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 №5 з дисципліни

«Основи програмування»

«Організація циклічних процесів. Складні цикли»

Варіант 34

Виконав студент ІП-1134 Шамков Іван Дмитрович
(прізвище, ім'я, по батькові)

Перевірив викладач Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота №5

Організація циклічних процесів. Складні цикли

Лабораторна робота 5

Організація циклічних процесів. Складні цикли

Мета – вивчити особливості організації складних циклів.

Варіант: 34

Умова задачі:

34. Визначити *n* перших простих чисел.

Математична модель:

Змінна	Тип	Ім’я	Призначення
Кількість простих чисел	Цілий	N	Початкове дане
Число, яке перевіряємо	Цілий	num_now	Проміжне значення
Число, на яке ділимо	Цілий	dil	Проміжне значення
Кількість уже виведених простих чисел	Цілий	kilk	Проміжне значення
Збережене значення кількості виведених простих чисел	Цілий	kilk1	Проміжне значення
Логічна змінна для перевірки умови ділення націло	Логічний	Bool	Проміжне значення

Постановка задачі:

Отже, математичне формулювання нашої задачі полягає в тому, щоб отримати значення n, яке є кількістю простих чисел, що нам потрібно вивести. Через арифметичний цикл ми пробігаємо по значенням непарних чисел, які перевіряємо, чи є вони простими.

Для кожного такого числа працює ітераційний цикл, який перевіряє дві умови:

1) Чи ділиться число націло?
2) Чи є дільник меншим за остачу від ділення?

Якщо перша умова ні одного разу не виконується для перевіряємого числа, то рано чи пізно виконається друга умова. У такому випадку наше число буде простим. Усе через те, що ми перевірили усі можливі варіанти утворення числа. Розглянемо це на прикладі числа 30 та простого числа 131:

30/2=15; 30/3=10; 30/5=6; 30/6=5; 30/10=3; 30/15=2;

Як бачимо, дільники та повторилися з результатами від ділення. Числа 5 та 6 є перехідною точкою. Так, щоб утворити число 30, достатньо помножити деяке число з проміжку (0;6) на інше певне число з проміжку [6;30]. Це працює в обидві сторони.

Тепер розглянемо число 131. Перевіримо, чи буде воно простим.

131/3=43.6 3<43.6

131/5=26.2 5<26.2

131/7=18.7 7<18.7

131/9=14.5 9<14.5

131/11=11.9 11<11.9

131/13=10.07 13>10.07

Виконавши ці дії ми точно можемо сказати, що 131 - просте число, адже воно може утворитися з чисел, одне з яких належить проміжку (0;13), а інше - 13;131. Серед першого проміжку ми не зустрілили ні одного цілого числа, що поділило б 131 націло.

Псевдокод:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо значення N.

Крок 3. Перевіряємо, чи N==1.

Крок 4. Перевіряємо, чи N >=2.

Крок 5.Знаходимо прості числа.

Крок 1:

Почато

Деталізуємо значення N

Перевірка значення N

Знаходимо прості числа

Кінець

Крок 2:

Початок

Введення N

Перевірка значення N

Знаходимо прості числа

Кінець

Крок 3:

Початок

Введення N

якщо N==1

то

kilk=1

Виведення 2

Перевірка, чи N>=2

все якщо

Знаходимо прості числа

Кінець

Крок 4:

Початок

Введення N

якщо N==1

то

kilk=1

Виведення 2

інакше якщо N>=2

то

kilk=2

Виведення 2 та 3

все якщо

Знаходимо прості числа

Кінець

Крок 5:

Початок

Введення N

якщо N==1

то

kilk=1

Виведення 2

інакше якщо N>=2

то

kilk=2

Виведення 2 та 3

все якщо

Для num_now від 5 з кроком 2 поки kilk!=n

dil=3

Bool=True

kilk1=kilk

поки Bool та kilk==kilk1

то

якщо num_now/dil != num_now//dil

то

Bool=False

інакше якщо num_now/dil<dil

то

kilk+=1

Виведення num_now

інакше

то

dil+=2

все якщо

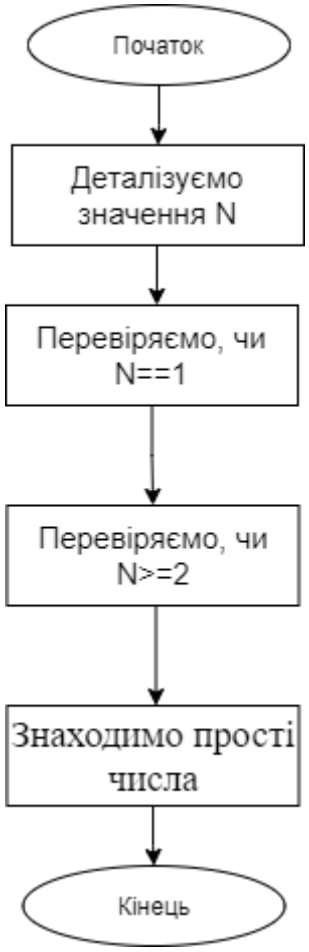
все повторити

все повторити

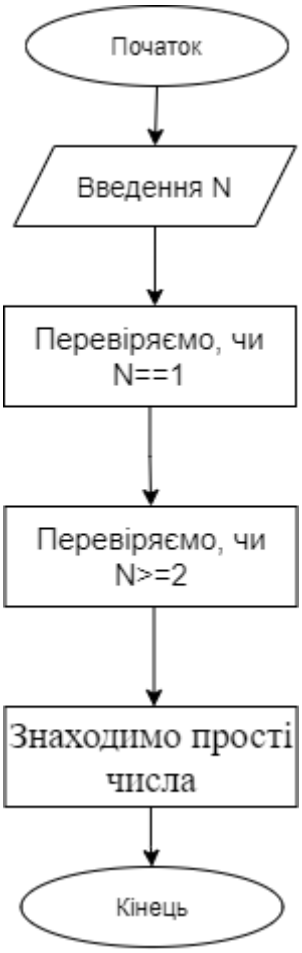
Кінець

Блок схема:

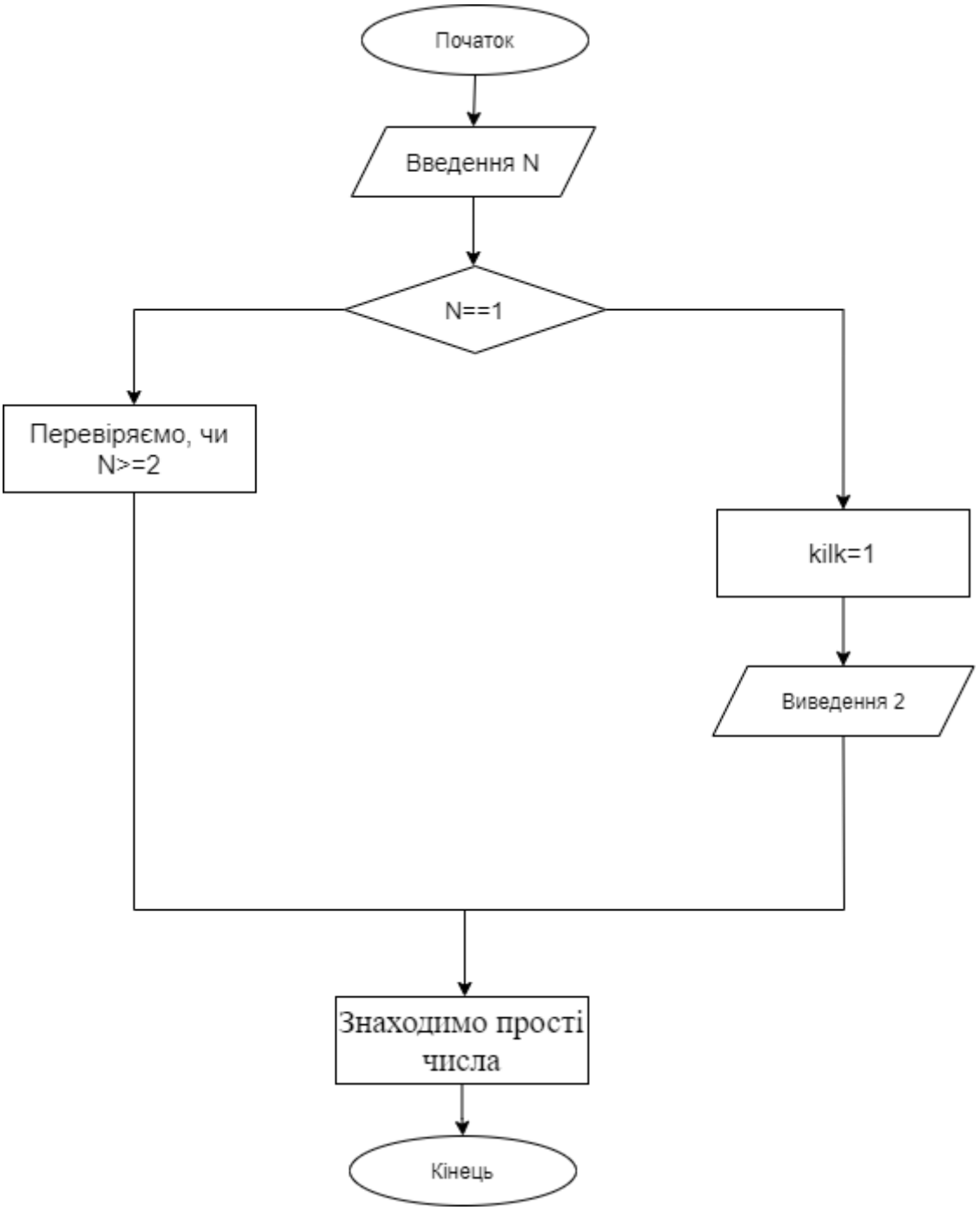
Крок 1



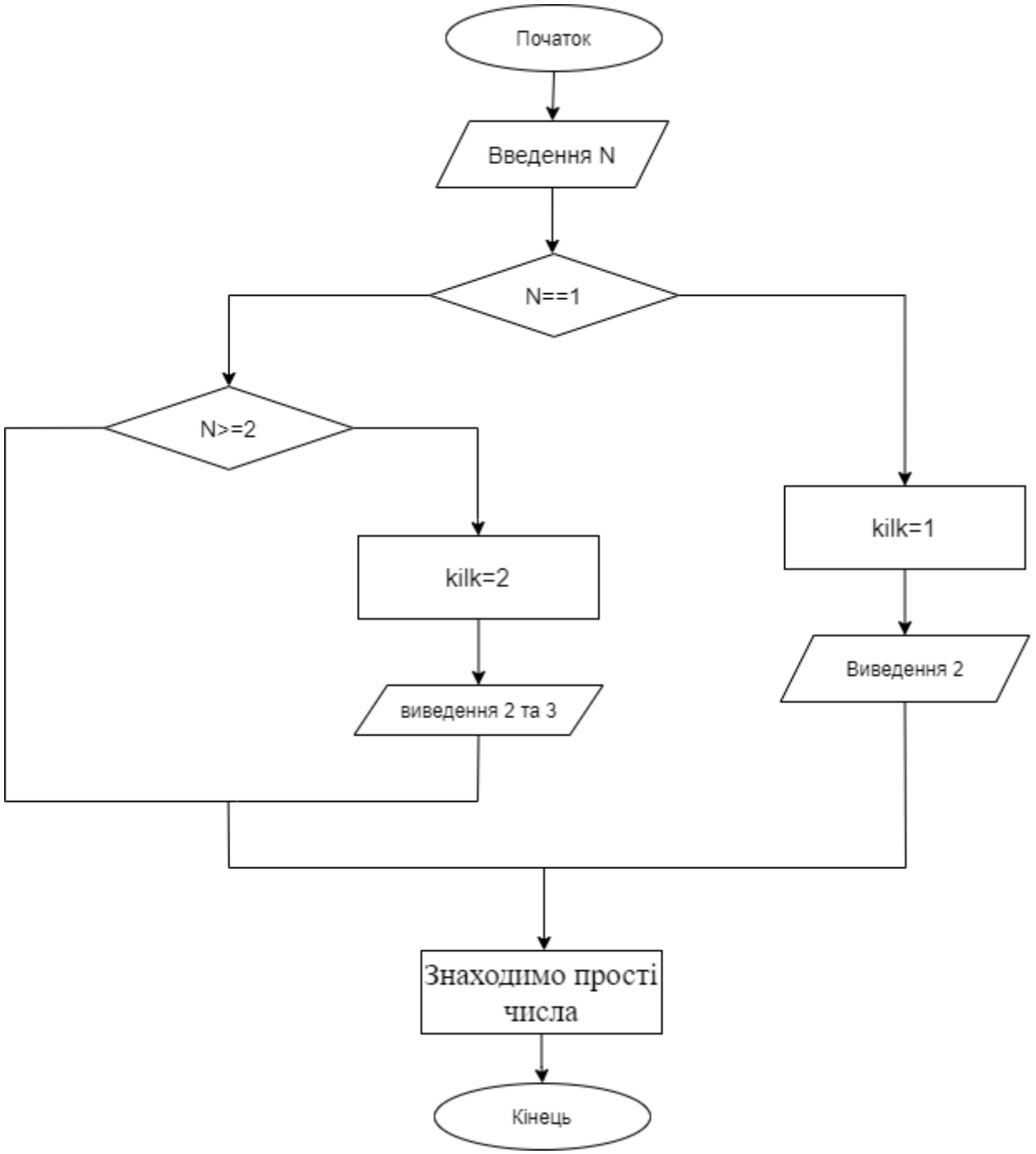
Крок 2


Початок
Введення N
Перевіряємо, чи N==1
Перевіряємо, чи N>=2
Знаходимо прості числа
Кінець

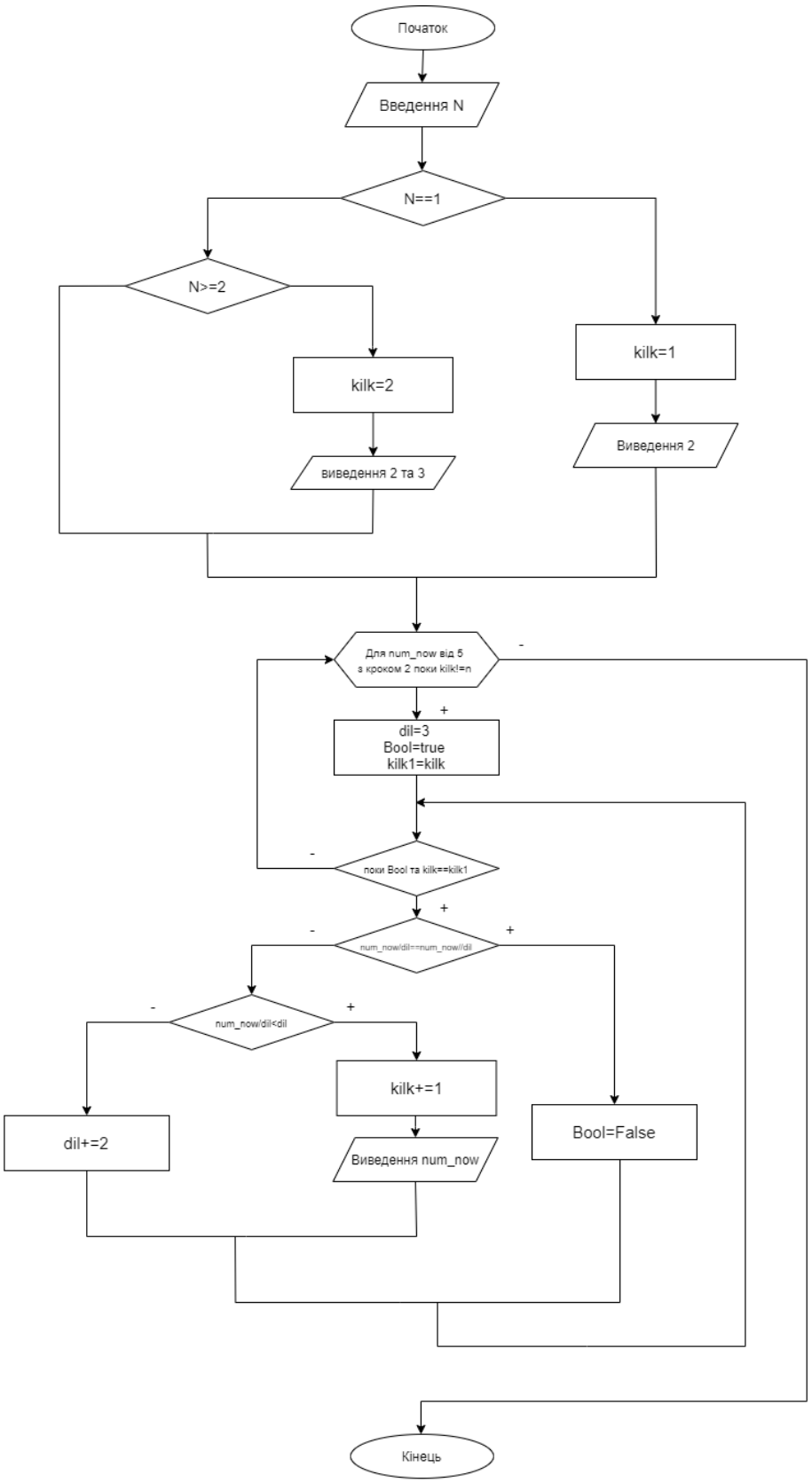
Крок 3


Початок
Введення N
N==1
Перевіряємо, чи N>=2
kilk=1
Виведення 2
Знаходимо прості числа
Кінець

Крок 4


Початок
Введення N
N==1
N>=2
kilk=2
виведення 2 та 3
kilk=1
Виведення 2
Знаходимо прості числа
Кінець

Крок 5


Початок
Введення N
N==1
N>=2
kilk=2
виведення 2 та 3
kilk=1
Виведення 2
Для num_now від 5
з кроком 2 поки kilk!=n
-
+
dil=3
Bool=true
kilk1=kilk
-
поки Bool та kilk==kilk1
+
-
num_now/dil==num_now//dil
+
-
num_now/dil<dil
+
dil+=2
kilk+=1
Виведення num_now
Bool=False
Кінець

Текст файла проекту:

Python:

```python
n=int(input("Enter quantity of prime numbers (n>=2)")) #Запит на кількість простих чисел
num_now=1#
if n==1:
    print("Prime number is 2")
    kilk=1
else:
    print("Prime number is 2") #Виводимо число 2 та 3 - перші прості числа
    print("Prime number is 3")
    kilk=2

while kilk<n:#Перевіряємо, чи є кількість виведених простих чисел меншою за задане
    num_now+=2
    dil=3
    kilk1=kilk#Записуємо останнє значення кількості виведених простих чисел в kilk1
    Bool=True
    while Bool and kilk==kilk1: #Перевіряємо значення булеової змінної та чи не змінилася кількість виведених простих чисел
        if num_now/dil == num_now//dil:#Перевіряємо, чи ділиться число націло. Якщо так, то виходимо з циклу
            Bool=False
        elif num_now/dil<dil:
            print("Prime number is", num_now)
            kilk+=1 #Виводимо просте число та збільшуємо кількість виведнеих чисел на 1
        else:
            dil+=2 #збільшуємо значення числа для перевірки ділення на 2
```

C++:

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n; //кількість простих чисел
    int kilk;//кількість виведених простих чисел
    int dil;//змінна для перевірки ділення
    bool Bool;

    cout << "enter quantity of prime numbers(n>=1(int))" << endl;//Запит на кількість простих чисел
    cin >> n;
    if (n == 1)
    {
        kilk = 1;
        cout << "Prime number is 2" << endl;
    }
    else if (n >= 2) {
        cout << "Prime number is 2" << endl;
        cout << "Prime number is 3" << endl;
        kilk = 2;
    }

    //cout << "Prime number is 2" << endl;//Виводимо число 2 та 3 - перші прості числа
    //cout << "Prime number is 3" << endl;//(це було зроблено покращення алгоритму та його пришвидшення. Але через ці модифікації він не здатен вивести ці числа самостійно)

    for (int num_now = 5; kilk != n; num_now += 2)//Перевіряємо, чи є числа num_now простими
    {
        dil = 3; //Задаємо числу для ділення значення три
        Bool = true;
        int kilk1 = kilk;//Записуємо останнє значення кількості виведених простих чисел в kilk1
        while (Bool && kilk==kilk1)//Перевіряємо значення булеової змінної та чи не змінилася кількість виведених простих чисел
        {
            if ((double(num_now) / dil) == (num_now / dil)) Bool = false; //Перевіряємо, чи ділиться число націло. Якщо так, то виходимо з циклу
            else if (num_now / dil < dil)                                 //Перевіряємо, чи результат ділення менший за дільник. Якщо так, то це і є наше просте число,
            {                                                             //адже до цього ми перевіряли, що воно не ділиться націло на попередні числа. Наприклад, число 131 - є простим. Після того, як ми поділимо його на 13:
                                                                          // 131 / 13 = 10.074, то нам немає сенсу далі ділити, адже наступні результати від ділення будуть повторювати попередні значення, які ми вже могли б отримати.

                cout << "prime number is " << num_now << endl;
                kilk += 1; // Виводимо просте число та збільшуємо кількість виведнеих чисел на 1
            }
            else dil += 2; // збільшуємо значення числа для перевірки ділення на 2
        }
    }
    return 0;

}
```

Копії екранних форм:

Python:

```
Enter quantity of prime numbers (n>=2)15
Prime number is 2
Prime number is 3
Prime number is 5
Prime number is 7
Prime number is 11
Prime number is 13
Prime number is 17
Prime number is 19
Prime number is 23
Prime number is 29
Prime number is 31
Prime number is 37
Prime number is 41
Prime number is 43
Prime number is 47
Press any key to continue . . .
```

C++:

```
enter quantity of prime numbers(n>=1(int))
15
Prime number is 2
Prime number is 3
prime number is 5
prime number is 7
prime number is 11
prime number is 13
prime number is 17
prime number is 19
prime number is 23
prime number is 29
prime number is 31
prime number is 37
prime number is 41
prime number is 43
prime number is 47
```

Випробування алгоритму

Блок	Дія
	Початок
1	Введення N=5, kilk=2
2	Цикл: для num_now від 5 з кроком 2 поки kilk!=N
3	Bool=true, dil=3, kilk1=kilk
4	Цикл: поки Bool та kilk1 == kilk=2
5	Якщо num_now/dil== num_now//dil: 5/3 не дорівнює 5//3
6	Якщо num_now/dil<dil: 5/3 менше за 3, тому це є просте число kilk+=1=3
7	Виходимо з ітераційного циклу, num_now+=2=7
8	Якщо num_now/dil== num_now//dil: 7/3 не дорівнює 7//3
9	Якщо num_now/dil<dil: 7/3 менше за 3, тому це є просте число kilk+=1=4
10	Виходимо з ітераційного циклу, num_now+=2=9
11	Якщо num_now/dil== num_now//dil: 9/3 дорівнює 9//3
12	Виходимо з ітераційного циклу, num_now+=2=11
13	Якщо num_now/dil== num_now//dil: 11/3 не дорівнює 11//3
14	Якщо num_now/dil<dil: 11/3 не менше за 3, тому це не просте число
15	dil+=2=5
16	Якщо num_now/dil== num_now//dil: 11/5 не дорівнює 11//5
17	Якщо num_now/dil<dil: 11/5 менше за 5, тому це просте число kilk+=1=5
	Кінець

Висновок

Отже, виконавши цю лабораторну роботу, ми навчилися створювати складні цикли. Проекти, на мою думку, розроблені коректно, адже заплановані елементи працюють, а саме: отримання даних від користувача, перевірка умови простоти числа за допомогою складного циклу та виведення такого числа. Алгоритм працює за принципом перебору непарних чисел та перевірки їхнього ділення на інші непарні числа. Перевіряємо, чи дільник менший за результат від ділення. Арифметичний цикл використовує лічильник num_now, який набуває значень непарних чисел до того моменту, поки не виведеться потрібна кількість простих чисел. Ітераційний цикл, що знаходиться в арифметичному, надає змінній dil значення непарних чисел, на які ділиться num_now. У такий спосіб ми перевіряємо, чи є простим число.

Підбираємо лише непарні числа, адже простим і парним числом є тільки двійка. У кінці кожного обрахунку виводимо просте число.