**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

Лабораторна робота №3  
*(вид роботи: лабораторна робота, індивідуальне завдання, курсова робота тощо)*

з дисципліни «Алгоритмізація та програмування»Тема роботи: «Розгалужений обчислювальний процес»  
**Варіант № 4**

Виконав(-ла) студент(-ка)  
групи АнД - 11  
Яковкін Микола Андрійович

Перевірив(-ла):  
ПІП викладача

Київ – 2021

**Завдання 1**

* 1. **Математична постановка задачі (МПЗ).**

*Вхідні дані:* A, B – цілі числа, сторони прямокутника C – ціле число, сторона квадрата;

*Вихідні дані:* amount\_a – ціле число кількості квадратів у заданому прямокутнику.

*Математична модель задачі:*

A = |A|; B = |B|; C = |C|; amount\_a = 0; amount\_b = 0;

Доки A ≥ C та B ≥ C: A = A – C;

B = B - C;

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 1.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| A, B – сторони прямокутника, цілі числа.  С – сторона квадрата, ціле число. | 1. A = |A|; B = |B|; C = |C|; amount\_a = 0; amount\_b = 0; 2. Доки A ≥ C та B ≥ C:   A = A – C;  B = B – C; | amount\_a - ціле число кількості квадратів у заданому прямокутнику. |

Схему алгоритму буде наведено після аналізу можливості застосування різних типів циклу.

**1.2 Тестовий приклад** *1.2.1 Тіло циклу виконується декілька разів.* Для введення подані числа 12; 5; 4: А = 12; В = 5; С = 4; amount\_a = 0; amount\_b = 0.

1ша ітерація. 12 ≥ 4 TRUE

A = 12 – 4 = 8;  
 1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено.

amount\_a = amount\_a + amount\_b = 0 + 1 = 1;

2га ітерація. 8 ≥ 4 TRUE

A = 8 – 4 = 4;

1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено

amount\_a = amount\_a + amount\_b = 1 + 1 = 2;

3тя ітерація. 4 ≥ 4 TRUE

A = 4 – 4 = 0;

1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено

amount\_a = amount\_a + 1 = 2 + 1 = 3

4та ітерація. 0 ≥ 4 FALSE. Цикл закінчено.

*1.2.2 Тіло циклу виконується один раз.*

Для введення подані числа 3; 3; 3: A = 3; B = 3; C = 3; amount\_a = 0; amount\_b = 0.

1ша ітерація. 3 ≥ 3 TRUE

A = A – C = 3 – 3 = 0;

1ша ітерація. 3 ≥ 3 TRUE

B = B – C = 3 – 3 = 0  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 0 ≥ 3 FALSE. Цикл закінчено.

amount\_a = amount\_a + amount\_b = 0 + 1 = 1;

2га ітерація. 0 ≥ 3 FALSE. Цикл закінчено.

*1.2.3 Тіло циклу не виконується*

Для введення подані числа 0; 2; 3: A = 0; B = 2; C = 3; amount\_a = 0; amount\_b = 0.

1ша ітерація. 0 ≥ 3 FALSE. Цикл закінчено.

* 1. **Аналіз можливості застосування циклу з постумовою.**

Для алгоритму з постумовою при значенні A = 0 або B = 0 тіло циклу не виконається жодного разу.

Проведемо аналіз можливості застосування циклу з постумовою.

*1.3.1 Тіло циклу виконується декілька разів.* Для введення подані числа 12, 5, 4: А = 12; В = 5; С = 4; amount\_a = 0; amount\_b = 0.

1ша ітерація.

A = 12 – 4 = 8;

1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено

amount\_a = amount\_a + 1 = 0 + 1 = 1;

8 ≥ 4 TRUE

2га ітерація.

A = 8 – 4 = 4;

1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено

amount\_a = amount\_a + 1 = 1 + 1 = 2;

4 ≥ 4 TRUE

3тя ітерація.

A = 4 – 4 = 0;

1ша ітерація. 5 ≥ 4 TRUE

B = 5 – 4 = 1;  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 4 FALSE. Цикл закінчено

amount\_a = amount\_a + 1 = 2 + 1 = 3;

0 ≥ 4 FALSE. Цикл закінчено

*1.3.2 Тіло циклу виконується один раз*

Для введення подані числа 3; 3; 3: А = 3; В = 3; С = 3; amount\_a = 0; amount\_b = 0.

1ша ітерація.

A = A – C = 3 – 3 = 0;

1ша ітерація. 3 ≥ 3 TRUE

B = B – C = 3 – 3 = 0  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 0 ≥ 3 FALSE. Цикл закінчено.

amount\_a = amount\_a + amount\_b = 0 + 1 = 1;

0 ≥ 3 FALSE. Цикл закінчено.

*1.3.3 Перевірка для випадка, коли А = 0 або В = 0.*

Для введення подані числа 0; 3; 2: А = 0; В = 3; С = 2; amount\_a = 0; amount\_b = 0.

1ша ітерація.

А = А – С = 0 – 2 = -2;

1ша ітерація. 3 ≥ 2 TRUE

B = B – C = 3 – 3 = 0  
 amount\_b = amount\_b + 1 = 0 + 1 = 1;

2га ітерація. 1 ≥ 2 FALSE. Цикл закінчено.

amount\_a = amount\_a + amount\_b = 0 + 1 = 1;

-2 ≥ 3 FALSE. Цикл закінчено

Можна зробити висновок, що для даної задачі цикл з передумовою підходить більше.

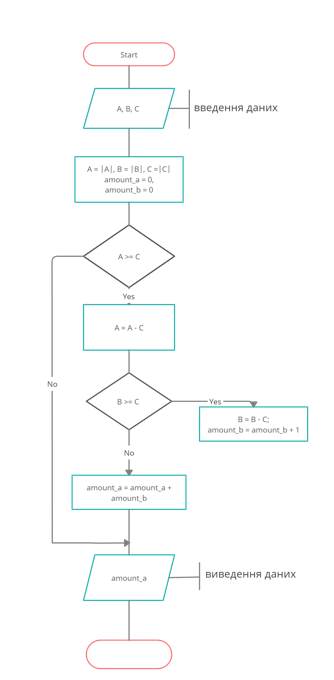
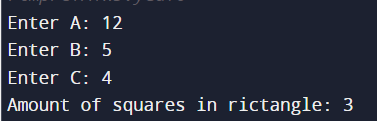


Рисунок 1.1 – Схема алгоритму задачі 1.

* 1. **Аналіз результатів роботи програми.**

Таблиця 1.2

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Розрахунок вихідних даних** | |
| **вручну** | **програма** |
| A = 12, B = 5, C = 4 | amount\_a = 3 | 3 |
| A = 3, B = 3, C = 3 | amount\_a = 1 | 1 |
| A = 0, C = 3, C = 2 | amount\_a = 0 | 0 |

  
Рисунок 1.2 – Тестування програми.

1. Робоче вікно програми для А = 12, В = 5, С = 4.

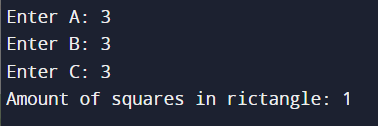


Рисунок 1.3 – Тестування програми.

1. Робоче вікно для А = 3, В = 3, С = 3.

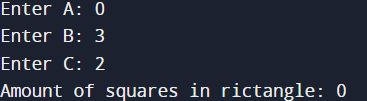


Рисунок 1.4 – Тестування програми.

1. Робоче вікно програми для А = 0, В = 3, С = 2.
   1. **Текст програмною мовою С++.**

#**include** <iostream>

**using namespace** std;

**int** main(){

**int** A, B, C;

int amount\_A = 0, amount\_B = 0;

cout << "Enter A: ";

cin >> A;

cout << "Enter B: ";

cin >> B;

cout << "Enter C: ";

cin >> C;

**while**(A >= C){

A -= C;

**while** (B>= C){

++amount\_B;

B -= C;

}

amount\_A += amount\_B;

}

cout << "Amount of squares in rictangle: " << amount\_A << endl;  
**return** 0;

}

**Завдання 2**

**2.1 Математична постановка задачі (МПЗ).**

*Вхідні дані:*  N – ціле число (N > 1);

*Вихідні дані:* К – значення степеню, ціле число.

*Математична модель задачі:*

N = |N|; K = 0;

Доки 3К + 1 < N: K = K + 1;

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 2.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| N – ціле число (N > 1). | 1. N = |N|; K = 0;  2.Доки 3К + 1 < N:  K = K + 1. | К – значення степеню, ціле число. |

Схема алгоритму буде наведено після аналізу можливості застосування різних типів циклу.

**2.2 Тестовий приклад.**

*2.2.1 Тіло циклу виконується декілька разів.*

Для введення подано число 10: N = 10; K = 0.

1ша ітерація. 30 + 1 < 10 TRUE

K = K + 1 = 0 + 1 = 1;

2га ітерація. 31 + 1 < 10 TRUE

K = K + 1 = 1 + 1 = 2;

3тя ітерація. 32 + 1 < 10 FALSE. Цикл закінчено.

*2.2.2 Тіло циклу виконується один раз.*

Для введення подано число 5: N = 5; K = 0;

1ша ітерація. 30 + 1 < 5 TRUE

K = K + 1 = 0 + 1 = 1;

2га ітерація. 31+1 < 5 FALSE. Цикл закінчено.

*2.2.3 Тіло циклу не виконується.*

Для введення подано число 3: N = 3; K =0.

1ша ітерація. 30 + 1 < 3 FALSE. Цикл закінчено.

**2.3 Аналіз можливості застосування циклу з постумовою.**

Для алгоритму з передумовою при значенні N = 0 тіло циклу не виконується жодного разу.

Проведемо аналіз можливості застосування циклу з постумовою.

*2.3.1 Тіло циклу виконується декілька разів.*

Для введення подано число 10: N = 10, K = 0;

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

31 + 1 < 10 TRUE

2га ітерація.

K = K + 1 = 1+ 1 = 2;

32 + 1 < 10 FALSE. Цикл закінчено.

*2.3.2 Тіло циклу виконується один раз.*

Для введення подано число 3: N = 3, K = 0.

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

30 + 1 < 3 FALSE. Цикл закінчено.

*2.3.3 Перевірка для випадка, коли N = 0.*

Для введення подано число 0: N = 0; K = 0.

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

31 + 1 < 0 FALSE. Цикл закінчено.

Можна зробити висновок, що для даної задачі цикл з передумовою підходить більше.

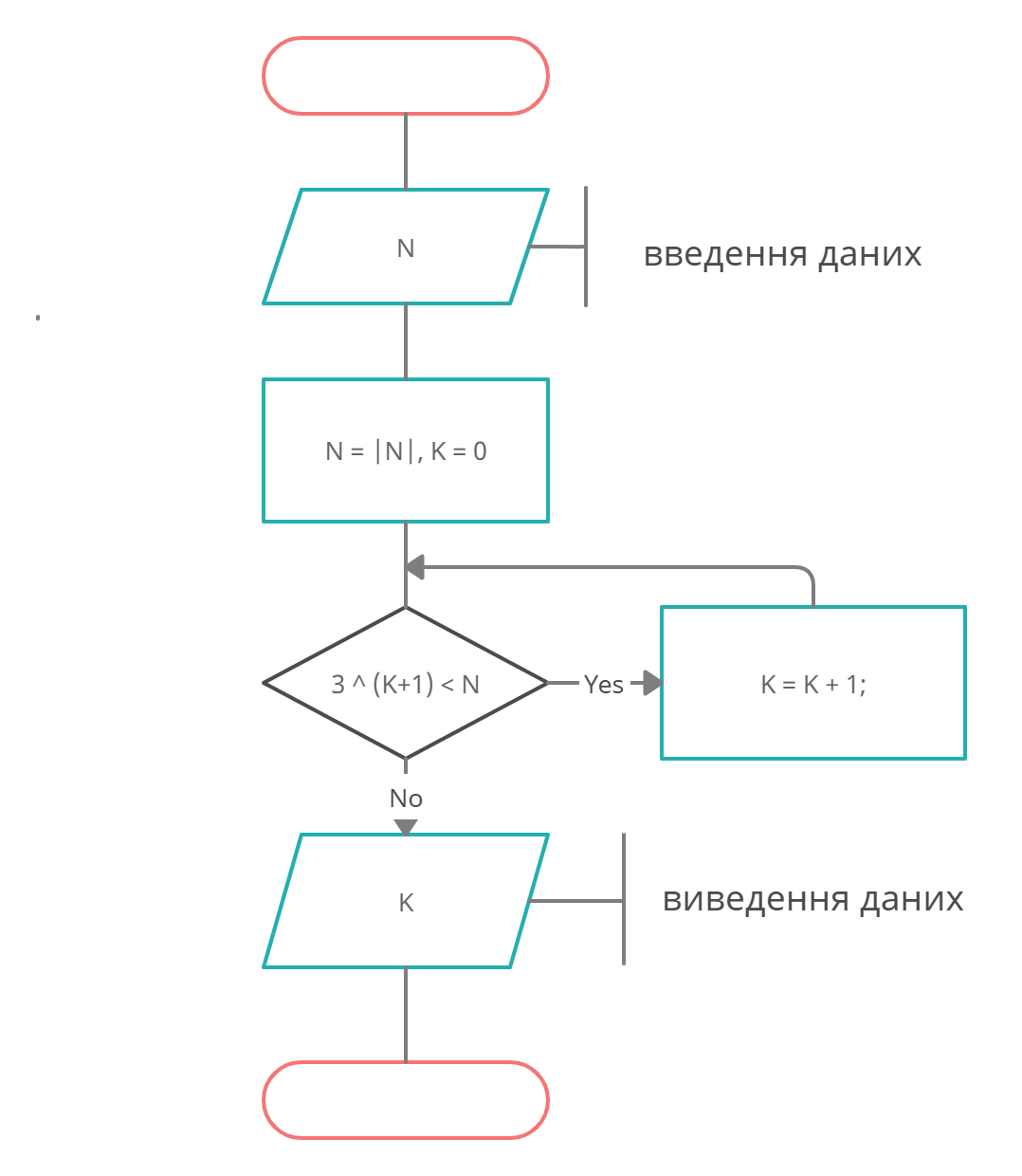


Рисунок 2.1 – Схема алгоритму задачі 2.

**­ 2.4 Аналіз результатів роботи програми.**

Таблиця 2.2

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Розрахунок вихідних даних** | |
| **вручну** | **програма** |
| N = 10 | K = 2 | 2 |
| N = 5 | K = 1 | 1 |
| N = 0 | Не існує | - |

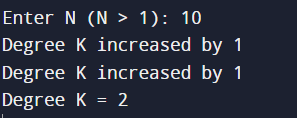


Рисунок 2.2 – Тестування програми.

1)Робоче вікно для програми N = 10.

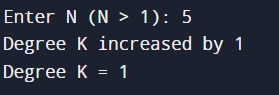


Рисунок 2.3 – Тестування програми.

2)Робоче вікно для програми N = 5.



Рисунок 2.4 – Тестування програми.

3) Робоче вікно для програми N = 0.

**2.5 Текст програмною мовою С++.**

#**include** <iostream>  
#**include** <cmath>

**using namespace** std;

**int** main(){

**int** N, K;

cout << "Enter N (N > 1): ";  
cin >> N;

K = 0;  
**while**(pow(3, K) < N){

++K;  
cout << "Degree K increased by 1" <<endl;

}

cout << "Degree K = " << K << endl;

**return** 0;

}

**Завдання 3**

**3.1 Математична постановка задачі (МПЗ).**

*Вхідні дані:* N – ціле число.

*Вихідні дані:* K – порядковий додаток, ціле число.

*Математична постановка задачі:*

K = 0; N = |N|; S = 0;

Доки S < N:

S = S + K

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 3.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| N – ціле число. | 1.K = 0; N = |N|; S = 0;  2.Доки S < N:  K = K + 1;  S = S + K. | K – порядковий додаток, ціле число. |

Схему алгоритму буде наведено після аналізу можливості застосування різнях типів циклу.

**3.2 Тестовий приклад**

*3.2.1 Тіло циклу виконується декілька разів.*

Для введення подано число 4: N = 4; K = 0; S = 0.

1ша ітерація. 0 < 4 TRUE

K = K + 1 = 0 + 1 = 1;

S = S + K = 0 + 1 = 1;

2га ітерація. 1 < 4 TRUE

K = K + 1 = 0 + 1 = 1;

S = S + K = 1 + 1 = 2;

3тя ітерація. 2 < 4 TRUE

K = K + 1 = 0 + 1 = 1;

S = S + K = 2 + 1 = 3;

4та ітерація. 3 < 4 TRUE

K = K + 1 = 0 + 1 = 1;

S = S + K = 3 + 1 = 4;

5та ітерація. 4 < 4 FALSE. Цикл закінчено.

*3.2.2 Тіло циклу виконується один раз.*

Для введення подано число 1: N = 1; K = 0; S= 0.

1ша ітерація. 0 < 1 TRUE

K = K + 1 = 0 + 1 = 1;

S = S + K = 0 + 1 = 1;

2га ітерація. 1 < 1 FALSE. Цикл закінчено.

*3.2.3 Тіло циклу не виконується*

Для введення подано число 0: N = 0; S = 0; K = 0

3тя ітерація. 0 < 0 FALSE. Цикл закінчено.

**3.3 Аналіз можливості застосування циклу з постумовою.**

Для алгоритму з передумовою при значенні N = 0 тіло циклу не виконується жодного разу.

Проведемо аналіз можливості застосування циклу з постумовою.

*3.3.1 Тіло циклу виконується декілька разів.*

Для введення подано число 4: N = 4; K = 0; S = 0.

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 0 + 1 = 1.

1 < 4 TRUE

2га ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 1 + 1 = 2.

2 < 4 TRUE

3тя ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 2 + 1 = 3;

3 < 4 TRUE

4та ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 0 + 1 = 4;

4 < 4 FALSE. Цикл закінчено.

*3.3.2 Тіло циклу виконується один раз.*

Для введення подано число 1: N = 1; S = 0; K = 0

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 0 + 1 = 1;

1 < 1 FALSE. Цикл закінчено.

*3.3.3 Перевірка випадка, коли N = 0.*

Для введення подано число 0: N = 0; S = 0; K = 0.

1ша ітерація.

K = K + 1 = 0 + 1 = 1;

S = S + 1 = 0 + 1 = 1;

1 < 0 FALSE. Цикл закінчено.

Можна зробити висновок, що для даної задачі цикл з передумовою підходить більше.

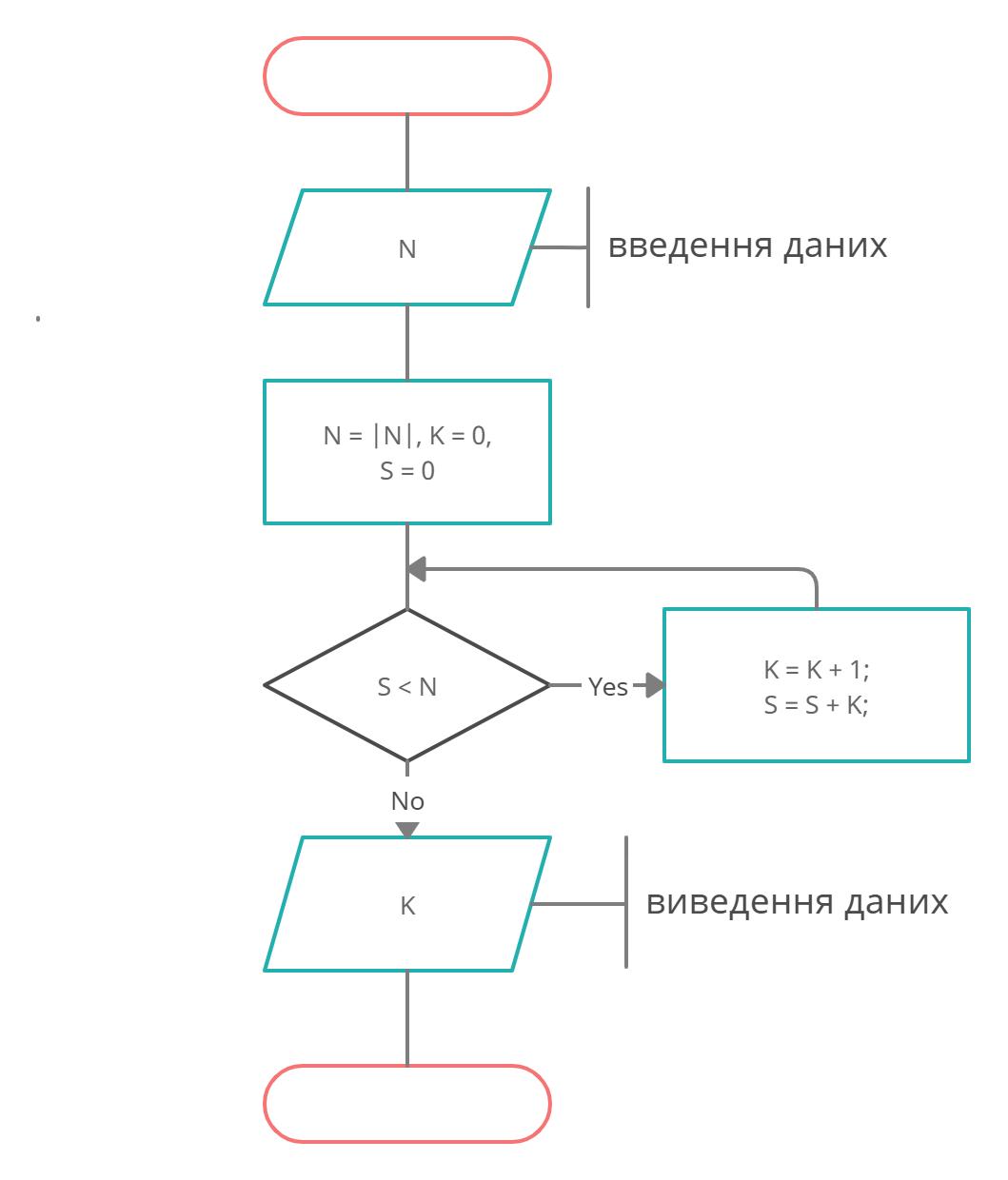


Рисунок 3.1 – Схема алгоритму задачі 3.

**3.4 Аналіз результатів роботи.**

Таблиця 3.2

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Розрахунок вихідних даних** | |
| **вручну** | **програма** |
| N = 4 | K = 4 | 4 |
| N = 1 | K = 1 | 1 |
| N = 0 | K = 0 | 0 |

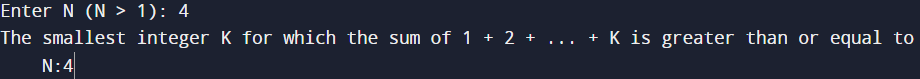


Рисунок 3.2 – Тестування програми.

1. Робоче вікно програми N = 4.

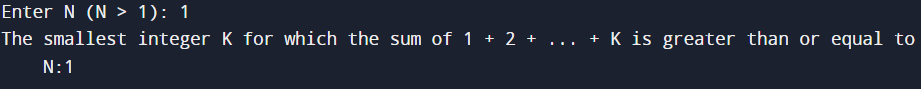


Рисунок 3.3 - Тестування програми.

1. Робоче вікно програми N = 1.

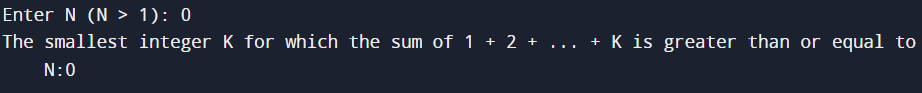


Рисунок 3.4 – Тестування програми.

1. Робоче вікно програми N = 0.

**3.5 Текст програмною мовою С++.**

#**include** <iostream>

using namespace std;

**int** main(){

**int** N, K, S;

K = 0;

cout << "Enter N (N > 1): ";  
cin >> N;

**while**(K < N){

K++;  
S += K;

}

cout << "The smallest integer K for which the sum of 1 + 2 + ... + K is greater than or equal to N:" << K;

**return** 0;

}

**Завдання 4**

**4.1 Математична постановка задачі (МПЗ).**

*Вхідні дані:* N – ціле число.

M – число, яке бажають видалити, ціле число (0 ≤ M ≤ 9).

*Вихідні дані:* new\_number – число після змін, ціле число.

*Математична модель задачі:*

new\_number = 0; place = 1; N = |N|; M = |M|;

Доки N > 0:

Якщо {{N / 10} / M} != 0:

new\_number = new\_number + {N / 10} \* place;

place = place \* 10;

N = [N / 10];

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 4.1

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Дії** | **Вихідні дані** |
| N – ціле число.  M – число, яке бажають видалити, ціле число  (0 ≤ M ≤ 9). | 1. new\_number = 0;  place = 1; N = |N|; M = |M|;    2.Доки N > 0:  Якщо {{N / 10} / M} != 0:  new\_number = new\_number + {N / 10} \* place;  place = place \* 10;  N = [N / 10]; | new\_number – число після змін, ціле число. |

Схему алгоритму буде наведено після аналізу можливості застосування різних типів циклу.

**4.2 Тестовий приклад.**

*4.2.1 Тіло циклу виконується декілька разів.*

Для введення подані числа 3445, 4: N = 3445; M = 4; new\_number = 0;   
place = 1.

1ша ітерація. 3445 > 0 TRUE

{{3445 / 10} / 4} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {344 / 10} \* 1 = 5;

place = place \* 10 = 10;

N = [N / 10] = [3445 / 10] = 344;

2га ітерація. 344 > 0 TRUE

{{344 / 10} / 4} ≠ 0 FALSE:

N = [N / 10] = [344 / 10] = 34;

3тя ітерація. 34 > 0 TRUE

{{34 / 10} / 4} ≠ 0 FALSE:

N = [N / 10] = [34 / 10] = 3;

4та ітерація. 3 > 0 TRUE

{{3 / 10} / 4} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {344 / 10} \* 1 = 35;

place = place \* 10 = 100;

N = [N / 10] = [3 / 10] = 0;

5та ітерація. 0 > 0 FALSE. Цикл закінчено.

*4.2.2 Тіло циклу виконується один раз.*

Для введення подані числа 5, 5: N = 5; M = 5; new\_number = 0; place = 1.

1ша ітерація. 5 > 0 TRUE

{{5 / 10} / 5} ≠ 0 FALSE:

N = [N / 10] = [5 / 10] = 0;

2га ітерація. 0 > 0 FALSE. Цикл закінчено.

*4.2.3 Тіло циклу коли M не належить своєму проміжку.*

Для введення подані числа 2325, 12: N = 1223; M = 21; new\_number = 1; place = 0.

1ша ітерація. 1223 > 0 TRUE

{{1223 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {1223 / 10} \* 10 = 3;

place = place \* 10 = 1 \* 10 = 10;

N = [N / 10] = [1223 / 10] = 122;

2га ітерація. 122 > 0 TRUE

{{122 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {122 / 10} \* 10 = 23;

place = place \* 10 = 10 \* 10 = 100;

N = [N / 10] = [122 / 10] = 12;

12 > 0 TRUE

3тя ітерація. 12 > 0 TRUE

{{12 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {12 / 10} \* 100 = 223;

place = place \* 10 = 100 \* 10 = 1000;

N = [N / 10] = [12 / 10] = 1;

1 > 0 TRUE

4та ітерація. 1 > 0 TRUE

{{1 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {1 / 10} \* 1000 = 1223;

place = place \* 10 = 1000 \* 10 = 10000;

N = [N / 10] = [1 / 10] = 0;

5та ітерація. 0 > 0 FALSE. Цикл закінчено

**4.3 Аналіз можливості застосування циклу з постумовою.**

Для алгоритму з передумовою при значенні N = 0 тіло циклу не виконується.

Проведемо аналіз можливості застосування циклу з постумовою.

*4.3.1 Тіло циклу виконується декілька разів.*

Для введення подані такі числа 3445, 4: N = 3445; M = 4, new\_number = 0, place = 1.

1ша ітерація.

{{3445 / 10} / 4} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {3445 / 10} \* 1 = 5;

place = place \* 10 = 1 \* 10 = 10;

N = [N / 10] = [3445 / 10] = 344;

344 > 0 TRUE

2га ітерація.

{{344 / 10} / 4} ≠ 0 FALSE:

N = [N / 10] = [344 / 10] = 34;

34 > 0 TRUE

3тя ітерація.

{{34 / 10} / 4} ≠ 0 FALSE:

N = [N / 10] = [34 / 10] = 3;

3 > 0 TRUE

4та ітерація.

{{3 / 10} / 4} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {3 / 10} \* 10 = 35;

place = place \* 10 = 10 \* 10 = 100;

N = [N / 10] = [3 / 10] = 0;

0 > 0 FALSE. Цикл закінчено.

*4.3.2 Тіло циклу виконується один раз.*

Для введення подані числа 5, 5: N = 5; M = 5; new\_number = 0; place = 1.

1ша ітерація:

{{5 / 10} / 5} ≠ 0 FALSE:

N = [N / 10] = [5 / 10] = 0;

0 > 0 FALSE. Цикл закінчено.

*4.3.3 Перевірка для випадка, коли M не належить проміжку.*

Для введення подані числа 0, 2: N = 1223; M = 21; new\_number = 0;   
place = 1.

1ша ітерація:

{{1223 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {1223 / 10} \* 10 = 3;

place = place \* 10 = 1 \* 10 = 10;

N = [N / 10] = [1223 / 10] = 122;

122 > 0 TRUE

2га ітерація:

{{122 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {122 / 10} \* 10 = 23;

place = place \* 10 = 10 \* 10 = 100;

N = [N / 10] = [122 / 10] = 12;

12 > 0 TRUE

3тя ітерація:

{{12 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {12 / 10} \* 100 = 223;

place = place \* 10 = 100 \* 10 = 1000;

N = [N / 10] = [12 / 10] = 1;

1 > 0 TRUE

4та ітерація:

{{1 / 10} / 21} ≠ 0 TRUE:

new\_number = new\_number + {N / 10} \* place = {1 / 10} \* 1000 = 1223;

place = place \* 10 = 1000 \* 10 = 10000;

N = [N / 10] = [1 / 10] = 0;

0 > 0 FALSE. Цикл закінчено.

Можна зробити висновок, що для даної задачі цикл з постумовою підходить більше.

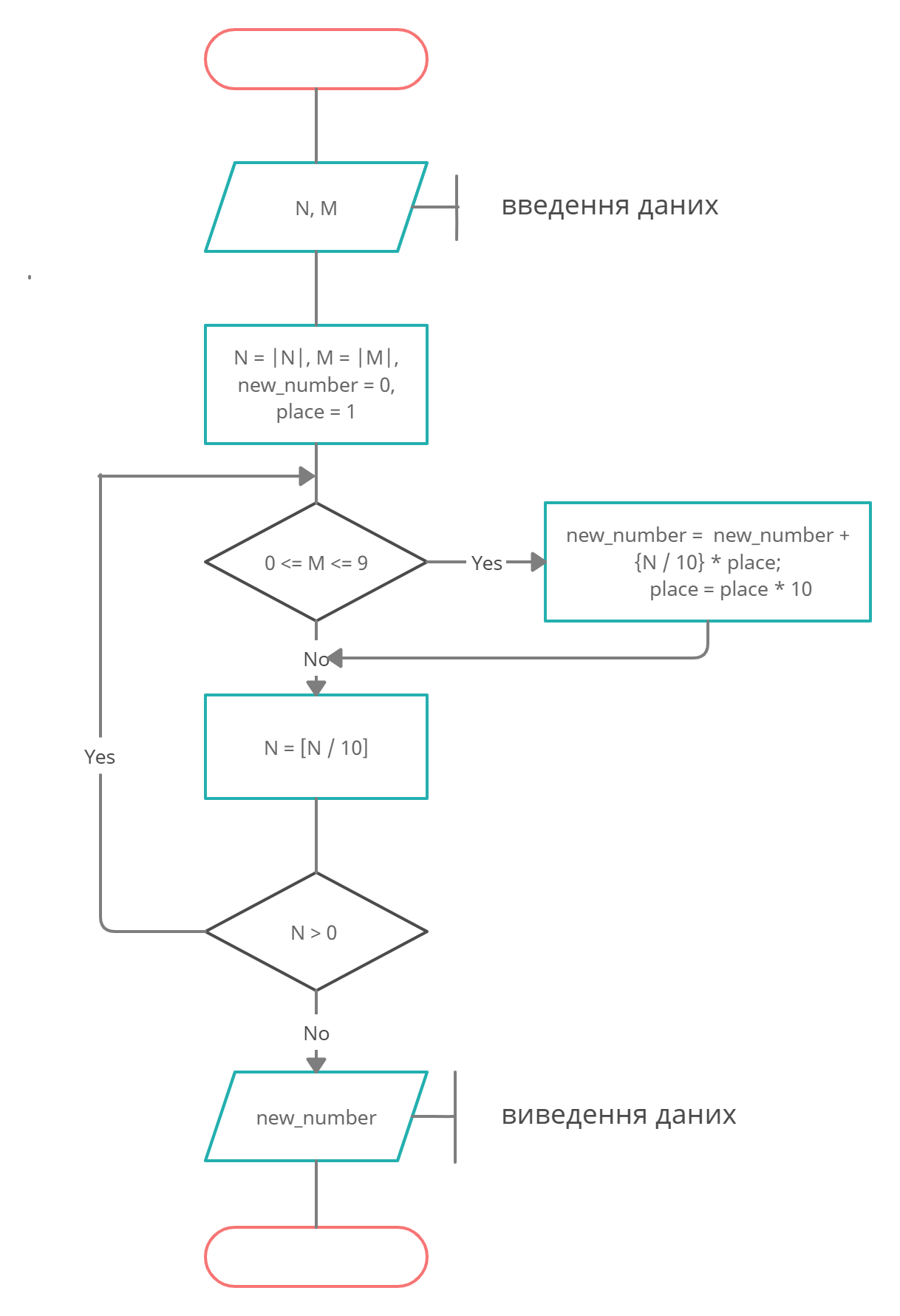


Рисунок 4.1 – Схема алгоритму задачі 4.

**4.4 Аналіз результатів роботи програми.**

Таблиця 4.2

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Розрахунок вихідних даних** | |
| **вручну** | **програма** |
| N = 3445, M = 4 | N = 35 | 35 |
| N = 5, M = 5 | N = 0 | 0 |
| N = 1223, M = 21 | N = 1223 | 1223 |

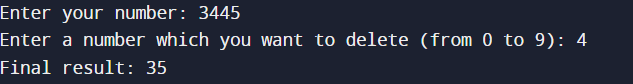
****

Рисунок 4.2 – Тестування програми.

1. Робоче вікно програми для N = 3445, M = 4.

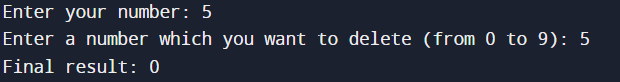
****

Рисунок 4.3 – Тестування програми.

1. Робоче вікно програми для N = 5, M =5.

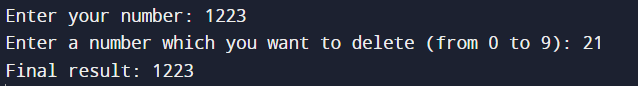


Рисунок 4.4 – Тестування програми.

1. Робоче вікно програми для N = 1223, M = 21.

**4.5 Текст програмною мовою С++.**

#**include** <iostream>

**using namespace** std;

**int** main(){

**int** N, M, place = 1, new\_number = 0;

cout << "Enter your number: ";  
cin >> N;

cout << "Enter a number which you want to delete (from 0 to 9): ";  
cin >> M;

**do**{

if((N % 10) % M != 0){

new\_number += N % 10 \* place;

place \*= 10;

}

N /= 10;

}

**while**(N > 0);

cout << "Final result: " << new\_number << endl;

**return** 0;

}

**Завдання 5**

**5.1 Математична постановка задачі (МПЗ):**

*Вхідні дані:* N – ціле число.

*Вихідні дані:* K – порядковий номер числа Фіабоніччі, ціле число.

*Математична модель задачі:*

K = 2; f1 = 0; f2 = 1; N = |N|;

Доки f ≠ N та N – число Фіабоніччі:

f = f1 + f2;

f1 = f2;

f2 = f;

K = K + 1;

Подання математичної постановки задачі у вигляді таблиці:

Таблиця 5.1

|  |  |  |
| --- | --- | --- |
| Вхідні дані | Дії | Вихідні дані |
| N – ціле число. | 1.K = 2; f1 = 0; f2 = 1;  N = |N|;  2.Доки f ≠ N та N – число Фіабоніччі:  f = f1 + f2;  f1 = f2;  f2 = f;  K = K + 1; | K – порядковий номер числа Фіабоніччі, ціле число. |

Схему алгоритму буде наведено після аналізу можливості застосування різних типів циклу.

**5.2 Тестовий приклад.**

*5.2.1 Тіло циклу виконується декілька разів*

Для введення подано число 5: N = 5; f1 = 0; f2 = 1; K = 2.

1ша ітерація. 0 ≠ 5 та N – число Фіабоніччі TRUE

f = f1 + f2 = 0 + 1 = 1;

f1 = f2 = 1;

f2 = f = 1;

K = K + 1 = 2 + 1 = 3;

2га ітерація. 1 ≠ 5 та N – число Фіабоніччі TRUE

f = f1 + f2 = 1 + 1 = 2;

f1 = f2 = 1;

f2 = f = 2;

K = K + 1 = 3 + 1 = 4;

3тя ітерація. 2 ≠ 5 та N – число Фіабоніччі TRUE

f = f1 + f2 = 1 + 2 = 3;

f1 = f2 = 2;

f2 = f = 3;

K = K + 1 = 4 + 1 = 5;

4та ітерація. 3 ≠ 5 та N – число Фіабоніччі TRUE

f = f1 + f2 = 2 + 3 = 5;

f1 = f2 = 3;

f2 = f = 5;

K = K + 1 = 5 + 1 = 6;

5та ітерація. 5 ≠ 5 та N – число Фіабоніччі FALSE. Цикл закінчено.

*5.2.2 Тіло циклу виконується один раз.*

Для введення подано число 1: N = 1; f1 = 0; f2 = 1; K = 2.

1ша ітерація. 0 ≠ 1 та N – число Фіабоніччі TRUE

f = f1 + f2 = 0 + 1 = 1;

f1 = f2 = 1;

f2 = f = 1;

K = K + 1 = 2 + 1 = 3;

2га ітерація. 1 ≠ 1 та N – число Фіабоніччі FALSE. Цикл закінчено.

*5.2.3 Тіло циклу виконується 0 разів.*

Для введення подано число 6: N = 6; f1 = 0; f2 = 1; K = 2.

1ша ітерація. 0 ≠ 6 та N – число Фіабоніччі FALSE. Цикл закінчено.

**5.3 Аналіз можливості застосування циклу з постумовою.**

Для алгоритму з передумовою при значенні N – не є число Фібаноччі тіло циклу не виконується жодного разу.

Проведемо аналіз можливості застосування циклу з постумовою.

*5.3.1 Тіло циклу виконується декілька разів.*

Для введення подано число 5: N = 5; f1 = 0; f2 = 1; K= 3.

1ша ітерація.

f = f1 + f2 = 0 + 1 = 1;

f1 = f2 = 1;

f2 = f = 1;

K = K + 1 = 2 + 1 = 3;

1 ≠ 5 та N – число Фіабоніччі TRUE

2га ітерація.

f = f1 + f2 = 1 + 1 = 2;

f1 = f2 = 1;

f2 = f = 2;

K = K + 1 = 3 + 1 = 4;

2 ≠ 5 та N – число Фіабоніччі TRUE

3тя ітерація.

f = f1 + f2 = 1 + 2 = 3;

f1 = f2 = 2;

f2 = f = 3;

K = K + 1 = 4 + 1 = 5;

3 ≠ 5 та N – число Фіабоніччі TRUE

4та ітерація.

f = f1 + f2 = 2 + 3 = 5;

f1 = f2 = 3;

f2 = f = 5;

K = K + 1 = 5 + 1 = 6;

5 ≠ 5 та N – число Фіабоніччі FALSE. Цикл закінчено.

*5.3.2 Тіло циклу виконується один раз.*

Для введення подано число 1: N = 1; f1 = 0; f2 = 1; K = 2.

1ша ітерація.

f = f1 + f2 = 0 + 1 = 1;

f1 = f2 = 1;

f2 = f = 1;

K = K + 1 = 2 + 1 = 3;

1 ≠ 1 та N – число Фіабоніччі FALSE. Цикл закінчено.

*5.3.3 Перевірка для випадка коли N – не є число Фіабоніччі.*

Для введення подано число 6: N = 6; f1 = 0; f2 = 1; K = 2.

1ша ітерація.

f = f1 + f2 = 0 + 1 = 1;

f1 = f2 = 1;

f2 = f = 1;

K = K + 1 = 2 + 1 = 3;

1 ≠ 6 та N – число Фіабоніччі FALSE. Цикл закінчено.

Можна зробити висновок, що для даної задачі цикл з постумовою підходить більше.

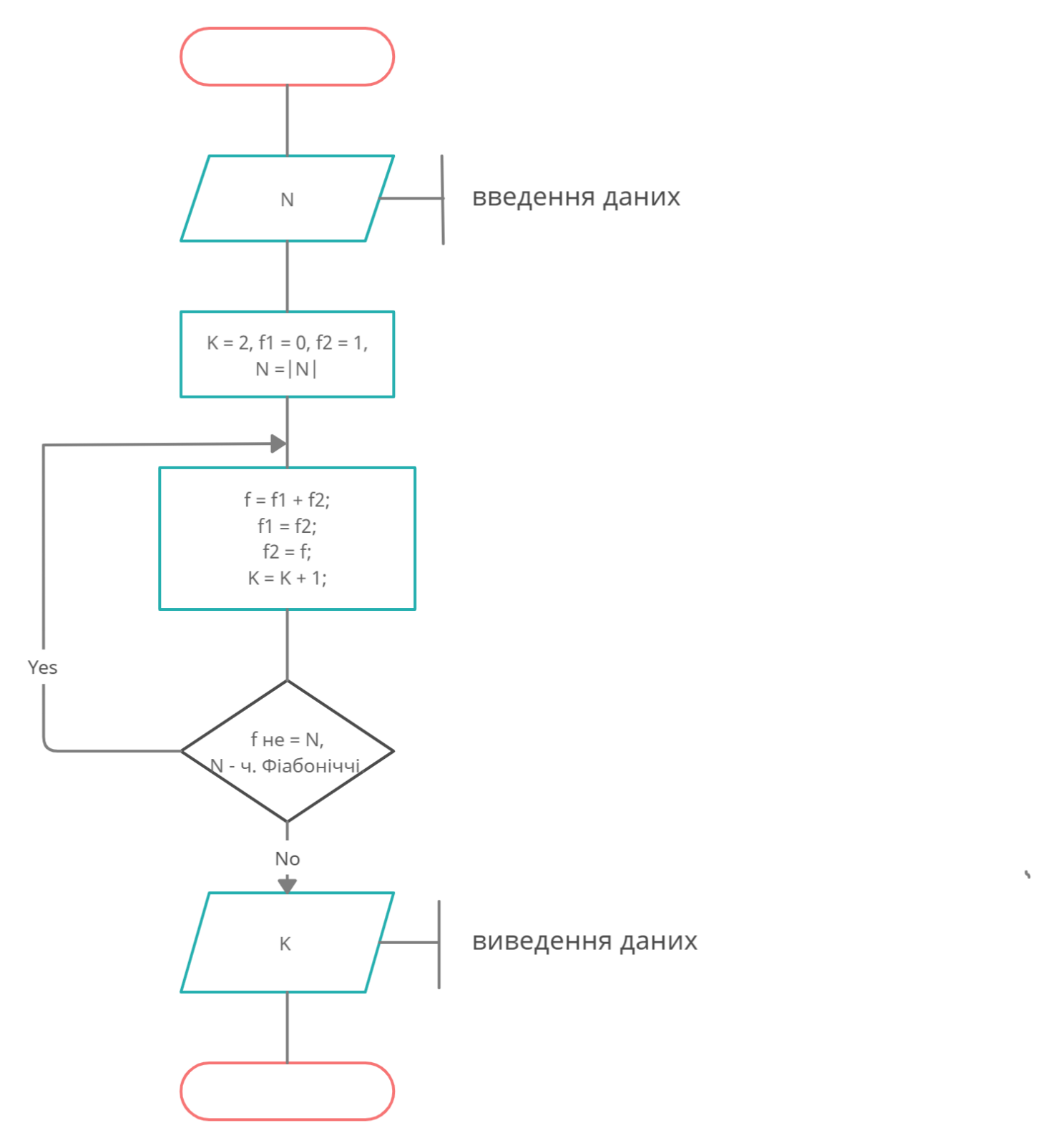


Рисунок 5.1 – Схема алгоритму задачі 5.

**5.4 Аналіз результатів роботи програми.**

Таблиця 5.2

|  |  |  |
| --- | --- | --- |
| **Вхідні дані** | **Розрахунок вихідних даних** | |
| **вручну** | **програма** |
| N = 5 | K = 6 | 6 |
| N = 1 | K = 3 | 3 |
| N = 6 | Не існує | - |



Рисунок 5.2 – Тестування програми.

1. Робоче вікно програми для N = 5.



Рисунок 5.3 – Тестування програми.

1. Робоче вікно програми для N = 1.



Рисунок 5.4 – Тестування програми.

1. Робоче вікно програми для N = 6.

**5.5 Текст програмною мовою С++.**

#**include** <iostream>

**using namespace** std;

**int** main(){

int N, K = 2;

int f1 = 0, f2 = 1, f;

cout << "Enter N (N is Fiabonicci number): ";  
cin >> N;

**do**{

f = f1 + f2;  
f1 = f2;  
f2 = f;  
K++;

}

**while**(f != N);

cout << "Sequential number of the number " << N <<" equels: " << K << endl;

**return** 0;

}