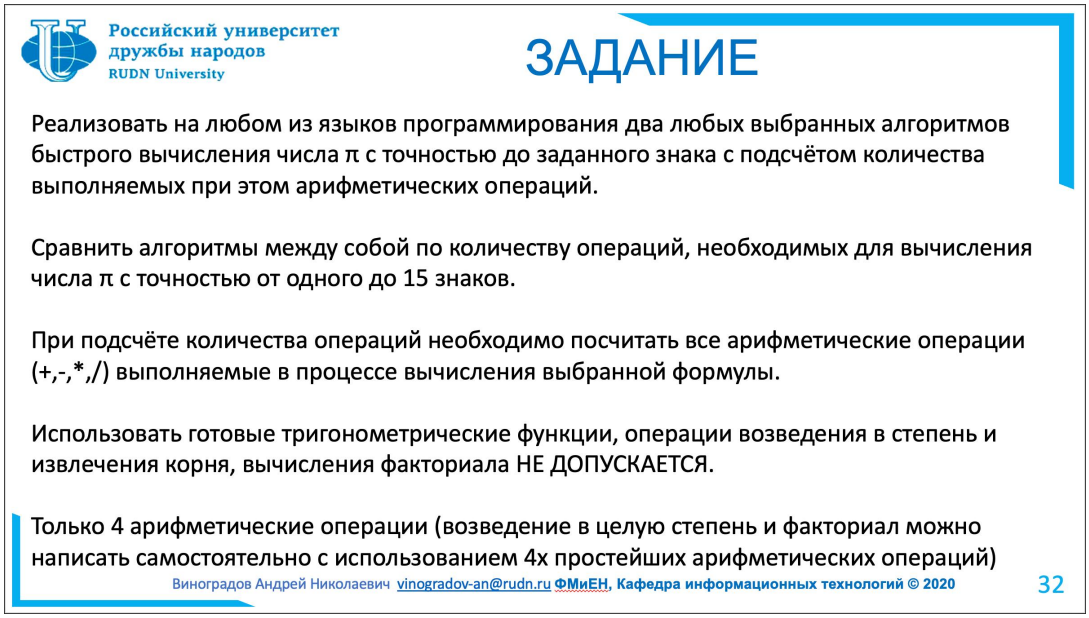
Лабораторная работа №1

Задание:



Выполнение:

1. Подготовка:

In [73]:
```
!pwd
```

```
/Users/artyem.petrov/dev/university/4-1/it-computer-practice/lab02
```

In [74]:
```
import math
from typing import Callable
import time
```

2. Реализуем класс для подсчета кол-ва операций:

In [ ]:
```
class Counter:
    def __init__(self):
        self.ops = 0

    def add(self, count: int=1):
        self.ops += count
        return self.ops
```

3. Реализуем ряд Лейбница:

In [76]:
```python
def pi_leibniz(n_terms: int, counter: Counter):
    pi = 0.0
    for k in range(n_terms):
        term = 4 / (2*k + 1)
        if k % 2 == 0:
            pi += term
        else:
            pi -= term
        counter.add(3)
    return pi
```

4. Реализуем ряд Нилаканты:

In [77]:
```python
def pi_nilakantha(n_terms: int, counter: Counter):
    pi = 3.0
    sign = 1
    for k in range(n_terms):
        a = 2*k + 2
        b = 2*k + 3
        c = 2*k + 4
        term = 4 / (a * b * c)
        pi += sign * term
        sign *= -1
        counter.add(7)
    return pi
```

5. Создадим функцию для определения необходимого кол-ва итераций для заданной точности:

In [78]:
```python
def iterations_for_precision(algorithm: Callable[int, Counter], pre
    target_error = 10 ** (-precision)
    prev_pi = 0
    for n in range(1, 10**6):
        counter = Counter()
        pi = algorithm(n, counter)
        if abs(pi - prev_pi) < target_error:
            return n, counter.ops
        prev_pi = pi
    return None
```

6. Запуск:

In [80]:
```python
print("Точность  | Итерации Лейбниц | Операции Лейбниц | Время на в
for prec in range(1, 16):
    start_time = time.perf_counter()
    n_leib, ops_leib = iterations_for_precision(pi_leibniz, prec)
    time_leib = time.perf_counter() - start_time

    start_time = time.perf_counter()
    n_nil, ops_nil = iterations_for_precision(pi_nilakantha, prec)
    time_nil = time.perf_counter() - start_time
```

```
    print(f"{prec:^9} | {n_leib:^16} | {ops_leib:^16} | {time_leib:
```

```
Точность  | Итерации Лейбниц | Операции Лейбниц | Время на выполнени
е Лейбниц | Итерации Нилаканта | Операции Нилаканта | Время на выпол
нение Нилаканта
    1     |        21        |        63        | 0.0001830420005717
3148  сек |         2          |         14         |   9.4580009317
72403e-06    сек
    2     |       201        |       603        |  0.007251958000779
268   сек |         4          |         28         |   1.4000001101
521775e-05   сек
    3     |       2001       |       6003       |   0.18824325000059
616   сек |         8          |         56         |    9.834000593
40056e-06    сек
    4     |      20001       |      60003       |    16.632389415999
57    сек |         17         |        119         |   2.5790999643
50462e-05    сек
```

```
---------------------------------------------------------------------
-------
KeyboardInterrupt                         Traceback (most recent cal
l last)
Cell In[80], line 4
      2 for prec in range(1, 16):
      3     start_time = time.perf_counter()
----> 4     n_leib, ops_leib = iterations_for_precision(pi_leibniz,
prec)
      5     time_leib = time.perf_counter() - start_time
      7     start_time = time.perf_counter()

Cell In[78], line 6, in iterations_for_precision(algorithm, precisio
n)
      4 for n in range(1, 10**6):
      5     counter = Counter()
----> 6     pi = algorithm(n, counter)
      7     if abs(pi - prev_pi) < target_error:
      8         return n, counter.ops

Cell In[76], line 9, in pi_leibniz(n_terms, counter)
      7     else:
      8         pi -= term
----> 9     counter.add(3)
     10 return pi

Cell In[75], line 5, in Counter.add(self, count)
      2 def __init__(self):
      3     self.ops = 0
----> 5 def add(self, count=1):
      6     self.ops += count
      7     return self.ops

KeyboardInterrupt:
```

Промежуточный вывод:

Моя реализация формулы Лейбница для точности > 4 знаков после запятой занимает очень много времени (10 минут для пятой итерации), поэтому реализуем другую функцию

Точность	Итерации Лейбниц	Операции Лейбниц	Время на выполнение Лейбниц	Итерации Нилаканта	Операции Нилаканта	Время на выполнение Нилаканта
1	21	63	0.0001465840323241 95 сек	2	14	1.0542018571868539e-05 сек
2	201	603	0.007557416975032538 сек	4	28	1.2584001524373889e-05 сек
3	2001	6003	0.19200208398979157 сек	8	56	8.250004611909 39e-06 сек
4	20001	60003	16.154134583019186 сек	17	119	2.4249980924651027e-05 сек

7. Реализуем формулу Мэчина:

7.1. Реализуем функцию для возведения в степень:

In [81]:
```
def power(x: float, n: int, counter: Counter) -> float:
    result = 1.0
    for _ in range(n):
        result *= x
        counter.add(1)
    return result
```

7.2. Реализуем функцю для вычисления арктангенса через ряд Тейлора:

In [82]:
```
def arctan_taylor(x: float, n_terms: int, counter: Counter):
    result = 0.0
    sign = 1
    for k in range(n_terms):
        exponent = 2*k + 1
        term = power(x, exponent, counter) / exponent
        result += sign * term
        sign *= -1
        counter.add(2)
    return result
```

7.3. Реализуем формулу Мэчина:

In [83]:
```
def pi_machin(n_terms: int, counter: Counter):
    # Вычисляем arctan(1/5)
    counter_atan1 = Counter()
    atan1 = arctan_taylor(1/5, n_terms, counter_atan1)

    # Вычисляем arctan(1/239)
    counter_atan2 = Counter()
    atan2 = arctan_taylor(1/239, n_terms, counter_atan2)

    pi = 4 * (4 * atan1 - atan2)

    # Суммируем операции
    counter.ops = counter_atan1.ops + counter_atan2.ops + 3  # 3 оп

    return pi
```

8. Повторное выполнение

In [84]:
```
print("Точность  | Итерации Мэчина   | Операции Мэчина  | Время на
for prec in range(1, 16):
    start_time = time.perf_counter()
    n_mac, ops_mac = iterations_for_precision(pi_machin, prec)
    time_mac = time.perf_counter() - start_time

    start_time = time.perf_counter()
    n_nil, ops_nil = iterations_for_precision(pi_nilakantha, prec)
    time_nil = time.perf_counter() - start_time

    print(f"{prec:^9} | {n_mac:^17} | {ops_mac:^16} | {time_mac:^23
```

Точность	Итерации Мэчина	Операции Мэчина	Время на выполнение Мэчина	Итерации Нилаканта	Операции Нилаканта	Время на выполнение Нилаканта
1	2	19	2.254199898743536e-05 сек	2	14	3.875000402331352e-06 сек
2	3	33	1.183299900731071e-05 сек	4	28	4.958001227350905 5e-06 сек
3	4	51	1.645799966354388e-05 сек	8	56	1.258300108020193 9e-05 сек
4	4	51	1.645799966354388e-05 сек	17	119	4.525000076682772 5e-05 сек
5	5	73	2.408300133538432e-05 сек	37	259	0.000189583000974 3804 сек
6	5	73	2.574999962234869e-05 сек	79	553	0.000844666999910 2596 сек
7	6	99	3.850000030070077e-05 сек	171	1197	0.00410604200078 5237 сек
8	7	129	5.80830001126742e-05 сек	368	2576	0.02029333300015 4234 сек
9	8	163	2.741700154729187e-05 сек	794	5558	0.03549320800084 388 сек
10	8	163	2.88749997707782e-05 сек	1710	11970	0.17011945900048 886 сек
11	9	201	3.46659999195253e-05 сек	3684	25788	0.8402602080004 726 сек
12	10	243	4.49159997515380e-05 сек	7937	55559	3.8700223330015 433 сек
13	10	243	4.46250014647375e-05 сек	17092	119644	18.7339839999986 сек
14	11	289	5.28750006196787e-05 сек	36850	257950	85.9806545829997 сек
15	12	339	6.10839997534640e-05 сек	76651	536557	387.5092577920004 сек

9. Результат: