Розвиваємо алгоритмічне мислення

Алгоритми з циклами

Нижче наведені умови задач та їх розв'язки, оформлені у вигляді блок-схем. Для першої задачі наведено програмну реалізацію запропонованого алгоритму.

Вам необхідно ***проаналізувати*** *ці* ***задачі та алгоритми*** *і* ***скласти програми їх розв'язку****, використовуючи наведені блок-схеми.*

Пропонуємо самостійно протестувати правильність складених програм за допомогою наведених прикладів вхідних даних та результатів виконання програм для цих даних.

Це завдання не оцінюється і не впливає на підсумкову оцінку за курс та отримання сертифікату.

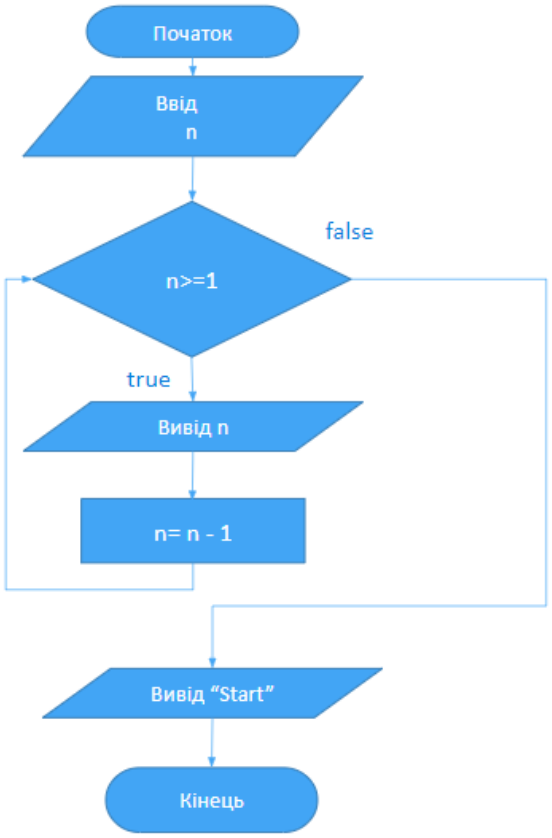
Задача 1.

Напишіть програму-таймер зворотного відліку, яка запитує у користувача кількість секунд n, з якої слід починати відлік.

Пояснення розв'язку

Організовуємо ввід числа `n`*. Потрібно організувати вивід всіх чисел від* `n` *до 1, для цього використаємо цикл* `while`*. Умова циклу* `n>=1`*. В тілі циклу організовуємо вивід числа* `n`*, після виводу зменшуємо значення числа* `n` *на одиницю. Після закінчення роботи циклу виводимо на екран "Start".*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
5	5 4 3 2 1 Start
3	3 2 1 Start

Код:

```
n = int(input())
while (n>=1):
    print(n)
    n=n-1
print("Start")
```

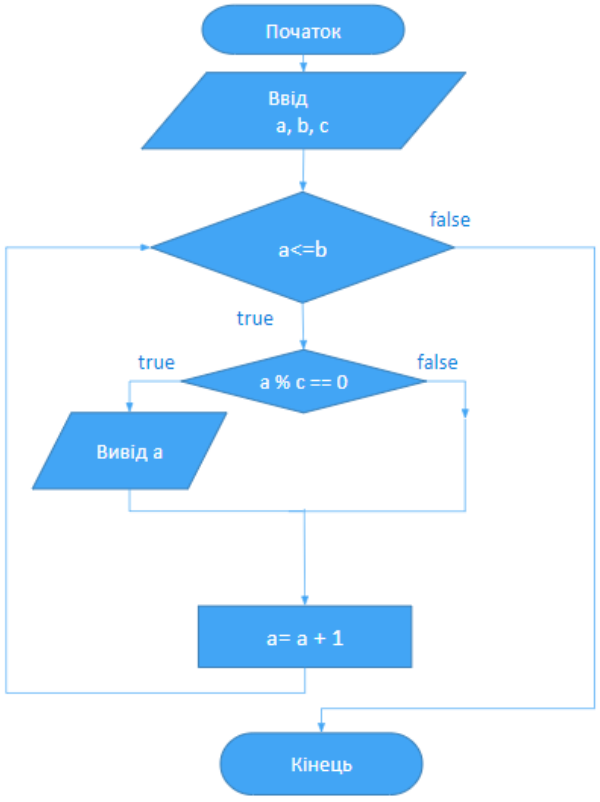
Задача 2.

Надрукувати всі цілі числа від a до b включно, кратні деякому числу c. Числа a, b, c - цілі числа, які вводить користувач.

Пояснення розв'язку

Організовуємо ввід трьох чисел `a, b, c`. *Для виводу послідовності чисел скористаємося циклом* `while`, *а умова виконання поки* `a<=b`. *В середині циклу для вибору саме тих чисел, що кратні* `c`, *перевіряємо чи остача від ділення* `a` *на* `c` *дорівнює нулю. Якщо умова виконується, то виводимо число на екран. Для переходу до наступного числа з діапазону збільшуємо значення* `a` *на одиницю. Після виходу з циклу завершуємо виконання програми.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
20 50 3	21 24 27 30 33 36 39 42 45 48
5 12 4	8 12

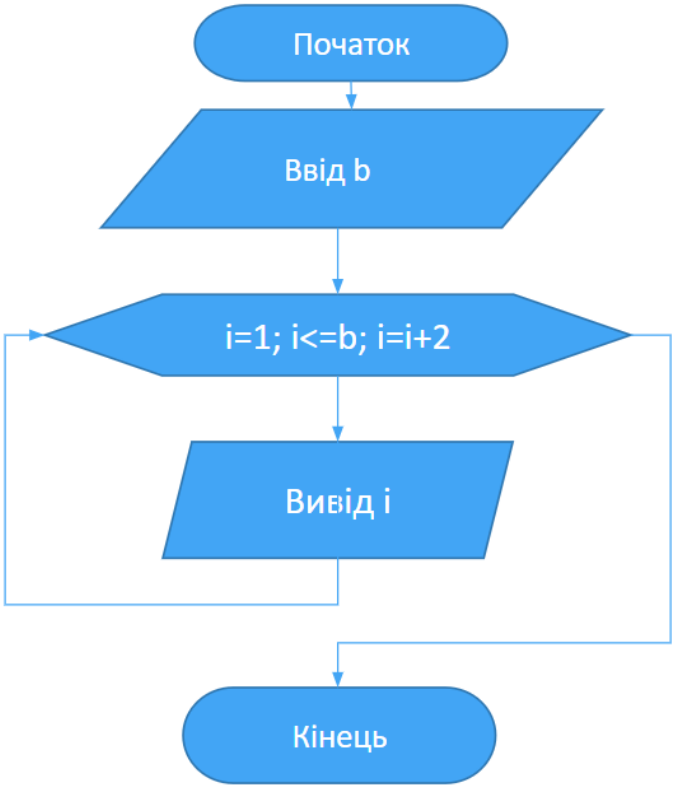
Задача 3.

Напишіть програму, яка друкує всі непарні числа з інтервалу [1, b], де b - ціле число, яке вводить користувач. Не можна використовувати конструкцію розгалуження.

Пояснення розв'язку

Вводимо значення змінної `b`*. Оскільки за умовою задачі не можна використовувати розгалуження, використаємо цикл з параметрами з початковим значенням параметру циклу* ***1****, та кроком зміни параметру циклу* ***2****, так як непарні числа це* ***1, 3, 5*** *і так далі. Цикл виконуємо до значення* `b`*.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
8	1 3 5 7
15	1 3 5 7 9 11 13 15

Задача 4.·

Задане ціле додатне число n, обчислити значення факторіалу цього числа n! -.

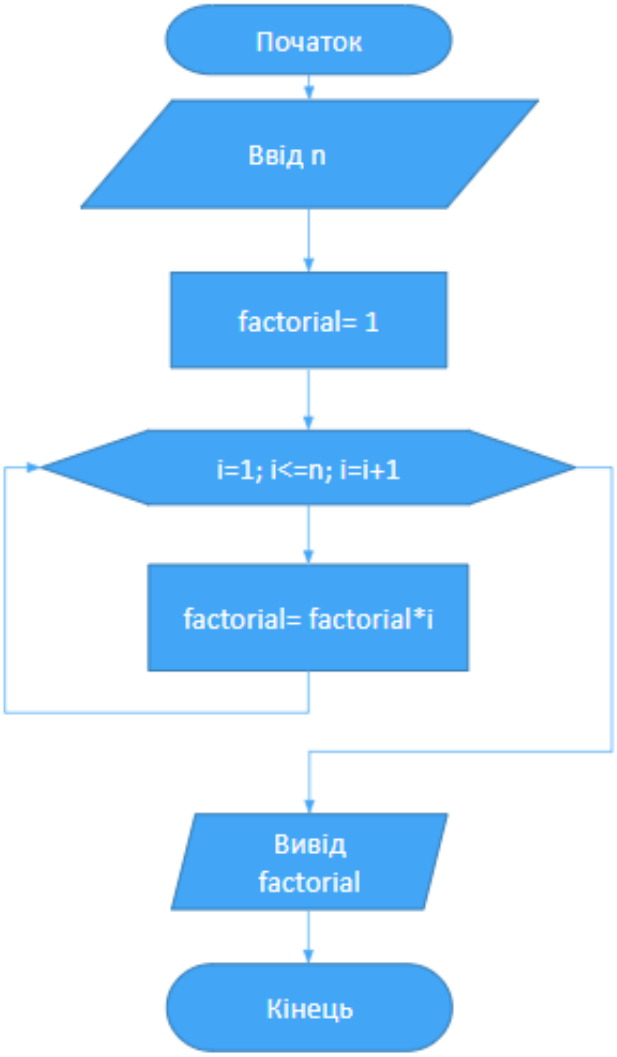
Приклад: 5! = 1 · 2 · 3 · 4 · 5 = 120

Пояснення розв’язку

Для обчислення факторіалу скористаємося циклом з параметрами, а результат множення зберігатимемо у змінну `factorial`*, якій для початку присвоюємо значення 1. Використовуємо цикл з параметрами, в якому параметр* `i` *буде змінюватися від 1 до* `n` *з кроком 1. В тілі циклу організовуємо множення* `factorial` *на значення параметру* `i`*. Після виконання циклу виводимо значення* `factorial` *на екран.*

* Факторіал числа `n` це - добуток всіх натуральних чисел починаючи від 1 до числа `n` включно (! - символфакторіал). Зазначу, що 0!= 1, 1! = 1, 2!= 1*2=2, 3!= 1*2*3=6 і так далі.

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
3	6
1	1
5	120

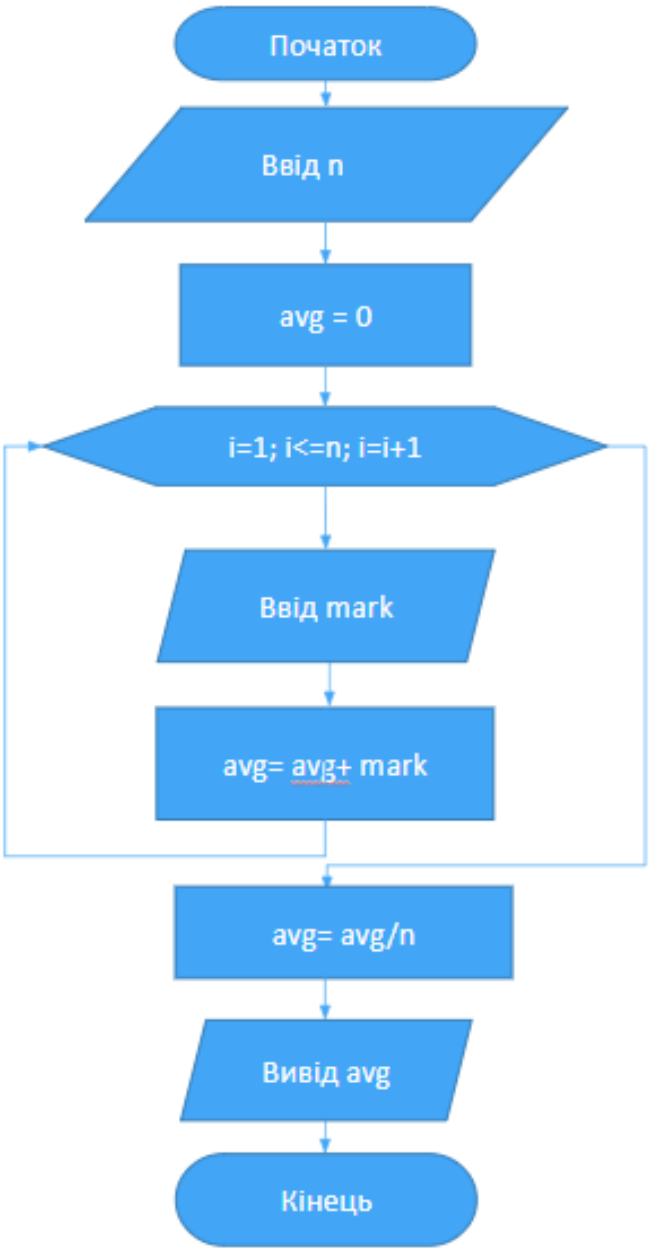
Задача 5.

Користувач вводить кількість навчальних предметів n, а потім оцінки учня з цих предметів. Визначте середню оцінку.

Пояснення розв’язку

Організовуємо ввід значення у змінну `n`*. Використаємо змінну* `avg` *для середньої оцінки, початково присвоюємо їй значення 0. Використовуємо цикл з параметрами для введення оцінок (змінна* `mark`*). Відразу в циклі знаходимо суму введених оцінок. Після завершення циклу знаходимо середню оцінку. Для цього ділимо значення* `avg` *на значення змінної* `n`*. Виводимо значення змінної* `avg` *на екран.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
5 10 11 9 8 10	9.60
3 10 12 11	11

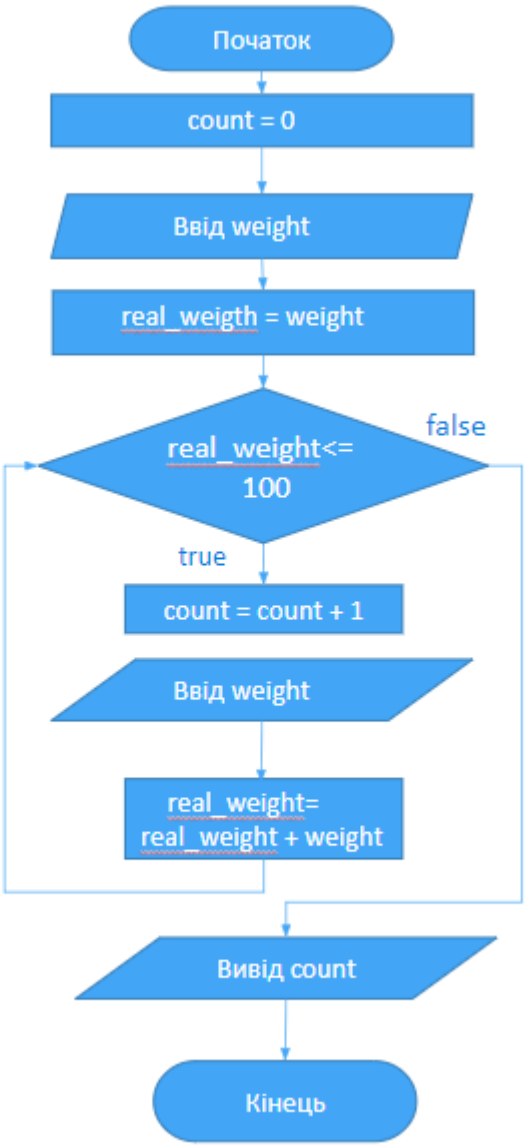
Задача 6.

На склад привозять однорідний вантаж на машинах різної вантажопідйомності. Користувач вводить інформацію про вагу вантажу чергової машини. Скласти програму підрахунку кількості машин, які прибули на склад до його заповнення, якщо місткість складу не більше 100 тонн

Пояснення розв'язку

Визначимо змінну `real_weight` *для збереження сумарної ваги привезеного машинами вантажу та змінну* `count` *для підрахунку кількості машин, які прибули на склад до його заповнення (присвоюємо цій змінній початкове значення 0). Вводимо вагу першої машини в змінну* `weight`*. Додаємо це значення до* `real_weight`*. Уциклі while, поки сумарна вага вантажів не більша за 100 тонн (* `real_weight<=100)` *збільшуємо кількість автомобілів* `count` *на 1, вводимо вагу наступного автомобіля* `weight` *та додаємо введене значення до змінної* `real_weight`*. Після завершення виконання циклу виводимо значення змінної* `count` *на екран.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
50 50	2
40 10 20 35	3
101	0

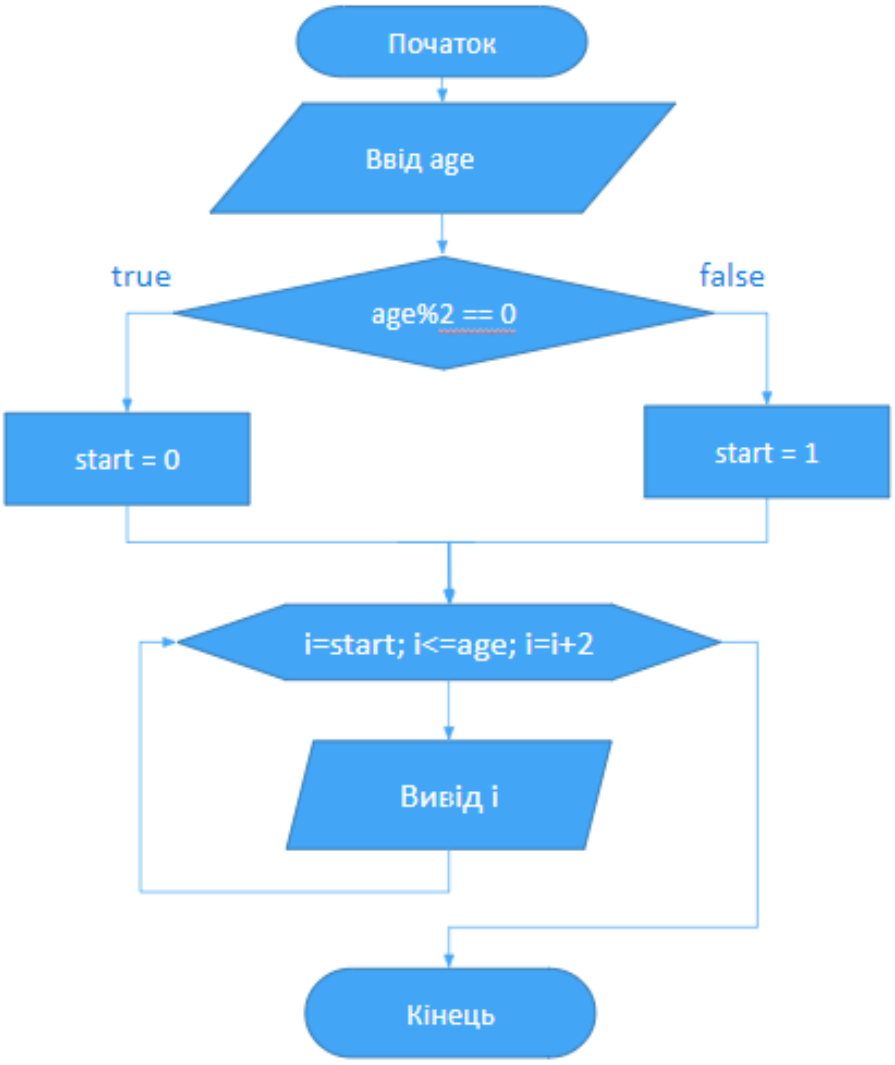
Задача 7.

Створити цикл, який виводить на екран парні числа починаючи з 0, поки не дійде до твого віку, якщо вік є парним числом, або виводить непарні числа починаючи з 1, поки не дійде до твого віку, якщо вікє непарним числом.

Пояснення розв'язку

Вводимо значення віку у змінну `age` *(ціле число). У змінній* `start` *будемо зберігати початкове значення для виводу. Перевіряємо умову, якщо* `age` *парне, то змінній* `start` *присвоюємо 0, інакше присвоюємо 1. Далі використовуємо цикл з параметром де крок зміни дорівнює 2. Початкове значення циклу -* `start`*. а кінцеве -* `age` *(включно). У тілі циклу організовуємо вивід значення параметра циклу.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
15	1 3 5 7 9 11 13 15

16	0 2 4 6 8 10 12 14 16

Задача 8.

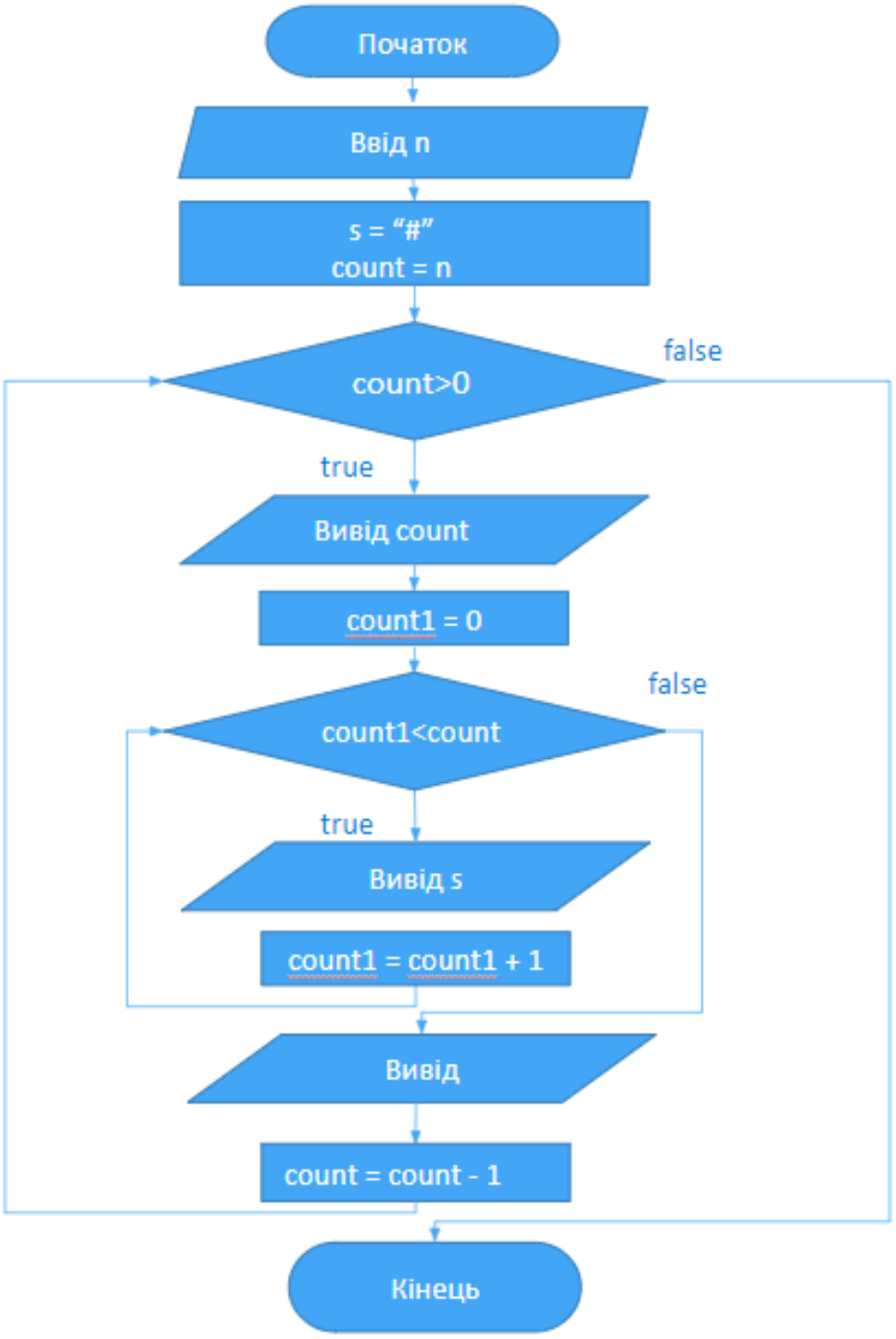
Напишіть програму для друку цілих чисел від n до 0 із виведенням біля кожного числа кількості символів #, що дорівнює значенню числа.

Пояснення розв'язку

Вводимо значення `n` *з клавіатури. Використаємо змінну* `s`*, якій присвоюємо значення "#". Для розв'язку потрібно створити два цикли. Зовнішній цикл організуємо із змінною* `count`*, якій перед циклом присвоюємо значення рівне* `n`*, а умова циклу буде* `count>0`*. В середині циклу виводимо значення змінної* `count` *після неї пробіл (але без переходу на новий рядок), змінній* `count1`*, яка використовуватиметься у внутрішньому циклі присвоюємо значення 0. Запускаємо внутрішній цикл з умовою* `count1<count`*. В тілі внутрішнього циклу організовуємо вивід значення* `s` *без переходу у новий рядок, далі збільшуємо значення* `count1` *на 1. Після завершення внутрішнього циклу використовуємо команду виводу без параметрів для переходу в новий рядок та зменшуємо значення* `count` *на 1. Після завершення зовнішнього циклу припиняємо виконання програми.*

* Вивід на екран змінної `num` без переходу на новий рядок `print(num, end= " ")` - з пробілом після неї, `print(num, end= "")`- без пробілу після неї, і `print()`– для переходу на новий рядок, без виводу змінних.

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
6	6 ###### 5 ##### 4 #### 3 ### 2 ##

	1 #
3	3 ### 2 ## 1 #

Задача 9.

З клавіатури вводять число. Вивести всі його цифри

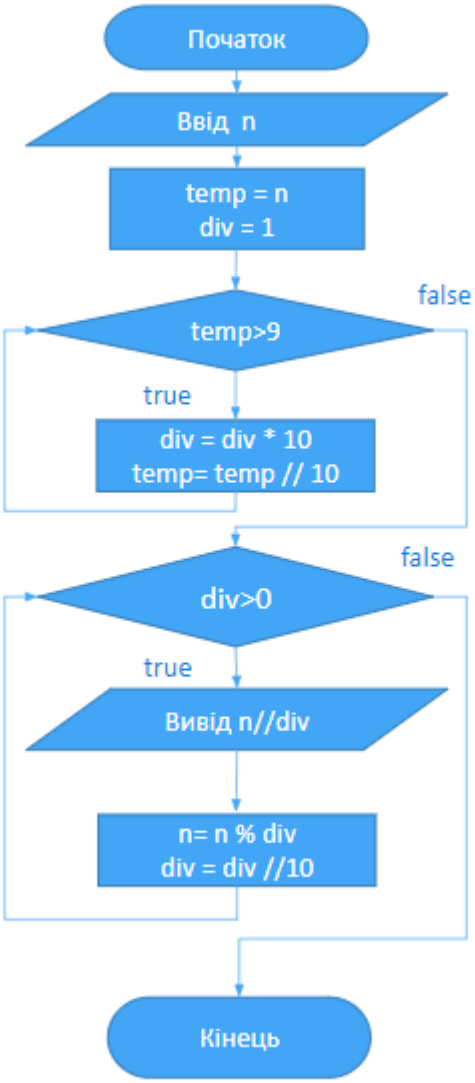
Пояснення розв'язку

Введемо число `n`*. Для отримання цифр числа* `n`*, скористаємося такими думками: наприклад,* `n`*=4216 і щоб отримати першу цифру цього числа можна поділити його націло на 1000, отримаємо 4, щоб отримати 2, потрібно спочатку знайти остачу від ділення 4216 на 1000, це буде 216, а потім 216 ділити націло на 100 і так далі поки не отримаємо всі цифри. Таким чином нам потрібно найперше сформувати число, що починатиметься з 1 і матиме кількість нулів на 1 менше ніж цифр в числі* `n` *назвемо цю змінну div .*

Використаємо додаткову змінну `temp` *(щоб зберегти початкове число) та присвоюємо їй значення* `n`*. Змінній div присвоюємо значення1. Організуємо цикл* `while` *з умовою* `temp` *> 9. В тілі циклу будемо ділити* `temp` *націло на 10, а* `div` *відповідно домножати на 10.*

Тепер організовуємо новий цикл `while` *з умовою* `div > 0`*. В тілі циклу будемо виводити на екран n поділено націло на* `div`*, далі* `n` *присвоюємо остачу від ділення* `n` *на* `div` *і* `div` *ділимо націло на 10.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
456	4 5 6
35421	3 5 4 2 1

Задача 10.

Перевірити, чи введене число просте.

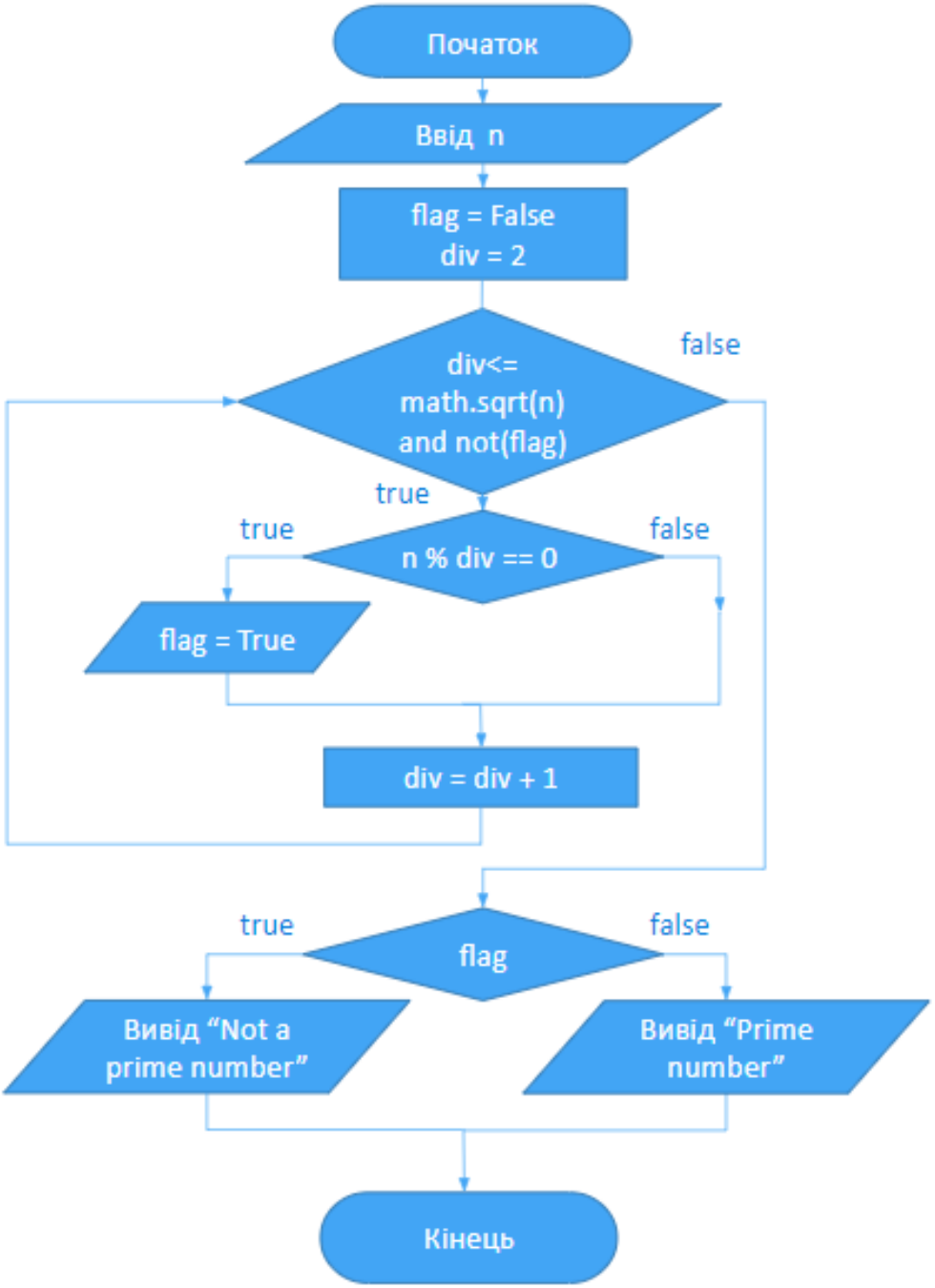
Пояснення розв'язку

Введемо число для перевірки у змінну `n`*. Використаємо змінну* `flag` *для позначення того, що знайшли дільник відмінний від 1. Початково присвоюємо* `flag` *значення* ***False.*** *Використаємо змінну div для зберігання дільників з початковим значенням 2. Використовуємо цикл* `while` *з умовою* `div<= math.sqrt(n) and not(flag)` *(така умова тому що, дільник якщо він є то найбільший може бути корінь з числа* `n`*, а інша частина "поки не знайдено іншого дільника відмінного від 1"). В тілі циклу у вказівці розгалуження перевіряємо чи* `n` *ділиться на* `div` *без остачі, якщо так то* `flag` *присвоюємо* ***True*** *і це відразу вплине на умову циклу і відбудеться вихід з нього. Після розгалуження збільшуємо* `div` *на 1.*

Вихід з циклу можливий у двох випадках, коли `flag` *стало рівним* ***True*** *- тоді число "не просте", або коли* `div` *стало більшим за* `math.sqrt(n)` *- в такому випадку число просте. Для перевірки цього використовуємо розгалуження, де перевіряємо значення* `flag`*. Якщо істина то вируковуємо на екран "Not a prime number", в іншому випадку друкуємо "Prime number".*

* Просте число це - число, яке ділиться націло лише на 1 і саме на себе. Тому будемо перевіряти чи число має інші дільники окрім 1 і самого себе.

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
13	Prime number
14	Not a prime number