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

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження складних циклічних алгоритмів»

Варіант 25

Виконав студент ІП-15, Плугатирьов Дмитро Валерійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна

(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота № 5

Дослідження складних циклічних алгоритмів

Мета - дослідити особливості роботи складних циклів та набути практичних навичок їх використання під час складання програмних специфікацій.

Варіант 25

Завдання

25. Дано число a. Знайти найближче до нього просте число.

1. Постановка задачі

Користувач вводить певне число. Воно проходить перевірку на цілочисельність. Потім, створивши зовнішній цикл, умовою якого буде наявність результату відмінного від нуля у змінній результату, відбувається реалізація вкладених в нього двох циклів: перевірки цілих чисел справа та зліва від введеного раніше користувачем на їх простоту. Якщо обидва прості числа знаходяться на однаковій відстані від введеного користувачем, то вони виведуться в результаті як найближчі прості. Інакше, виведеться лише одне число.

Результатом виконання програми є значення найближчого простого числа або двох простих чисел.

2. Побудова математичної моделі

Змінна	*Тип*	*Ім'я*	*Призначення*
Дане число	Дійсний	a	Початкові дані
Число, лівіше за дане	Цілочисельний	lNum	Проміжні дані
Число, правіше за дане	Цілочисельний	rNum	Проміжні дані
Просте число	Цілочисельний	isResult	Результат
Простота лівішого числа	Логічний	isLPrime	Проміжні дані
Простота правішого числа	Логічний	isRPrime	Проміжні дані
Лічильник циклу знаходження лівішого числа	Цілочисельний	i	Проміжні дані
Лічильник циклу знаходження правішого числа	Цілочисельний	y	Проміжні дані

Кількість дільників лівішого числа	Цілочисельний	count	Проміжні дані
Кількість дільників правішого числа	Цілочисельний	count2	Проміжні дані

- реалізація процесу пошуку простих чисел починається із зовнішнього циклу, який виконує ітерації по власному тілу та припиняє свою роботу після отримання результату;
- завдяки перевіркам на знаки чисел, їх рівність певним значенням та спосіб ітерації по множині значень. Наприклад, поява мінуса в числі змусить проходити по множині від'ємних чисел за наявності потребуючої умови. Таким чином, будь-яке введене користувачем число є доцільним для прийняття участі у веденні розрахунків;
- прохід по множині чисел відбувається шляхом попарного виокремлення чисел з обох боків того числа, яке ввів користувач, з кроком розмірністю в одиницю (вліво для лівішого числа та вправо для правішого);
- ми можемо використовувати операцію віднімання за модулем з метою до виявлення найближчого до введеного числа з обох оцінюваних значень.

Дія floor(x) означає округлення числа x до меншого цілого.

Дія ceil(x) означає округлення числа x до більшого цілого.

Дія abs(x) означає взяття модуля від змінної з цілочисельним значенням.

Дія fabs(x) означає взяття модуля від змінної з дійсним значенням.

Дія (double)x означає приведення числа x до дійсного типу.

Дія (int)x означає приведення числа x до цілочисельного типу.

Дія x % v означає остачу від ділення числа x на число v.

3. Р о з в ' я з а н н я

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

Крок 1. Визначити основні дії.

Крок 2. Перевірити значення змінної на цілочисельність.

Крок 3. Відшукати значення простих чисел зліва та справа від введеного користувачем.

Крок 4. Визначити, яке просте число або числа будуть результатом виконання програми.

4. П с е в д о к о д

Крок 1

початок

перевірити значення змінної на цілочисельність

відшукати значення простих чисел зліва та справа від введеного користувачем

визначити, яке просте число або числа будуть результатом виконання програми

кінець

Крок 2

початок

isResult := 0

введення a

якщо ((double)a / (int)a) != 1 **та** a != 0

то

lNum := floor(a)

rNum := ceil(a)

інакше

lNum := a - 1

rNum := a + 1

все якщо

isLPrime := false

isRPrime := false

i := 0

y := 0

count := 0

count2 := 0

відшукати значення простих чисел зліва та справа від введеного користувачем

визначити, яке просте число або числа будуть результатом виконання програми

кінець

Крок 3

початок

isResult := 0

введення a

якщо ((double)a / (int)a) != 1 **та** a != 0

то

lNum := floor(a)

rNum := ceil(a)

інакше

lNum := a - 1

rNum := a + 1

все якщо

isLPrime := false

isRPrime := false

i := 0

y := 0

count := 0

count2 := 0

повторити

поки isResult == 0

повторити

поки ((i <= lNum **та** lNum > 0) **або** (i >= lNum **та** lNum < 0)) **або** ((y <= rNum **та** rNum > 0) **або** (y >= rNum **та** rNum < 0))

якщо i != 0 **та** (lNum % i) == 0

count := count + 1

якщо (count == 2 **та** i == lNum) **або** (count == 1 **та** abs(lNum) == 1)

```
                    то
                        isLPrime := true
                все якщо
            все якщо
            якщо lNum < 0
                то
                i := i – 1
            інакше якщо lNum > 0
                то
                i := i + 1
            все якщо
            якщо y != 0 та (rNum % y) == 0
                то
                count2 := count2 + 1
                якщо (count2 == 2 та y == rNum) або (count2 == 1 та
abs(rNum) == 1)
                    то
                        isRPrime := true
                все якщо
            все якщо
            якщо rNum < 0
                то
                y := y – 1
            інакше якщо rNum > 0
                то
                y := y + 1
            все якщо
        все повторити
        i := 0
        y := 0
```

count := 0

count2 := 0

визначити, яке просте число або числа будуть результатом виконання програми

все повторити

кінець

Крок 4

початок

isResult := 0

введення a

якщо ((double)a / (int)a) != 1 **та** a != 0

то

lNum := floor(a)

rNum := ceil(a)

інакше

lNum := a - 1

rNum := a + 1

все якщо

isLPrime := false

isRPrime := false

i := 0

y := 0

count := 0

count2 := 0

повторити

поки isResult == 0

повторити

поки ((i <= lNum **та** lNum > 0) **або** (i >= lNum **та** lNum < 0)) **або** ((y <= rNum **та** rNum > 0) **або** (y >= rNum **та** rNum < 0))

```
якщо i != 0 та (lNum %  i) == 0
        count := count + 1
        якщо (count == 2 та i == lNum) або (count == 1 та
abs(lNum) == 1)
                то
                isLPrime := true
        все якщо
все якщо
якщо lNum < 0
        то
        i := i – 1
інакше якщо lNum > 0
        то
        i := i + 1
все якщо
якщо y != 0 та (rNum % y) == 0
        то
        count2 := count2 + 1
        якщо (count2 == 2 та y == rNum) або (count2 == 1 та
abs(rNum) == 1)
                то
                isRPrime := true
        все якщо
все якщо
якщо rNum < 0
        то
        y := y – 1
інакше якщо rNum > 0
        то
        y := y + 1
все якщо
```

```
все повторити
i := 0
y := 0
count := 0
count2 := 0
якщо isLPrime == false та isRPrime == false
        то
        lNum := lNum - 1
        rNum := rNum + 1
інакше якщо (isLPrime == true та isRPrime == false ) або (isLPrime == false та isRPrime == true)
        то
        якщо isLPrime == true
                то
                isResult := lNum
                вивести isResult
        інакше якщо isRPrime == true
                то
                isResult := rNum
                вивести isResult
        все якщо
інакше якщо isLPrime == true та isRPrime == true
        то
        якщо ((double)a / (int)a) != 1 та a != 0
                то
                якщо fabs((double)rNum - a) < fabs((double)lNum - a)
                        то
                        isResult := rNum
                        вивести isResult
                інакше
```

isResult := lNum

вивести isResult

все якщо

інакше

isResult := lNum

вивести isResult

isResult := rNum

вивести isResult

все якщо

все якщо

isRPrime := false

isLPrime := false

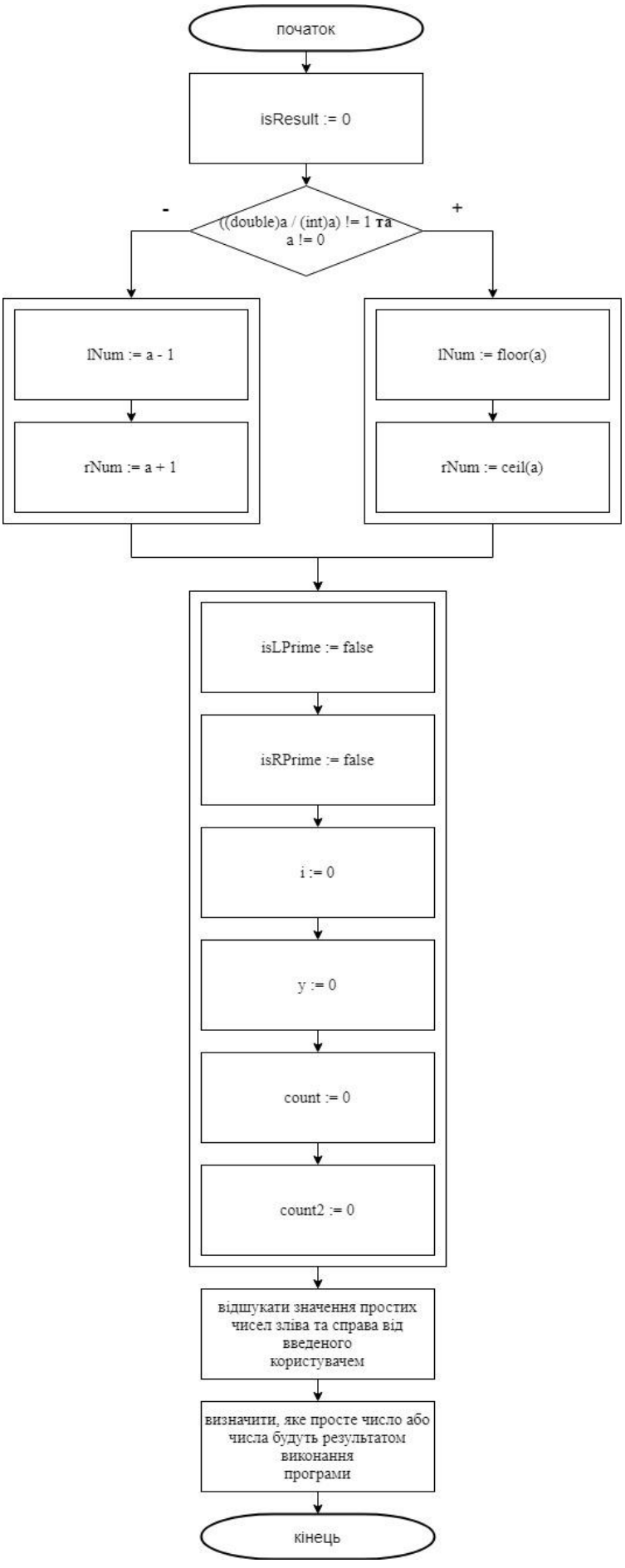
все повторити

кінець

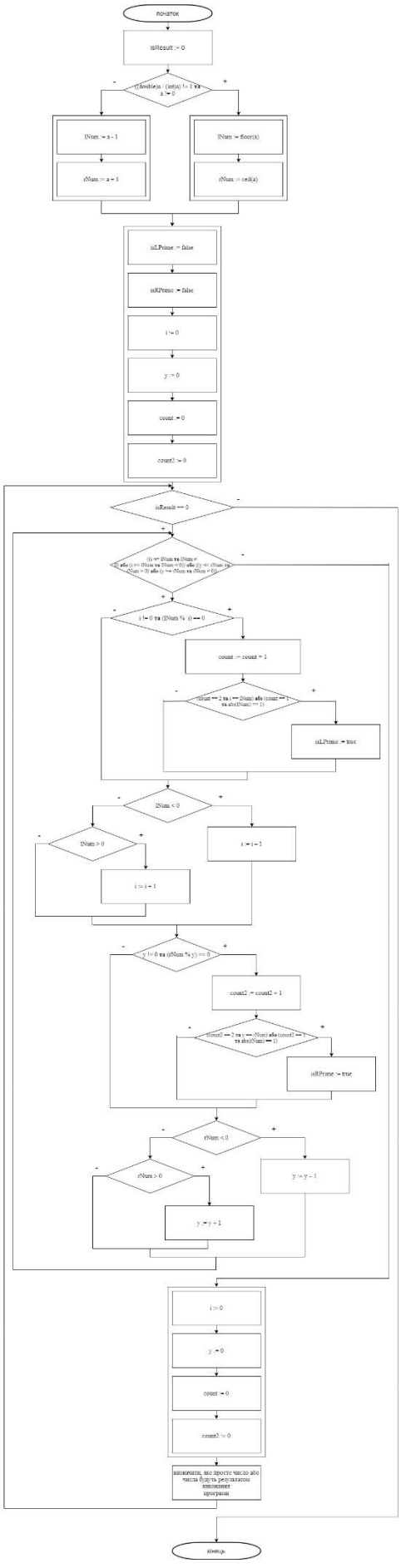
Блок-схема

Крок 1

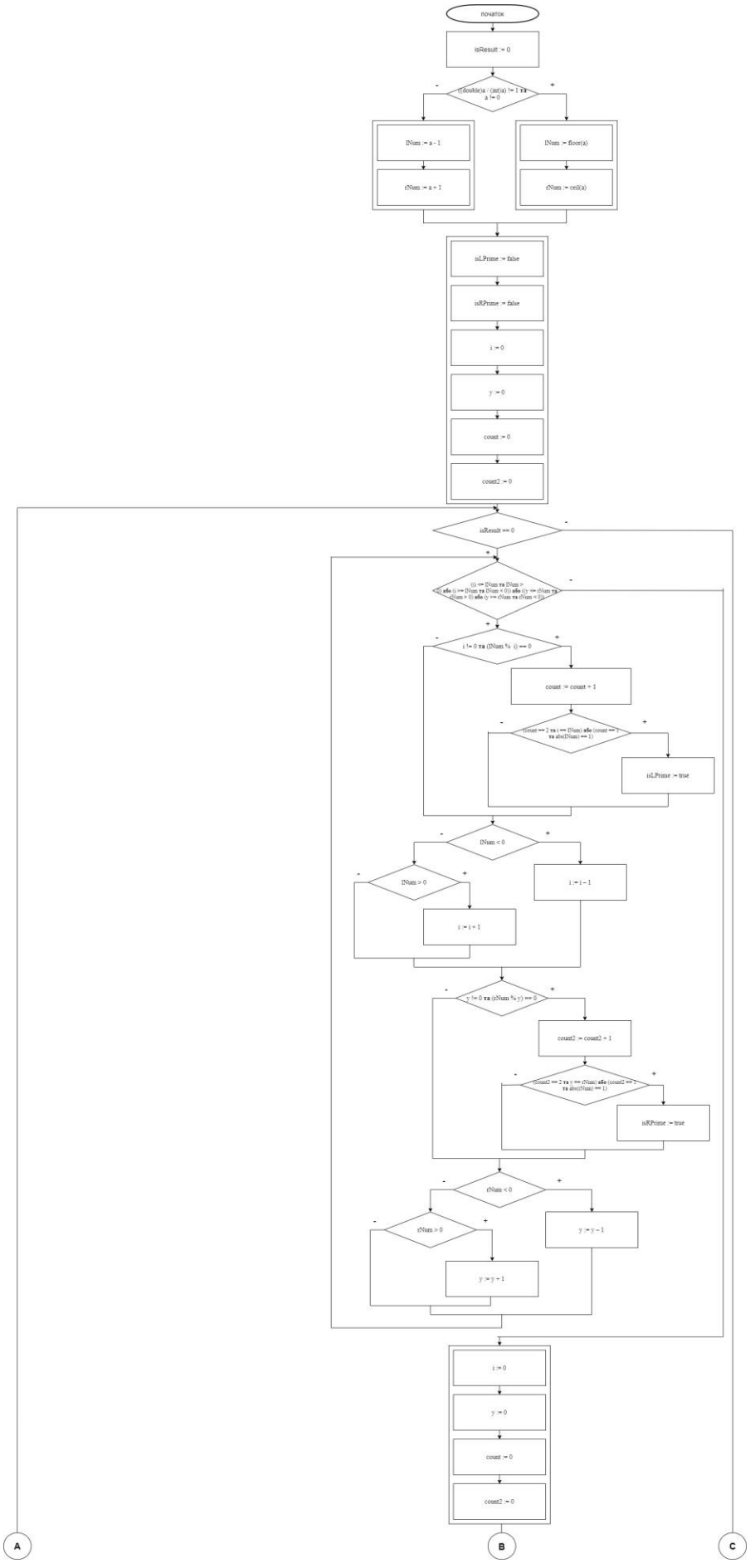
Крок 2

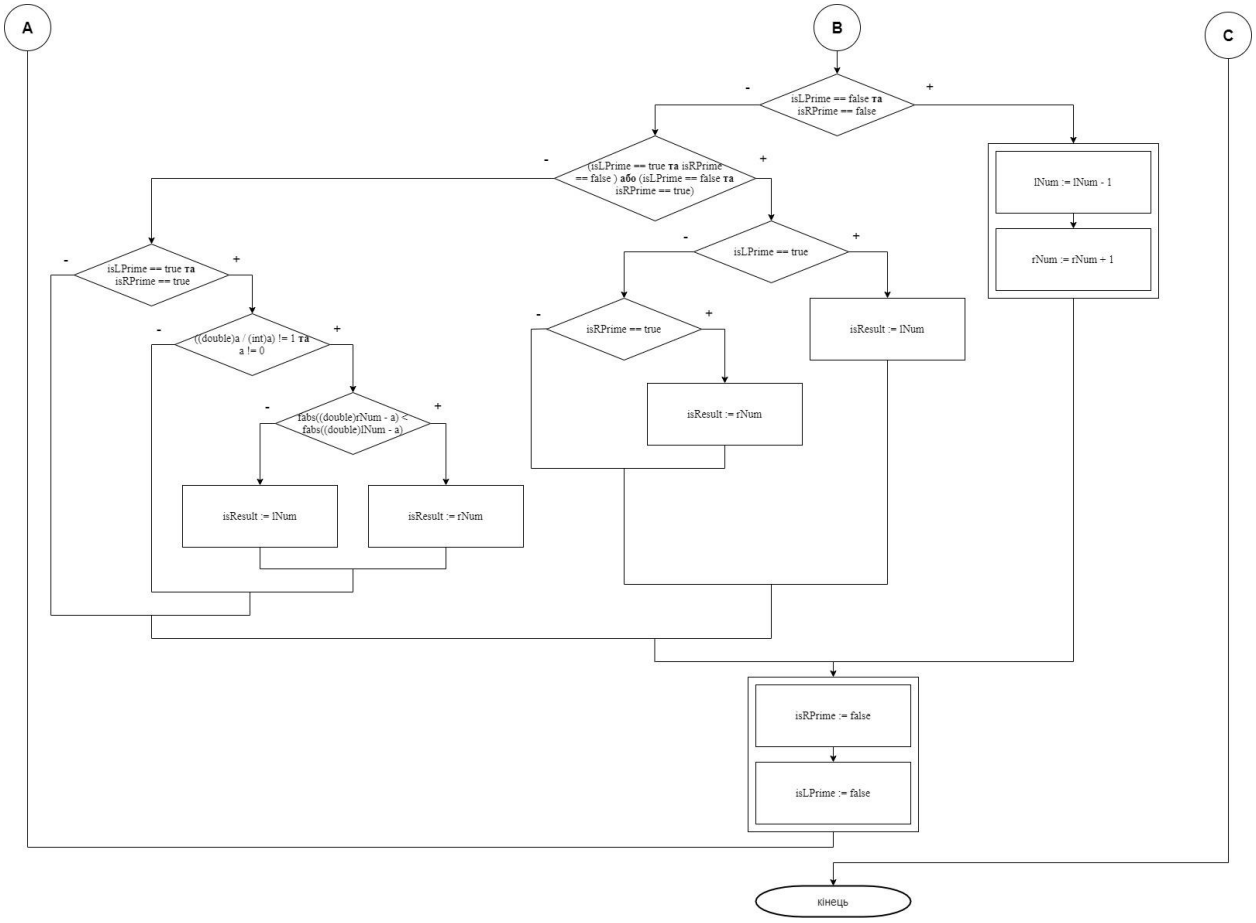

початок
isResult := 0
((double)a / (int)a) != 1 та a != 0
-
+
lNum := a - 1
rNum := a + 1
lNum := floor(a)
rNum := ceil(a)
isLPrime := false
isRPrime := false
i := 0
y := 0
count := 0
count2 := 0
відшукати значення простих чисел зліва та справа від введеного користувачем
визначити, яке просте число або числа будуть результатом виконання програми
кінець

Крок 3


початок
isResult := 0
(double)a - (int)a != 1 та a != 0
lNum := a - 1
rNum := a + 1
lNum := floor(a)
rNum := ceil(a)
isLPrime := false
isRPrime := false
i := 0
y := 0
count := 0
count2 := 0
isResult == 0
(i <= lNum та lNum > 0) або (i >= lNum та lNum < 0) або (i >= rNum та rNum > 0) або (i <= rNum та rNum < 0)
i != 0 та (lNum % i) == 0
count := count + 1
(count == 2 та i == lNum) або (count == 1 та abs(lNum) == 1)
isLPrime := true
lNum < 0
lNum > 0
i := i + 1
i := i + 1
y != 0 та (rNum % y) == 0
count2 := count2 + 1
(count2 == 2 та y == rNum) або (count2 == 1 та abs(rNum) == 1)
isRPrime := true
rNum < 0
rNum > 0
y := y - 1
y := y + 1
i := 0
y := 0
count := 0
count2 := 0
алгоритм, яке просте число або числа будуть результатом виконання програми
кінець

Крок 4


початок
isResult := 0
(double)a / (int)a != 1 та a != 0
lNum := a - 1
rNum := a + 1
lNum := floor(a)
rNum := ceil(a)
isLPrime := false
isRPrime := false
i := 0
y := 0
count := 0
count2 := 0
isResult == 0
(i) <= lNum та lNum > або (i >= lNum та lNum < 0) або (i) <= rNum та rNum > 0) або (i >= rNum та rNum < 0)
i != 0 та (lNum % i) == 0
count := count + 1
(count == 2 та i == lNum) або (count == 1 та abs(lNum) == 1)
isLPrime := true
lNum < 0
lNum > 0
i := i - 1
i := i + 1
y != 0 та (rNum % y) == 0
count2 := count2 + 1
(count2 == 2 та y == rNum) або (count2 == 1 та abs(rNum) == 1)
isRPrime := true
rNum < 0
rNum > 0
y := y - 1
y := y + 1
i := 0
y := 0
count := 0
count2 := 0
A
B
C



5. Т е с т у в а н н я

Блок	Дія 1		Дія 2
	Початок		Початок
1	isResult := 0	1	isResult := 0
2	a := 369	2	a := -18.4
3	lNum := 368, rNum := 370	3	lNum := -19, rNum := -18
4	isLPrime = false, isRPrime = false	4	isLPrime = false, isRPrime = false
5	i = 0, y = 0, count = 0, count2 = 0	5	i = 0, y = 0, count = 0, count2 = 0
6	i := 1	6	i := 1
7	y := 1	7	y := 1
8	count := 1	8	count := 1
9	i := 2	9	i := 2
10	count2 := 1	10	count2 := 1
11	y := 2	11	y := 2
12	count := 2	12	count := 2
13	i := 3	13	i := 3
14	count2 := 2	14	count2 := 2
15	y := 3	15	y := 3
…	…	…	…

1468	**i := 367**	**217**	**i := -19, count := 2, lNum := -19**
1469	**lNum := 367**	**…**	**…**
1470	**isResult := 367**	**230**	**i = 0, y = 0, count = 0, count2 = 0**
-	**-**	**…**	**…**
-	**-**	**235**	**isResult := -19**
	Кінець		**Кінець**

6. В и с н о в о к

В цій лабораторній роботі мені довелося дослідити особливості роботи складних циклів та набути практичних навичок їх використання під час складання програмних специфікацій. А саме, я скористався ітераційними циклами, адже в завданні я вирішив не обмежуватись тільки поступовим збільшенням або зменшенням лічильника циклу.