МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Основи програмування»**

**на тему: «Розробка програм з циклічною обробкою даних»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2019

**Лабораторна робота №3**

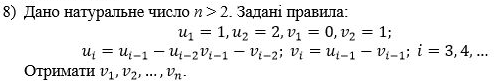
**Тема.** Розробка програм з циклічною обробкою даних.

**Мета.** Вивчити типи операторів циклу. Навчитися використовувати різні типи циклів для **розв’язання задач.**

**Постановка задачи**

1.Загальна постановка: розробити три програми відповідно до обраного варіанту завдання на основі трьох видів циклів.

2.Індивідуальна постановка:



3.Вимоги до програми:

-вхідні дані вводяться з клавіатури;

-передбачити перевірку вхідних даних на відповідність діапазону значень і некоректні символи;

-результати роботи програми виводяться на екран;

-забороняється використовувати масиви (!!!).

4.Вимоги до тексту програми:

-коментарі щодо призначення програми, її вхідних і вихідних даних;

-коментарі щодо призначення кожного блоку програми, дій окремих операторів для пояснення алгоритму;

-самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

**Зовнішні специфікації**

1. Вхідні дані

Вхідні дані:

**Формат вхідних даних**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Номер числа в рядку | n | Ціле число | 5 |

2. Вихідні дані:

**Формат вихідних даних**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Результат V | V\_result |  | -2 |
| 2 | Результат u | u\_result |  | -3 |

3. Функціональні вимоги до програми

Програма повинна реалізувати такі дії:

-введення даних у програму

-перевірку заданих даних

-вивід результату перевірки

**Вибір методу рішення задачі**

Правила з постановки задачі: *u*i *=u*i-1 – *u*i-2*v*i-1 – *v*i-2; *v*i *=u*i-1 – *v*i-1 взяв як загальні правила, а кожні нові значення привласнював до тих які у мене уже були => ui-1 = ui-2;

vi-1 = vi-2;

ui-2 = u\_result; (u\_result – результат до числа u)

vi-2 = V\_result; (V\_result – результат до числа v)

Тобто в кожному колі циклу значення числа, яке на одиницю менше (за порядком) приймає значення числа, яке на два менше (за порядком), а числа, за порядком які на два менше, приймають значення результату.

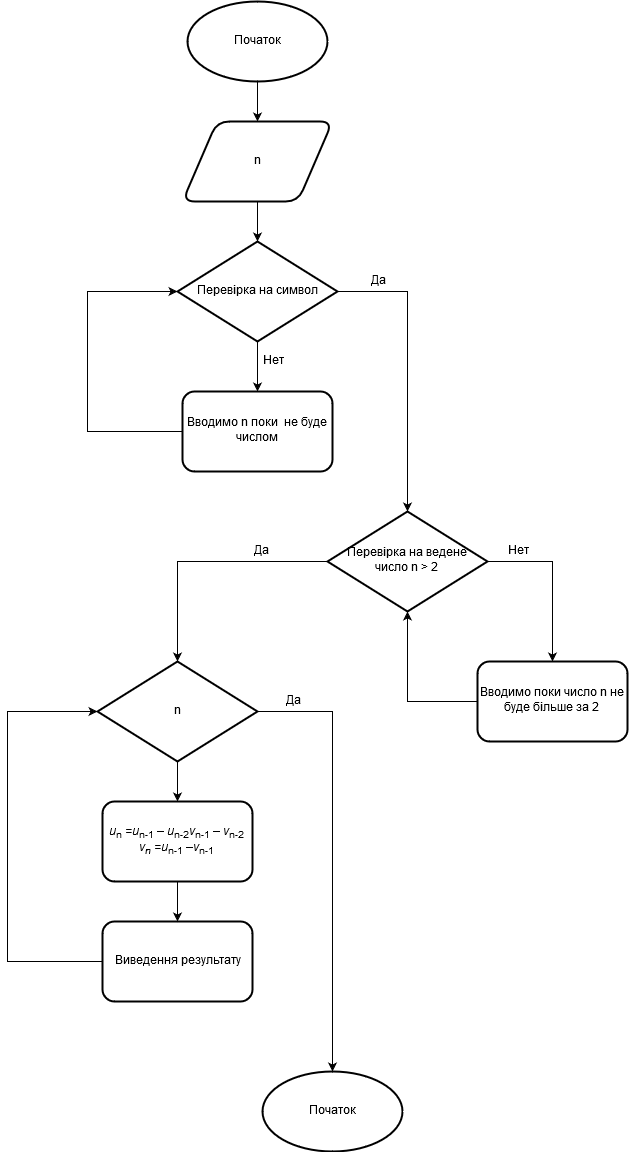
**Розробка тестів**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікувані результати |
| 1 | Некоректне значення числа | n = 1 | “Помилка вводу! Повторить ввод. n повино бути цілим” |
| 2 | Мінімальне значення числа | n = 3 | v3 = 1 |
| 3 | Введене замість числа символ | n = f | “Помилка вводу! Повторить ввод. n має бути цілим” |

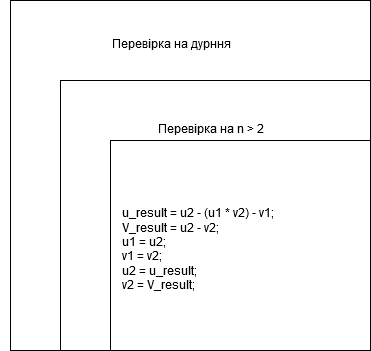
**Алгоритм програм**

**а)Для програми з циклом for:**

1.Блок-схема



2.Діаграма Н-Ш



3.Текст програми

/\*

Заданими правилами за допомогою циклу знайти v ти u

\*/

#include<iostream>

#include<Windows.h>

using namespace std;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

double V\_result, u\_result;

double v1 = 0, v2 = 1, u1 = 1, u2 = 2;

int n;

cout << "Введіть n = ";

while (!(cin >> n) || (cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

cin.clear();

while (cin.get() != '\n');

cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << endl;

cout << "Введіть n = ";

}

while ((n < 3) || (cin.peek() != '\n')) // цикл для перевірки правила на границю числа

{

cin.clear();

while (cin.get() != '\n');

cerr << "! Число n повино бути більше 2" << endl;

cout << "Введіть n = ";

cin >> n;

}

for (int i = 3; i <= n; i++)

{

u\_result = u2 - (u1 \* v2) - v1;

V\_result = u2 - v2;

cout << endl << "Відповідь V" << i <<" = " << V\_result << endl; // цей блок для виведення реулитату кожного кола

//cout << " Відповідь u = " << ur << " коло номер " << i << endl;

u1 = u2;

v1 = v2;

u2 = u\_result;

v2 = V\_result;

}

/\*cout << " Відповідь V = " << V\_result << endl; // цей блок зроблен для виведення результату кінцевого

cout << " Відповідь u = " << u\_result << endl;\*/

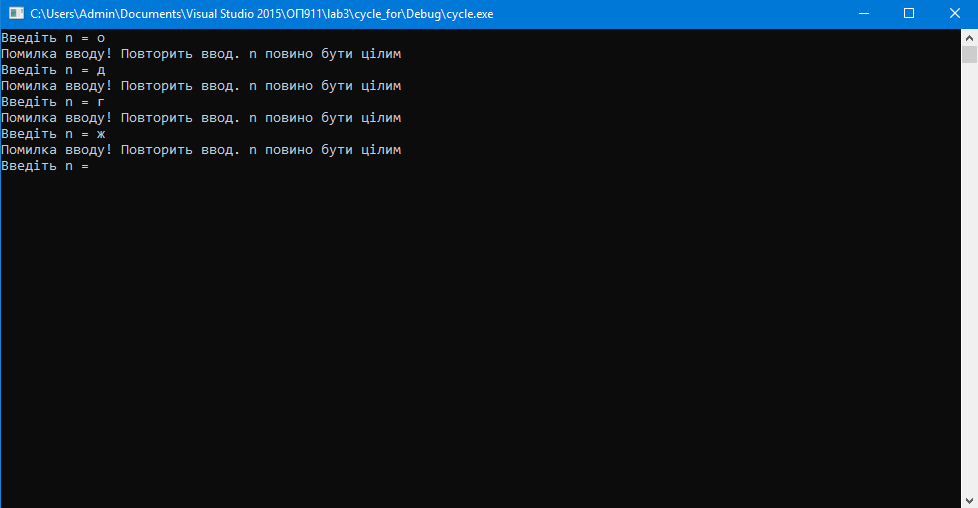
system("pause");

return;

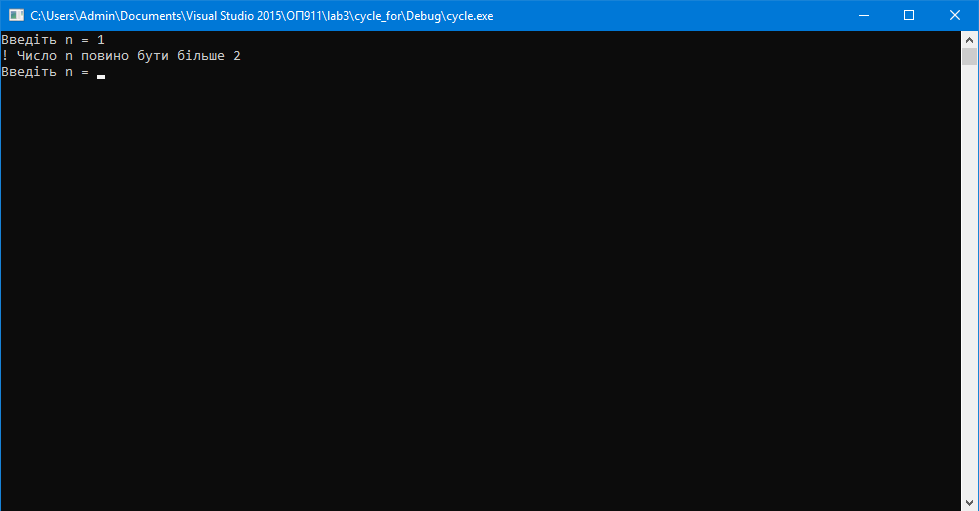
}

4.Результати виповнення програми

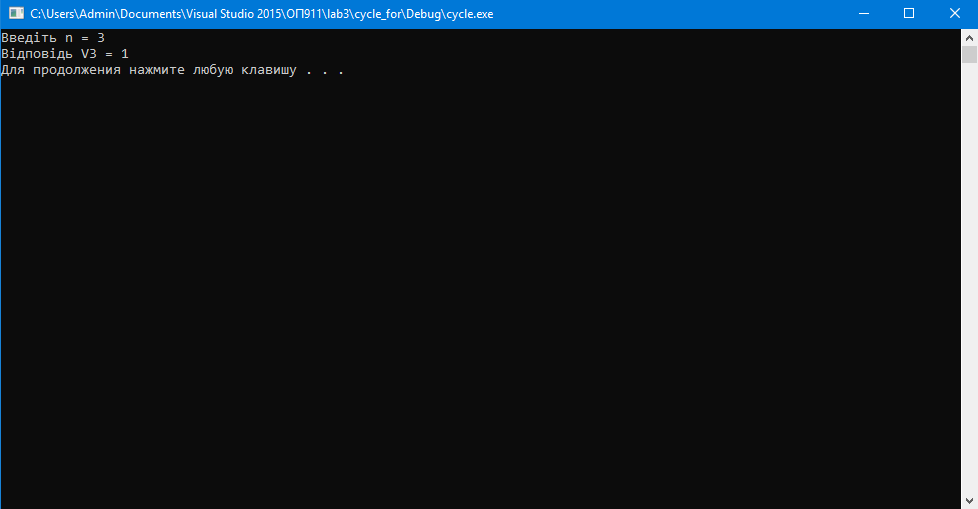
4.1



4.2

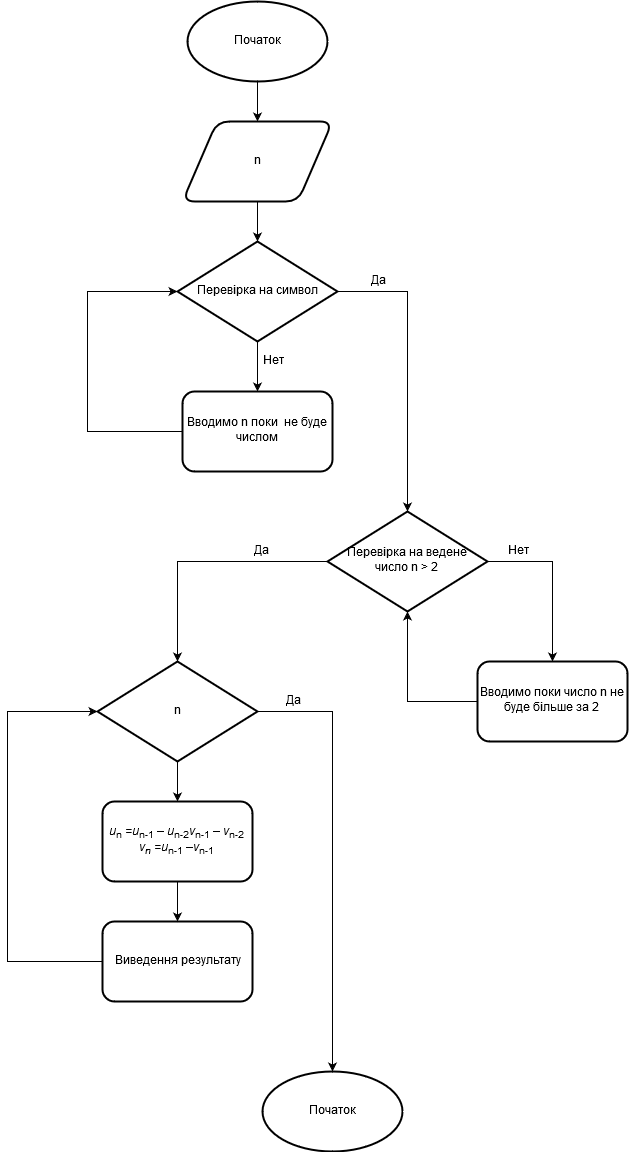


4.3

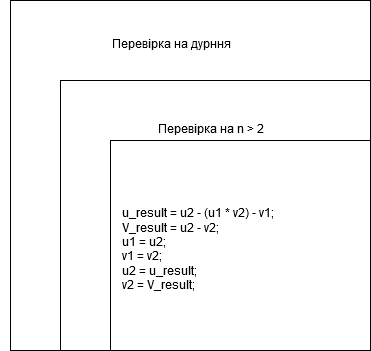


**б)Для програми з циклом while:**

1.Блок-схема



2.Діаграма Н-Ш



3.Текст програми

/\*

Заданими правилами за допомогою циклу знайти v ти u

\*/

#include<iostream>

#include<Windows.h>

using namespace std;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

double V\_result, u\_result;

double v1 = 0, v2 = 1, u1 = 1, u2 = 2;

int n;

cout << "Введіть n = ";

while (!(cin >> n) || (cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

cin.clear();

while (cin.get() != '\n');

cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << endl;

cout << "Введіть n = ";

}

while ((n < 3) || (cin.peek() != '\n')) // цикл для перевірки правила на границю числа

{

cin.clear();

while (cin.get() != '\n');

cerr << "! Число n повино бути більше 2" << endl;

cout << "Введіть n = ";

cin >> n;

}

int i = 3;

while (i <= n)

{

u\_result = u2 - (u1 \* v2) - v1;

V\_result = u2 - v2;

cout << "Відповідь V" << i << " = " << V\_result << endl; // цей блок для виведення реулитату кожного кола

//cout << "Відповідь u" << i << " = " << u\_result << endl;

u1 = u2;

v1 = v2;

u2 = u\_result;

v2 = V\_result;

++i;

}

//cout << " Відповідь V = " << V\_result << endl;

//cout << " Відповідь u = " << u\_result << endl;

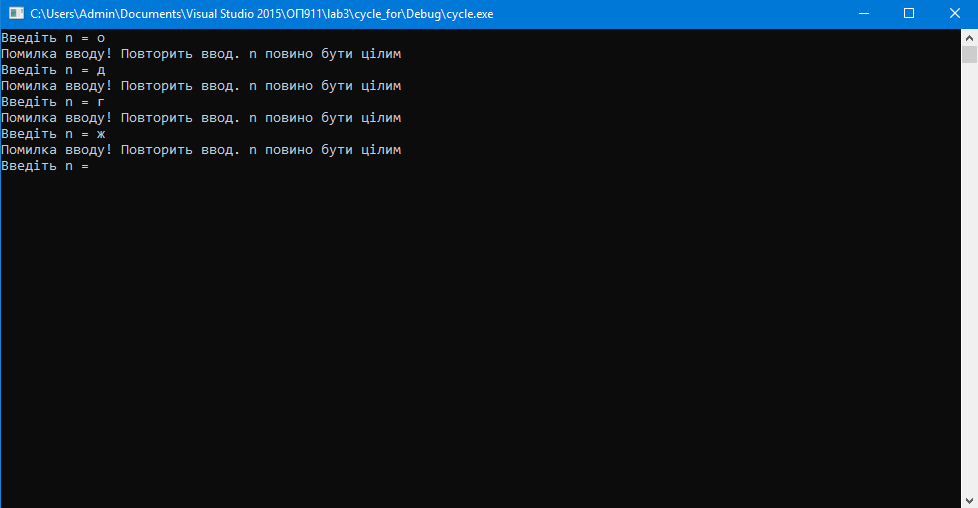
system("pause");

return;

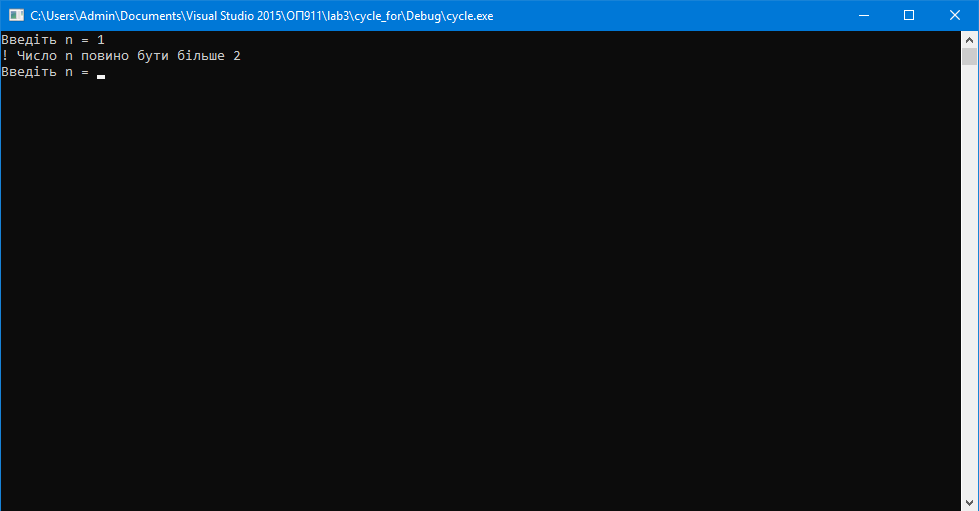
}

4.Результати виповнення програми

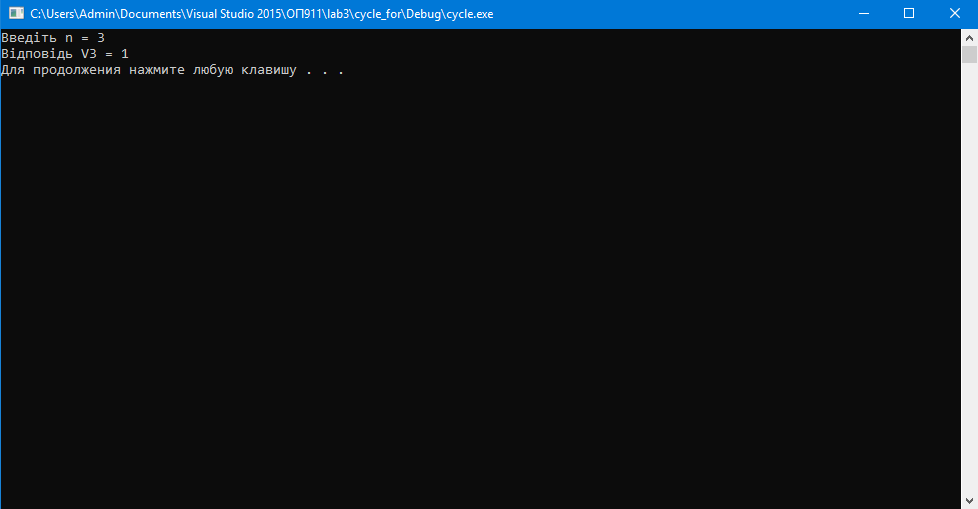
4.1



4.2

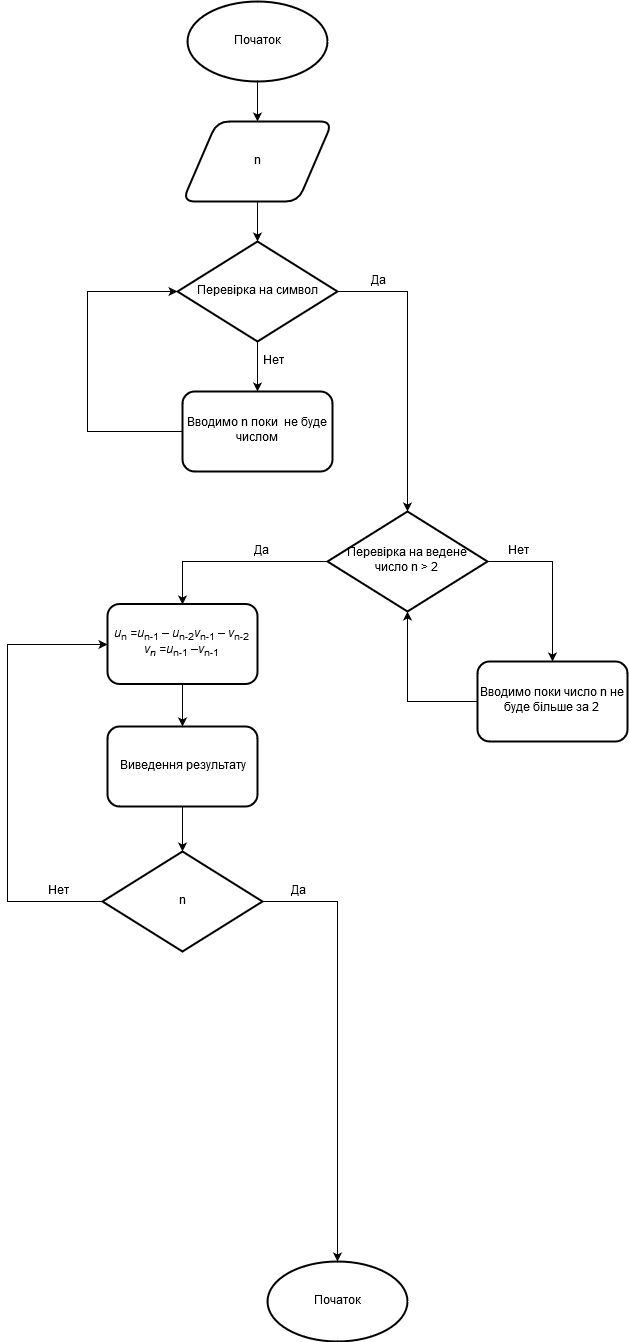


4.3

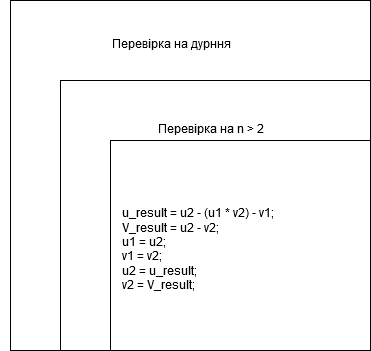


**в)Для програми з циклом do\_while:**

1.Блок-схема



2.Діаграма Н-Ш



3.Текст програми

/\*

Заданими правилами за допомогою циклу знайти v ти u

\*/

#include<iostream>

#include<Windows.h>

using namespace std;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

double V\_result, u\_result;

double v1 = 0, v2 = 1, u1 = 1, u2 = 2;

int n;

cout << "Введіть n = ";

while (!(cin >> n) || (cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

cin.clear();

while (cin.get() != '\n');

cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << endl;

cout << "Введіть n = ";

}

while ((n < 3) || (cin.peek() != '\n')) // цикл для перевірки правила на границю числа

{

cin.clear();

while (cin.get() != '\n');

cerr << "! Число n повино бути більше 2" << endl;

cout << "Введіть n = ";

cin >> n;

}

int i = 3;

do

{

u\_result = u2 - (u1 \* v2) - v1;

V\_result = u2 - v2;

cout << "Відповідь V" << i << " = " << V\_result << endl; // цей блок для виведення реулитату кожного кола

//cout << "Відповідь u" << i << " = " << u\_result << endl;

u1 = u2;

v1 = v2;

u2 = u\_result;

v2 = V\_result;

--n;

} while (i <= n);

/\*cout << " Відповідь V = " << V\_result << endl; // цей блок зроблен для виведення результату кінцевого

cout << " Відповідь u = " << u\_result << endl;\*/

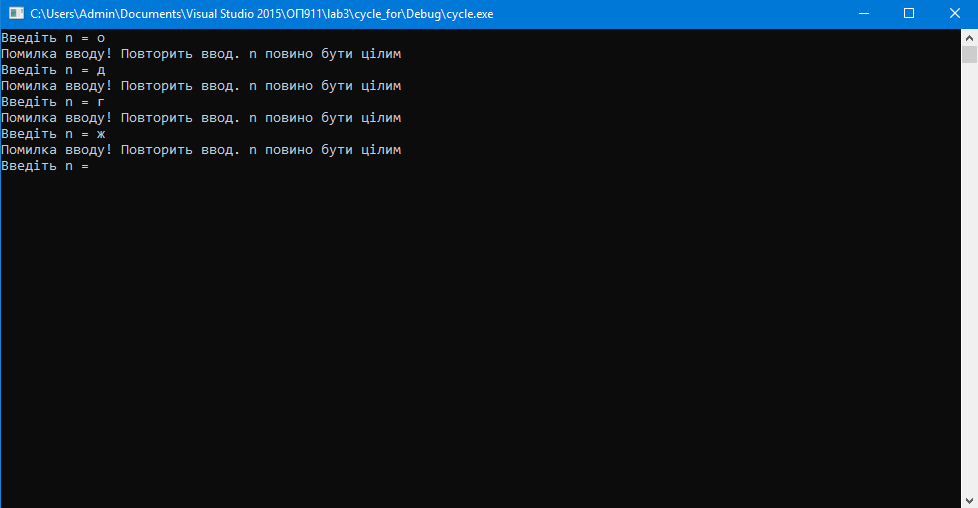
system("pause");

return;

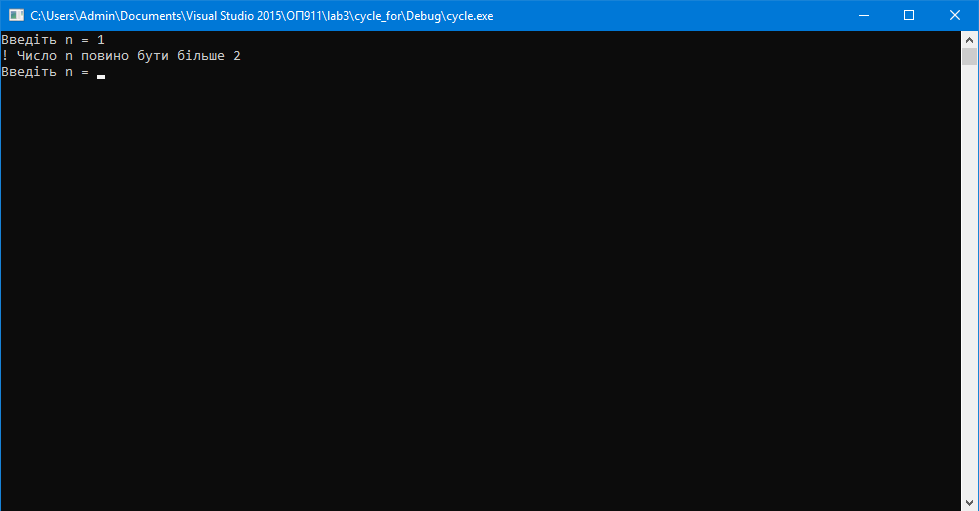
}

4.Результати виповнення програми

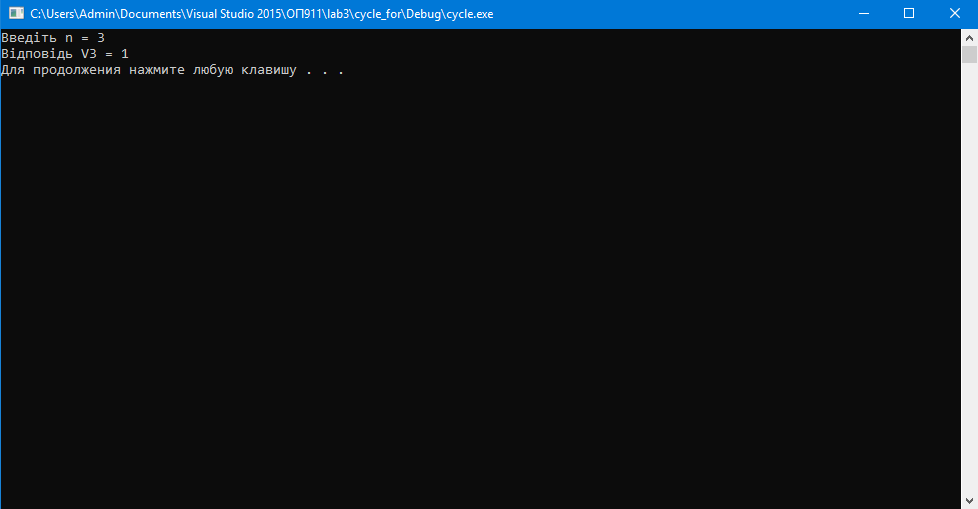
4.1



4.2



4.3



Висновок: цикл - різновид керуючої конструкції в високорівневих мовах програмування, призначена для організації багаторазового виконання набору інструкцій. В даній лабораторній роботі я використав всі види циклів: for, while та do while. В do while мені прийшлось замість інкремента використати декремент, так як do while завжди виконує одну дію и виводив би результат на n+1.