МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет імені Михайла Драгоманова

Факультет математики, інформатики та фізики

Кафедра інформаційних технологій та програмування

**Звіт**

з лабораторної роботи №3

«»

з дисципліни «Програмування»

Виконав:

студент ІІІ курсу групи 31І

Бритвич О.В.

Перевірила:

викладач Устименко О.Б.

Оцінка

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ – 2023

Зміст

Мета роботи 3

1 Постановка задачі 4

2 Основна частина 5

2.1 Опис вхідних та вихідних даних 7

2.1 Блок-схема 8

2.2 Опис вхідних та вихідних даних 12

2.2 Блок-схема 13

Висновки 16

Список літератури 17

Додатки 18

**Мета роботи**

Розробити процедури для обчислення факторіалу та визначення чисел-паліндромів. Під час написання цих двох програм необхідно враховувати оптимальність та ефективність виконання, щоб забезпечити швидке та правильне виконання завдань.

**1 Постановка задачі**

1. Задача №1.

За даним цілим додатнім числом *n* обчисліть *n!* – значення факторіалу цього числа.

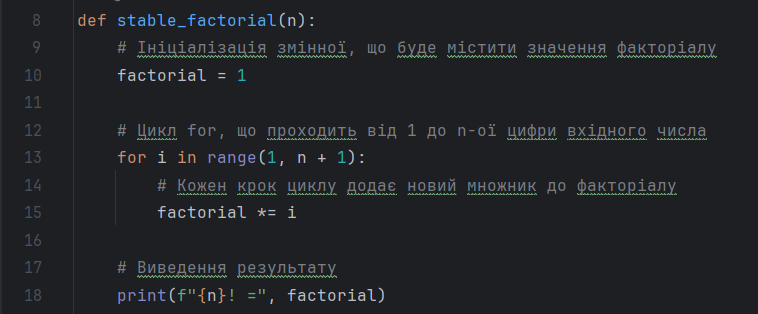
1. Задача №2.

Назвемо число паліндромом, якщо воно не змінюється при перестановці його цифр у зворотному порядку. Напишіть програму, яка за введеним користувачем числом *n (1 ≤ n ≤ 100000)* виводить числа-паліндроми, які не перевищують *n*.

**2 Основна частина**

Задача №1

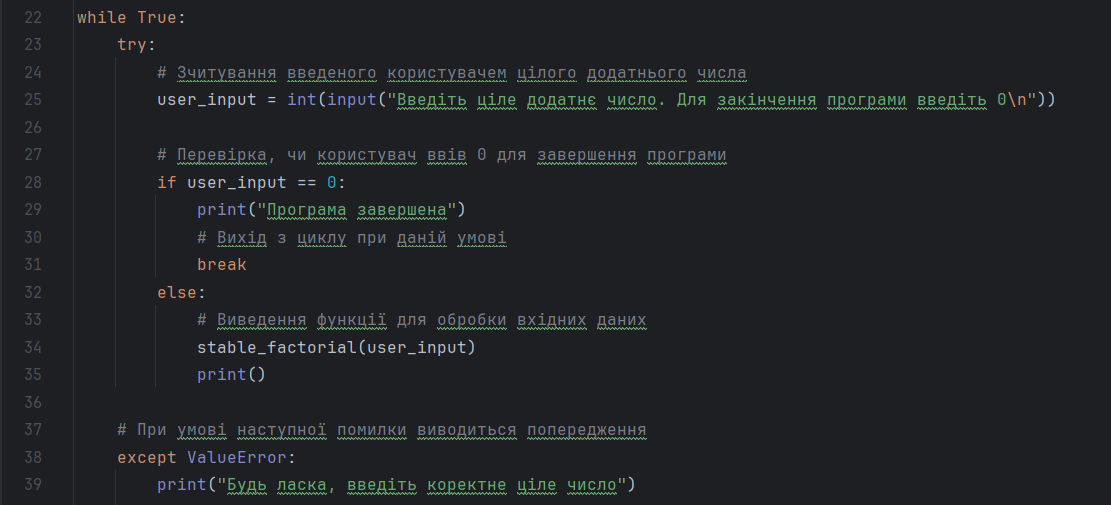
Наступна функція, *stable\_factorial*, отримує аргумент *n*, який представляє вхідне число, для якого потрібно обчислити факторіал. Далі ініціалізується змінна *factorial* з початковим значенням *1* для подальшого використання циклу *for*, який проходить від *1* до *n + 1* включно. На кожній ітерації циклу до значення *factorial* додається новий множник, що дорівнює поточному значенню ітератора *i*. Результат обчислення виводиться на екран:



Безкінечний цикл *while True* використовується для постійного введення користувачем цілих чисел і обчислення факторіалу для цих чисел.



Обробка помилок з блоком *try* та *except* використовується для обробки помилок, якщо користувач введе неціле число. Якщо введено некоректне значення, виводиться відповідне повідомлення про помилку. Введення користувача зчитується за допомогою функції *input*. Якщо введене число дорівнює *0*, програма виводить повідомлення про завершення і виходить з циклу за допомогою *break*. В іншому випадку (при утриманні умови задачі) викликається функція *stable\_factorial* для обчислення факторіалу введеного числа:



**2.1 Опис вхідних та вихідних даних**

Вхідні дані:

user\_input – змінна для зберігання введеного користувачем цілого додатнього числа для подальшої обробки.

Вихідні дані:

factorial – змінна для обробки та збереження факторіалу числа

**2.1 Блок-схема**

Початок

def stable\_factorial(n)

if user\_input == 0

Увести user\_input

try

factorial \*= i

for i in range(1, n + 1)

Вивести

f"{n}! =", factorial

while True

factorial = 1

Вивести текст про закінчення програми

Вивести текст про некоректно-введені дані

except ValueError

else

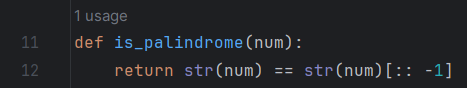
break

stable\_factorial(user\_input)

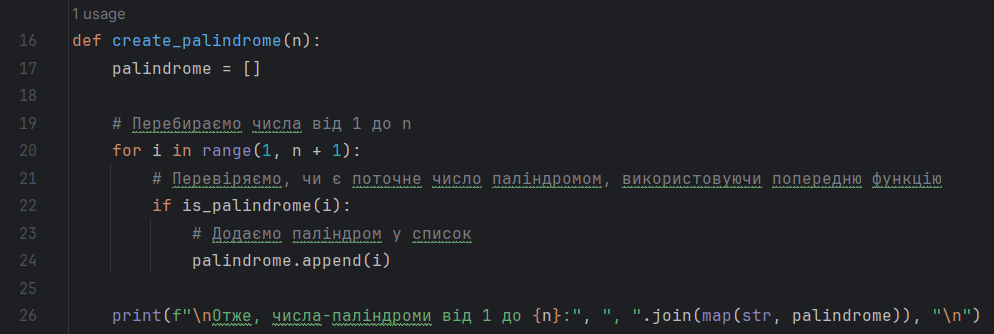
Кінець

Задача №2

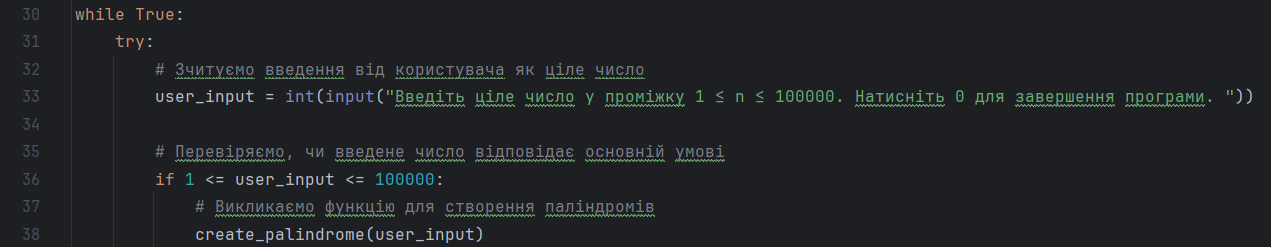
Перша функція *is\_palindrome* перевіряє, чи є передане число паліндромом, порівнюючи його рядкове представлення з його рядком, записаним у зворотному порядку:

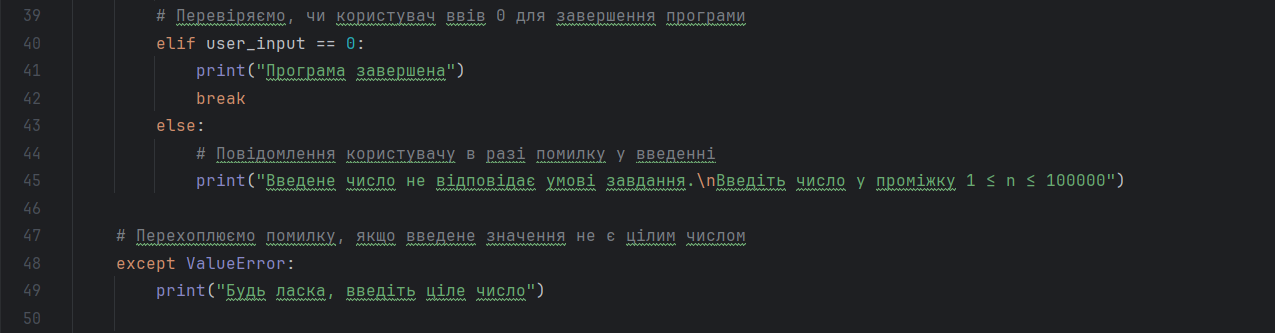


У наступній функції, *create\_palindrome*, створюється список чисел-паліндромів в заданому діапазоні від *1* до *n*. Використовується попередня функція *is\_palindrome* для перевірки, чи кожне число є паліндромом. Результат виводиться у вигляді текстового повідомлення:



Основний цикл програми чекає на введення від користувача. Введені значення перевіряються на відповідність умовам, і в залежності від введеного числа програма викликає функцію для створення паліндромів або завершує роботу. При введенні некоректних значень виводиться відповідне повідомлення:





**2.2 Опис вхідних та вихідних даних**

Вхідні дані:

user\_input – змінна для збереження цілого числа в проміжку від 1 до 100000 включно.

Вихідні дані:

всі паліндроми від 1 до введеного числа

**2.2 Блок-схема**

Початок

def is\_palindrome(num)

while True

for i in range(1, n + 1)

Вивести palindrome

palindrome.append(i)

palindrome = []

def create\_palindrome(n)

if is\_palindrome(i)

return str(num) == str(num)[:: -1]

try

Вивести текст про завершення програми

elif user\_input == 0

if 1 <= user\_input <= 100000

Увести user\_input

create\_palindrome(user\_input)

break

else

Вивести текст про невідповідність умові задачі

except ValueError

Вивести текст про некороктно-введені дані

Кінець

**Висновки**

Під час роботи над лабораторною роботою були розроблені дві програми, призначені для вирішення конкретних завдань.

Перша задача передбачала обчислення факторіалу для заданого цілого додатнього числа *n*. Для цього була розроблена програма, яка ефективно виконує обчислення факторіалу та виводить результат.

Друга задача стосувалася пошуку та виведення паліндромів, не перевищуючи задане користувачем число *n*. Розроблена програма ефективно впоралася з цим завданням.

В цілому, обидві програми виявилися функціональними та працездатними в рамках поставлених завдань. Їх можна успішно використовувати для рішення відповідних задач.

**Список літератури**

1Вивчаємо Python, В двох томах. 5-е видання. Марк Лутц (комплект з двох книг) – [Режим доступу] : <https://leader-books.com.ua/komp-yuterna-literatura/vivchayemo-python-v-dvoh-tomah-5-e-vidannya-mark-lutc-komplekt-z-dvoh-knig?gclid=CjwKCAiAgeeqBhBAEiwAoDDhnxzTW_ZFWceIMmEQLdDM0XPtB2uG9tUdKMSjvmw6BX7zHEoS49h0gxoC_skQAvD_BwE>

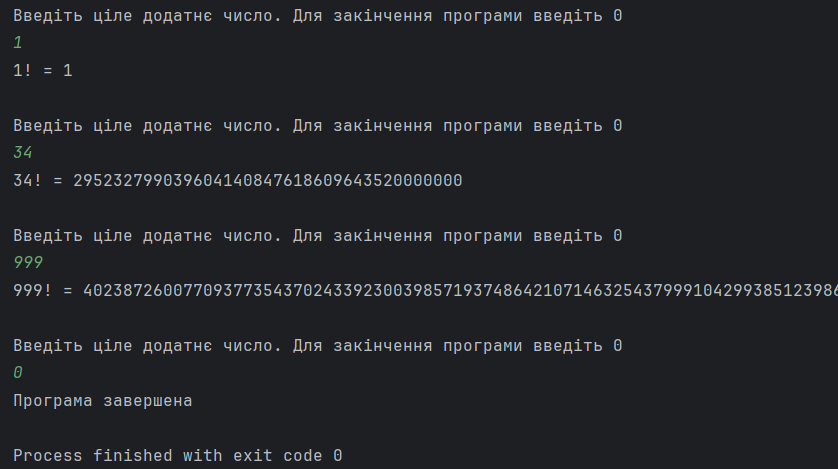
2 Форум з технічних питань – [Режим доступу] : <https://stackoverflow.com/>

**Додатки**

Текст програми №1

# Функція для знаходження "готового результату" факторіалу  
def stable\_factorial(n):  
 # Ініціалізація змінної, що буде містити значення факторіалу  
 factorial = 1  
  
 # Цикл for, що проходить від 1 до n-ої цифри вхідного числа  
 for i in range(1, n + 1):  
 # Кожен крок циклу додає новий множник до факторіалу  
 factorial \*= i  
  
 # Виведення результату  
 print(f"{n}! =", factorial)  
  
  
# Безкінечний цикл для введення числа  
while True:  
 try:  
 # Зчитування введеного користувачем цілого додатнього числа  
 user\_input = int(input("Введіть ціле додатнє число. Для закінчення програми введіть 0\n"))  
  
 # Перевірка, чи користувач ввів 0 для завершення програми  
 if user\_input == 0:  
 print("Програма завершена")  
 # Вихід з циклу при даній умові  
 break  
 else:  
 # Виведення функції для обробки вхідних даних  
 stable\_factorial(user\_input)  
 print()  
  
 # При умові наступної помилки виводиться попередження  
 except ValueError:  
 print("Будь ласка, введіть коректне ціле число")

Результат роботи програми №1



Текст програми №2

# Функція для перевірки, чи число є паліндромом  
def is\_palindrome(num):  
 return str(num) == str(num)[:: -1]  
  
  
# Функція для створення списку паліндромів в заданому діапазоні  
def create\_palindrome(n):  
 palindrome = []  
  
 # Перебираємо числа від 1 до n  
 for i in range(1, n + 1):  
 # Перевіряємо, чи є поточне число паліндромом, використовуючи попередню функцію  
 if is\_palindrome(i):  
 # Додаємо паліндром у список  
 palindrome.append(i)  
  
 print(f"\nОтже, числа-паліндроми від 1 до {n}:", ", ".join(map(str, palindrome)), "\n")  
  
  
# Безкінечний цикл для введення  
while True:  
 try:  
 # Зчитуємо введення від користувача як ціле число  
 user\_input = int(input("Введіть ціле число у проміжку 1 ≤ n ≤ 100000. Натисніть 0 для завершення програми. "))  
  
 # Перевіряємо, чи введене число відповідає основній умові  
 if 1 <= user\_input <= 100000:  
 # Викликаємо функцію для створення паліндромів  
 create\_palindrome(user\_input)  
 # Перевіряємо, чи користувач ввів 0 для завершення програми  
 elif user\_input == 0:  
 print("Програма завершена")  
 break  
 else:  
 # Повідомлення користувачу в разі помилку у введенні  
 print("Введене число не відповідає умові завдання.\nВведіть число у проміжку 1 ≤ n ≤ 100000")  
  
 # Перехоплюємо помилку, якщо введене значення не є цілим числом  
 except ValueError:  
 print("Будь ласка, введіть ціле число")

Результат роботи програми №2

