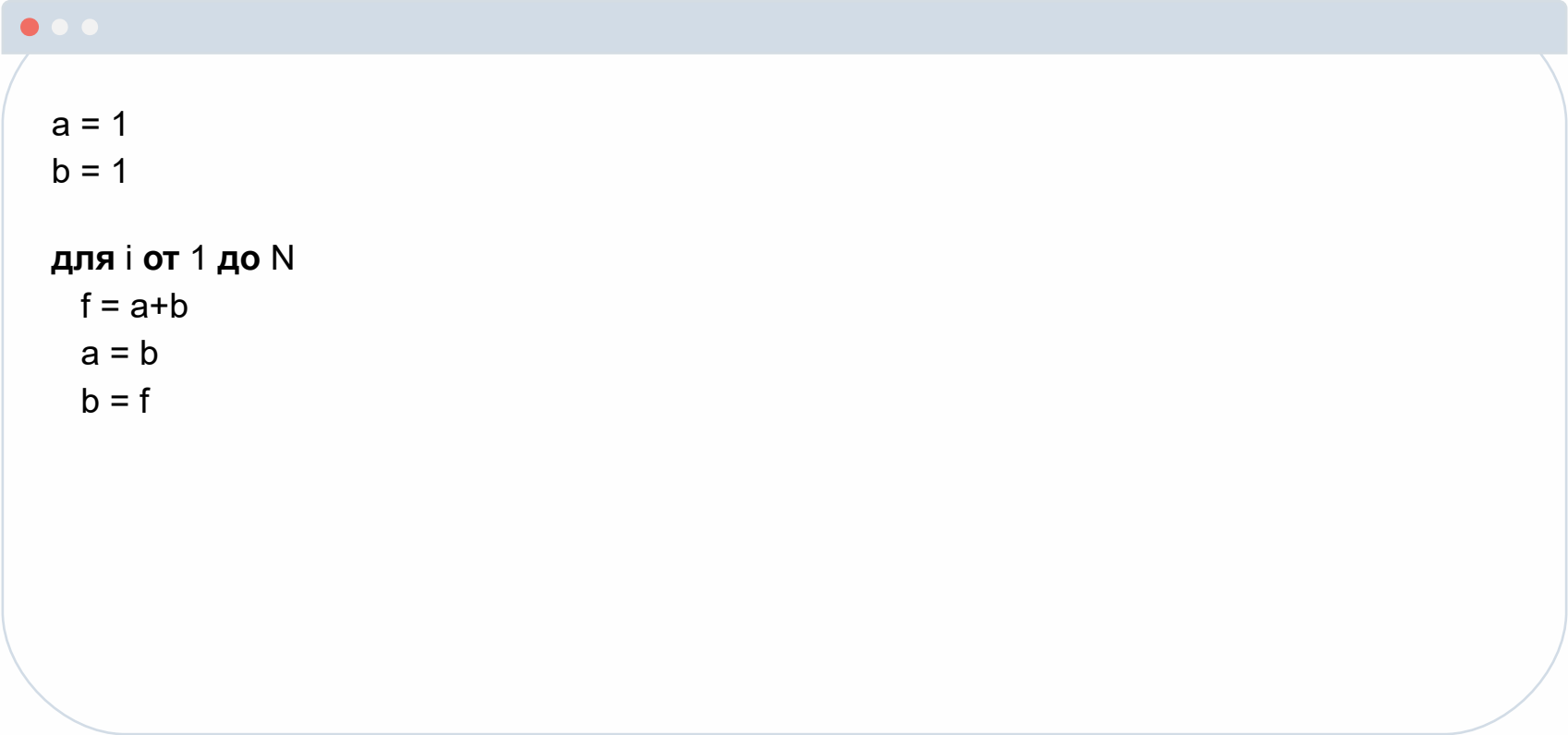


рекурсия



```
функция f (n)  
  если n < 2  
    вернуть 1  
  иначе  
    вернуть f(n-1)*f(n-2)
```

Динамическое программирование



```
a = 1
```

```
b = 1
```

```
для i от 1 до N
```

```
  f = a+b
```

```
  a = b
```

```
  b = f
```


Рекуррентная формула через золотое сечение

$$x^n = F_n = F_{n-1} + F_{n-2} = x^{n-1} + x^{n-2},$$

$$\phi = \frac{1 + \sqrt{5}}{2}$$

$$F_n = \left\lfloor \frac{\phi^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor.$$

Задание - написать код за 10 минут

- золотое сечение
- динамическое программирование
- Вычислить для $N = 1200$
- Поставить ! в чат кто будет готов

Матричный алгоритм

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}.$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}.$$

Умножение матриц

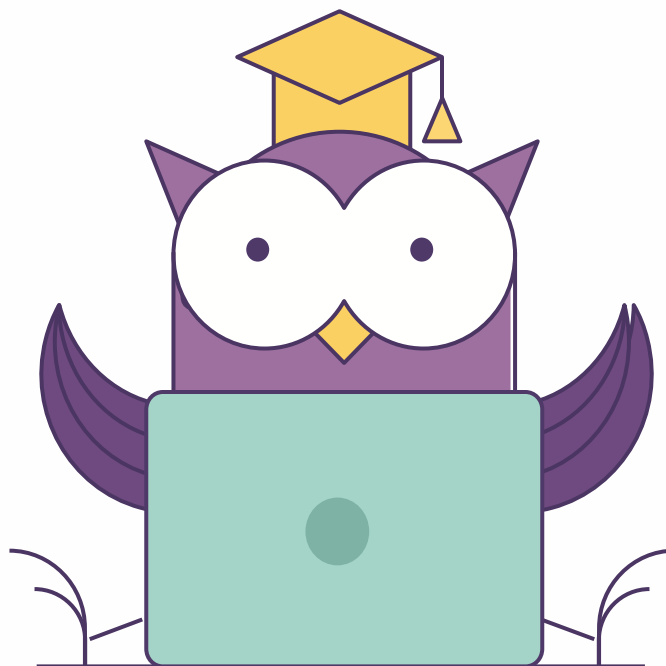
$$A = \begin{pmatrix} 5 & 2 \\ 3 & 1 \end{pmatrix} B = \begin{pmatrix} 4 & 6 \\ 5 & 2 \end{pmatrix}$$

$$A \cdot B = \begin{pmatrix} 5 & 2 \\ 3 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 6 \\ 5 & 2 \end{pmatrix} = \begin{pmatrix} 5 \cdot 4 + 2 \cdot 5 & 5 \cdot 6 + 2 \cdot 2 \\ 3 \cdot 4 + 1 \cdot 5 & 3 \cdot 6 + 1 \cdot 2 \end{pmatrix} = \begin{pmatrix} 30 & 34 \\ 17 & 20 \end{pmatrix}$$

$$B \cdot A = \begin{pmatrix} 4 & 6 \\ 5 & 2 \end{pmatrix} \cdot \begin{pmatrix} 5 & 2 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 4 \cdot 5 + 6 \cdot 3 & 4 \cdot 2 + 6 \cdot 1 \\ 5 \cdot 5 + 2 \cdot 3 & 5 \cdot 2 + 2 \cdot 1 \end{pmatrix} = \begin{pmatrix} 38 & 14 \\ 31 & 12 \end{pmatrix}$$

Опять-таки $A \cdot B \neq B \cdot A$.

- Алгоритм Евклида
- Быстрое возведение в степень
- Решето Эратосфена
- Быстрое вычисление чисел Фибоначчи



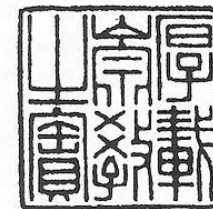
Числа Фибоначчи через перемножение матриц



Китайский математик

Сунь Цзы

~ III - V век н.э.



Если **натуральные числа** a_1, a_2, \dots, a_n **попарно взаимно просты**, то для любых целых r_1, r_2, \dots, r_n таких, что $0 \leq r_i < a_i$ при всех $i \in \{1, 2, \dots, n\}$, найдётся число N , которое при делении на a_i даёт остаток r_i при всех $i \in \{1, 2, \dots, n\}$. Более того, если найдутся два таких числа N_1 и N_2 , то $N_1 \equiv N_2 \pmod{a_1 \cdot a_2 \cdot \dots \cdot a_n}$.

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

Ответ 23

$$23 \% 3 = 2$$

$$23 \% 5 = 3$$

$$23 \% 7 = 2$$



**Тактика без стратегии –
это просто суета
перед поражением.**

Сунь-цзы

- Восстановление числа по остаткам
- В криптографическом алгоритме RSA при расшифровке

Шаг 1. Вычисляем $M = \prod_{i=1}^n a_i$.

Шаг 2. Для всех $i \in \{1, 2, \dots, n\}$ находим $M_i = \frac{M}{a_i}$.

Шаг 3. Находим $M_i^{-1} = \frac{1}{M_i} \bmod a_i$ (например, используя [расширенный алгоритм Евклида](#)).

Шаг 4. Вычисляем искомое значение по формуле $x = \sum_{i=1}^n r_i M_i M_i^{-1} \bmod M$.

```
for (int i = 0; i < n; i++) {  
    x[i] = remainders[i];  
    for (int j = 0; j < i; j++) {  
        x[i] = inverses[j][i] * (x[i] - x[j]);  
        x[i] = x[i] % a[i];  
        if (x[i] < 0)  
            x[i] += a[i];  
    }  
}
```

Спасибо
за внимание!



Заполните, пожалуйста, опрос о
занятии

