

# **Лабораторна робота №6. Організація підпрограм**

Міністерство освіти і науки України  
Національний технічний університет України «Київський  
політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації  
і управління

Звіт

з лабораторної роботи № 6 з дисципліни  
«Основи програмування»

«Дослідження складних циклічних алгоритмів»

Варіант \_\_4\_\_

Виконав студент \_\_\_\_\_ Берлінський Ярослав Владленович \_\_\_\_\_  
(шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_  
( прізвище, ім'я, по батькові)

Київ 2020

**Назва роботи:** організація підпрограм.

**Варіант:** 4

**Умова задачі:**

4. Для заданого цілого  $x$ , використовуючи розкладання функції  $e^{-x}$  в ряд Тейлора

$$e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots,$$

обчислити із заданою точністю  $\varepsilon$  значення

$$y = \begin{cases} e^{-x} + e^{-2x}, & 0 \leq x \leq 2 \\ \frac{1}{e^{-(x+5)}} - e^{-(x+2)}, & x > 2 \end{cases}.$$

**Постановка задачі.** З клавіатури вводяться  $x$  – змінна, для якої обчислюється вираз – та точність  $\varepsilon$ , до якої безпосередньо вираз обчислюється.

Результатом програми буде значення  $y$ , яке залежить від від умови, описаної вище.

**Розв'язок:**

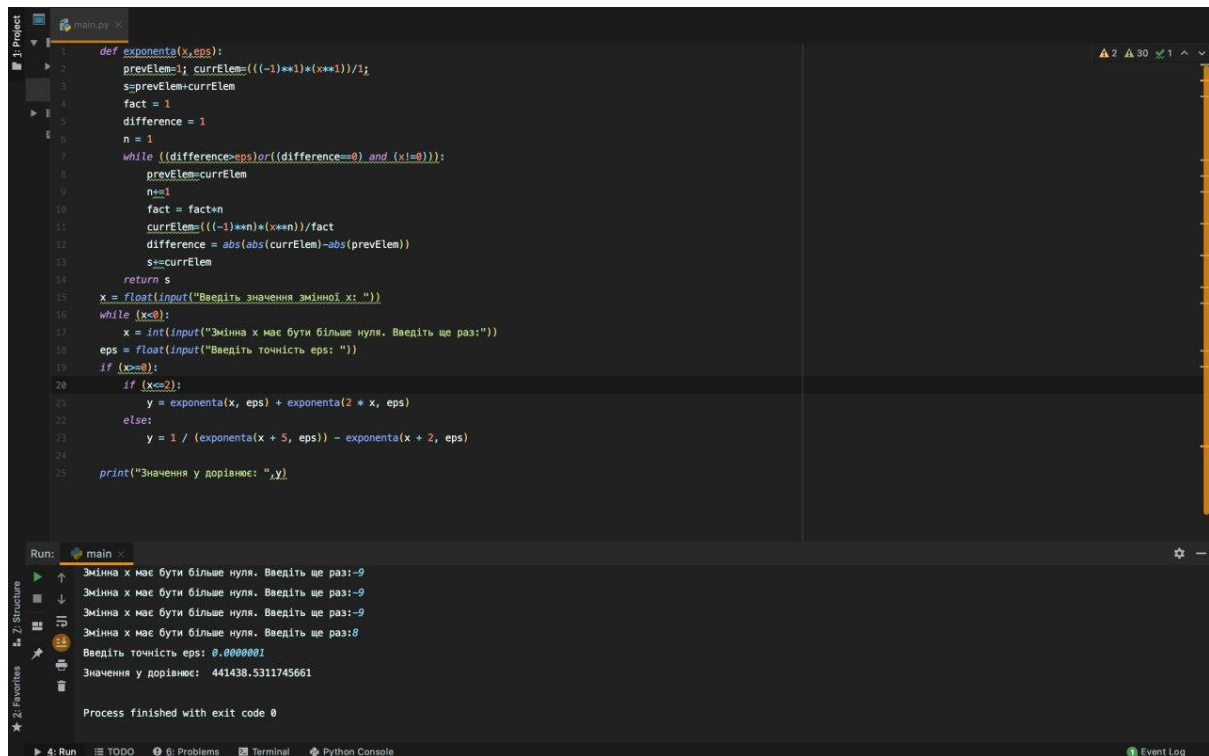
## 1. C++:

```
LAB6/main.cpp at main · berlinium1/LAB6 · GitHub
Тішин В.В. - Дискретная математика в приме... Домашнее задание № 9.pdf - Google Диск ПЗ_9.pdf - Google Диск Meet - tpf-vmzj-hga

8 #include <iostream>
9 #include <iomanip>
10 #include <math.h>
11 using namespace std;
12 long double exponenta(long double, long double);
13 int main(){
14     long double x, eps;
15     long double y;
16     cout<<"Введіть значення змінної x: ";
17     cin>>x;
18     cout<<"Введіть точність eps: ";
19     cin>>eps;
20     while (x<0){
21         printf("\nЗмінна x має бути більше нуля. Введіть ще раз: ");
22         cin>>x;
23     }
24     if (x<=2){
25         y=exponenta(x, eps) + exponenta(2*x, eps);
26     }
27     else y = 1/(exponenta(x+5, eps))-exponenta(x+2, eps);
28     cout<<"\nЗначення y дорівнює: "<<y<<endl;
29 }
30
31
32
33 long double exponenta(long double x, long double eps){
34     long double diff;
35     int n;
36     long double s = 1;
37     long double elem0 = 1;
38     long double elem1 = (pow(-1,1)*pow(x,1))/1;
39     s = elem0 + elem1;
40     long int fact = 1;
41     diff = 1;
42     n=1;
43     while ((diff>eps) || ((diff==0)&&(x!=0))){
44         elem0=elem1;
45         n++;
46         fact = fact*n;
47         elem1 = (pow(-1,n)*pow(x,n))/fact;
48         diff = fabs(fabs(elem1)-fabs(elem0));
49         s+=elem1;
```

```
48         diff = fabs(fabs(elem1)-fabs(elem0));
49         s=s+elem1;
50     }
51     return s;
52 }
53
```

## 2. Python:



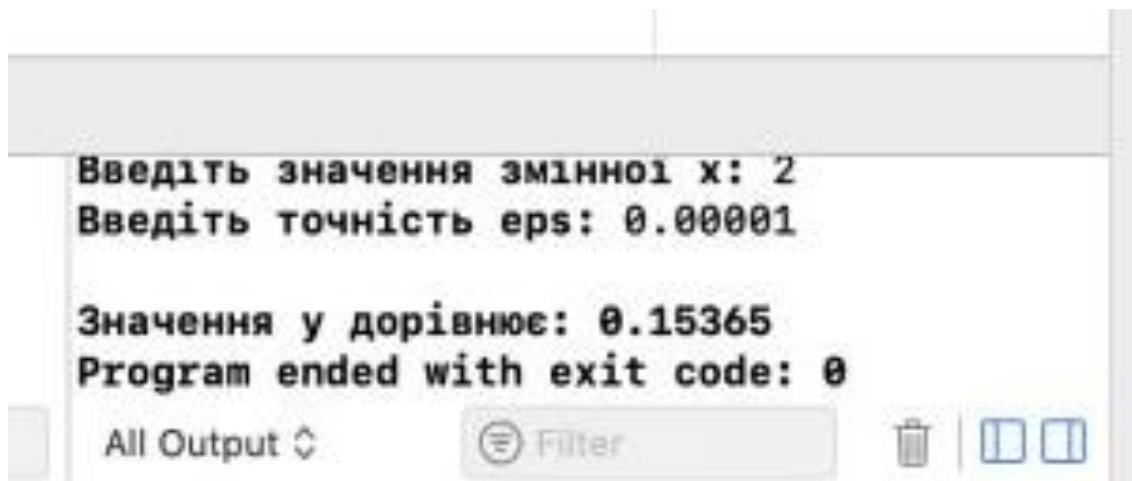
```
1 def exponenta(x, eps):
2     prevElem = 1; currElem = (((-1)**1)*(x**1))/1;
3     s = prevElem + currElem
4     fact = 1
5     difference = 1
6     n = 1
7     while ((difference > eps) or ((difference == 0) and (x != 0))):
8         prevElem = currElem
9         n = n + 1
10        fact = fact * n
11        currElem = (((-1)**n)*(x**n))/fact
12        difference = abs(currElem) - abs(prevElem)
13        s = currElem + currElem
14    return s
15    x = float(input("Введіть значення змінної x: "))
16    while (x <= 0):
17        x = int(input("Змінна x має бути більше нуля. Введіть ще раз: "))
18    eps = float(input("Введіть точність eps: "))
19    if (x == 0):
20        if (x == 2):
21            y = exponenta(x, eps) + exponenta(2 * x, eps)
22        else:
23            y = 1 / (exponenta(x + 5, eps) - exponenta(x + 2, eps))
24    print("Значення у дорівнює: ", y)
```

Run: main

Змінна x має бути більше нуля. Введіть ще раз: -9  
Змінна x має бути більше нуля. Введіть ще раз: -9  
Змінна x має бути більше нуля. Введіть ще раз: -9  
Змінна x має бути більше нуля. Введіть ще раз: 8  
Введіть точність eps: 0.000001  
Значення у дорівнює: 441438.5311745661  
Process finished with exit code 0

## Тестування вхідних даних(C++):

1)  $x=2$ ,  $eps=0.00001$ :



2)  $x=0$ ,  $\text{eps}=0.01$

```
Введіть значення змінної x: 0
Введіть точність eps: 0.01

Значення у дорівнює: 2
Program ended with exit code: 0
```

3)  $x=1$ ,  $\text{eps}=0.00001$ :

```
Введіть значення змінної x: -1
Введіть точність eps: 0.00001

Змінна x має бути більше нуля. Введіть ще раз: 1

Значення у дорівнює: 0.503215
Program ended with exit code: 0
```

4)  $x=2.0001$ ,  $\text{eps}=0.00001$ :

```
Введіть значення змінної x: 2.0001
Введіть точність eps: 0.00001

Значення у дорівнює: nan
Program ended with exit code: 0
```

Як бачимо, значення  $y = \text{nan}$ . Проаналізувавши роботу обчислення виразу, доходимо до висновку, що для випадку(C++), коли ітерація дійшла до  $21!+$ , то трапляється семантична помилка через переповнення розрядної сітки.

```
20) Сума дорівнює 0.00918604
факторіал(1)=2432902008176640000
під степенем(1)= 7.98151e+16
різниця= 0.0986774
факторіал(2)= -4249290049419214848
під степенем(2)= 5.58713e+17
елемент(1)= 0.0328065
елемент(2)= 0.131484
```

```
21) Сума дорівнює 0.14067
факторіал(1)=-4249290049419214848
під степенем(1)= 5.58713e+17
різниця= 2.9957
факторіал(2)= -1250660718674968576
під степенем(2)= 3.91105e+18
елемент(1)= 0.131484
елемент(2)= -3.12719
```

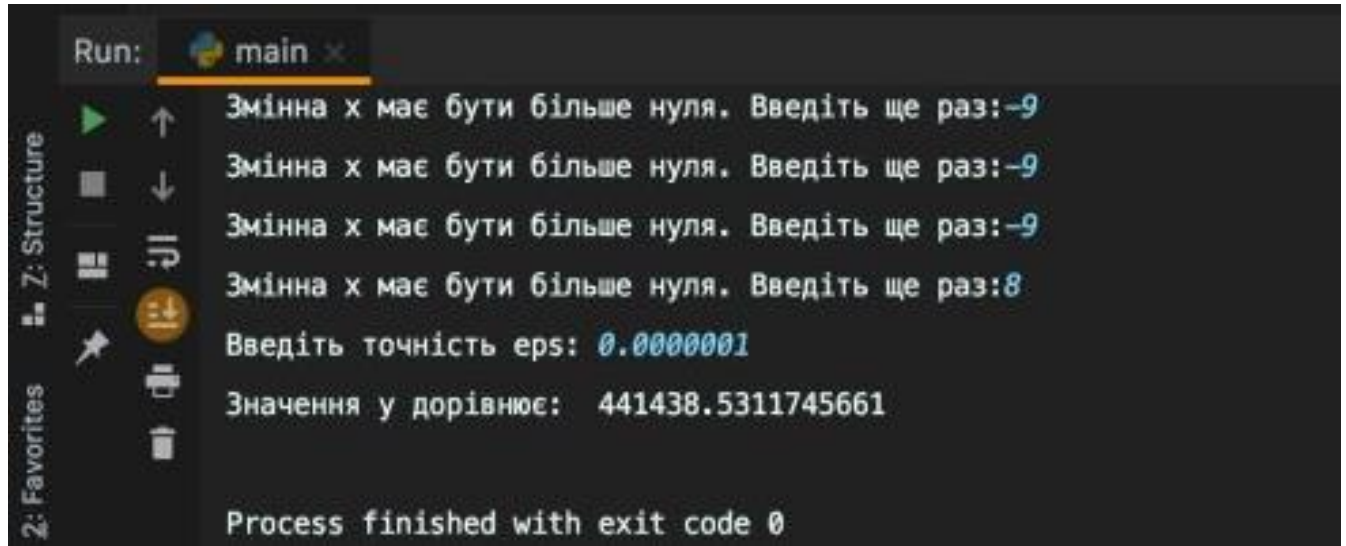
```
22) Сума дорівнює -2.98652
факторіал(1)=-1250660718674968576
```

All Output ↕

\*факторіал на 21-й ітерації стає від'ємним(семантична помилка).

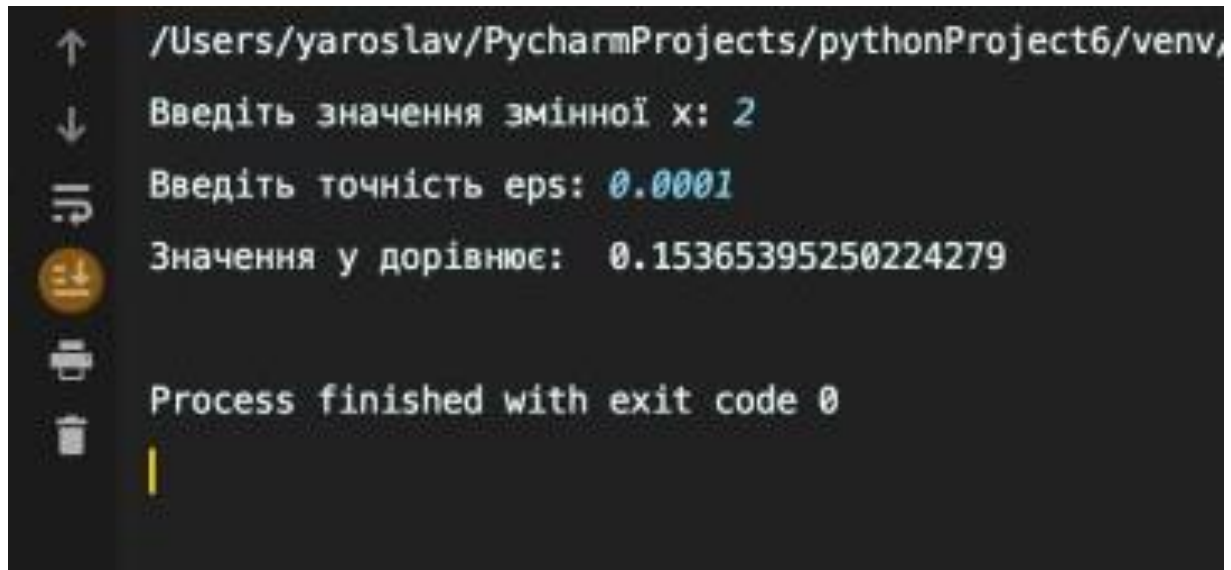
## Тестування аналогічних вхідних даних(Python):

1)  $x=8$ ,  $\text{eps}=0.0000001$ :



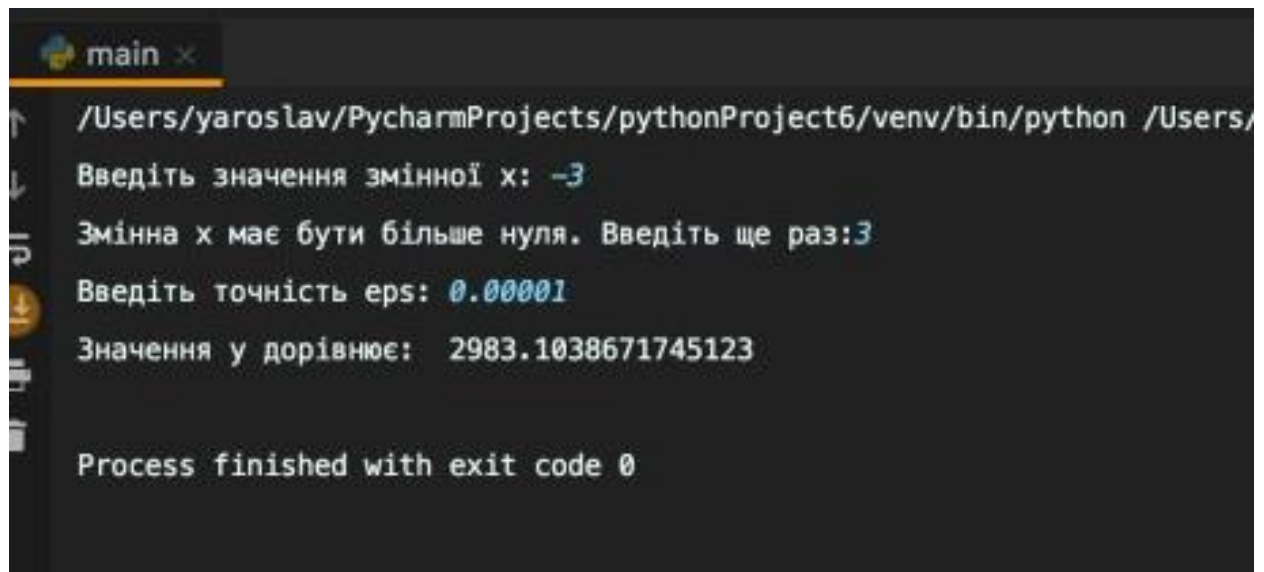
```
Run: main x
Змінна x має бути більше нуля. Введіть ще раз:-9
Змінна x має бути більше нуля. Введіть ще раз:-9
Змінна x має бути більше нуля. Введіть ще раз:-9
Змінна x має бути більше нуля. Введіть ще раз:8
Введіть точність eps: 0.0000001
Значення y дорівнює: 441438.5311745661
Process finished with exit code 0
```

2)  $x=2$ ,  $\text{eps}=0.0001$ :



```
/Users/yaroslav/PycharmProjects/pythonProject6/venv
Введіть значення змінної x: 2
Введіть точність eps: 0.0001
Значення y дорівнює: 0.15365395250224279
Process finished with exit code 0
```

3)  $x=3$ ,  $\text{eps}=0.00001$ :



```
main x
/Users/yaroslav/PycharmProjects/pythonProject6/venv/bin/python /Users/
Введіть значення змінної x: -3
Змінна x має бути більше нуля. Введіть ще раз:3
Введіть точність eps: 0.00001
Значення у дорівнює: 2983.1038671745123

Process finished with exit code 0
```



Нескладно перевірити вірність результатів. Інші дані можна перевірити самостійно, перейшовши безпосередньо до коду програми:

[GitHub](#)

**Висновок:** отже, програма розрахунку виразу з використанням рядів Тейлора із заданою точністю успішно виконана. Нескладно помітити, що для точних розрахунків є активним використання довгої арифметики, чим C++ не може забезпечити користувача без зайвих махінацій(для умови  $x > 2$  навіть для  $x = 2.0001$  спрацьовують семантичні помилки). Тож для вирішення заданого виразу більше підійде Python.

Сам алгоритм на двох мовах був реалізований за допомогою використання підпрограм для підрахунку кожного з елементів по ряду Тейлора.