Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №6 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«Дослідження рекурсивних алгоритмів»

Варіант 33

Виконав студент ІП-12 Чікінеєва Дар’я Олегівна

(шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота 6**

**Дослідження рекурсивних алгоритмів**

**Мета** – дослідити особливості роботи рекурсивних алгоритмів та набути практичних навичок їх використання під час складання програмних специфікацій підпрограм.

**Варіант 33**

**Задача:** Перетворити натуральне число з десяткової системи числення у шістнадцятирічне значення.

1. Постановка задачі: знайти шістнадцятирічне значення заданого числа за допомогою алгоритму покрокового поділу на 16 та запису остачі від кожного поділу у зворотньому порядку, для виконання котрого й буде використовуватися рекурсія.
2. Побудова математичної моделі

| змінна | тип | ім’я | призначення |
| --- | --- | --- | --- |
| натуральне число х | натуральне число | х | початкові дані |
| залишок від ділення числа n на 16 | натуральне число | num | допоміжна змінна |
| обчислення шістнадцятирічного значення числа | функція | six\_rep(х) | допоміжна функція/  результат |
| обчислення значення числа у 16-й системі, якщо воно належить проміжку [10,16) | функція | six\_let(x) | Допоміжна функція |
| виведення значення числа у 16-й системі, для кейсів, коли воно співпадає зі значенням у 10-ій | функція | just\_x(x) | Допоміжна функція |
| масив значень A-F | масив | let[] | масив літер |
| індекс відповідника числа х у масиві let[] у функції six\_let(x) | натуральне число | k | допоміжна змінна |

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

*Крок 1.* Визначимо основні дії.

*Крок 2.* Деталізуємо дію перевірки введеного числа на невід’ємність

*Крок 3.* Деталізуємо перевірку на співпадіння значень х у 10-ій та 16-ій системах

*Крок 4.* Деталізуємо перевірку на буквене значення х у 16-ій системі

*Крок 5.* Деталізуємо функцію just\_x(x)-виведення числа, якщо його значення у 10-ій та 16-ій системах співпадають

*Крок 6.* Деталізуємо функцію six\_rep(x)-знаходження значення числа у шістнадцятирічній системі

*Крок 7.* Деталізуємо функцію six\_let(x)-знаходження символьного значення числа

1. Псевдокод алгоритму

***Крок 1***

початок

введення числа х

перевірка х на невід’ємність

перевірка на співпадіння значень х у 10-ій та 16-ій системах

перевірка на буквене значення х у 16-ій системі

знаходження значення числа у шістнадцятирічній системі

виведення знайденого числа

кінець

***Крок 2***

початок

введення числа х

*якщо* х<0

виведення “x cannot be used”

*інакше*:

перевірка на співпадіння значень х у 10-ій та 16-ій системах

перевірка на буквене значення х у 16-ій системі

знаходження значення числа у шістнадцятирічній системі

виведення знайденого числа

*все якщо*

кінець

***Крок 3***

початок

введення числа х

*якщо* х<0

виведення “x cannot be used”

*інакше якщо* х>=0 && x<=9:

just\_x(x)

перевірка на буквене значення х у 16-ій системі

знаходження значення числа у шістнадцятирічній системі

виведення знайденого числа

все якщо

кінець

***Крок 4***

початок

введення числа х

*якщо* х<0

виведення “x cannot be used”

*інакше якщо* х>=0 && x<=9:

just\_x(x)

*інакше якщо* x>9 && x<=15:

six\_let(x)

*інакше:*  six\_rep(x)

все якщо

кінець

***Крок 5 (псевдокод функції just\_x(x))***

виведення: x in hex system is x

повернути 0

кінець just\_x

***Крок 6 (псевдокод функції six\_rep(x))***

початок six\_rep(х)

num=x%16

x=x/16

*якщо* (х>0)

six\_rep(x)

*якщо* (num>=10 && num<16)

виведення six\_let(num)

*інакше*

виведення num

*все якщо*

повернути 0

кінець six\_rep

***Крок 7 (псевдокод функції six\_let(x))***

початок six\_let(x)

k=x-10

let[6]’={‘A’,’B’,’C’,’D’,’E’,’F’}

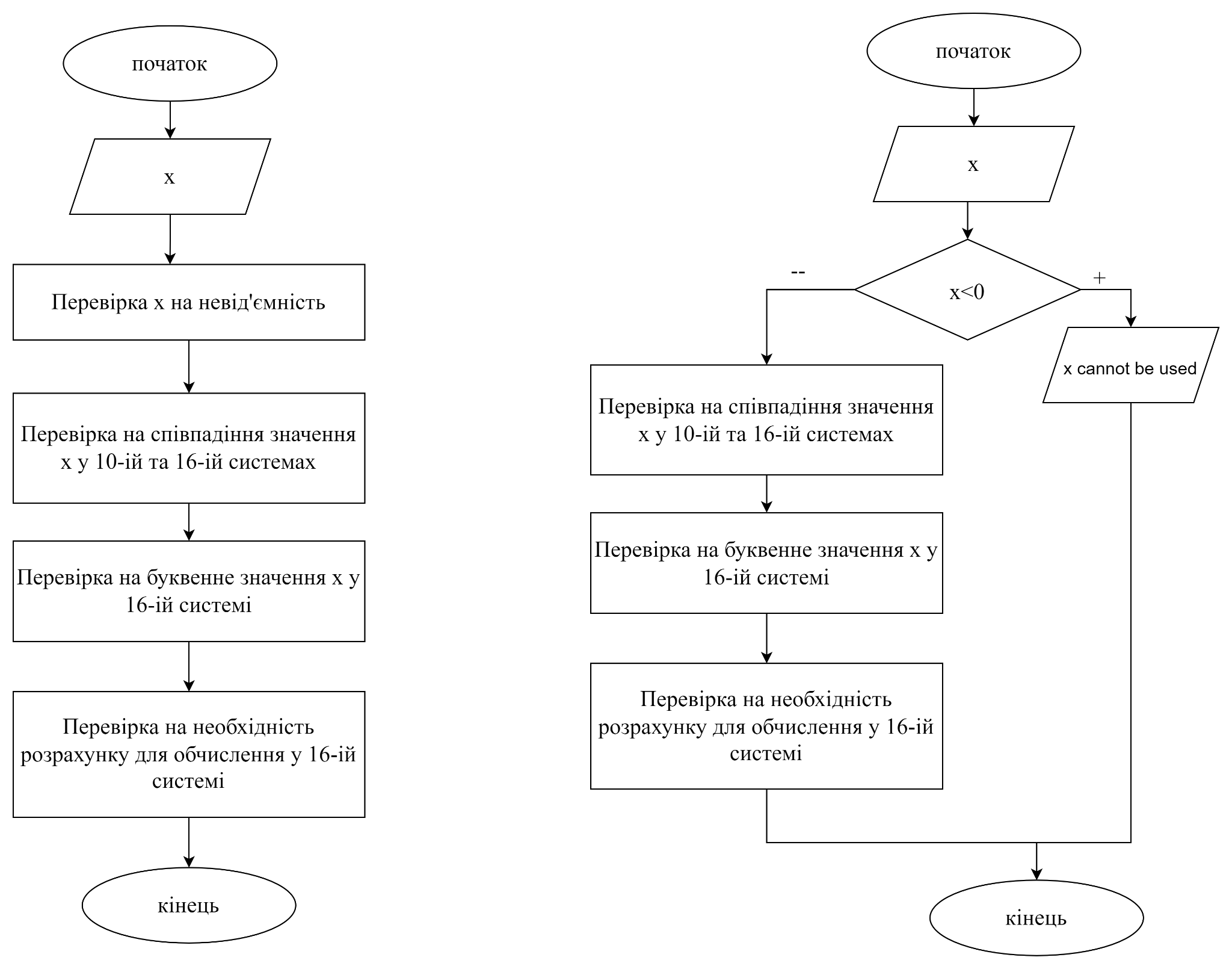
n = let[k]

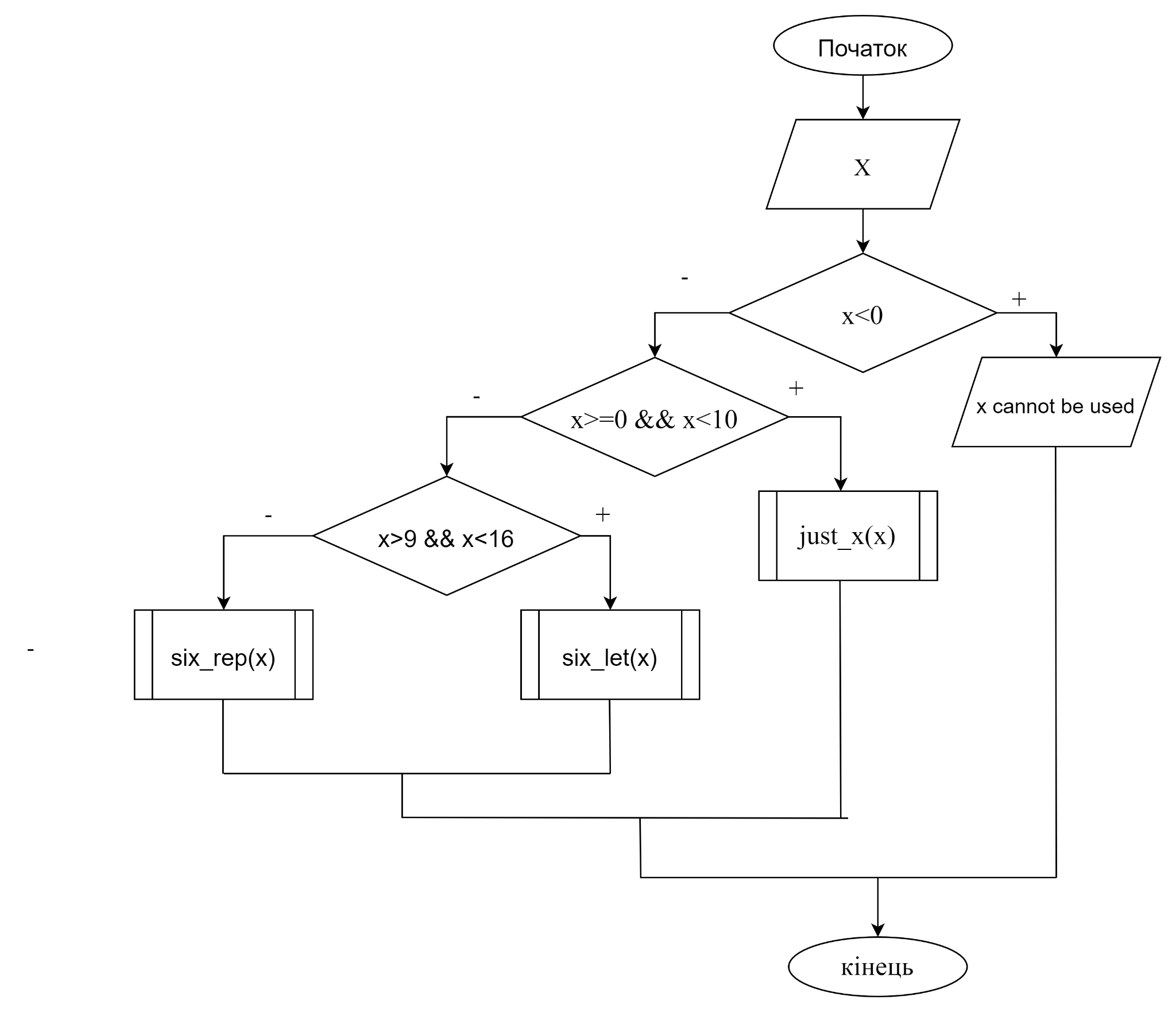
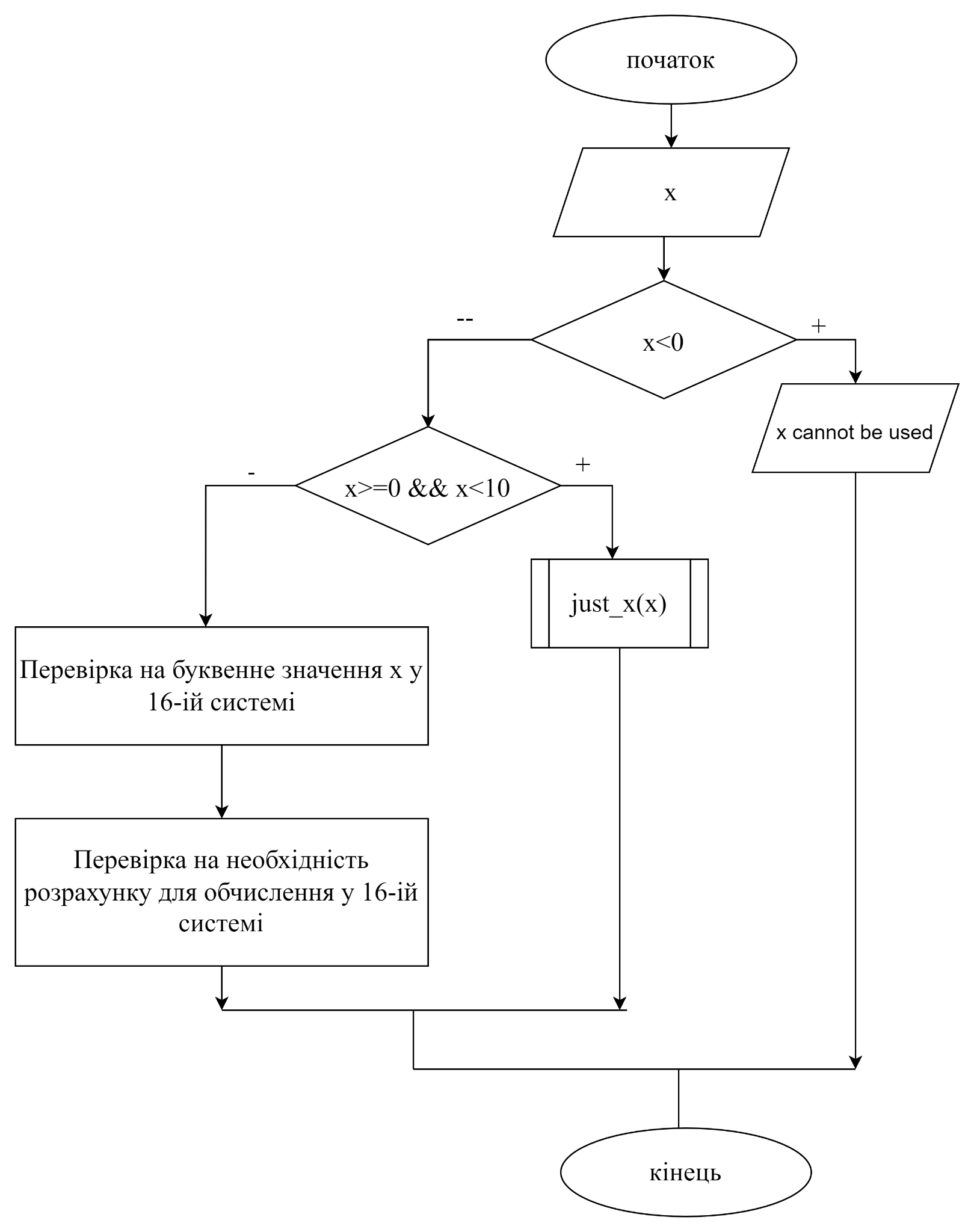
повернути n

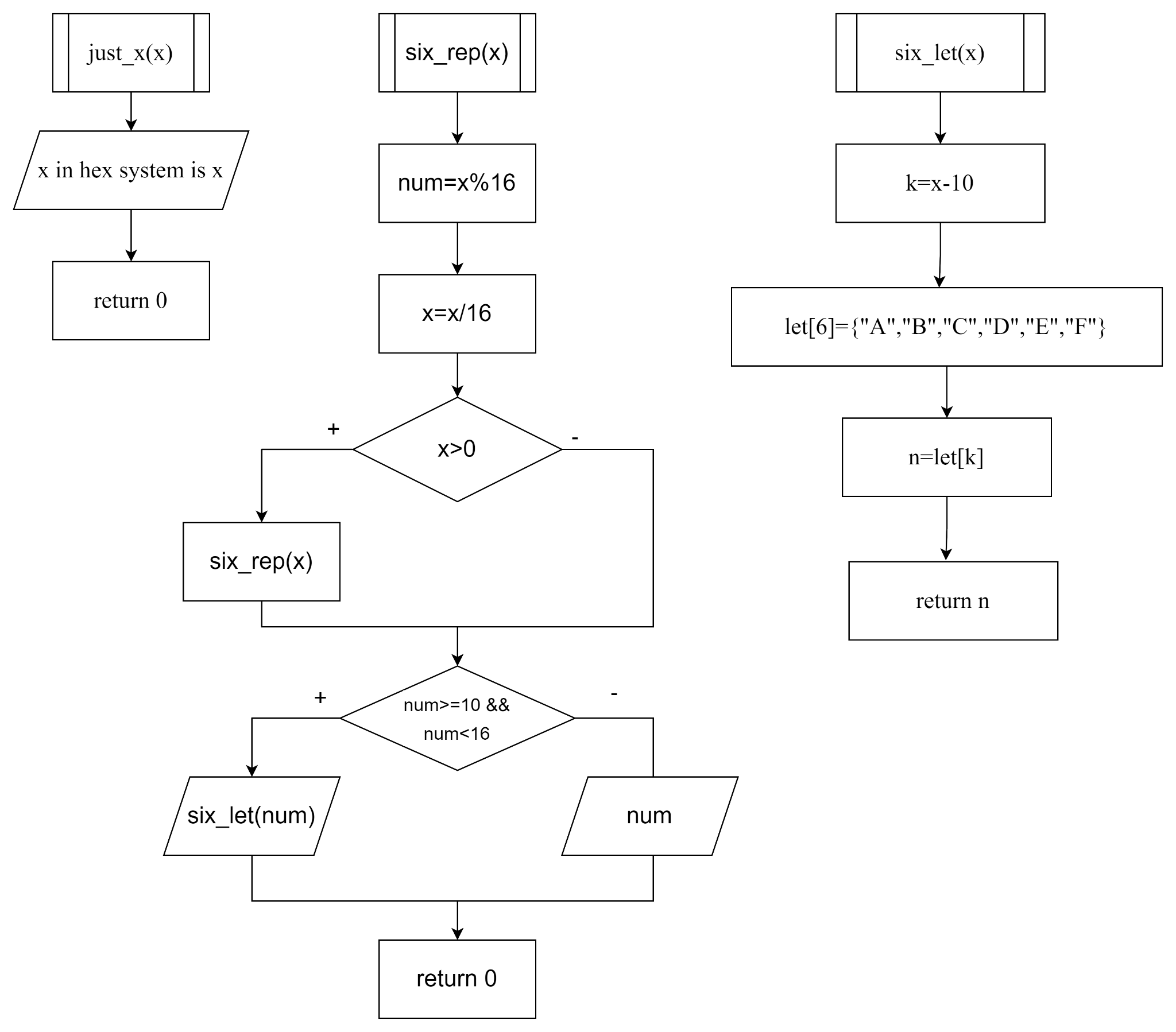
кінець six\_let

1. Блок-схема алгоритму

крок 1 крок 2







1. Код алгоритму

#include <iostream>

using namespace std;

int just\_x(int x) {

cout << x << " in hex system is " << x << endl;

return 0;

}

char six\_let(int x) {

int k = x - 10;

char let[6] = { 'A','B','C','D','E','F' };

char n = let[k];

return n;

}

int six\_rep(int x) {

int num;

num = x % 16;

x = x / 16;

if (x > 0) {

six\_rep(x);

}

if (num >= 10 && num < 16) {

cout << six\_let(num);

}

if (num < 10) {

cout << num;

}

return 0;

}

int main() {

int x;

cout << "enter x" << endl;

cin >> x;

if (x < 0) {

cout << "x cannot be used";

}

else if (x >= 0 && x <= 9) {

cout << x << " in hex system is " << x << endl;

}

else if (x > 9 && x <= 15) {

cout<<x<< " in hex system is " << six\_let(x);

}

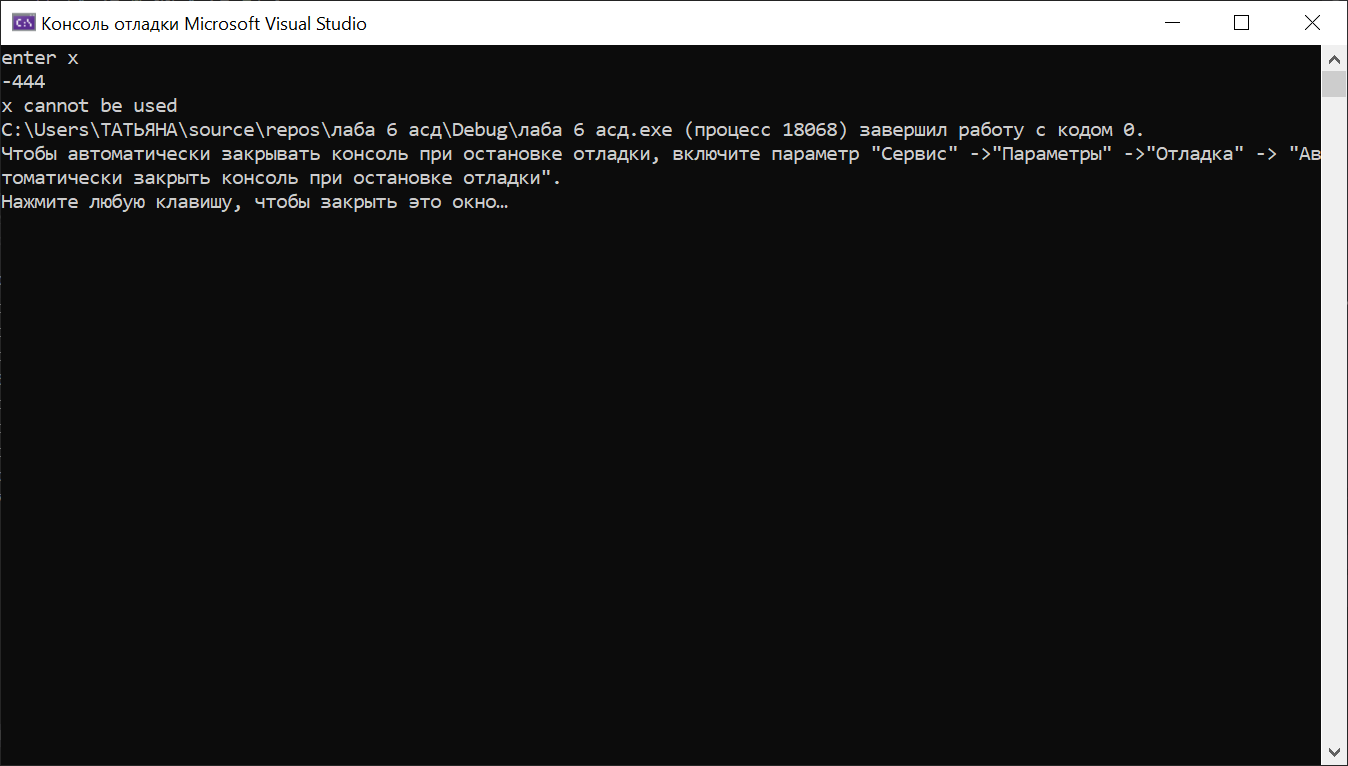
else

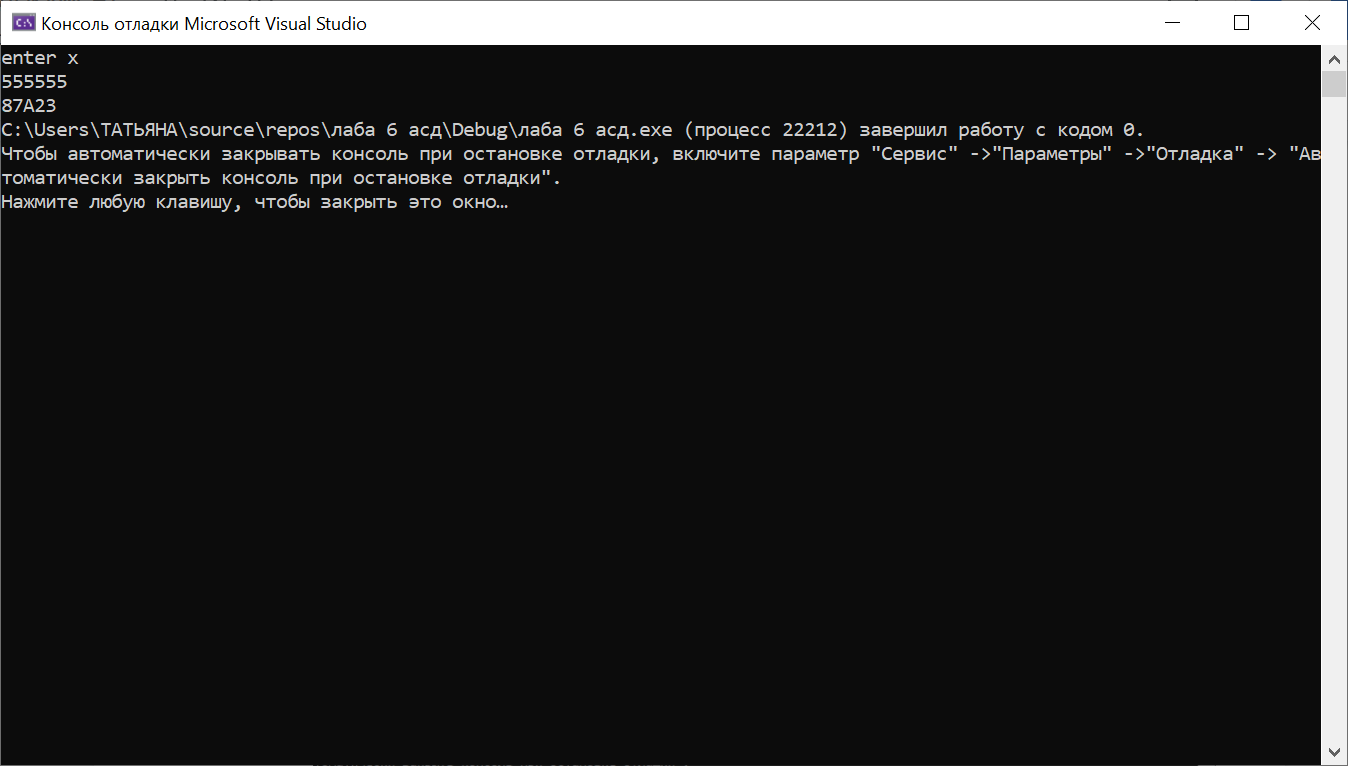
six\_rep(x);

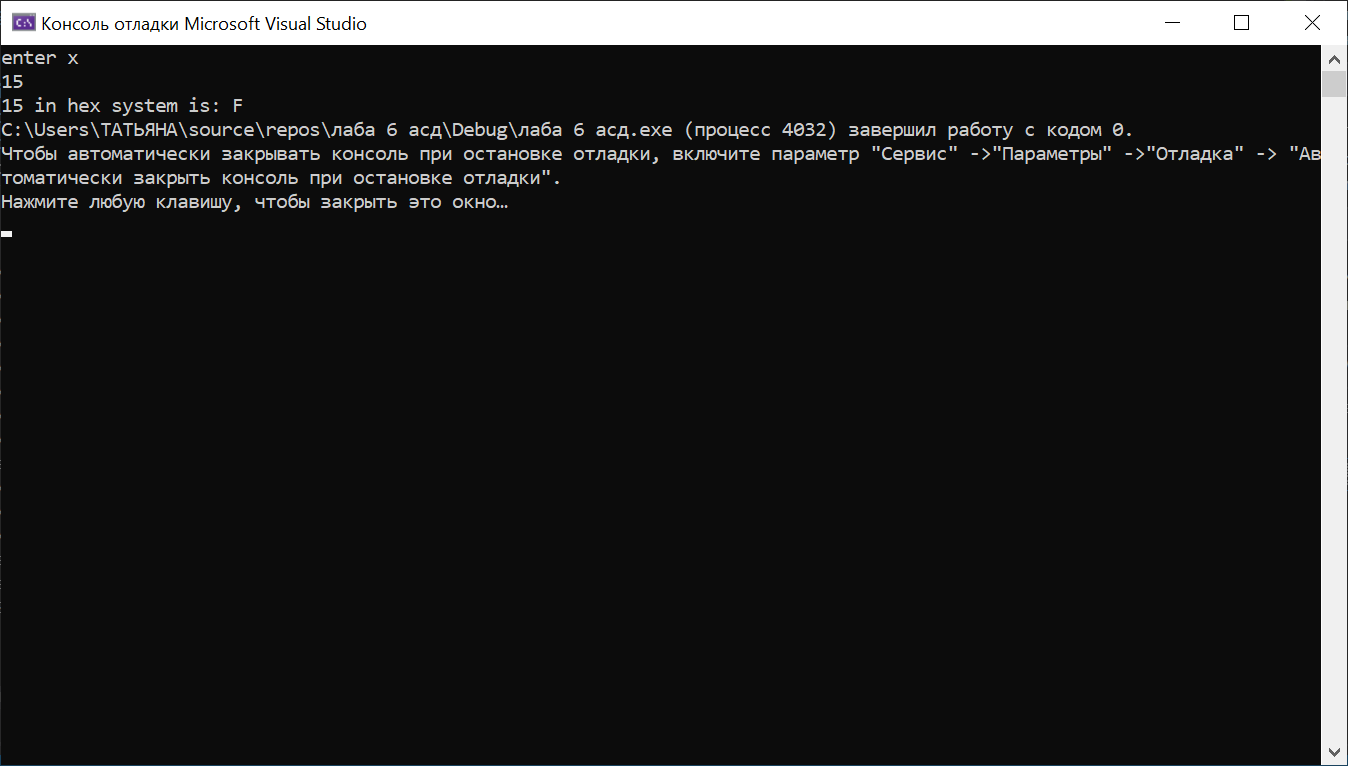
return 0;

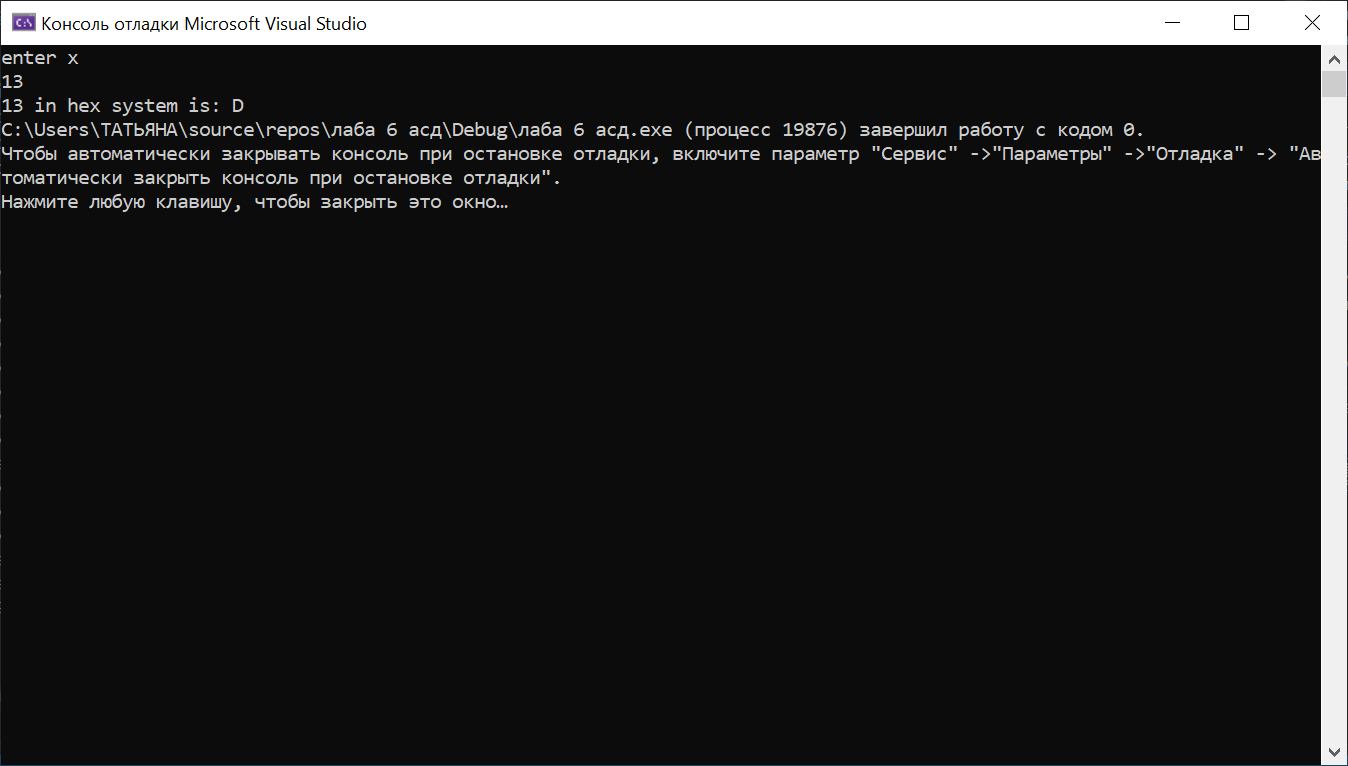
}

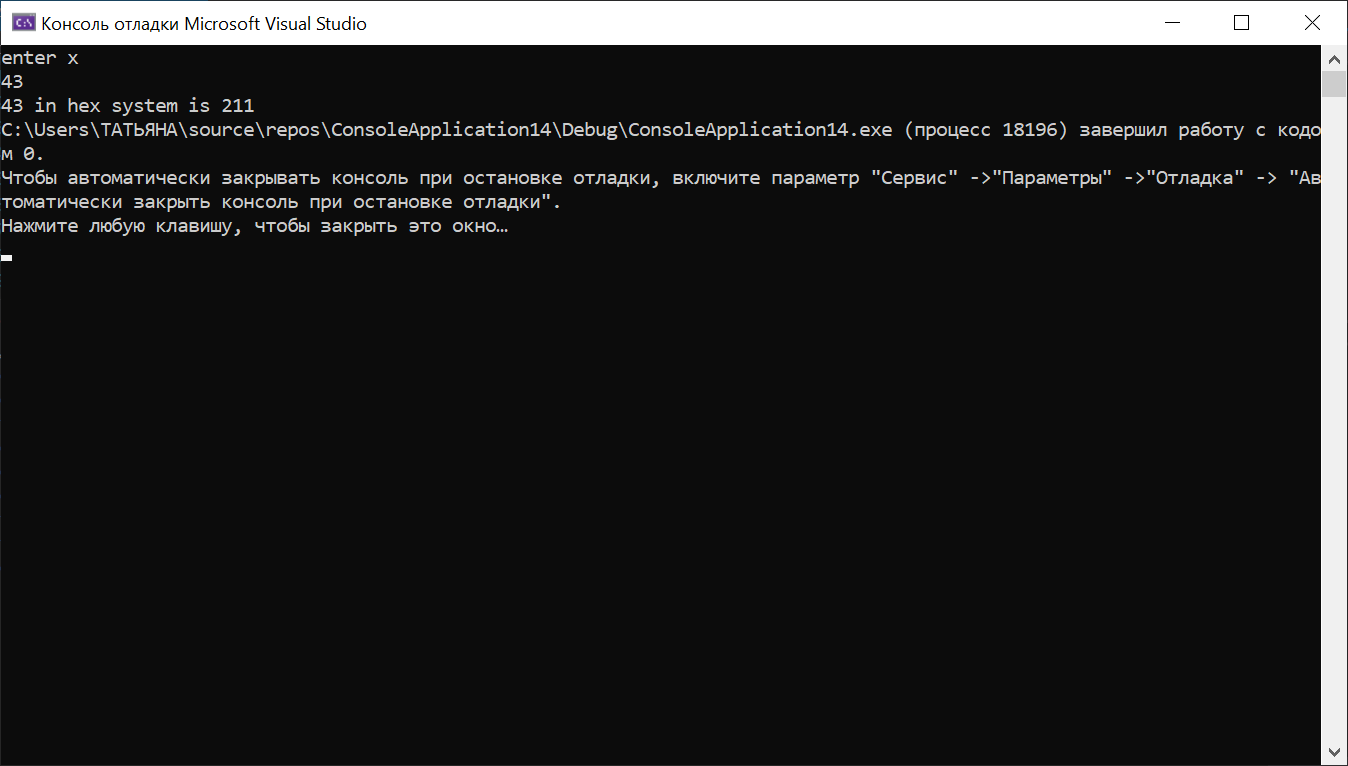
1. Тестування програми

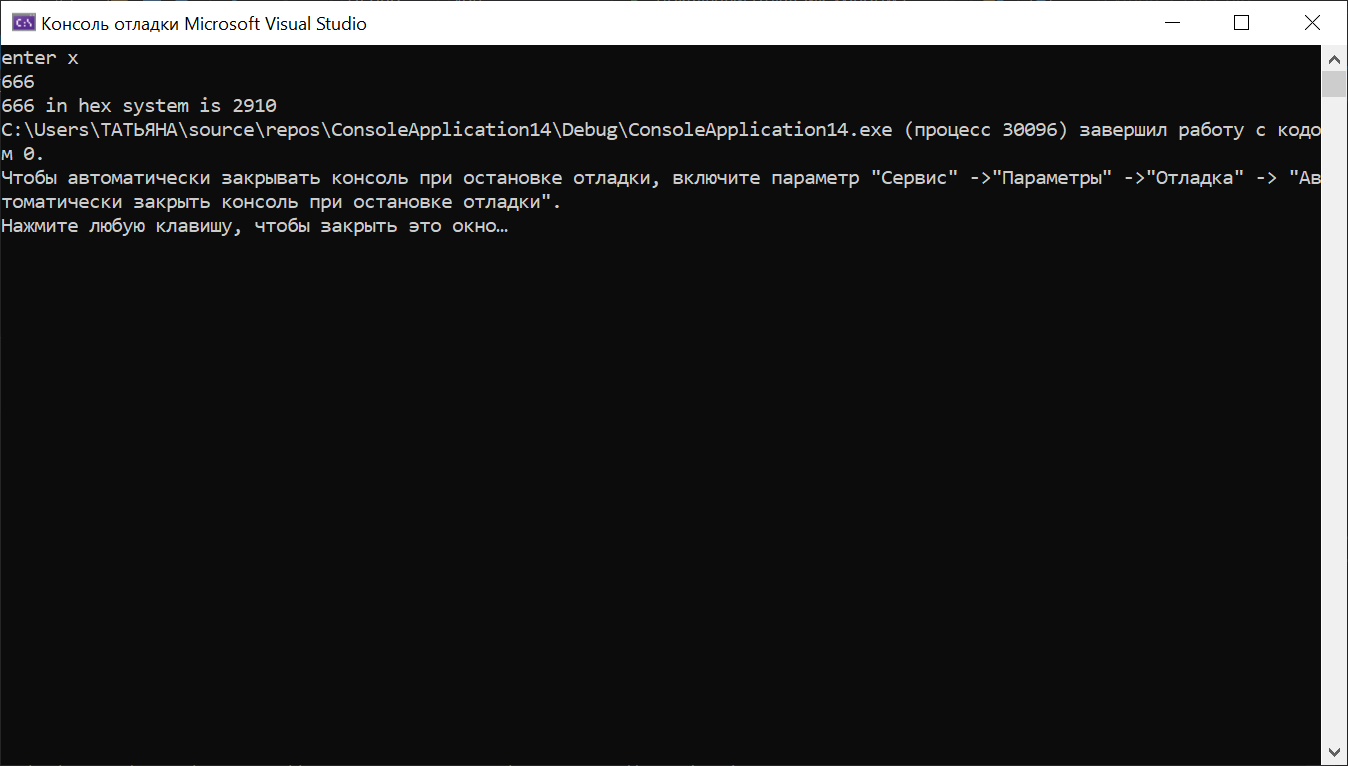


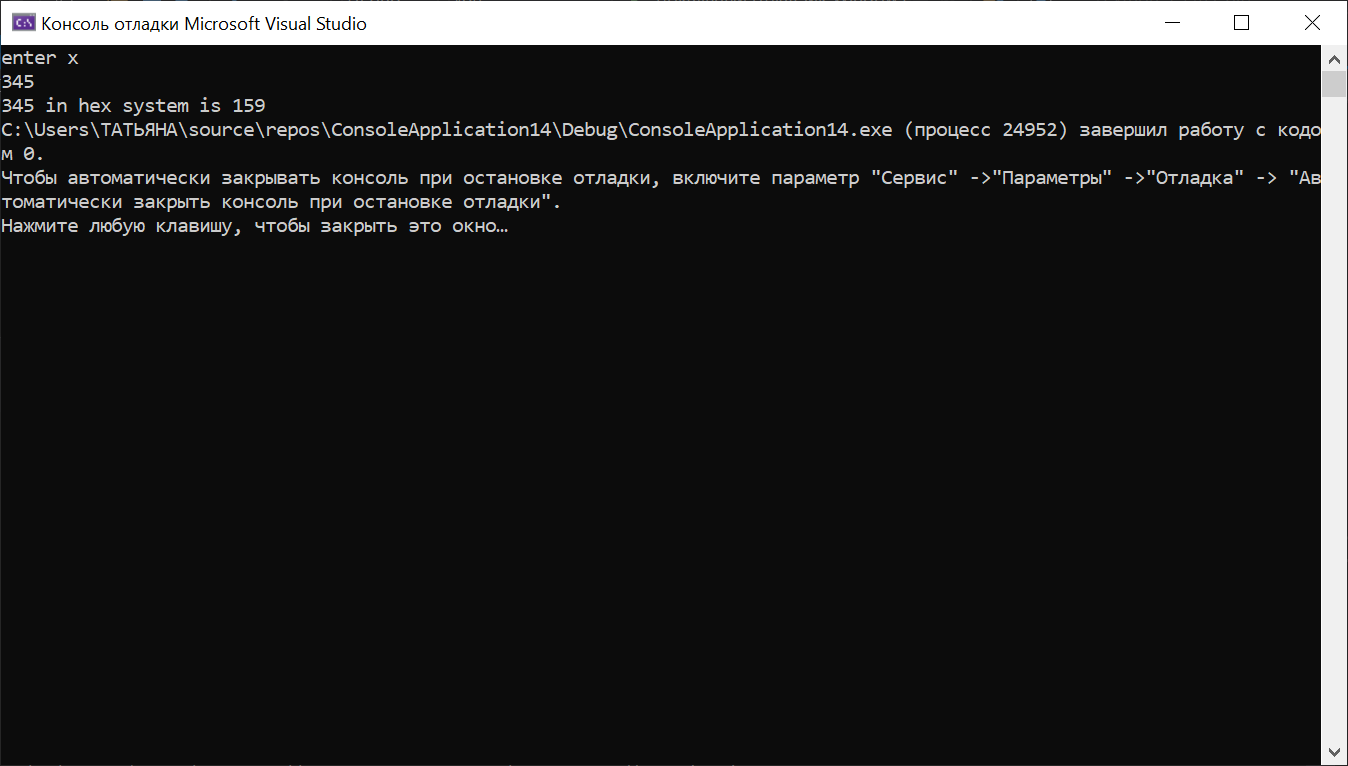


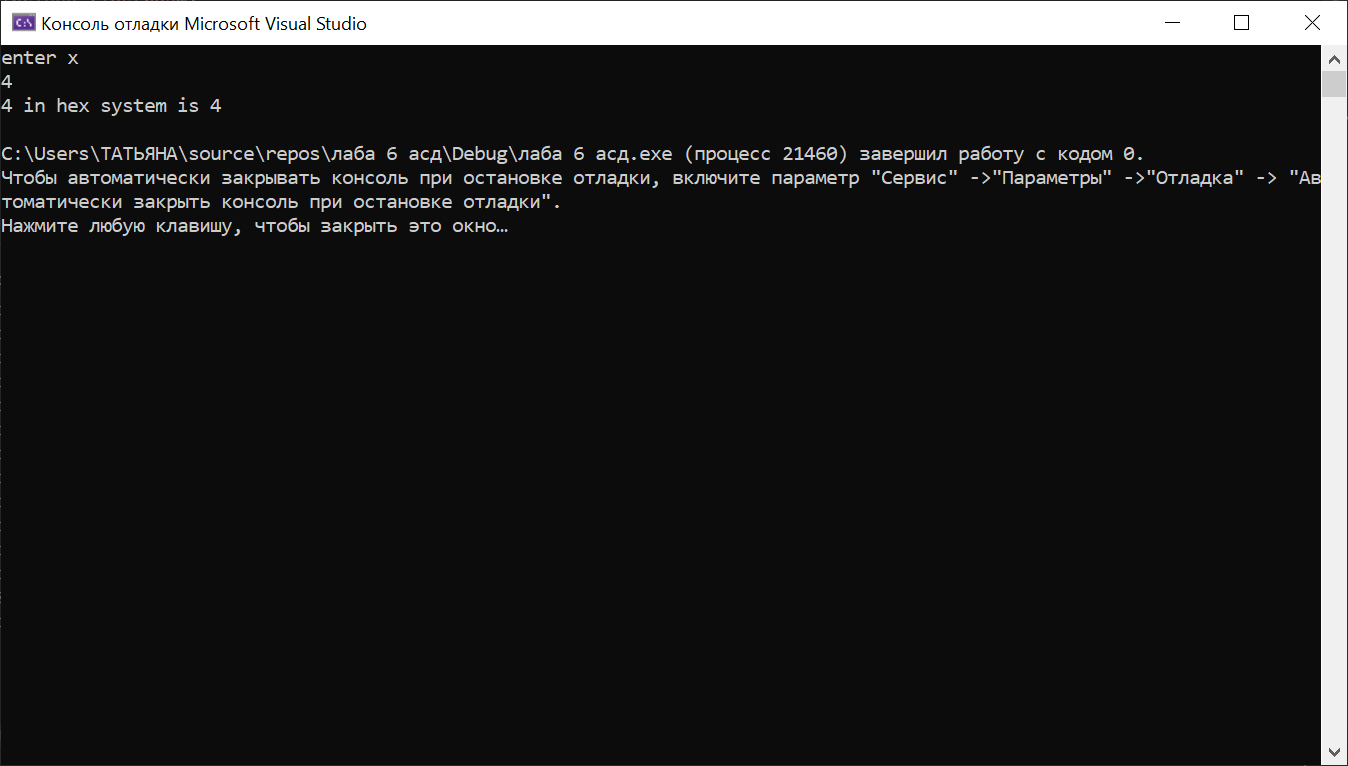












1. Випробування алгоритму

x=43

| num=43%16  num=11 | num=2%16  num=2 |
| --- | --- |
| x=43/16  x=2 | x=2/16  x=0 |

вивід: 211

x=666

| num=666%16  num=10 | num=41%16  num=9 | num=2%16  num=2 |
| --- | --- | --- |
| x=666/16  x=41 | x=41/16  x=2 | x=0 |

вивід:2910

x=345

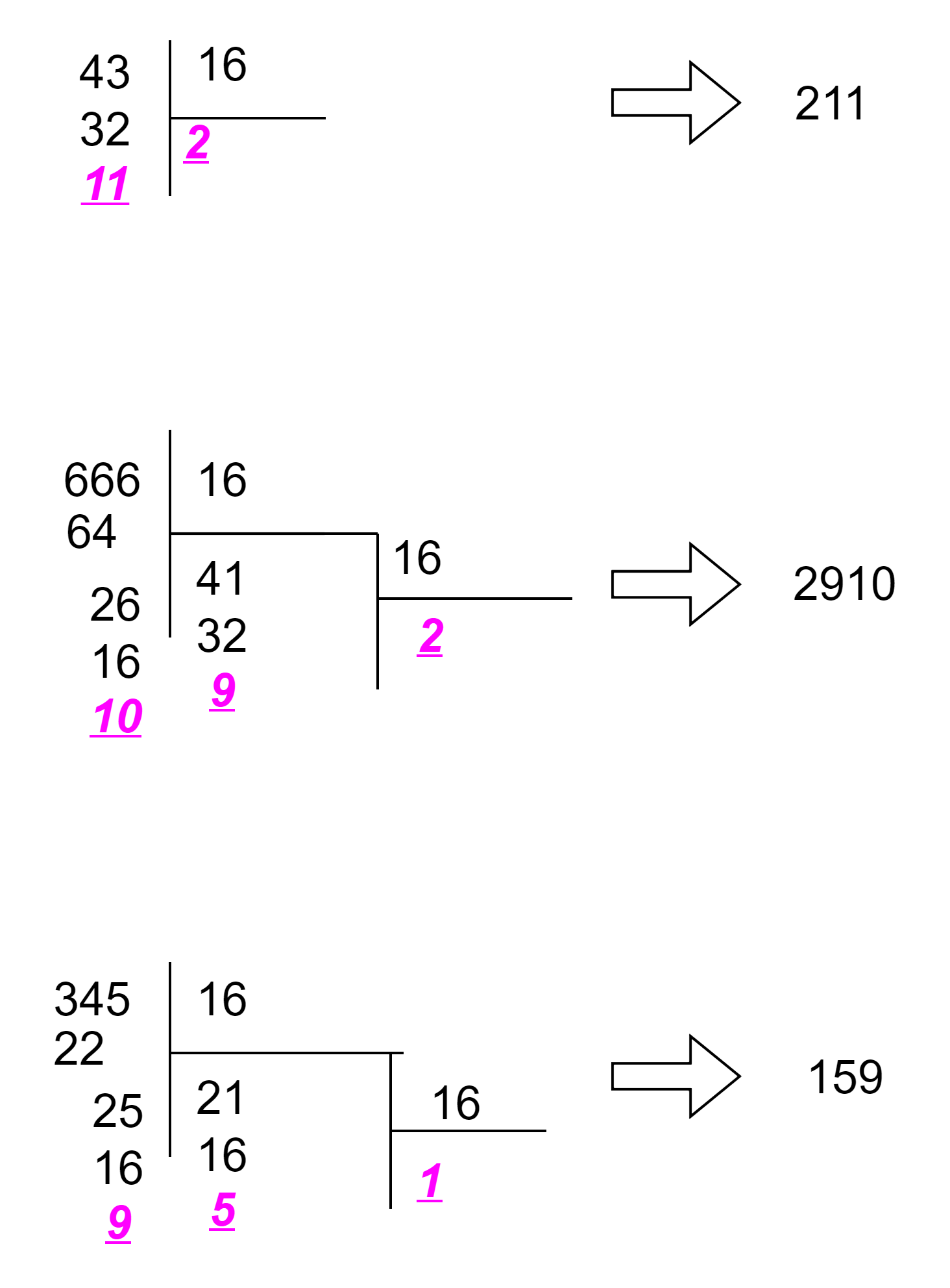
| num=345%16  num=9 | num=21%16  num=5 | num=1%16  num=1 |
| --- | --- | --- |
| x=345/16  x=21 | x=21/16  x=1 | x=0 |

вивід: 159

x=8888

| num=8 | num = 11=B | num = 2 | num = 2 |
| --- | --- | --- | --- |
| x=555 | x=34 | x=2 | x=0 |

вивід: 22B8



1. Висновки

Було досліджено особливості роботи рекурсивних алгоритмів та набуто практичних навичок їх використання під час складання програмної специфікації підпрограми. Для практики була дана задача з перетворення натурального числа у шістнадцятирічну систему, для якої було написано програму на мові с++ з використанням рекурсії.