МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 7

з навчальної дисципліни

“Базові методології та технології програмування”

на тему

«ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРОБЛЕННЯ МАСИВІВ ДАНИХ ТА СИМВОЛЬНОЇ ІНФОРМАЦІЇ ЗА СТАНДАРТОМ UNICODE»

ВИКОНАВ

студент академічної групи КБ-23

ЛінніковаСофія

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

Дрєєва Г.М.

Кропивницький – 2024

**Лабораторна робота №7**

**Тема:** програмна реалізація симетричного алгоритму шифрування підстановками.

**Мета:**набуття 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 С (ISO/IEC 9899:2018) задля реалізації програмних засобів у вільному кросплатформовомуCode::Blocks IDE.

**Завдання:**

1. Створити персональний обліковий запис 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.

Хід роботи

1. У веббраузері перейти за посиланням https://github.com/ та зареєструватись на вебсервісі системи контролю версій GitHub.

2. Після створення облікового запису (account) і підтвердження реєстрації через електронну пошту, у Personalsettings заповнити Publicprofile: Name: власне ім’я й прізвище латиницею, Profilepicture: офіційне фото (ділового стилю), URL: www.kntu.kr.ua, Company: CentralUkrainianNationalTechnicalUniversity, Location: Ukraine, Kropyvnytskyi.

3. Продемонструвати викладачеві створений обліковий запис, після чого надіслати GitHubusername на dorenskyiop@kntu.kr.ua.

4. Отримати у викладача завдання за варіантом, задокументувати мету лабораторної роботи, завдання, номер варіанту.

5. На одному з локальних дисків ПК (D:, E: чи ін.) створити теку BMTP-LAB7-прізвище, в ній ― теки prj, Software та TestSuite.

6. Виконати аналіз умови і постановку задачі 7.1, встановлених вимог до програмного засобу, який розроблятиметься, на основі чого розробити й задокументувати тест-сьют TS\_7\_1, який повинен складатись з не менш як десяти тест-кейсів.

7. Виконати процедурно-орієнтовану алгоритмізацію задачі 7.1 з врахуванням вимоги про програмне оброблення текстової інформації у UTF-8, при чому кодування вхідних і вихідних даних залежить від джерела й приймача інформації відповідно; здобутий алгоритм розв’язування задачі 7.1 задокументувати.

8. В Code::Blocks IDE створити новий проект консольного застосунку з назвою прізвище-task\_7\_1 у /prj.

9. Алгоритм розв’язування задачі 7.1 реалізувати мовою програмування С (С18), проект прізвище-task\_7\_1 закрити.

10. З теки проекта прізвище-task\_7\_1 за допомогою "Блокнот" відкрити файл main.c.

11. Перезберегти файл main.c у UTF-8 (Файл → Зберегти як → Кодування → UTF-8 → Зберегти).

12. Закрити файл main.c.

13. В Code::Blocks IDE відкрити прізвище-task\_7\_1, скомпілювати і закрити проект; створений у /prj файл програми прізвищеtask\_7\_1.ехе скопіювати у /BMTP-LAB7-прізвище/Software.

14. За допомогою тестового набору TS\_7\_1 із теки /TestSuite, виконати системне тестування прізвище-task\_7\_1.ехе, який знаходиться у теці /BMTP-LAB7-прізвище/Software; отриманий тестовий артефакт — заповнений тест-сьют TS\_7\_1 — зберегти та долучити до звіту як додаток.

15. За умови, що статус всіх тест-кейсів TS\_7\_1 є passed, вихідний код (лістинг) проекту прізвище-task\_7\_1 задокументувати; у іншому випадку слід виконати відлагодження запропонованого алгоритму і/або ПЗ, проект прізвище-task\_7\_1 скомпілювати, системне тестування додатка прізвище-task\_7\_1.ехе повторити, результати повторної реалізації й тестування задокументувати.

16. Виконати аналіз умови і постановку задачі 7.2, встановлених вимог до програмного засобу, який розроблятиметься, на основі чого розробити й задокументувати тест-сьют TS\_7\_2, який повинен складатись з не менш як десяти тест-кейсів.

17. Виконати процедурно-орієнтовану алгоритмізацію задачі 7.2, отриманий алгоритм розв’язування задачі 7.2 задокументувати.

18. В Code::Blocks IDE створити новий проект консольного застосунку з назвою прізвище-task\_7\_2 у /prj.

19. Алгоритм розв’язування задачі 7.2 реалізувати мовою програмування С (С18), скомпілювати проект; створений файл програми прізвище-task\_7\_2.ехе скопіювати у /Software.

20. За допомогою TS\_7\_2 із /TestSuite, виконати системне тестування /BMTP-LAB2-прізвище/Software/прізвище-task\_7\_2.ехе; заповнений TS\_7\_2 долучити до звіту як додаток.

21. За умови, що статус всіх тест-кейсів TS\_7\_2 є passed, вихідний код (лістинг) проекту прізвище-task\_7\_2 задокументувати; у іншому випадку слід виконати відлагодження запропонованого алгоритму і/або ПЗ, проект прізвище-task\_7\_2 скомпілювати, системне тестування додатка прізвище-task\_7\_2.ехе повторити, результати повторної реалізації й тестування задокументувати.

22. Одержані результати виконання завдань лабораторної роботи (/BMTP-LAB7-прізвище) зберегти на флеш-накопичувач, у хмарному сховищі даних тощо. 45

23. Проаналізувати хід виконання завдань і одержані результати, сформулювати обґрунтовані висновки15 та викласти їх обсягом не менше двох сторінок машинного (комп’ютерного) тексту; також слід зазначити особисті враження від виконання лабораторної роботи, викласти вмотивовані пропозиції, обґрунтовані зауваження, конструктивну критику16, рекомендації тощо.

24. Інсталювати Git (GitBash) https://git-scm.com/book/uk/v2/ВступІнсталяція-Git, використовуючи “ProGit” <https://gitscm.com/book/uk/v2>.

25. На github.com/student-name створити Git-репозиторій (New → Create a newrepository) та за допомогою Git (GitBash) завантажити у нього теку BMTP-LAB7-прізвище (gitinit, gitadd, gitcommit, gitbranch, gitremote, gitpush; https://techglimpse.com/git-push-github-token-based-passwordless/), а також звіт, оформлений згідно з ДСТУ 3008:2015 зі структурою й змістом, визначеними цим порядком виконання роботи.

26. Додати викладача і одногрупників до спільної роботи над проєктом BMTP-LAB7-прізвище: Settings → Access → Collaborators → Addpeople → Add NAME to REPOSITORY (NAME викладача — на дошці в аудиторії, у якій проходить заняття).

27. У комітахGit-репозиторію BMTP-LAB7-прізвище (https://github.com/ . .. /commits/main/) відреагувати і/або відповісти на коментарі співучасників ІТ-проєкту ( Commits), захистити результати й здобути зарахування роботи викладачем.

**Варіант 20**

**Завдання 7.1**

Користувач вводить слово (послідовність символів літер без проміжків), яке закінчується «!» і складається з не менше 6 літер. Вивести кількість складів "ко" у введеному слові; якщо зазначений склад відсутній, то вивести введений рядок з попарно зміненими місцями літерами

Аналіз:

Вхідні: слова без проміжків, що закінчується на !, не менше 6 літер;

Вихідні: кількість «ко», якщо відсутній змінити літери попарно;

Алгоритм:

Задати розмір масиву для зберігання введеного слова.

Зчитати введене слово в масив.

Перевірити, чи введене слово закінчується на "!" і має не менше 6 літер.

Якщо умова не виконується, вивести повідомлення про некоректний ввід.

Якщо умова виконується, існують дві можливості:

Порахувати кількість складів "ко".

Якщо склад "ко" відсутній, попарно змінити місцями літери в рядку і вивести результат.

Лістинг програми:

#include<stdio.h>

#include<string.h>

intmain() {

charword[50];

// Введення слова від користувача

printf("Введіть слово, яке закінчується '!' та має не менше 6 літер: ");

scanf("%s", word);

// Перевірка чи слово закінчується '!' та має не менше 6 літер

if (word[strlen(word) - 1] == '!' &&strlen(word) >= 6) {

intcount\_ko = 0;

// Підрахунок кількості складів "ко"

for (int i = 0; i <strlen(word) - 1; i++) {

if (word[i] == 'к' &&word[i + 1] == 'о') {

count\_ko++;

}

}

// Виведення результатів

if (count\_ko> 0) {

printf("Кількість складів 'ко' у слові: %d\n", count\_ko);

} else {

// Попарна зміна місцями літер

for (int i = 0; i <strlen(word) - 1; i += 2) {

chartemp = word[i];

word[i] = word[i + 1];

word[i + 1] = temp;

}

printf("Слово з попарно зміненими місцями літер: %s\n", word);

}

} else {

printf("Некоректне введення.\n");

}

return 0;

}

**Завдання 2**

Вихід: найбільший і найменший елемент масиву, середнє арифметичне всіх додатних елементів; масив, у якому перші 15 елементів змінені місцями з останніми 15 елементами.

**Аналіз:**

Вхідні: масив з 30 дійсних чисел

Вихідні: найбільший і найменший елемент масиву, середнє арифметичних додатних елементів, перші 15 елементів змінені місяцями

Алгоритм:

1. Ініціалізувати масив з 30 дійсних чисел
2. Зчитати 30 чисел у масив.
3. Знайти найбільший та найменший елементи у масиві.
4. Обчислити середнє арифметичне всіх додатніх елементів.
5. Змінити місцями перші 15 елементів масиву з останніми 15 елементами.

**Лістинг:**

#include<stdio.h>

intmain() {

// Ініціалізація масиву та змінних

char\*locale=setlocale(LC\_ALL, “”)

doublenumbers[30];

int i;

doublesum\_positive = 0;

intcount\_positive = 0;

// Ввід масиву

printf("Введіть 30 дійсних чисел:\n");

for (i = 0; i < 30; i++) {

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

// Обчислення суми та кількості додатніх чисел

if (numbers[i] > 0) {

sum\_positive += numbers[i];

count\_positive++;

}

}

// Пошук найбільшого та найменшого елементів

doublemax = numbers[0];

doublemin = numbers[0];

for (i = 1; i < 30; i++) {

if (numbers[i] >max) {

max = numbers[i];

}

if (numbers[i] <min) {

min = numbers[i];

}

}

// Обчислення середнього арифметичного додатніх елементів

doubleaverage\_positive = count\_positive> 0 ? sum\_positive / count\_positive : 0;

// Зміна місцями перших та останніх 15 елементів масиву

for (i = 0; i < 15; i++) {

doubletemp = numbers[i];

numbers[i] = numbers[30 - 15 + i];

numbers[30 - 15 + i] = temp;

}

// Виведення результатів

printf("\nНайбільший елемент: %lf\n", max);

printf("Найменший елемент: %lf\n", min);

printf("Середнє арифметичне додатніх елементів: %lf\n", average\_positive);

printf("\nМасив змінений місцями:\n");

for (i = 0; i < 30; i++) {

printf("%lf ", numbers[i]);

}

return 0;

}

Висновок: Із заданої теми та мети лабораторної роботи виникає висновок про те, що її основна мета — набуття 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 С (ISO/IEC 9899:2018) задля реалізації програмних засобів у вільному кросплатформовомуCode::Blocks IDE. Основні результати роботи включають розробку програм, які розв'язують задачі, а також їхню компіляцію та виконання у вказаному середовищі. Проаналізувавши результати виконання лабораторнихможна зробити наступні висновки:

Використання мови програмування C: Розроблені програми відповідають стандартам мови C (ISO/IEC 9899:2018). Використаноалгоритми повторення для вирішення конкретних завдань, що відповідає поставленим вимогам.

Робота в середовищі Code::Blocks: Програми успішно компілюються та виконуються у вільному кросплатформовому інтегрованому середовищі Code::Blocks з використанням GNU GCC Compiler.

Аналіз та оптимізація коду: Визначено можливості оптимізації коду, таких як вилучення зайвих змінних та покращення структури програм. Рекомендації з покращення читабельності коду та ефективності виконання.

Особисті враження: Мені вдалося виконати усі поставлені задачі та це був цікавий досвід.

Отже, виконання лабораторної роботи відображає успішне набуття базових навичок програмування та використання мови C у конкретному інтегрованому середовищі також допомагає здобути досвід роботи у GitHub, який є невід’ємною частиною роботи програміста.