Программирование (Python)

Программирование (Python)

§ 19. Символьные строки

Что такое символьная строка?

Символьная строка – это последовательность символов.

- строка – единый объект
- длина строки может меняться во время работы программы

Символьные строки

Присваивание:

```
s = "Вася пошёл гулять"
```

Ввод с клавиатуры:

```
s = input()
```

Вывод на экран:

```
print(s)
```

Длина строки:

```
n = len(s)
```

length – длина

Сравнение строк

```
print("Введите пароль: ")
s = input()
if s == "sEzAm":
  print("Слушаюсь и повинуюсь!")
else:
  print("Пароль неправильный")
```

? Какой правильный пароль?

? Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

Сравнение строк

```
s1 = "паровоз"
s2 = "пароход"
if s1 < s2:
  print(s1, "<", s2)
elif s1 == s2:
  print(s1, "=", s2)
else:
  print(s1, ">", s2)
```

? Что выведет?

паровоз < пароход

первые отличающиеся буквы

Сравниваем с начала:
паровоз
пароход

! в < х!

«в»: код 1074

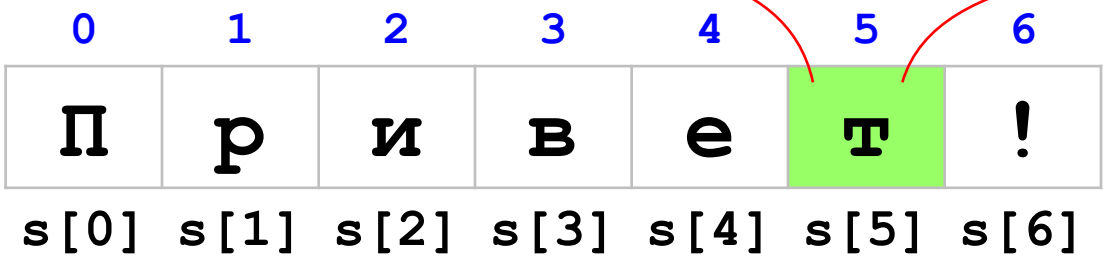
«х»: код 1093

Обращение к символу по номеру

```
print ( s[5] )
```

```
print ( s[-2] )
```

s[len(s)-2]



! Символы нумеруются с нуля!

составить «кот»

```
s = "информатика"
kot = s[-2]+s[3]+s[-4]
```

Посимвольная обработка строк

```
s[4] = "a"
```

Строка неизменна!

Задача. Ввести строку и заменить в ней все буквы «э» на буквы «е».

строим новую строку!

```
sNew = ""
for i in range(len(s)):
    if s[i] == "э":
        sNew += "е"
    else:
        sNew += s[i]
```

для каждого символа строки

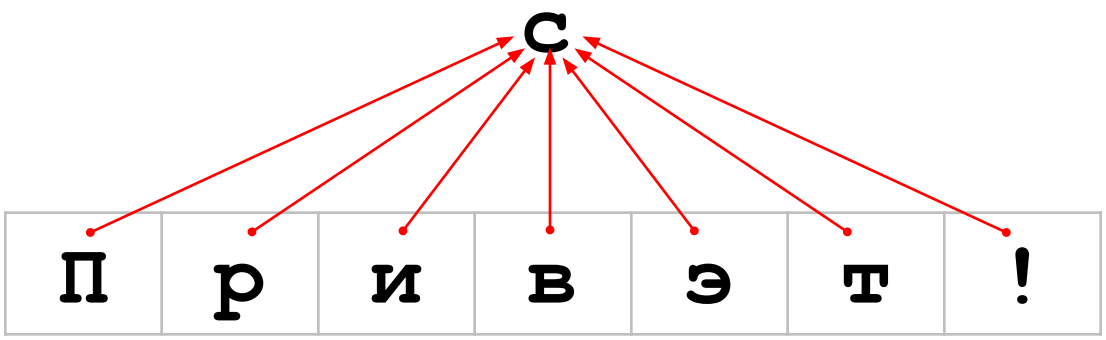
0	1	2	3	4	5	6
П	р	и	в	э	т	!

`len(s)-1`

Цикл перебора символов

```
sNew = ""
for c in s:
    if c == "э":
        sNew += "е"
    else:
        sNew += c
```

перебрать все символы строки



Задачи

«A»: Напишите программу, которая вводит строку, состоящую только из точек и букв X, и заменяет в ней все точки на нули и все буквы X на единицы.

Пример:

```
Введите строку: ..X.XX.
Двоичный код: 0010110
```

«B»: Напишите программу, которая в символьной строке заменяет все нули на единицы и наоборот. Остальные символы не должны измениться.

Пример:

```
Введите строку: 10a01Bx1010c
Инверсия: 01a10Bx0101c
```

Задачи

«С»: Введите битовую строку и дополните её последним битом, который должен быть равен 0, если в исходной строке чётное число единиц, и равен 1, если нечётное (в получившейся строке должно всегда быть чётное число единиц).

Пример:

```
Введите битовую строку: 01101010110
Результат: 011010101100
```

Операции со строками

Объединение *(конкатенация)* :

```
s1 = "Привет"
s2 = "Вася"
s  = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

Умножение:

```
s = "АУ"
s5 = s*5
```

```
s5 = s + s + s + s + s
```

АУАУАУАУАУ

? Что получим?

Срезы строк (выделение части строки)

```
s = "0123456789"
s1 = s[3:8]       # "34567"
```

с какого символа

до какого (**не включая 8**)

```
s = "0123456789"
s1 = s[:8]        # "01234567"
```

от начала строки

```
s = "0123456789"
s1 = s[3:]        # "3456789"
```

до конца строки

Срезы строк

Срезы с отрицательными индексами:

```
s = "0123456789"
s1 = s[:-2]              # "01234567"
```

len(s)-2

```
s = "0123456789"
s1 = s[-6:-2]            # "4567"
```

len(s)-6 len(s)-2

Операции со строками

Удаление:

```
s = "0123456789"
s1 = s[:3] + s[9:]
```

"012" "9" "0129"

Вставка:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
```

"012" "3456789"

"012ABC3456789"

Поиск в строках

```
s = "Здесь был Вася."
n = s.find ( "с" )    # n = 3
if n >= 0:
  print ( "Номер символа", n )
else:
  print ( "Символ не найден." )
```

Находит первое слева вхождение подстроки!

Поиск с конца строки:

```
s = "Здесь был Вася."
n = s.rfind ( "с" )     # n = 12
```

Задачи

«A»: Ввести с клавиатуры в одну строку фамилию и имя, разделив их пробелом. Вывести первую букву имени с точкой и потом фамилию.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр
П. Иванов
```

«B»: Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

Пример:

```
Введите фамилию, имя и отчество:
Иванов Петр Семёнович
П.С. Иванов
```

Задачи

«С»: Ввести адрес файла и «разобрать» его на части, разделенные знаком "/". Каждую часть вывести в отдельной строке.

Пример:

```
Введите адрес файла:
C:/Фото/2015/Байкал/shaman.jpg
C:
Фото
2015
Байкал
shaman.jpg
```

Преобразования «строка» → «число»

Из строки в число:

```
s = "123"
N = int ( s )          # N = 123
s = "123.456"
X = float ( s )        # X = 123.456
```

Из числа в строку:

```
N = 123
s = str ( N )                # s = "123"
s = "{:5d}".format(N)        # s = "  123"

X = 123.456
s = str ( X )                # s = "123.456"
s = "{:7.2f}".format(X)      # s = " 123.46"
s = "{:10.2e}".format(X)     # s = "  1.23e+02"
```

Задачи

«A»: Напишите программу, которая вычисляет сумму двух чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3
Ответ: 15
```

«B»: Напишите программу, которая вычисляет сумму трёх чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45
Ответ: 60
```

Задачи

«C»: Напишите программу, которая вычисляет сумму произвольного количества чисел, введенную в форме символьной строки. Все числа целые.

Пример:

```
Введите выражение:
12+3+45+10
Ответ: 70
```

«D»: Напишите программу, которая вычисляет выражение, содержащее целые числа и знаки сложения и вычитания.

Пример:

```
Введите выражение:
12+134-45-17
Ответ: 84
```

Программирование (Python)

§ 20. Обработка массивов

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
prx = int(input())
```

Найди ошибку!

```
count = 0          ut())
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

Как это записать?

```
сумма = 0
пока не введён 0:
  если число оканчивается на "5" то
     сумма:= сумма + число
```

```
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

«A»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.

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

Задачи

«C»: 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

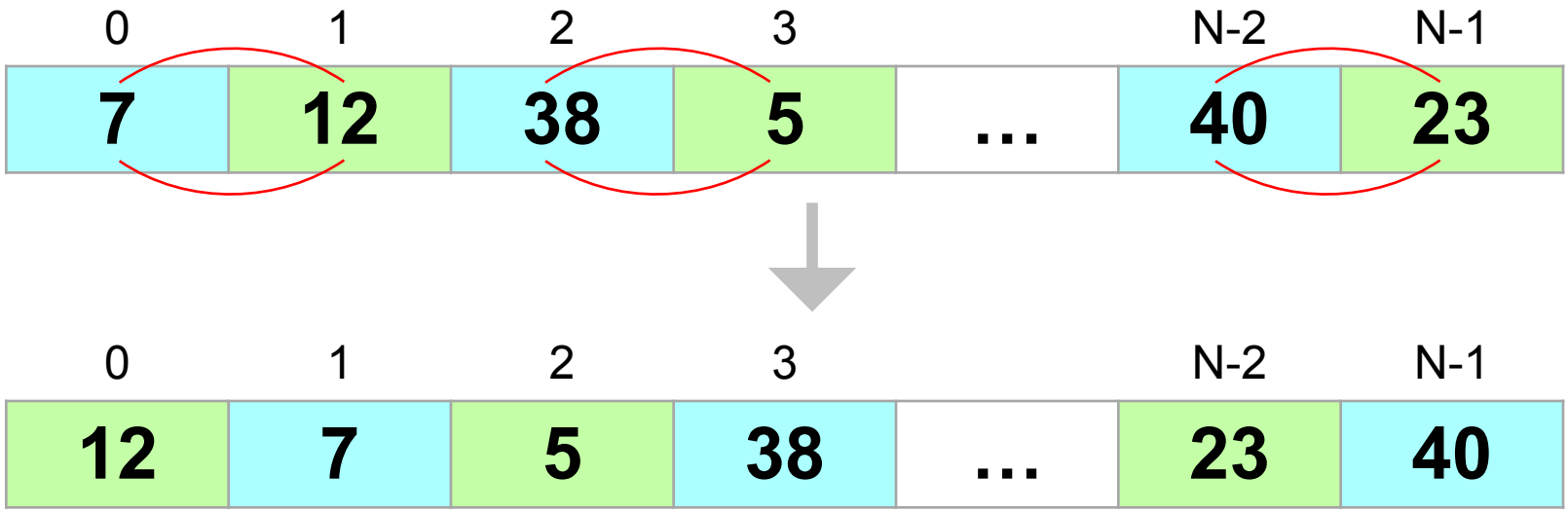
```
c = a
a = b
b = c
```

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

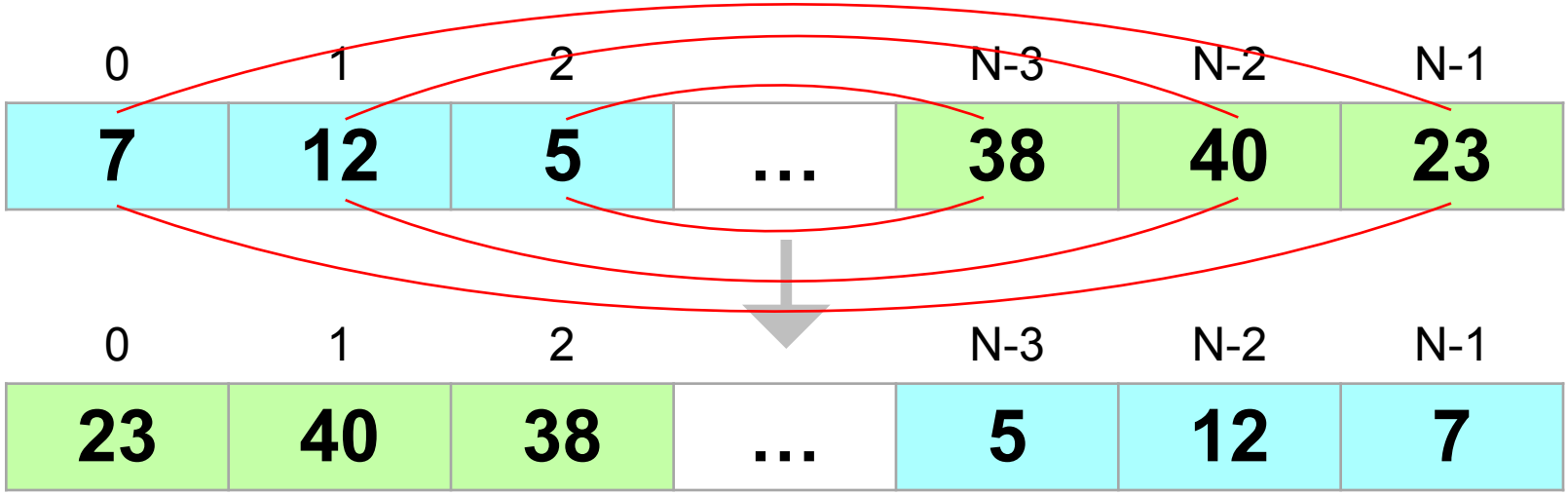
Перестановка пар соседних элементов

```
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]↔A[2], A[3]↔A[4], …, A[N-1]↔A[N]

Реверс массива

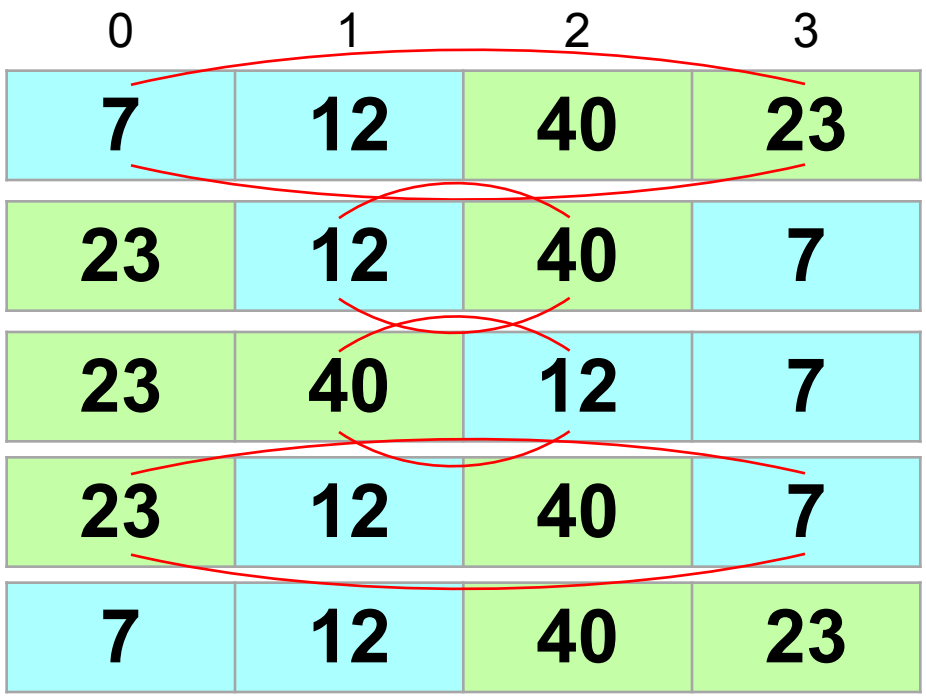
Задача. Переставить элементы массива в обратном порядке (выполнить *реверс*).




```
A[0]↔A[N-1]      0+N-1 = N-1
A[1]↔A[N-2]      1+N-2 = N-1
A[i]↔A[N-1-i]    i+??? = N-1
A[N-1]↔A[0]      N-1+0 = N-1
```

Реверс массива

```
for i in range (N % 2):
    поменять местами A[i] и A[N+1-i]
```



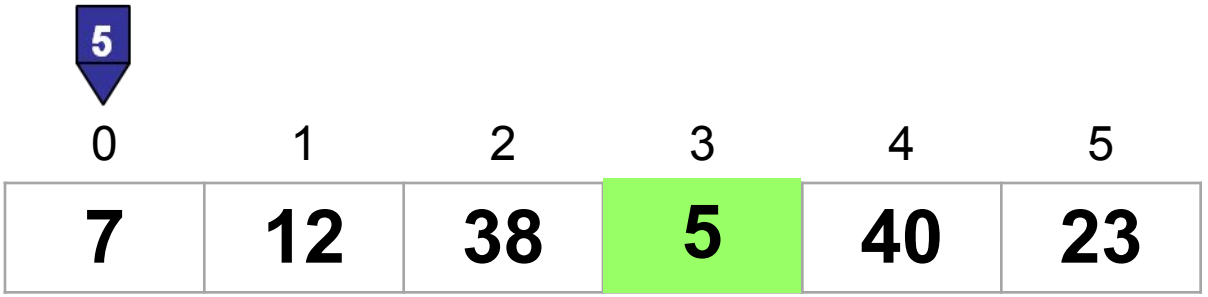
? Что плохо?

? Как исправить?

Линейный поиск в массиве

Задача. Найти в массиве элемент, равный X, и его номер.

X = 5




```
i = 0
while A[i]!=X:
    i += 1
print("A[", i, "]=", X)
```

? Что плохо?

? Если искать 4?

! Нельзя выходить за границы массива!

Линейный поиск в массиве

не выходим за границу

```
i = 0
while i<=N and A[i]!=X:
   i += 1
if i <= N:
   print( "A[",i,"]=",X )
else:
   print( "Не нашли!" )
```

? Как проверить, нашли или нет?

Досрочный выход из цикла

Задача. Найти в массиве элемент, равный X, и его **номер**.

```
nX = 0 # номер элемента
for i in range(N):
  if A[i]==X:
    nX = i # запомнить номер
    break
if nX > 0:
  print( "A[", nX, "]=", X )
else:
  print( "Не нашли!" )
```

нашли!

сразу выйти из цикла

Поиск в массиве

Варианты в стиле Python:

```
for i in range ( N ):
  if A[i] == X:
    print ( "A[", i, "]=", X, sep = "" )
    break
else:
  print ( "Не нашли!" )
```

если не было досрочного выхода из цикла

```
if X in A:
  nX = A.index(X)
  print ( "A[", nX, "]=", X, sep = "" )
else:
  print ( "Не нашли!" )
```

Задачи

«A»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20], выводит его на экран, а затем находит индекс первого элемента, равного введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Что ищем: 13
A[4] = 13
```

Задачи

«B»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [-10,10], выводит его на экран, а затем находит индекс **последнего** элемента, равного введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: -5 -6 2 3 -3 0 8 -3 0 9
Что ищем: 0
A[9] = 0
```

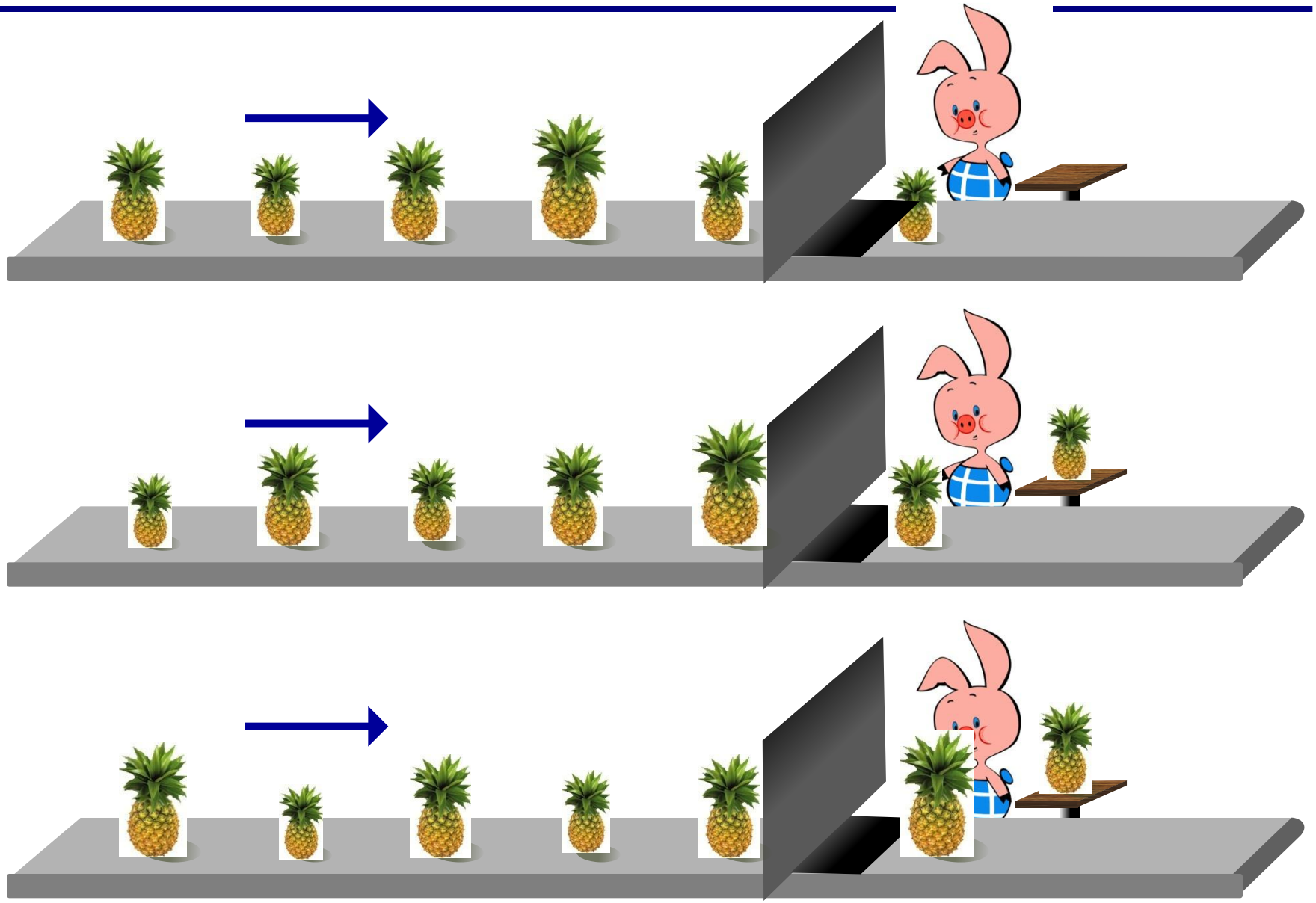
Задачи

«С»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [10,50], выводит его на экран, а затем находит индексы всех элементов, равных введённому числу *X*. Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

Пример:

```
Массив: 12 45 30 18 30 15 30 44 32 17
Что ищем: 30
A[3] = 30
A[5] = 30
A[7] = 30
```

Поиск максимального элемента



Поиск максимального элемента

? Какие переменные нужны?

```
for i in range(N):
    if A[i] > M:
        M = A[i]
print( M )
```

? Чего не хватает?

? Какое начальное значение взять для **M**?

1) **M** – значение, которое заведомо меньше всех элементов массива

или

2) **M = A[0]** (или любой другой элемент)

максимальный не меньше, чем **A[0]**

Поиск максимального элемента

```
M = A[0]
for i in range(1,N):
    if A[i] > M:
        M = A[i]
print( M )
```

начинаем с `A[1]`, так как `A[0]` мы уже посмотрели

? Как найти минимальный?

Поиск максимального элемента (Python)

```
M = A[0]
for x in A:
    if x > M:
        M = x
print( M )
```

перебрать все элементы в массиве A

!

Не нужно знать размер!

```
print( max(A) )
```

```
print( min(A) )
```

Номер максимального элемента

Задача. Найти в массиве максимальный элемент и его номер.

? Какие переменные нужны?

```
M = A[0]; nMax = 0
for i in range(1,N):
    if A[i] > M:
        M = A[i]
        nMax = i
print( "A[", nMax, "]=", M )
```

? Можно ли убрать одну переменную?

Номер максимального элемента

Если знаем **nMax**, то **M=A[nMax]**!

```
M = A[0]; nMax = 0
for i in range(1,N):
   if A[i]>A[nMax]:
      M = A[i]
      nMax = i
print( "A[", nMax, "]=", A[nMax] )
```

Максимальный элемент и его номер

Вариант в стиле Python:

```
M = max(A)
nMax = A.index(M)
print ( "A[", nMax, "]=", M )
```

номер заданного элемента (первого из…)

Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M = A[0]
for i in range(1,N):
   if A[i]<0 and A[i]>M:
      M = A[i]
print( M )
```

? Что плохо?

? Как исправить?

0	1	2	3	4
5	–2	8	3	–1

M = 5

Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M = A[0]
for i in range(1,N):
    if A[i] < 0:
        if M >= 0 or A[i]> M:
            M = A[i]
print( M )
```

сначала записали неотрицательный!

? Если нет отрицательных?

Максимальный не из всех (Python)

Задача. Найти в массиве максимальный из отрицательных элементов.

```
B = [ x for x in A
          if x < 0]
print( max(B) )
```

отбираем нужные

?

Если нет отрицательных?

```
if len(B):
  print( max(B) )
else:
  print("Нет таких!")
```

```
if len(B)!= 0:
```

Задачи (без min и max)

«A»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [50; 150] и находит в нём минимальный и максимальный элементы и их номера.

«B»: Напишите программу, которая получает с клавиатуры значения элементов массива и выводит количество элементов, имеющих максимальное значение.

«C»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [100; 200] и находит в нём пару соседних элементов, сумма которых минимальна.

Задачи

«D»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [–100; 100] и находит в каждой половине массива пару соседних элементов, сумма которых максимальна.

Задачи-2 (максимум в потоке)

«A»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное и максимальное среди полученных чисел.

«B»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное число, делящееся на 3, среди полученных чисел.

«C»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Напишите программу, которая находит максимальное двузначное число, заканчивающееся на 6, среди полученных чисел.

Задачи-2 (максимум в потоке)

«D»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Напишите программу, которая находит среди полученных чисел пару полученных друг за другом чисел, сумма которых максимальна.

Сортировка

Сортировка — это расстановка элементов списка (массива) в заданном порядке.

Задача. Отсортировать элементы в порядке **возрастания** (*неубывания* – если есть одинаковые).

Алгоритмы сортировки:

- простые, но медленные (при больших N)
- быстрые, но сложные…

Сортировка выбором

? Где должен стоять минимальный элемент?

- нашли минимальный, поставили его на первое место

```
c = A[nMin]
A[0],A[nMin] = A[nMin],A[0]
A[0] = c
```

? Как?

- из оставшихся нашли минимальный, поставили его на второе место и т.д.

? Что дальше?

5	–2	8	3	–1
–2	5	8	3	–1
–2	–1	8	3	5

Сортировка выбором

```
for i in range(N-1):
    # ищем минимальный среди A[i]..A[N-1]
    nMin = i
    for j in range(i+1,N):
        if A[j] < A[nMin]:
            nMin = j
    # переставляем A[i] и A[nMin]
    A[i],A[nMin] = A[nMin],A[i]
```

не трогаем те, которые уже поставлены

? Почему цикл N-1 раз?

Решение в стиле Python:

```
A.sort()
```

Задачи

«A»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20] и сортирует его в порядке убывания.

Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Сортировка: 18 16 16 14 13 13 9 5 3 2
```

«B»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [10,100] и сортирует его по возрастанию последней цифры числа (сначала идут все числа, которые заканчиваются на 0, потом все, которые заканчиваются на 1, и т.д.).

Пример:

```
Массив: 12 10 31 40 55 63 28 87 52 92
Сортировка: 10 40 31 12 52 92 63 55 87 28
```

Задачи

«С»: Напишите программу, которая заполняет массив из *N* = 10 элементов случайными числами в диапазоне [0,20] и сортирует его в порядке возрастания. На каждом шаге цикла выполняется поиск максимального (а не минимального!) элемента.

Пример:

```
Массив: 5 16 2 13 3 14 18 13 16 9
Сортировка: 2 3 5 9 13 13 14 16 16 18
```

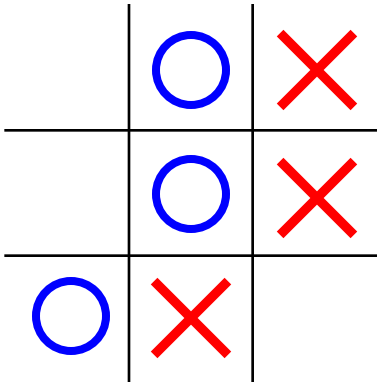
Программирование (Python)

§ 21. Матрицы (двумерные массивы)

Что такое матрица?

нолик

нет знака



крестик

строка 2, столбец 3

? Как закодировать?

Матрица — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.).

Каждый элемент матрицы имеет два индекса – номера строки и столбца.

Создание матриц

Матрица – это массив масивов!

```
A = [[-1, 0, 1],
     [-1, 0, 1],
     [0, 1, -1]]
```

перенос на другую строку внутри скобок

или так:

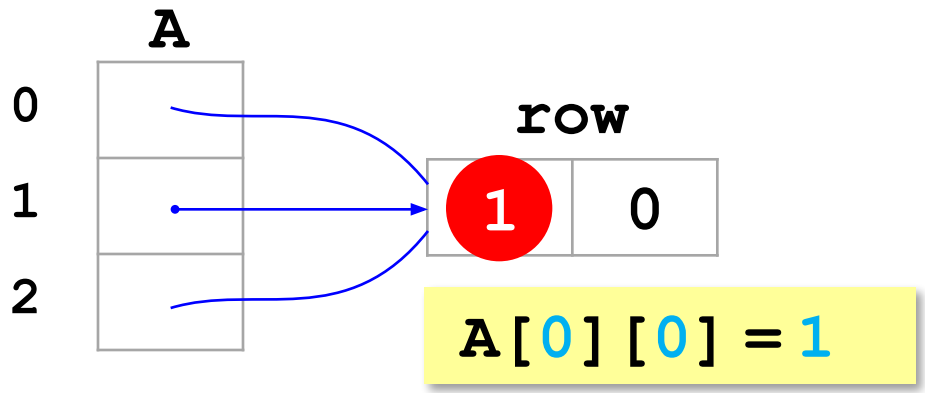
```
A = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
```

Нумерация элементов с нуля!

Создание матриц

Нулевая матрица:

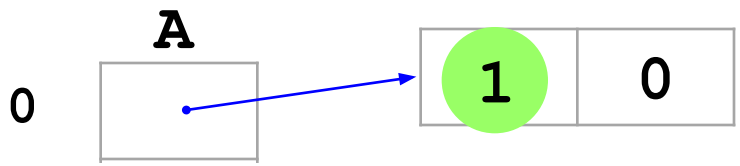
```
N = 3
M = 2
row = [0]*M
A = [row]*N
```



A[0][0] = 1

а правильно так:

```
A = []
for i in range(N):
    A.append( [0]*M )
```



A[0][0] = 1

Вывод матриц

```
print ( A )
```

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

```
def printMatrix( A ):
  for row in A:
    for x in row:
      print ( "{:4d}".format(x), end="" )
    print ()
```

```
   1   2   3
   4   5   6
   7   8   9
```

Зачем форматный вывод?

Простые алгоритмы

Заполнение случайными числами:

```
from random import randint
for i in range(N):
  for j in range(M):
    A[i][j] = randint ( 20, 80 )
    print ( "{:4d}".format(A[i][j]),
            end = "" )
  print()
```

Вложенный цикл!

Суммирование:

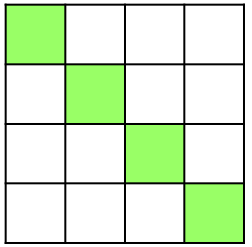
```
s = 0
for i in range(N):
  for j in range(M):
    s += A[i][j]
print ( s )
```

```
s = 0
for row in A:
  s += sum(row)
print ( s )
```

Перебор элементов матрицы

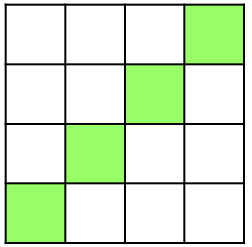
Главная диагональ:

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



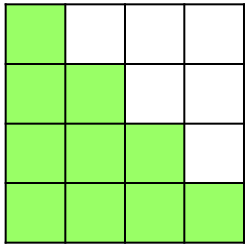
Побочная диагональ:

```
for i in range(N):
  # работаем с A[i][N-1-i]
```



Главная диагональ и под ней:

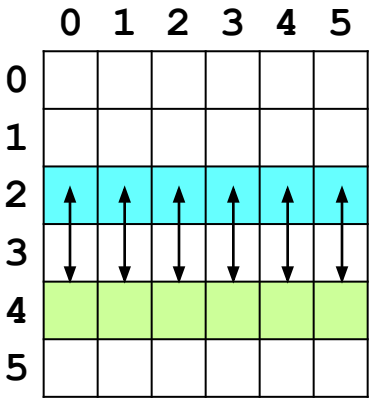
```
for i in range(N):
  for j in range(i+1):
    # работаем с A[i][j]
```



Перестановка строк

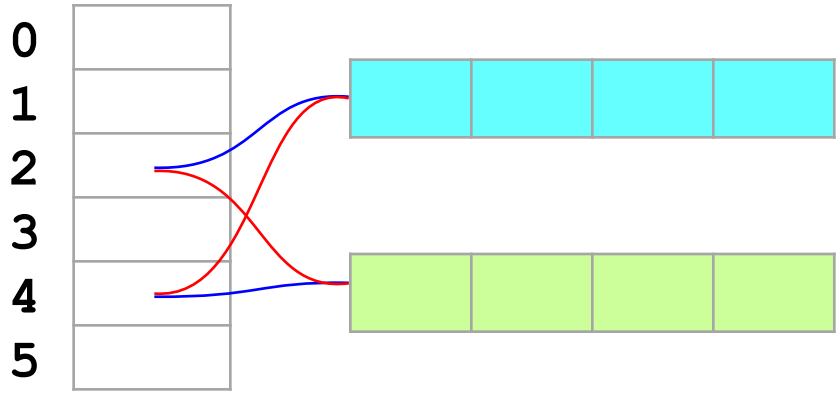
2-я и 4-я строки:

```
for j in range(M):
    A[2][j],A[4][j] = A[4][j],A[2][j]
    A[2][j] = A[4][j]
    A[4][j] = c
```



Решение в стиле Python:

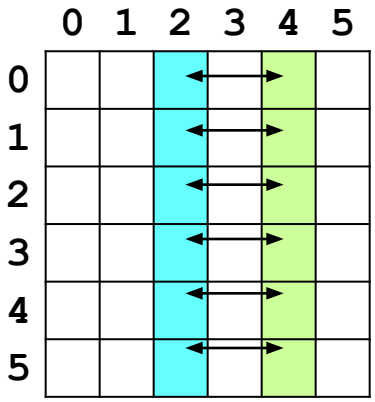
```
A[2],A[4] = A[4],A[2]
```



Перестановка столбцов

2-й и 4-й столбцы:

```
for i in range(N):
    A[i][2],A[i][4] = A[i][4],A[i][2]
    A[i][2] = A[i][4]
    A[i][4] = c
```



Задачи

«А»: Напишите программу, которая заполняет матрицу случайными числами и находит максимальный элемент на главной диагонали квадратной матрицы.

Пример:

```
Матрица A:
12 34 14 65
71 88 23 45
87 46 53 39
76 58 24 92
Результат: A[3][3] = 92
```

Задачи

«B»: Напишите программу, которая заполняет матрицу случайными числами и находит максимальный элемент матрицы и его индексы (номера строки и столбца).

Пример:

```
Матрица A:
12 34 14 65
71 88 23 98
87 46 53 39
76 58 24 92
Максимум: A[1][3] = 98
```

Задачи

«С»: Напишите программу, которая заполняет матрицу случайными числами и находит минимальный из чётных положительных элементов матрицы. Учтите, что таких элементов в матрице может и не быть.

Пример:

```
Матрица A:
16 34 14 65
71 88 23 45
87 12 53 39
76 58 24 92
Результат: A[2][1] = 12
```

Программирование (Python)

§ 22. Сложность алгоритмов

Как сравнивать алгоритмы?

- быстродействие (**временна́я сложность**)
- объём требуемой памяти (**пространственная сложность**)
- понятность

Обычно не бывает все хорошо!

Время работы алгоритма – это количество элементарных операций *T*, выполненных исполнителем.

зависит от количества данных (размера массива *N*)

Функция *T(N)* называется **временно́й сложностью алгоритма**

$T(N) = 2N^3$

Как увеличится время работы при увеличении *N* в 10 раз?

Примеры определения сложности

Задача 1. Вычислить сумму первых трёх элементов массива (при $N \geq 3$).

```
Sum = A[1] + A[2] + A[3]
```

$T(N) = 3$

2 сложения + запись в память

Задача 2. Вычислить сумму всех элементов массива.

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

$T(N) = 2N + 1$

N сложений, N+1 операций записи

Примеры определения сложности

Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
for i in range(N-1):
  nMin = i
  for j in range(i+1,N):
    if A[i] < A[nMin]:
      nMin = j
  A[i],A[nMin] = A[nMin],A[i]
```

Число сравнений:

$$T_c(N) = (N-1) + (N-2) + \ldots + 2 + 1 = \frac{N(N-1)}{2} = \frac{1}{2}N^2 - \frac{1}{2}N$$

Число перестановок: $T_n(N) = N - 1$

Примеры определения сложности

Задача 4. Найти сумму элементов квадратной матрицы размером *N*×*N*.

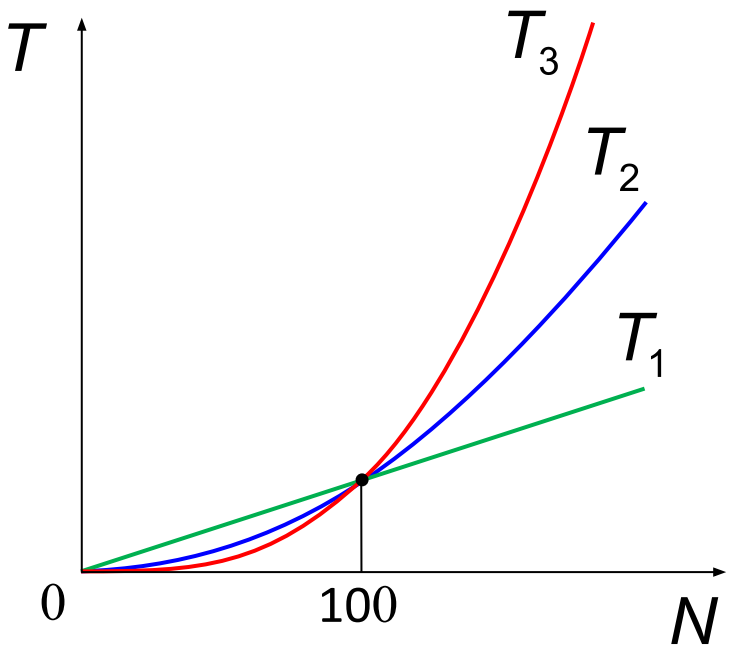
```
Sum = 0
for i in range(N):
  for j in range(N):
    Sum += A[i,j]
```

Самостоятельно! ☺

Сравнение алгоритмов по сложности

$T_1(N) = 10000 \cdot N$ $T_2(N) = 100 \cdot N^2$ $T_3(N) = N^3$

? Какой алгоритм выбрать?



при $N < 100$:

$$T_3(N) < T_2(N) < T_1(N)$$

при $N > 100$:

$$T_3(N) > T_2(N) > T_1(N)$$

! Нужно знать размер данных!

Асимптотическая сложность

Асимптотическая сложность – это оценка скорости роста количества операций при больших значениях N.

линейная

постоянная

сложность $O(N)$ $\Leftrightarrow$ $T(N) \le c \cdot N$ для $N \ge N_0$

сумма элементов массива:

$$T(N) = 2 \cdot N - 1 \le 2 \cdot N \text{ для } N \ge 1 \Rightarrow O(N)$$

квадратичная

сложность $O(N^2)$ $\Leftrightarrow$ $T(N) \le c \cdot N^2$ для $N \ge N_0$

сортировка методом выбора:

$$T_c(N) = \frac{1}{2}N^2 - \frac{1}{2}N \le \frac{1}{2}N^2 \text{ для } N \ge 0 \Rightarrow O(N^2)$$

Асимптотическая сложность

кубичная

сложность $O(N^3)$ ⇔ $T(N) \le c \cdot N^3$ для $N \ge N_0$

сложность $O(2^N)$

сложность $O(N!)$

задачи оптимизации, полный перебор вариантов

Факториал числа *N*: $N! = 1 \cdot 2 \cdot 3 \ldots \cdot N$

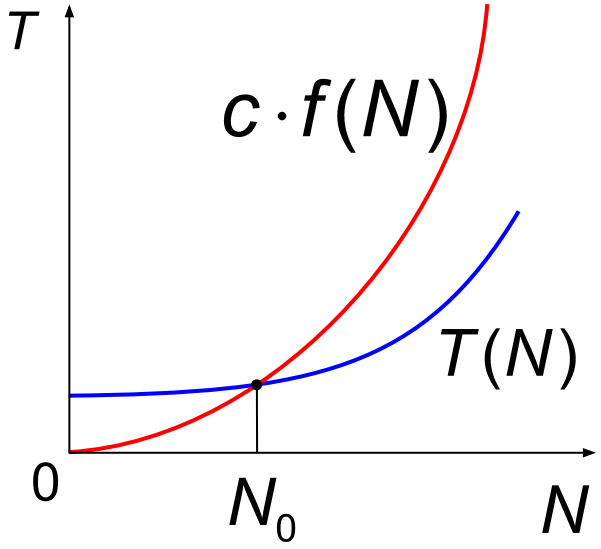
$T(N)$	**время выполнения**
N	100 нс
N^2	10 мс
N^3	0,001 с
2^N	10^{13} лет

N = 100,
1 млрд оп/с

Асимптотическая сложность

Алгоритм относится к классу $O(f(N))$, если найдется такая постоянная c, что начиная с некоторого $N = N_0$ выполняется условие

$$T(N) \le c \cdot f(N)$$



это верхняя оценка!

$$O(N) \Rightarrow O(N^2) \Rightarrow O(N^3) \Rightarrow O(2^N)$$

«Алгоритм имеет сложность $O(N^2)$».

обычно – наиболее точная верхняя оценка!

Программирование (Python)

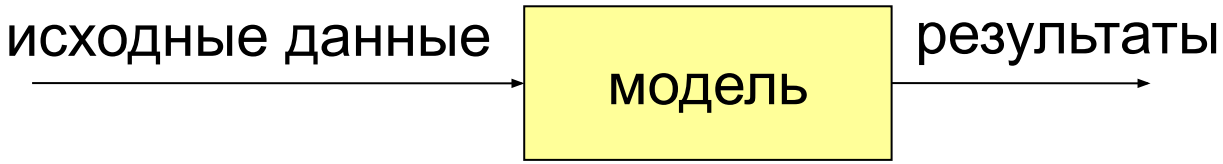
§ 23. Как разрабатывают программы

Этапы разработки программ

I. Постановка задачи

Документ: *техническое задание*.

II. Построение модели



Формализация: запись модели в виде формул (на формальном языке).

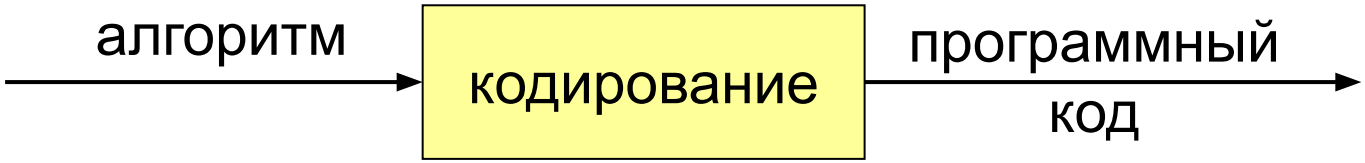
III. Разработка алгоритма и способа хранения данных

«Алгоритмы + структуры данных = программы»
(Н. Вирт)

Этапы разработки программ

IV. Кодирование

Запись алгоритма на языке программирования.



V. Отладка

Поиск и исправление ошибок в программах:

- **синтаксические** – нарушение правил языка программирования
- **логические** – ошибки в алгоритме

могут приводить к **отказам** – аварийным ситуациям во время выполнения *(run-time error)*

Этапы разработки программ

VI. Тестирование

Тщательная проверка программы во всех режимах:

- **альфа-тестирование** – внутри компании (тестировщики)
- **бета-тестирование** – (доверенные) пользователи

VII. Документирование

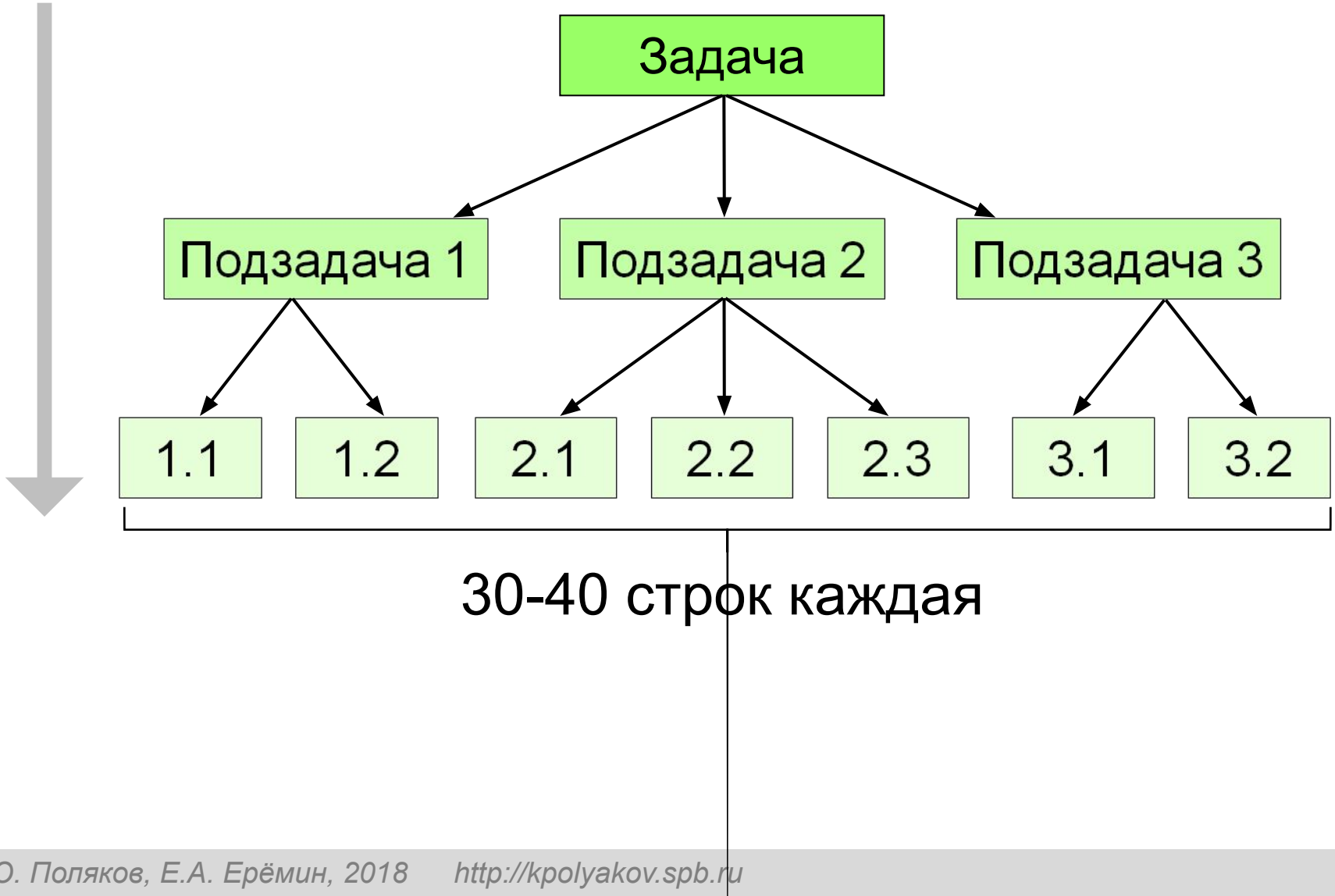
Технические писатели

VIII. Внедрение и сопровождение

- обучение пользователей
- исправление найденных ошибок
- техподдержка

Методы проектирования программ

«Сверху вниз» (последовательное уточнение)



Методы проектирования программ

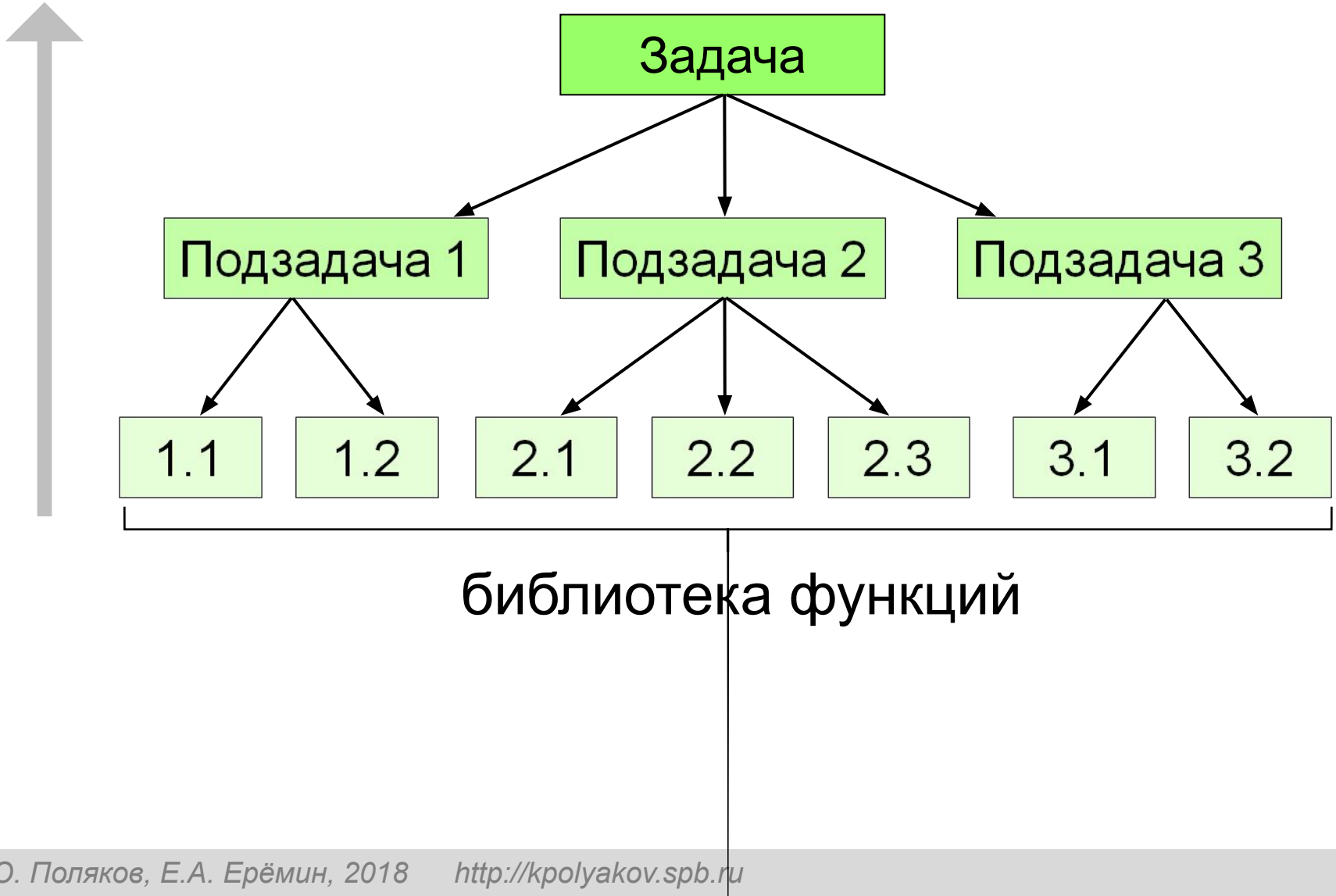
«Сверху вниз» (последовательное уточнение)

- сначала задача решается «в целом»
- легко распределить работу
- легче отлаживать программу (всегда есть полный работающий вариант)

- в нескольких подзадачах может потребоваться решение одинаковых подзадач нижнего уровня
- быстродействие не известно до последнего этапа (определяется нижним уровнем)

Методы проектирования программ

«Снизу вверх» (восходящее)



Методы проектирования программ

«Снизу вверх» (восходящее)

- нет дублирования
- сразу видно быстродействие

- сложно распределять работу
- сложнее отлаживать (увеличение числа связей)
- плохо видна задача «в целом», может быть нестыковка на последнем этапе

Почти всегда используют оба подхода!

Отладка 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?

Документирование программы

- назначение программы
- формат входных данных
- формат выходных данных
- примеры использования программы

Назначение:

программа для решения уравнения

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

Формат входных данных:

значения коэффициентов a, b и c вводятся с клавиатуры через пробел в одной строке

Документирование программы

Формат выходных данных:

значения вещественных корней уравнения; если вещественных корней нет, выводится слово «нет»

Примеры использования программы:

1. Решение уравнения $x^2 - 5x + 6 = 0$

```
Введите a, b, c: 1 -5 6
x1=3 x2=2
```

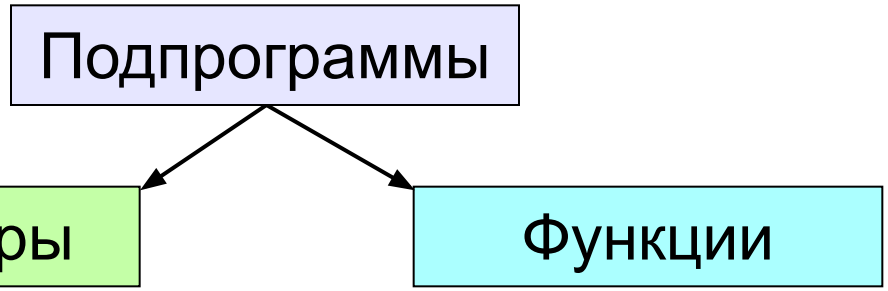
2. Решение уравнения $x^2 + x + 6 = 0$

```
Введите a, b, c: 1 1 6
Нет.
```

Программирование (Python)

§ 24. Процедуры

Два типа подпрограмм



выполняют действия

+ возвращают некоторый результат

? Процедура или функция?

а) рисует окружность на экране
б) определяет площадь круга
в) вычисляет значение синуса угла
г) изменяет режим работы программы
д) возводит число x в степень y
е) включает двигатель автомобиля
ж) проверяет оставшееся количество бензина в баке
з) измеряет высоту полёта самолёта

Простая процедура

define – определить

```
def printLine():
    print("----------")

...
printLine()
...
```

вызов процедуры

какие-то операторы

Что делает?

- можно вызывать сколько угодно раз
- нет дублирования кода
- изменять – в одном месте

Линии разной длины

```
def printLine5():
  print("-----")
```

```
def printLine10():
  print("----------")
```

```
def printLine10():
  print("-"*10)
```

```
def printLine( n ):
  print("-"*n)
```

Как улучшить?

параметр процедуры

Процедура с параметром

Параметр – величина, от которой зависит работа процедуры.

```
def printLine( n ):
    ...
```

```
...
printLine(10)
...
printLine(7)
printLine(5)
printLine(3)
```

? Что делает?

Аргумент – значение параметра при конкретном вызове.

Несколько параметров

символьная строка

```
def printLine(c, n):
  print(c*n)
```

? Что изменилось?

? Как вызывать?

✓
```
printLine( "+", 5 )
```

✓
```
printLine( "+-+", 5 )
```

✓
```
printLine( 5, "+" )
```

В других языках программирования

Паскаль:

```
procedure printLine(c: string; n: integer);
var i: integer;
begin
  for i:=1 to n do
    write(c);
  writeln
end;
```

В других языках программирования

C:

```
void printLine(int n)
{
   int i;
   for (i=1; i<=n; i++)
     putchar("-");
   putchar("\n");
}
```

Как не нужно писать процедуры

```
def summa():
    print(x + y)
```

```
x = 10
y = 5
summa()
```

Что плохо?

```
def summa( x, y ):
    print(x + y)
```

```
x = 10
y = 5
summa( x, y )
summa( 2*x+y, 7 )
```

- только $x + y$
- не перенести в другую программу

! Процедура принимает данные только через параметры!

Задачи

«A»: Напишите процедуру, которая принимает параметр – натуральное число N – и выводит на экран две линии из N символов "–".

Пример:

```
Длина цепочки: 7
-------
-------
```

«B»: Напишите процедуру, которая принимает один параметр – натуральное число *N*, – и выводит на экран прямоугольник длиной *N* и высотой 3 символа.

Пример:

```
Длина прямоугольника: 7
ooooooo
o     o
ooooooo
```

Задачи

«С»: Напишите процедуру, которая выводит на экран квадрат со стороной *N* символов. При запуске программы *N* нужно ввести с клавиатуры.

Пример:

```
Сторона квадрата: 5
ooooo
o   o
o   o
o   o
ooooo
```

Задачи

«D»: Напишите процедуру, которая выводит на экран треугольник со стороной *N* символов. При запуске программы *N* нужно ввести с клавиатуры.

Пример:

```
Сторона: 5
o
oo
ooo
oooo
ooooo
```

Рекурсия

Задача. Вывести на экран двоичный код натурального числа.

```
def printBin( n ):
    ...
```

Алгоритм перевода через остатки:

```
while n!=0:
    print(n % 2, end="")
    n = n // 2
```

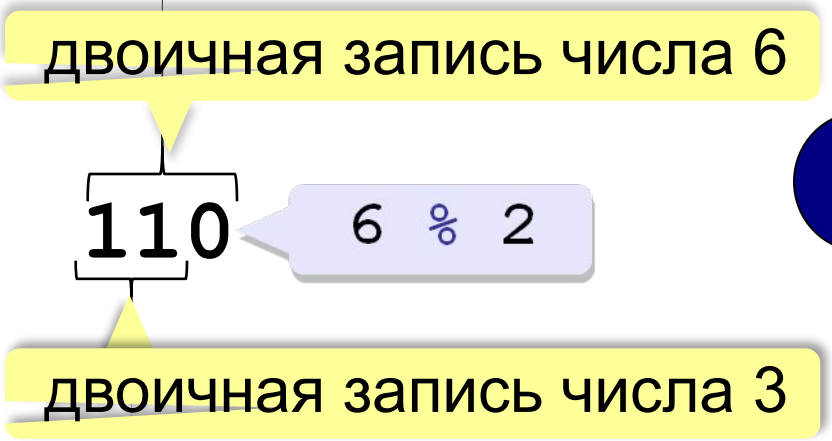
Что получится при **n = 6**?

011 ✗

в обратном порядке!

Рекурсия

Чтобы вывести двоичную запись числа **n**, нужно сначала вывести двоичную запись числа **(n // 2)**, а затем — его последнюю двоичную цифру, равную **(n % 2)**.



Чтобы решить задачу, нужно решить ту же задачу для меньшего числа!

Это и есть **рекурсия**!

Чтобы понять рекурсию, нужно понять рекурсию! ☺

Рекурсивная процедура

```
def printBin( n ):
  printBin(n % 2)
  print(n % 2, end = "")
```

вызывает сама себя!

Рекурсивная процедура — это процедура, которая вызывает сама себя.

```
printBin(6)
  printBin(3)
    printBin(1)
      printBin(0)
        printBin(0)
```

? Что получится? `printBin(6)`

бесконечные вызовы

? Как исправить?

Рекурсивная процедура

```
def printBin( n ):
    if n = 0: return
    printBin(n // 2)
    print(n % 2)
```

? Что получится?
printBin(6)

```
printBin(6)
    printBin(3)
        printBin(1)
            printBin(0)
        print(1 % 2)
    print(3 % 2)
print(6 % 2)
```

рекурсия заканчивается!

1 1 0

Задачи

«A»: Напишите рекурсивную процедуру, которая переводит число в восьмеричную систему.

Пример:

```
Введите число: 66
В восьмеричной: 102
```

«B»: Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 9.

Пример:

```
Введите число: 75
Основание: 6
В системе с основанием 6: 203
```

Задачи

«C»: Напишите рекурсивную процедуру, которая переводит число в шестнадцатеричную систему.

Пример:

```
Введите число: 123
В шестнадцатеричной: 7B
```

«D»: Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 36.

Пример:

```
Введите число: 350
Основание: 20
В системе с основанием 20: HA
```

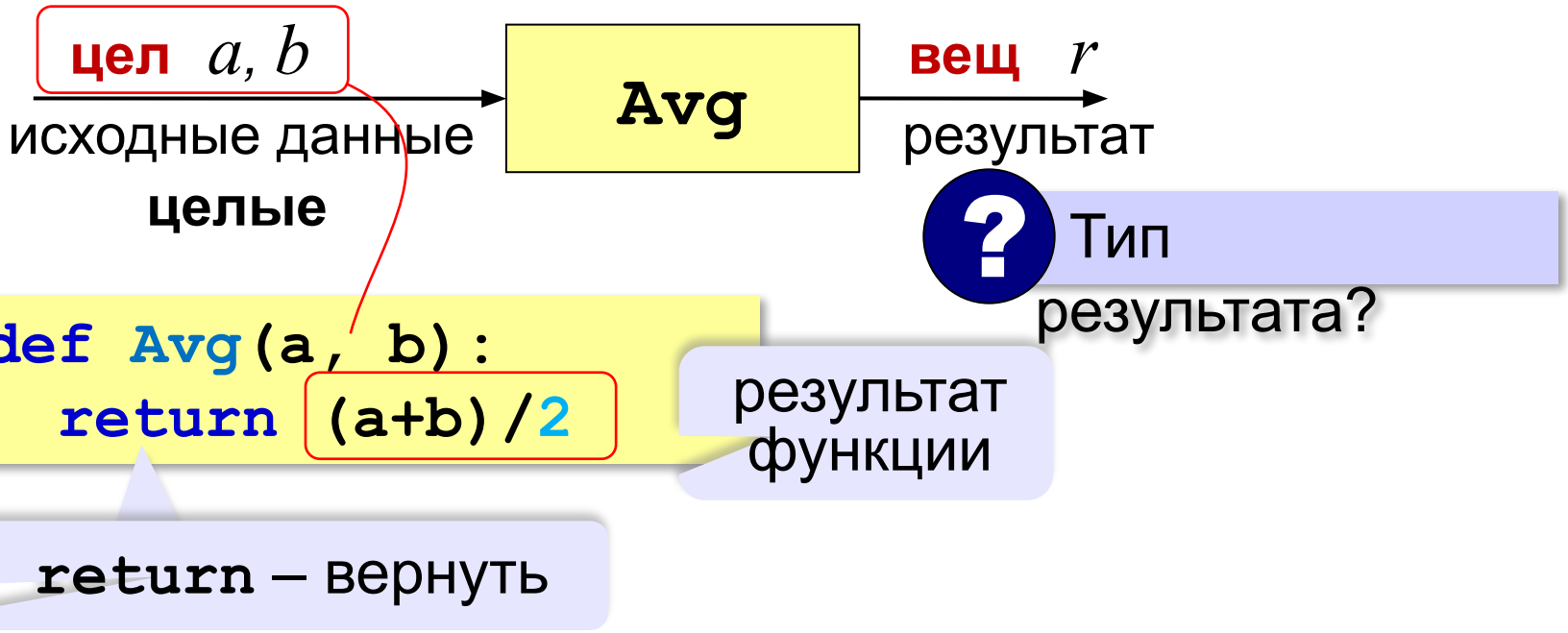
Программирование (Python)

§ 25. Функции

Что такое функция?

Функция — это вспомогательный алгоритм, который возвращает результат (число, строку символов и др.).

Задача. Написать функцию, которая вычисляет среднее арифметическое двух целых чисел.



Как вызывать функцию?

Запись результата в переменную:

```
sr = Avg(5, 8)
```

6.5

? Чему равно?

```
x = 2; y = 5
sr = Avg(x, 2*y+8)
```

10

Вывод на экран:

```
x = 2; y = 5
sr = Avg(x, y+3)
print( Avg(12,7) )
print( sr + Avg(x,12) )
```

5
9.5
12

Как вызывать функцию?

Использование в условных операторах:

```
a = int(input())
b = int(input())
if Avg(a,b) > 5:
   print("Да!")
else:
   print("Нет!");
```

? Когда печатает «Да»?

Как вызывать функцию?

Использование в циклах:

```
a = int(input())
b = int(input())
while Avg(a,b) > 0:
   print("Нет!")
   a,b = map(int, input().split())
print("Угадал!");
```

ввод двух чисел в одной строчке

? Когда напечатает «Угадал»?

В других языках программирования

Паскаль:

```
function Avg(a, b: integer): real;
begin
  Avg :=(a+b)/2
end.
```

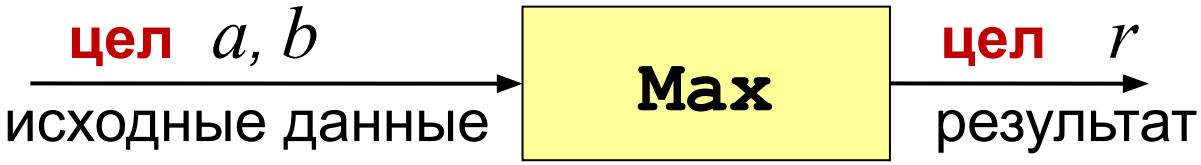
специальная переменная для записи результата функции

C:

```
float Avg(int a, int b)
{
  return (a+b)/2.0;
}
```

Максимум из двух (трёх) чисел

Задача. Составить функцию, которая определяет наибольшее из двух целых чисел.




```
def Max(a, b):
  if a > b then
    return a
  else:
    return b
```

?

Как с её помощью найти максимум из трёх?

```
def Max3(a, b, c):
  return Max( Max(a,b), c )
```

Сумма цифр числа

Задача. Составить функцию, которая вычисляет сумму значений цифр натурального числа.

```
def sumDigits( N ):
  sum = 0              # накапливаем сумму с 0
  while N!=0:
    d = N % 10         # выделим последнюю цифру
    sum += d           # добавим к сумме
    N = N // 10        # удалим последнюю цифру
  return sum
```

Задачи

«A»: Напишите функцию, которая вычисляет среднее арифметическое пяти целых чисел.

Пример:

```
Введите 5 чисел: 1 2 3 4 6
Среднее: 3.2
```

«B»: Напишите функцию, которая находит количество цифр в десятичной записи числа.

Пример:

```
Введите число: 751
Количество цифр: 3
```

Задачи

«С»: Напишите функцию, которая находит количество единиц в двоичной записи числа.

Пример:

```
Введите число: 75
Количество единиц: 4
```

Логические функции

Логическая функция — это функция, возвращающая логическое значения (**да** или **нет**).

- можно ли применять операцию?
- успешно ли выполнена операция?
- обладают ли данные каким-то свойством?

Логические функции

Задача. Составить функцию, которая возвращает «True», если она получила чётное число и «False», если нечётное.

```
def Even( N ):
  if N % 2 == 0:
    return True
  else:
    return False
```

```
def Even( N ):
  return (N % 2 == 0)
```

Рекурсивные функции

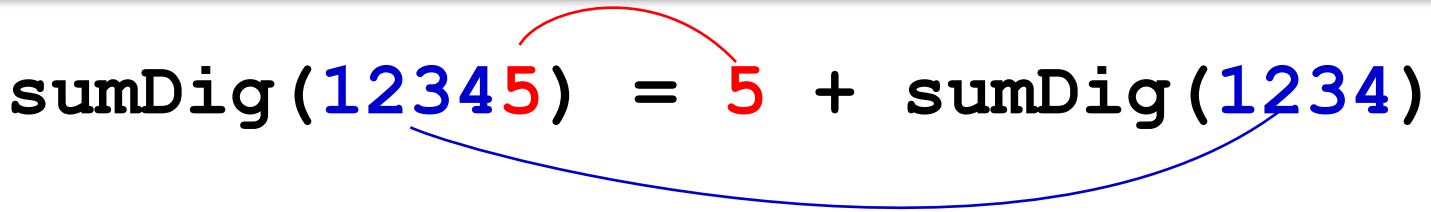
Рекурсивная функция — это функция, которая вызывает сама себя.

Задача. Составить рекурсивную функцию, которая вычисляет сумму цифр числа.

? Как сформулировать решение рекурсивно?

Сумму цифр числа N нужно выразить через сумму цифр другого (меньшего) числа.

Сумма цифр числа N равна значению последней цифры плюс сумма цифр числа, полученного отбрасыванием последней цифры.



Рекурсивная функция

Сумма цифр числа N

Вход: натуральное число **N**.

Шаг 1: `d = N % 10`

Шаг 2: `M = N // 10`

Шаг 3: **s** = сумма цифр числа **M**

Шаг 4: `sum = s + d`

Результат: **sum**.

последняя цифра

число без последней цифры

? Что забыли?

? Когда остановить?

Сумма цифр числа (рекурсия)

```
def sumDigRec( N ):
    if N == 0: return 0
    else:
        d = N % 10
        sum = sumDigRec(N // 10)
        return sum + d
```

? Зачем это?

? Где рекурсивный вызов?

Задачи

«A»: Напишите логическую функцию, которая возвращает значение «истина», если десятичная запись числа заканчивается на цифру 0 или 1.

Пример:

```
Введите число: 1230
Ответ: Да
```

«B»: Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число помещается в 8-битную ячейку памяти.

Пример:

```
Введите число: 751
Ответ: Нет
```

Задачи

«С»: Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число простое (делится только на само себя и на единицу).

Пример:

```
Введите число: 17
Число простое!
```

Пример:

```
Введите число: 18
Число составное!
```

Конец фильма

ПОЛЯКОВ Константин Юрьевич
д.т.н., учитель информатики
ГБОУ СОШ № 163, г. Санкт-Петербург
kpolyakov@mail.ru

ЕРЕМИН Евгений Александрович
к.ф.-м.н., доцент кафедры мультимедийной дидактики и ИТО ПГГПУ, г. Пермь
eremin@pspu.ac.ru

Источники иллюстраций

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