Екзаменаційна робота

з навчальної дисципліни „Алгоритми та структури даних"

спеціальність 124 Системний аналіз

рівень вищої освіти перший (бакалаврський)

здобувача Гарт Анна Вікторівна

курс 2, група ПС-24-1

дата проведення екзамену 10 січня 2026 р.

Білет №4

(I) Циклічні алгоритми. Рекурсивні алгоритми

У програмуванні для автоматизації багаторазових обчислень або дій використовуються два фундаментальних підходи: ітерація (циклічні алгоритми) та рекурсія (рекурсивні алгоритми). Обидва методи дозволяють ефективно розв'язувати задачі, що мають повторювану структуру, але відрізняються за своєю логікою та застосуванням.

Циклічні алгоритми базуються на ітерації — багаторазовому виконанні певної послідовності інструкцій доти, доки виконується задана умова. Цикли є основою структурного програмування і зазвичай простіші для розуміння, оскільки чітко визначають початок, кінець та крок повторення.

Основні типи циклів:

1. Цикл з лічильником (for) – використовується, коли кількість ітерацій відома заздалегідь. Наприклад:

```
for (int i=0; i<n; i++) {
    // виконати дію
}
```

2. Цикл з умовою (while або do-while) – виконується, доки умова істинна:

```
1) while (умова) {
       // виконати дію
   }

2) do {
       // виконати дію
   } while (умова);
```

Основною перевагою циклів є їхня ефективність, оскільки вони часто потребують менше ресурсів (пам'яті) в порівнянні з рекурсією, адже не створюють додаткових викликів функцій та накладних витрат на організацію стека. Крім того, цикли пропонують простіший контроль за станом процесу, оскільки всі змінні оновлюються в межах однієї ітерації, що істотно спрощує процес відлагодження коду.

Проте головним недоліком циклічного підходу є його часто громіздка реалізація для задач, що природно мають рекурсивну структуру, наприклад, при обході графів, де код на основі циклів може стати складним для читання та підтримки.

Приклад застосування циклу для обчислення факторіалу $n!$:

```
int factorial = 1;
for (int i=1; i<n; i++){
    factorial = factorial * i
}
```

Рекурсивний алгоритм визначається через самовиклик функції для розв'язання підзадач того ж типу, що й початкова. Рекурсія застосовується, коли задача природно розкладається на менші екземпляри самої себе.

Складові рекурсії:

1. Базовий випадок - умова виходу з рекурсії, що перериває самовиклики.

2. Рекурсивний крок - виклик тієї самої функції з іншими параметрами.

Приклад застосування рекурсивної функції для обчислення факторіалу $n!$ (для $n \geq 0$):

```
int factorial(int n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * factorial(n-1);
    }
}
```

Головна перевага рекурсії полягає в лаконічності коду для задач, рекурсивно визначених за своєю природою. Вона дозволяє писати короткі та наочні рішення для обходу дерев, графів або реалізації алгоритмів швидкого сортування. Рекурсія спрощує розуміння складних проблем, розбиваючи їх на ідентичні підзадачі.

Основним недоліком рекурсії є значні витрати пам'яті, оскільки кожен рекурсивний виклик зберігає свій контекст у стеку викликів, що при глибокій рекурсії може призвести до критичної помилки переповнення стека. Крім того, без додаткової оптимізації багато рекурсивних алгоритмів страждають від неефективності через множинне обчислення одних і тих же підзадач, як це відбувається в найвідомішому прикладі з числами Фібоначчі.

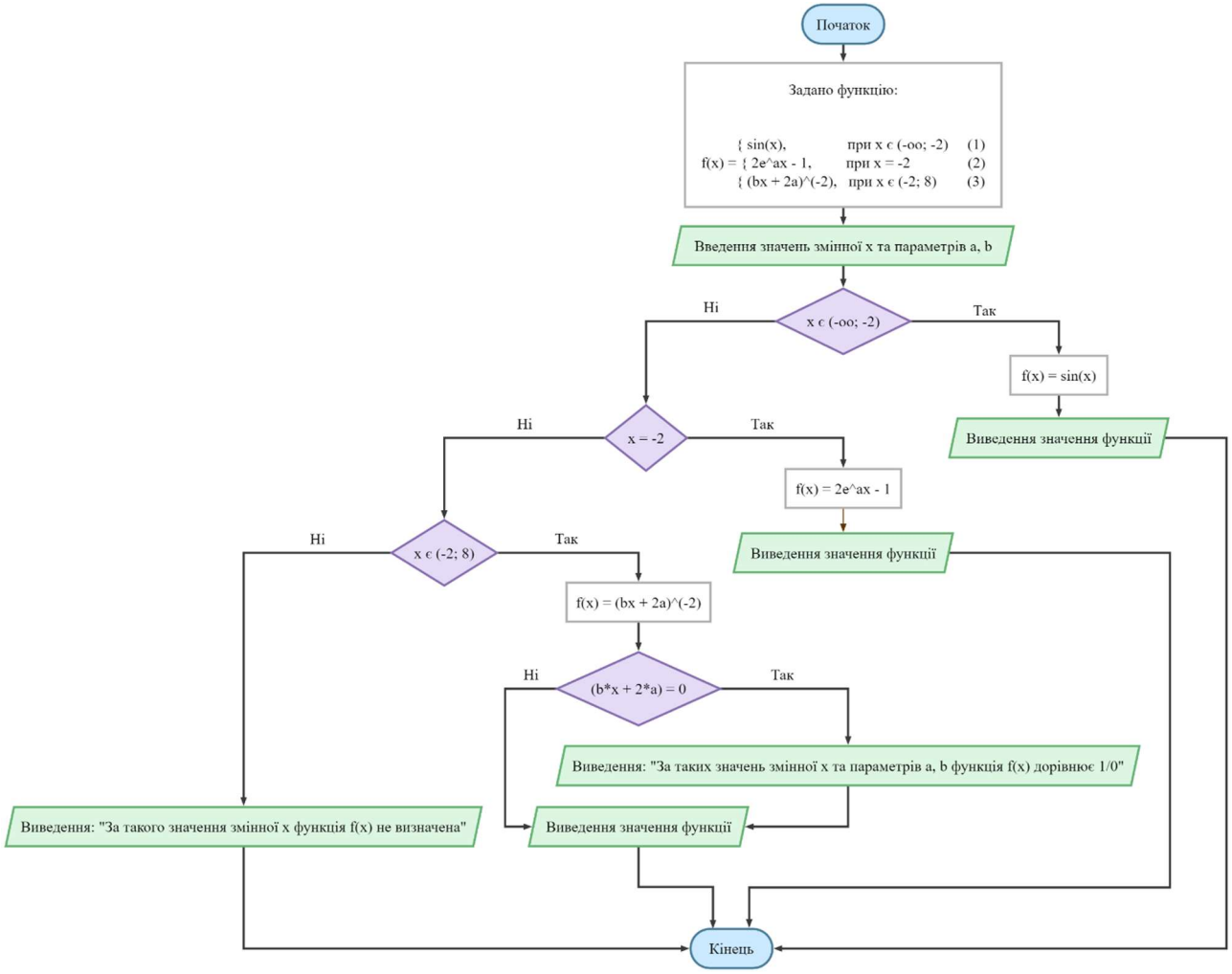
Висновок. Ітеративні та рекурсивні алгоритми — це взаємодоповнюючі підходи, розуміння яких дозволяє розробнику обрати оптимальне рішення, балансуючи між ефективністю та простотою коду.

II. Задача

Текст програми:

```
// Гарт Анна Вікторівна, група ПС-24-1, білет №4, завдання №2

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
    double a, b, x, y;
    system("chcp 65001");
    system("cls");

    printf ("\nОбчислення значення функції\n\n");

    printf ("       { sin(x),           при x є (-oo; -2)      (1)\n");
    printf ("f(x) = { 2e^ax - 1,        при x = -2             (2)\n");
    printf ("       { (bx + 2a)^(-2),   при x є (-2; 8)        (3)\n\n");

    printf ("Введіть параметри a, b: ");
    scanf_s ("%lf%lf", &a, &b);
    printf ("Введіть значення x: ");
    scanf_s("%lf", &x);

    // Розгалуження (1)
    if (x < -2) {
        printf ("\nОбчислення за розгалуженням (1):\n");
        y = sin(x);
        printf("f(%4.2f) = sin(%4.2f) = %6.4f\n", x, x, y);
    }
    // Розгалуження (2)
    else if (x == -2) {
        printf ("\nОбчислення за розгалуженням (2):\n");
        y = 2*exp(a*x) - 1;
        printf("f(-2) = 2e^(%4.2f)*(-2) - 1 = 2e^(%4.2f) - 1 = 2 * (%6.4f) - 1 =
%6.4f\n", a, a*x, exp(a*x), y);
    }
    // Розгалуження (3)
    else if (x > -2 && x < 8) {
        printf ("\nОбчислення за розгалуженням (3):\n");
        y = pow((b*x + 2*a), -2);
        if ((b*x + 2*a) != 0) {
            printf("f(%4.2f) = ( (%4.2f)*(%4.2f) + 2*(%4.2f) )^(-2) = ( (%4.2f) +
(%4.2f) )^(-2) = (%4.2f)^(-2) = %6.4f\n", x, b, x, a, b*x, 2*a, b*x+2*a, y);
        }
        else {
            printf("За параметрів a = %4.2f, b = %4.2f функція f(x) у точці x =
%4.2f набуває значення 1/0. Отже\n", a, b, x);
            printf("f(%4.2f) = ( (%4.2f)*(%4.2f) + 2*(%4.2f) )^(-2) = ( (%4.2f) +
(%4.2f) )^(-2) = (%4.2f)^(-2) = %6.4f\n", x, b, x, a, b*x, 2*a, b*x+2*a, y);
        }
    }
    else {
        printf ("\nФункція f(x) не визначена у точці x = %4.2f", x);
    }

    printf ("\n\n");
    system("pause");
    return 0;
}
```

Алгоритм:

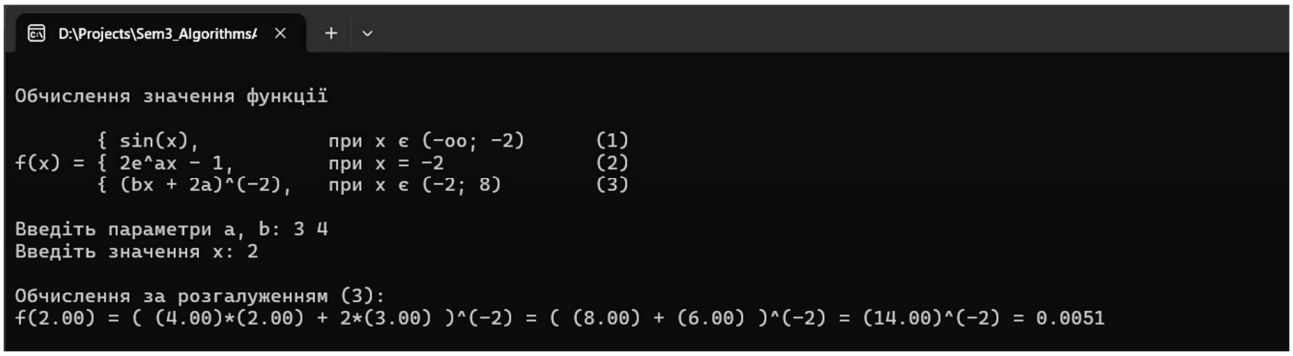


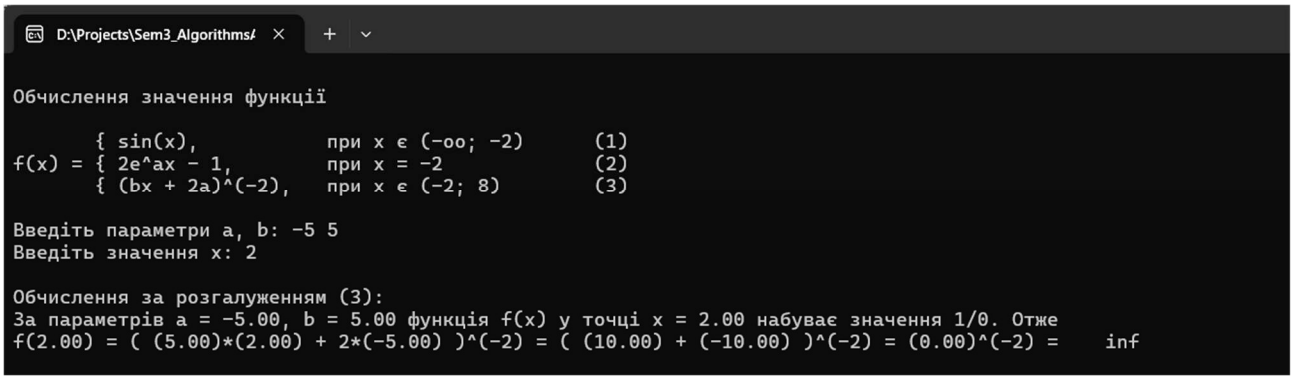
Для побудови алгоритму було використано сайт: https://www.yworks.com/yed-live

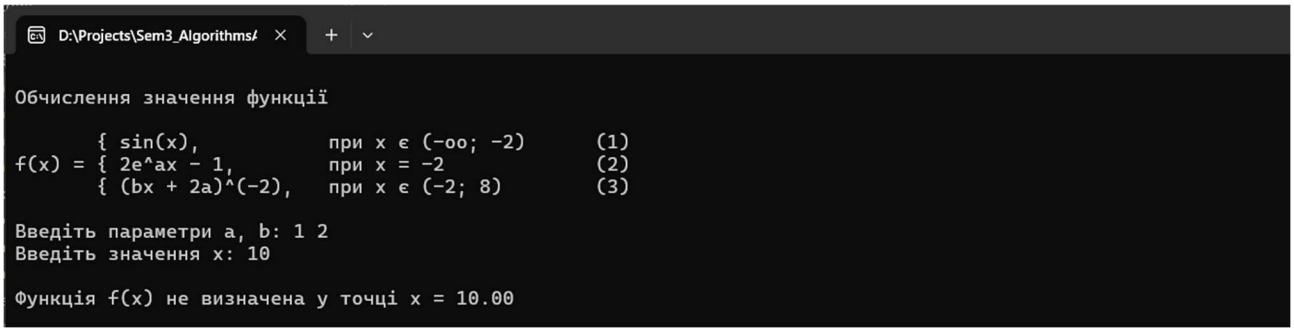
Результати роботи програми:

```
D:\Projects\Sem3_Algorithms\

Обчислення значення функції

      { sin(x),          при x є (-oo; -2)      (1)
f(x) = { 2e^ax - 1,      при x = -2             (2)
      { (bx + 2a)^(-2),  при x є (-2; 8)        (3)

Введіть параметри a, b: 1 1
Введіть значення x: -5

Обчислення за розгалуженням (1):
f(-5.00) = sin(-5.00) = 0.9589
```

```
D:\Projects\Sem3_Algorithms\

Обчислення значення функції

      { sin(x),          при x є (-oo; -2)      (1)
f(x) = { 2e^ax - 1,      при x = -2             (2)
      { (bx + 2a)^(-2),  при x є (-2; 8)        (3)

Введіть параметри a, b: -1 2
Введіть значення x: -2

Обчислення за розгалуженням (2):
f(-2) = 2e^(-1.00)*(-2) - 1 = 2e^(2.00) - 1 = 2 * (7.3891) - 1 = 13.7781
```







III. Задача

Текст програми:

```c
// Гарт Анна Вікторівна, група ПС-24-1, білет №4, завдання №3

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {

    system("chcp 65001");
    system("cls");

    printf("Sum = ∑(1/k), ∀k ∈ ℤ, k ∈ (1; e^x), де x ∈ ℝ, x > 0\n");

    double x;
    printf("\nEnter x (x > 0): ");
    scanf_s("%lf", &x);
    double upper = exp(x);      // e^x

    if (x > 0) {
        printf("\nSum = ∑(1/k), ∀k ∈ ℤ, k ∈ (1; %.3f)\n", upper);
        double sum = 0.0;
        printf("\nSum = 0.500");
        for (int k = 2; k <= (int)upper; k++) {
            sum += 1.0 / k;
```

```
            if (k != 2) {
                printf(" + %.3f", 1.0 / k);
            }
        }
        printf(" = %.3f\n", sum);
    }
    else {
        printf("\nПомилка. Введене значення x не задовольняє умову поставленої
задачі");
    }

    printf ("\n\n");
    system("pause");
    return 0;
}
```

Результати роботи програми: