Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

**Лабораторна робота №** 3

по дисципліні «Теорія алгоритмів»

на тему: Рекурсія

|  |  |
| --- | --- |
| Виконав:  студент групи ІА-з91  Москаленко В.В.  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірив:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ 2020

**0.Завдання.**

1. Використовуючи команди print(x) лише при х=0..9, написати рекурсивну програму друку десяткового запису цілого позитивного числа n. *(потрібно рекурсивно ділити число на 10, передавати результат ділення в рекурсивну функцію, і потім виводити залишок від ділення на консоль) f(x) = f(x/10); print(x%10).*
2. Написати функцію *C(m,n)* обчислення біноміальних коефіцієнтів за наступною формулою:
3. **Постановка проблеми.**

У лабораторній роботі №3 ми розглядаємо тему рекурсія. Ми маємо виконати 2 завдання рекурсивним способом. Нам відомо вхідні дані, також результат у виді друку десяткового запису цілого позитивного числа або підрахунок біноміальних коефіцієнтів для другої задачі. Для 2ї задачі розв’язок буде однозначним, так як задана стандартна формула. Для 1го завдання буде використаний стандартний рекурсивний функції в середині самої себе. Надлишкових даних не має.

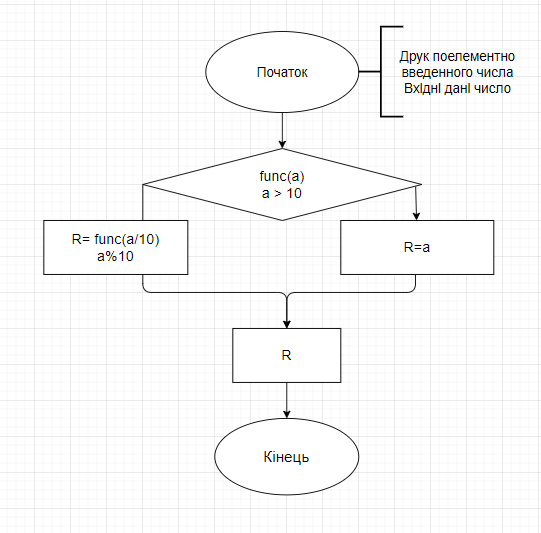
1. **Побудова моделі.**

В цій роботі ми будемо використовувати класичний механізм рекурсії. Стандартний алгоритм розвязку передбачає рекурсивний виклик функції в середині самої себе, початкову умову рекурсії, тобто базовий випадок, а також умову виходу з рекурсії.

1. **Розроблення алгоритму.**

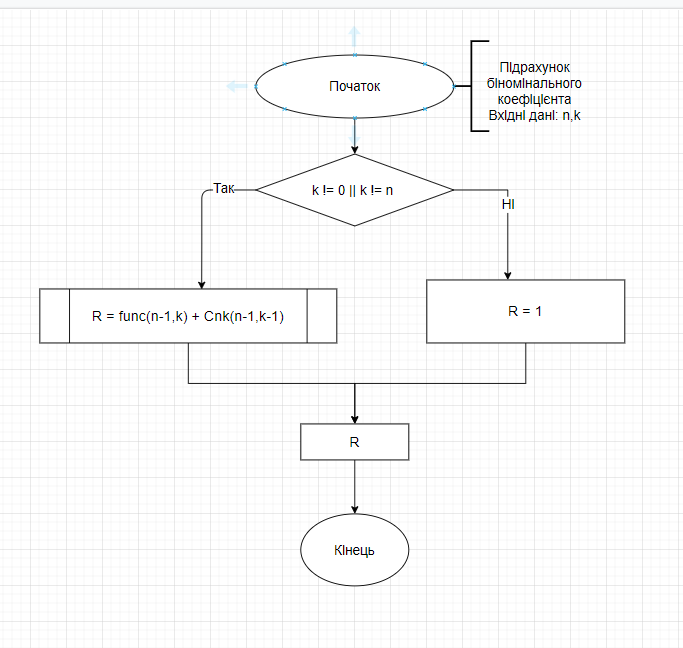
В 1му завданні базовим випадком є, коли число задане користувачем менше 10, у іншому випадку функція викликається рекурсивно. Після цього ми виконуємо операцію остачі від ділення на 10, як необхідну умову для наступної ітерації циклу, до того моменту поки не наступить базовий випадок.

**Алгоритм друку поелементно введеного числа**



У 2й задачі ми виконуємо у разі базового алгоритму при k=o та k=n, ми повертаємо одиницю, у іншому випадку ми виконуємо рекурсивний виклик функції із зменшення на 1 параметрів n та k.

**Алгоритм підрахунку біноміального коефіцієнту**



1. **Перевірка правильності алгоритму.**

Алгоритм коректно працює для всіх вхідних натуральних чисел, у іншому випадку алгоритм викидає exception. Даний алгоритм виконує умову наявності функції виходу з рекурсії і повертає результат за попередньо наданими формулами. На початку, ми виконуємо рекурсивний виклик, перевіряючи умови до того моменту поки не настане базовий випадок, тобто варіант виходу з рекурсії. Коли настає базовий випадок, ми виконуємо у випадку 2го завдання попередньо задану умову, тобто повертаємо 1. Це означає, що рекурсивний цикл буде мати вихід після розкрутки рекурсивних викликів зі стеку.

1. **Реалізація алгоритму.**

import java.io.\*;

public class BinomKoef {

public static void main(String []args){

System.out.println("Hello World");

int n = 10, k = 5;

// Scanner scan = new Scanner(System.in);

// System.out.print("Please enter the n: ");

// int n = scan.nextInt();

// System.out.print("Please enter the k: ");

// int k = scan.nextInt();

// scan.close();

int res = Cnk(n,k);

System.out.println(res);

}

public static int Cnk(int n, int k)

{

if (k == 0 || k == n) return 1;

return Cnk(n - 1, k) + Cnk(n - 1, k - 1);

}

}

import java.io.\*;

public class Print {

public static void main(String []args){

System.out.println("Hello World");

int num = 12345;

// Scanner scan = new Scanner(System.in);

// System.out.print("Please enter the number: ");

// int num = scan.nextInt();

// scan.close();

WriteNumber(num);

}

public static void WriteNumber(int a) {

if (a < 10)

System.out.print(a);

else {

WriteNumber(a / 10);

System.out.print(a % 10);

}

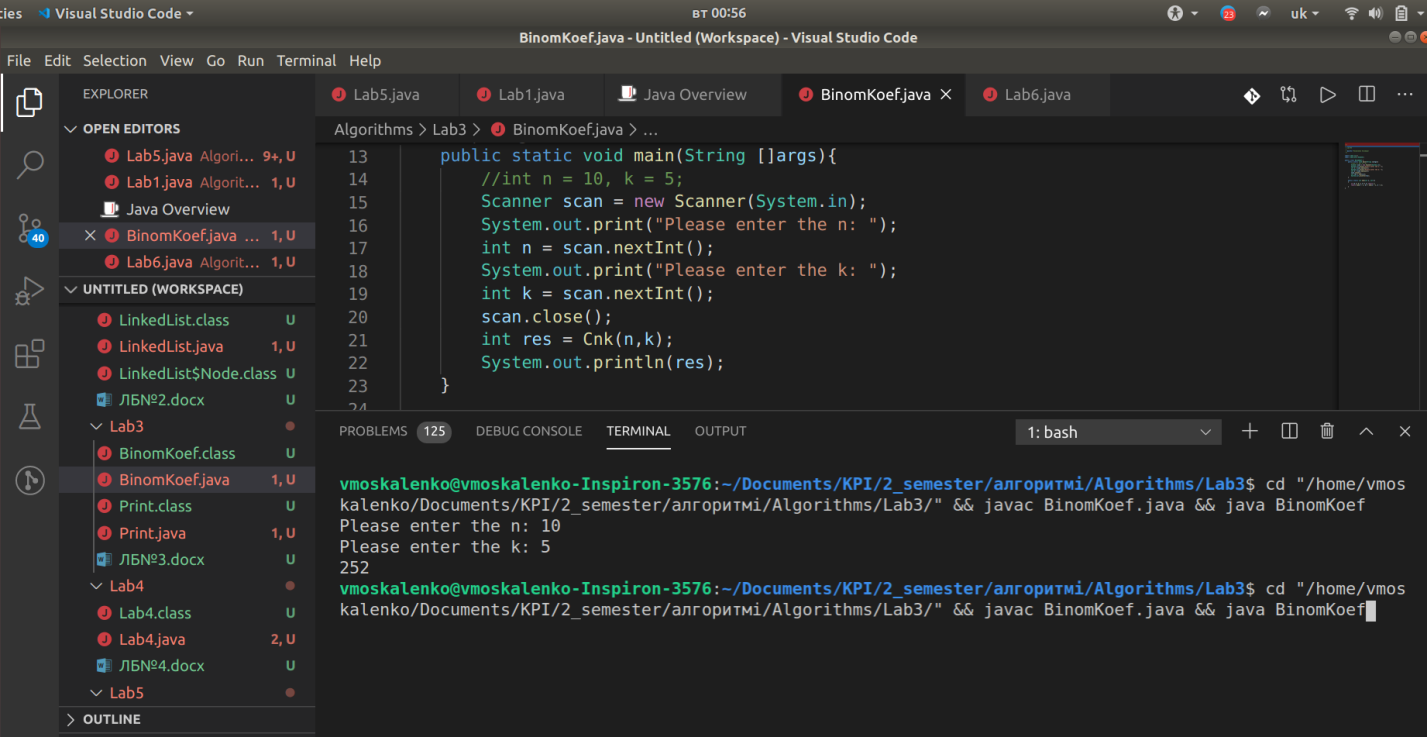
}

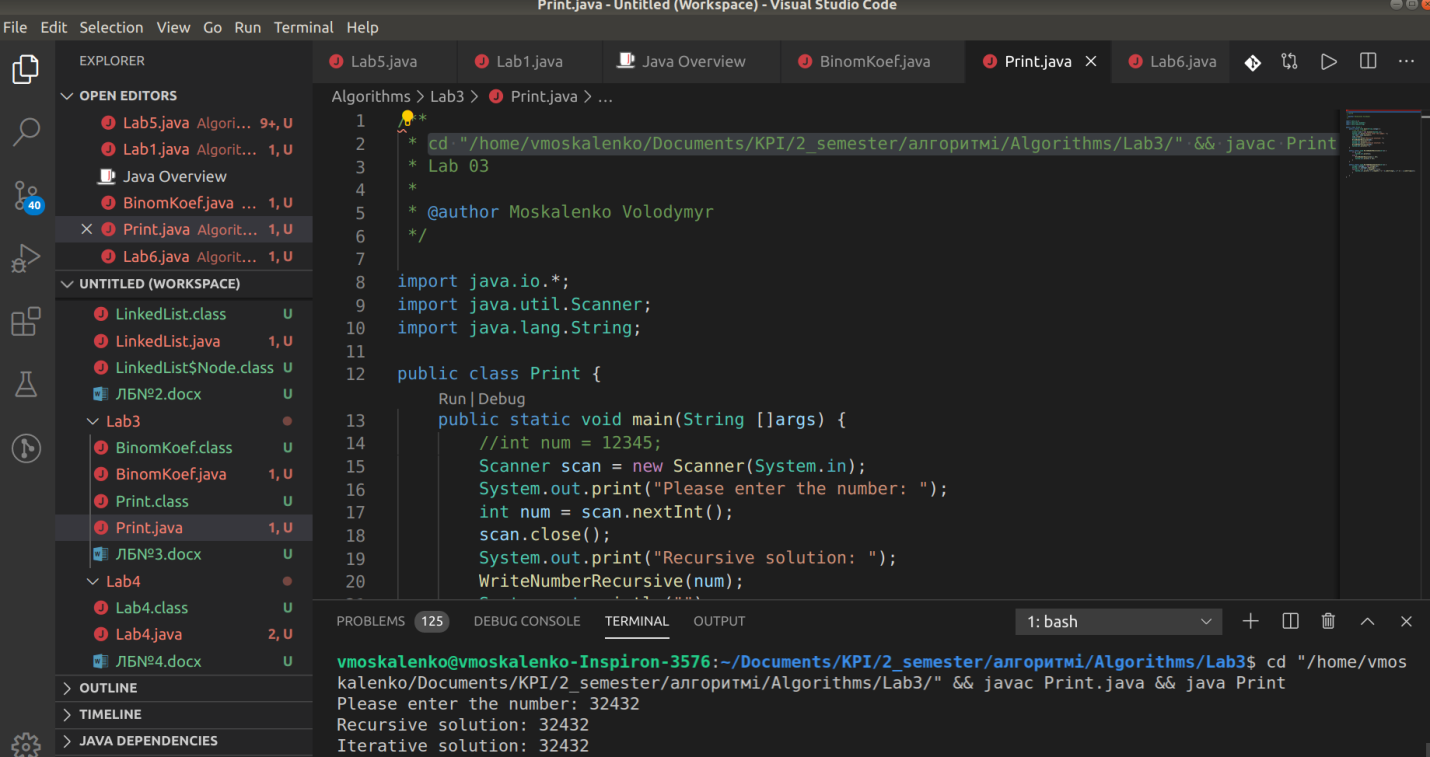
}

**6. Перевірка програми.**

Алгоритми були перевірені на натуральних числах так і на інших варіантах вхідних даних. Можливим випадком крешу даної програми є закінчення стеку в даній конкретній системі при великій кількості рекурсивних викликів. Також, буде спостерігатись збільшення часу роботи в порівняні з ітеративним циклом від об’єму даних, тобто чим більше буде рекурсивний стек, тим більше рекурсія буде програвати ітеративній реалізації.

7. Вивід програми





8. Висновок

Рекурсія - виклик функції (процедури) з неї ж самої, безпосередньо (проста рекурсія) або через інші функції (складна або непряма рекурсія). Кількість вкладених викликів функції або процедури називається глибиною рекурсією. Рекурсивна програма дозволяє описати повторюване або навіть потенційно нескінченне обчислення, причому без явних повторень частин програми і використання циклів.