Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної і теплової енергетики

Кафедра цифрових технологій в енергетиці

**Проектування та аналіз обчислювальних алгоритмів**

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ № 2

«Рекурсивний виклик функції»

*(Тема роботи)*

Варіант № 16

Виконав: студент 2-го курсу

гр. ТР-12

Каркушевський В.Л.

*(П.І.Б.)*

Оцінка \_\_\_\_\_\_\_\_\_ Перевірив: Крячок О.С.

Дата « 16 » жовтня 2022 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(П.І.Б., підпис)*

Київ – 2022

**Завдання**

Ознайомитись з рекурсивним викликом функції. Розробити алгоритмом розрахунку значення функції за її розкладенням у ряд за умови отримання результату з заданою точністю. Врахувати діапазон дозволених значень для змінної x.

У якості індивідуального завдання необхідно написати програмний код, що реалізує алгоритм розрахунку значень функцій за їх розкладенням в ряд із заданою користувачем точністю. Використати математичні моделі геометричних фігур з Додатку В-2. Завдання обрати згідно свого варіанта.

РЕЗУЛЬТАТ РОБОТИ ПОТРІБНО:

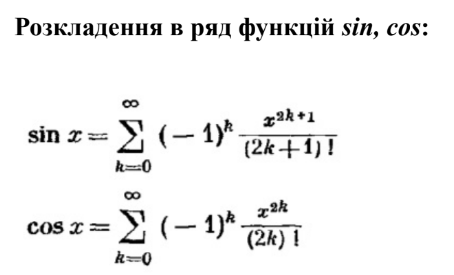
1. Роздрукувати (вивести на екран) точне значення функції, її приблизне значення та точність розрахунку.

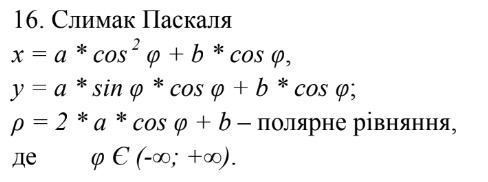
2. Відкритий для редагування програмний код розмістити на сайті

<https://replit.com/> (посилання через кнопку «+ Invite»).

3. Звіт до комп’ютерного практикуму No 2 додати в свій Клас на ресурсі <https://classroom.google.com/>.

Індивідуальне завдання





**Теоретична частина**

*“Щоб зрозуміти, що таке рекурсія, необхідно зрозуміти, що таке рекурсія.”*

**Рекурсивні функції** – це функції, які викликають самі себе.

**Є два важливі для розуміння рекурсії визначення:**

* **Базис рекурсії** – умова виходу з блоку рекурсивних викликів – базисне розв'язання задачі за умов, коли немає необхідності викликати рекурсію.
* **Крок рекурсії** – виклик функцією себе при зміні параметрів.

Класичний приклад застосування рекурсивної функції – обчислення факторіалу від числа. Розберемо два варіанта обчислення факторіала: ітераційний та рекурсивний.

1) Ітераційний:

public static long factorialItter(long n) {

long result;

for (result = 1 ; n > 1 ; result \*=(n--));

return result;

}

2) Рекурсивний:

public static long factorial(long n) {

if (n < 0){

throw new NullPointerException("factorial only positive numbers");

}

if (n == 0) return 1;

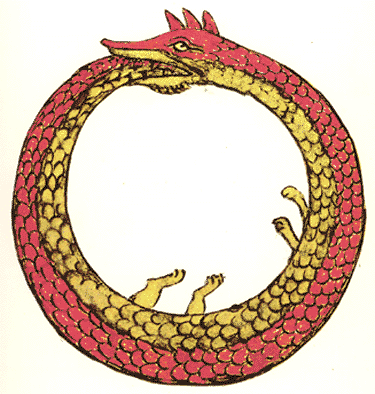
return n \* (factorial(n - 1));

}

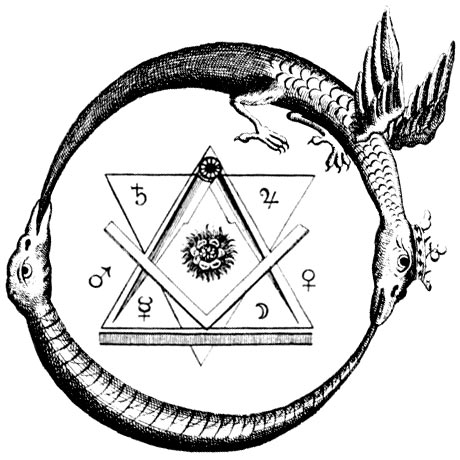
У чому перевага одного методу перед іншим? Здається, що великої різниці немає, але насправді безліч рекурсивних викликів негативно позначиться на продуктивності та споживаної пам'яті: стек викликів – практично неконтрольований ресурс і за різних умов виклику однієї і тієї ж рекурсивної функції, ми можемо отримати або не отримати проблеми, пов'язані з цим ресурсом. Практично всі завдання, які вирішуються за допомогою ітерацій (циклів типу for-each), можна вирішити і рекурсивно. Перевага рекурсії в читання та простоті написання, про недоліки ми говорили вище: можливість «вистрілити собі в ногу» неілюзорна. Ще більш обережним треба бути при використанні так званої «складної рекурсії»: Функція A()викликає функцію B(), що викликає функціюA().Для вирішення таких завдань необхідне повне розуміння того, як працює рекурсія.

**Цікавий факт**

Просту рекурсію часто порівнюють з античним змієм Уроборос, який пожирає свій хвіст.



А складну рекурсію порівнюють з таким малюнком



**Описання програми**

У цій програмі розроблені такі методи як:

* factorial(long n) – метод який рекурсивно обчислює факторіал числа n.
* mySin(int x, double e) – метод який розкладає в ряд функцію sin з заданою точністю.
* myCos(int x, double e) – метод який розкладає в ряд функцію cos з заданою точністю.
* pascalsSlug(double a, double b, int alpha, double e) – метод який обчислює завдання згідно варіанту за допомогою власноруч написаних методів для обчислення синуса та косинуса.
* pascalsSlugMath (double a, double b, int alpha) – метод який обчислює завдання згідно варіанту за допомогою бібліотеки Math.
* accuracy(double [] firstRes, double [] secondRes) – метод який порівнює результати обрахунків метода pascalsSlug() та pascalsSlugMath().
* сomparison() – метод який виводить порівняльну таблицю власноруч написаних синусів та косинусів та з використанням бібліотеки Math.

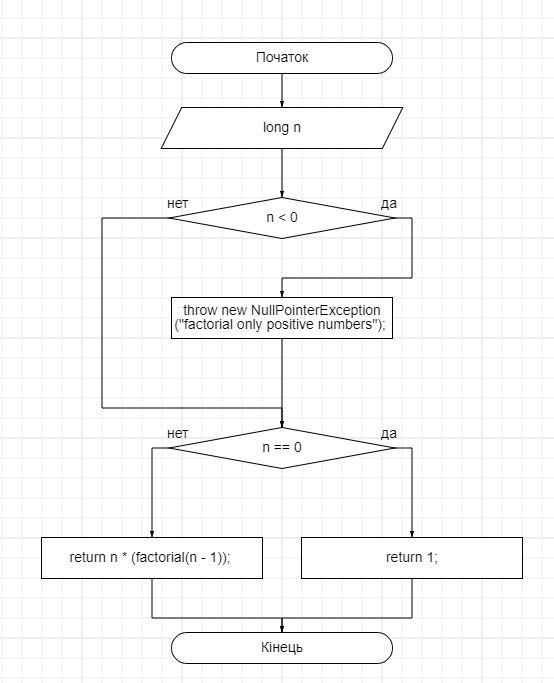


Рисунок. Блок схема обчислення факторіалу рекурсивним способом

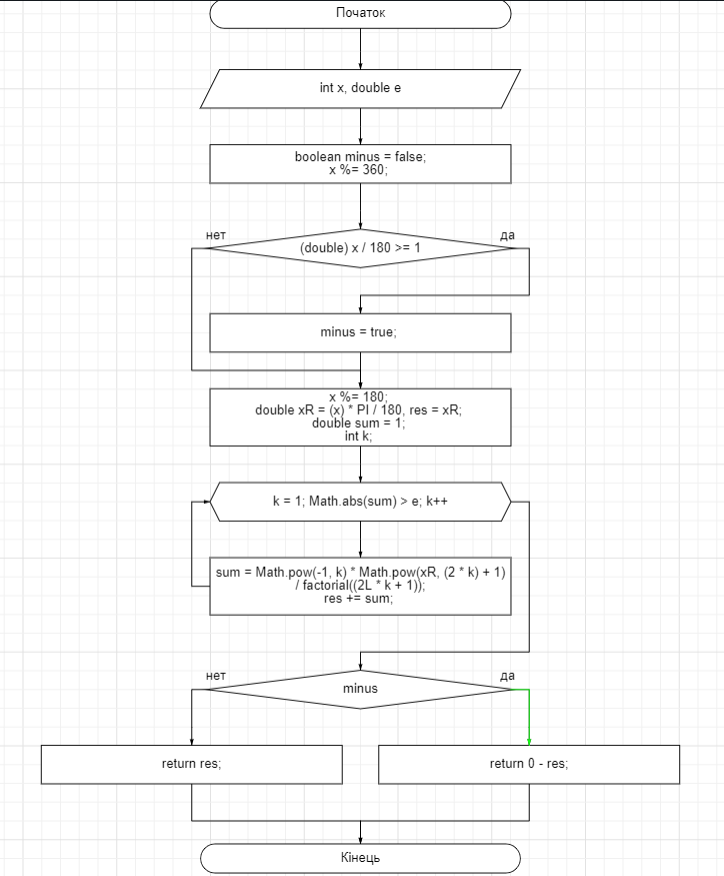


Рисунок. Блок схема обчислення синуса розкладанням в ряд

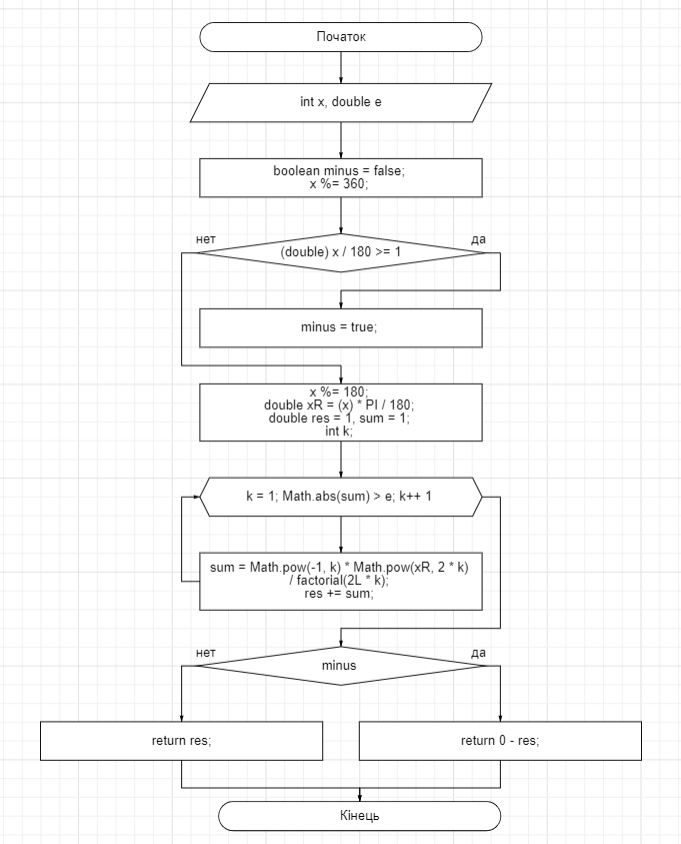
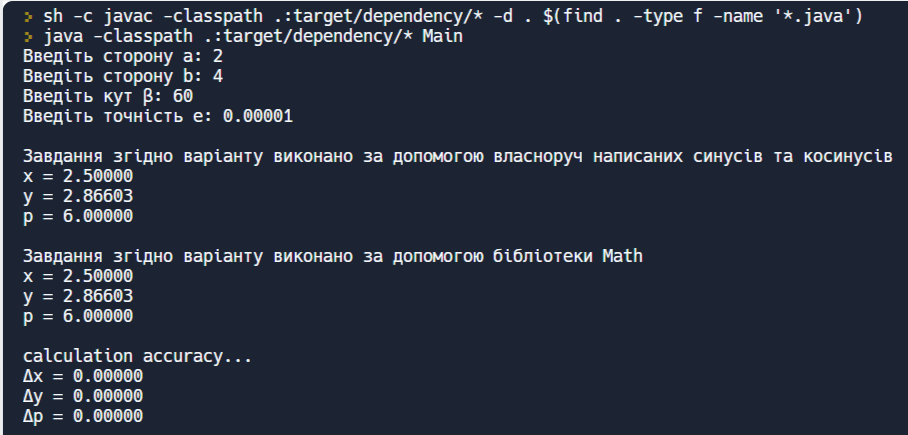
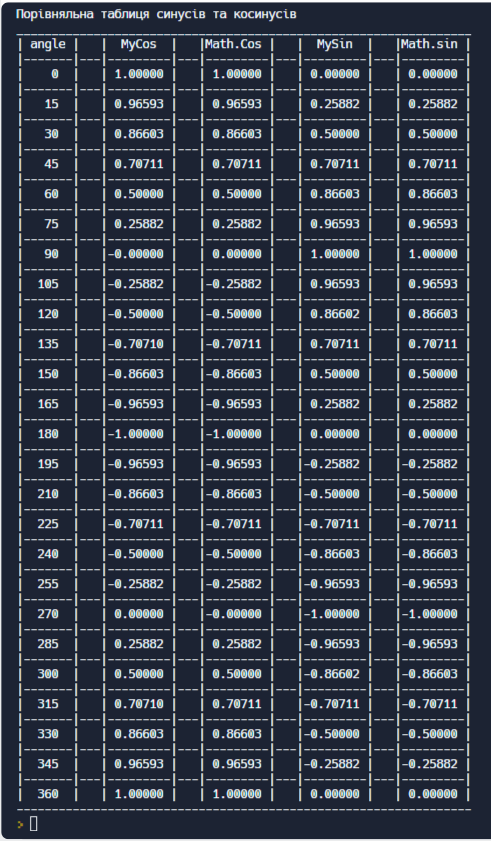


Рисунок. Блок схема обчислення косинуса розкладанням в ряд

**Результати роботи**

****

****

**Висновки**

На цій лабораторній роботі ознайомлено з рекурсивним викликом функції. Розроблено алгоритм розрахунку значення функції за її розкладенням у ряд за умови отримання результату з заданою точністю. У якості індивідуального завдання використано математичну модель: равлик Паскаля. Також розроблено блок схеми до алгоритмів та створено порівняльну таблицю синусів та косинусів написаних власноруч, та обчислених за допомогою бібліотеки Math.

**Додатки**

**Посилання на Replit.com**

<https://replit.com/@TR-12-Karkushie/Lab2#Main.java>

**Код програми**

*/\*  
 \* Лабораторна робота №2  
 \* Завдання: Ознайомитись з рекурсивним викликом функції.  
 \* Розробити алгоритмом розрахунку значення функції за її розкладенням у ряд за умови отримання результату  
 \* з заданою точністю. Врахувати діапазон дозволених значень для змінної x.  
У якості індивідуального завдання необхідно написати програмний код,  
\* що реалізує алгоритм розрахунку значень функцій за їх розкладенням в ряд із заданою користувачем точністю.  
\* Використати математичні моделі геометричних фігур з Додатку В-2. Завдання обрати згідно свого варіанта.  
  
 \* Індивідуальне завдання:  
 \* 16) Слимак Паскаля  
 \* x = a \* cos^2(φ) + b \* cos(φ)  
 \* y = a \* sin(φ) \* cos(φ) + b \* cos(φ)  
 \* p = 2 \* a \* cos(φ) + b - полярне рівняння,  
 \* де φ Є (-∞; +∞),  
 \*  
 \* Виконав студент групи ТР-12  
 \* Каркушевський Владислав  
 \* Номер варіанту: 16  
 \* \*/  
  
import* java.util.Scanner;  
  
*public class* Main {  
  
 *private static final double* ***PI*** = Math.***PI***;  
  
 *public static void* main(String[] args) {  
  
 Scanner sc = *new* Scanner(System.***in***);  
 System.***out***.print("Введіть сторону а: ");  
 *double* a = sc.nextDouble();  
  
 System.***out***.print("Введіть сторону b: ");  
 *double* b = sc.nextDouble();  
  
 System.***out***.print("Введіть кут β: ");  
 *int* beta = sc.nextInt();  
  
 System.***out***.print("Введіть точність e: ");  
 *double* e = sc.nextDouble();  
  
 *double* [] firstRes = *pascalsSlug*(a, b, beta, e);  
  
 *double* [] secondRes = *pascalsSlugMath*(a, b, beta);  
 *accuracy*(firstRes, secondRes);  
  
 *comparison*();  
 }  
  
 *public static long* factorial(*long* n) {  
 *if* (n < 0) {  
 *throw new* NullPointerException("factorial only positive numbers");  
 }  
 *if* (n == 0) *return* 1;  
 *return* n \* (*factorial*(n - 1));  
 }  
  
 *public static double* mySin(*int* x, *double* e) {  
 *boolean* minus = *false*;  
 x %= 360;  
 *if* ((*double*) x / 180 >= 1) {  
 minus = *true*;  
 }  
 x %= 180;  
 *double* xR = (x) \* ***PI*** / 180, res = xR;  
 *double* sum = 1;  
 *int* k;  
 *for* (k = 1; Math.*abs*(sum) > e; k++) {  
 sum = Math.*pow*(-1, k) \* Math.*pow*(xR, (2 \* k) + 1) / *factorial*((2L \* k + 1));  
 res += sum;  
 }  
  
 *return* minus ? 0 - res : res;  
 }  
  
 *public static double* myCos(*int* x, *double* e) {  
 *boolean* minus = *false*;  
 x %= 360;  
 *if* ((*double*) x / 180 >= 1) {  
 minus = *true*;  
 }  
 x %= 180;  
 *double* xR = (x) \* ***PI*** / 180;  
 *double* res = 1, sum = 1;  
 *int* k;  
 *for* (k = 1; Math.*abs*(sum) > e; k++) {  
 sum = Math.*pow*(-1, k) \* Math.*pow*(xR, 2 \* k) / *factorial*(2L \* k);  
 res += sum;  
 }  
  
 *return* minus ? 0 - res : res;  
 }  
  
 *public static double*[] pascalsSlug(*double* a, *double* b, *int* alpha, *double* e) {  
  
  
 System.***out***.println("\nЗавдання згідно варіанту виконано за допомогою власноруч написаних синусів та косинусів");  
 *double* [] res = *new double*[3];  
  
 res[0] = a \* Math.*pow*(*myCos*(alpha, e), 2) + b \* *myCos*(alpha, e);  
 res[1] = a \* *mySin*(alpha, e) \* *myCos*(alpha, e) + b \* *myCos*(alpha, e);  
 res[2] = 2 \* a \* *myCos*(alpha, e) + b;  
  
 System.***out***.printf("x = %.5f\n", res[0]);  
 System.***out***.printf("y = %.5f\n", res[1]);  
 System.***out***.printf("p = %.5f\n", res[2]);  
  
 *return* res;  
 }  
  
 *public static double*[] pascalsSlugMath(*double* a, *double* b, *int* alpha) {  
  
 System.***out***.println("\nЗавдання згідно варіанту виконано за допомогою бібліотеки Math");  
 *double* [] res = *new double*[3];  
  
 *double* alphaR = (alpha % 360) \* ***PI*** / 180;  
 res[0] = a \* Math.*pow*(Math.*cos*(alphaR), 2) + b \* Math.*cos*(alphaR);  
 res[1] = a \* Math.*sin*(alphaR) \* Math.*cos*(alphaR) + b \* Math.*cos*(alphaR);  
 res[2] = 2 \* a \* Math.*cos*(alphaR) + b;  
  
 System.***out***.printf("x = %.5f\n", res[0]);  
 System.***out***.printf("y = %.5f\n", res[1]);  
 System.***out***.printf("p = %.5f\n", res[2]);  
 *return* res;  
  
 }  
  
 *public static void* accuracy(*double* [] firstRes, *double* [] secondRes){  
 System.***out***.println("\ncalculation accuracy...");  
 System.***out***.printf("Δx = %.5f\n", (Math.*abs*(firstRes[0] - secondRes[0])));  
 System.***out***.printf("Δy = %.5f\n", (Math.*abs*(firstRes[1] - secondRes[1])));  
 System.***out***.printf("Δp = %.5f\n", (Math.*abs*(firstRes[2] - secondRes[2])));  
 }  
  
 *public static void* comparison() {  
 System.***out***.println("\nПорівняльна таблиця синусів та косинусів");  
 *double* e = 0.0001;  
 *int* alpha = 0;  
 System.***out***.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.***out***.println("| angle | | MyCos | |Math.Cos | | MySin | |Math.sin |");  
 System.***out***.println("|-------|---|---------|---|---------|---|---------|---|---------|");  
 *while* (alpha <= 360) {  
 System.***out***.printf("|%5d | |%8.5f | |%8.5f | |%8.5f | |%8.5f |\n", alpha, *myCos*(alpha, e), Math.*cos*((alpha % 360) \* ***PI*** / 180), *mySin*(alpha, e), Math.*sin*((alpha % 360) \* ***PI*** / 180));  
 *if* (alpha != 360){  
 System.***out***.println("|-------|---|---------|---|---------|---|---------|---|---------|");  
  
 }  
 alpha += 15;  
 }  
 System.***out***.println("-----------------------------------------------------------------");  
 }  
  
}