# Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Досліждення рекурсивних алгоритмів»

Варіант 19

Виконав студент ІП-11 Лисенко Андрій Юрійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Мартинова Оксана Петрівна

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота№ 6**

**Дослідження рекурсивних алгоритмів**

Мета – дослідити особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм.

**Індивідуальне завдання**

**Варіант 19**

Знайти натуральне число від 1 до *n* з максимальною сумою дільників.

**Постановка задачі**

Використаємо чотири підпрограми для знаходження числа з максимальною сумою дільників з проміжку від 1 до *n*. Першу та другу підпрограму використаємо для введення та виведення. Використаємо третю підпрограму (*Sum\_divisors*) з параметром *k* для знаходження суми дільників цього числа. Підпрограма *solution(n)* побудує розв'язок задачі рекурентно. Таким чином основна програма буде складатися з цих трьох програм - введення, виведення та *Sum\_divisors(k)*. Зазначимо, що умова вихода з рекурентного циклу підпрограми *solution(n)* буде *n* =0*.*

**Побудова математичної моделі**

Складемо таблицю змінних:

|  |  |  |  |
| --- | --- | --- | --- |
| Ім'я | Тип | Змінна | Призначення |
| *n* | int | Границя обчислення | Зберігати інформацію про границю обчислення |
| *max\_S\_div* | int | Максимальная сума дільників | Зберігати інформацію про максимальну суму дільників |
| *max\_n* | int | Число з максимальною сумою дільників | Зберігати інформацію про число з максимальною сумою дільників |
| *S\_div* | int | Сума дільників | Зберігати інформацію про суму дільників числа *k* |
| *div* | int | Дільник | Зберігати інформацію про дільник числа *k* |

Складемо таблицю підпрограм:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Підпрограма | Тип | Ім’я | Параметри | Призначення |
| Введення | void | *Input* | - | Введення границі обчислення |
| Сума дільників | int | *Sum\_divisors* | *k* | Знаходження суми дільників числа |
| Рішення | void | *solution* | - | Знайдення рішення задачі |
| Виведення | void | *Output* | - | Виведення результату |

**Розв’язання**

Програмні специфікації за пишемо у псевдокоді та графічній формі у вигляді блок-схеми.

**Основна програма:**

***Крок 1:*** Визначимо основні дії

**Підпрограма *input()*:**

***Крок 1:*** Введення *n*

**Підпрограма *output()*:**

***Крок 1:*** Виведення *max\_n, max\_S\_div*

**Підпрограма *Sum\_divisors(k):***

***Крок 1:*** Визначимо основні дії

***Крок 2:*** Конкретизуємо дію знаходження суми дільників числа *k*

***Крок 3:*** Конкретизуємо діювизначення чи є поточне значення *div* дільником числа *k* та знаходження суми

**Підпрограма *solution():***

***Крок 1:*** Визначимо основні дії

***Крок 2:*** Конкретизуємо дію знаходження числа з максимальною сумою дільників з поміж попередніх ітерацій

**Псевдокод**

**Основна програма:**

***Крок 1:* Початок**

*input();*

*solution();*

*output();*

**Кінець**

**Підпрограма** *input()***:**

***Крок 1:* Початок** *input()*

**Введення** *n***;**

**Кінець** *input(n)*

**Підпрограма** *output()***:**

***Крок 1:* Початок** *output()*

**Виведення** *max\_k, S\_div\_max;*

**Кінець** *output()*

**Підпрограма** *Sum\_divisors(k):*

***Крок 1:* Початок** *Sum\_divisors(k)*

*S\_div = 0;*

Знаходження суми дільників числа *k;*

**Кінець** *Sum\_divisors(k)*

***Крок 2******:* Початок** *Sum\_divisors(k)*

*S\_div = 0;*

**Повторити для** *div* **від** 1 **до** *k:*

Визначення чи є поточне значення *div* дільником числа *k* та знаходження суми;

**Все повторити**

**Кінець** *Sum\_divisors(k)*

***Крок 3:*** **Початок** *Sum\_divisors(k)*

*S\_div = 0;*

**Повторити для** *div* **від** 1 **до** *k:*

**Якщо** *k % div == 0* **то:**

*S\_div += div;*

**Все якщо**

**Все повторити**

**Кінець** *Sum\_divisors(k)*

**Підпрограма solution(n):**

***Крок 1:*** **Початок** *solution(n)*

**Якщо** *n = 0* **то:**

**Повернути;**

**Все якщо**

**Виведення** *n, S\_div;*

Визначеняя числа з максимальною сумою дільників з поміж попередніх ітерацій;

***Solution(n-1);***

**Кінець** *solution(n)*

***Крок 2:* Початок** *solution(n)*

**Якщо** *n = 0* **то:**

**Повернути;**

**Все якщо**

**Виведення** *n, S\_div;*

**Якщо** *S\_div > max\_S\_div* **то:**

*max\_n = n;*

*max\_S\_div = S\_div;*

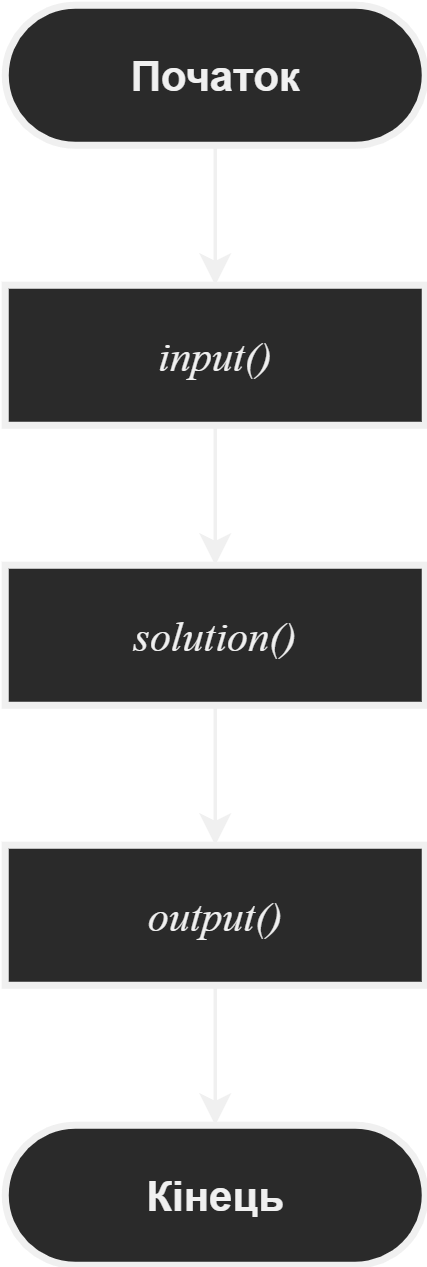
**Все якщо**

***Solution(n-1);***

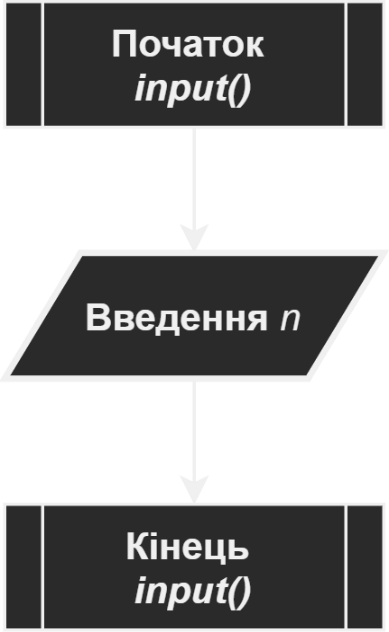
**Кінець** *solution(n)*

**Блок-схеми**

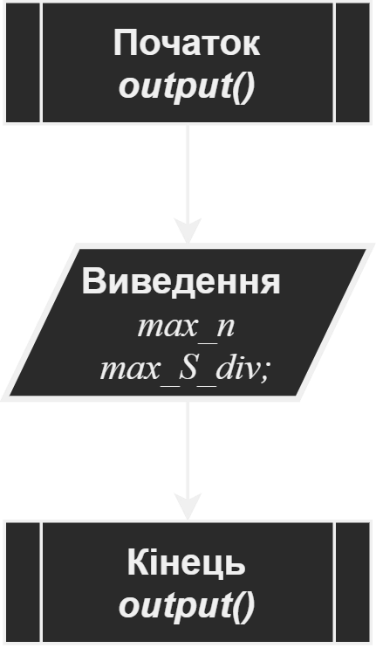
**Основна програма**

******

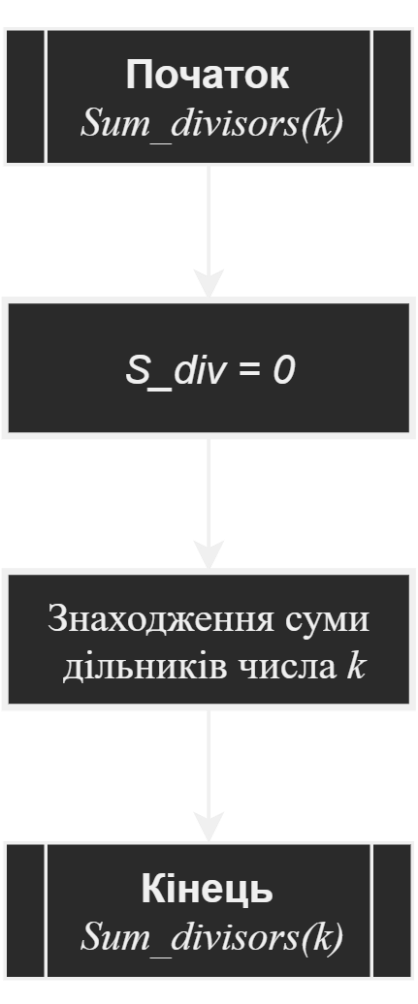
**Підпрограма *input(n)***

****

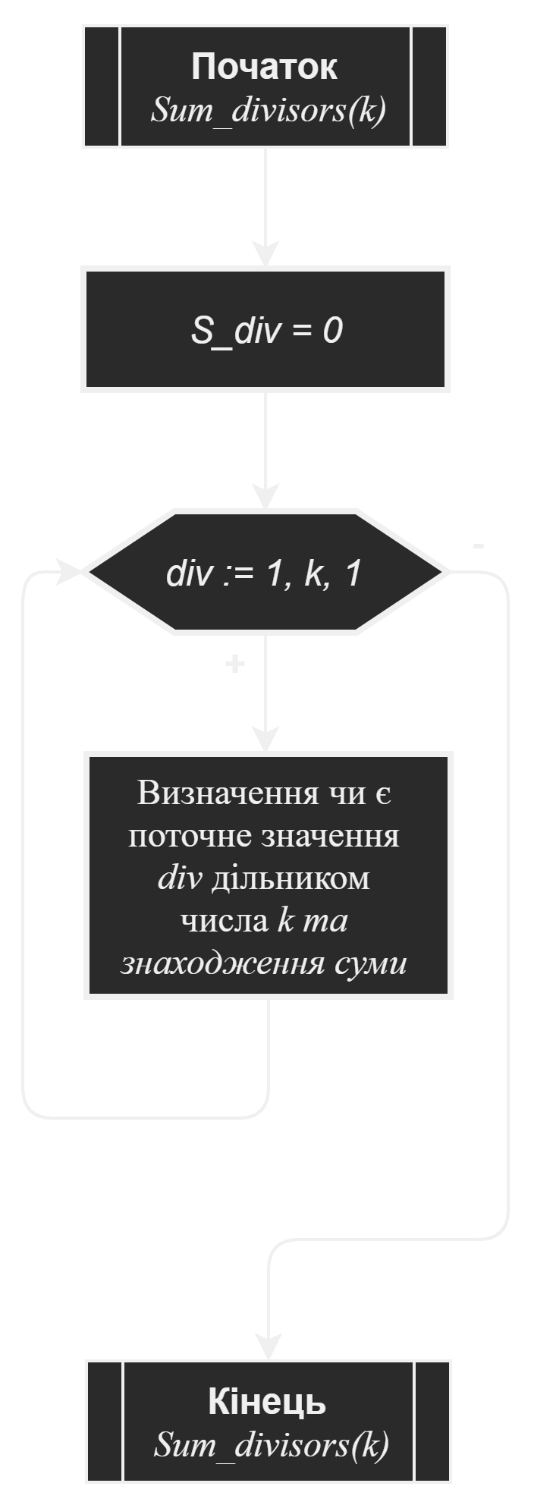
**Підпрограма *output()***

******

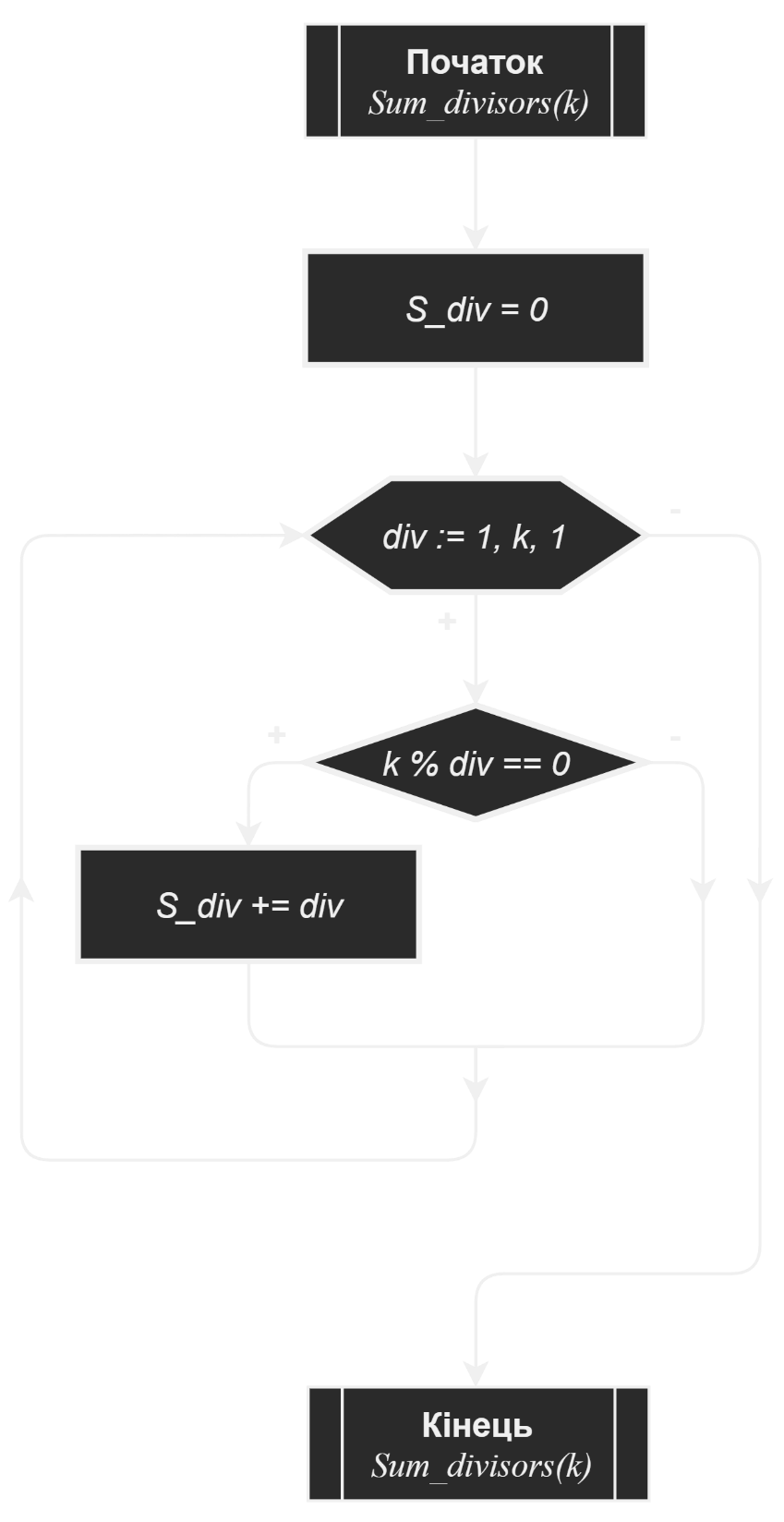
**Підпрограма *Sum\_divisors(k) - крок 1***

******

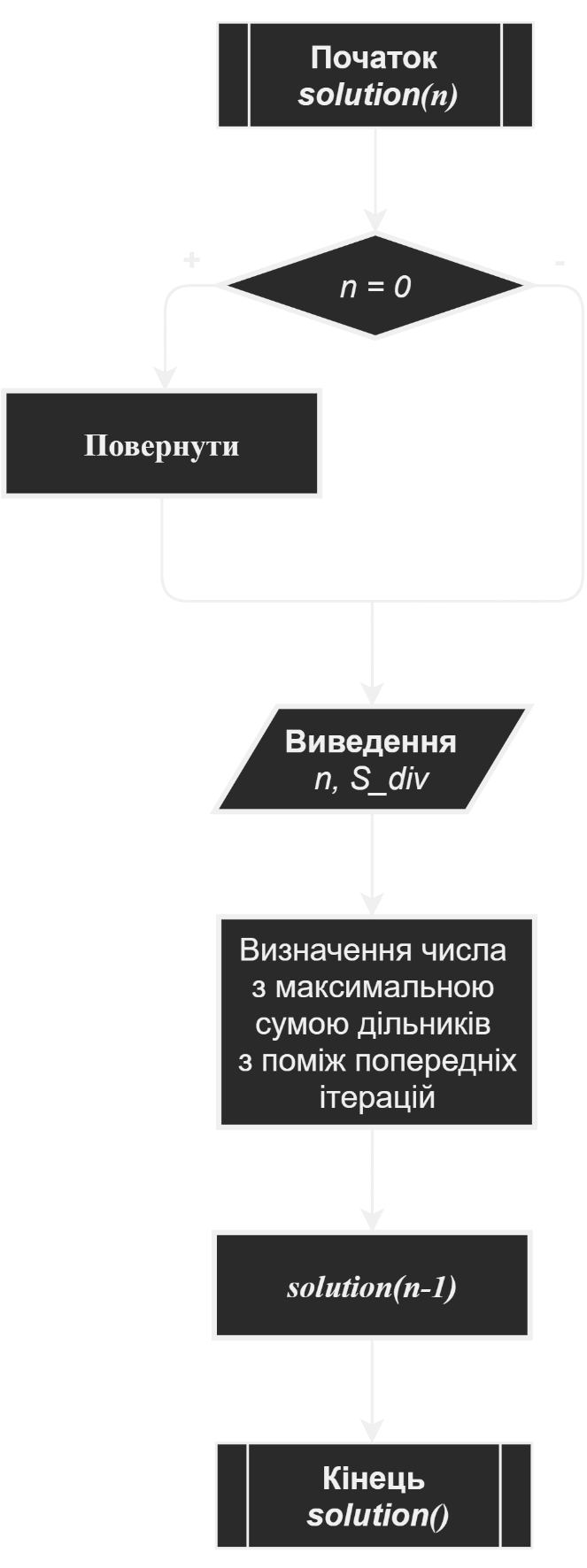
**Підпрограма *Sum\_divisors(k) - крок 2***

******

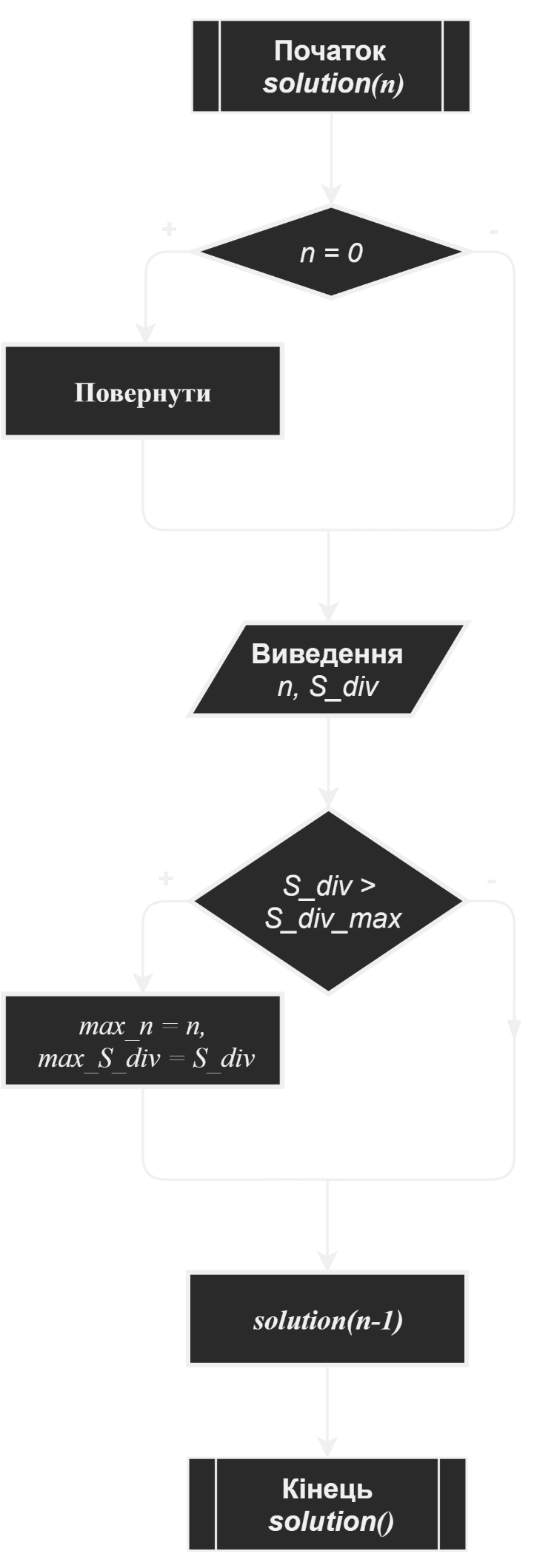
***Підпрограма Sum\_divisors(k) - крок 3***

******

**Підпрограма *solution(n) - крок 1***

******

**Підпрограма *solution(n) - крок 1***

******

**Код програми на мові С++**

#include <iostream>

using namespace std;

int n;

int max\_S\_div = 0, max\_n = 1;

void input();

void solution(int n);

void output();

int Sum\_divisors(int);

int main() {

input();

solution(n);

output();

system("pause>0");

return 0;

}

void input() {

cout << "Enter n: ";

cin >> n;

}

void output() {

cout << "The value between 1 and n with maximum sum of divisors is: " << max\_n << endl;

cout << "Its sum of divisors is: " << max\_S\_div;

}

int Sum\_divisors(int k) {

int S\_div = 0;

for (int div = 1; div <= k; div++) {

if (k % div == 0) S\_div += div;

}

return S\_div;

}

void solution(int n) {

if (n == 0) return;

cout << "n = " << n << ": S\_div = " << Sum\_divisors(n) << endl;

if (Sum\_divisors(n) > max\_S\_div) {

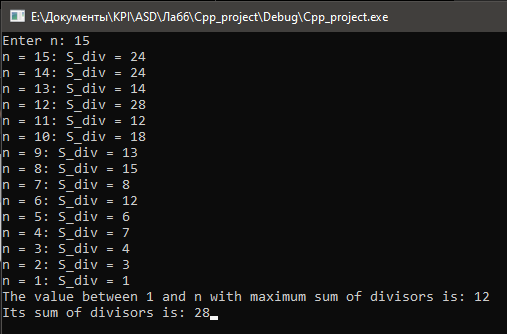
max\_n = n;

max\_S\_div = Sum\_divisors(n);

}

solution(n - 1);

}

****

**Тестування програми**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | *n = 1* |
| 2 | S\_div = 0 |
| 3 | div = 1 |
| 4 | 1 % 1 = 0 |
| 5 | S\_div = 1 |
| 6 | S\_div > S\_div\_max |
| 7 | max\_n = 1 |
| 8 | max\_S\_div = 1 |
| 9 | n = 2 |
| 10 | S\_div = 0 |
| 11 | div = 1 |
| 12 | 2 % 1 = 0 |
| 13 | S\_div = 1 |
| 14 | 2 % 2 = 0 |
| 15 | S\_div = 3 |
| 16 | S\_div > max\_S\_div |
| 17 | max\_n = 2 |
| 18 | max\_S\_div = 3 |
| 19 | n = 3 |
| 20 | S\_div = 0 |
| 21 | div = 1 |
| 22 | 3 % 1 = 0 |
| 23 | S\_div = 1 |
| 24 | 3 % 2 = 1 |
| 25 | div = 3 |
| 26 | 3 % 3 = 0 |
| 27 | S\_div = 4 |
| 28 | S\_div > max\_S\_div |
| 29 | max\_n = 3 |
| 30 | max\_S\_div = 4 |
|  | Кінець |

**Висновок**

Ми використали декілька функції типу void для введенння та виведення даних та резульатів. Також ми використали рекурсивну функцію типу int для знайдення суми дільників з формальним параметром k. У цій функції ми використали ключове слово return для повернення результатів обчислення, у данному випадку суму дільників числа *k*. *Solution(n) -* рекурентна функція, з умовою повернення *n = 0*. Основна програма включала в себе функції введення, виведення та функції *Sum\_divisors(k), solution(n)*. Отже, ми дослідили особливості роботи рекурсивних алгоритмів та набули практичних навичок їх використання під час складання програмних специфікацій підпрограм.