Лабораторна робота №4

Функції. Швидке сортування. Робота з відлагоджувачем.

Мета

Вивчити можливості мови С щодо роботи з функціями, навчитися за їх допомогою розбивати програму (задачу) на підпрограми (підзадачі). Вивчити алгоритм «Швидке сортування». Ознайомитися з можливостями, які надає відлагоджувач.

Завдання 1

Реалізувати алгоритм «Швидке сортування» (див. лекції). Проілюструвати роботу алгоритму на масиві, заповненим випадковими числами та на масиві, введеним користувачем. Заповнення масиву числами, виведення масиву на екран реалізувати як окремі функції мови С. Необхідний напрям сортування наведений у додатку А.

Завдання 2

Відкрити проєкт з реалізованим завданням 1, обрати конфігурацію проєкту Debug, встановити точку останову на виклиці функції `quicksort`.

Запустити програму, після зупинки на точці продовжити виконання спочатку за допомогою команди Step Into, потім за допомогою Step Over. Вказати різницю у виконанні цих двох команд.

Відключити встановлену раніше точку останову, обравши пункт Disable Breakpoint у контекстному меню індикатора точки. Запустити програму, вказати відмінності від попереднього запуску.

Встановити курсор на рядок з циклом, призначеним для виведення відсортованого масиву на екран, виконати команду Run to Cursor з контекстного меню. Відкрити вікна Local та Autos, помістити до звіту скріншоти з описом вмісту цих вікон та поясненням відмінностей у них.

Встановити точку останову на першу інструкцію функції `split`, створити вікно Watch, до якого додати всі елементи масиву a (a[0], a[1], a[2] і т.д.), індекси low та high, змінну part_element. Виконати програму по крокам до завершення функції `split`, зробити скріншоти вікна Watch на кожному кроці та включити їх до звіту. Під час захисту лабораторної роботи пояснити роботу алгоритму розділу масиву, застосовуючи дані скріншоти.

Завдання 3

Реалізувати функцію згідно з варіантом завдання та розробити програму, що демонструє роботу даної функції. Для функції необхідно привести прототип, у коментарі до якого необхідно навести опис всіх параметрів та повертаного значення. Варіанти завдань наведено у додатку Б.

Розробити модульні тести з використанням Google Test Framework для перевірки коректності роботи функції.

Завдання 4

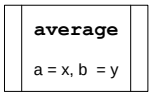
Відкомпілювати програми з завдань 1 та 3 спочатку у конфігурації Debug, потім у конфігурації Release. Заповнити нижченаведену таблицю:

№ завдання	Ім'я виконуваного файлу	Розмір файлу у конфігурації Debug	Розмір файлу у конфігурації Release

Зробити висновки щодо відмінностей у отриманих розмірах файлів, вказати причину таких відмінностей.

Виклик функцій на схемі алгоритму необхідно показувати за допомогою такого символу (ДСТУ ISO 5807:2016, п. 9.2.2.1):

Всередині символу необхідно вказати ім'я функції та передаванні аргументи, наприклад:



де `a` та `b` – параметри функції `average`, а `x` та `y` – передаванні аргументи.

Алгоритм роботи кожної функції необхідно представити як окрему схему.

Тексти програм мають починатися з коментаря, в якому зазначається ім'я файлу з вихідним кодом, прізвище та групу студента, дату складання, номер і назву лабораторної роботи, номер завдання:

```
/**
 * @file   lab4_1.cpp
 * @author Петренко П.П, гр. 515
 * @date   21 березня 2022
 * @brief  Лабораторна робота № 4
 *
 * Функції. Швидке сортування. Робота з відлагоджувачем. Завдання 1
 */
```

До прототипу кожної функції має бути записаний коментар, що містить інформацію щодо її призначення, параметрів та повертаного значення:

```
/**
 * Обчислення факторіалу числа з використанням рекурсії
 *
 * @param n число
 * @return факторіал числа (0, якщо n<0)
 */
int fact_rec (int n);
```

Вимоги до звіту

Звіт з даної лабораторної роботи має включати:

1. титульний аркуш із зазначенням номеру та назви лабораторної роботи;
2. мету роботи;
3. варіант і тексти завдань;
4. схеми алгоритмів;
5. вихідні тексти програм;
6. результати роботи програм (скріншоти або текст, який виводять програми);
7. результати перевірки на тестових набори + посилання на онлайн проєкти з тестовими наборами;
8. скріншоти вікон відлагоджувача з поясненням для завдання 2;
9. модульні тести та результати їх виконання для завдання 3;
10. висновки (що було зроблено, за допомогою яких засобів, що було вивчено і т.ін.).

Контрольні питання

1. Оголошення та визначення функції.
2. Параметри та аргументи функцій. Відмінності, приклади.
3. Які типи даних може повертати функція?
4. Що таке прототип функції?
5. Чи дозволені вкладені функції у мові C?
6. Для чого потрібен спеціальний тип даних `void`?
7. Яка функція називається рекурсивною?
8. Якими є правила запису імен функцій?
9. Чи може функція застосовуватися до її визначення? У яких випадках?
10. У чому полягають відмінності локальних та глобальних змінних?
11. Що таке синтаксична помилка? Чи виявляється вона компілятором?
12. Що таке логічна помилка? Як можна її виявити?
13. Що таке точка останову?
14. Чим відрізняється покрокове виконання за допомогою Step Into та Step Over?
15. Як продовжити виконання програми у звичайному режимі після її зупинки або покрокового виконання?
16. Для чого призначена команда відлагоджувача Run to Cursor?
17. Для чого призначені вікна відлагоджувача Local, Autos та Watch?
18. Як тимчасово відключити усі точки останову у програмі?
19. Як видалити усі точки останову з програми?
20. У чому полягає відмінність конфігурацій проєкту Debug та Release у середовищі Microsoft Visual Studio?

Додаток А. Варіанти завдань (завдання 1)

№	*Напрям сортування*
1.	За зростанням
2.	За спаданням
3.	За зростанням
4.	За спаданням
5.	За спаданням
6.	За зростанням
7.	За зростанням
8.	За спаданням
9.	За зростанням
10.	За спаданням
11.	За спаданням
12.	За зростанням
13.	За спаданням
14.	За зростанням
15.	За спаданням
16.	За спаданням
17.	За зростанням
18.	За спаданням
19.	За спаданням
20.	За зростанням
21.	За спаданням
22.	За спаданням
23.	За зростанням
24.	За спаданням
25.	За спаданням
26.	За зростанням
27.	За спаданням
28.	За спаданням
29.	За зростанням
30.	За спаданням

Додаток Б. Варіанти завдань (завдань 3)

№	*Завдання*
1.	Реалізувати функцію обчислення площі круга `circle_square`. Розробити програму обчислення площі кільця за значеннями внутрішнього та зовнішнього радіусів, застосовуючи функцію `circle_square`.
2.	Реалізувати функцію обчислення довжини відрізку `segment_length`, що з'єднує дві точки, за координатами цих точок. Розробити програму обчислення периметра трикутника за координатами трьох його вершин, застосовуючи функцію `segment_length`.
3.	Реалізувати функцію підрахунку суми цифр цілого числа `sum_digits`. Розробити програму, що запитує у користувача два цілих числа та за допомогою функції `sum_digits` визначає, сума цифр якого з них є більшою.
4.	Реалізувати функцію `find_greater_prime`, що знаходить просте число, яке є більшим ніж задане число. Розробити програму для перевірки роботи функції `find_greater_prime`.
5.	Реалізувати функцію `gcd` для обчислення найбільшого спільного дільника двох чисел. Розробити програму, що запитує у користувача три цілих числа та за допомогою функції `gcd` визначає найбільший спільний дільник трьох введених чисел. Застосовувати математичний факт: якщо *x*, *y*, *z* – три натуральних числа, то НСД (*x*,*y*,*z*)= НСД (НСД (*x*,*y*), *z*).
6.	Реалізувати функцію `is_prime`, що визначає, чи є її аргумент простим числом. Розробити програму, у якій застосовувати функцію `is_prime` для встановлення факту, чи є серед трьох введених чисел хоча б одне просте.
7.	Реалізувати функцію `sum_numbers`, що визначає суму всіх чисел до заданого натурального числа *n*. Так, наприклад, `sum_numbers(4)=10`. Розробити програму, що використовує функцію `sum_numbers` у програмі для введених з клавіатури чисел *a* та *b* для обчислення `sum_numbers(a) + sum_numbers(b)`.

8.	Реалізувати функцію `mul_numbers`, що визначає добуток усіх натуральних чисел до заданого числа *n*. Так, наприклад, `mul_numbers(4)=24`. Розробити програму, що використовує функцію `mul_numbers` у програмі для введених з клавіатури чисел *a* та *b* для обчислення `mul_numbers(a) + mul_numbers(b)`.
9.	Створити функцію `triangle_square` для обчислення площі трикутника за довжинами трьох його сторін (за формулою Герону). Розробити програму для знаходження площі чотирикутника за довжинами своїх сторін *a*, *b*, *c*, *d* та однієї діагоналі *e*. Застосовувати створену функцію `triangle_square`.
10.	Розробити програму знаходження кутів трикутника за довжинами його сторін *a*, *b*, *c* із застосуванням функції `get_corner` для знаходження кута.
11.	Реалізувати функцію `is_square`, що дозволяє перевірити, чи є число повним квадратом. Розробити програму, що запитує у користувача три числа та за допомогою викликів функції `is_square` перевіряє, чи є хоча б одно з них повним квадратом.
12.	Реалізувати функцію `angle`, що обчислює для прямокутного трикутника, заданого значеннями катетів *a* та *b*, кут, що лежить навпроти катету *a*. Функція `angle` має повертати значення кута у градусах. Розробити програму, що демонструє роботу даної функції.
13.	Реалізувати функцію `power` для зведення цілого числа до степені. З використанням функції `power` розробити програму для обчислення виразу $x^n+y^n+z^n$. Значення *x*, *y*, *z* и *n* запитати у користувача.
14.	Реалізувати функцію `gcd` для обчислення найбільшого спільного дільника двох чисел. З використанням функції `gcd` розробити програму для вирішення такої задачі: представити як звичайний нескоротний дріб суму двох дробив, вводимих користувачем.
15.	Реалізувати функцію `triangle_square` для обчислення площі трикутника за довжинами трьох його сторін (за формулою Герону). Розробити програму для знаходження сумарної площі двох трикутників з використанням реалізованої функції `triangle_square`.
16.	Розробити програму обчислення площі трикутника за координатами трьох його вершин, застосовуючи функцію обчислення довжини відрізка `segment_length`, що з'єднує дві точки.

17.	Реалізувати функцію `find_smaller_prime`, що знаходить просте число, яке є меншим ніж задане число. Розробити програму для перевірки роботи функції `find_smaller_prime`.
18.	Реалізувати функцію `is_cube`, що дозволяє перевірити, чи є число повним кубом. Розробити програму, що запитує у користувача три числа та за допомогою викликів функції `is_cube` перевіряє, чи є хоча б одне з них повним кубом.
19.	Реалізувати функцію `sum_numbers`, що визначає суму усіх чисел від заданого натурального числа *m* до заданого натурального числа *n*. Так, наприклад, `sum_numbers(4,6)=4+5+6=15`. Розробити програму, що використовує функцію `sum_numbers` у програмі для введених з клавіатури чисел *a*, *b*, *c*, *d* для обчислення `sum_numbers(a,b) + sum_numbers(c,d)`.
20.	Реалізувати функцію `is_palindrome`, що дозволяє перевірити, чи є число числовим паліндромом (числовий паліндром – це натуральне число, яке читається однаково зліва направо та справа наліво). Так, наприклад, `is_palindrome(121)` має дорівнювати 1. Розробити програму, що запитує у користувача три числа та за допомогою викликів функції `is_palindrome` перевіряє, чи є вони числовими паліндромами.
21.	Реалізувати функцію `get_first_digit`, що повертає першу цифру десяткового числа. Так, наприклад, `get_first_digit(486)` має дорівнювати 4, а `get_first_digit(2786)` має дорівнювати 2. Розробити програму, що запитує у користувача 4 числа та виводить найбільшу з перших цифр введених чисел.
22.	Реалізувати функцію `is_automorphic`, що дозволяє перевірити, чи є число автоморфним (автоморфне число – це число, десятковий запис квадрату якого закінчується цифрами самого цього числа). Так, наприклад, `is_automorphic(6)` має дорівнювати 1, тому що $6^2=36$ закінчується на 6. Аналогічно, `is_automorphic(25)` також має дорівнювати 1, тому що $25^2=625$ закінчується на 25. Розробити програму, що запитує у користувача число та виводить найближче до нього автоморфне число (або саме введене число, якщо воно є автоморфним).

23.	Реалізувати функцію `get_fibonacci`, що дозволяє отримати і-й член послідовності Фібоначчі (послідовність Фібоначчі будується таким чином: перший та другий члени послідовності дорівнюють 1, а кожен наступний дорівнює сумі двох попередніх, тобто 1, 1, 2, 3, 5, 8, 13, ...). Так, наприклад, `get_fibonacci(7)` має дорівнювати 13. Розробити програму, що запитує у користувача число N та виводить (N-1)-й, N-й и (N+1)-й члени послідовності Фібоначчі.
24.	Реалізувати функцію `is_trimorphic`, що дозволяє перевірити, чи є число триморфним (триморфне число – це число, десятковий запис куба якого закінчується цифрами самого цього числа). Так, наприклад, `is_trimorphic(4)` має дорівнювати 1, тому що 4^3=64 закінчується на 4. Аналогічно, `is_trimorphic(24)` також має дорівнювати 1, тому що 24^3=13824 закінчується на 24. Розробити програму, що перевіряє, чи є хоча б одне з введених користувачем трьох чисел триморфним.
25.	Реалізувати функцію `is_prime`, що визначає, чи є її аргумент простим числом. Розробити програму, у якій застосовувати функцію `is_prime` для підтвердження гіпотези Гольдбаха у диапазоні, заданим користувачем (гіпотеза Гольдбаха полягає у тому, що будь-яке парне число, починаючи з 4, можна представити як суму двох простих чисел).
26.	Реалізувати функцію `get_pronic`, що дозволяє отримати і-й член послідовності прямокутних чисел за формулою i*(i+1) Так, наприклад, `get_pronic (7)` має дорівнювати 56. Розробити програму, що запитує у користувача число N та виводить (N-1)-й, N-й и (N+1)-й члени послідовності прямокутних чисел.
27.	Реалізувати функцію `get_jacobsthal`, що дозволяє отримати і-й член послідовності Якобсталя за формулою $\frac{2^i-(-1)^i}{3}$ Так, наприклад, `get_jacobsthal (7)` має дорівнювати 43. Розробити програму, що запитує у користувача число N та виводить (N-1)-й, N-й и (N+1)-й члени послідовності Якобсталя.
28.	Реалізувати функцію підрахунку добутку цифр цілого числа `mul_digits`. Розробити програму, що запитує у користувача два цілих числа та за допомогою функції `mul_digits` визначає, добуток цифр якого з них є більшим.

29.	Реалізувати функцію `factorial`, що визначає факторіал натурального числа *n*. Так, наприклад, `factorial(4)=24`. Розробити програму, що використовує функцію `factorial` у програмі для введених з клавіатури чисел *a* та *b* для обчислення `factorial(a) + factorial(b)`.
30.	Реалізувати функцію `mul_numbers`, що визначає добуток усіх чисел від заданого натурального числа *m* до заданого натурального числа *n*. Так, наприклад, `mul_numbers(4,6)=4*5*6=120`. Розробити програму, що використовує функцію `mul_numbers` у програмі для введених з клавіатури чисел *a*, *b*, *c*, *d* для обчислення `mul_numbers(a,b) + mul_numbers(c,d)`.