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 № 6 з дисципліни  
«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 24

Виконав студент ІП-11 Печковський Олександр Костянтинович

(шифр, прізвище, ім'я, по батькові)

Мартинова Оксана Петрівна

Перевірила

(прізвище, ім'я, по батькові)

Київ 2021

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

**Умова задачі:** визначити перші 8 п’ятизначних паліндромів, що не є простими числами.

Постановка задачі:

**Підпрограма перевірки числа на простоту prime(number, i):**

1. якщо i<math.ceil(((number)\*\*(1/2))+1), тобто лічильник i менший за округлене до цілого значення (((number)\*\*(1/2))+1), то виконуємо:
2. якщо number%i==0 (остача від ділення числа на лічильник i дорівнює 0), то:
3. повернути False (бо число не є простим, адже воно ділиться націло на i)

(2) інакше:

(3) i=i+1

(3) повернути prime(number, i) – звертаємось до цієї ж підпрограми з більшим на одиницю значенням лічильника i

(1) інакше:

(2) повернути True (якщо дійшло до цього моменту, то число просте)

**Основна програма:**

Для k в діапазоні (10000, 99999) виконуємо такі дії:

(1) якщо лічильник кількості непростих п’ятизначних паліндромів a<8, це означає, що нам потрібно шукати ще такі паліндроми, тоді перевіряємо, чи є число k паліндромом:

(2) якщо виконуються умова (k//10000)=(k%10) та умова ((k//1000)%10)=((k%100)//10), це означає, що число є паліндромом, тоді:

(3) якщо не prime(k, 2), тобто число не є простим, то:

(4) виводимо на екран значення k (значення п’ятизначного числа, яке пройшло всі перевірки)

(4) до значення a додаємо 1 (збільшуємо лічильник кількості непростих п’ятизначних паліндромів)

(1) до значення k додаємо 1 (збільшуємо п’ятизначне число на 1 для його перевірки в наступній ітерації)

Математична модель:

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Лічильник підпрограми для перевірки числа на простоту | Натуральний | i | Допоміжна змінна |
| П’ятизначний паліндром, який перевіряється в підпрограмі | Натуральний | number | Допоміжна змінна |
| Лічильник кількості непростих п’ятизначних паліндромів | Натуральний | a | Допоміжна змінна |
| Лічильник зовнішнього циклу перевірки усіх п’ятизначних чисел | Натуральний | k | Допоміжна змінна |
| Умова простоти числа | Логічний | IsPrime | Допоміжна змінна |

Крок 1. Визначення основних дій

Крок 2. Деталізація дії знаходження та перебору паліндромів

Крок 3. Деталізація дії перевірки простоти числа за допомогою рекурсії

**Псевдокод:**

*Крок 1.*

**Підпрограма prime (number, i):**

Визначення простоти числа за допомогою рекурсії

**Початок**

Знаходження та перебір паліндромів

**Кінець**

*Крок 2.*

**Підпрограма prime (number, i):**

Визначення простоти числа за допомогою рекурсії

**Початок**

1. a=0, k=0
2. Для k в діапазоні (10000, 99999):
   1. . Якщо a<8:
      1. Якщо (k//10000)=(k%10) та ((k//1000)%10)=((k%100)//10):
         1. Якщо не prime(k, 2):
            1. Виведення k
            2. a+=1
   2. . k+=1

**Кінець**

*Крок 3.*

**Підпрограма** **prime (number, i):**

1. Якщо i<math.ceil(((number)\*\*(1/2))+1):

1.1. Якщо number % i == 0:

1.1.1. повернути False

1.1. Інакше:

1.1.1. i = i + 1

1.1.2. Повернути prime(number, i)

1. Інакше:

1.1. Повернути True

**Початок**

1. a=0, k=0
2. Для k в діапазоні (10000, 99999):
   1. . Якщо a<8:
      1. Якщо (k//10000)=(k%10) та ((k//1000)%10)=((k%100)//10):
         1. Якщо не prime(k, 2):
            1. Виведення k
            2. a+=1
   2. . k+=1

**Кінець**

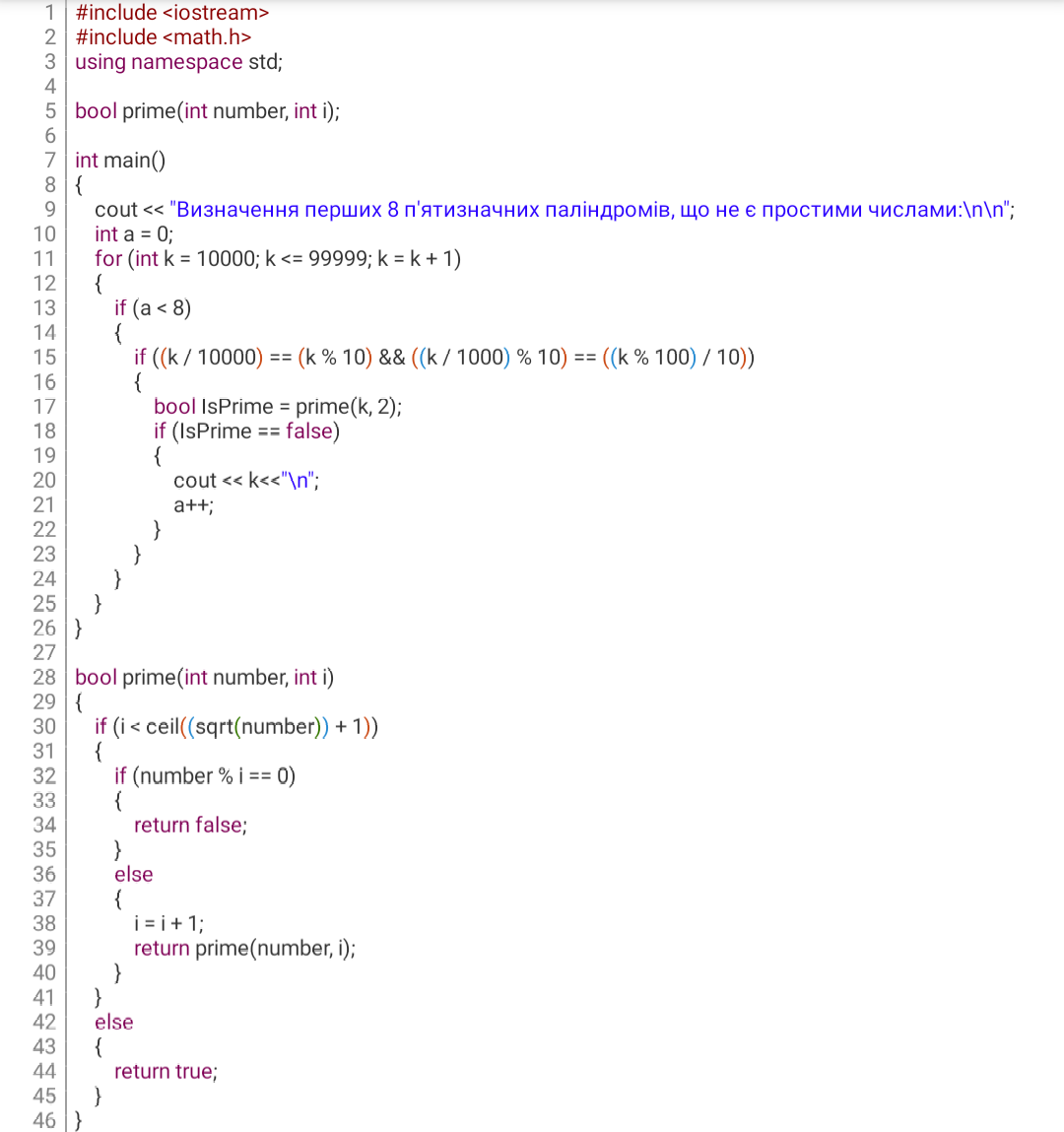
**Блок-схема:**



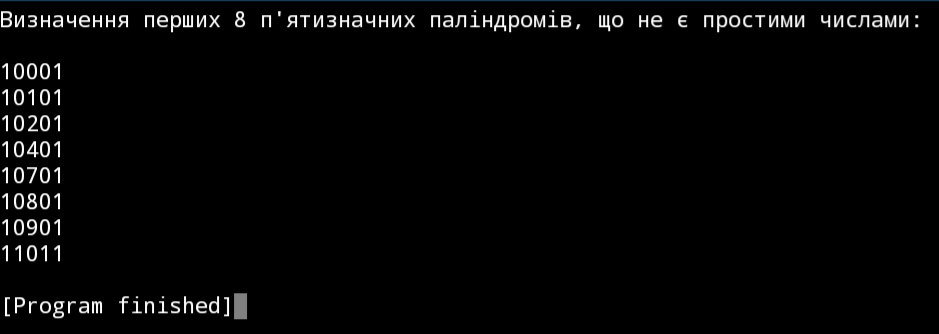




Код програми:



Робота програми:



Випробування алгоритму:

|  |  |
| --- | --- |
| Блок | Дія |
|  | Підпрограма prime (number, i) |
| 1 | Результати перевірки чисел на простоту:  Повертає True для таких значень number:   * 10301 * 10501 * 10601   Повертає False для таких значень number:   * 10001 * 10101 * 10201 * 10401 * 10701 * 10801 * 10901 * 11011 |
|  | Початок |
| 1 | Коментар: У 1 ітерації перевіряється число 10000, у 2 – 10001 і так далі. Виводяться на екран ті числа, які пройшли усі перевірки.  Вивід 2 ітерації: «10001» - є непростим п’ятизначним паліндромом  Вивід 102 ітерації: «10101» - є непростим п’ятизначним паліндромом  Вивід 202 ітерації: «10201» - є непростим п’ятизначним паліндромом  Вивід 402 ітерації: «10401» - є непростим п’ятизначним паліндромом  Вивід 702 ітерації: «10701» - є непростим п’ятизначним паліндромом  Вивід 802 ітерації: «10801» - є непростим п’ятизначним паліндромом  Вивід 902 ітерації: «10901» - є непростим п’ятизначним паліндромом  Вивід 1012 ітерації: «11011» - є непростим п’ятизначним паліндромом |
|  | Кінець |

Висновок: виконуючи лабораторну роботу, я набув практичних навичок використання рекурсивних алгоритмів під час складання програмних специфікацій. Я навчився використовувати рекурсію, описувати її у псевдокоді та позначати на блок-схемі.