**Алгоритми та структури даних. Основи алгоритмізації**

*Додаток 1*

Міністерство освіти і науки України

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

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації

і управління

Звіт

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження лінійних алгоритмів»

Варіант\_\_\_21\_\_

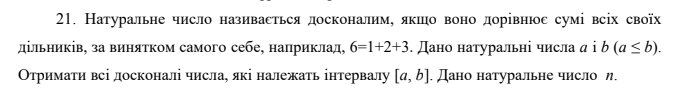
Виконав студент \_\_\_Мартаков Данило Кирилович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 (шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 ( прізвище, ім'я, по батькові) 

Київ 2021

**Назва роботи:** Дослідження складних циклічних алгоритмів  
**Мета:** дослідити особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм..  
**Варіант**: 21  
**Умова задачі**: 

**Постановка задачі:** Результатом буде усі досконалі числа у певному інтервалі

Досконалим називається число, яке дорівнює сумі своїх дільників (за винятком самого себе)

Інших початкових даних для розв’язку не потрібно.  
**Побудова математичної моделі:** складемо таблицю імен змінних.

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Нижня Границя | Цілий | a | Початкове дане |
| Верхня Границя | Цілий | b | Початкове дане |
| Сума | Натуральне число | sum | Тимчасове допоміжне дане |
| Основна функція знаходження досконалих чисел | Функція | FindAllDosk(lowLimit, upLimit) | Основна функція/ результат |
| Функція Перевірки числа на досконалість | Функція | FindDosk(number, sum, current) | Допоміжна функція |

**Крок 1: Визначимо основні дії  
Крок 2: Деталізуємо ввід початкових даних  
Крок 3: Деталізуємо тіло та умови основної функції FindAllDosk – перебір чисел та їх перевірка на досконалість  
Крок 4: Деталізуємо рекурсію функції FindDosk – перевірки чи є число досконалим**

**Псевдокод**

Крок 1

Введеня проміжку

Розрахувати досконалі числа у проміжку

Кінець

Крок 2

Ввести a, b

FindAllDosk(a, b)

Кінець

Крок 3 (псевдокод функції FindAllDosk(lowLImit, upLimit))

для i = lowLimit; i < upLimit; i++

sum = 0

sum = FindDosk(i, sum, 1)

якщо sum == i

Вивести sum

все якщо

все для

кінець FindAllDosk

Крок 4 (псевдокод фуекції FindDosk(number, sum, current))

якщо current < number

якщо number % current == 0

sum += current

все якщо

повернути FindDosk(number, sum, current + 1)

все якщо

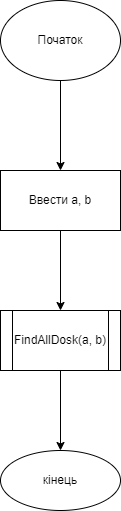
інакше

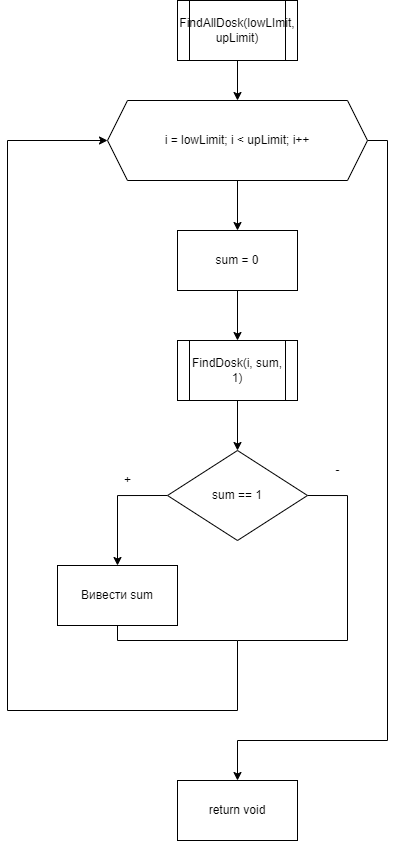
повернути sum

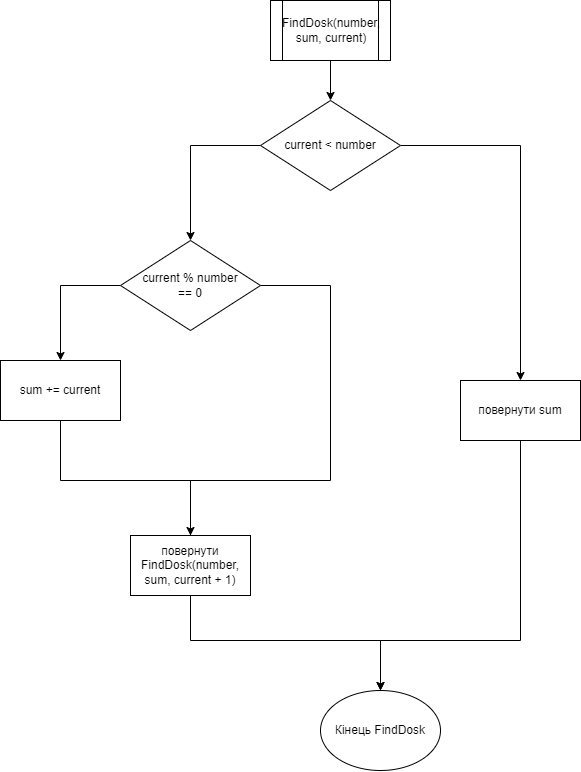
все інакше

кінець FindDosk

**Блок Схема**





**Код Алгоритму**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace asdlab6

{

class Program

{

static void Main(string[] args)

{

int a = Convert.ToInt32(Console.ReadLine());

int b = Convert.ToInt32(Console.ReadLine());

FindAllDosk(a, b);

Console.ReadLine();

}

public static void FindAllDosk(int lowLimit, int upLimit)

{

for (int i = lowLimit; i < upLimit; i++)

{

int sum = 0;

sum = FindDosk(i, sum, 1);

if (sum == i)

Console.WriteLine(i);

}

}

public static int FindDosk(int number, int sum, int current)

{

if (current < number)

{

if (number % current == 0)

sum += current;

return FindDosk(number, sum, current + 1);

}

else

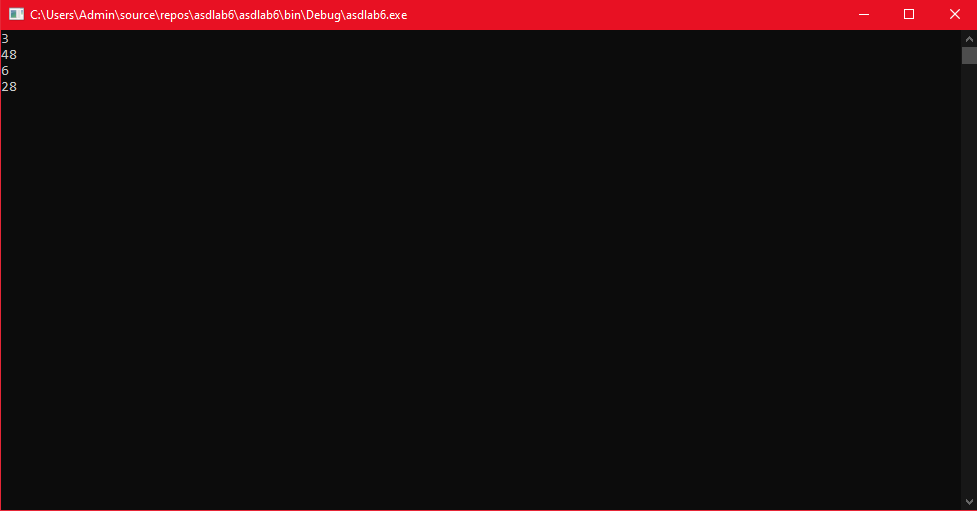
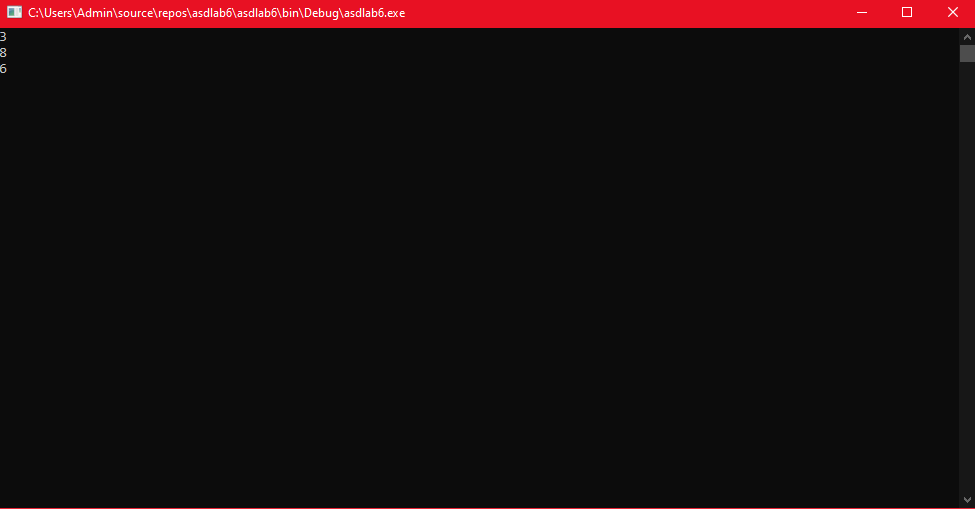
return sum;

}

}

}

**Випробування алгоритму**

****

1)Введеня a = 3, b = 48  
Досконалі числа у проміжку – 6, 28

2) Введеня a = 3, b = 8  
Досконалі числа у проміжку – 6

**Висновок**

Було досліджено особливості роботи рекурсивних алгоритмів та набуто практичних навичок їх використання під час складання програмної специфікації підпрограми. Для практики була дана задача на знаходження усіх досконалих чисел у проміжку, для якої було написано програму на мові с# з використанням рекурсії.