Дз2

№1

x(max)=01111111011111111111111111111111 x(10)=3.4*10^38

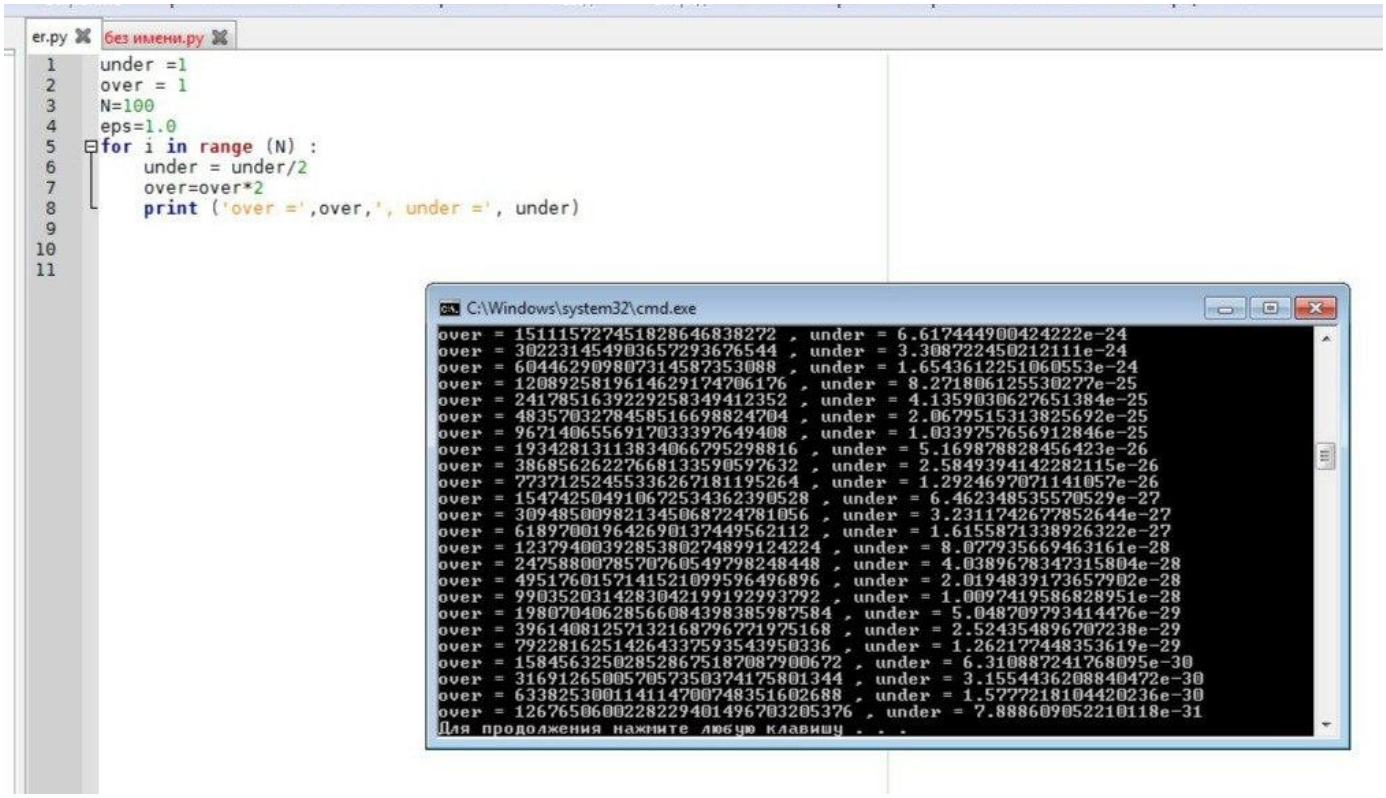
№2

x(min)=00000000000000000000000000000001 x(10)=3.4*10^-45

№3

```
<under =1
over = 1
N=100
eps=1.0
for i in range (N) :
    under = under/2
    over=over*2
    print ('over =',over,', under =', under)>
```

over = 1267650600228229401496703205376 , under = 7.888609052210118e-31



№4

```
<N=100
eps=1.0
for i in range (N) :
  eps = eps /2
  one_Plus_eps =1.0 +eps
  print ('eps =',eps,', one + eps =', one_Plus_eps)>
```

eps= 2.220446049250313e-16 , one + eps = 1.0000000000000002

er.py | без имени.py | rt.py

```
1  N=100
2  eps=1.0
3  for i in range (N) :
4      eps = eps /2
5      one_Plus_eps =1.0 +eps
6      print ('eps =',eps,', one + eps =', one_Plus_eps)
7
8
9
```

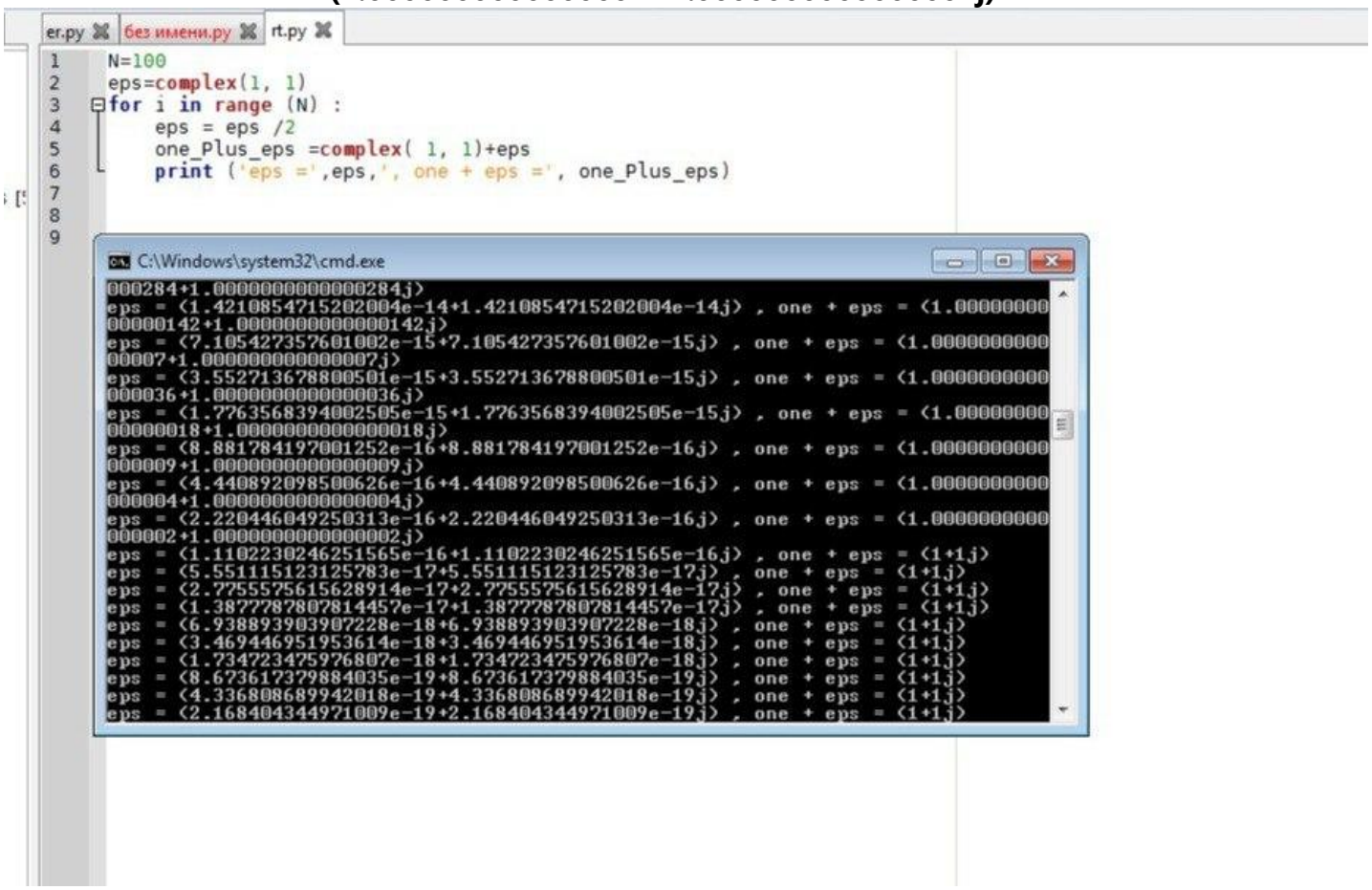
C:\Windows\system32\cmd.exe

```
eps = 5.8207660913467 41e-11 , one + eps = 1.0000000000582077
eps = 2.9103830456733704e-11 , one + eps = 1.0000000000291038
eps = 1.4551915228366852e-11 , one + eps = 1.0000000000014552
eps = 7.275957614183426e-12 , one + eps = 1.0000000000007276
eps = 3.637978807091713e-12 , one + eps = 1.0000000000003638
eps = 1.8189894035458565e-12 , one + eps = 1.0000000000001819
eps = 9.094947017729282e-13 , one + eps = 1.0000000000009095
eps = 4.547473508864641e-13 , one + eps = 1.0000000000004547
eps = 2.2737367544323206e-13 , one + eps = 1.0000000000002274
eps = 1.1368683772161603e-13 , one + eps = 1.0000000000001137
eps = 5.684341886080802e-14 , one + eps = 1.0000000000000568
eps = 2.842170943040401e-14 , one + eps = 1.0000000000000284
eps = 1.4210854715202004e-14 , one + eps = 1.0000000000000142
eps = 7.105427357601002e-15 , one + eps = 1.0000000000000007
eps = 3.552713678800501e-15 , one + eps = 1.0000000000000036
eps = 1.7763568394002505e-15 , one + eps = 1.0000000000000018
eps = 8.881784197001252e-16 , one + eps = 1.0000000000000009
eps = 4.440892098500626e-16 , one + eps = 1.0000000000000004
eps = 2.220446049250313e-16 , one + eps = 1.0000000000000002
eps = 1.1102230246251565e-16 , one + eps = 1.0
eps = 5.551115123125783e-17 , one + eps = 1.0
eps = 2.7755575615628914e-17 , one + eps = 1.0
eps = 1.3877787807814457e-17 , one + eps = 1.0
eps = 6.938893903907228e-18 , one + eps = 1.0
eps = 3.469446951953614e-18 , one + eps = 1.0
```

№5

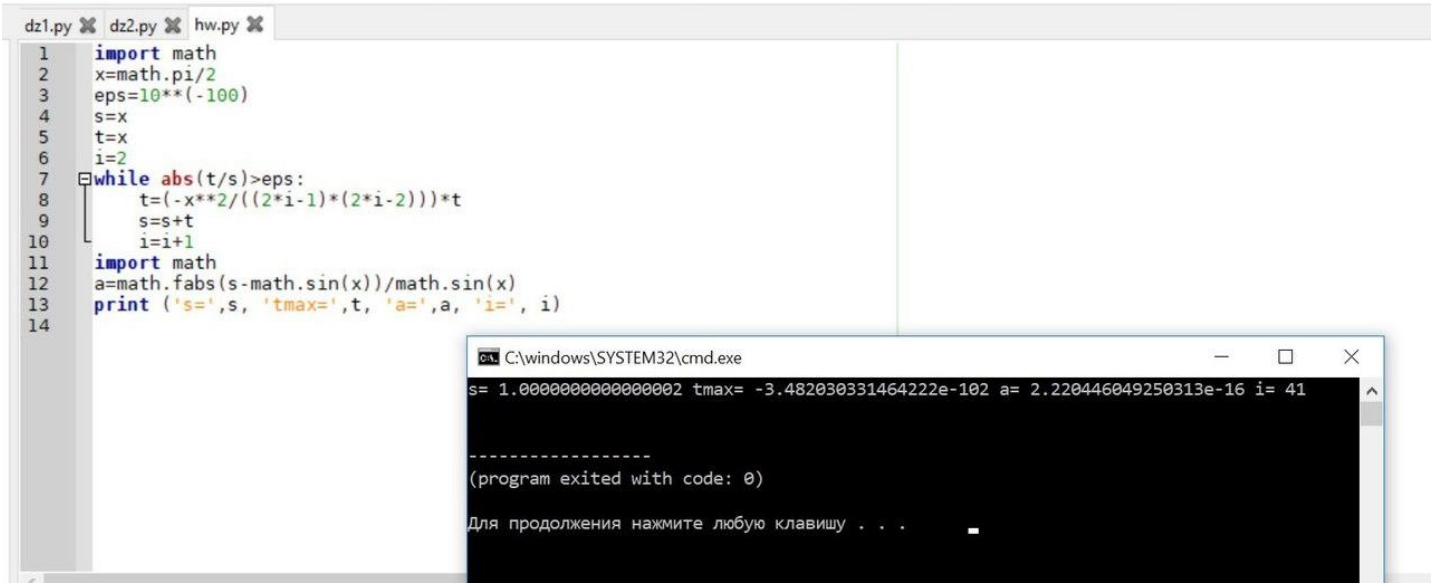
```
<N=100
eps=complex(1, 1)
for i in range (N) :
        eps = eps /2
        one_Plus_eps =complex( 1, 1)+eps
        print ('eps =',eps,', one + eps =', one_Plus_eps)
>
```

eps = (2.220446049250313e-16+2.220446049250313e-16j) , one + eps = (1.0000000000000002+1.0000000000000002j)



№6

```
<import math
x=math.pi/2
eps=10**(-5)
s=x
t=x
i=2
while abs(t/s)>eps:
        t=(-x**2/((2*i-1)*(2*i-2)))*t
        s=s+t
        i=i+1
import math
a=math.fabs(s-math.sin(x))/math.sin(x)
print ('s=',s, 'tmax=',t, 'a=',a, 'i=', i)
>
```



sum	t max	ошибка	степень eps
0.9999999999939768	-6.688035109811466e-106.023181953196399e-12	6.023181953196399e-12	-8
1.0000000000000437	6.066935731106194e-12	4.374278717023117e-14	-10
1.0000000000000002	3.4140529700331598e-31	2.220446049250313e-16	-30
1.0000000000000002	6.382145039734999e-51	2.220446049250313e-16	-50

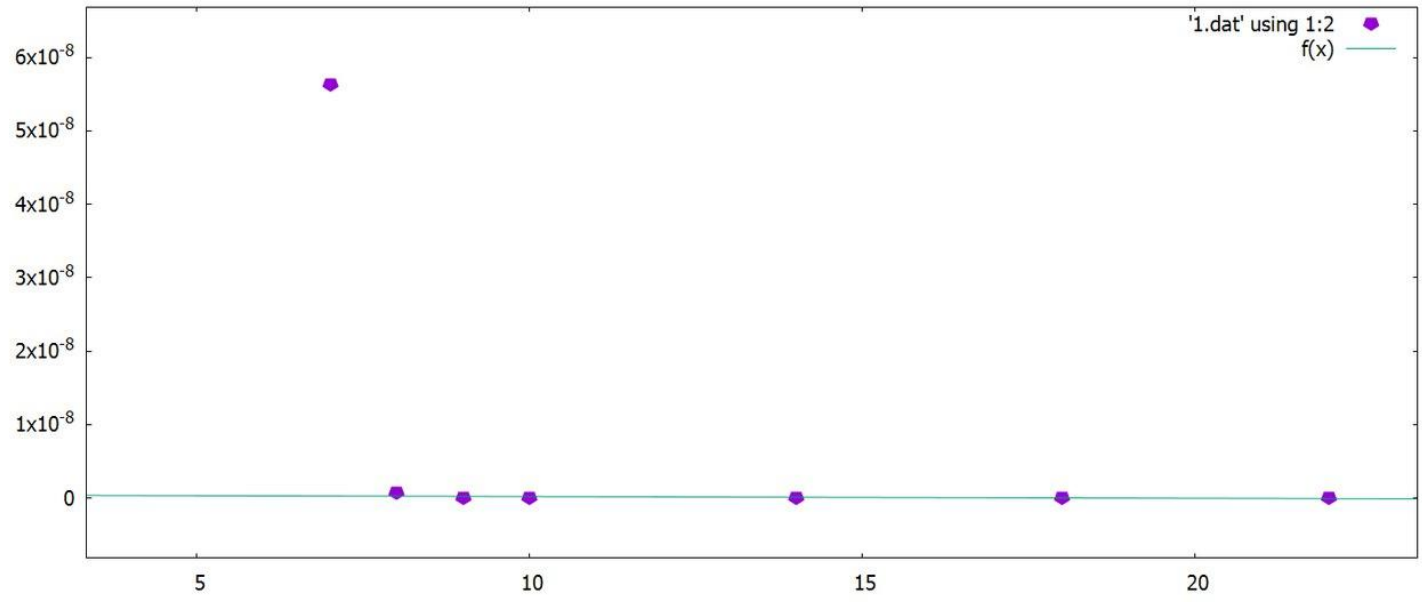
- Из данной таблицы мы видим, что при достаточно малых значениях x алгоритм сходится к 1, Значение тем ближе к 0, чем меньше точность.
- После того, как наша точность превысила машинную, значение суммы ряда остается постоянным(при x=pi/2 =1.0000000000000002)
- Без использования формулы приведения (sin(x+2*n*pi)=sin(x)) существует диапазон x, для которых алгоритм сходится, но не к 0, а к 1. Что видно из данной таблицы.

sum	делитель pi (pi/k)
0.9980267284282714	50
0.9995065603657319	100
0.9999950652018581	1000

- Припримененении формул приведения алгоритм также сходится при малым x.
- "Плохая" версия алгоритма

```
<import math
x=math.pi*2
eps=10**(-50)
s=x
t=x
i=2
while abs(t/s)>eps:
        import math
        t=(-1)**(i-1)*(x**(2*i-1))/math.factorial((2*i-1))*t
        s=s+t
        i=i+1
import math
a=math.fabs(s-math.sin(x))/math.sin(x)
print ('s=',s, 'tmax=',t, 'a=',a, 'i=', i)
>
```

"Хороший"	"Плохой"
0.019757158783946512	0.019758418960677735
0.15643446504023087	0.1569781647954669
4.3878919606925197e-16	-1782193851.3401673



- Из данного графика мы видим, что при больших точностях функция ошибок от количества слагаемых выходит на константу

№7

```
<import math
import sys

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

```
if a == 0:
        print ('не является квадратным')
        sys.exit()

b = float(input("b = "))
c = float(input("c = "))

D = b**2- 4*a*c
if D == 0:
        m=-b/2*a
        print ('x =', m)
        sys.exit()
if D > 0:
        n=(-b + math.sqrt(D))/ 2*a
        m=(-b - math.sqrt(D))/ 2*a
        print ('x1 =',n,'x2 =',m)
else:
        D<0
        D=complex(D**0.5)
        n=(-b+complex((D**0.5)))/2*a
        m=(-b-complex((D**0.5)))/2*a
        print ('x1 =',n,'x2 =',m)

>
```

dz1.py | dz2.py | hw.py | kvadr.py | kvad2.py

```
3
4   a= float(input("a = "))
5   if a == 0:
6       print ('не является квадратным')
7       sys.exit()
8
9   b = float(input("b = "))
10  c = float(input("c = "))
11
12  D = b**2- 4*a*c
13  if D == 0:
14      m=-b/2*a
15      print ('x =', m)
16      sys.exit()
17  if D > 0:
18      n=(-b + math.sqrt(D))/ 2*a
19      m=(-b - math.sqrt(D))/ 2*a
20      print ('x1 =',n,'x2 =',m)
21  else:
22      D<0
23      D=complex(D**0.5)
24      n=(-b+complex((D**0.5)))/2*a
25      m=(-b-complex((D**0.5)))/2*a
26      print ('x1 =',n,'x2 =',m)
27
```