

# Программирование (Python)

- § 17. Введение
- § 18. Линейные программы
- § 19. Ветвления
- § 20. Программирование  
циклических алгоритмов
- § 21. Массивы
- § 22. Алгоритмы обработки  
массивов

# Программирование (Python)

## § 17. Введение

# Что такое программирование?

**Программирование** — это создание программ для компьютеров. Этим занимаются **программисты**.

Чем занимаются **программисты**:

**анализ задачи** (выделение исходных данных, связей между ними, этапов решения задачи)

системные аналитики

разработка **алгоритмов**

алгоритмисты

написание и отладка **программ**

кодировщики

**тестирование** программ

тестировщики

написание **документации**

технические писатели

# Направления в программировании

---

**системный программист**

операционные системы,  
утилиты, драйверы

**прикладной программист**

прикладные программы, в  
т.ч. для мобильных  
устройств

**веб-программист**

веб-сайты

**программист баз данных**

системы управления  
базами данных

# Простейшая программа

```
# Это пустая программа
```



Что делает эта программа?

комментарии после #  
не обрабатываются

кодировка utf-8  
по умолчанию)

```
# coding: utf-8
```

```
# Это пустая программа
```

```
"""
```

```
Это тоже комментарий
```

```
"""
```

# Вывод на экран

оператор  
вывода

**Оператор** — это команда  
языка программирования.

```
print ( "Привет!" )
```

```
print ( "Привет", Вася! )
```



Что плохо?

```
print ( "Привет, Вася!" )
```

вся строка в  
кавычках

## Переход на новую строку

---

```
print ( "Привет, Вася!" )  
print ( "Привет, Петя!" )
```

Результат:

```
Привет, Вася!  
Привет, Петя!
```

переход на новую строку автоматически

Нужно в одной строке:

```
Привет, Вася!Привет, Петя!
```

Решение:

```
print ( "Привет, Вася!" , end="" )  
print ( "Привет, Петя!" )
```

после вывода данных ничего не выводить

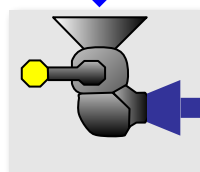
# Системы программирования

**Системы программирования** — это средства для создания новых программ.

**Транслятор** — это программа, которая переводит тексты программ, написанных программистом, в машинные коды (команды процессора).

- **компилятор** — переводит всю программу в машинные коды, строит исполняемый файл (**.exe**)

```
program Hello;  
begin  
  write('Привет!')  
end.
```



**privet.exe**

- **интерпретатор** — сам выполняет программу по частям (по одному оператору).



**Python – интерпретатор!**



# Системы программирования

**Отладчик** — это программа для поиска ошибок в других программах.

- **пошаговый режим** — выполнение программы по шагам (по одному оператору)
- **просмотр значений переменных** во время выполнения программы
- **точки останова** – операторы в программе, перед выполнением которых нужно остановиться.

**Среда программирования (IDE):**

- редактор текста программ
- транслятор
- отладчик

# Задачи

---

«В»: Вывести на экран текст «лесенкой»

Вася

пошел

гулять

«С»: Вывести на экран рисунок из букв

```
Ж
ЖЖЖ
ЖЖЖЖЖ
ЖЖЖЖЖЖЖ
НН НН
ZZZZZ
```

# Программирование (Python)

## § 18. Линейные программы

# Пример задачи

---

*Задача.* Ввести два числа и вычислить их сумму.

```
# ввести два числа  
# вычислить их сумму  
# вывести сумму на экран
```



Выполнится?

**Псевдокод** – алгоритм на русском языке с элементами языка программирования.



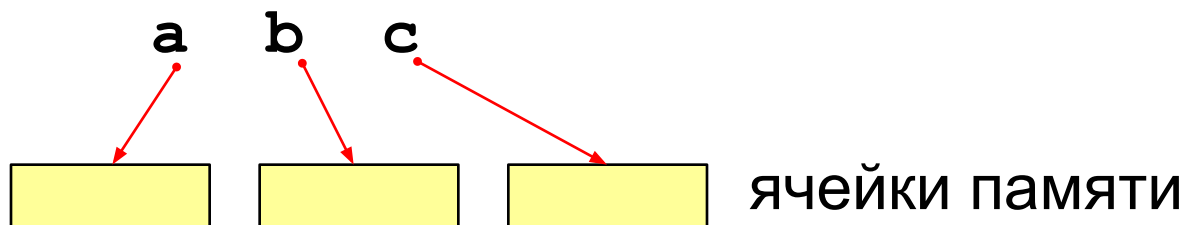
Компьютер не может исполнить псевдокод!

# Зачем нужны переменные?

```
# ввести два числа  
# вычислить их сумму  
# вывести сумму на экран
```

Где запомнить?

**Переменная** — это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.



# Имена переменных

**Идентификатор** — это имя программы или переменной.

a b c

заглавные и строчные  
буквы **различаются**

**МОЖНО** использовать

- латинские буквы (A-Z, a-z)
- цифры



Имя не может начинаться с цифры!

- знак подчеркивания \_

**НЕЛЬЗЯ** использовать ~~скобки, знаки ", &, |, \*, +, =, !, ? и др.~~

Какие имена правильные?

**AXby R&B 4Wheel Вася "PesBarbos"**  
**TU154 [QuQu] \_ABBA A+B**

# Работа с переменными

## Присваивание (запись значения)

```
a = 5
```

оператор  
присваивания

$a \leftarrow 5$

```
a = 5  
a = 18
```

?

Что будет храниться в *a*?

## Вывод на экран

```
print(a)
```

?

В чём разница?

```
c = 14  
print(c)
```

14

```
c = 14  
print("c")
```

c

# Работа с переменными

## Изменение значения

```
i = i + 1
```

увеличить на 1

$$i \leftarrow i + 1$$

```
a = 4
b = 7
a = a + 1
b = b + 1
a = a + b
b = b + a
a = a + 2
b = b + a
```

a	b
4	
	7
5	
	8
13	
	21
15	
	36

Python:

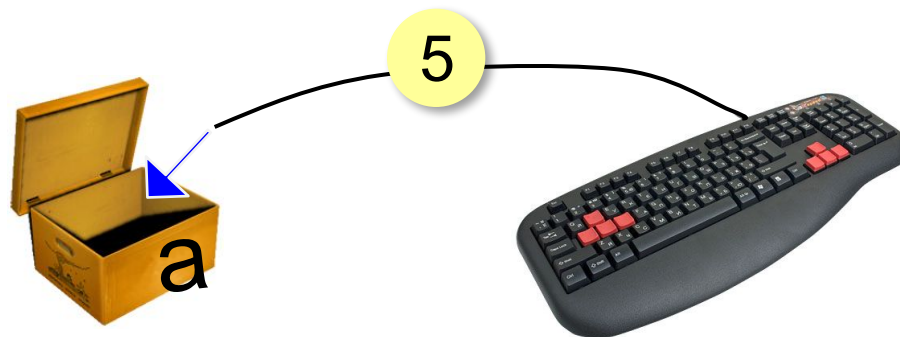
```
a, b = 4, 7
a += 1
b += 1
a += b
b += a
a += 2
b += a
```



# Ввод с клавиатуры

Цель – изменить исходные данные, не меняя программу.

```
a = input()
```



1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

# Ввод с клавиатуры

```
a = input ()
```

ввести строку с клавиатуры  
и связать с переменной a

```
b = input ()
```

```
c = a + b
```

```
print ( c )
```

Протокол:

21

33

2133



Почему?



Результат функции `input` – строка символов!

преобразовать в  
целое число

```
a = int ( input () )
```

```
b = int ( input () )
```

## Ввод с подсказкой

Введите число: 26

```
a = input("Введите число: ")
```



Что не так?

подсказка

```
a = int(input("Введите число: "))
```



Что будет?

преобразовать  
в целое число

Введите число: Qu-Qu

**ValueError: invalid literal for int() with base 10: 'Qu-Qu'**

# Ввод вещественных чисел

---

```
print( "Введите число:" )  
x = float( input() )
```

или так:

```
x = float( input("Введите число:") )
```

# Программа сложения чисел

```
a = int ( input ( ) )  
b = int ( input ( ) )  
c = a + b  
print ( c )
```



Что плохо?

ожидание:

Введите два числа:

5

7

5+7=12

реальность:

5

7

12



Как улучшить диалог?

# Вывод данных с текстом

значение *a*

значение *b*

значение *c*

5+7=12

ТЕКСТ

```
print(a, "+", b, "=", c)
```

ожидание:

5+7=12

реальность:

5 + 7 = 12

это пробелы не заказывали!

```
print(a, "+", b, "=", c, sep=" ")
```

*separator*

пустой

# Программа сложения чисел

---

```
print ( "Введите два числа: " )  
a = int ( input () )  
b = int ( input () )  
c = a + b  
print ( a, "+", b, "=", c, sep="" )
```



Как переделать для 3-х чисел?

# Ввод двух чисел в одной строке

```
a, b = map ( int, input().split() )
```

21 33

`input()`

ввести строку с клавиатуры

21 33

`input().split()`

целые

применить

разделить строку на части по пробелам

21 33

`map ( int, input().split() )`

эту операцию

к каждой части

```
a, b = map ( int, input().split() )
```



# Задачи

---

**«А»:** Ввести три числа, найти их сумму.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

**«В»:** Ввести три числа, найти их сумму и произведение.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

$$4*5*7=140$$

# Задачи

---

**«С»:** Ввести три числа, найти их сумму, произведение и среднее арифметическое.

*Пример:*

Введите три числа:

4

5

7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7) / 3 = 5.333333$$

# Арифметические выражения

---

$$a \leftarrow \frac{c + b - 1}{2} \cdot d$$

Линейная запись (в одну строку):

```
a = (c + b - 1) / 2 * d
```

Операции: + -

\* – умножение

/ – деление

\*\* – возведение в степень ( $x^2 \rightarrow \mathbf{x**2}$ )

# Порядок выполнения операций

3      1    2      4            5    6

```
a = (c + b**5*3 - 1) / 2 * d
```

**Приоритет** (старшинство):

- 1) скобки
- 2) возведение в степень \*\*
- 3) умножение и деление
- 4) сложение и вычитание

```
a = (c + b**5*3 - 1) \
      / 2 * d
```

```
a = (c + b**5*3
      - 1) / 2 * d
```

$$a = \frac{c + b^5 \cdot 3 - 1}{2} \cdot d$$

перенос на  
следующую строку

перенос внутри  
скобок разрешён

# Деление

---

Классическое деление:

```
a = 9; b = 6
x = 3 / 4      # = 0.75
x = a / b      # = 1.5
x = -3 / 4     # = -0.75
x = -a / b     # = -1.5
```

Целочисленное деление (округление «вниз»!):

```
a = 9; b = 6
x = 3 // 4     # = 0
x = a // b     # = 1
x = -3 // 4    # = -1
x = -a // b    # = -2
```

# Частное и остаток

---

// – деление нацело (остаток отбрасывается)

% – остаток от деления

175 сек = 2 мин 55 сек



Как получить 2 и 55?


```
t = 175
```

```
m = t // 60
```

```
s = t % 60
```

# Частное и остаток

---

 Что получится?

```
n = 123
d = n // 10
k = n % 10
```

При делении на 10 нацело отбрасывается последняя цифра числа.

Остаток от деления на 10 – это последняя цифра числа.

# Операторы // и %

```
a = 1234
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10 :
```

4

3

2

1



# Сокращенная запись операций

```
a += b # a = a + b
a -= b # a = a - b
a *= b # a = a * b
a /= b # a = a / b
a //= b # a = a // b
a %= b # a = a % b
```

```
a += 1
```

увеличение на 1

# ФОРМАТНЫЙ ВЫВОД

```
a = 1; b = 2; c = 3
print( a, b, c )
```

1 2 3

форматная строка

```
print( "{}{}{}".format( a, b, c ) )
```

123

ТУТ НУЖНО ЧТО-ТО  
ВЫВЕСТИ

```
print( "{}{:3}{:5}".format( a, b, c ) )
```

КОЛИЧЕСТВО ЗНАКОВ  
НА ВЫВОД ЧИСЛА

1 2 3  
3 5



Сколько знаков для вывода *a*?

# ФОРМАТНЫЙ ВЫВОД

---

```
a = 1; b = 2
```

```
print("{}+{}={}".format(a,b,c))
```



```
1+2=3
```

# Задачи

---

«А»: Ввести число, обозначающее количество секунд.  
Вывести то же самое время в минутах и секундах.

**Пример:**

Введите число секунд: **175**  
2 мин. 55 с.

«В»: Ввести число, обозначающее количество секунд.  
Вывести то же самое время в часах, минутах и секундах.

**Пример:**

Введите число секунд: **8325**  
2 ч. 18 мин. 45 с

# Задачи

---

«С»: Занятия в школе начинаются в 8-30. Урок длится 45 минут, перерывы между уроками – 10 минут. Ввести номер урока и вывести время его окончания.

**Пример:**

**Введите номер урока : 6**  
**13-50**

# ФОРМАТНЫЙ ВЫВОД

```
x=12.345678  
print("x={}".format(x))
```

x=12.345678

всего на  
число

в дробной  
части

```
print("x={:10.3f}".format(x))
```

→     12.346

          3

       10

```
print("x={:8.2f}".format(x))
```

→   12.34

# ФОРМАТНЫЙ ВЫВОД

```
print ("x={:2.2f}".format(x))
```

→ 12.34

```
print ("x={:.2f}".format(x))
```

→ 12.34

МИНИМАЛЬНО  
ВОЗМОЖНОЕ

```
print ("x={:0.1f}".format(x))
```

→ 12.3

# Научный формат чисел

---

```
x=123456789
```

```
print ("x={:e}".format(x))
```

→ 1.234568e+008

1,234568 · 10<sup>8</sup>

```
x=0.0000123456789
```

```
print ("x={:e}".format(x))
```

→ 1.234568e-005

1,234568 · 10<sup>-5</sup>



# Операции с вещественными числами

---

**int** – целая часть числа

```
x=1.6  
print(int(x))
```



1

**round** – ближайшее целое число

```
x=-1.2  
print(round(x))
```



-1

# Математические функции

загрузить  
модуль `math`

= подключить математические  
функции

```
import math
# квадратный корень
print( math.sqrt(25) )
r = 50 # радиус окружности
print( 2*math.pi*r )
print( math.pi*r**2 )
```



Что считаем?

число  $\pi$

# Операции с вещественными числами

$$1/3 = 0,33333\dots$$

бесконечно много знаков



Большинство вещественных чисел хранятся в памяти компьютера с ошибкой!

```
x = 1/2
y = 1/3
z = 5/6 # 5/6=1/2+1/3
print(x+y-z)
```



`-1.110223e-016`

# Задачи

---

**«А»:** Ввести число, обозначающее размер одной фотографии в Мбайтах. Определить, сколько фотографий поместится на флэш-карту объёмом 2 Гбайта.

## Пример:

Размер фотографии в Мбайтах: **6.3**

Поместится фотографий: 325.

# Задачи

---

«В»: Оцифровка звука выполняется в режиме стерео с частотой дискретизации 44,1 кГц и глубиной кодирования 24 бита. Ввести время записи в минутах и определить, сколько Мбайт нужно выделить для хранения полученного файла (округлить результат в большую сторону).

## Пример:

Введите время записи в минутах: **10**

Размер файла **152 Мбайт**

# Задачи

---

«С»: Разведчики-математики для того, чтобы опознать своих, используют числовые пароли. Услышав число-пароль, разведчик должен возвести его в квадрат и сказать в ответ первую цифру дробной части полученного числа. Напишите программу, которая по полученному паролю (вещественному числу) вычисляет число-ответ.

**Пример:**

**Введите пароль: 1.92**

**Ответ: 6**

потому что  $1,92^2 = 3,6864\dots$ , первая цифра дробной части – 6

# Случайные и псевдослучайные числа

## Случайные явления

- встретил слона – не встретил слона
- жеребьёвка на соревнованиях
- лотерея
- случайная скорость (направление выстрела ) в игре
- ...



**Случайные числа** — это последовательность чисел, в которой невозможно предсказать следующее число, даже зная все предыдущие.

# Случайные и псевдослучайные числа

**!** Компьютер неслучаен!

**Псевдослучайные числа** — похожи на случайные, но строятся по формуле.

следующее

предыдущее

$$X_{n+1} = (a * X_n + b) \% c \# \text{от } 0 \text{ до } c-1$$

$$X_{n+1} = (X_n + 3) \% 10 \# \text{от } 0 \text{ до } 9$$

$X = 0 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 2 \rightarrow 5 \rightarrow 8$

$8 \rightarrow 1 \rightarrow 4 \rightarrow 7 \rightarrow 0$

зерно

зацикливание



# Датчик случайных чисел

## Целые числа на отрезке:

ПОДКЛЮЧИТЬ функцию `randint`  
ИЗ модуля `random`

```
from random import randint
K = randint(1, 6) # отрезок [1, 6]
L = randint(1, 6) # это уже другое число!
```

англ. *integer* – целый  
*random* – случайный



Не нужно имя модуля!

```
K = random.randint(1, 6)
```

# Датчик случайных чисел

---

## Вещественные числа:

```
from random import random, uniform
x = random()           # полуинтервал [0,1)
y = 7*random()
z = 7*random()+5
```

## Вещественные числа на отрезке [a, b]:

```
from random import uniform
x = uniform(1.5, 2.8)  # [1,5; 2,8]
y = uniform(5.25, 12.75) # [5,25; 12,75]
```

# Задачи

---

- «А»: В игре «Русское лото» из мешка случайным образом выбираются бочонки, на каждом из которых написано число от 1 до 90. Напишите программу, которая выводит наугад первые 5 выигрышных номеров.
- «В»: + Доработайте программу «Русское лото» так, чтобы все 5 значений гарантированно были бы разными (используйте разные диапазоны).

# Задачи

---

«С»: + Игральный кубик бросается три раза (выпадает три случайных значения). Из этих чисел составляется целое число, программа должна найти его квадрат.

## Пример:

Выпало очков:

1 2 3

Число 123

Его квадрат 15129

# Задачи

---

**«D»:** + Получить случайное трёхзначное число и вывести в столбик его отдельные цифры.

**Пример:**

Получено число 123

сотни: 1

десятки: 2

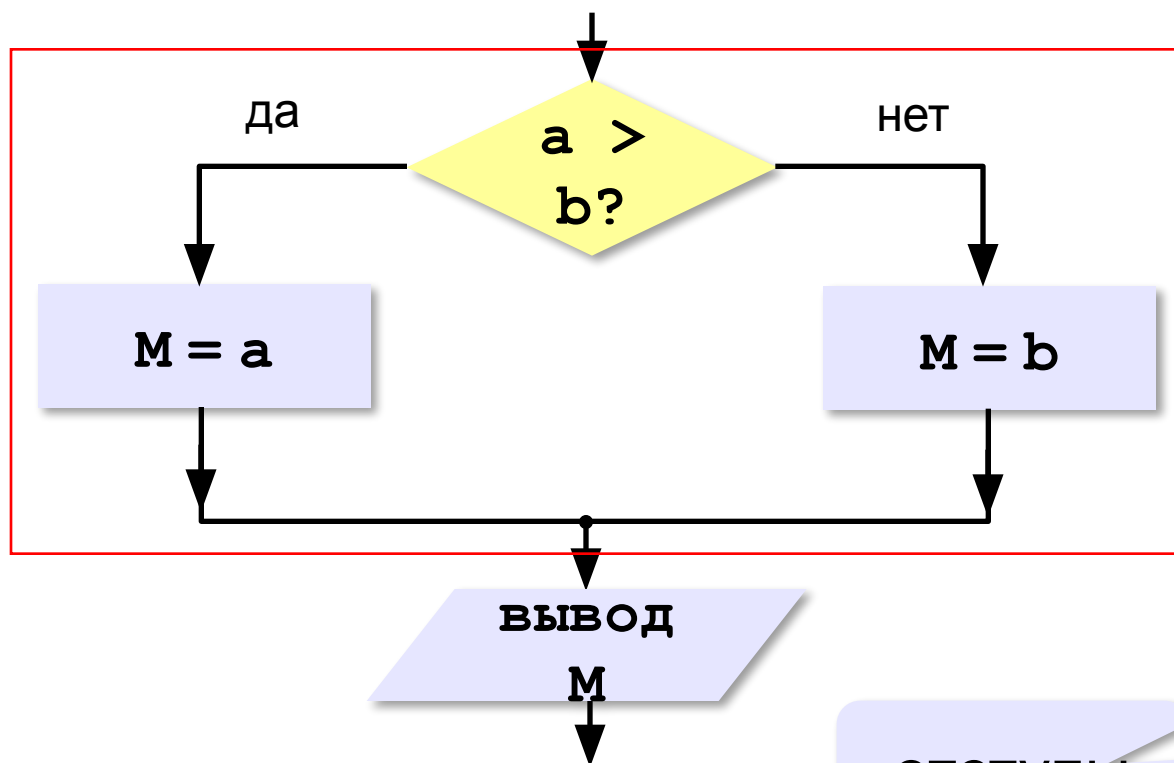
единицы: 3

# Программирование (Python)

## § 19. Ветвления

# Выбор наибольшего из двух чисел

Задача: **изменить порядок действий** в зависимости от выполнения некоторого условия.



полная  
форма  
ветвления



Если  $a = b$ ?

```
if a > b:
    M = a
else:
    M = b
```

отступы

# Вариант 1. Программа

```
print("Введите два целых числа")  
a = int(input())  
b = int(input())
```

```
if a > b:
```

```
    M = a
```

```
else:
```

```
    M = b
```

полная форма  
условного  
оператора

```
print("Наибольшее число", M)
```

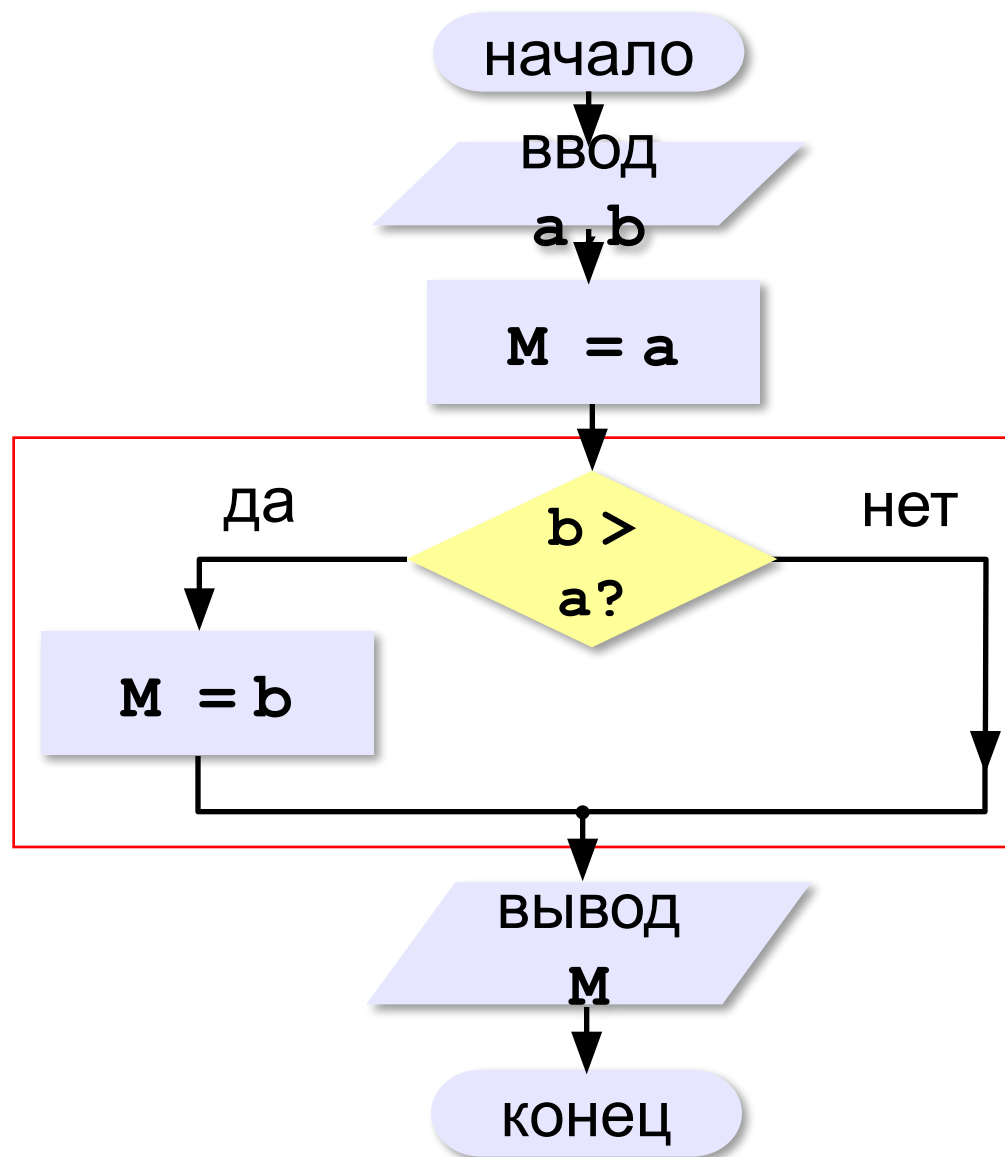
Решение в стиле Python:

```
M = max(a, b)
```

```
M = a if a > b else b
```



# Выбор наибольшего из двух чисел-2



неполная  
форма  
ветвления

## Вариант 2. Программа

```
print("Введите два целых числа")
a = int(input())
b = int(input())
M = a
if b > a:
    M = b
print("Наибольшее число", M)
```

неполная форма  
условного  
оператора

# Примеры

---

## Поиск минимального:

```
if a < b:  
    M = a  
if b < a:  
    M = b
```



Что плохо?



Когда работает неверно?

# Примеры

```
if a < b:
```

```
    c = a
```

```
    a = b
```

```
    b = c
```



Что делает эта программа?



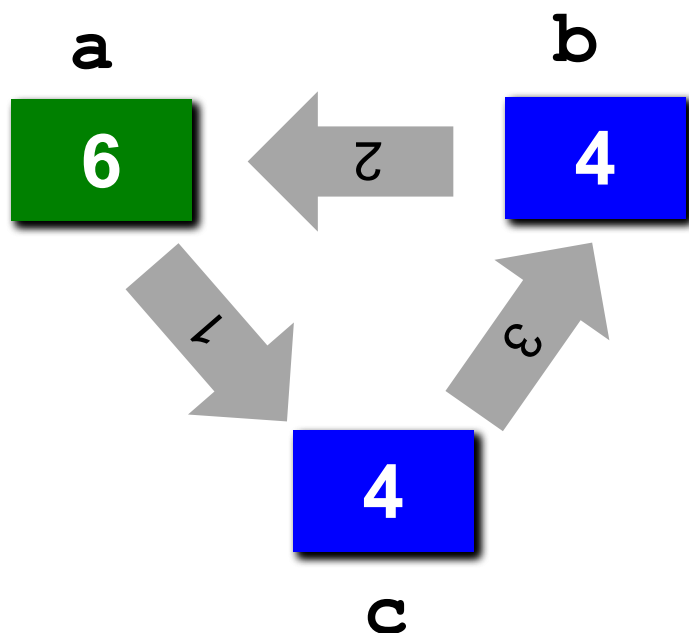
В чём отличие?

```
if a < b:
```

```
    c = a
```

```
    a = b
```

```
    b = c
```



Решение в стиле Python:

```
a, b = b, a
```

# В других языках программирования

---

## Паскаль:

```
if a < b then begin
    c := a;
    a := b;
    b := c;
end;
```

## C:

```
if (a < b) {
    c = a;
    a = b;
    b = c;
}
```

# Знаки отношений

---

**>** **<** больше, меньше

**>=** больше или равно

**<=** меньше или равно

**==** равно

**!=** не равно

# Вложенные условные операторы

Задача: в переменных **a** и **b** записаны возрасты Андрея и Бориса. Кто из них старше?



Сколько вариантов?

```
if a > b:
    print("Андрей старше")
else:
    if a == b:
        print("Одного возраста")
    else:
        print("Борис старше")
```



Зачем нужен?

вложенный  
условный оператор

# Каскадное ветвление

---

```
if a > b:  
    print("Андрей старше")  
elif a == b:  
    print("Одного возраста")  
else:  
    print("Борис старше")
```



**elif = else if**



# Каскадное ветвление

```
cost = 1500
if cost < 1000:
    print ( "Скидок нет." )
elif cost < 2000:
    print ( "Скидка 2%." )
elif cost < 5000:
    print ( "Скидка 5%." )
else:
    print ( "Скидка 10%." )
```

первое сработавшее  
условие



Что выведет?

Скидка 2%.

## Задачи (без функций **min** и **max**!)

---

**«А»:** Ввести два целых числа, найти наибольшее и наименьшее из них.

**Пример:**

Введите два целых числа :

**1 5**

Наибольшее число 5

Наименьшее число 1

**«В»:** Ввести четыре целых числа, найти наибольшее из них.

**Пример:**

Введите четыре целых числа :

**1 5 4 3**

Наибольшее число 5

# Задачи

---

**«С»:** Ввести последовательно возраст Антона, Бориса и Виктора. Определить, кто из них старше.

**Пример:**

Возраст Антона: 15

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Борис старше всех.

**Пример:**

Возраст Антона: 17

Возраст Бориса: 17

Возраст Виктора: 16

Ответ: Антон и Борис старше Виктора.

## Сложные условия

---

**Задача.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ "подходит" или "не подходит").

**Особенность:** надо проверить, выполняются ли два условия одновременно:

**возраст  $\geq$  25**

**возраст  $\leq$  40**



Можно ли решить известными методами?

# Плохое решение

```
print("Введите ваш возраст")
```

```
x = int(input())
```

```
if x >= 25:
```

```
    if x <= 40:
```

```
        print("Подходит!")
```

```
    else:
```

```
        print("Не подходит.")
```

```
else:
```

```
    print("Не подходит.")
```

вложенный  
условный  
оператор

## Хорошее решение (операция «И»)

---

Задача: набор сотрудников в возрасте **25-40 лет**  
(включительно).

сложное условие

```
if v >= 25 and v <= 40 :  
    print ("подходит")  
else:  
    print ("не подходит")
```

**and** «И»: одновременное выполнение  
всех условий!

# Примеры

---

Задача. Вывести "Да", если число в переменной `a` – двузначное.

```
if 10 <= a and a <= 99:  
    print("Да")
```

Задача. Вывести "Да", если число в переменной `a` – двузначное и делится на 7.

```
if 10 <= a and a <= 99 and (a % 7) == 0:  
    print("Да")
```

## Сложные условия: «ИЛИ»

**Задача.** Самолёт летает по понедельникам и четвергам.  
Ввести номер дня недели и определить, летает ли в этот день самолёт.

**Особенность:** надо проверить, выполняется ли **одно из двух** условий:

день = 1

день = 4

```
if d == 1 or d == 4 :  
    print ("Летает")  
else:  
    print ("Не летает")
```

сложное  
условие

**or** «ИЛИ»: выполнение **хотя бы одного**  
из двух условий!



## Ещё пример

---

**Задача.** Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ "подходит" или "не подходит"). Использовать «ИЛИ».

```
if v < 25 or v > 40 :  
    print ("не подходит")  
else:  
    print ("подходит")
```

## Сложные условия: «НЕ»

---

```
if not (a < b):  
    print("Старт!")
```



Как без «НЕ»?

**not** «НЕ»: если выполняется обратное условие

```
if a >= b:  
    print("Старт!")
```

# Простые и сложные условия

## Простые условия (отношения)

равно

<    <=    >    >=    ==    !=

не равно

**Сложное условие** – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью логических операций:

- **and** – одновременное выполнение условий

`x >= 25 and x <= 40`

- **or** – выполнение хотя бы одного из условий

`x <= 25 or x >= 40`

- **not** – отрицание, обратное условие

`not (x > 25)`    ⇔

`x <= 25`

# Порядок выполнения операций

---

- выражения в скобках
- <, <=, >, >=, =, !=
- not
- and
- or

```
      4      1      6      2      5      3  
if not a > 2 or c != 5 and b < a:  
    ...
```

# Сложные условия

Истинно или ложно при  $a = 2$ ;  $b = 3$ ;  $c = 4$

`not (a > b)`

Да

`a < b and b < c`

Да

`a > c or b > c`

Нет

`a < b and b > c`

Нет

`a > c and b > d`

Нет

`not (a >= b) or c = d`

Да

`a >= b or not (c < b)`

Да

`a > c or b > c or b > a`

Да

# Задачи

---

**«А»:** Напишите программу, которая получает три числа - рост трёх спортсменов, и выводит сообщение «По росту.», если они стоят по возрастанию роста, или сообщение «Не по росту!», если они стоят не по росту.

**Пример:**

Введите рост трёх спортсменов:

**165 170 172**

По росту.

**Пример:**

Введите рост трёх спортсменов:

**175 170 172**

Не по росту!

# Задачи

---

**«В»:** Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

**Пример:**

**Введите номер месяца :**

**5**

**Весна .**

**Пример:**

**Введите номер месяца :**

**15**

**Неверный номер месяца .**

# Задачи

---

**«С»:** Напишите программу, которая получает возраст человека (целое число, не превышающее 120) и выводит этот возраст со словом «год», «года» или «лет». Например, «21 год», «22 года», «25 лет».

**Пример:**

Введите возраст: **18**

Вам 18 лет.

**Пример:**

Введите возраст: **21**

Вам 21 год.

**Пример:**

Введите возраст: **22**

Вам 22 года.



# Логические переменные

```
b = True  
b = False  
type(b)
```

ТОЛЬКО ДВА  
ВОЗМОЖНЫХ  
ЗНАЧЕНИЯ

```
<class 'bool'>
```

логическая (булевская)  
переменная



Джордж Буль

Пример:

```
freeDay = (d==6 or d==7)  
...  
if not freeDay:  
    print("Рабочий день.")  
else:  
    print("Выходной!")
```

# Задачи

---

**«А»:** Напишите программу, которая получает с клавиатуры целое число и записывает в логическую переменную значение «да» (True), если это число трёхзначное. После этого на экран выводится ответ на вопрос: «Верно ли, что было получено трёхзначное число?».

**Пример:**

Введите число: **165**

Ответ: да.

**Пример:**

Введите число: **1651**

Ответ: нет.

# Задачи

---

**«В»:** Напишите программу, которая получает с клавиатуры трёхзначное число и записывает в логическую переменную значение «да» (True), если это число – палиндром, то есть читается одинаково слева направо и справа налево. После этого на экран выводится ответ на вопрос: «Верно ли, что введённое число – палиндром?».

**Пример:**

Введите число: **165**

Ответ: нет.

**Пример:**

Введите число: **656**

Ответ: да.

# Задачи

---

**«С»:** Напишите программу, которая получает с клавиатуры трёхзначное число и записывает в логическую переменную значение «да» (True), если это все его цифры одинаковы. После этого на экран выводится ответ на вопрос: «Верно ли, что все цифры введённого числа одинаковы?»

**Пример:**

Введите число: **161**

Ответ: нет.

**Пример:**

Введите число: **555**

Ответ: да.

# Экспертная система

**Экспертная система** — это компьютерная программа, задача которой — заменить человека-эксперта при принятии решений в сложной ситуации.

**База знаний** = факты + правила вывода:

- если у животного есть перья, то это **птица**;
- если животное кормит детенышей молоком, то это — **млекопитающее**;
- если животное — млекопитающее и ест мясо, то это — **хищник**.

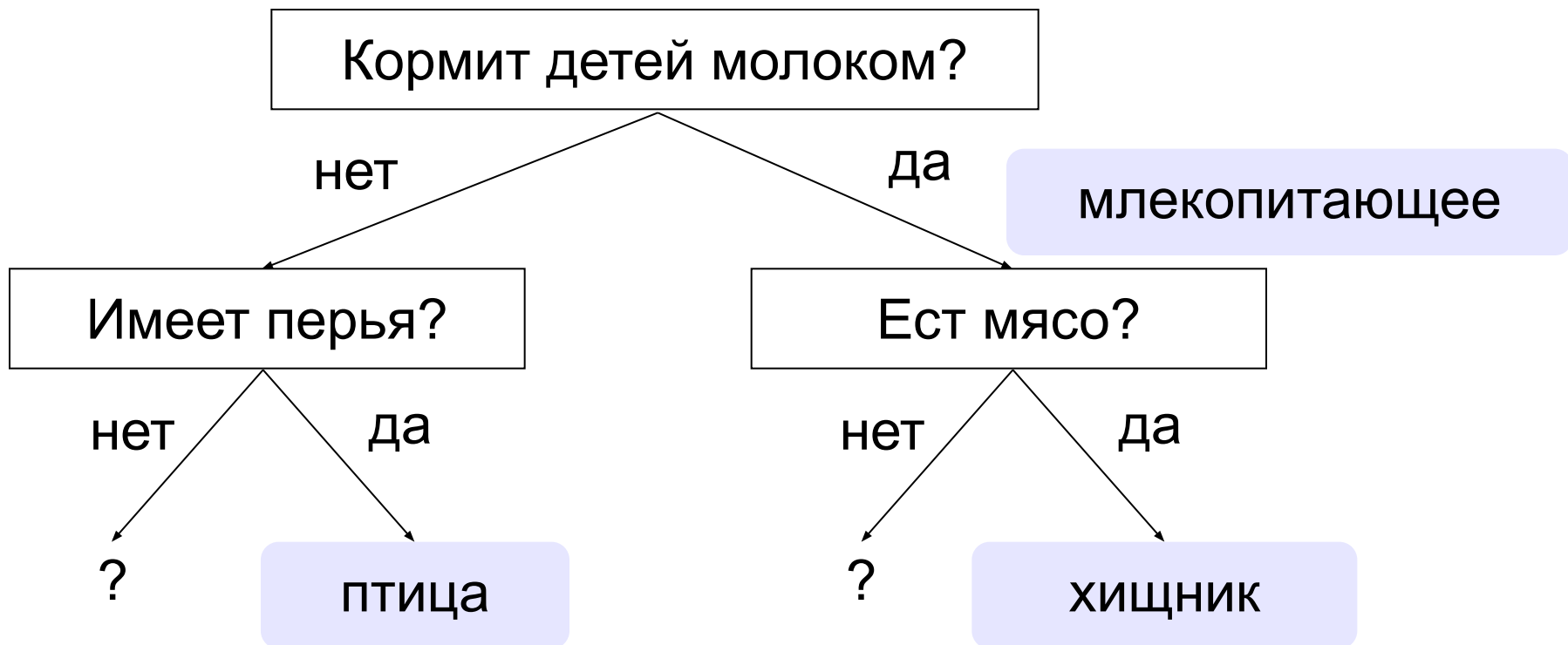
**Диалог:**

Это животное кормит детей молоком? **Нет**

Это животное имеет перья? **Да**

Это **птица**.

# Дерево решений



# Программирование экспертной системы

Ответы пользователя: **да** и **нет** – символьные строки.

```
ans = input("Кормит детей молоком? ")
if ans == "да":
    ... # вариант 1
else:
    ... #
```

```
    # вариант 1
    print("Млекопитающее.")
    ans = input("Ест мясо? ")
    if ans == "да":
        print("Хищник.")
    else:
        print("Не знаю.")
```

# Заглавные и строчные буквы

```
if ans == "да":  
    ...
```

не сработает  
на "Да"



Как исправить?

```
if ans == "да" or ans == "Да":  
    ...
```

Ещё лучше:

```
if ans.lower() == "да":  
    ...
```

преобразовать все  
заглавные в строчные

```
if ans.upper() == "ДА":  
    ...
```



# Программирование (Python)

## § 23. Отладка программ

# Виды ошибок

---

**Синтаксические** ошибки – нарушение правил записи операторов языка программирования.

Обнаруживаются транслятором.

**Логические** ошибки – неверно составленный алгоритм.



**Отказ** (ошибка времени выполнения) – аварийная ситуация во время выполнения программы.

**Отладка** – поиск и исправление ошибок в программе.

# Пример отладки программы

Программа решения квадратного уравнения

$$ax^2 + bx + c = 0$$

```
from math import sqrt
print("Введите a, b, c: ")
a = float(input())
b = float(input())
c = float(input())
D = b*b - 4*a*a
x1 = (-b+sqrt(D))/2*a
x2 = (-b-sqrt(D))/2*a
print("x1=", x1, " x2=", x2, sep=" ")
```

float – преобразовать в вещественное число

# Тестирование

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

`x1=-1.0 x2=-1.0`

Реальность:

`x1=-1.0 x2=-1.0`



Тест 2.  $a = 1, b = -5, c = 6.$

`x1=3.0 x2=2.0`

`x1=4.791 x2=0.209`



Найден вариант, когда программа работает неверно.  
Ошибка **воспроизводится!**

## Возможные причины:

- неверный ввод данных
- неверное вычисление дискриминанта
- неверное вычисление корней
- неверный вывод результатов

$$D = b^2 - 4ac$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

# Отладочная печать

---

Идея: выводить все промежуточные результаты.

```
a = float(input())
```

```
b = float(input())
```

```
c = float(input())
```

```
print(a, b, c)
```

```
D = b*b - 4*a*a
```

```
print("D=", D)
```

```
...
```

# Отладочная печать

Идея: выводить все промежуточные результаты.

Результат:

Введите a, b, c:

1

-5

6

1.0 -5.0 6.0

D= 21.0

$$D = b^2 - 4ac = 25 - 4 \cdot 1 \cdot 6 = 1$$

```
D = b*b - 4*a*c ;
```



Одна ошибка найдена!

# Отладка программы

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

`x1=-1.0 x2=-1.0`

Реальность:

`x1=-1.0 x2=-1.0`



Тест 2.  $a = 1, b = -5, c = 6.$

`x1=3.0 x2=2.0`

`x1=3.0 x2=2.0`



Программа работает верно?

Тест 3.  $a = 8, b = -6, c = 1.$

`x1=0.5 x2=0.25`

`x1=32.0 x2=16.0`



`x1 = (-b+sqrt(D)) / (2*a)`

`x2 = (-b-sqrt(D)) / (2*a)`



Что неверно?

# Задачи

---

«А»: Загрузите программу, которая должна вычислять сумму цифр трёхзначного числа:

```
N = input(int("N = "))
```

```
d0 = N % 10
```

```
d1 = N % 100
```

```
d2 = N // 100
```

```
d0 + d2 = s
```

```
print(s)
```

Выполните отладку программы:

- исправьте синтаксические ошибки
- определите ситуации, когда она работает неверно
- исправьте логические ошибки.



# Задачи

---

«В»: Доработайте программу из п. А так, чтобы она правильно работала с отрицательными трёхзначными числами: при вводе числа «−123» программа должна выдавать ответ 6.

# Задачи

---

«С»: Загрузите программу, которая должна вычислять наибольшее из трёх чисел:

```
a = input("a = ")
b = int("b = ")
c = input("c = ")
if a > b:    M = a
else       M = b
if c > b    M = b
else:      M = c
input(M)
```

Выполните отладку программы:

- исправьте синтаксические ошибки
- определите ситуации, когда она работает неверно
- исправьте логические ошибки.

# Программирование (Python)

## § 20. Программирование циклических алгоритмов

# Зачем нужен цикл?

Задача. Вывести 5 раз «Привет!».

```
print ("Привет")  
print ("Привет")  
print ("Привет")  
print ("Привет")  
print ("Привет")
```



А если 5000?

Цикл «N раз»:

```
сделай 5 раз  
print ("Привет")
```

такого оператора нет  
в Python!

# Как работает цикл?

**!** Нужно запоминать, сколько раз цикл уже выполнен!

переменная-счётчик

```
счётчик = 0
пока счётчик < 5
    print("Привет")
    счётчик += 1
```

ещё не делали

сделали ещё раз

```
c = 0
while c < 5:
    print("Привет")
    c += 1
```

## Ещё один вариант

Идея: запоминать, сколько шагов осталось.

```
счётчик = 5
пока счётчик > 0
    print("Привет")
    счётчик -= 1
```

```
с = 5
while с > 0:
    print("Привет")
    с -= 1
```



Как записать иначе  
последнюю строку?

## Цикл с предусловием

---

- условие проверяется при входе в цикл
- как только условие становится ложным, работа цикла заканчивается
- если условие ложно в самом начале, цикл не выполняется **ни разу**

```
while условие:
```

```
    ...
```

тело цикла



Если условие никогда не станет ложно?

```
while True:
```

```
    ...
```

бесконечный цикл  
(зацикливание)

# Сколько раз выполняется цикл?

```
a = 4; b = 6  
while a < b: a += 1
```

2 раза  
a = 6

```
a = 4; b = 6  
while a < b: a += b
```

1 раз  
a = 10

```
a = 4; b = 6  
while a > b: a += 1
```

0 раз  
a = 4

```
a = 4; b = 6  
while a < b: b = a - b
```

1 раз  
b = -2

```
a = 4; b = 6  
while a < b: a -= 1
```

**защелкивание**



# Сумма цифр числа

---

*Задача.* Вычислить сумму цифр введённого числа.

$$123 \rightarrow 1 + 2 + 3 = 6$$

Выделить последнюю цифру числа в переменной  $N$ :

$$d = N \% 10$$

$$123 \rightarrow 3$$

Отбросить последнюю цифру числа в переменной  $N$ :

$$N = N // 10$$

$$123 \rightarrow 12$$

Добавить к переменной  $sum$  значение переменной  $d$ :

$$sum += d$$

$$sum = 6 \rightarrow 6 + 4 = 10$$

$$d = 4$$

# Сумма цифр числа

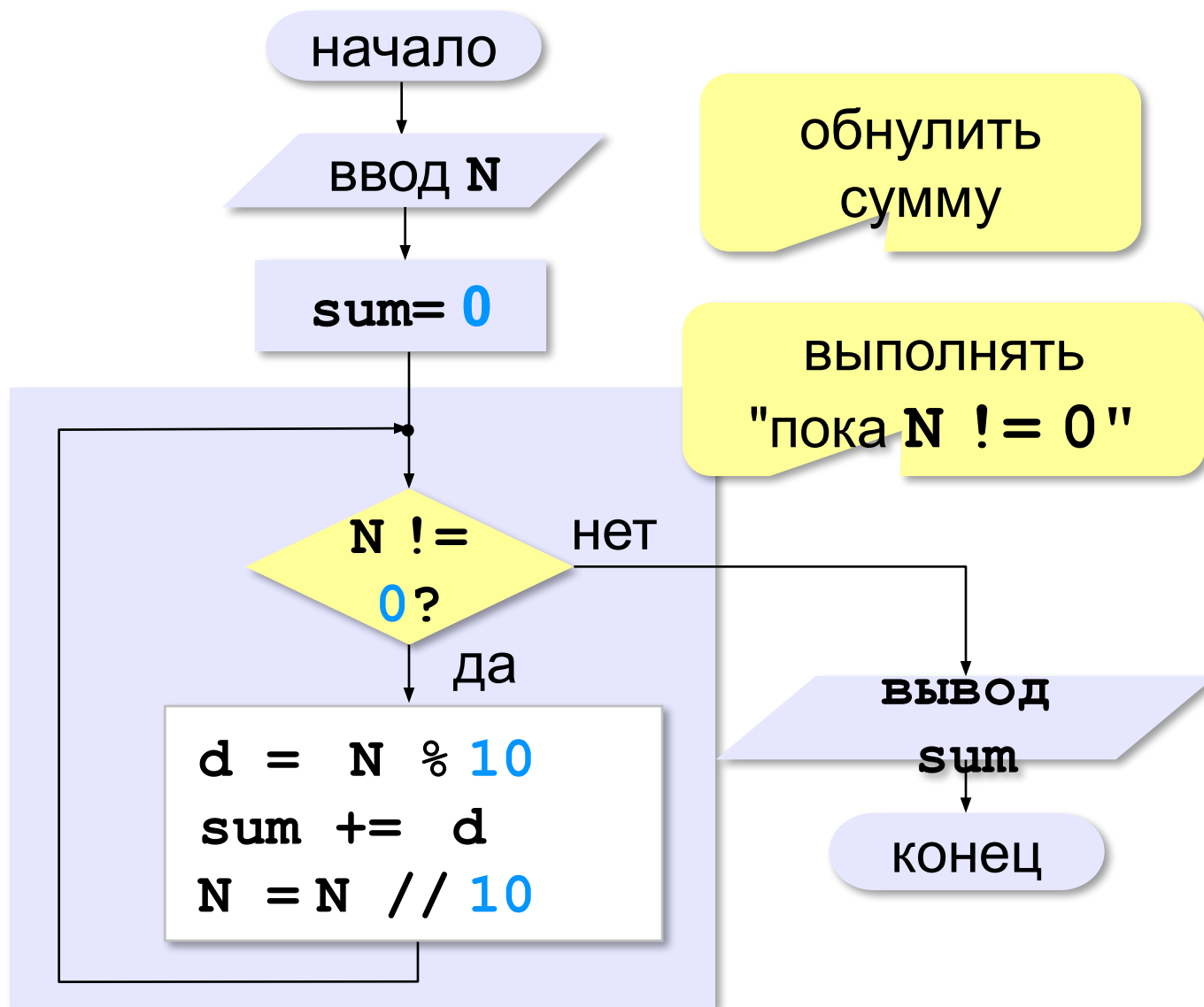
---

- выделяем последнюю цифру числа (%)
- увеличиваем сумму на значение цифры (`sum+=d`)
- отсекаем последнюю цифру числа (//)

N	d	sum
123		0

начальные значения

# Сумма цифр числа



# Сумма цифр числа

```
N = int(input("Введите целое число")); N1=N
sum = 0
while N != 0:
    d = N % 10
    sum += d
    N = N // 10
print("Сумма цифр числа", N1, " равна", sum)
```



Что плохо?

# Задачи

---

**«А»:** Напишите программу, которая получает с клавиатуры количество повторений и выводит столько же раз какое-нибудь сообщение.

**Пример:**

Сколько раз повторить? **3**

Привет!

Привет!

Привет!

**«В»:** Напишите программу, которая получает с клавиатуры натуральное число и определяет, сколько раз в его десятичной записи встречается цифра 1.

**Пример:**

Введите число? **311**

Единиц: **2**

# Задачи

---

**«С»:** Напишите программу, которая получает с клавиатуры натуральное число и находит наибольшую цифру в его десятичной записи.

**Пример:**

Введите число: **311**

Наибольшая цифра: **3**

**«D»:** Напишите программу, которая получает с клавиатуры натуральное число и определяет, есть ли в его десятичной записи одинаковые цифры, стоящие рядом.

**Пример:**

Введите число: **553**

Ответ: **да.**

Введите число: **535**

Ответ: **нет.**

# Алгоритм Евклида

**Задача.** Найти наибольший общий делитель (НОД) двух натуральных чисел.

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

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(a-b, b) \\ &= \text{НОД}(a, b-a)\end{aligned}$$



Евклид  
(365-300 до. н. э.)

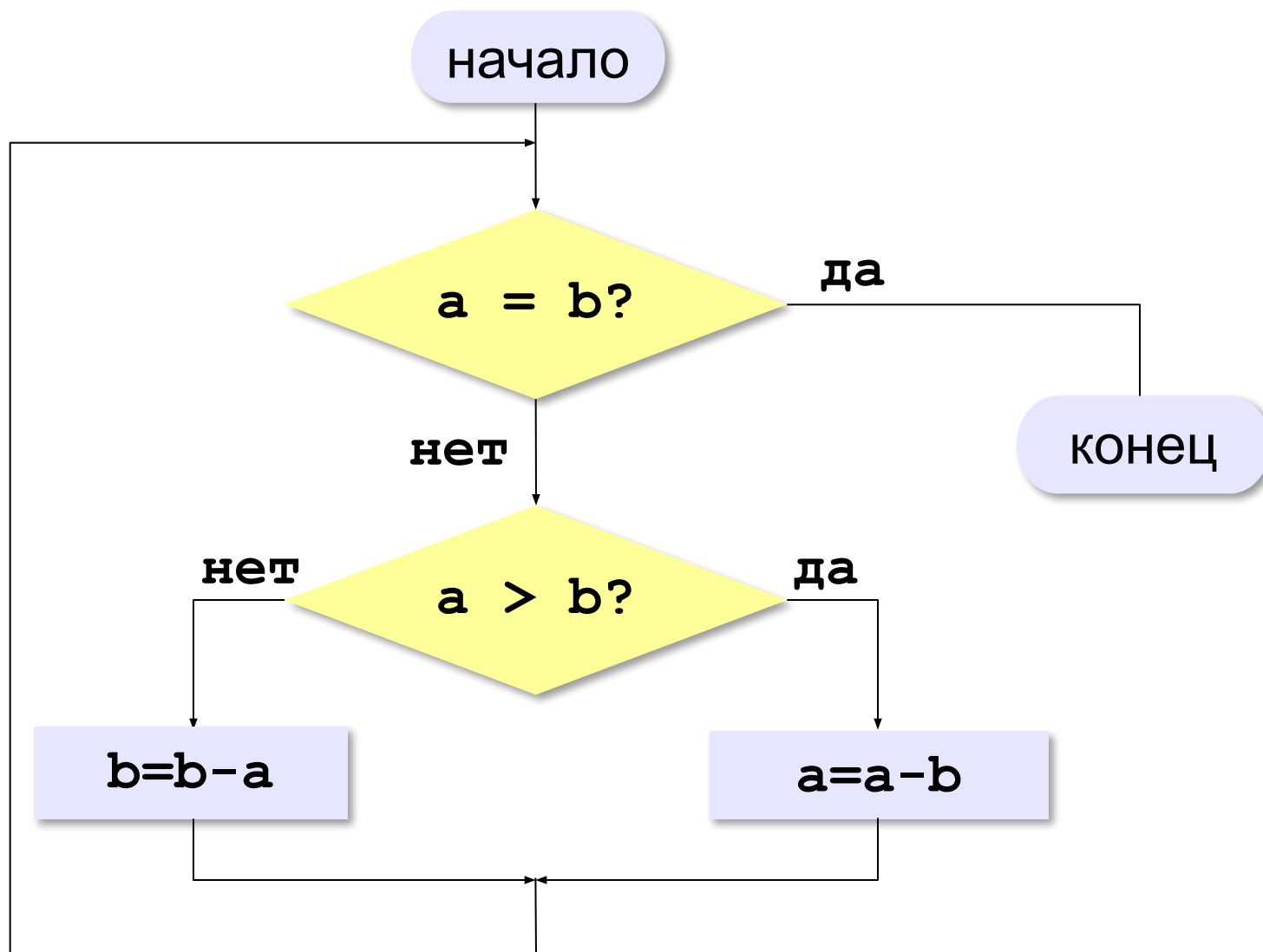
## Пример:

$$\begin{aligned}\text{НОД}(14, 21) &= \text{НОД}(14, 21-14) = \text{НОД}(14, 7) \\ &= \text{НОД}(7, 7) = 7\end{aligned}$$

⊖ много шагов при большой разнице чисел:

$$\text{НОД}(1998, 2) = \text{НОД}(1996, 2) = \dots = 2$$

# Алгоритм Евклида





# Алгоритм Евклида

```
while a != b:  
    if a > b:  
        a -= b  
    else:  
        b -= a
```

как заменить?

? Где будет НОД? Как его вывести?

? Как вывести НОД в формате НОД(14,21) = 7?

? А без дополнительных переменных?

# Модифицированный алгоритм Евклида

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

$$\begin{aligned}\text{НОД}(a, b) &= \text{НОД}(a \% b, b) \\ &= \text{НОД}(a, b \% a)\end{aligned}$$

**Пример:**

$$\text{НОД}(14, 21) = \text{НОД}(14, 7) = \text{НОД}(0, 7) = 7$$

# Модифицированный алгоритм

```
while a != 0 and b != 0:  
    if a > b:  
        a = a % b  
    else:  
        b = b % a
```



Где будет НОД? Как его вывести?

```
if a != 0:  
    print(a)  
else:  
    print(b)
```



```
print( a+b )
```

# В стиле Python

```
while b!=0:  
    a, b = b, a % b  
print(a)
```



Почему работает?

заменить **a** на **b** и **b** на  
(**a % b**)

a	b
21	14

a	b
14	21



**a > b!**

# В других языках программирования

---

Паскаль:

```
while (a<>0) and
      (b<>0) do
  if a>b then
    a := a mod b
  else
    b := b mod a;
```

C:

```
while (a!=0 && b!=0)
{
  if (a > b)
    a = a % b;
  else
    b = b % a;
}
```

# Задачи

---

«А»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью алгоритма Евклида.

**Пример:**

Введите два числа:

21 14

НОД (21 , 14) = 7

«В»: Ввести с клавиатуры два натуральных числа и найти их НОД с помощью **модифицированного** алгоритма Евклида. Заполните таблицу:

a	64168	358853	6365133	17905514	549868978
b	82678	691042	11494962	23108855	298294835
НОД (a , b)					

# Задачи

---

**«С»:** Ввести с клавиатуры два натуральных числа и сравнить количество шагов цикла для вычисления их НОД с помощью обычного и модифицированного алгоритмов Евклида.

## Пример:

Введите два числа:

**1998 2**

НОД(1998, 2) = 2

Обычный алгоритм: 998

Модифицированный: 1

# Обработка потока данных

---

**Задача.** На вход программы поступает поток данных — последовательность целых чисел, которая **заканчивается нулём**. Требуется найти сумму элементов этой последовательности.

```
while x != 0:  
    # добавить x к сумме  
    # x = следующее число
```



Откуда возьмётся **x** в первый раз?



# Обработка потока данных

```
Sum = 0
x = int(input()) # первое число
while x != 0:
    Sum += x
    x = int(input()) # ввести следующее
print("Сумма ", Sum)
```



Как найти количество чисел?



Как найти сумму положительных?

# Задачи

---

- «А»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.
- «С»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.

## Задачи на циклы

---

- «А»: Напишите программу, которая предлагает ввести число-пароль и не переходит к выполнению основной части, пока не введён правильный пароль. Основная часть – вывод на экран «секретных сведений».
- «В»: Напишите программу, которая получает с клавиатуры натуральное число, которое больше 1, и определяет, простое оно или нет. Для этого нужно делить число на все натуральные числа, начиная с 2, пока не получится деление без остатка.
- «С»: Напишите программу, которая получает с клавиатуры два целых числа и вычисляет их произведение, используя только операции сложения.

# Задачи

---

«D»: Напишите программу, которая получает с клавиатуры натуральное число и вычисляет целый квадратный корень из него – наибольшее число, квадрат которого не больше данного числа.

# Цикл по переменной

Задача. Вывести на экран степени числа 2 от  $2^0$  до  $2^{10}$ .

```
k = 0
```

```
N = 2
```

```
while k <= 10 :
```

```
    print(N)
```

```
    N = N*2
```

```
    k = k + 1
```



Работа с `k` в трёх местах!

Идея: собрать всё вместе.

с нуля!

не включая 11!

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
N = 2
```

```
for k in range(11) :
```

```
    print(N)
```

```
    N = N*2
```

сделать 11 раз

# Цикл по переменной

```
for k in range(11):  
    print(k)
```



Что выведет?

0

1

2

...

10

```
for k in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:  
    print(k)
```

Начать на с 0, а с 1:

```
for k in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:  
    print(k)
```

начальное  
значение

```
for k in range(1, 11):  
    print(k)
```

## Цикл по переменной

---

Задача. Найти сумму чисел от 1 до 1000.

```
S = 0
for i in range(1, 1001):
    S += i
```

Задача. Вывести квадраты чисел от 10 до 1 по убыванию.

```
for k in [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]:
    print(k*k)
```



не включая 0

шаг

```
for k in range(10, 0, -1):
    print(k*k)
```

# Цикл по переменной

Задача. Найти сумму чётных чисел от 2 до 1000.

```
sum = 0
for i in range(2, 1001):
    if i % 2 == 0:
        sum += i
```



Что плохо?



```
sum = 0
for i in range(2, 1001, 2):
    sum += i
```

шаг



# В других языках программирования

## Паскаль:

```
sum := 0;  
for i=1 to 1000 do  
    sum := sum + i;
```

шаг только 1 или  
-1 (downto)

## C:

```
int sum, i;  
sum = 0;  
for (i=1; i<=1000; i++)  
    sum += i;
```

sum=sum+i;

i=i+1;

# Задачи

---

«А»: Напишите программу, которая находит количество четырёхзначных чисел, которые делятся на 7.

«В»: Ипполит задумал трёхзначное число, которое при делении на 15 даёт в остатке 11, а при делении на 11 даёт в остатке 9. Напишите программу, которая находит все такие числа.

«С»: С клавиатуры вводится натуральное число  $N$ . Программа должна найти факториал этого числа (обозначается как  $N!$ ) – произведение всех натуральных чисел от 1 до  $N$ . Например,

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120.$$

# Задачи

---

«D»: Натуральное число называется **числом Армстронга**, если сумма цифр числа, возведенных в  $N$ -ную степень (где  $N$  – количество цифр в числе) равна самому числу. Например,  $153 = 1^3 + 5^3 + 3^3$ .  
Найдите все трёхзначные Армстронга.

# Программирование (Python)

## § 21. Массивы

# Что такое массив?

---



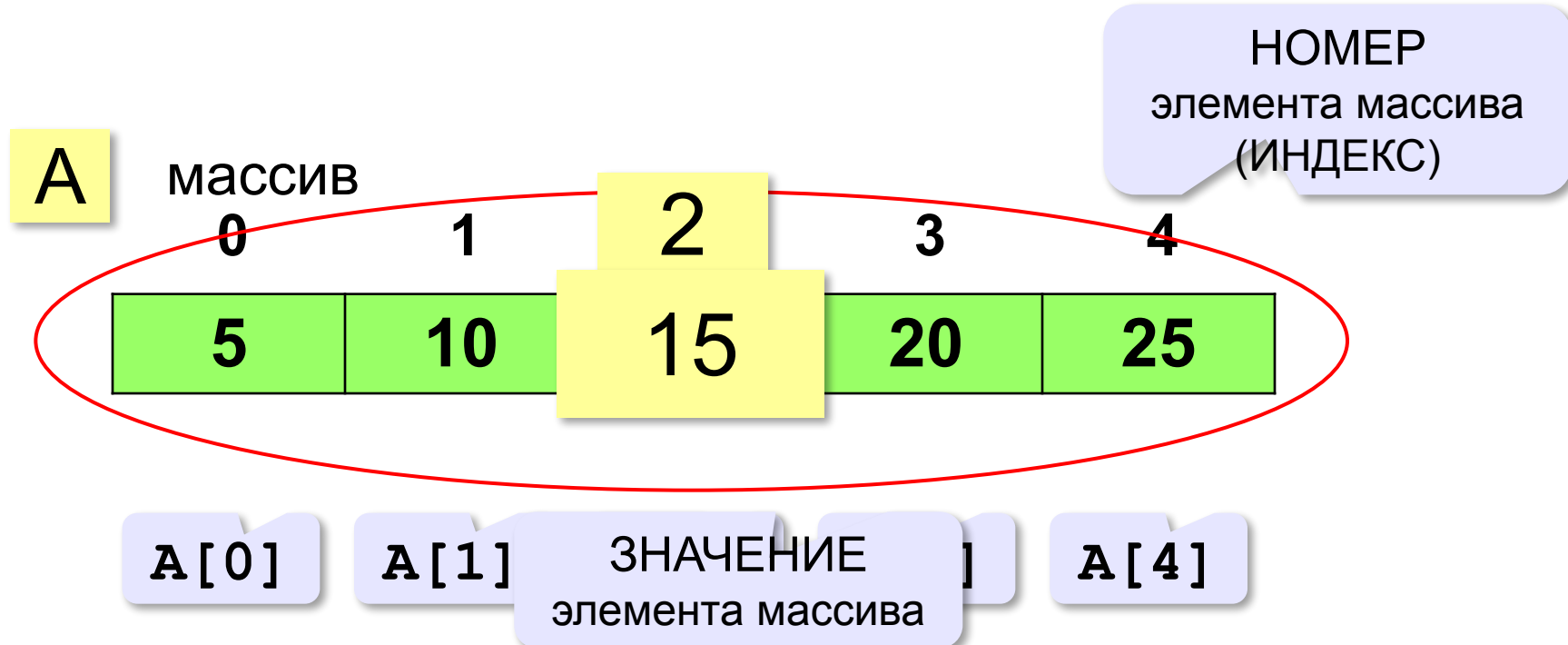
Как ввести 10000 переменных?

**Массив** – это группа переменных одного типа, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя.

**Надо:**

- выделять память
- записывать данные в нужную ячейку
- читать данные из ячейки

# Обращение к элементу массива



**Индекс элемента** — это значение, которое указывает на конкретный элемент массива.

**!** Нумерация с нуля!

# Создание массива

---

11	22	35	41	53
----	----	----	----	----

```
A = [11, 22, 35, 41, 53]
```

```
A = [11, 22] + [35, 41] + [53]
```

```
A = [11] * 5
```

```
A = [11] + [11] + [11] + [11] + [11]
```

11	11	11	11	11
----	----	----	----	----

# Обращение к элементу массива

**A[2]**

ИНДЕКС элемента массива: 2

ЗНАЧЕНИЕ элемента массива

0	1	2	3	4
23	12	7	43	51

```
i = 1
A[2] = A[i] + 2*A[i-1] + A[2*i+1]
print( A[2]+A[4] )
```

**?** Что получится?

```
A[2] = A[1] + 2*A[0] + A[3]    101
print( A[2]+A[4] )           152
```



# Что неверно?

0	1	2	3	4
1	2	3	4	5

```
A = [1, 2, 3, 4, 5]
x = 1
print( A[x-8] )
A[x+4] = A[x-1] + A[2*x]
```



Что плохо?



```
print( A[-7] )
A[5] = A[0] + A[2]
```

**Выход за границы массива** — это обращение к элементу с индексом, который не существует в массиве.

# Перебор элементов массива

**Перебор элементов:** просматриваем все элементы массива и, если нужно, выполняем с каждым из них некоторую операцию.

```
N = 10
```

```
A = [0]*N # память уже выделена
```

```
0, 1, 2, 3, ..., N-1
```

```
for i in range(N):
```

```
    # здесь работаем с A[i]
```

# Заполнение массива

[0, 1, 2, 3, ..., N-1]

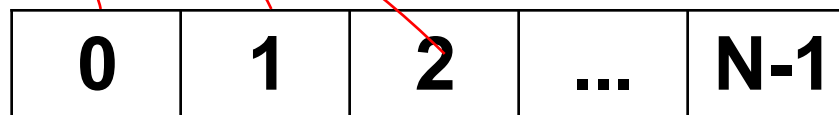
```
for i in range(N):  
    A[i] = i
```



Что произойдёт?

В развёрнутом виде

```
A[0] = 0  
A[1] = 1  
A[2] = 2  
...  
A[N-1] = N-1
```



В стиле Python:

```
A = [ i for i in range(N) ]
```

# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

```
A[0] = N
A[1] = N-1
A[2] = N-2
...
A[N-1] = 1
```

```
X = N
for i in range(N):
    A[i] = X
    X = X - 1
```

**?** Как меняется X?

X = N, N-1, ..., 2, 1

начальное  
значение

уменьшение  
на 1

# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

$$A[i] = X$$

**?** Как связаны  $i$  и  $X$ ?

$i$	$X$
0	N
1	N-1
2	N-2
...	...
N-1	1

+1      -1

```
for i in range(N):
    A[i] = N - i
```

В стиле Python:

```
A = [ N-i
      for i in range(N) ]
```

**!** Сумма  $i$  и  $X$  не меняется!

$$i + X = N$$

$$X = N - i$$

# Вывод массива на экран

Весь массив сразу:

```
print( A )
```

```
[1, 2, 3, 4, 5]
```

По одному элементу:

```
for i in range(N):  
    print( A[i] )
```

В столбик

или так:

```
for x in A:  
    print( x )
```

для всех элементов в массиве A



Как вывести  
в строчку?

```
for x in A:  
    print( x, end=" " )
```

пробел между  
элементами

# Вывод массива на экран (Python)

---

[1, 2, 3, 4, 5]

```
print ( *A )
```



```
print ( 1 , 2 , 3 , 4 , 5 )
```

разбить список  
на элементы

```
1 2 3 4 5
```

# Ввод с клавиатуры

```
for i in range(N):  
    A[i] = int(input())
```



Что плохо?

или так:

```
A = [int(input())  
      for i in range(N)]
```

С подсказкой для ввода:

```
for i in range(N):  
    s = "A[" + str(i) + "]= "  
    A[i] = int(input(s))
```

```
A[0] = 5  
A[1] = 12  
A[2] = 34  
A[3] = 56  
A[4] = 13
```



# Ввод с клавиатуры (Python)

---

Ввод всех чисел в одной строке:

```
data = input()      # "1 2 3 4 5"  
s = data.split()   # ["1", "2", "3", "4", "5"]  
A = [ int(x) for x in s ]  
                    # [1, 2, 3, 4, 5]
```

или так:

```
A = [int(x) for x in input().split()]
```

# В других языках программирования

---

## Паскаль:

объявление массива

```
const N = 10;  
var A: array[0..N-1] of integer;  
...  
for i:=0 to N-1 do  
    A[i] = i;  
for i:=0 to N-1 do  
    write(A[i], ' ');
```

# В других языках программирования

C++:

```
int A[N], i;  
for (i = 0; i < N; i++)  
    A[i] = i;  
for (i = 0; i < N; i++)  
    cout << A[i] << " ";
```



Нумерация элементов  
всегда с нуля!

# Задачи

---

- «А»: а) Заполните все элементы массива из 10 элементов значением  $X$ , введённым с клавиатуры.
- б) Заполните массив из 10 элементов последовательными натуральными числами, начиная с  $X$  (значение  $X$  введите с клавиатуры).
- «В»: а) Заполните массив из 10 элементов натуральными числами в обратном порядке, начиная со значения  $X$ , введённого с клавиатуры. Последний элемент должен быть равен  $X$ , предпоследний равен  $X-1$  и т. д.
- б) Заполните массив из 10 элементов степенями числа 2 (от  $2^1$  до  $2^N$ ), так чтобы элемент с индексом  $i$  был равен  $2^i$ .

## Задачи

---

- «С»: а) Заполните массив из 10 элементов степенями числа 2, начиная с конца, так чтобы последний элемент массива был равен 1, а каждый предыдущий был в 2 раза больше следующего.
- б) С клавиатуры вводится целое число  $X$ . Заполните массив из 11 элементов целыми числами, так чтобы средний элемент массива был равен  $X$ , слева от него элементы стояли по возрастанию, а справа – по убыванию. Соседние элементы отличаются на единицу. Например, при  $X = 3$  массив из 5 элементов заполняется так: 1 2 3 2 1.

# Заполнение случайными числами

из библиотеки  
(модуля) random

взять функцию randint

```
from random import randint
N = 10          # размер массива
A = [0]*N      # выделить память
for i in range(N):
    A[i] = randint(20, 100)
```

В краткой форме:

```
from random import randint
N = 10
A = [ randint(20, 100)
      for i in range(N) ]
```

# Обработка элементов массива

```
N = 10
```

```
A = [0]*N # память уже выделена
```

```
for i in range(N):  
    # здесь работаем с A[i]
```

Вывести на экран в столбик:

```
for i in range(N):  
    print( A[i] )
```



Что вместо «???»?

Вывести на экран в строчку:

```
for i in range(N):  
    print( A[i], end = " " )
```

```
print( *A )
```

# Обработка элементов массива

---

Вывести числа, на 1 большие, чем  $A[i]$ :

```
for i in range(N):  
    print( A[i]+1 )
```



Что вместо «**???**»?

Вывести последние цифры:

```
for i in range(N):  
    print( A[i]%10 )
```



# Обработка элементов массива

---

Заполнить нулями:

```
for i in range(N) :  
    A[i] = 0
```



Что вместо «???»?

Увеличить на 1:

```
for i in range(N) :  
    A[i] += 1
```

Умножить на 2:

```
for i in range(N) :  
    A[i] *= 2
```

## Задачи-2

---

**«А»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[0, 10]$ , выводит его на экран, а затем выводит на экран квадраты всех элементов массива.

**Пример:**

**Массив:** 5 6 2 3 1 4 8 7

**Квадраты:** 25 36 4 9 1 16 64 49

**«В»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[100, 300]$  и выводит его на экран. После этого на экран выводятся средние цифры (число десятков) всех чисел, записанных в массив.

**Пример:**

**Массив:** 142 324 135 257 167 295 126 223 138 270

**Число десятков:** 4 2 3 5 6 9 2 2 3 7

## Задачи-2

---

**«С»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[100, 500]$  и выводит его на экран. После этого на экран выводятся суммы цифр всех чисел, записанных в массив.

**Пример:**

**Массив:** 162 425 340 128 278 195 326 414 312 177

**Суммы цифр:** 9 11 7 11 17 15 11 9 6 15

# Программирование (Python)

## § 21. Алгоритмы обработки массивов

# Сумма элементов массива

Задача. Найти сумму элементов массива из N элементов.

**?** Какие переменные нужны?

5	2	8	3	1
---	---	---	---	---

i	Sum
	0
0	5
1	7
2	15
3	18
4	19

```
Sum = 0
for i in range(N):
    Sum += A[i]
print( Sum )
```

В стиле Python:

```
print( sum(A) )
```

# Сумма элементов массива (Python)

Задача. Найти сумму элементов массива A.

```
Sum = 0
for x in A:
    Sum += x
print ( Sum )
```

для всех  
элементов из A



Не нужно знать размер!

или так:


```
print ( sum (A) )
```

встроенная  
функция

# Сумма не всех элементов массива

---

Задача. Найти сумму чётных элементов массива.

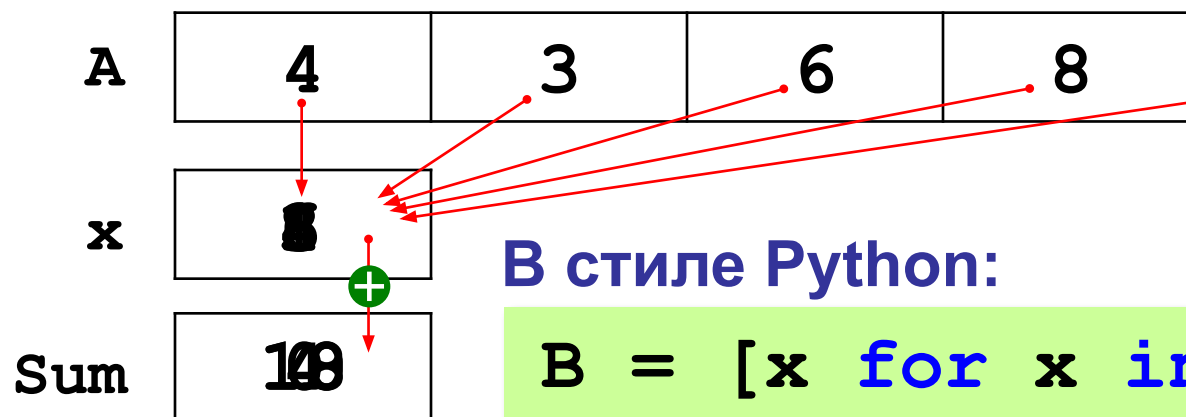
 Что делаем с нечётными?

```
Sum = 0
for i in range(N):
    if A[i] % 2 == 0:
        Sum += A[i]
print( Sum )
```

# Сумма не всех элементов массива

Задача. Найти сумму чётных элементов массива.

```
Sum = 0
for x in A:
    if x % 2 == 0:
        Sum += x
print( Sum )
```



отбираем в новый массив все нужные значения

В стиле Python:

```
B = [x for x in A
      if x % 2 == 0]
print( sum(B) )
```



# Задачи

---

- «А»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-5; 5]$  и находит сумму положительных элементов.
- «В»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-2; 2]$  и находит произведение ненулевых элементов.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[100; 1000]$  и находит отдельно сумму элементов в первой и во второй половинах массива.

# Подсчёт элементов по условию

Задача. Найти количество чётных элементов массива.

**?** Какие переменные нужны?

переменная-  
счётчик

```
count = 0
for i in range(N):
    if A[i] % 2 == 0:
        count += 1
print(count)
```

**?** Что тут делаем?

# Подсчёт элементов по условию (Python)

Задача. Найти количество чётных элементов массива.

```
count = 0
for x in A:
    if x % 2 == 0:
        count += 1
print( count )
```

В стиле Python:

```
B = [x for x in A
      if x % 2 == 0]
print( len(B) )
```

размер массива

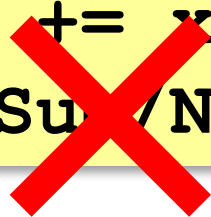
# Среднее арифметическое

Задача. Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

```
Sum = 0
for x in A:
    if x > 180:
        Sum += x
print( Sum / N )
```



Что плохо?




# Среднее арифметическое

Задача. Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

 Какие переменные нужны?

```
Sum = 0
count = 0
for x in A:
    if x > 180:
        count += 1
        Sum += x
print( Sum/count )
```

 Что тут делаем?

## Среднее арифметическое (Python)

---

*Задача.* Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

```
B = [ x for x in A
      if x > 180 ]
print ( sum(B) / len(B) )
```

отбираем нужные

# Задачи

---

- «А»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [0; 200] и считает число элементов, которые делятся на 10.
- «В»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [0; 200] и считает число двузначных чисел в массиве.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [10; 100] и считает число пар соседних элементов, сумма которых делится на 3.

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено положительных чисел.

- 1) нужен счётчик
- 2) счётчик увеличивается
- 3) нужен цикл
- 4) это цикл с условием (число шагов неизвестно)

 ?

Когда увеличивать счётчик?

 ?

Какой цикл?

**счётчик = 0**

**пока не введён 0:**

**если введено число > 0 то**

**счётчик := счётчик + 1**



# Обработка потока данных

```
count = 0
x = int(input())
while x != 0:
    if x > 0:
        count += 1
    x = int(input())
print( count )
```

откуда взять x?



Что плохо?

# Найди ошибку!

---

```
count = 0
x = int(input())
while x != 0:
    if x > 0:
        count += 1
pr x = int(input())
```

# Найди ошибку!

---

```
count = 0
while x == 0:
    if x > !=:
        count += 1
    x = int(input())
print( count )
```

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

- 1) нужна переменная для суммы
- 2) число добавляется к сумме, если оно заканчивается на "5"
- 3) нужен цикл с условием

```
сумма = 0
```

```
пока не введён 0:
```

```
    если число оканчивается на "5" то
```

```
        сумма := сумма + число
```



Как это записать?

```
if x % 10 == 5:
```

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

```
sum = 0
x = int(input())
while x != 0:
    if x % 10 == 5:
        sum += x
    x = int(input())
print( sum )
```



Чего не хватает?

# Найди ошибку!

---

```
sum = 0
x = int(input())
    if x % 10 == 5:
        sum += x
    x = int(input())
print( sum )
```

# Задачи

---

- «А»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.

# Задачи

---

- «С»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти среднее арифметическое всех двузначных чисел, которые делятся на 7.
- «D»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.



# Перестановка элементов массива

**?** Как поменять местами значения двух переменных  $a$  и  $b$ ?

вспомогательная  
переменная

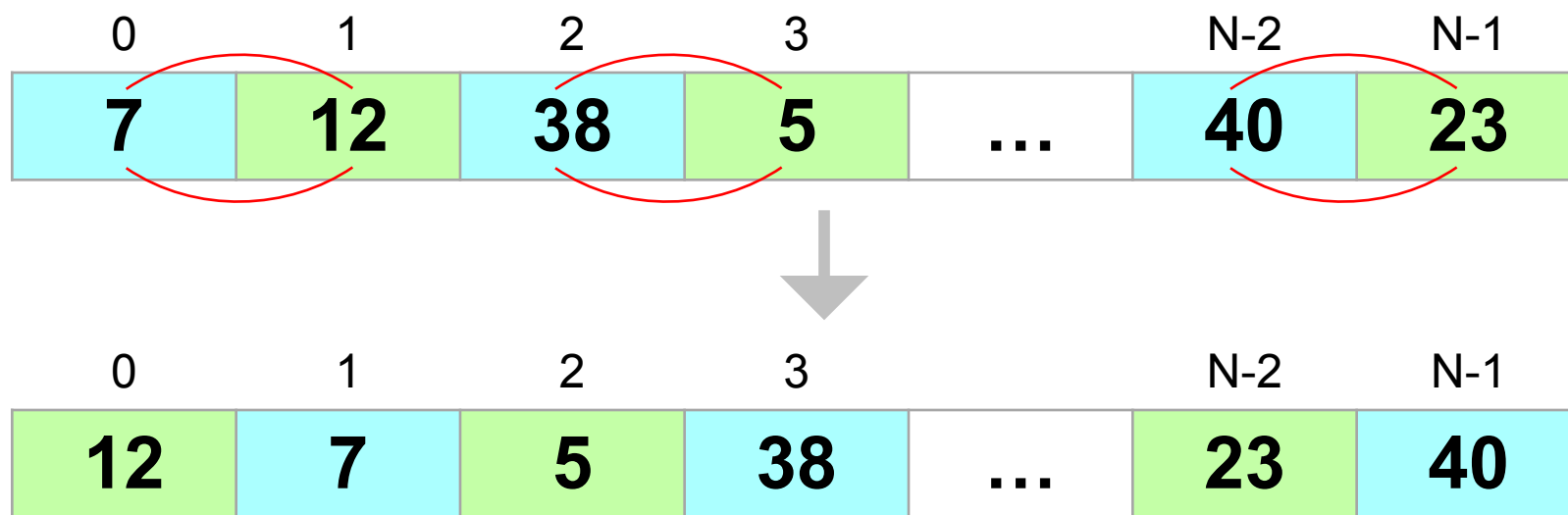
```
c = a
a = b
b = c
```

элементы массива:

```
c = A[i]
A[i] = A[k]
A[k] = c
```

## Перестановка пар соседних элементов

**Задача.** Массив  $A$  содержит чётное количество элементов  $N$ . Нужно поменять местами пары соседних элементов: 0-й с 1-м, 2-й — с 3-м и т. д.



# Перестановка пар соседних элементов

```
for i in range(N) :
```

```
    поменять местами A[i] и A[i+1]
```



Что плохо?

0	1	2	3	4	5
7	12	38	5	40	23
12	7	38	5	40	23
12	38	7	5	40	
12	38	5	7	40	23
12	38	5	40	7	23
12	38	5	40	23	7

выход за границы массива



## Перестановка пар соседних элементов

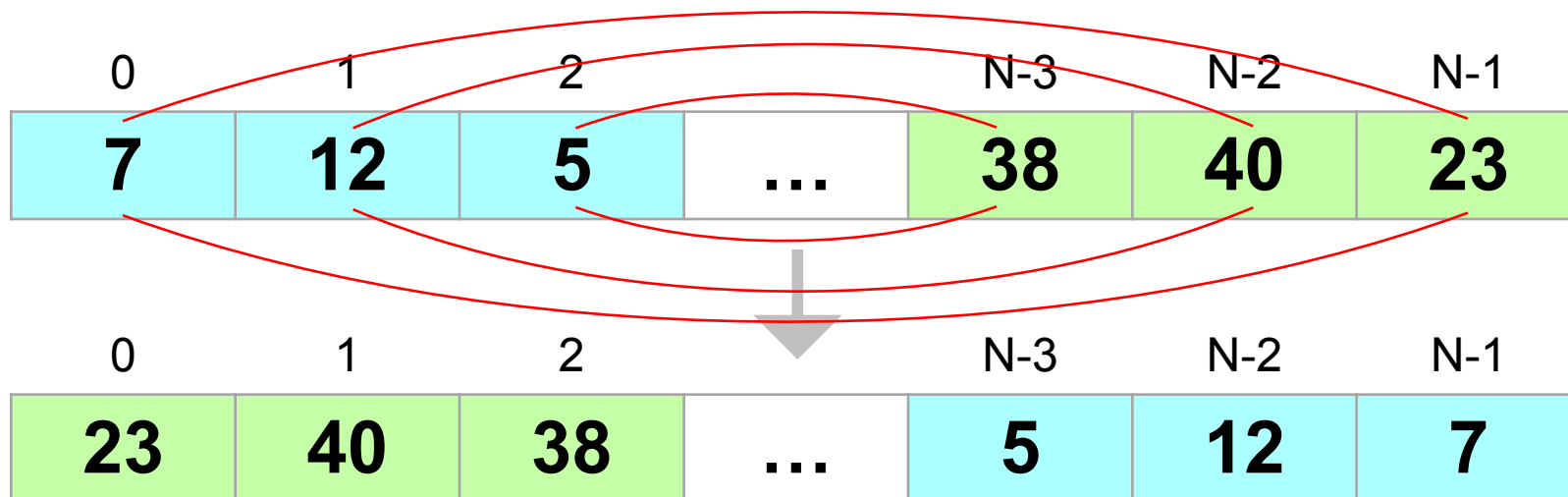
---

```
for i in range(0, N, 2):  
    # переставляем A[i] и A[i+1]  
    c = A[i]  
    A[i] = A[i+1]  
    A[i+1] = c
```

$A[1] \leftrightarrow A[2], A[3] \leftrightarrow A[4], \dots, A[N-1] \leftrightarrow A[N]$

# Реверс массива

Задача. Переставить элементы массива в обратном порядке (выполнить *реверс*).



$$A[0] \leftrightarrow A[N-1]$$

$$0 + N - 1 = N - 1$$

$$A[1] \leftrightarrow A[N-2]$$

$$1 + N - 2 = N - 1$$

$$A[i] \leftrightarrow A[N-1-i]$$

$$i + ??? = N - 1$$

$$A[N-1] \leftrightarrow A[0]$$

$$N - 1 + 0 = N - 1$$

# Реверс массива

```
for i in range(N % 2):
```

```
    поменять местами A[i] и A[N+1-i]
```

0	1	2	3
7	12	40	23
23	12	40	7
23	40	12	7
23	12	40	7
7	12	40	23

$i=0$

$i=1$

$i=2$

$i=3$



Что плохо?



Как исправить?

# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)

**ЕРЕМИН Евгений Александрович**

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

[eremin@pspu.ac.ru](mailto:eremin@pspu.ac.ru)

# Источники иллюстраций

---

1. иллюстрации художников издательства «Бином»
2. авторские материалы