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

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАВ
студент академічної групи КБ-23
Литвин М.В

ПЕРЕВІРИВ
викладач кафедри кібербезпеки
та програмного забезпечення
Дрєєва Г.М.

Кропивницький – 2024

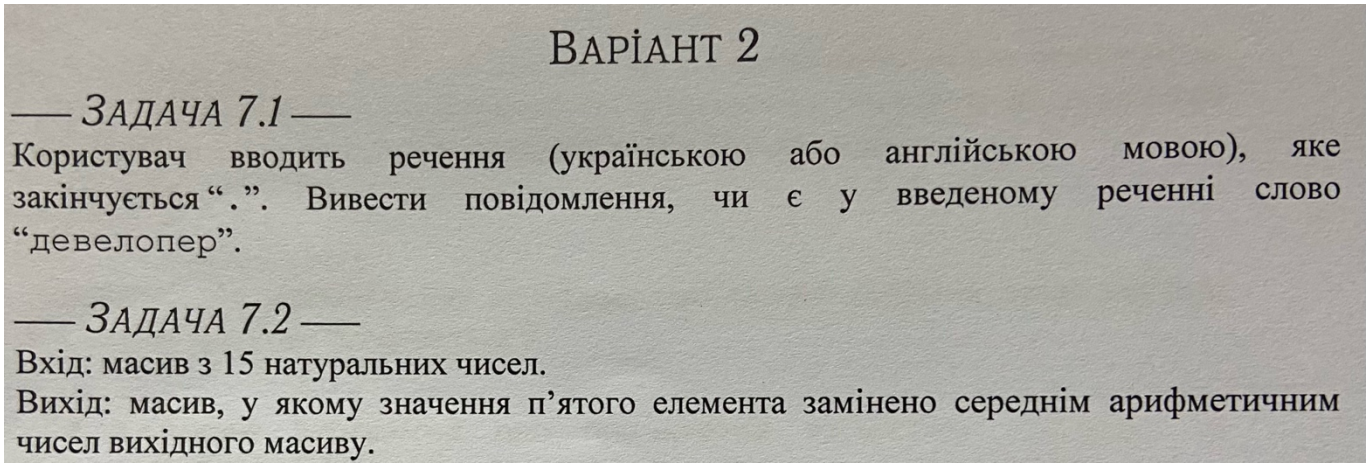
Лабораторна робота 7

Тема: Програмна реалізація оброблення масивів данних та символьної інформації за стандартом Unicode.

Мета: Полягає у набутті ґрунтовних вмінь і практичних навичок синтезу алгоритмів оброблення масивів даних та символьної (текстової) інформації у кодуваннях UTF-8 і CP866, їх програмної реалізації мовою програмування мовою програмування C (ISO/IEC 9899:2018) задля реалізації програмних засобів у вільному кросплатформовому Code::Blocks IDE.

Завдання:

1. Створитиперсональнийобліковийзапис GitHub.
2. Реалізувати програмне забезпечення розв’язування задачі 7.1.
3. Реалізувати програмне забезпечення розв’язування задачі 7.2.
4. Створити Git-репозиторій для спільної роботи над проєктом з контролем версій.



ВАРІАНТ 2

— ЗАДАЧА 7.1 —

Користувач вводить речення (українською або англійською мовою), яке закінчується “.”. Вивести повідомлення, чи є у введеному реченні слово “девелопер”.

— ЗАДАЧА 7.2 —

Вхід: масив з 15 натуральних чисел.

Вихід: масив, у якому значення п’ятого елемента замінено середнім арифметичним чисел вихідного масиву.

Варіант 2

Алгоритм 7.1:

1. Оголосити змінну **sentence** для зберігання введеного речення.
2. Оголосити змінні **word_eng** та **word_eng_upper** для англійських версій слова "developer" з різним регістром.
3. Оголосити змінні **word_ukr** та **word_ukr_upper** для українських версій слова "девелопер" з різним регістром.
4. Оголосити змінні **found_eng** та **found_ukr** для позначення наявності відповідних слів у реченні (ініціалізувати їх як 0).
5. Вивести повідомлення "Enter a sentence: ".
6. Зчитати введене речення за допомогою **fgets** і зберегти його в змінну **sentence**.
7. Видалити символ нового рядка з кінця речення, якщо він присутній.
8. Розділити речення на окремі слова за пробілами та іншими символами розділення за допомогою функції **strtok**.
9. Для кожного слова в реченні перевірити, чи є воно одним зі слова "developer" англійською мовою або словом "девелопер" українською мовою. Для цього порівняти кожне слово з відповідними словами, враховуючи різницю в регістрі (за допомогою **strcmp** та **strcasecmp**).
10. Якщо слово знайдено, встановити відповідний прапорець **found_eng** або **found_ukr**.

11. Вивести результат: якщо **found_eng** або **found_ukr** встановлені, вивести відповідне повідомлення про наявність слова, в іншому випадку вивести повідомлення про його відсутність.

Лістинг програми 7.1

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char sentence[1000];
    char *word_eng = "developer";
    char *word_eng_upper = "Developer";
    char *word_ukr = "девелопер";
    char *word_ukr_upper = "Девелопер";
    int found_eng = 0;
    int found_ukr = 0;

    printf("Enter a sentence: ");
    fgets(sentence, sizeof(sentence), stdin);

    sentence[strcspn(sentence, "\n")] = 0;

    char *token = strtok(sentence, " ");
    while (token != NULL) {
        if (strcmp(token, word_eng) == 0 || strcmp(token, word_eng_upper) == 0 ||
strcasecmp(token, word_eng) == 0) {
            found_eng = 1;
        }
        if (strcmp(token, word_ukr) == 0 || strcmp(token, word_ukr_upper) == 0 ||
strcasecmp(token, word_ukr) == 0) {
            found_ukr = 1;
        }
        token = strtok(NULL, " ");
    }

    if (found_eng) {
        printf("The word \"developer\" is found in the input sentence.\n");
    } else {
        printf("The word \"developer\" is not found in the input sentence.\n");
    }

    if (found_ukr) {
        printf("Слово \"девелопер\" знайдено у введеному реченні.\n");
    } else {
        printf("Слово \"девелопер\" не знайдено у введеному реченні.\n");
    }

    return 0;
}
```

Лістинг програми 7.1 з коментарями

```
#include <stdio.h>  // Підключення бібліотеки введення/виведення
#include <string.h> // Підключення бібліотеки для роботи з рядками
#include <ctype.h>  // Підключення бібліотеки для роботи з символами

int main() {
    char sentence[1000];                // Оголошення масиву для зберігання введеного
речення
    char *word_eng = "developer";       // Оголошення англійського слова "developer"
    char *word_eng_upper = "Developer"; // Оголошення англійського слова "Developer"
    char *word_ukr = "девелопер";       // Оголошення українського слова "девелопер"
```

```
    char *word_ukr_upper = "Девелопер"; // Оголошення українського слова "Девелопер"
    int found_eng = 0;                   // Прапорець для виявлення англійського
слова
    int found_ukr = 0;                   // Прапорець для виявлення українського
слова

    printf("Enter a sentence: ");       // Виведення повідомлення про введення
речення
    fgets(sentence, sizeof(sentence), stdin); // Отримання введеного речення

    sentence[strcspn(sentence, "\n")] = 0;   // Видалення символу нового рядка з
кінця речення

    char *token = strtok(sentence, " "); // Розділення речення на слова
    while (token != NULL) {
        if (strcmp(token, word_eng) == 0 || strcmp(token, word_eng_upper) == 0 ||
strcasecmp(token, word_eng) == 0) {
            found_eng = 1; // Позначення знайденого англійського слова
        }
        if (strcmp(token, word_ukr) == 0 || strcmp(token, word_ukr_upper) == 0 ||
strcasecmp(token, word_ukr) == 0) {
            found_ukr = 1; // Позначення знайденого українського слова
        }
        token = strtok(NULL, " "); // Перехід до наступного слова
    }

    if (found_eng) {
        printf("The word \"developer\" is found in the input sentence.\n"); //
Виведення результату для англійської мови
    } else {
        printf("The word \"developer\" is not found in the input sentence.\n"); //
Виведення результату для англійської мови
    }

    if (found_ukr) {
        printf("Слово \"девелопер\" знайдено у введеному реченні.\n"); // Виведення
результату для української мови
    } else {
        printf("Слово \"девелопер\" не знайдено у введеному реченні.\n"); //
Виведення результату для української мови
    }

    return 0; // Повернення 0 для позначення успішного завершення програми
}
```

Алгоритм 7.2:

1. Оголосити масив **numbers** розміром 15 для зберігання натуральних чисел.
2. Оголосити змінні **sum** і **count** для обчислення суми чисел та кількості введених чисел відповідно.
3. Вивести повідомлення "Введіть 15 натуральних чисел:".
    3.1 Запустити цикл **for** для отримання 15 натуральних чисел введених користувачем.
    3.2 У циклі перевірити, чи введене значення є натуральним числом (більше за нуль).
    3.3 Якщо введене значення не задовольняє умову, вивести повідомлення про некоректне введення та попросити ввести натуральне число знову.
    3.4 Додати введене число до суми **sum**, зберегти його в **numbers[i]** та збільшити лічильник **count**.
4. Перевірити, чи користувач ввів всі 15 чисел. Якщо **count** менше 15, вивести повідомлення про недостатню кількість чисел та завершити програму.

5. Обчислити середнє арифметичне чисел, поділивши суму на кількість чисел.
6. Замінити п'ятий елемент масиву на отримане середнє арифметичне.
7. Вивести змінений масив.

Лістинг програми 7.2

```
#include <stdio.h>

int main() {
    int numbers[15];
    int sum = 0;
    int count = 0;

    printf("Введіть 15 натуральних чисел:\n");
    for (int i = 0; i < 15; i++) {
        int num;
        if (scanf("%d", &num) != 1 || num <= 0) {
            printf("Некоректне введення. Введіть натуральне число.\n");
            while (getchar() != '\n');
            i--;
            continue;
        }
        numbers[i] = num;
        sum += numbers[i];
        count++;
    }

    if (count < 15) {
        printf("Введено недостатню кількість чисел. Потрібно ввести ще %d чисел.\n",
15 - count);
        return 1;
    }

    float average = (float)sum / 15;

    numbers[4] = average;

    printf("Змінений масив:\n");
    for (int i = 0; i < 15; i++) {
        printf("%d ", numbers[i]);
    }
    printf("\n");

    return 0;
}
```

Лістинг програми 7.2 з коментарями

```
#include <stdio.h> // Підключення бібліотеки введення/виведення
int main() {
    int numbers[15]; // Оголошення масиву чисел розміром 15
    int sum = 0; // Ініціалізація змінної для зберігання суми чисел
    int count = 0; // Ініціалізація змінної для підрахунку введених чисел

    printf("Введіть 15 натуральних чисел:\n"); // Виведення повідомлення про введення
чисел
    for (int i = 0; i < 15; i++) { // Цикл для введення 15 чисел
        int num;
        // Перевірка правильності введення та перевірка на натуральність числа
        if (scanf("%d", &num) != 1 || num <= 0) {
            printf("Некоректне введення. Введіть натуральне число.\n"); //
Повідомлення про некоректне введення
            while (getchar() != '\n'); // Очищення буфера введення
            i--; // Повторення введення для поточного числа
```

```
            continue; // Перехід до наступної ітерації циклу
        }
        numbers[i] = num; // Збереження введеного числа у відповідну комірку масиву
        sum += numbers[i]; // Додавання введеного числа до суми
        count++; // Збільшення лічильника введених чисел
    }

    if (count < 15) { // Перевірка, чи введено всі 15 чисел
        printf("Введено недостатню кількість чисел. Потрібно ввести ще %d чисел.\n",
15 - count); // Повідомлення про недостатню кількість чисел
        return 1; // Повернення значення 1 для позначення помилки
    }

    float average = (float)sum / 15; // Обчислення середнього арифметичного

    numbers[4] = average; // Заміна п'ятого елемента масиву на середнє арифметичне

    printf("Змінений масив:\n"); // Виведення повідомлення про змінений масив
    for (int i = 0; i < 15; i++) { // Цикл для виведення елементів масиву
        printf("%d ", numbers[i]); // Виведення елементу масиву
    }
    printf("\n"); // Виведення символу нового рядка

    return 0; // Повернення значення 0 для позначення успішного завершення програми
}
```

Висновок: В ході вивчення теми "Програмна реалізація оброблення масивів даних та символьної інформації за стандартом Unicode" ми отримали цінні навички і знання, необхідні для розробки програм, які працюють з різними типами даних і текстовою інформацією в кодуваннях UTF-8 і CP866. Мета нашої роботи полягала у вивченні ґрунтовних принципів роботи з масивами даних і текстовою інформацією, а також у освоєнні практичних навичок реалізації алгоритмів на мові програмування C (згідно із стандартом ISO/IEC 9899:2018). Ми оволоділи роботою з інтегрованим середовищем розробки, що дозволяє зручно та ефективно створювати програмні засоби. Під час вивчення теми ми ознайомилися з особливостями роботи з різними кодуваннями, що є важливим для розробки програм, що працюють з міжнародними даними і текстом. Обидва вони спрямовані на отримання цінних навичок і знань для розробки програм, які працюють з різними типами даних і текстовою інформацією в кодуваннях UTF-8 і CP866. Обидва також мають на меті вивчення принципів роботи з масивами даних і текстовою інформацією, а також освоєння практичних навичок реалізації алгоритмів на мові програмування C за стандартом ISO/IEC 9899:2018. Отже, обидва годування допомагають оволодіти роботою з інтегрованим середовищем розробки для створення програмних засобів. Однак перше годування може бути більш фокусованим на роботі з міжнародними даними і текстом, оскільки воно включає ознайомлення з особливостями роботи з різними кодуваннями, що є важливим для розробки програм, що працюють з міжнародними даними і текстом. Отже, завдяки вивченню цієї теми ми здобули не лише теоретичні знання, але й практичні навички, які будуть корисні нам у подальших програмних проектах.