Розвиваємо алгоритмічне мислення

Допоміжні алгоритми

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 функцію, яка приймає ціле число і друкує інформацію про парність чи непарність числа.

Пояснення розв'язку

Описуємо допоміжний алгоритм, який назвемо `parity_check`*. Нехай вхідним параметром буде* `number`*. В тілі допоміжного алгоритму використаємо розгалуження з умовою* `number%2 == 0`*. Якщо умова виконується то виводимо на екран "The number is odd", іншому випадку виводимо "The number is even".*
Організовуємо у головній програмі ввід числа у змінну `n`*. Викликаємо допоміжний алгоритм* `parity_check` *передавши йому* `n` *як параметр.*

Блок-схема:

Допоміжний алгоритм

Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
2	The number is even
9	The number is odd

Код:

```
def parity_check(number):
    if number % 2 == 0:
        print("The number is even")
    else:
        print("The number is odd")

n = int(input())
parity_check(n)
```

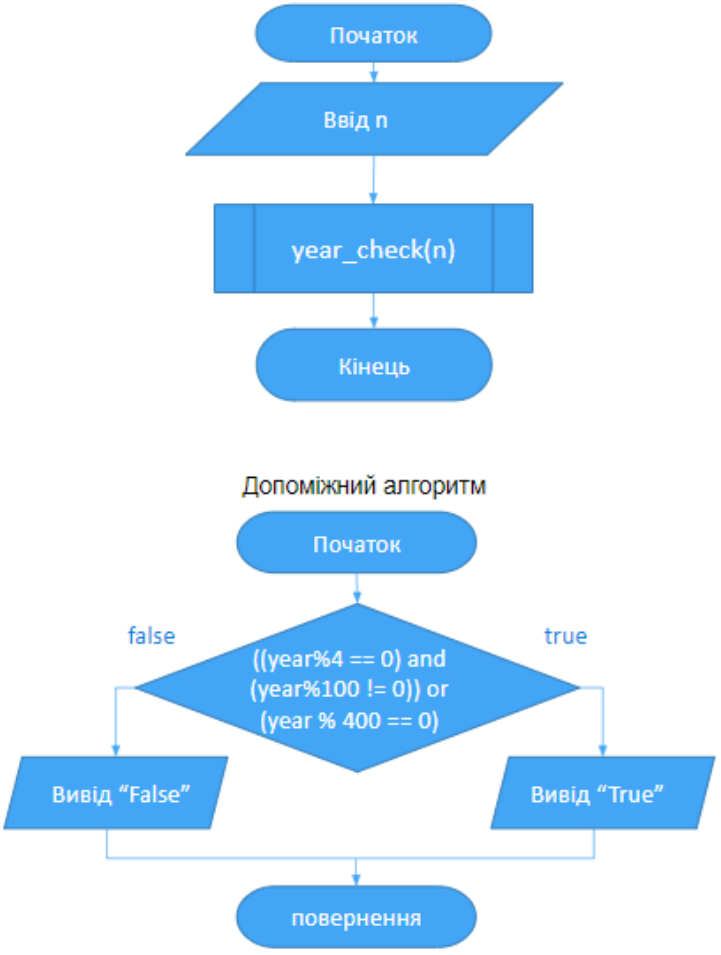
Задача 2.

Напишіть функцію яка перевірятиме чи введений рік високосний.

Пояснення розв'язку

Описуємо допоміжний алгоритм, який назвемо `year_check`*. Нехай вхідним параметром буде* `year`*. В тілі допоміжного алгоритму використаємо розгалуження з умовою* `((year%4 == 0) and (year%100 != 0)) or (year % 400 == 0)`*. Якщо умова виконується то виводимо на екран "True", іншому випадку виводимо "False". Організовуємо у головній програмі ввід числа у змінну* `n`*. Викликаємо допоміжний алгоритм* `year_check` *передавши йому* `n` *як параметр.*

Блок-схема:



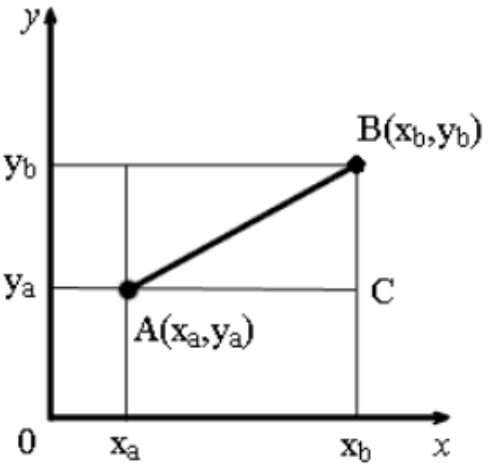
Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
2023	False
2020	True

Задача 3.

Дано чотири числа x1, y1, x2, y2. Написати функцію dist(x1, y1, x2, y2), яка обчислює відстань між двома точками (x1, y1) та (x2, y2).

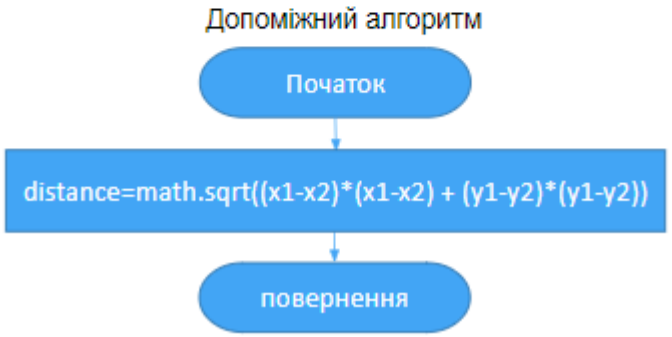
Пояснення розв'язку



Опишемо допоміжний алгоритм, який назвемо dist. Нехай вхідними параметрами будуть `x1, y1, x2, y2`*. В тілі алгоритму обчислюємо значення змінної* `distance` *за формулою* `math.sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2))`*. Це значення і повертатиметься в головний алгоритм.*

Організовуємо у головній програмі введення 4 чисел, відповідно у змінні `coordx1, coordy1` *(для першої точки),* `coordx2, coordy2` *(для другої точки). Викликаємо допоміжний алгоритм* `dist` *з параметрами* `coordx1, coordy1, coordx2, coordy2` *та повертаємо значення у змінну* `distance_between_points`*. Виводимо змінну* `distance between points` *на екран.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
2 2 1 3	1.4142135623730951
1 2 4 5	4.242640687119285

Задача 4.

Напишіть функцію max_two, яка визначає більше з двох чисел a, b. Використовуючи цю функцію, знайдіть найбільше з чотирьох чисел.

Пояснення розв'язку

Опишемо допоміжний алгоритм, який називається `max_two`*. Нехай вхідними параметрами будуть* `a, b`*. В тілі допоміжного алгоритму використаємо розгалуження з умовою* `a>b`*. Якщо умова виконується то змінній* `max` *присвоюємо*

значення `a`, *в іншому випадку* `max` *присвоюємо значення* `b`. *Значення* `max` *повертатиметься в головний алгоритм.*
Організовуємо у головній програмі введення 4 чисел, відповідно у змінні `number1`, `number2`, `number3`, `number4`. *Викликаємо допоміжний алгоритм* `max_two` *з параметрами* `number1`, `number2` *та повертаємо значення у змінну* `more_than_two1`. *Повторно викликаємо допоміжний алгоритм* `max_two` *з параметрами* `number3`, `number4` *та повертаємо значення у змінну* `more_than_two2`. *Втретє викликаємо допоміжний алгоритм* `max_two` *з параметрами* `more_than_two1`, `more_than_two2` *та повертаємо значення у змінну* `more_than_four`. *Виводимо змінну* `more_than_four` *на екран.*

Блок-схема:

Допоміжний алгоритм

Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
2 2 1 3	3
1 2 4 1	4

Задача 5.

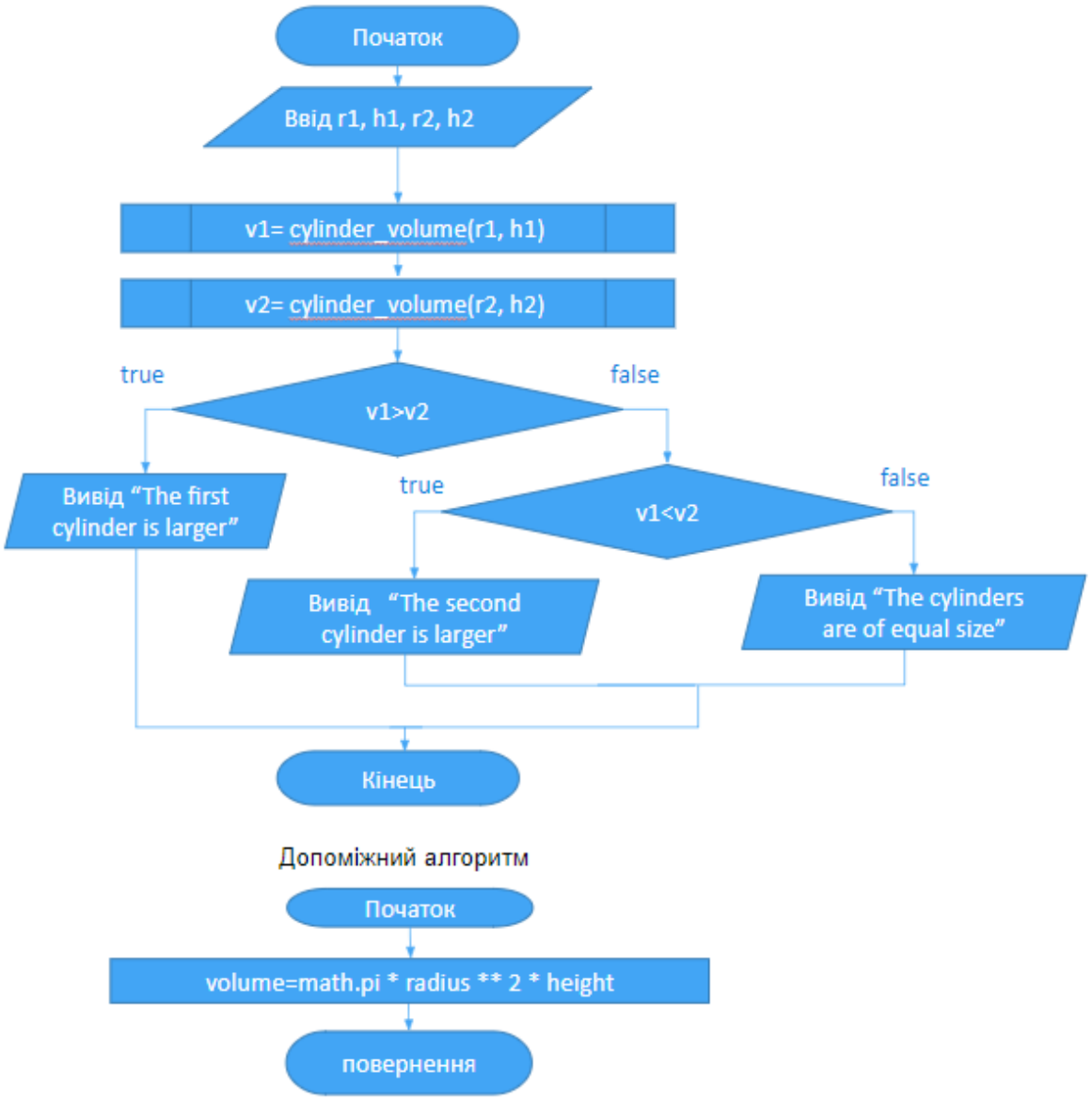
Дано два циліндра задані радіусами основи та висотами. Визначити, який з циліндрів має більший об’єм.
Створіть функцію cylinder для обчислення об'єму циліндра. У параметрах вказати радіус основи та висоту.

* Об'єм циліндра шукають за формулою pi * r ** 2 * h.

Пояснення розв’язку

Опишемо допоміжний алгоритм, який назвемо `cylinder_volume`*. Нехай вхідними параметрами будуть* `radius` *та* `height`*. В тілі алгоритму обчислюємо значення змінної* `volume` *за формулою* `math.pi * radius ** 2 * height`*. Це значення і повертатиметься в головний алгоритм.*
Організовуємо у головній програмі введення 4 чисел, відповідно у змінні `r1, h1` *(для першого циліндра),* `r2, h2` *(для другого циліндра). Викликаємо допоміжний алгоритм* `cylinder_volume` *з параметрами* `r1, h1` *та повертаємо значення у змінну* `v1` *(для об'єму першого циліндра). Ще раз викликаємо* `cylinder_volume` *але вже з параметрами* `r2, h2` *та повертаємо результат у змінну* `v2` *(для об'єму другого циліндра). Тепер використовуючи розгалуження порівнюємо об'єми циліндрів. Є три випадки, тому спочатку використовуємо порівняння* `v1> v2`*, якщо умова виконується виводимо на екран “The first cylinder is larger”. Якщо ж умова не виконується, то використовуємо ще одне розгалуження з умовою* `v1<v2`*, у випадку істинності виводимо “The second cylinder is larger”, а у випадку хибності “The cylinders are of equal size”*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
10, 5, 5, 10	The first cylinder is larger
2, 5, 5, 3	The second cylinder is larger

Задача 6.

Напишіть функцію, яка перевіряє, чи існує трикутник із введеними сторонами a, b, c.

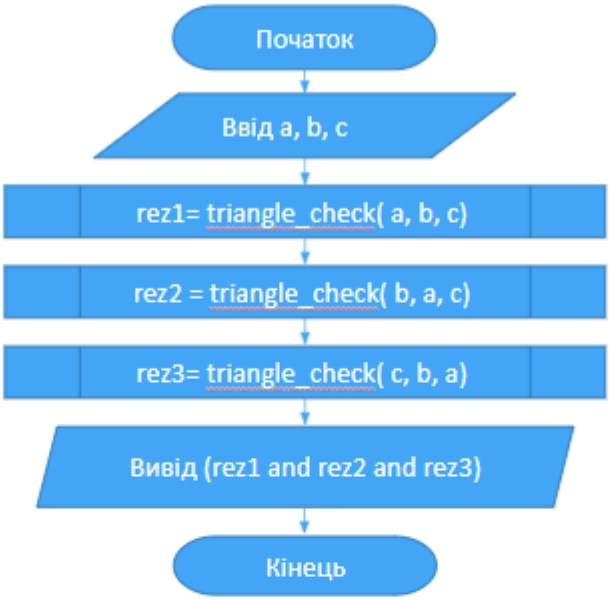
Пояснення розв'язку

Трикутник існує, якщо сума довжин будь-яких двох його сторін більша за довжину третьої сторони.

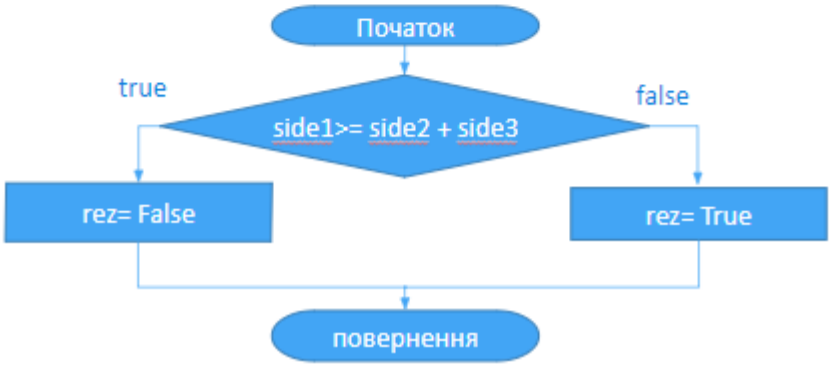
Опишемо допоміжний алгоритм, який назвемо `triangle_check`*. Нехай вхідними параметрами будуть* `side1, side2` *та* `side3`*. В тілі алгоритму використовуємо розгалуження з умовою* `side1> = side2 + side3`*, якщо умова істинна то змінній* `rez` *присвоюємо значення "False", у іншому випадку змінній* `rez` *присвоюємо"True".*

Організовуємо у головній програмі введення трьох чисел `a, b, c`*. Викликаємо допоміжний алгоритм* `triangle_check` *з параметрами* `a, b, c`*, а результат виконання присвоюємо змінній* `rez1`*. Повторно викликаємо допоміжний алгоритм* `triangle_check`*, але вже з параметрами* `b, a, c`*, а результат виконання присвоюємо змінній* `rez2`*. І втретє викликаємо допоміжний алгоритм* `triangle_check`*, але вже з параметрами* `c, b, a`*, а результат виконання присвоюємо змінній* `rez3`*. Організовуємо вивід (*`rez1 and rez2 and rez3`*), якщо жодна із змінних* `rez1, rez2, rez3`*, не матиме значення "False", то результат на екрані буде "True", інакше на екрані буде виведено "False".*

Блок-схема:



Допоміжний алгоритм



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
2 2 1	True
1 2 4	False

Задача 7.

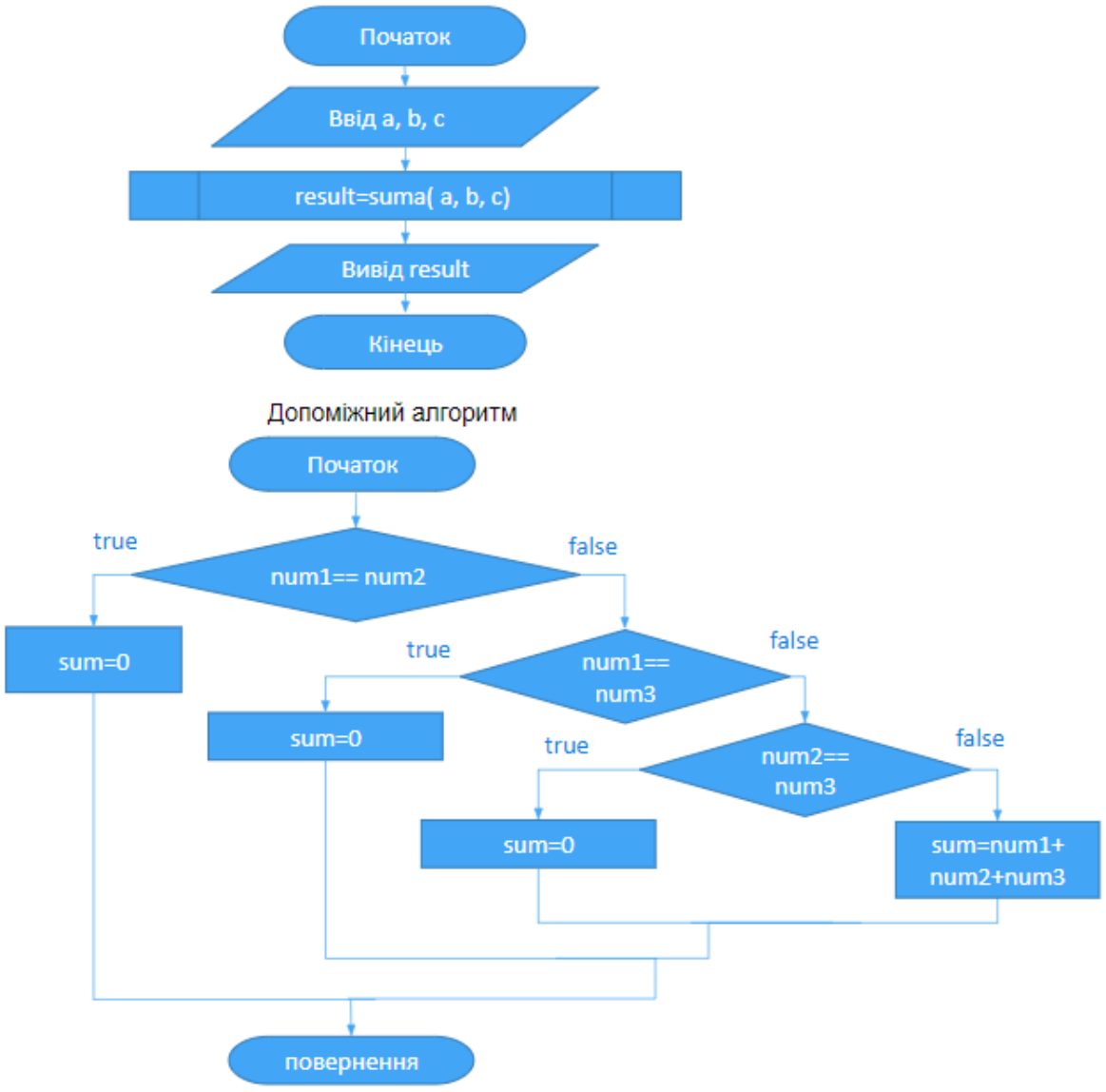
Напишіть функцію для визначення суми трьох цілих чисел. Якщо будь-які два значення однакові, необхідно вивести нуль.

Пояснення розв’язку

Опишемо допоміжний алгоритм, який назвемо `suma`*. Нехай вхідними параметрами будуть* `num1`*,* `num2` *та* `num3`*. В тілі алгоритму використовуємо змінну* `sum` *для обчислення суми. Далі йде розгалуження з умовою* `num1== num2`*, якщо умова істинна то змінній* `sum` *присвоюємо значення 0. Якщо умова хибна, то знову використовуємо розгалуження з умовою* `num1==num3`*. У випадку істинності також* `sum` *присвоюємо 0, а у випадку хибності ще одне розгалуження, але умова* `num2== num3`*. Якщо умова виконується то* `sum` *присвоюємо значення 0, у випадкухибності* `sum` *присвоюємо значення* `num1+num2+num3`*.*

Організовуємо у головній програмі введення 3 чисел `a, b, c`*. Викликаємо допоміжний алгоритм* `suma` *з параметрами* `a, b, c`*, а результат записуємо у змінну* `result`*. Виводимо значення* `result` *на екран.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані

1 2 3	6
1 2 2	0

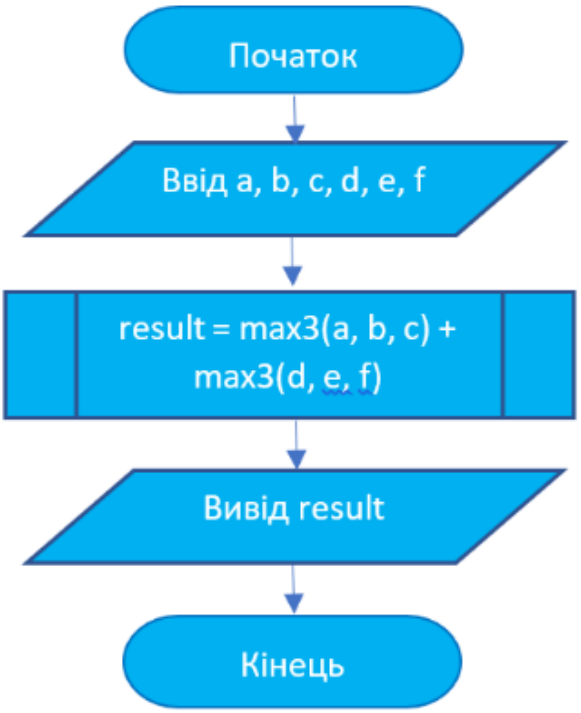
Задача 8.

Використовуючи функцію max2(a,b), яка визначає максимальне з двох заданих значень, написати функцію max3(a,b,c), що визначатиме максимальне з трьох даних значень і організувати виклик цієї функції для обчислення суми найбільших значень двох трійок довільних дійсних чисел

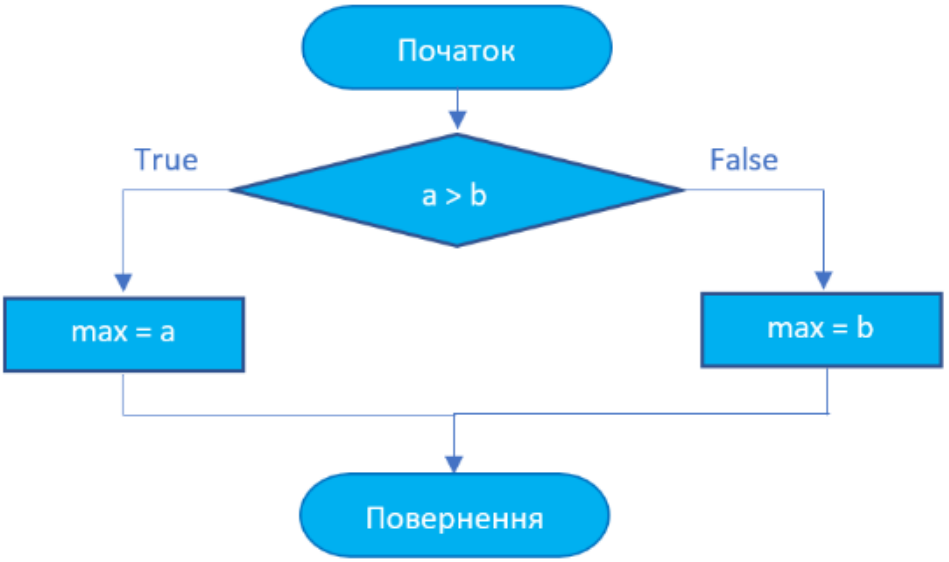
Пояснення розв'язку

Допоміжний алгоритм `max2(a,b)` *аналогічний допоміжному алгоритму* `max_two(a,b)` *з задачі 4. Щоб знайти найбільше з трьох чисел у допоміжному алгоритмі* `max3(a,b,c)` *викличемо у ньому алгоритм* `max2(a,b)` *– так ми знайдемо найбільше з перших двох чисел* `a` *і* `b` *, а потім знайдений результат і третє число* `c` *будуть аргументами другого виклику допоміжного алгоритму* `max2(max2(a,b),c)`*.*

Блок-схема:



Допоміжний алгоритм `max2(a, b)`



Допоміжний алгоритм `max3(a,b,c)`

Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
1 2 3 4 5 6	9
5 5 2 7 2 4	12

Задача 9.

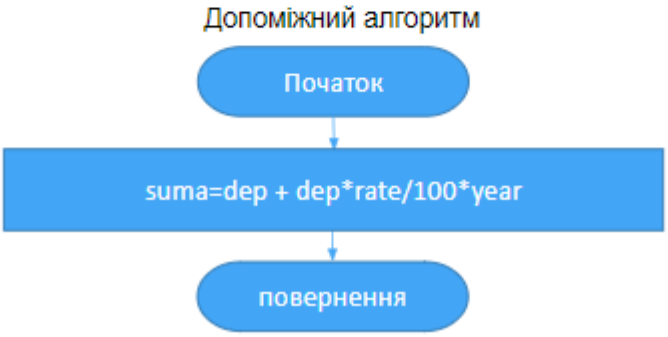
Вкладник розмістив суму розміром n грн. в банку. Визначте, яку суму отримає вкладник через m років, якщо відсоткова ставка складає p% в рік. Дані вводяться в порядку n, p, m як у вхідних даних.

Пояснення розв'язку

Опишемо допоміжний алгоритм, який назвемо `deposit`. *Нехай вхідними параметрами будуть* `dep, rate` *та* `year`. *В тілі алгоритму використовуємо змінну suma для обчислення суми яка буде через певну кількість років за формулою:* `dep + dep*rate/100*year`.

Організовуємо у головній програмі введення 3 числа `n, p, m`. *Викликаємо допоміжний алгоритм* `deposit` *з параметрами* `n, p, m`, *а результат записуємо у змінну* `result`. *Виводимо значення result на екран.*

Блок-схема:



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
5000	6800.00

18 2	

Задача 10.

Написати функцію обчислення НСД (найбільший спільний дільник). Скористайтеся алгоритмом Евкліда

Пояснення розв'язку

Опишемо допоміжний алгоритм, який назвемо `gsd`.

Для розв'язання цієї задачі скористаємося рекурентним співвідношенням, яке оформимо у вигляді допоміжного алгоритму `gsd(num1,num2)`, *де* `num1` *та* `num2` *– числа для яких потрібно знайти НСД.*

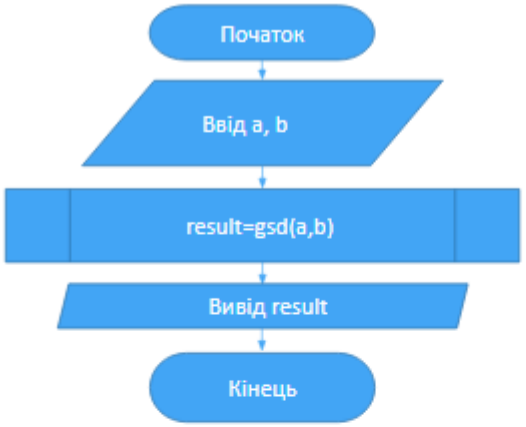
Рекурентне співвідношення – це формула, яка дозволяє обчислити наступні члени числової послідовності через значення попередніх членів.

Отже вхідними параметрами будуть `num1` *та* `num2`. *В тілі алгоритму використовуємо розгалуження з умовою* `num2==0`, *якщо умова істинні то повертаємо з функції значення* `num1`. *Якщо умова хибна, то знову викликаємо допоміжний алгоритм* `gsd`, *але вже з параметрами* `num2` *та* `num1%num2` *та повертаємо результат його виконання.*

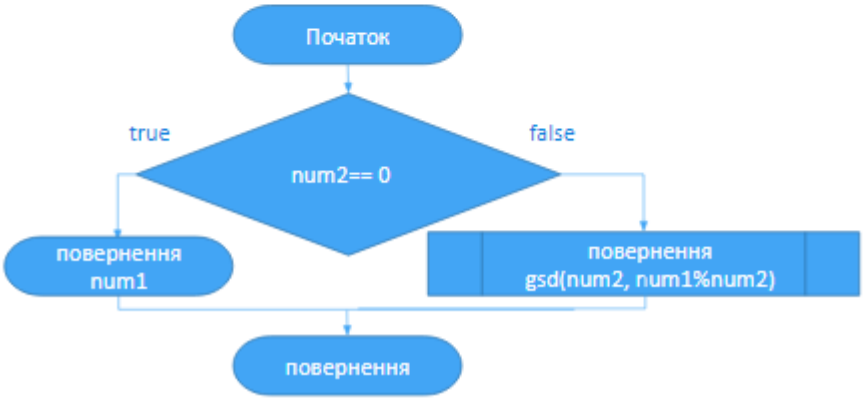
І так до того часу поки не виконається умова `num2==0`.

Організовуємо у головній програмі введення 2 числа `a, b`. *Викликаємо допоміжний алгоритм* `gsd` *з параметрами* `a, b`, *а результат записуємо у змінну* `result`. *Виводимо значення* `result` *на екран.*

Блок-схема:



Допоміжний алгоритм



Приклади вхідних даних та результатів

Вхідні дані	Вихідні дані
15 5	5
15 10	5