МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних і лабораторних робіт

«Основи програмування мовою С++. Програмування циклів (оператори while, do-while, for) »

з курсів «Інформатика», «Основи інформаційних технологій», «Структури і алгоритми обробки даних» для студентів спеціальностей
151 «Автоматизація та комп'ютерно-інтегровані технології»,
152 «Метрологія та вимірювальна техніка»,
172 «Телекомунікації та радіотехніка»

Рекомендовано
редакційно-видавничою
радою університету,
протокол № 1 від 25.02.2021р.

Харків НТУ
«ХПІ»
2021

Методичні вказівки до виконання практичних і лабораторних робіт «Основи програмування мовою С++. Створення найпростіших програм» з курсу «Інформатика» «Основи інформаційних технологій», «Структури і алгоритми обробки даних», для студентів спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології», 152 «Метрологія та вимірювальна техніка», 172 «Телекомунікації та радіотехніка» денної та заочної форм навчання / уклад.: О. Є. Тверитникова, О. М. Євсеєнко, В. А. Крилова. – Харків : НТУ «ХПІ». – 48 с.

Укладачі: О. Є. Тверитникова
О. М. Євсеєнко
В. А. Крилова

Рецензент Б. М. Горкунов

Кафедра інформаційно-вимірювальних технологій і систем

ВСТУП

Розвиток сучасних технологій неможливий без використання комп’ютерної техніки та програмного забезпечення. Підготовка фахівців у галузі інформаційно-вимірювальної техніки, автоматики, телекомунікацій вимагає широких знань і навичок володіння обчислювальною технікою, знання основ алгоритмізації та програмування мовами високого рівня, такими як *C/C++*.

Методичні вказівки призначені для вивчення мови програмування *C++* під час проведення практичних занять та виконання лабораторних робіт, а також для самостійного освоєння. У методичних вказівках на прикладах розглядаються основні засоби програмування, а також створення блок-схем алгоритмів програм мовою *C++*, а саме: цикл з передумовою *while*, цикл з постумовою *do ... while*, параметричний цикл *for*. Наведено індивідуальні завдання для виконання лабораторних робіт і завдання для самостійного вивчення основ програмування. Розглянуто приклади розв’язання завдань, які допоможуть ефективному освоєнню матеріалу.

ЛАБОРАТОРНА РОБОТА 1

РОЗРОБКА ПРОГРАМ МОВОЮ *C++* З ВИКОРИСТАННЯМ ЦИКЛУ З ПЕРЕДУМОВОЮ *while*

Мета роботи: придбання і закріплення практичних навичок при складанні циклічних програм мовою *C++* з використанням циклу з передумовою ***while***.

Цикл з передумовою *while*

Цикл – це спеціальний оператор мови програмування, за допомогою якого ту чи іншу дію можна виконати потрібну кількість разів, залежно від певної умови. Дії, які повторюються, називаються <*тіло циклу*> (набір повторюваних операторів). Умова, протягом якої дія виконується, називається <*умова виконання циклу*>. При написанні будь-якого циклу треба мати на увазі, що в ньому завжди явно чи неявно присутні чотири елементи: початкові установки, тіло циклу, модифікація параметра циклу і перевірка умови продовження циклу.

Мовою програмування *C++* цикл з передумовою реалізований оператором ***while*** (додаток А, рис. А.1). У циклі з передумовою ***while*** перед початком виконання тіла циклу перевіряється умова виконання циклу. Параметр *умова* – вираз логічного типу, або буде до нього приведений. Спочатку перевіряється параметр *умова*. Якщо значення *true*, то виконується оператор, описаний у тілі циклу. Якщо *false*, то цикл не виконується і відбувається вихід з циклу. Якщо умова завжди *true*, то оператор у тілі циклу буде виконуватися завжди, і цикл стане нескінченним. Необхідно створити ситуацію, коли параметр *умова* приймає значення *false*, для того щоб забезпечити вихід з циклічного алгоритму. Один із способів – використання зміни змінних усередині циклу і перевірка зміни при перевірці умови. Таким чином, перевірка *умови* повторюється при кожному виконанні циклу. Як тільки вона перестає бути вірною, цикл завершується. Якщо умова помилкова (*false*) з самого початку, дія всередині циклу (*тіло циклу*) не буде виконана жодного разу.

Приклад.

Написати програму (використовуючи цикл ***while***), яка знаходить суму всіх цілих чисел від 1 до 10 включно.

```
int main ( )
{
    int  i, s;
    s = 0; // змінна для накопичення суми
    i = 1; // керуюча змінна циклу
    while (i <= 10) // перевірка умови, порівняння керуючої змінної з закінченням діапазону
    {
       s = s + i; // накопичення суми
       i++; // зміна керуючої змінної
    }
    cout<< "s = " << s << '\n'; // відображення результату на екрані
    return 0;
}
```

На початку програми оголошується змінна ***i* = 1,** а також змінна для накопичення суми ***s* = *0***. Далі в умові циклу перевіряється умова – значення змінної ***i*** менше або дорівнює **10**. Оскільки саме від цієї змінної залежить, буде цикл виконуватися чи ні, то така змінна називається керуючою змінною циклу. У тілі циклу виконується накопичення суми ***s* = *s* + *i*** – до попереднього значення змінної ***s*** додається поточне значення змінної ***i*** й нове значення перезаписується у змінну ***s.*** Також у тілі циклу виконується збільшення значення змінної ***i*** на **1**. Дана дія є ***обов'язковою,*** бо якщо не змінювати значення керуючої змінної циклу, результат перевірки умови теж ніколи не зміниться. Це може привести до того, що цикл буде виконуватися ***нескінченно*** (***вічний цикл***). Щоб уникнути таких помилок, потрібно уважно стежити, щоб усередині тіла циклу відбувалася зміна керуючої змінної. Далі програма повертається на перевірку умови циклу (***while (i* <= 10*)***): якщо умова приймає значення *true,* то знову виконується тіло циклу, якщо умова – *false,* то відбувається вихід з циклу і виконання оператора, що йде за циклом, а саме – виведення на екран значення змінної ***s.***

ПРИКЛАДИ РОЗВЯЗАННЯ ЗАВДАНЬ

Завдання 1.1

Скласти програму обчислення виразу $y = 2 \cdot x!$. Результат вивести на екран.

I. *Вибір методу*

Факторіал числа обчислюється як добуток попереднього числа на наступне (збільшене на одиницю). Для цього виразу необхідно скористатися формулою $x! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-1) \cdot x$. Таким чином, для обчислення факторіала числа x у програмі необхідно організувати цикл з початковими установками, рівними одиниці, а також умовою виходу з циклу, коли початкові установки досягнуть значення x. Тіло циклу – добуток факторіала і збільшення на одиницю початкових установок.

II. *Опис розв'язання завдання за допомогою псевдокоду*

1. Початок.
2. Увести з клавіатури значення x.
3. Обчислити значення $f = x!$.
4. Обчислити значення $y = 2 \cdot f$.
5. Вивести результат y на екран.
6. Кінець.

III. *Схема алгоритму роботи програми*

Блок-схему алгоритму програми подано на рис. 1.1.

IV. *Розробка тексту програми*

1. Підключаємо у файлі ***stdafx.h*** необхідні для роботи програми бібліотеки:

 # include <iostream> – для роботи операторів уведення / виведення;

using namespace std;

2. Розробка розділу опису змінних:

int x, y, f, i;

x – ціле число, що вводиться з клавіатури;

f – обчислення факторіала $x!$;

y – обчислення $y = 2 \cdot f$.

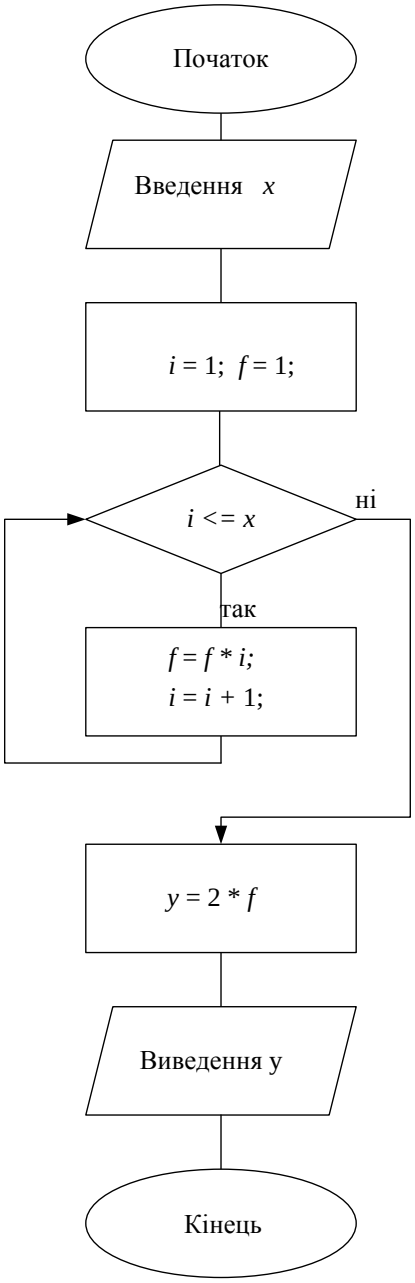

Рисунок 1.1 – Блок-схема алгоритму програми завдання 1.1

3. Розроблення програми.

Уводимо з клавіатури початкові дані – ціле число *x*. Для вказівки користувачеві, що вводити, використовуємо оператор виведення ***cout***<< з відповідним текстом. Потім використовуємо оператор введення ***cin***>> із зазначенням необхідних змінних.

cout<< ” Введіть ціле число x \n”;

cin>> x;

Для обчислення факторіала необхідно встановити

```
i = 1;
f = 1;
```

Цикл з передумовою для обчислення факторіала числа *x*

```
while (i <= x)
{
f = f * i;
i = i + 1;
}
```

Синтаксис програми

```
# include<iostream>
using  namespace std;
int main( )
{
    setlocale(LC_CTYPE,  "ukr");  // дозволити використання української мови
    int x, i, f, y; // опис змінних цілого типу
    cout<<" Уведіть ціле число x \n";
    cin>>x; // уведення значення змінної x
    i = 1;
    f = 1;
    while (i <= x) // умова виконання циклу
    {
    f = f * i; // добуток попереднього значення на наступне
    i = i + 1; // збільшення на одиницю попереднього значення
    }
    y =2 * f;
    cout<< " y= "<<y<<'\n'; // відображення значення y на екрані
    return 0;
}
```

4. Налагодження та запуск програми.

Для налагодження програми використовуємо клавішу *F7*, переконуємось у відсутності помилок і запускаємо програму на виконання за допомогою комбінації клавіш *Ctrl+F5*. Нижче наведені результати роботи програми.

Уведіть ціле число x

5

y = 240.

Завдання **1.2**

Написати програму, яка виводить на екран суму цифр цілого числа *n*, уведеного з клавіатури.

I. *Вибір методу*

Ціле число *n* уводиться з клавіатури користувачем. Завчасно невідомо, яка кількість цифр у цьому числі, тому що число, яке вводиться користувачем, може бути двозначним, тризначним, чотиризначним і т.д. Таким чином, для знаходження суми цифр числа *n* необхідно організувати цикл, в якому кожна цифра числа *n*, починаючи з молодшої, буде знаходитися як залишок від ділення числа *n* на 10. При цьому поточне значення змінної *n* у циклі кожного разу після знаходження остачі від ділення має змінювати своє значення за допомогою виразу: $n = n / 10$. Умовою виходу з циклу є обнулення значення змінної *n*.

Наприклад: ціле число $n = 543$.

1. $a = n \% 10$; $a = 3$;
   $n = n / 10$; $n = 54$;
2. $a = n \% 10$; $a = 4$;
   $n = n / 10$; $n = 5$;
3. $a = n \% 10$; $a = 5$;
   $n = n / 10$; $n = 0$.

Для знаходження суми цифр числа *n* необхідно організувати в тілі циклу вираз для суми, яка буде збільшуватися на значення цифри, і перезаписати нове значення.

II. *Опис рішення задачі за допомогою псевдокоду*

1. Початок.
2. Увести значення *n*.
3. Обчислити значення виразу $n \% 10$.
4. Збільшити значення суми *s*.
5. Обчислити значення $n = n / 10$.
6. Якщо *n* не дорівнює 0, то повторити пп. 3, 4 та 5, інакше – пп. 7.
7. Вивести на екран значення суми *s*.
8. Кінець.

III. *Схема алгоритму розв'язання задачі*

Блок-схему алгоритму програми показано на рис. 1.2.

IV. *Розробка тексту програми*

1. Підключаємо у файлі ***stdafx.h*** необхідні для роботи програми бібліотеки:

 # include <iostream>– для роботи операторів уведення / виведення;

using namespace std;

2. Розробка розділу опису змінних

int n, a, s;

n – ціле число, що вводиться з клавіатури;

a – значення цифри;

s – значення суми.

3. Розробка тіла програми.

Вихідні дані вводимо з клавіатури. Для підказки, що користувач повинен ввести ціле число, використовуємо оператор виведення ***cout<<*** з відповідним текстом. Потім використовуємо оператор уведення ***cin>>*** із зазначенням необхідних змінних:

```
cout<< " Уведіть ціле число n \n ";
cin>>n;
```

Для обчислення суми початкове значення змінної необхідно обнулити

```
s = 0;
```

Знаходження і підсумовування цифр числа *n* виконується в циклі з передумовою

```
while (n != 0)
        {
                a = n % 10;
                s = s + a;
                n = n / 10;
        }
```

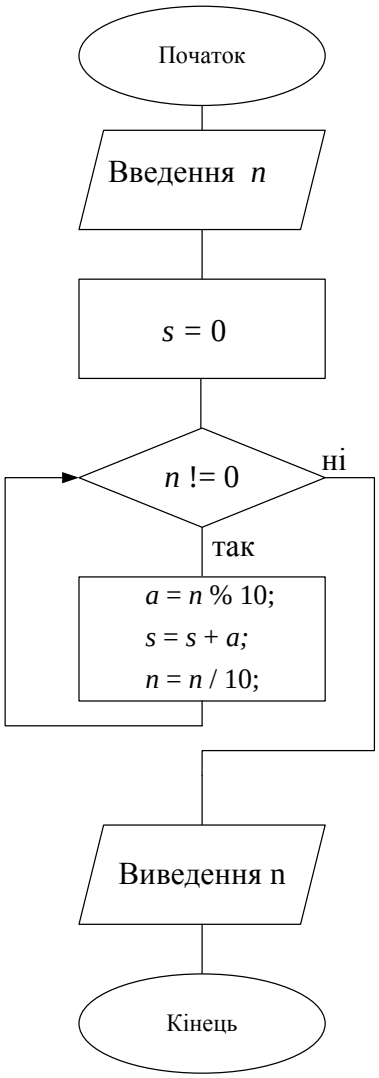

Рисунок 1.2 – Блок-схема алгоритму програми завдання 1.2

Синтаксис програми

```
#include <iostream>
using  namespace std;
int main ( )
{
        int n, a, s; // опис змінних цілого типу
        cout<< " Уведіть ціле число n \n ";
        cin>>n; // уведення з клавіатури цілого числа n
        s = 0; // обнуління суми
        while (n != 0) // умова виходу з циклу з передумовою
        {
                a = n % 10; // обчислення цифри числа n
```

```
        s = s + a; // обчислення суми
        n = n / 10; // зміна значення числа n
    }
    cout<< " Сума цифр числа "<<n<< " = " <<s<<'\n'; // відображення на екрані результату суми
}
```

4. Налагодження та запуск програми.

Для налагодження програми використовуємо клавішу *F7*, переконуємось у відсутності помилок і запускаємо програму на виконання за допомогою комбінації клавіш *Ctrl+F5*. Нижче наведені результати роботи програми.

Уведіть ціле число
849
Сума цифр числа 849 = 21

Порядок виконання лабораторної роботи

1. Відповідно до номера в журналі виберіть індивідуальне завдання.
2. Розробіть алгоритм розв'язання завдання.
3. Складіть текст програми.
4. Створіть проєкт в інтегрованому середовищі розробки *Microsoft Visual Studio* з назвою *lab5_прізвище.cpp.*
5. Уведіть текст програми.
6. Скомпілюйте програму. Якщо в програмі є помилки, їх необхідно виправити. Якщо помилок немає, то з'явиться повідомлення про успішну компіляцію.
7. Запустіть програму на виконання, проаналізуйте результати і переконайтеся в правильності розв'язання завдання.
8. Виконайте звіт з лабораторної роботи, в якому повинні міститися:

− титульна сторінка;
− мета роботи;
− індивідуальне завдання;
− алгоритм роботи програми;
− текст програми;
− результати роботи програми;
− висновки.

Індивідуальні завдання

1. Підрахувати кількість цифр у десятковому записі цілого невід'ємного числа *n* і вивести результат на екран.

2. Визначити, чи є задане натуральне число *n* паліндромом, тобто таким, яке читається однаково зліва направо і справа наліво.

3. Вивести на екран усі дільники заданого натурального числа *n*.

4. Визначити число, отримане виписуванням у зворотному порядку цифр заданого натурального числа *n*. Результат вивести на екран.

5. Скласти програму знаходження суми всіх дільників цілого числа *n*, уведеного з клавіатури. Результат вивести на екран.

6. Скласти програму знаходження добутку всіх дільників цілого числа *n*, уведеного з клавіатури. Результат вивести на екран.

7. Скласти програму знаходження середнього арифметичного всіх дільників цілого числа *n*, уведеного з клавіатури. Результат вивести на екран.

8. Скласти програму обчислення суми квадратів непарних чисел від *a* до *b*, де *a і b* – цілі числа, уведені з клавіатури.

9. Скласти програму обчислення виразу $y = (3 + n)!$. Результат вивести на екран.

10. Послідовність чисел a_0, a_1, a_2, ... утворюється за законом: $a_0 = 1$; $a_k = k \cdot a_{k-1} + 1 / k$, де $(k = 1, 2, \ldots n)$. Натуральне число *n* уводиться з клавіатури. Скласти програму отримання a_1, a_2, ..., a_n. Результат вивести на екран.

11. Скласти програму знаходження найменшого спільного кратного (НСК) двох натуральних чисел.

12. Уведіть послідовність дійсних чисел, доти не введете 0. Скласти програму визначення суми введених чисел.

13. Скласти програму, що визначає, чи є задане ціле число *n* ступенем числа 5. Результат вивести на екран.

14. Скласти програму, що визначає всі цілі числа з проміжку від 100 до 300, в яких сума дільників дорівнює 50. Результат вивести на екран.

15. Скласти програму пошуку перших 20 натуральних чисел, що діляться без остачі на 13 або 17 і більші 500. Результат вивести на екран.

16. Скласти програму для знаходження всіх натуральних розв'язань (*x* та *y*) рівняння $x^2 + y^2 = n$, де *x*, *y* та *n* знаходяться в інтервалі від 1 до 30. Розв'язання, які отримуються від перестановки *x* та *y*, вважати збіжними. Результат вивести на екран.

17. Скласти програму, що визначає кількість точок з цілочисловими координатами, які належать колу радіусом *R* з центром на початку координат.

18. Скласти програму пошуку кількості «щасливих» квитків, у яких сума перших трьох цифр дорівнює суми останніх трьох цифр числа.

19. Скласти програму пошуку кількості тризначних натуральних чисел, сума квадратів яких дорівнює заданому цілому числу *n*. Результат вивести на екран.

20. З клавіатури вводиться ціле число *n*. Скласти програму, яка виводить на екран цифри даного числа *n*.

Контрольні запитання

1. Що буде на екрані після виконання наступного фрагмента коду?

```
int a = 0;
while (a = 1)
{
      if (a % 2 == 1)
      {
        break;
      }
      a++;
}
cout<<a;
```

Варіанти відповідей: **А.** 0; **Б.** 1; **В.** 2; **Г.** Помилка на етапі компіляції; **Д.** Помилка на етапі виконання.

2. Чому дорівнюватиме значення змінної *i* після виконання наступного фрагмента коду?

```
int i = –3, a = –3;
while (a – i) {
        a = i++;
}
cout<<i;
```

Варіанти відповідей: **А.** *i* = –6; **Б.** *i* = –3; **В.** *i* = –2; **Г.** *i* = 1.

3. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 3;
while (i > 0) {
        i ––;
        while (i == 2) {
                break;
                i = –1;
        }
}
cout<<i;
```

Варіанти відповідей: **А.** *i* = 1; **Б.** *i* = 2; **В.** *i* = 0; **Г.** Помилка на етапі компіляції; **Д.** Вічний цикл.

4. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 1, a = 2;
while (i > a)
{
        i++; a++;
}
cout<<i<<a;
```

Варіанти відповідей: **А.** *i* = 1, *a* = 2; **Б.** *i* = 2, *a* = 1; **В.** *i* = 0, *a* = 0; **Г.** Вічний цикл.

5. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 2, a;
while (i != a) {
        a = i % 3; i++;
}
cout<<a;
```

Варіанти відповідей: **А.** *a* = 2; **Б.** *a* = 0; **В.** *a* = 1; **Г.** Вічний цикл.

ЛАБОРАТОРНА РОБОТА 2

РОЗРОБКА ПРОГРАМ МОВОЮ С++ З ВИКОРИСТАННЯМ ЦИКЛУ З ПОСТУМОВОЮ *do … while*

Мета роботи: придбання і закріплення практичних навичок при написанні циклічних програм мовою *С++* з використанням циклу з постумовою ***do … while***.

Цикл з постумовою *do … while*

Мовою програмування *С++* цикл з постумовою реалізований оператором ***do …while*** (додаток Б, рис. Б.1). Цикл з постумовою схожий на цикл з передумовою ***while.*** Різниця полягає в тому, що в циклі з постумовою перевірка умови проводиться відразу ж при вході в цикл, і лише потім, – якщо умова *істинна* – виконується тіло циклу. У циклі з передумовою в будь-якому випадку спочатку виконується тіло циклу і тільки потім іде перевірка умови. Якщо умова *істинна*, виконання дії триває, а якщо ні, то виконання передається наступному за ***while*** оператору. Іншими словами, на відміну від ***while*** всередині циклу з постумовою (***do … while***), тіло циклу хоча б один раз виконується. Якщо синтаксис мови вимагає, щоб у тілі циклу був присутній тільки один оператор, то операторні дужки не вказуються. Якщо в тілі циклу необхідно виконати декілька операторів, то необхідно використовувати складений оператор, тобто укласти тіло циклу в фігурні дужки *{ }*. Оператор ***do … while*** зазвичай використовують, коли цикл потрібно обов’язково виконати хоча б один раз (наприклад, якщо в циклі проводиться введення даних).

Приклад.

Написати програму (використовуючи цикл з постумовою ***do … while***), яка знаходить суму всіх цілих чисел від 1 до 10 включно.

```
int main ( )
{
    int i, s;
    s = 0; // змінна для накопичення суми
    i = 1; // змінна для керування циклу
    do
    {
```

```
        s = s + i; // накопичення суми
        i++; // зміна керуючої змінної
    }
    while (i <= 10); // перевірка умови, порівняння керуючої змінної з закінченням діапазону
    cout<< ” s = ” << s <<‘\n’; // відображення результатів на екрані
    return 0;
}
```

На початку програми оголошується змінна ***i* = 1,** а також змінна для накопичення суми ***s* = 0**. Далі в тілі циклу виконується накопичення суми ***s* = *s* + *i***, до попереднього значення змінної ***s*** додається поточне значення змінної ***i*** й нове значення перезаписується в змінну ***s.*** Також у тілі циклу виконується збільшення значення змінної ***i*** на **1**. Далі відбувається перевірка умови виконання тіла циклу (***while*** (***i*** <= **10**)) – значення змінної ***i*** менше або дорівнює **10**. Якщо умова приймає значення *true,* то знову виконується тіло циклу, якщо умова – *false,* то відбувається вихід з циклу і виконання оператора, що йде слідом за циклом, а саме – виведення на екран значення змінної ***s.***

ПРИКЛАДИ РОЗВ’ЯЗАННЯ ЗАВДАНЬ

Завдання 1.1

Написати програму, яка виводить на екран кількість дільників цілого числа *n*, що вводиться з клавіатури.

І. *Вибір методу*

Дільником цілого числа *n* є число, яке ділиться на *n* без остачі, тобто остача дорівнює нулю. Для пошуку дільників числа *n*, що вводиться з клавіатури, необхідно організувати цикл, у тілі якого буде перевірятися залишок від ділення числа *n* на цілі числа з діапазону від 1 до *n* / 2. При цьому якщо число є дільником, тобто залишок від ділення дорівнює нулю, то необхідно збільшити лічильник на 1.

Наприклад: число 12 має 5 дільників – 1, 2, 3, 4, 6.

ІІ. *Опис розв’язання завдання за допомогою псевдокоду*

1. Початок.

2. Увести значення *n*.

3. Перевірити, чи є число *i* дільником *n*.

4. Якщо *i* – дільник, то збільшити на одиницю лічильник *k*, інакше – повторити пункт 3.

5. Вивести результат *y* на екран.

6. Кінець.

III. *Схема алгоритму роботи програми*

Блок-схему алгоритму програми показано на рис. 1.3.

Рисунок 1.3 – Блок-схема алгоритму програми завдання 1.1

IV. *Розробка тексту програми*

1. Підключаємо у файлі ***stdafx.h*** необхідні для роботи програми бібліотеки:

 # include <iostream> – для роботи операторів уведення / виведення.

using namespace std;

2. Розробка розділу опису змінних:

int n, i, k;

n – ціле число, що вводиться з клавіатури;

i – дільники числа *n*;

k – кількість дільників числа *n*.

3. Розробка тіла програми.

Уводимо з клавіатури вихідні дані – ціле число *n*. Для вказівки користувачеві, що вводити, використовуємо оператор виведення ***cout<<*** з відповідним текстом. Далі використовуємо оператор введення ***cin>>*** із зазначенням необхідних змінних.

```
cout<< " Уведіть ціле число n \n";
cin>> n;
```

Для визначення кількості дільників необхідно обнулити змінну *k* і визначити початкові установки для циклу

```
k = 0;
i = 1;
```

Цикл з постумовою для визначення дільників цілого числа *n*

```
do
{
    if (n % i = = 0)
        {
        k++;
        cout<< i << '\n';
        }
    i++;
}
while (i <= n / 2);
```

Синтаксис програми

```
# include <iostream>
using  namespace  std;
int main ( )
{
    int n, i, k; // Опис змінних цілого типу
    cout<< ”Уведіть ціле число n \n”;
    cin>>n; // уведення значення змінної n
    i = 1; // установлення початкового значення циклу
    k = 0; // обнуління лічильника для визначення кількості дільників
    cout<< ”Дільники числа “ << n << ” : “;
    do// початок циклу з постумовою
    {
      if  (n % i = = 0)  // перевірка умови, чи дорівнює остача від ділення нулю
                {
                k++; // збільшення на одиницю лічильника, якщо і – дільник
                cout<< i << ” ,  “; // виведення дільника на екран
                }
      i++; // наступний дільник
    }
    while (i <= n / 2); // умова виходу з циклу з постумовою
    cout<< “ Кількість дільників числа “ << n << ”–“ << k <<”\n”; // виведення на екран результату
    return 0;
}
```

4. Налагодження та запуск програми.

Для налагодження програми використовуємо клавішу *F7*, переконуємось у відсутності помилок і запускаємо програму на виконання за допомогою комбінації клавіш *Ctrl* + *F5*. Нижче наведені результати роботи програми.

Уведіть ціле число n – 12

Дільники числа 12: 1, 2, 3, 4, 6

Кількість дільників числа 12 – 5

Порядок виконання лабораторної роботи

1. Відповідно до номера прізвища в журналі виберіть індивідуальне завдання.

2. Розробіть алгоритм розв’язання завдання.

3. Складіть текст програми.

4. Створіть проєкт в інтегрованому середовищі розробки *Microsoft Visual Studio* з назвою *lab6_прізвище.cpp.*

5. Уведіть текст програми.

6. Скомпілюйте програму. Якщо в програмі є помилки, їх необхідно виправити. Якщо помилок немає, то з’явиться повідомлення про успішну компіляцію.

7. Запустіть програму на виконання, проаналізуйте результати і переконайтеся в правильності розв’язання завдання.

8. Виконайте звіт з лабораторної роботи, в якому повинні міститися:

− титульна сторінка;

− мета роботи;

− індивідуальне завдання;

− алгоритм роботи програми;

− текст програми;

− результати роботи програми;

− висновки.

Індивідуальні завдання

1. Скласти програму знаходження суми всіх дільників цілого числа *n*, що вводиться з клавіатури.

2. Скласти програму знаходження добутку всіх дільників цілого числа *n*, що вводиться з клавіатури.

3. Увести послідовність цілих чисел, доти вони парні. Скласти програму пошуку їх середнього арифметичного.

4. Увести послідовність дійсних чисел, доти не введете 0. Скласти програму знаходження суми від’ємних чисел.

5. Скласти програму знаходження найбільшого спільного дільника (НСД) двох натуральних чисел, уведених з клавіатури.

6. Скласти програму, що визначає, чи є задане ціле число, уведене з клавіатури, ступенем числа 3.

7. Увести послідовність цілих чисел, доти не введете 0. Скласти програму знаходження середнього арифметичного чисел, кратних 3.

8. Скласти програму пошуку тризначних цілих чисел, що діляться без остачі на 9. Результат вивести на екран.

9. Скласти програму, що визначає всі цілі числа з проміжку від 300 до 600, в яких сума дільників кратна 10.

10. Скласти програму, що визначає, скільки і які двозначні числа відповідають умові: сума квадратів цифр кратна 13.

11. Скласти програму, що визначає кількість точок з цілочисловими координатами, що належать колу радіуса R з центром на початку координат.

12. Скласти програму пошуку всіх тризначних чисел, сума цифр яких ділиться на n без остачі.

13. Скласти програму пошуку кількості тризначних натуральних чисел, сума квадратів цифр яких дорівнює заданому цілому числу n, уведеному з клавіатури.

14. З клавіатури вводиться ціле число n. Скласти програму пошуку середнього арифметичного його цифр.

15. З клавіатури вводиться ціле число n. Визначити, скільки разів у ньому зустрічається цифра a.

16. Відомий факторіал числа n. Скласти програму пошуку цього числа.

17. Знайти всі натуральні числа менше 100, які після піднесення до квадрата утворюють паліндром.

18. Уведіть послідовність дійсних чисел, доти не введете 0. Знайти суму тризначних додатних чисел.

19. Вивести на екран усі дільники цілого числа n, що вводиться з клавіатури.

20. Надрукувати в порядку зростання всі числа від 100 до 999, в десятковому записі яких немає однакових цифр.

Контрольні запитання

1. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 0, a = 3;
do
{
    i++;
    a++;
}
while (i > a);
cout<<i<<a;
```

Варіанти відповідей: **А.** 0, 3; **Б.** 1, 3; **В.** 2, 4; **Г.** Помилка на етапі компіляції; **Д.** 1, 4.

2. Чому дорівнюватиме значення змінної *i* після виконання наступного фрагмента коду?

```
int i = –2;
int  a = –3;
do
{
    a = i++;
}
while (a != 2);
cout<<”i=”<<i;
```

Варіанти відповідей: **А.** *i* = 3; **Б.** *i* = –3; **В.** *i* = –2; **Г.** *i* = 1; **Д.** Помилка на етапі виконання.

3. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 3;
do
{
    i –=;
    while (i == 2)
    {
        break;
        i = –1;
    }
}
```

```
while (i > 0)
cout<<" i= "<<i;
```

Варіанти відповідей: **А.** $i = 1$; **Б.** $i = 2$; **В.** $i = 0$; **Г.** Помилка на етапі компіляції; **Д.** Вічний цикл.

4. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 1, a = 2;
do
{
        i++; a++;
}
while (i > a);
cout<<"i="<<i<<"a="<<a;
```

Варіанти відповідей: **А.** $i = 3$, $a = 2$; **Б.** $i = 2$, $a = 1$; **В.** $i = 2$, $a = 3$; **Г.** Вічний цикл; **Д.** Помилка на етапі виконання.

5. Що буде на екрані після виконання наступного фрагмента коду?

```
int i = 2, a;
do
{
        a = i % 3; i++;
        cout<<a;
}
while (i != a);
```

Варіанти відповідей: **А.** $a = 2$; **Б.** $a = 0$; **В.** $a = 1$; **Г.** Вічний цикл.

ЛАБОРАТОРНА РОБОТА 3
РОЗРОБКА ПРОГРАМ МОВОЮ С++ З ВИКОРИСТАННЯМ ЦИКЛУ З ПАРАМЕТРАМИ *for*

Мета роботи: придбання і закріплення практичних навичок при написанні циклічних програм мовою С++ з використанням циклу з параметрами ***for.***

Параметричний цикл *for*

Цикл з параметрами (параметричний цикл) ***for*** дозволяє зручно і наочно описувати необхідні параметри циклу тоді, коли заздалегідь відомі початкове і кінцеве значення параметра циклу, а також відомо, як змінюється параметр циклу. Даний оператор теоретично є повною аналогією циклу ***while,*** а практично дозволяє організувати цикл з більш зручним управлінням. Оператор ***for*** використовується для організації циклів з лічильниками, тобто з цілочисловими змінними, які змінюють своє значення при кожному проходженні циклу (наприклад, збільшуються на одиницю). Блок-схему алгоритму і синтаксис параметричного циклу ***for*** подано на рис. В.1 (додаток В). Синтаксис параметричного циклу ***for*** мовою програмування *С++* має вигляд:

```
for (ініціалізація_змінної; перевірка_умови; зміна_змінної)
{
        тіло циклу;
}
```

Основні параметри і елементи циклу

- Початкові установлення (*ініціалізація_змінної*) – задання початкового значення, яке приймає змінна – параметр циклу перед першим виконанням циклічної дії. При цьому задання та ініціалізація циклу завжди виконуються тільки один раз.
- Тіло циклу (*тіло циклу*) – оператори, які виконуються в циклі.
- Модифікація параметра циклу (*зміна_змінної*) – вираз, який визначає, як змінюється змінна параметра циклу.

• Перевірка умови (*перевірка_умови*) – кінцеве значення, яке визначає умову виходу з циклу. Цикл закінчується, коли змінна параметра циклу досягає кінцевого значення.

Принцип роботи циклу з параметрами for

1. Ініціалізація початкового значення змінної.
2. Перевірка умови.
3. Виконання тіла циклу, якщо умова істинна.
4. Якщо умова помилкова – виконання наступного за циклом оператора.
5. Якщо умова була істинна – зміна змінної параметра циклу.
6. Перевірка умови. Далі знову виконуємо пп. 3 і 4.

Приклад.

Написати програму (використовуючи цикл з параметрами ***for***), яка знаходить суму всіх цілих чисел від 1 до 10 включно:

```
int main ( )
{
    int  i, s;
    s = 0; // змінна для накопичення суми
    for (i = 1; i <= 10; i++) // параметри циклу
       s = s + i; // накопичення суми
    cout<< ”s = ” << s << ‘\n’; // відображення результату на екрані
    return 0;
}
```

Розглянемо приклад використання циклу ***for*** у програмі, яка виводить на екран числа від 1 до 5.

```
int main ( )
{
    for (int i = 1; i <= 5; i++)
       cout<< i;
    return 0;
}
```

У параметрах циклу ***for*** ініціалізується змінна ***i,*** що дорівнює **1**. Далі здійснюється перевірка значення цієї змінної за допомогою умови ***i*** **<= 5**. Якщо умова істинна (так буде доти, доки ***i*** не досягне значення 6), виконується тіло циклу, а саме – виведення на екран значення ***i (cout<< i)***

та зміна значення параметра циклу на **1** (***i***++). Далі знову перевіряється умова. Якщо умова помилкова (тобто значення змінної параметра циклу стало рівне 6), то програма переходить на наступний рядок за тілом циклу.

Ініціалізація керуючої змінної

1. Ініціалізація і створення змінної здійснюються в циклі

```
for (int x = 1; x <= 100; x++)
    cout<< x;
```

2. Створення змінної проводиться до циклу, а ініціалізація – в тілі циклу:

```
int x;
for (x = 1; x <= 100; x++)
    cout<< x;
```

3. Ініціалізація та створення змінної проводяться до циклу

```
int x = 1;
for (; x <= 100; x++)
    cout<< x;
```

Зміна керуючої змінної

Зміну керуючої змінної можна перенести всередину тіла циклу, як це відбувається в ***while*** та ***do…while.***

```
for (int x = 1; x <= 100;)
  {
        cout<< x;
        x++;
  }
```

Умова

Умову конструкції також можна пропустити, однак у цьому випадку вона буде вважатися за умовчанням істинною. Таким чином, ми отримуємо постійно справжню умову і як наслідок ВІЧНИЙ ЦИКЛ.

```
for (int x = 1;  ; x++)
        cout<< x;
```

Вкладені цикли

Принцип роботи програми, що реалізує вкладений цикл, базований на тому, що внутрішній цикл повністю виконується на кожному кроці зовнішнього циклу від початку і до кінця. Іншими словами, доти програма не вийде з вкладеного циклу – виконання зовні не продовжиться.

Приклад.

Написати програму, яка виводить на екран таблицю множення.

```
# include <iostream>
using  namespace  std;
int main ( )
{
   int i, j, p;
   for (i = 1; i < 10; i++)
   {
       for (j = 1; j < 10; j++)
       {
              p = i * j;
              cout<< p << ”\t”;
       }
   cout<<‘\n’;
   }
   return 0;
}
```

Блок-схему до даної програми показано на рис. 1.4.

Коментар до програми.

1. Керуючі змінні зовнішнього і внутрішнього циклів (***i, j***) здійснюють функції множників.

2. Керуюча змінна ***i*** створюється та ініціалізується значенням 1.

3. Програма перевіряє умову ***i*** **<** **10**, тому що 1 менше 10, умова є істинною і програма входить у зовнішній цикл.

4. Керуюча змінна ***j*** створюється та ініціалізується значенням 1.

5. Програма перевіряє умову ***j*** **<** ***10***, тому що 1 менше 10, умова є істинною і програма входить у внутрішній цикл.

6. Здійснюється показ на екрані добутку ***i*** на ***j – 1.***

7. Здійснюється зміна керуючої змінної ***j.***

8. Знову перевіряється умова ***j < 10***, тому що 2 менше 10, умова є істинною і програма знову входить у внутрішній цикл.

9. Здійснюється відображення на екрані добутку ***i*** на ***j – 2***.

10. Здійснюється зміна керуючої змінної ***j***.

Дії з 5 по 7 – повторюються до тих пір, доти ***j*** не стане дорівнювати 10, при цьому поточне значення ***i = 1*** множиться на кожне значення ***j*** (від 1 до 9 включно), результат виводиться на екрані. Виходить рядок таблиці множення на 1. Потім програма виходить з внутрішнього циклу і переводить екранний курсор на два рядки вниз. Після цього здійснюється збільшення змінної ***i*** на одиницю і знову вхід у внутрішній цикл, тепер уже для виведення ланцюжка множення на 2. Таким чином, на екрані з’являється таблиця множення:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

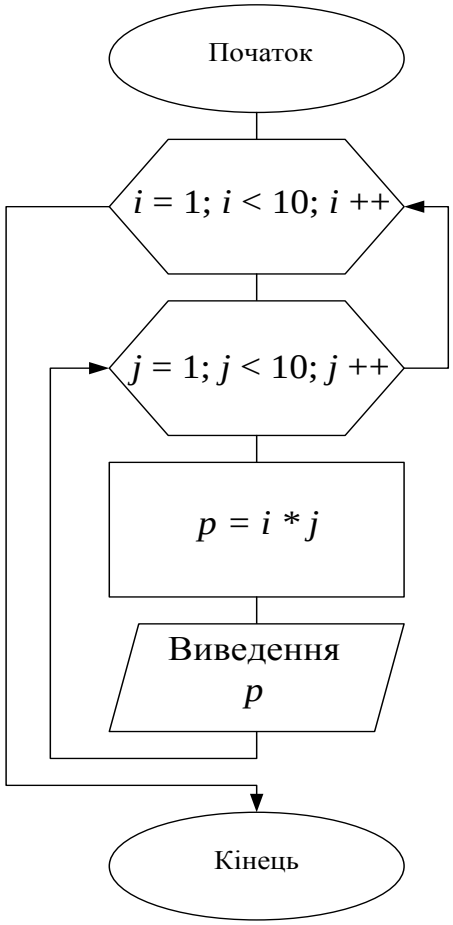

Рисунок 1.4 – Блок-схема алгоритму програми

ПРИКЛАДИ РОЗВЯЗАННЯ ЗАВДАНЬ

Завдання 1.1

Обчислити суму ряду $y = x^{10} + 2 \cdot x^9 + 3 \cdot x^8 + \ldots + 10 \cdot x + 11$. Округлити отримане значення y до найближчого цілого числа. Визначити, чи є отримане число:

1) простим числом;

2) досконалим числом, тобто рівним сумі всіх своїх додатних дільників, крім самого цього числа. (Наприклад, число 6 є досконалим, 6 = 1 + 2 + 3, де 1, 2 та 3 – дільники числа 6).

І. *Вибір методу*

Суму ряду $y = x^{10} + 2 \cdot x^9 + 3 \cdot x^8 + \ldots + 10 \cdot x + 11$ можна обчислити за допомогою послідовного накопичення або за схемою Горнера, використовуючи рівність:

$$y = x^{10} + 2 \cdot x^{9} + 3 \cdot x^{8} + ... + 10 \cdot x + 11 =$$

$$= 11 + x(10 + x(9 + x(8 + x(7 + x(6 + x(5 + x(4 + x(3 + x(2 + x))))))))$$

Для округлення значення змінної *y* необхідно скористатися оператором приведення типу ***int k*** = (***int***) ***y***. Для визначення, чи є число *z* простим, необхідно організувати цикл з умовою знаходження дільників числа *z* в інтервалі від 2 до *z* – 1. Просте число в зазначеному інтервалі дільників не має. Для визначення, чи є число досконалим, необхідно організувати цикл з пошуком дільників цього числа, а також накопичення суми всіх додатних дільників.

II. *Опис розв'язання завдання за допомогою псевдокоду*

1. Початок.
2. Увести значення *x*.
3. Обчислити *y*.
4. Визначити, чи є округлене число *y* простим.
5. Визначити, чи є округлене число *y* досконалим.
6. Кінець.

III. *Схема алгоритму роботи програми*

Блок-схему алгоритму програми подано на рис. 1.5., рис. 1.6.

IV. *Розробка тексту програми*

1. Підключаємо у файлі ***stdafx.h*** необхідні для роботи програми бібліотеки:

 # include <iostream> – для роботи операторів уведення / виведення;

using namespace std;

2. Розробка розділу опису змінних

float y, x;

int s, i, k;

bool r;

x – дійсне число, що вводиться з клавіатури;

y – дійсне значення функції;

k – округлене значення змінної *y;*

i – дільники числа *k*;

s – сума дільників числа *k*;

r – логічне значення прапора.

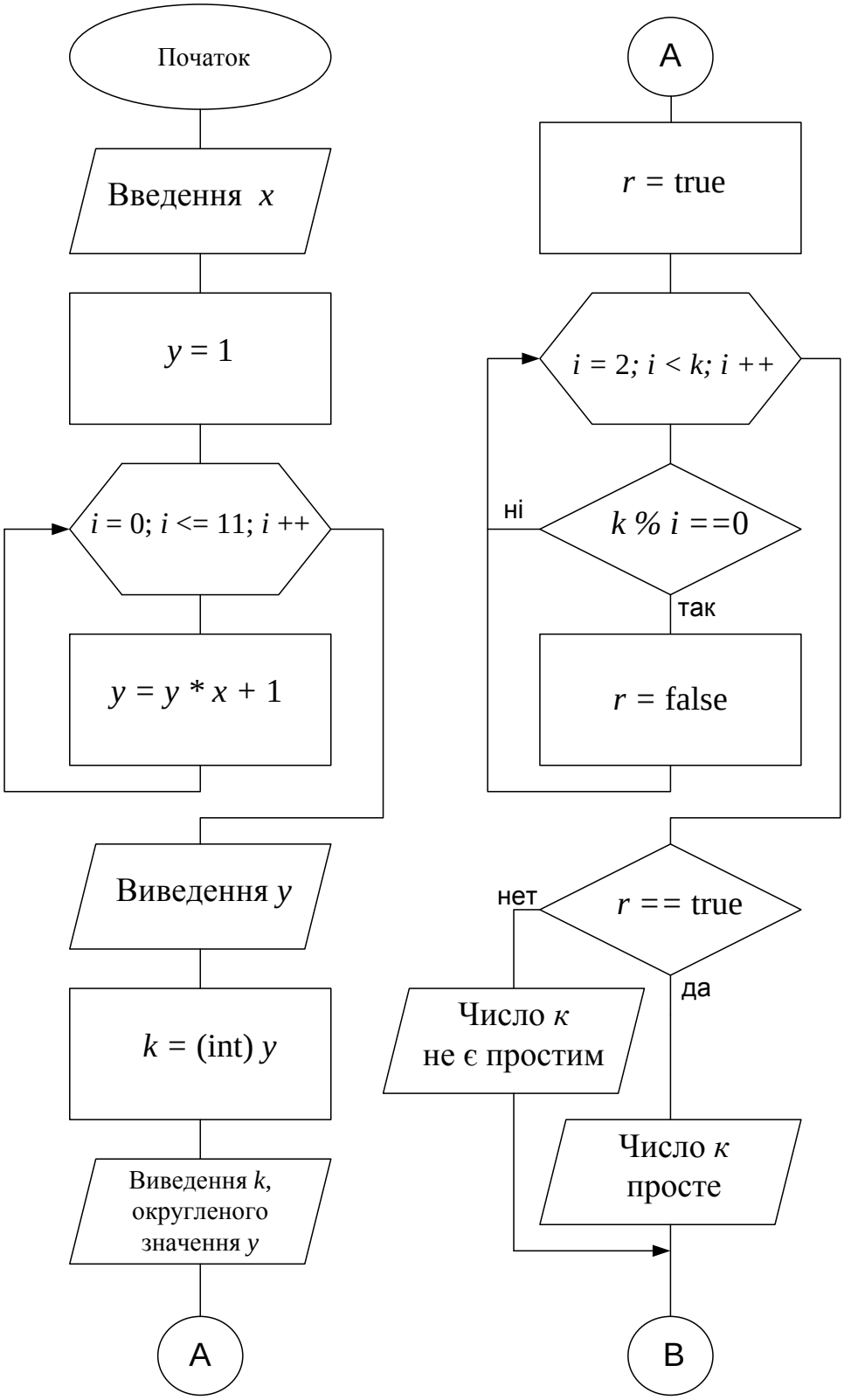

Рисунок 1.5 – Блок-схема алгоритму програми завдання 1.1 (початок)

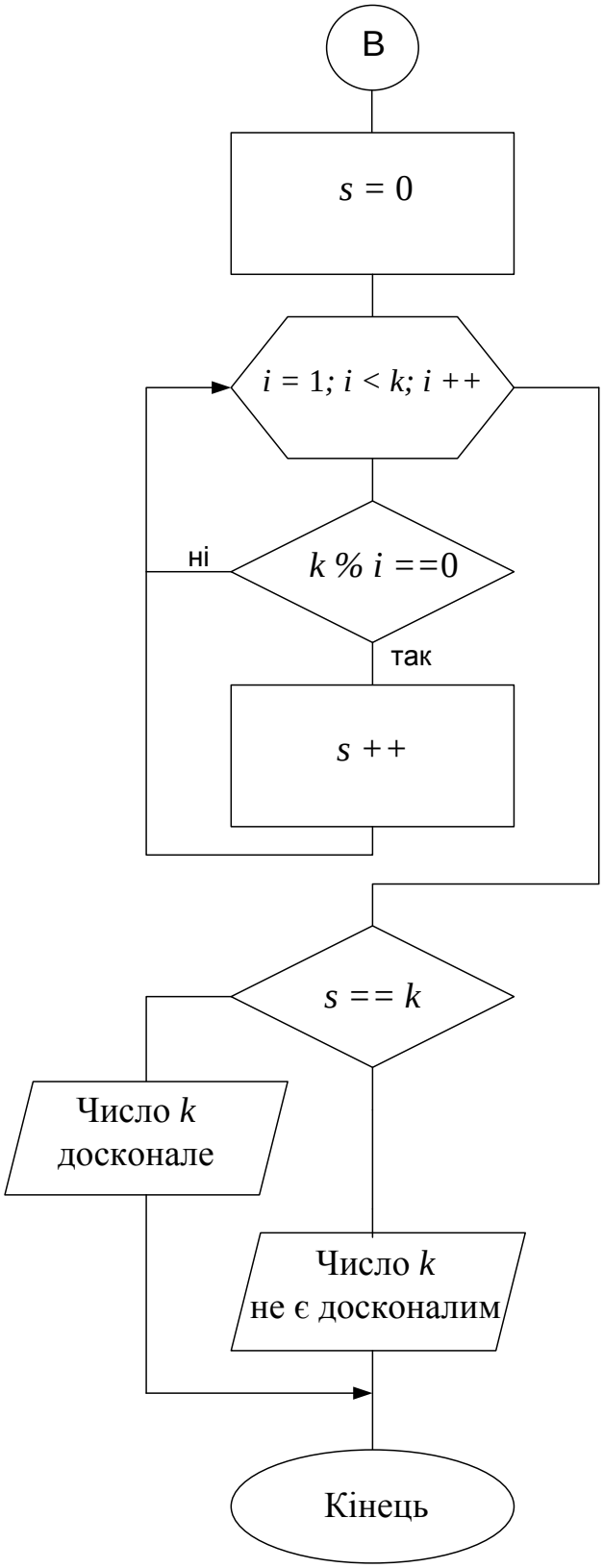

Рисунок 1.6 – Блок-схема алгоритму програми завдання 7.1 (продовження)

3. Розробка тіла програми.

Уводимо з клавіатури вихідні дані – дійсне число *x*. Для того щоб указати користувачеві, що вводити, використовуємо оператор виведення ***cout***<< з відповідним текстом. Потім використовуємо оператор уведення ***cin***>> із зазначенням необхідних змінних.

```
cout<< ” Уведіть дійсне число x \n”;
cin>> x;
```

Для обчислення суми рядка $y = x^{10} + 2 \cdot x^9 + 3 \cdot x^8 + \ldots + 10 \cdot x + 11$ необхідно організувати параметричний цикл, у тілі циклу якого буде послідовне накопичення суми:

```
y = 1;
f = 0;
for (i = 0; i <= 11; i++)
        y = y*x + i;
```

Для округлення значення *y* до цілого числа скористаємося оператором приведення типу:

```
int k = (int) y;
```

Для того щоб визначити, чи є округлене число *y* (ціле число *k*) простим числом, необхідно встановити прапор – значення логічної змінної *r*, рівне *true*, та організувати цикл з параметрами для пошуку дільників числа *k* *y* діапазоні від 2 до $k - 1$.

```
r = true;
for (i = 2; i < k; i++)
        if (k % i = = 0)
                r = false;
if (r = = true)
        cout<< “ Число просте”;
else
        cout<< “ Число не є простим ”;
```

Для визначення, чи є округлене число *y* (у подальшому це ціле число *k*) досконалим числом, за допомогою циклу з параметрами організуємо пошук дільників числа *k* з накопиченням суми цих дільників, потім порівняємо отриману суму зі значенням змінної *k:*

```
s = 0;
for (i = 1; i < k; i++)
        if (k % i = = 0)
```

```
            s++;
if (s = = k)
        cout<< " Число досконале ";
else
        cout<< "Число не є досконалим ";
```

Синтаксис програми

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

int main ( )
{
    float y, x;
    int s, i, k;
    bool r;
    // уведення з клавіатури значення змінної x
    cout<< " Уведіть значення змінної x \n";
    cin>> x;
    // цикл обчислення значення виразу y
    y= 1; // установлення початкового значення
    for (i = 0; i <= 11; i++)
       y = y*x + i; // накопичення суми
    cout<< " y = "<<y<<'\n';
    k = (int) y; // округлення змінної y до найближчого цілого числа
    cout<< " Округлене значення y = "<< k << '\n';
    // визначення, чи є число k простим
    r = true; // установлення початкового значення
    for (i = 2; i < k; i++) // цикл для знаходження дільників числа k
       if (k % i = = 0) // перевірка умови, чи є і дільником числа k
            r = false;
    if (r = = true) // перевірка, чи змінна r змінила своє значення
       cout<< " Число "<< k << " просте " <<'\n';
    else

       cout<< " Число " << k << " не є простим " <<'\n';
    // визначення, чи є число k досконалим
```

```
    s = 0; // установлення початкового значення
    for (i = 1; i < k; i++) // цикл для знаходження дільників числа k
       if (k % i = = 0) // перевірка, чи є і дільником числа k
             s++; // накопичення суми дільників
    if (s = = k) // перевірка, чи дорівнює сума дільників числу k
       cout<< " Число " << k << " досконале " << '\n';
    else
       cout<< " Число " << k << " не є досконалим " <<'\n';
    return 0;
}
```

4. Налагодження і запуск програми.

Для налагодження програми використовуємо клавішу *F7*, переконуємось у відсутності помилок і запускаємо програму на виконання за допомогою комбінації клавіш *Ctrl+F5*. Нижче наведено результати роботи програми.

Уведіть значення змінної x
0.3
y = 15.1020419
Округлене значення y = 15
Число 15 *не є простим*
Число 15 *не є досконалим*

Порядок виконання лабораторної роботи

1. Відповідно до номера свого прізвища в журналі виберіть індивідуальне завдання.
2. Розробіть алгоритм вирішення завдання.
3. Складіть текст програми.
4. Створіть проєкт в інтегрованому середовищі розробки *Microsoft Visual Studio* з назвою *lab7_прізвище.cpp*.
5. Уведіть текст програми.
6. Скомпілюйте програму. Якщо в програмі є помилки, їх необхідно виправити. Якщо помилок немає, то з'явиться повідомлення про успішну компіляцію.
7. Запустіть програму на виконання, проаналізуйте результати і переконайтеся в правильності розв'язання завдань.

8. Виконайте звіт з лабораторної роботи, в якому повинні міститися:

− титульна сторінка;

− мета роботи;

− індивідуальне завдання;

− алгоритм роботи програми;

− текст програми;

− результати роботи програми;

− висновки.

Індивідуальні завдання

1. Написати програму, яка обчислює функцію $y(x) = 10x!$. Результат вивести на екран.

2. Обчислити найбільший спільний дільник двох цілих чисел, уведених з клавіатури. Результат вивести на екран.

3. Підрахувати кількість цифр у десятковому записі цілого невід'ємного числа n, уведеного з клавіатури.

4. Скласти програму пошуку перших n парних чисел (значення зміної n уводиться з клавіатури) у послідовності чисел Фібоначчі.

5. Скласти програму пошуку всіх простих чисел у проміжку [$n1$; $n2$], де $n1$ та $n2$ – два цілих числа, уведених з клавіатури.

6. Скласти програму, яка визначає, чи є ціле число n, уведене з клавіатури, – простим.

7. Скласти програму, яка виводить перші n чисел Фібоначчі.

8. Написати програму, яка обчислює суму ряду $y = \cos x^2 + \cos x^3 + \cos x^4 + \ldots + \cos x^{20}$. Результат вивести на екран.

9. Написати програму, яка обчислює суму ряду $y = 1! + 2! + 3! + \ldots + n!$, $(n > 1)$. Результат вивести на екран.

10. Обчислити суму квадратів усіх цілих чисел, що потрапляють в інтервал $(\ln x, e^x)$, де $x > 1$ – дійсне число, уведене з клавіатури.

11. З клавіатури вводиться число Фібоначчі. Скласти програму, яка визначить його порядковий номер у послідовності Фібоначчі.

12. Обчислити кількість точок з цілочисловими координатами, що потрапляють у коло радіусом R ($R > 0$) з центром на початку координат.

13. Надрукувати в порядку зростання всі числа від 100 до 999, в десятковому записі яких немає однакових цифр.

14. Визначити кількість тризначних цілих чисел, сума цифр яких дорівнює *n*. Результат вивести на екран.

15. Визначити, чи є задане натуральне число паліндромом, тобто таким, яке читається однаково зліва направо і справа наліво.

16. Визначити число, отримане виписуванням у зворотному порядку цифр заданого натурального числа.

17. Написати програму пошуку всіх натуральних чисел менше 100, які при зведенні в квадрат дають паліндром.

18. Написати програму пошуку максимального натурального числа з інтервалу від *a* до *b* з максимальною сумою дільників.

19. Скласти програму всіх дружніх чисел в інтервалі від 1 до 350 (числа є дружніми, якщо кожне з них дорівнює сумі дільників іншого).

20. Якщо до суми цифр двозначного числа додати квадрат різниці цифр, то вийде те ж саме число. Скласти програму пошуку всіх таких чисел.

Контрольні запитання

1. Що буде на екрані в результаті виконання наступного фрагмента коду?

```
int   k = 20;
for (k = 3; k < 20; k++)
{
    k++;
    cout<<" 1 ";
}
```

Варіанти відповідей:

А. 20 одиниць

Б. 9 одиниць

В. 17 одиниць

Г. Помилка на етапі компіляції

Д. Помилка на етапі виконання

2. Що буде на екрані в результаті виконання наступного фрагмента коду?

```
int k;
for (k = 0; k < 10; k++)
    k += k;
cout<<k;
```

Варіанти відповідей:

А. 10

Б. 15

В. 20

Г. Вічний цикл

Д. Помилка на етапі компіляції

3. Що буде на екрані в результаті виконання наступного фрагмента коду?

```
for (int i = 0; i < 10; i++)
    for (int j = 0; j < 10; j++)
       if (i == 0)
    {
            cout<<“$$”;
       else
            cout<<“&&”;
    }
```

Варіанти відповідей:

А. $$&&

Б. &&$$

В. $&$&

Г. &$&$

Д. Помилка на етапі компіляції.

4. Що буде на екрані в результаті виконання наступного фрагмента коду?

```
for (int i = 0; i < 2; i++)
    for (int j; j < 3; j++)
       cout<<1;
```

Варіанти відповідей:

А. 11111

Б. 111111

В. Вічний цикл

Г. 1

Д. Помилка на етапі компіляції.

5. Що буде на екрані в результаті виконання наступного фрагмента коду?

```
for (int i = 0; i < 3; i++)
    for (int j = 2; j <= 10; j––)
       cout<<j;
```

Варіанти відповідей:

А. На екран нічого не виводиться

Б. Вічний цикл

В. Помилка на етапі компіляції

Г. 2

Д. 21.

ДОДАТКИ

ДОДАТОК А

Цикл з передумовою *while*

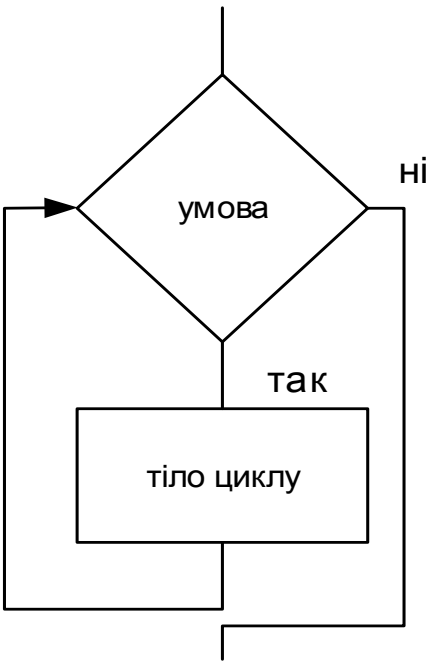

Рисунок А.1 – Блок-схема циклу з передумовою ***while***

Синтаксис циклу з передумовою ***while***

```
while (умова)
        оператор;
```

Складовий оператор

```
while (умова)
{
  оператор 1;
  оператор 2;
  …;
}
```

ДОДАТОК Б

Цикл з передумовою *do … while*

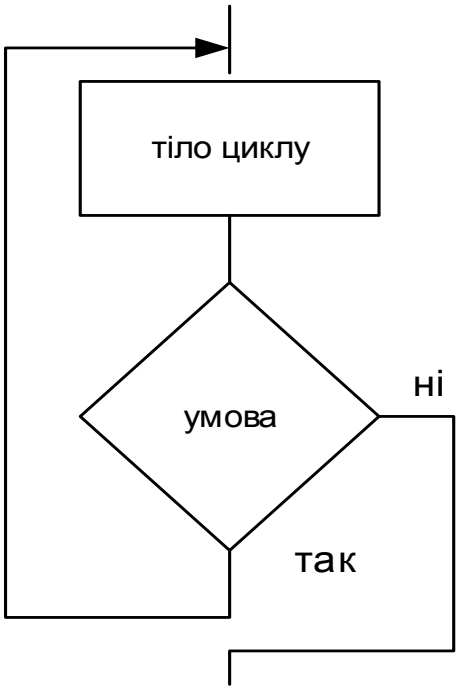

Рисунок Б.1 – Блок-схема циклу з постумовою ***do … while***

Синтаксис циклу з постумовою ***do … while***

```
do
{
    оператор;
}
while (умова);
```

Складовий оператор

```
do
{
    оператор 1;
    оператор 2;
    …,
}
while (умова);
```

ДОДАТОК В

Цикл з параметрами *for*

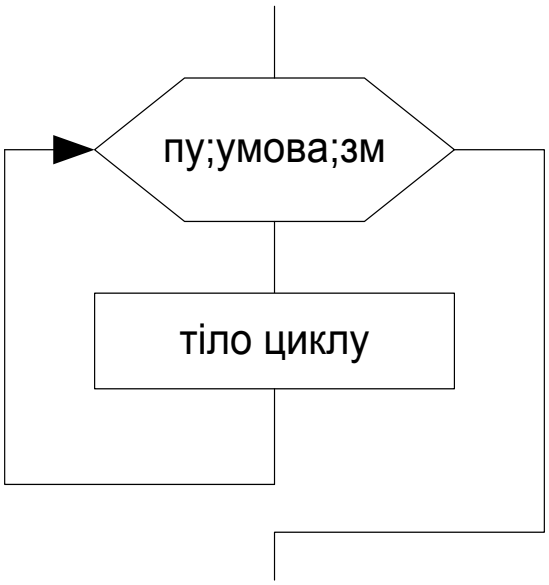

Рисунок В.1 – Блок-схема циклу з параметрами ***for***

Синтаксис циклу з параметрами ***for***

for (Початкові Умови; Умова; Зміна Змінної)

оператор;

Складовий оператор

for (ПУ; умова; ЗМ)
{
оператор 1;
оператор 2;
…
оператор n;
}

ДОДАТОК Г

Оператор завершення, продовження циклу

Оператор завершення використовується в операторах *while, do..while, for, switch*. Призначення оператора – припинення виконання зазначених вище операторів, тобто раніше, ніж це передбачено синтаксисом.

Оператор завершення задається ключовим словом *break*.

Розглянемо виконання оператора на прикладі розв'язання наступної задачі. Необхідно обчислити суму довільної кількості введених чисел, додаток припиняє виконання при введенні негативного числа.

```
#include <iostream>
void main()
{
int sum = 0, val;
for (int i = 0; i <100; ++ i)
        {
        cin >> val;
        if (val> = 0)
                s + = val;
        else
                break;
        }
cout << «Сума =« << sum << '\ n';
}
```

У прикладі змінна *sum* використовується для накопичення суми, змінна *val* служить для введення поточного значення послідовності, змінна і оголошена в параметричному циклі і є його лічильником. У тілі циклу здійснюється введення нового значення послідовності. Далі виконується перевірка, чи є введене число позитивним. Якщо умова виконується, то відбувається накопичення часткової суми послідовності. Якщо ж було введено від'ємне значення, то умова оператора *if* стає хибною, виконується оператор *break* і виконання циклу закінчується.

Оператор продовження. Оператор продовження використовується в операторах *while, do..while, for*. Призначення оператора – припинення виконання поточної ітерації циклу і перехід на перевірку умови завершення циклічних дій.

Оператор продовження задається ключовим словом *continue*.

Розглянемо такий приклад. Обчислимо суму невід'ємних чисел, що вводяться з клавіатури, причому введених чисел повинно бути 10.

```
#include <iostream.h>
void main()
{
int sum = 0, val;
for(int i =0; i <10; ++ i)
        {
        cin >> val;
        if(val <0)
                continue;
        sum + = val;
        }
cout << «Сума =« << sum << '\ n';
}
```

Якщо виконується умова *val*<0, то виконується оператор continue. При цьому пропускається в тілі циклу всі оператори, які йдуть за оператором continue. Відбувається перехід до заголовку циклу, змінюється значення параметра циклу, перевіряється умова виконання і т.д.

СПИСОК ЛІТЕРАТУРИ

1. Васильченков О.Г. Основы алгоритмизации и программирования на С++. Учебное пособие / О.Г. Васильченков, О.Є. Тверитникова, В.А. Крылова. – Харьков: НТУ «ХПІ», 2015. – 228 с.

2. Тверитникова О.Є. Базові алгоритми та основи програмування. Теорія і практика. Навчальний посібник / О.Є. Тверитникова, В.А. Крилова, О.Г. Васильченков. – Харків, 2020. – 264 с.

3. Тверитникова О.Є. Методичні вказівки з інформатики «Арифметичні та логічні основи вимірювальної техніки» / О.Є. Тверитникова, В.А. Крилова, О.Г. Васильченков. – Харків, 2013. – 52 с.

4. Тверитникова О.Є. Методические указания по информатике «Основы программирования на С++». Часть 1. / О.Є. Тверитникова, В.А. Крылова. – Харьков: НТУ «ХПІ», 2013. – 52 с.

5. Тверитникова Е.Е. Методические указания по информатике «Основы программирования на С++». Часть 2. / Е.Е. Тверитникова, В.А. Крилова, М.И. Опришкина. – Харків: НТУ «ХПІ», 2014. – 68 с.

6. Тверитникова Е.Е. Методические указания по информатике «Основы программирования на С++». Часть 3. Двухмерные массивы. / Е.Е. Тверитникова, В.А. Крилова. – Харків: НТУ «ХПІ», 2017. – 52 с.

7. Тверитникова О.Є. Методичні вказівки з дисципліни «Основи інформаційних технологій», «Структури і алгоритми обробки даних». Сортування мовою програмування С++ / О.Є. Тверитникова, В.А. Крилова. – Харків: НТУ «ХПІ», 2020. – 28 с.

8. Тверитникова О.Є., Крилова В.А. Методичні вказівки з дисципліни «Основи інформаційних технологій», «Структури і алгоритми обробки даних». Використання функцій / О.Є. Тверитникова, В.А. Крилова. – Харків: НТУ «ХПІ», 2020. – 32 с.

9. Програмування мовами С та С++ в середовищі Windows. Навчальний посібник. Вінниця: УНІВЕРСУМ-Вінниця, 2003. – 128 с.

10. Ткачук В.М. Програмування на С++. Лабораторний практикум Івано-Франківськ: Видавництво Прикарпатського національного університету імені Василя Стефаника / В.М. Ткачук, 2011. – 160 с.

11. Жуковський С.С. Програмування мовою С++. Структурне програмування (лабораторний практикум). Навчальний посібник для студентів фізико-математичного факультету / С.С. Жуковський, Т.А. Вакалюк. Житомир: Вид-во ЖДУ, 2011. – 92 с.

ЗМІСТ

Навчальне видання

Укладачі: ТВЕРИТНИКОВА Олена Євгенівна
ЄВСЕЄНКО Олег Миколайович
КРИЛОВА Вікторія Анатоліївна

ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ С++. ПРОГРАМУВАННЯ ЦИКЛІВ (ОПЕРАТОРИ WHILE, DO-WHILE, FOR)

Методичні вказівки
до виконання практичних та лабораторних робіт

для студентів спеціальностей
151 «Автоматизація та комп'ютерно-інтегровані технології»,
152 «Метрологія та вимірювальна техніка»,
172 «Телекомунікації та радіотехніка»

Відповідальний за випуск Качанов П. О.
Роботу до друку рекомендував Дудник О. В.

Редактор О.І. Шпільова

План 2021 р., Поз. 73

Підписано до друку 29.03.21 р. Формат 60Х84 1/16. Папір друк. № 2.
Друк – ризографія. Гарнітура Times New Roman. Умов. друк. арк. 3.
Наклад 100 прим. Зам. № . Ціна договірна.

Видавець Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Друкарня НТУ «ХПІ», 61002, Харків, вул. Кирпичова, 2.