МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 7
На тему: *"Вказівники на функції. Рекурсивні функції."*
З дисципліни: *"Основи програмування"*

Лектор:
ст.викл. каф. ПЗ
Тарас МУХА

Виконала:
ст. гр. ПЗ-15
Анна КОРНІЙЧУК

Прийняла:
ст.викл. каф. ПЗ
Іванна БУТРАК

« ____» _____ 2025 р.

∑= ____ .....

Львів – 2025

Тема роботи: вказівники на функції. Рекурсивні функції.

Мета роботи: поглиблене вивчення можливостей функцій в мові С з використанням механізмів рекурсії та вказівників.

Лабораторне завдання

1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.
2. Одержати індивідуальне завдання з Додатку 1.
3. Скласти програму на мові С у відповідності з розробленим алгоритмом.
4. Виконати обчислення по програмі. Навести характеристики продуктивності комп'ютера на якому було виконано програму.
5. Одержати індивідуальне завдання з Додатку 2.
6. Розробити алгоритм розв'язання індивідуального завдання і подати його у вигляді блок схеми.
7. Скласти програму на мові С у відповідності з розробленим алгоритмом.
8. Виконати обчислення по програмі при різних значеннях точності і порівняти отримані результати.
9. Одержати індивідуальне завдання з Додатку 3.
10. Розробити алгоритм розв'язання індивідуального завдання і подати його у вигляді блок схеми.
11. Скласти програму на мові С у відповідності з розробленим алгоритмом.
12. Виконати обчислення по програмі при різних значеннях точності і порівняти отримані результати.Підготувати та здати звіт про виконання лабораторної роботи.

Теоретичні відомості

Функція, як і дані, зберігається у пам'яті, має адресу, і на неї можна встановити вказівник на функцію. Оголошується такий вказівник за допомогою конструкції тип_результату (* ім'я_вказівника) (список_параметрів);, де круглі дужки навколо імені вказівника відрізняють його від оголошення функції, що повертає вказівник на дані. Оскільки ім'я функції в С є вказівником на першу комірку в пам'яті, для ініціалізації вказівнику просто присвоюється ім'я функції (наприклад, fst = f3;).

Виклик функції через вказівник може здійснюватися як (*fst)(a) або fst(a). Вказівники на функції широко застосовуються, наприклад, як аргументи бібліотечних функцій, для розробки універсальних функцій (чисельне інтегрування), або для організації меню за допомогою масивів вказівників на функції.

Паралельно, в мові С функції можуть бути рекурсивними — це функції, які звертаються самі до себе. Рекурсивний обчислювальний процес повинен бути організований так, щоб на кожному кроці задача спрощувалася до тих пір, поки не стане можливим нерекурсивне рішення (базовий випадок). Для створення рекурсивної функції важливо забезпечити передачу нових даних при кожному виклику, визначити умову припинення подальшого виклику та передачу результату в точку повернення. Типовими прикладами застосування рекурсії є обчислення факторіала та чисел Фібоначі , а також алгоритм швидкого сортування (сортування Хоара).

Індивідуальне завдання (Варіант 11)
Завдання з додатку 1

Використовуючи вищенаведені функції swap та qs_sort, які реалізують алгоритм швидкого сортування масиву, написати програму мовою С для порівняння ефективності алгоритмів сортування масивів великих обсягів (наприклад, 100000 елементів). Програма повинна також реалізувати один з класичних алгоритмів сортування масиву згідно з варіантом індивідуального завдання. У програмі використати два одинакових масиви, які заповнити випадковими числами, здійснити перевірку впорядкованості елементів масиву, перевірку ідентичності масивів до і

після сортування, а також за допомогою стандартної функції time, оцінити час виконання реалізованих алгоритмів сортування.

Сортування в порядку спадання “бульбашковим” методом без додаткової перевірки чи масив вже відсортований.

Назва файлу: main.c

#include <stdio.h> #include <stdlib.h> #include <time.h> #include "functions.h"

#define SIZE 100000

int main() { int* a = malloc(SIZE * sizeof(int)); int* b = malloc(SIZE * sizeof(int));

```
if (!a || !b) {
    printf("Memory allocation failed\n");
    return 1;
}

srand(time(NULL));
for (long i = 0; i < SIZE; i++) {
    int v = rand();
    a[i] = v;
    b[i] = v;
}

clock_t c1 = clock();
qs_sort(a, 0, SIZE - 1);
clock_t c2 = clock();

clock_t c3 = clock();
bubble_sort_desc(b, SIZE);
clock_t c4 = clock();

double qs_time = (double)(c2 - c1) * 1000.0 / CLOCKS_PER_SEC;
double bs_time = (double)(c4 - c3) * 1000.0 / CLOCKS_PER_SEC;

printf("QuickSort ascending sorted: %s\n", is_sorted_asc(a, SIZE) ? "YES" : "NO");
printf("Bubble sort descending sorted: %s\n", is_sorted_desc(b, SIZE) ? "YES" : "NO");
printf("Arrays identical before sorting: YES\n");

printf("QuickSort time: %.3f ms\n", qs_time);
printf("BubbleSort time: %.3f ms\n", bs_time);

free(a);
free(b);
return 0;


}
```

Назва файлу: functions.c

#include "functions.h"

void swap(int array[], long pos1, long pos2) { long tmp; tmp = array[pos1]; array[pos1] = array[pos2]; array[pos2] = tmp; }

void qs_sort(int array[], long start, long end) { long head = start, tail = end - 1, tmp; long diff = end - start; long pe_index;

```
if (diff < 1) return;

if (diff == 1) {
    if (array[start] > array[end]) {
        swap(array, start, end);
        return;
```

```
    }
    return;
}

long m = (start + end) / 2;
if (array[start] <= array[m]) {
    if (array[m] <= array[end]) pe_index = m;
    else if (array[end] <= array[start]) pe_index = start;
    else pe_index = end;
}
else {
    if (array[start] <= array[end]) pe_index = start;
    else if (array[end] <= array[m]) pe_index = m;
    else pe_index = end;
}

long pe = array[pe_index];
swap(array, pe_index, end);

while (1) {
    while (array[head] < pe)
        ++head;
    while (array[tail] > pe && tail > start)
        --tail;

    if (head >= tail) break;

    swap(array, head++, tail--);
}

swap(array, head, end);

long mid = head;

qs_sort(array, start, mid - 1);
qs_sort(array, mid + 1, end);


}
```

void bubble_sort_desc(int array[], long size) { for (long i = 0; i < size - 1; i++) { for (long j = 0; j < size - i - 1; j++) { if (array[j] < array[j + 1]) { int tmp = array[j]; array[j] = array[j + 1]; array[j + 1] = tmp; } } } }

int is_sorted_asc(int array[], long size) { for (long i = 0; i < size - 1; i++) if (array[i] > array[i + 1]) return 0; return 1; }

int is_sorted_desc(int array[], long size) { for (long i = 0; i < size - 1; i++) if (array[i] < array[i + 1]) return 0; return 1; }

int arrays_equal(int a[], int b[], long size) { for (long i = 0; i < size; i++) if (a[i] != b[i]) return 0; return 1; }

Назва файлу: functions.h

#pragma once #ifndef FUNCTIONS_H #define FUNCTIONS_H

void swap(int array[], long pos1, long pos2); void qs_sort(int array[], long start, long end); void bubble_sort_desc(int array[], long size); int is_sorted_asc(int array[], long size); int is_sorted_desc(int array[], long size); int arrays_equal(int a[], int b[], long size);

#endif

Результат виконання програми на рис. 1

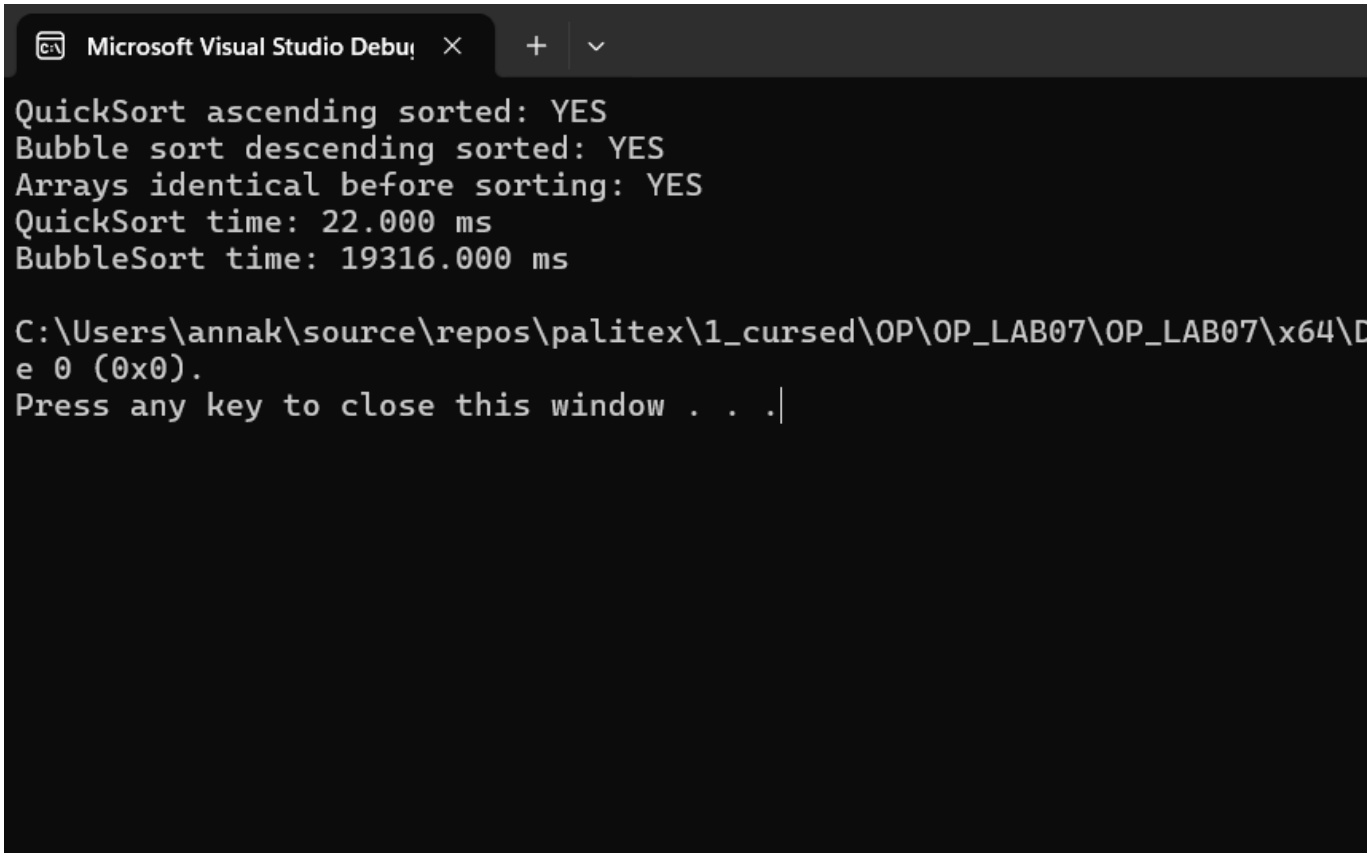

Рис. 1

Завдання з додатку 2

Написати мовою С три функції, щоб протабулювати, задану згідно варіанту, функцію на проміжку [a , b] з кроком h, використавши: а) для першої функції оператор циклу for; б) для другої – оператор циклу while; в) для третьої – – оператор циклу do…while. Вибір способу табулювання реалізувати через вказівник на відповідну функцію.

. $f = x\sqrt[3]{1-x}, a=1, b=9$;

Назва файлу: main.c

#include <stdio.h> #include "functions.h"

int main() { double a = 1.0; double b = 9.0; double h; int choice;

```
printf("Enter step h: ");
scanf("%lf", &h);

printf("1 - for\n2 - while\n3 - do while\nChoose: ");
scanf("%d", &choice);

void (*tab_ptr)(double, double, double) = 0;
```

```
if (choice == 1) tab_ptr = tab_for;
else if (choice == 2) tab_ptr = tab_while;
else if (choice == 3) tab_ptr = tab_do_while;
else {
    printf("Wrong option\n");
    return 0;
}

tab_ptr(a, b, h);

return 0;


}
```

Назва файлу: functions.c

#include <stdio.h> #include <math.h> #include "functions.h"

double f(double x) { return x * cbrt(1.0 - x); }

void tab_for(double a, double b, double h) { for (double x = a; x <= b; x += h) printf("x = %.3f f(x) = %.6f\n", x, f(x)); }

void tab_while(double a, double b, double h) { double x = a; while (x <= b) { printf("x = %.3f f(x) = %.6f\n", x, f(x)); x += h; } }

void tab_do_while(double a, double b, double h) { double x = a; if (h <= 0) return; do { printf("x = %.3f f(x) = %.6f\n", x, f(x)); x += h; } while (x <= b); }

Назва файлу: functions.h

#ifndef FUNCTIONS_H #define FUNCTIONS_H

double f(double x);

void tab_for(double a, double b, double h); void tab_while(double a, double b, double h); void tab_do_while(double a, double b, double h);

#endif

Результат виконання програми на рис. 2

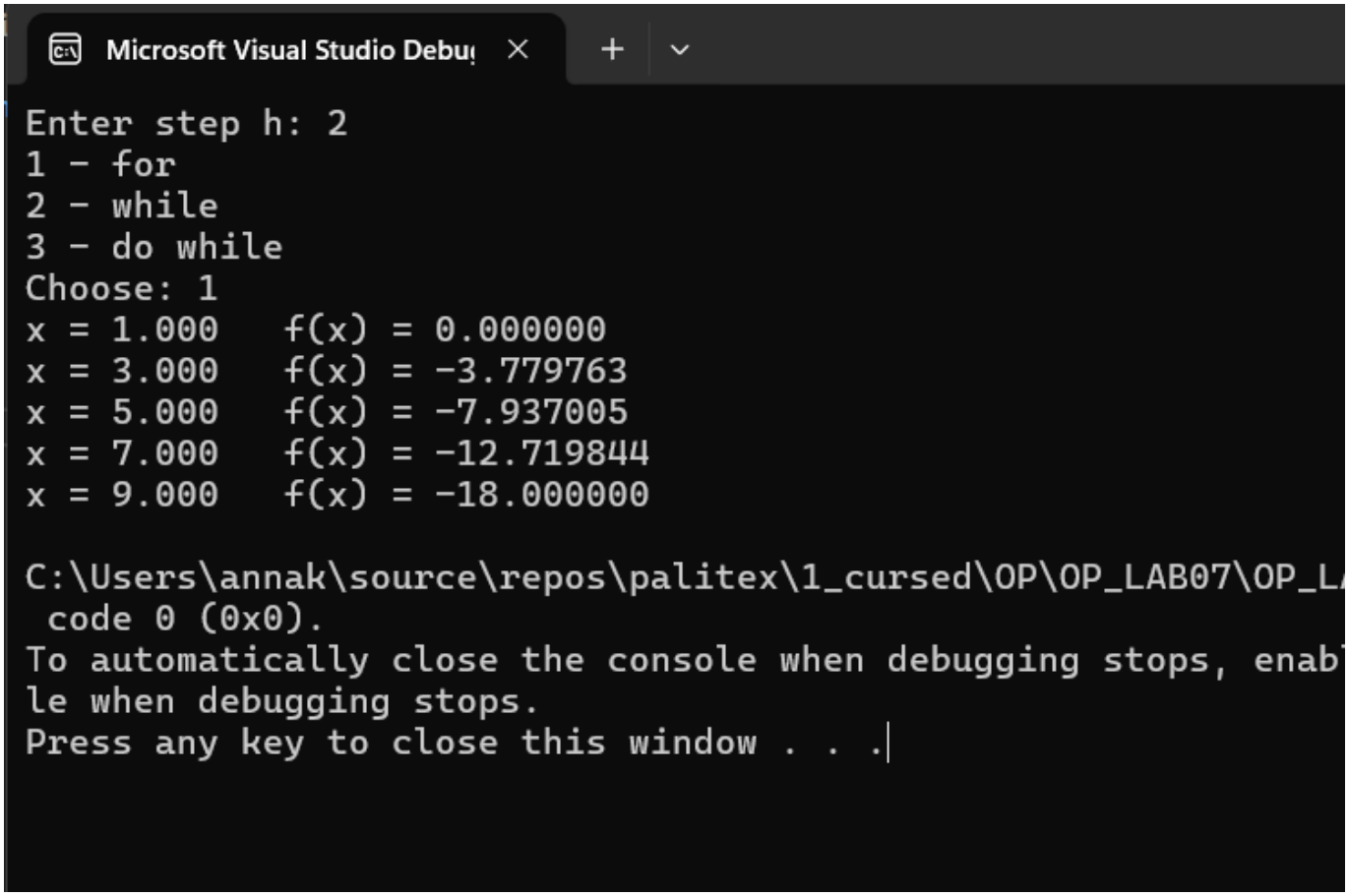

Рис. 2

Завдання з додатку 3

Використовуючи рекурсію, для кожного елемента з набору цілих чисел вирахувати залишок від ділення на 3. Масив не оголошувати.

Назва файлу: main.c

```
#include <stdio.h>

#include "functions.h"


int main()

{

    int n;

    printf("Enter count:\n");


    scanf_s("%d", &n);


    process(n);

    return 0;

}
```

Назва файлу: functions.c

```
#include <stdio.h>
#include "functions.h"

void process(int n)
{
    if (n == 0)
        return;

    int x;
    printf("Enter number: ");
    scanf_s("%d", &x);

    printf("%d %% 3 = %d\n", x, x % 3);

    process(n - 1);
}
```

Назва файлу: functions.h

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

void process(int n);

#endif
```

Результат виконання програми на рис. 3

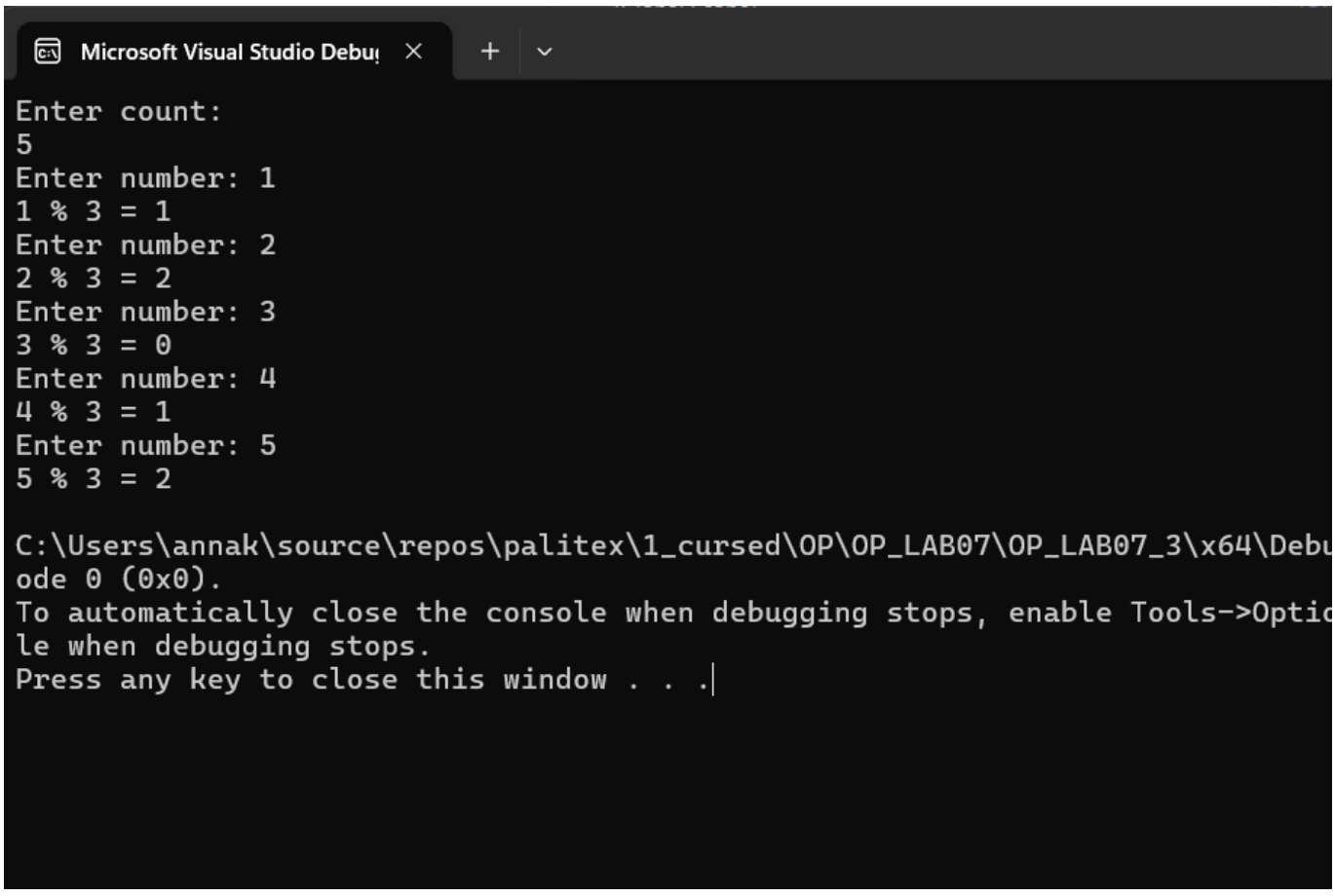

Рис. 3

Висновки

На цій лабораторній роботі я навчилася описувати та використовувати вказівники на функції в мові С, що дозволяє реалізовувати універсальні механізми вибору функцій. Крім того, я вивчила концепцію та правила створення рекурсивних функцій і ознайомилася з їх застосуванням на класичних прикладах, таких як обчислення факторіалу, чисел Фібоначі та швидке сортування.