**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

Лабораторна робота №5  
*(вид роботи: лабораторна робота, індивідуальне завдання, курсова робота тощо)*

з дисципліни «Алгоритмізація та програмування»Тема роботи: «Функції мови С/С++»  
**Варіант № 4**

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

Перевірив(-ла):  
ПІП викладача

Київ – 2021

**Завдання 1**

* 1. **Математична постановка задачі для функції sumRange(A, B).**

*Вхідні дані:* А, В – границі суми чисел, цілого типу.

*Вихідні дані:* sum – сума всіх чисел від А до В, цілого типу.

Якщо подати цю функцію, як «чорну скриньку», то маємо:

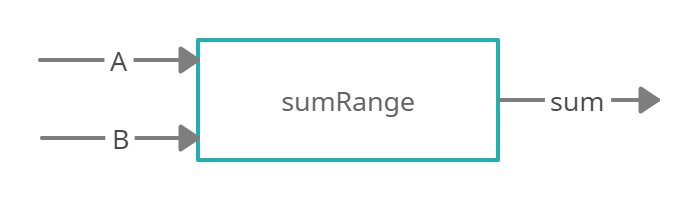


Рисунок 1.1 – Чорна скринька для функції sumRange.

*Математична модель задачі:*

Якщо А > B то функція повертає 0;

Інакше sum = sum + sumRange(A+1, B);

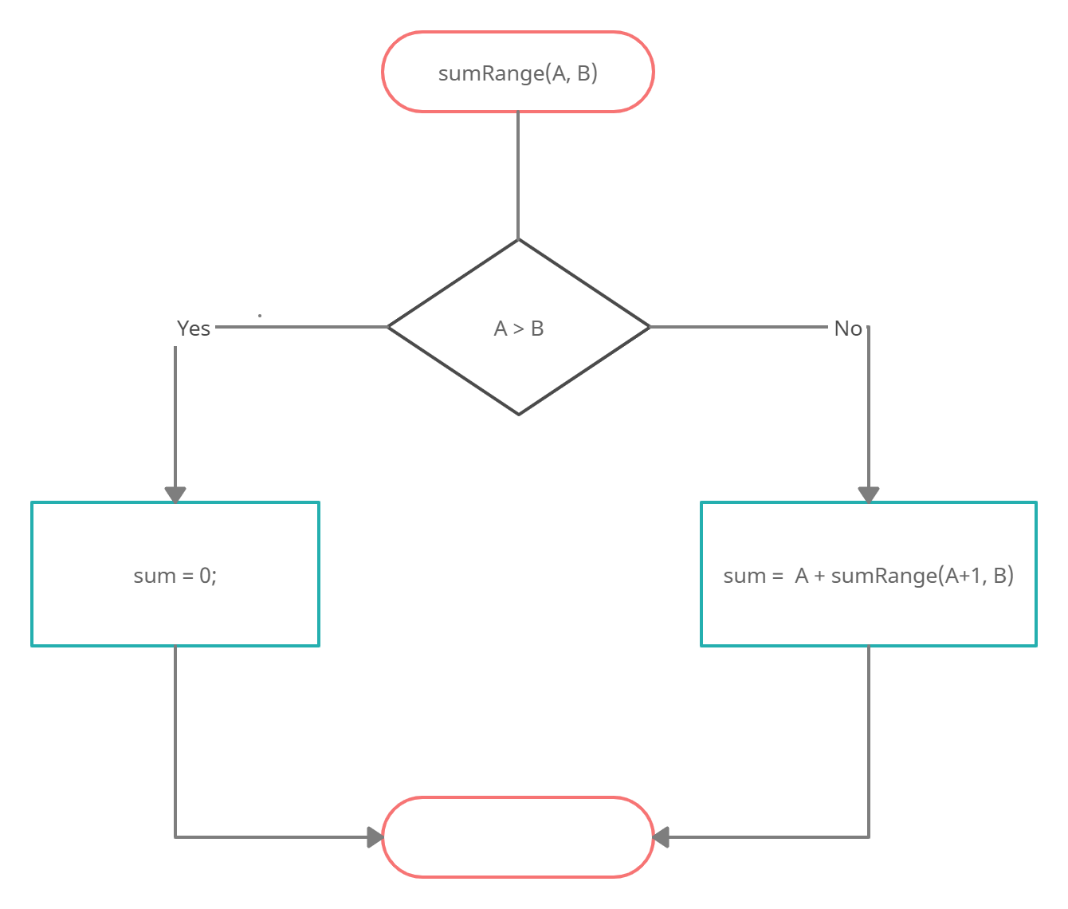


Рисунок 1.2 – Схема роботи функції sumRange().

* 1. **Математична постановка задачі для головної програми.**

*Вхідні дані:* firstPoint, secondPoint, thirdPoint – границі сум чисел, цілого типу.

*Вихідні дані:* sum – сума чисел для перших, а потім для других границь, цілого типу

Якщо подати головну програму, як «чорну скриньку», маємо:

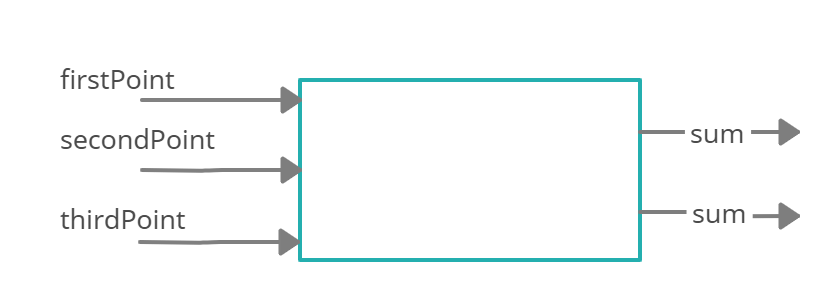


Рисунок 1.3 – Чорна скринька для основної програми.

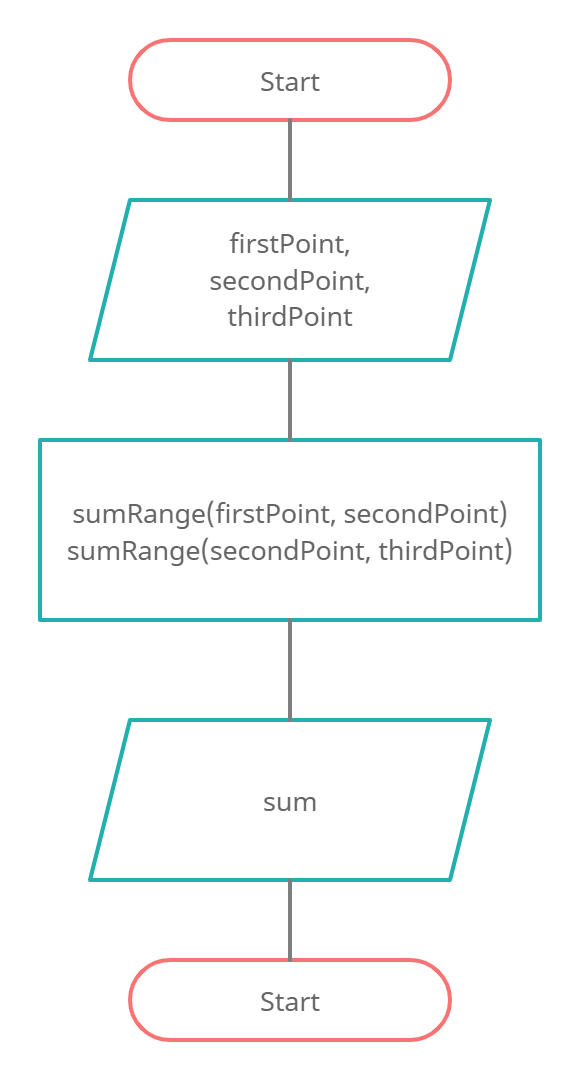


Рисунок 1.4 – Схема роботи основної програми.

* 1. **Тестовий приклад роботи програми.**

1. firstPoint = 1, secondPoint = 3, thirdPoint = 6;
2. sum = 1 + 2 + 3 = 6;
3. sum = 3 + 4 + 5 + 6 =18;

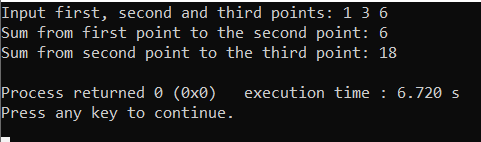


Рисунок 1.5 – Тестування програми.

1. firstPoint = 1, secondPoint = 5, thirdPoint = 7;
2. sum = 1 + 2 + 3 + 4 + 5 = 15;
3. sum = 5 + 6 + 7 = 18;

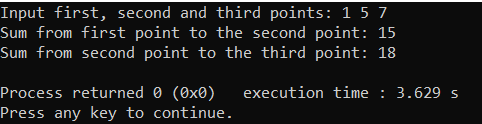


Рисунок 1.6 – Тестування програми.

1. firstPoint = 2, secondPoint = 3, thirdPoint = 4;
2. sum = 2 + 3 = 5;
3. sum = 3 + 4 = 7;

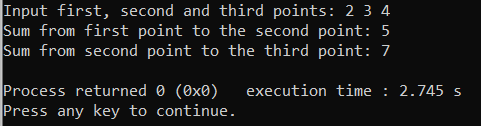


Рисунок 1.7 – Тестування програми.

*Висновок:* результати роботи програми підтверджено тестовими розрахунками.

* 1. **Текст програмною мовою С++.**

#**include** <iostream>

#**include** <cmath>

using namespace **std**;

**int** sumRange(**int** A, **int** B)

{

**int** sum = 0;

**if**(A > B) **return** 0;

**else** sum = A + sumRange(A+1, B);

**return** sum;

}

**int** main()

{

**int** firstPoint, secondPoint, thirdPoint;

cout << "Input first, second and third points: ";

cin >> firstPoint >> secondPoint >> thirdPoint;

cout << "Sum from first point to the second point: " << sumRange(firstPoint, secondPoint) << endl;

cout << "Sum from second point to the third point: " << sumRange(secondPoint, thirdPoint) << endl;

**return** 0;

}

**Завдання 2**

**2.1 Математична постановка задачі для функції isSquare(K).**

Вхідні дані: K – число цілого типу.

Вихідні дані: відповідь логічного типу.

Якщо подати цю функцію, як чорну скриньку, маємо:



Рисунок 2.1 – Чорна скринька для функції isSquare().

Математична модель задачі:

number = ;

Якщо number \* number = K, тоді повертати логічне значення «істина».

Інакше повертати логічне значення «хиба».

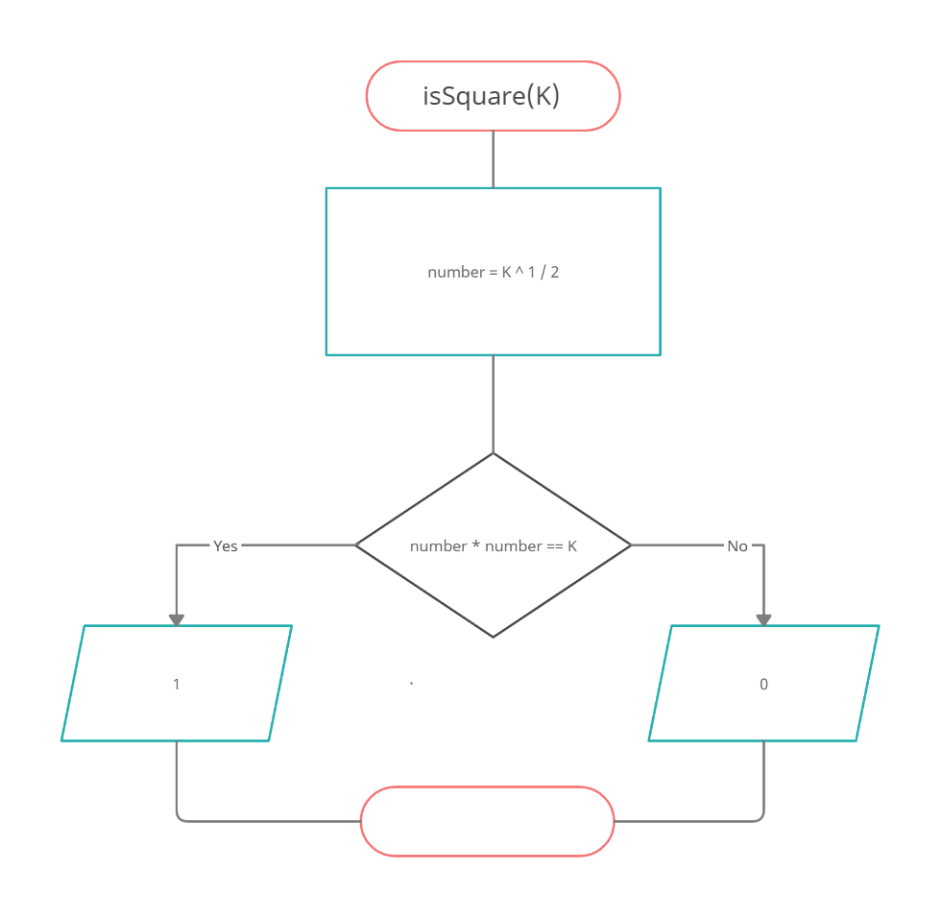


Рисунок 2.2 – Робота функції isSquare().

**2.2 Математична постановка задачі для основної програми.**

*Вхідні дані:* checkNumber – число цілого типу.

*Вихідні дані:* відповідь символьного типу.

Якщо представити головну програму, як «чорну скриньку», маємо:

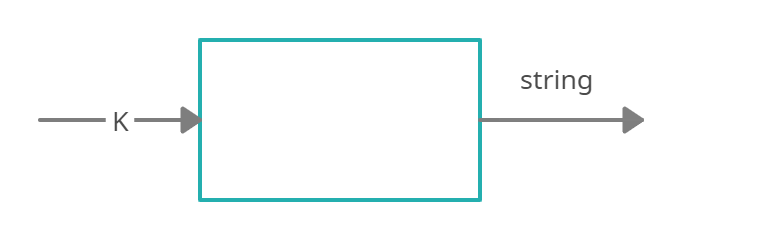


Рисунок 2.3 – Чорна скринька основної програми.

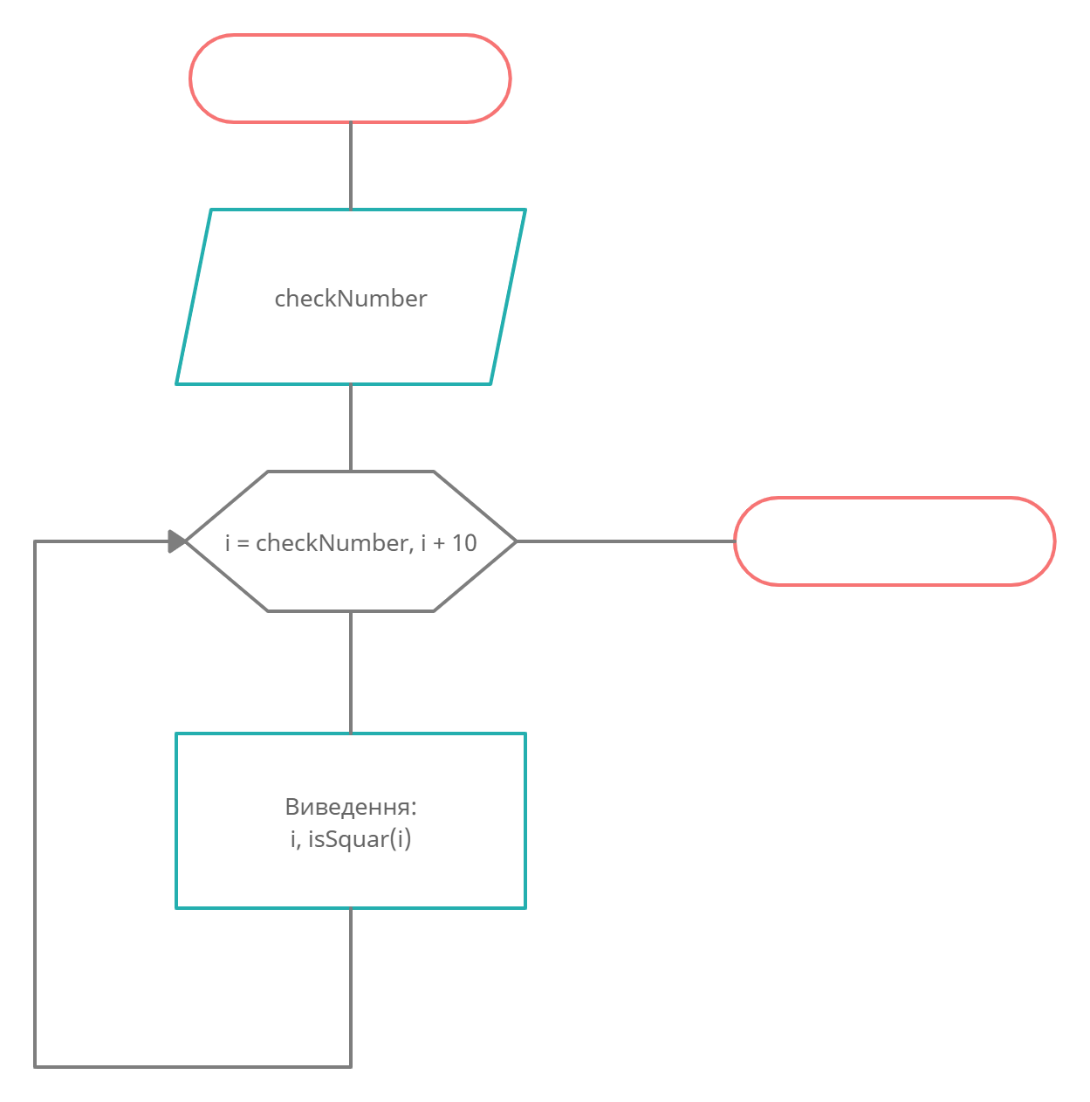


Рисунок 2.4 – Схема роботи основної програми.

**2.3 Тестовий приклад роботи програми.**

1. checkNumber = 1;

1 ітерація. 1 – 0;

2 ітерація. 2 – 0;

3 ітерація. 3 – 0;

4 ітерація. 4 – 1;

5 ітерація. 5 – 0;

6 ітерація. 6 – 0;

7 ітерація. 7 – 0;

8 ітерація. 8 – 0;

9 ітерація. 9 – 1;

10 ітерація. 10 – 0;

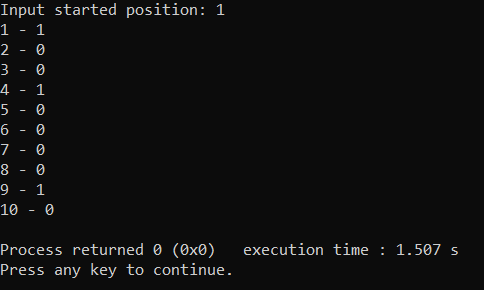


Рисунок 2.5 – Тестування програми.

2. checkNumber = 10;

1 ітерація. 10 – 0;

2 ітерація. 11 – 0;

3 ітерація. 12 – 0;

4 ітерація. 13 – 0;

5 ітерація. 14 – 0;

6 ітерація. 15 – 0;

7 ітерація. 16 – 1;

8 ітерація. 17 – 0;

9 ітерація. 18 – 0;

10 ітерація. 20 – 0;

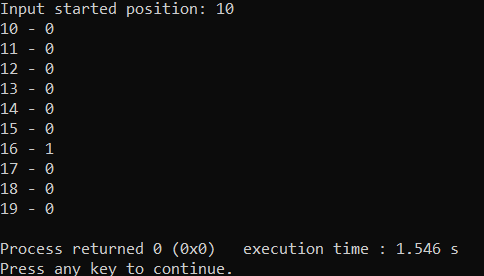


Рисунок 2.6 – Тестування програми.

3. checkNumber = -9;

1 ітерація. -9 – 0;

2 ітерація. -8 – 0;

3 ітерація. -7 – 0;

4 ітерація. -6 – 0;

5 ітерація. -5 – 0;

6 ітерація. -4 – 0;

7 ітерація. -3 – 0;

8 ітерація. -2 – 0;

9 ітерація. -1 – 0;

10 ітерація. 0 – 1;

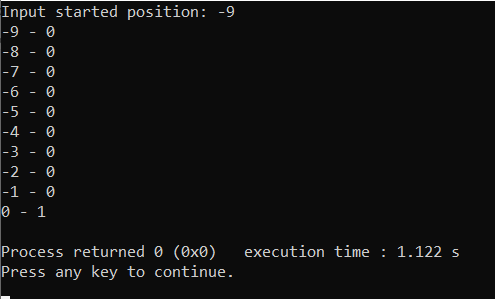


Рисунок 2.7 – Тестування програми.

*Висновок:* результати роботи програми підтверджено тестовими розрахунками.

**2.4 Тестування програми мовою С++.**

#**include** <iostream>

#**include** <cmath>

using namespace **std**;

**bool** isSquare(**int** K)

{

**int** number = sqrt(K);

**if**(number\*number == K) **return** **true**;

**else** **return** **false**;

}

**int** main()

{

**int** checkNumber;

cout << "Input started position: ";

cin >> checkNumber;

**for**(**int** i = checkNumber; i < checkNumber + 10; i++)

{

cout << i << " - " << isSquare(i) << endl;

}

**return** 0;

}

**Завдання 3**

**3.1 Математична постановка задачі для функції addLeftDigit(D, K).**

*Вхідні дані:*

D – число, яке треба дописати зліва (1, 9), цілого типу,

K – число цілого типу, до якого дописують числа зліва.

*Вихідні дані:*

K – змінене число цілого типу.

Якщо подати цю функцію, як «чорну скриньку», маємо:



Рисунок 3.1 – Чорна скринька для функції addLeftDigit().

*Математична модель задачі:*

temp = |K|, K = 0.

step = 1, temp > 0:

Якщо temp < 10, то K = K + {temp / 10} \* step + D \* step \* 10;

Інакше: К = К + {temp / 10} \* step;

temp = [temp / 10];

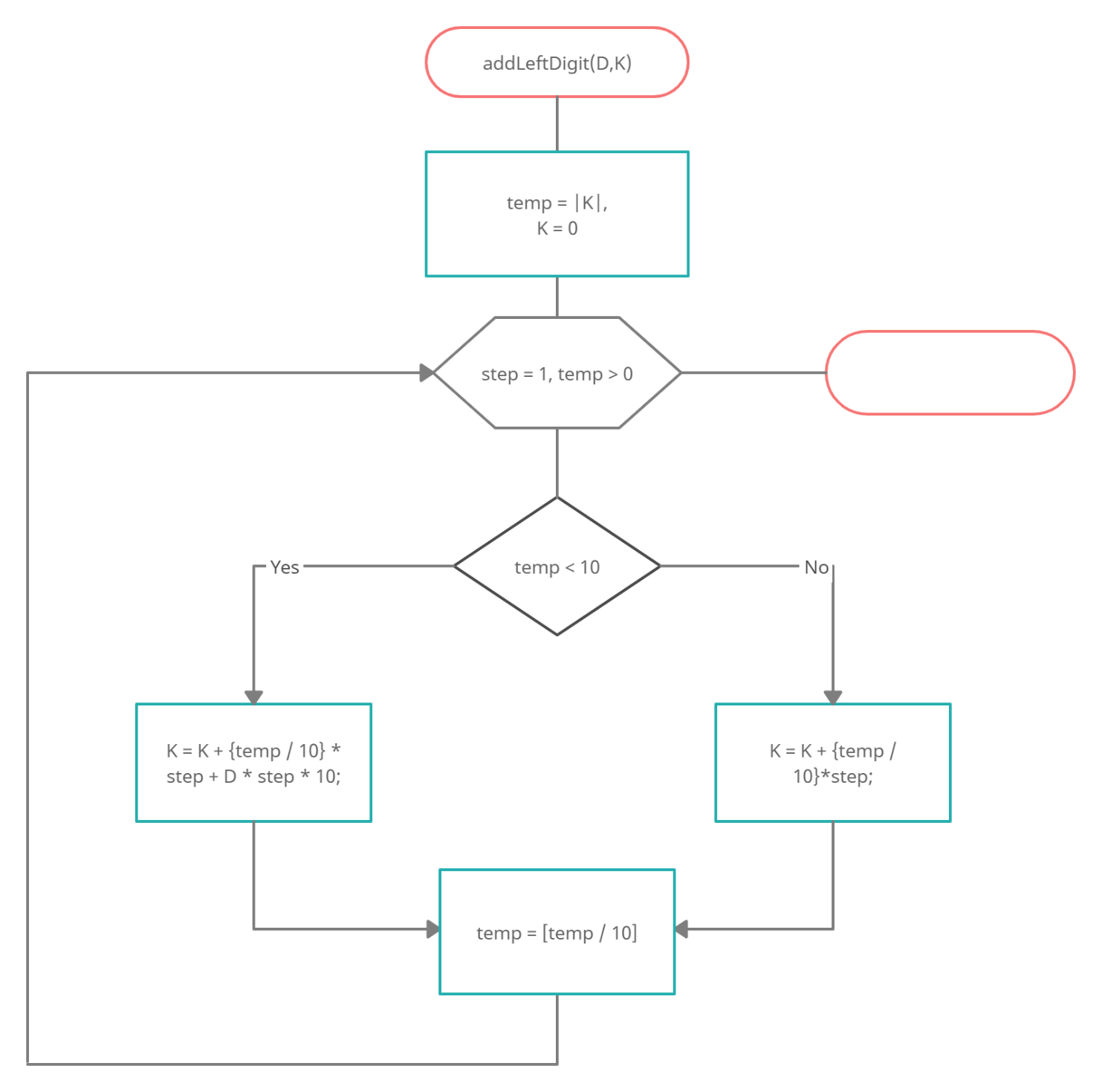


Рисунок 3.2 – Робота функції addLeftDigit().

Таблиця 3.1 – Варіант реалізації addLeftDigit.

|  |  |
| --- | --- |
| *Параметри-змінні як посилання* | *Параметри-змінні як покажчик* |
| **void** addLeftDigit(**int** D, **int** &K)  {  **int** temp = K;  K = 0;  **for**(**int** step = 1; temp > 0; step\*=10)  {  **if** (temp < 10) K += (temp%10)\*step + D\*step\*10;  **else** K += (temp%10)\*step;  temp /= 10;  }  } | **void** addLeftDigit(**int** D, **int** \*K)  {  **int** temp = \*K;  \*K = 0;  **for**(**int** step = 1; temp > 0; step\*=10)  {  **if** (temp < 10) \*K += (temp%10)\*step + D\*step\*10;  **else** \*K += (temp%10)\*step;  temp /= 10;  }  } |

* 1. **Математична постановка для основної програми.**

*Вхідні дані:*

D1, D2 – числа цілого типу, які додаються зліва.

K – число до якого додають, цілого типу.

*Вихідні дані:*

К – змінене число цілого типу.

Якщо представити головну програму, як «чорну скриньку», маємо:



Рисунок 3.3 – Чорна скринька основної програми.

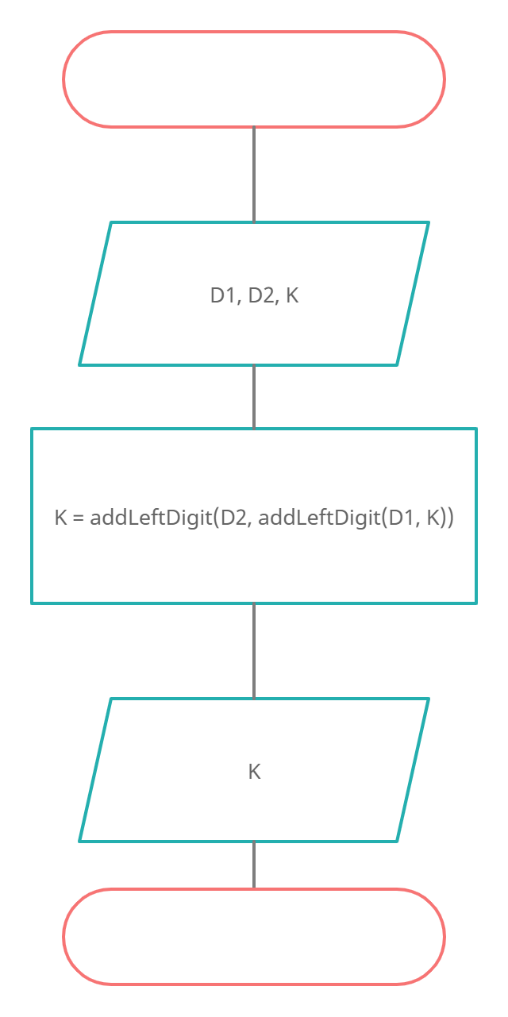


Рисунок 3.4 – Схема роботи основної програми.

* 1. **Тестовий приклад роботи програми.**

1. D1 = 2, D2 = 3, K = 900;

K = 32900.

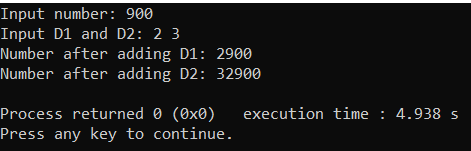


Рисунок 3.5 – Тестування програми.

1. D1 = 4, D2 = 9, K = 793;

K = 94793.

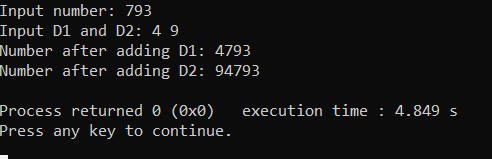


Рисунок 3.6 – Тестування програми.

1. D1 = 2, D2 = 8, K = 9;

K = 829.

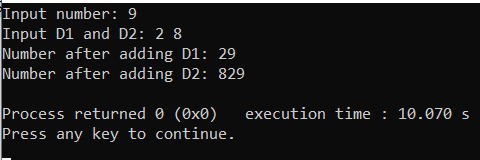


Рисунок 3.7 – Тестування програми.

* 1. **Текст програмною мовою С++.**

#**include** <iostream>

using namespace **std**;

**int** addLeftDigit(**int** D, **int** K)

{

**int** temp = K;

K = 0;

**for**(**int** step = 1; temp > 0; step\*=10)

{

**if** (temp < 10) K += (temp%10)\*step + D\*step\*10;

**else** K += (temp%10)\*step;

temp /= 10;

}

**return** K;

}

**int** main()

{

**int** D1, D2, K;

cout << "Input number: ";

cin >> K;

cout << "Input D1 and D2: ";

cin >> D1 >> D2;

cout << "Number after adding D1: " << addLeftDigit(D1, K) << endl;

cout << "Number after adding D2: " << addLeftDigit(D2, addLeftDigit(D1, K)) << endl;

**return** 0;

}

**Завдання 4**

**4.1 Математична постановка задачі для функції shiftLeft(A, B, C).**

*Вхідні дані:* A, B, C – числа, цілого типу.

*Вихідні дані:* нічого

Якщо подати цю функцію, як «чорну скриньку», маємо:



Рисунок 4.1 – Чорна скринька для функції shiftLeft.

*Математична модель задачі:*

temp = A;

A = B;

B = C;

C = temp.

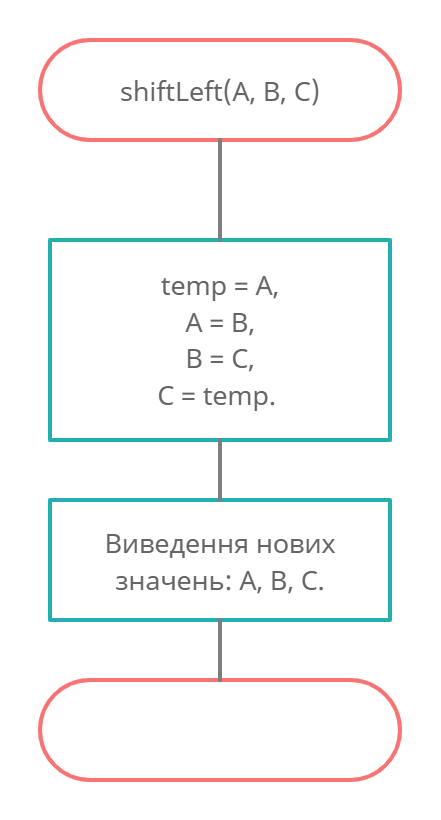


Рисунок 4.2 – Схема роботи функції shiftLeft().

Таблиця 4.1 – Варіант реалізації shiftLeft.

|  |  |
| --- | --- |
| *Параметри-змінні як посилання* | *Параметри-змінні як покажчик* |
| **void** shiftLeft(**int** &A, **int** &B, **int** &C)  {  **int** temp = 0;  temp = A;  A = B;  B = C;  C = temp;  } | **void** shiftLeft(**int** \*A, **int** \*B, **int** \*C)  {  **int** temp = 0;  temp = \*A;  \*A = \*B;  \*B = \*C;  \*C = temp;  } |

**4.2 Математична постановка задачі для основної програми.**

Вхідні дані:

firstNumGroupOne, SecondNumGroupOne, ThirdNumGroupOne – числа першої групи, цілого типу.

firstNumGroupTwo, SecondNumGroupTwo, ThirdNumGroupTwo – числа другої групи, цілого типу.

Вихідні дані:

firstNumGroupOne, SecondNumGroupOne, ThirdNumGroupOne – числа першої групи, цілого типу.

firstNumGroupTwo, SecondNumGroupTwo, ThirdNumGroupTwo – числа другої групи, цілого типу.

Якщо представити головну програму, як «чорну скриньку», маємо:



Рисунок 4.3 – Чорна скринька для основної програми.



Рисунок 4.4 – Схема роботи основної програми.

**4.3 Тестовий приклад програми.**

1. firstNumGroupOne = 1, SecondNumGroupOne = 2, ThirdNumGroupOne = 3,

firstNumGroupTwo = 4, SecondNumGroupTwo = 5, ThirdNumGroupTwo = 6.

firstNumGroupOne = 2 SecondNumGroupOne = 3, ThirdNumGroupOne = 1,

firstNumGroupTwo = 5, SecondNumGroupTwo = 6, ThirdNumGroupTwo = 4.

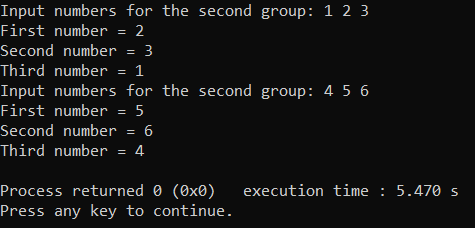


Рисунок 4.5 – Тестування програми.

2. firstNumGroupOne = 12, SecondNumGroupOne = 21, ThirdNumGroupOne = 9,

firstNumGroupTwo = 21, SecondNumGroupTwo = 9, ThirdNumGroupTwo = 12.

firstNumGroupOne = 55 SecondNumGroupOne = 23, ThirdNumGroupOne = 17,

firstNumGroupTwo = 23, SecondNumGroupTwo = 17, ThirdNumGroupTwo = 55.

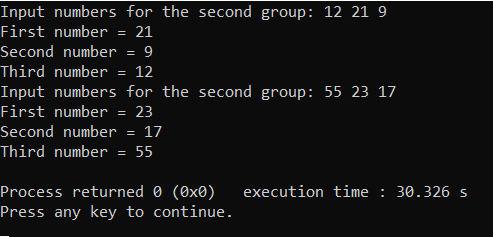


Рисунок 4.6 – Тестування програми.

2. firstNumGroupOne = 9, SecondNumGroupOne = 8, ThirdNumGroupOne = 7,

firstNumGroupTwo = 8, SecondNumGroupTwo = 7, ThirdNumGroupTwo = 9.

firstNumGroupOne = 11 SecondNumGroupOne = 34, ThirdNumGroupOne = 25,

firstNumGroupTwo = 34, SecondNumGroupTwo = 25, ThirdNumGroupTwo = 11.

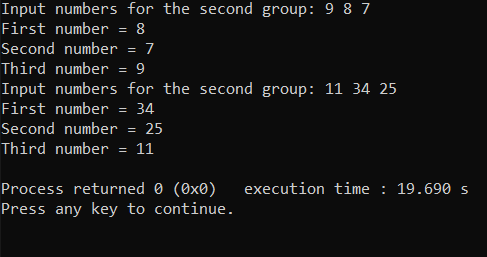


Рисунок 4.7 – Тестування програми.

**4.4 Текст мовою С++.**

#**include** <iostream>

using namespace **std**;

**void** shiftLeft(**int** A, **int** B, **int** C)

{

**int** temp = 0;

temp = A;

A = B;

B = C;

C = temp;

cout << "First number = " << A << endl << "Second number = " << B << endl << "Third number = " << C << endl;

}

int main()

{

**int** FirstNumGroupOne, SecondNumGroupOne, ThirdNumGroupOne;

**int** FirstNumGroupTwo, SecondNumGroupTwo, ThirdNumGroupTwo;

cout << "Input numbers for the second group: ";

cin >> FirstNumGroupOne >> SecondNumGroupOne >> ThirdNumGroupOne;

shiftLeft(FirstNumGroupOne, SecondNumGroupOne, ThirdNumGroupOne);

cout << "Input numbers for the second group: ";

cin >> FirstNumGroupTwo >> SecondNumGroupTwo >> ThirdNumGroupTwo;

shiftLeft(FirstNumGroupTwo, SecondNumGroupTwo, ThirdNumGroupTwo);

**return** 0;

}

**Завдання 5**

**5.1 Тестові приклади роботи функції sumSquare(A, B) з символьними параметрами.**

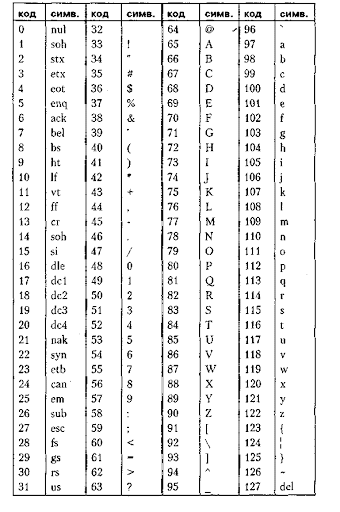
****

Рисунок 5.1 – ASCII.

1. А = ‘3’, В = ‘2’;

За таблицею знак «3» відповідає значенню 51, а знак «2» 50.

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 512 + 502 = 2500 + 2601 = 5101

Далі він шукає знак під таким номером та виводить його нам на екран.

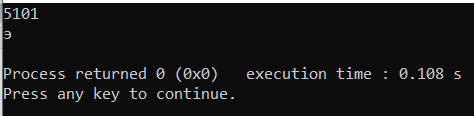


Рисунок 5.2 – Тестування програми.

2. А = ‘5’, B = ‘4’;

За таблицею знак «5» відповідає значенню 53, а знак «4» 52.

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 532 + 522 = 2809 + 2704 = 5513

Далі знаходимо кількість проходів по цій таблиці (ціле значення)

Кількість = 5513 / 256 = 21

Знаходимо номер нового символа:

5513 – 256 \* 21 = 137 - Й

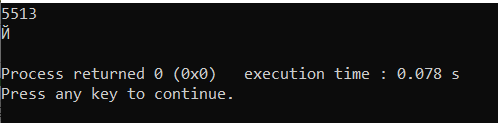


Рисунок 5.3 – Тестування програми.

3. А = ‘1’, B = ‘2’;

За таблицею знак «1» відповідає значенню 49 , а знак «2» 50.

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 492 + 502 = 2401 + 2500 = 4901

Далі знаходимо кількість проходів по цій таблиці (ціле значення)

Кількість = 4901 / 256 = 19

Знаходимо номер нового символа:

4901 – 256 \* 19 = 37 - %

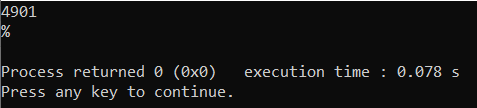


Рисунок 5.4 – Тестування програми.

**5.2 Текст мовою С++.**

**char** sumSquare(**char** A, **char** B)

{

**return** A\*A+B\*B;

}

Виклик функції:

cout << sumSquares(‘2’, '1') << endl;

**5.3 Тестові приклади роботи функції sumSquare(A, B) з цілочисельними параметрами.**

1. A = 243, B = 454;

sum = 2432 + 4542 = 59049 + 206116 = 256165

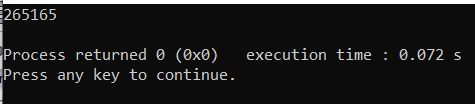


Рисунок 5.5 – Тестування програми.

2. A = 300, B = 400;

sum = 3002 + 4002 = 90000 + 160000 = 250000

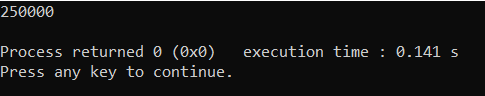


Рисунок 5.6 – Тестування програми.

3. A = 67, B = 5;

sum = 672 + 52 = 4489 + 25 = 4514

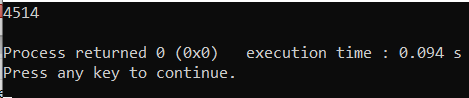


Рисунок 5.7 – Тестування програми.

**5.4 Текст програмною мовою С++.**

**long** sumSquare(**long** A, **long** B)

{

**return** A\*A + B\*B;

}

Виклик функції:

cout << sumSquares(67, 5) << endl;

**5.5** **Тестові приклади роботи функції sumSquare(A, B) з дійсними параметрами.**

1. А = 2.5 В = 4.2

sum = 2.52 + 4.22 = 6.25 + 17.64 = 23.89

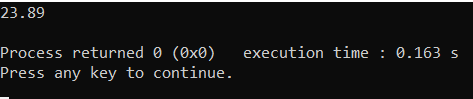


Рисунок 5.8 – Тестування програми.

2. А = 0.3 В = 1.2

sum = 0.32 + 1.22 = 0.09 + 1.44 = 1.53

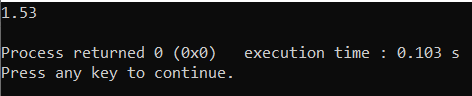


Рисунок 5.9 – Тестування програми.

2. А = 0.08 В = 12.5

sum = 0.082 + 12.52 = 0.0064 + 156.26 = 156.2564

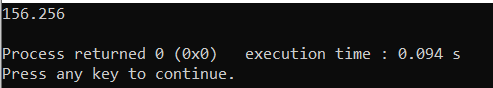


Рисунок 5.10 – Тестування програми

**5.6 Текст програмною мовою С++.**

**double** sumSquare(**double** A, **double** B)

{

return A\*A + B\*B;

}

Виклик функції:

cout << sumSquares(0.08, 12.5) << endl;

**5.7 Тестовий приклад для 3 перевантажених функцій зазначених попередньо.**

1. Тестовий приклад функції з символьними параметрами:

A = ‘1’, B = 1

За таблицею знак «1» відповідає значенню 49

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 492 + 12 = 2401 + 1 = 2402

Далі знаходимо кількість проходів по цій таблиці (ціле значення)

Кількість = 2402 / 256 = 9

Знаходимо номер нового символа:

2402 – 256 \* 21 = 98 - b

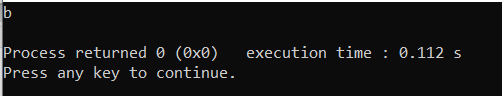


Рисунок 5.11 – Тестовий приклад програми.

1. Тестовий приклад з цілочисельними параметрами.

А = 23, В = 4.5

sum = 4.52 + 232 = 20.25 + 529 = 549.25

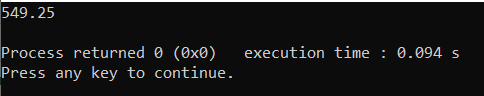


Рисунок 5.12 – Тестовий приклад програми.

1. Тестовий приклад з дійсними параметрами.

А = ‘0’, В = 4.5

За таблицею знак «0» відповідає значенню 48

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 482 + 4.52 = 2304 + 20.25 = 2324.25

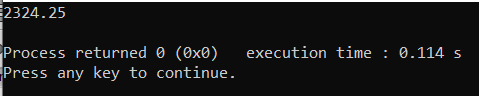


Рисунок 5.13 – Тестовий приклад програми.

**5.8 Текст програмною мовою С++.**

**char** sumSquare(**char** A, **int** B)

{

**return** A\*A+B\*B;

}

**long** sumSquare(**double** A, **long** B)

{

**return** A\*A + B\*B;

}

**double** sumSquare(**double** A, **char** B)

{

**return** A\*A + B\*B;

}

Виклик функцій:

cout << sumSquares(1, '1') << endl;

cout << sumSquares(4.5, 23) << endl;

cout << sumSquares(4.5, '0') << endl;

**5.9 Тестовий приклад перевантаженої функції з одним заданим параметрам за замовченням.**

1. Параметр B – не задається

A = 3;

sum = 9 + 81 = 90

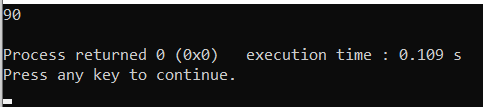


Рисунок 5.14 – Тестування програми 2.

Параметр B – задається

A = 3, В = 4;

sum = 9 + 16 = 25;

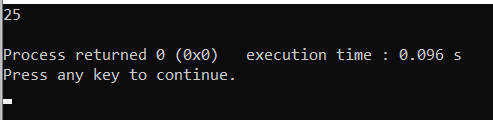


Рисунок 5.15 – Тестування програми.

3. Параметр B – задається

A = 8, В = 9;

sum = 64 + 81 = 145;

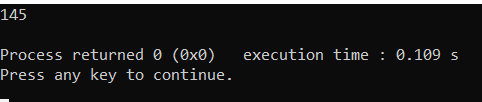


Рисунок 5.16 – Тестування програми.

**5.10 Текст програмною мовою С++.**

**int** sumSquare(**long** A, **int** B = 9)

{

**return** A\*A + B\*B;

}

Виклик функції:

cout << sumSquare(8) << endl;

**5.11 Тестові приклади функції шаблону.**

1. A = ‘1’, B = 4

За таблицею знак «1» відповідає значенню 49

Ми підставляємо ці значення та рахуємо їх суму квадратів

sum = 492 + 42 = 2401 + 16 = 2417

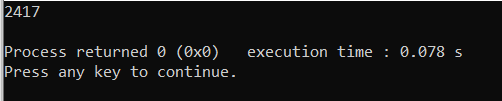


Рисунок 5.17 – Тестування програми.

2. A = 1.5, B = 2

sum = 1.52 + 22 = 2.25 + 4 = 6.25

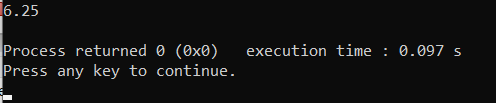


Рисунок 5.18 – Тестування програми.

3. A = 1, B = 2

sum = 12 + 22 = 1 + 4 = 5

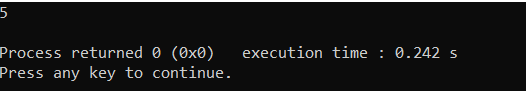


Рисунок 5.19 – Тестування програми.

**5.12 Текст програмною мовою С++.**

**template**<**typename** T1, **typename** T2>

**T2** sumSquares(**T1** A, **T2** B)

{

**return** A \* A + B \* B;

}

Виклик функції:

cout << sumSquares(2, 1) << endl;

**5.13 Текст програмною мовою С++.**

#**include** <iostream>

using namespace **std**;

**template**<**typename** T1, **typename** T2>

**T1** sumSquares(**T1** A, **T2** B)

{

**return** A \* A + B \* B;

}

/\*\*

**char** sumSquare(**int** A, **int** B)

{

**return** A\*A+B\*B;

}

**long** sumSquare(**double** A, **long** B)

{

**return** A\*A + B\*B;

}

**double** sumSquare(double A, char B)

{

**return** A\*A + B\*B;

}

**int** sumSquare(long A, int B = 9)

{

**return** A\*A + B\*B;

}

\*\*/

**int** main()

{

cout << sumSquares(1, '1') << endl;

cout << sumSquares(2, 1.5) << endl;

cout << sumSquares(2, 'A') << endl;

cout << sumSquares(8) << endl;

cout << sumSquares(2, 4) << endl;

**return** 0;

}