Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 7

з навчальної дисципліни

“Базові методології та технології програмування”

ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРОБЛЕННЯ МАСИВІВ ДАНИХ ТА СИМВОЛЬНОЇ ІНФОРМАЦІЇ

ЗА СТАНДАРТОМ UNICODE

ЗАВДАННЯ ВИДАВ

доцент кафедри кібербезпеки та програмного забезпечення

Доренський О. П.

[https://github.com/odorenskyi/](https://github.com/odorenskyi/Dmytro-Parkhomenko-KB18)

ВИКОНАВ

студент академічної групи КБ-24

Кондратенко Дмитро

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Коваленко А. С.

Кропивницький – 2025

Мета роботи – розробити програмний засіб для обробки введеного користувачем слова, яке відповідає заданому формату, із забезпеченням коректного оброблення текстової інформації у кодуванні UTF-8 і CP866.

Аналіз умови і постановка задачі 7.1

1. Аналіз умови задачі

* Користувач вводить слово (послідовність літер без пробілів), яке закінчується одним із символів: '/', '%' або '$'.  
  Потрібно виконати такі дії:
* Підрахувати кількість складів "ку" у введеному слові.
* Якщо "ку" не зустрічається у слові, вивести рядок у зворотному порядку.

1.1 Вхідні дані:

* Строка (слово), що містить лише літери та закінчується одним із символів "/", "%", "$".

1.2 Вихідні дані:

* - Число — кількість входжень підрядка "ку" у слові.
* - АБО, якщо "ку" не знайдено, розгорнута версія введеного рядка (без кінцевого символу).

2. Постановка задачі

2.1 Вхідні обмеження:

* Введене слово містить лише літери (без пробілів, цифр, інших знаків).
* Слово обов’язково закінчується одним із символів "/", "%", "$".
* Мінімальна довжина слова — 2 символи (з урахуванням кінцевого знака).

2.2 Вихідні вимоги:

* Якщо "ку" є у слові → вивести кількість його входжень.
* Якщо "ку" відсутнє → вивести слово у зворотному порядку (без кінцевого знака).

3. Формалізований опис:

Дано: слово input, що закінчується на один із "/", "%", "$".

Потрібно:

Видалити останній символ (оскільки це стоп-символ).

Підрахувати входження "ку".

Якщо "ку" є → вивести його кількість.

Якщо "ку" немає → вивести слово задом наперед.

Процедурно-орієнтована алгоритмізація задачі 7.1 у вигляді псевдокоду:

ПРОЦЕДУРА count\_syllables(Рядок S)

Змінна count ← 0

Для i від 0 до Довжина(S) - 1 виконуємо:

Якщо S[i] = 'к' І S[i + 1] = 'у' ТОДІ

count ← count + 1

КІНЕЦЬ ЯКЩО

КІНЕЦЬ ДЛЯ

Повернути count

КІНЕЦЬ ПРОЦЕДУРИ

ПРОЦЕДУРА reverse\_string(Рядок S)

Змінна len ← Довжина(S)

Для i від 0 до (len / 2) виконуємо:

Обміняти S[i] ↔ S[len - 1 - i]

КІНЕЦЬ ДЛЯ

КІНЕЦЬ ПРОЦЕДУРИ

ГОЛОВНА ПРОЦЕДУРА main()

Вивести "Введіть слово, що закінчується '/', '%' або '$':"

Зчитати input

len ← Довжина(input)

last\_char ← input[len - 1]

Якщо last\_char ≠ '/' І last\_char ≠ '%' І last\_char ≠ '$' ТОДІ

Вивести "Некоректний ввід! Слово повинно закінчуватися '/', '%' або '$'."

Вийти з програми

КІНЕЦЬ ЯКЩО

Видалити останній символ input

count ← count\_syllables(input)

Якщо count > 0 ТОДІ

Вивести "Кількість складів 'ку': ", count

ІНАКШЕ

reverse\_string(input)

Вивести "Рядок у зворотному порядку: ", input

КІНЕЦЬ ЯКЩО

КІНЕЦЬ ПРОЦЕДУРИ

Лістинг проекту Kondratenko-task\_7\_1:

#include <stdio.h>

#include <string.h>

#include <windows.h>

#include <locale.h>

int count\_syllables(const char \*str) {

setlocale(LC\_ALL, "");

int count = 0;

for (int i = 0; str[i] != '\0'; i++) {

if (str[i] == 'к' && str[i + 1] == 'у') {

count++;

}

}

return count;

}

void reverse\_string(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

char temp = str[i];

str[i] = str[len - 1 - i];

str[len - 1 - i] = temp;

}

}

int main() {

// Встановлюємо Windows-кодування CP1251 (для української мови)

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char input[100];

printf("Введіть слово, що закінчується '/', '%%' або '$': ");

scanf("%99s", input);

int len = strlen(input);

char last\_char = input[len - 1];

if (last\_char == '/' || last\_char == '%' || last\_char == '$') {

input[len - 1] = '\0'; // Видаляємо останній символ

int count = count\_syllables(input);

if (count > 0) {

printf("Кількість складів 'ку': %d\n", count);

} else {

reverse\_string(input);

printf("Рядок у зворотному порядку: %s\n", input);

}

} else {

printf("Некоректний ввід! Слово повинно закінчуватися '/', '%%' або '$'.\n");

}

return 0;

}

Аналіз умови задачі 7.2:

**Вхідні дані**:

* Є масив із 16 дійсних чисел (числа з плаваючою комою).

**Вихідні дані**:

* Потрібно визначити порядковий номер (індекс) найбільшого та найменшого елементів масиву.

**Деталі реалізації**:

* Необхідно знайти максимальний і мінімальний елемент масиву.
* Визначити їх позиції в масиві (індексація може починатися з 0 або 1 — це потрібно уточнити у вимогах до реалізації).
* Вивести отримані індекси.

Процедурно-орієнтована алгоритмізація задачі 7.2 у вигляді псевдокоду:

Оголосити КОНСТАНТУ SIZE = 16

Оголосити МАСИВ numbers розміром SIZE (дійсні числа)

Оголосити ЗМІННІ min\_index, max\_index (цілі числа)

ФУНКЦІЯ find\_min\_max(arr, size, min\_index, max\_index)

Встановити min\_index = 0

Встановити max\_index = 0

ДЛЯ i від 1 ДО size - 1

ЯКЩО arr[i] < arr[min\_index] ТОДІ

min\_index = i

КІНЕЦЬ ЯКЩО

ЯКЩО arr[i] > arr[max\_index] ТОДІ

max\_index = i

КІНЕЦЬ ЯКЩО

КІНЕЦЬ ДЛЯ

КІНЕЦЬ ФУНКЦІЇ

ПРОЦЕДУРА main()

Вивести "Введіть 16 дійсних чисел:"

ДЛЯ i від 0 ДО SIZE - 1

Вивести "Елемент ", i + 1, ": "

Ввід numbers[i]

КІНЕЦЬ ДЛЯ

Викликати find\_min\_max(numbers, SIZE, min\_index, max\_index)

Вивести "Найменший елемент має номер: ", min\_index + 1

Вивести "Найбільший елемент має номер: ", max\_index + 1

КІНЕЦЬ ПРОЦЕДУРИ main

# Лістинг програмного коду проекту Kondratenko-task\_7\_2:

#include <stdio.h>

#include <locale.h>

#define SIZE 16 // Розмір масиву

void find\_min\_max(double arr[], int size, int \*min\_index, int \*max\_index) {

\*min\_index = 0;

\*max\_index = 0;

for (int i = 1; i < size; i++) {

if (arr[i] < arr[\*min\_index]) {

\*min\_index = i;

}

if (arr[i] > arr[\*max\_index]) {

\*max\_index = i;

}

}

}

int main() {

setlocale(LC\_ALL, "");

double numbers[SIZE];

int min\_index, max\_index;

// Ввід масиву

printf("Введіть %d дійсних чисел:\n", SIZE);

for (int i = 0; i < SIZE; i++) {

printf("Елемент %d: ", i + 1);

scanf("%lf", &numbers[i]);

}

// Пошук індексів мінімального та максимального елементів

find\_min\_max(numbers, SIZE, &min\_index, &max\_index);

// Вивід результату (номери починаються з 1)

printf("Найменший елемент має номер: %d\n", min\_index + 1);

printf("Найбільший елемент має номер: %d\n", max\_index + 1);

return 0;

}

**Аргументи досягнення мети лабораторної роботи:**

* 1. Ознайомлення з принципами обробки масивів у мові C.
  2. Розуміння методів пошуку мінімального та максимального значень у масиві.
  3. Вивчення принципів побудови алгоритмів у процедурному стилі.
  4. Ознайомлення з функціями вводу/виводу в C.
  5. Вивчення механізму передачі масивів у функції.
  6. Дослідження методів роботи з покажчиками у функціях.
  7. Аналіз ефективності алгоритмів пошуку в масиві.
  8. Ознайомлення зі стандартними бібліотеками для роботи з числами.
  9. Вивчення локалізації в програмуванні.
  10. Дослідження впливу кодування символів на вивід даних.
  11. Реалізація введення та виведення дійсних чисел.
  12. Розробка функції для пошуку мінімального та максимального елемента масиву.
  13. Реалізація коректного виводу порядкових номерів елементів масиву.
  14. Оптимізація роботи алгоритму пошуку мін/макс значень.
  15. Вивчення роботи з константами у програмі.
  16. Використання покажчиків для зміни змінних у функціях.
  17. Реалізація циклів для обробки елементів масиву.
  18. Використання умовних конструкцій для перевірки числових значень.
  19. Розвиток навичок роботи з компілятором та відлагоджувачем.
  20. Практика роботи з локалізацією виводу програми.
  21. Використання зрозумілих імен змінних та функцій.
  22. Дотримання принципів структурного програмування.
  23. Застосування коментарів для пояснення коду.
  24. Дотримання правил оформлення коду згідно з код-стилем.
  25. Реалізація безпечного вводу даних.
  26. Мінімізація використання глобальних змінних.
  27. Використання ефективних алгоритмів для обробки даних.
  28. Дотримання правил типізації змінних.
  29. Запобігання можливим помилкам при введенні даних.
  30. Використання модульного підходу при побудові функцій.
  31. Перевірка коректності роботи програми на різних наборах даних.
  32. Аналіз крайових випадків (усі числа рівні, мін/макс – перші або останні елементи).
  33. Виявлення та виправлення помилок у роботі алгоритму.
  34. Використання інструментів для відлагодження коду.
  35. Оптимізація швидкодії програми.
  36. Порівняння отриманих результатів з очікуваними.
  37. Дослідження роботи програми в різних середовищах виконання.
  38. Перевірка правильності обчислень та їх коректного виводу.
  39. Оцінка ефективності реалізованого алгоритму.
  40. Аналіз поведінки програми при введенні некоректних даних.
  41. Опис структури програми у звіті.
  42. Аналіз досягнутих результатів.
  43. Висновки про ефективність реалізованого алгоритму.
  44. Опис можливих покращень коду.
  45. Порівняння отриманих знань із теоретичними очікуваннями.
  46. Оцінка застосування отриманих знань у реальних проєктах.
  47. Оформлення звіту згідно з методичними рекомендаціями.
  48. Обговорення результатів із викладачем або колегами.
  49. Презентація виконаної роботи та її демонстрація.
  50. Використання отриманих знань у наступних лабораторних роботах.

Відповіді на контрольні запитання:

* git init – Ініціалізує новий локальний Git-репозиторій у поточній теці. Виконується один раз на початку роботи з новим проєктом.
* git add (наприклад, git add “file”) – Додає файли до індексу (стадії підготовки). git add . додає всі змінені та нові файли у поточній теці.
* git commit -m "текст\_коміту" – Фіксує (зберігає) зміни у локальному репозиторії з коментарем "текст\_коміту", що пояснює внесені зміни.
* git remote add origin git@github.com:username/BMTP-LAB7-прізвище.git – Додає віддалений (remote) репозиторій із заданою URL-адресою (у цьому випадку на GitHub). "origin" – це умовна назва цього віддаленого репозиторію.
* git push – Відправляє локальні коміти у віддалений репозиторій. Якщо це перше відправлення, зазвичай використовується git push -u origin main (main – приклад), що встановлює відстеження гілки.