МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 10

з навчальної дисципліни

“Базові методології та технології програмування”

Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням

ВИКОНАВ

Студент

академічної групи КН-23

Царенко Станіслав

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

Ганна ДРЄЄВА

Кропивницький – 2024

**ТЕМА:** Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням

**МЕТА:**— полягає у набутті ґрунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структура, об’єднання, множина, перелік, перетворення типів даних, використання файлових потоків та функцій стандартних бібліотек для оброблення символьної інформації.

**ЗАВДАННЯ:**

1. Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище лабораторних робіт №8–9).

2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.

**ПОРЯДОК ВИКОНАННЯ:**

1. Завантажити власний Git-репозиторій https://github.com/odorenskyi/ student-name (в \Lab4\tasks містяться умови задач 10.1–10.3)

2. У \Lab10 заповнити файл README.md, створити теки prj, TestSuite, Software, Report; отриманий вміст теки \Lab10 завантажити до Git-репозиторію https://github.com/odorenskyi/student-name; надалі здійснювати означену дію за позначкою .

3. До звіту з лабораторної роботи (далі ― звіт) записати мету роботи, номер варіанту, завдання.

4. Почергово виконати аналіз і постановку задач 10.1, 10.2, 10.3, аналіз вимог до ПЗ та вмісту вхідного текстового файлу (див. умови задач), проектування архітектури, детальне проектування програмних модулів; одержані артефакти задокументувати й включити до звіту ; під час проектування функцій слід врахувати, що вхідні дані модуля – ім’я вхідного і/або вихідного файлу; якщо вихідний файл існує, то його вміст знищується, інакше – файл створюється.

5. Розробити три тест-сьюти (набори контрольних прикладів) задля проведення автоматизованого unit-тестування програмних модулів розв’язування задач 10.1–10.3; отримані тестові артефакти задокументувати (наприклад, у вигляді таблиці), зберегти у \Lab10\TestSuite та включити до звіту ; рекомендовано таку структуру тест-кейса: Preliminary Steps – ім’я вхідного файлу та його вміст (текст) і/або ім’я вихідного файлу; Action (test steps) – виклик відповідного модуля з вхідними даними (значення аргументів); Expected Result – вміст вхідного/вихідного текстового файлу.

6. В Code::Blocks IDE відкрити проект статичної бібліотеки ModulesПрізвище з \Lab8\prj, створений під час виконання лабораторної роботи № 8.

7. За отриманими під час проектування програмних модулів артефактами виконати конструювання функцій: мовою програмування С++ реалізувати функції, які за наданим інтерфейсом реалізовують розв’язування задач 10.1, 10.2 та 10.3 відповідно; проект ModulesПрізвище доповнюється новими функціями.

8. Скомпілювати проект статичної бібліотеки ModulesПрізвище .

9. Відкрити проект заголовкового файлу ModulesПрізвище, створений під час виконання лабораторної роботи № 8, та доповнити його прототипами реалізованих функцій 10.1–10.3.

10. У середовищі Code::Blocks створити в теці \Lab10\prj проект консольного додатка, іменувати його TestDriver.

11. Мовою програмування С++ реалізувати консольний застосунок – тестовий драйвер для модульного тестування функцій розв’язування задач 10.1–10.3 за допомогою розроблених тест-сьютів з \Lab10\TestSuite та вхідного і/або вихідного текстового файлу . необхідно реалізувати протоколювання процесу тестування: запис у файл вхідних даних (аргументів функції, яка тестується), отриманий результат та статус тест-кейса (passed або failed); назва вхідного/вихідного файлу(ів) передається відповідним функціям з ModulesПрізвище.h як аргументи; для включення функцій бібліотеки libModulesПрізвище.а до вихідного коду драйвера слід використати заголовковий файл ModulesПрізвище.h та належно налаштувати опції компілятора (Build options…); контрольні приклади рекомендовано представити константними масивами, елементи яких у циклі передаються на оброблення відповідною функцією з ModulesПрізвище.h; є можливим реалізація тестовим драйвером або повністю автоматизованого, або напівавтоматизованого unit-тестування: для автоматизованого слід реалізувати алгоритм виконання відповідного тест-кейса: А1) драйвер за Preliminary Steps створює вхідний файл та записує в нього вхідний текст і/або створює вихідний файл, А2) за Action викликається функція з аргументами – ім’я/іменами файлів, А3) відкривається модифікований функцією з кроку А2 файл, зчитується текст з нього та порівнюється з текстом із поля Expected Result; А4) результат порівняння (Test Result passed / failed) виводиться у стандартний файловий потік; для напіватоматизованого кроки А1, А3 та А4 алгоритму автоматизованого виконання тест-кейса реалізовуються тестувальником вручну за допомогою текстового редактора, а крок А2 ― тестовим драйвером.

12. Створений застосунок TestDriver.ехе перемістити у \Lab10\Software .

13. За допомогою TestDriver.ехе виконати автоматизоване тестування розроблених функцій розв’язування задач 10.1–10.3. у випадку невиконання тест-кейса(ів) слід виконати відлагодження відповідної функції, перекомпілювати проект статичної бібліотеки ModulesПрізвище, модульне тестування повторити.

14. Результати модульного тестування відповідних функцій статичної бібліотеки libModulesПрізвище.а задокументувати шляхом включення (копіювання) результатів роботи тестового драйвера \Lab4\Software\TestDriver.ехе до звіту .

15. Вихідний код (текст) проектів ModulesПрізвище та TestDriver включити до звіту як додатки.

16. Проаналізувати хід виконання лабораторних завдань і самостійно одержані результати, на основі чого сформулювати обґрунтовані висновки 1 з виконаної лабораторної роботи, викласти їх обсягом не менше двох сторінок машинного (комп’ютерного) тексту та включити до звіту .

17. Підготувати й зберегти у \Lab10\Report звіт про виконання лабораторної роботи, оформлений згідно з ДСТУ 3008:2015 “Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання”, та зі змістом, визначеним цим порядком виконання лабораторної роботи .

18. Представити до захисту звіт з виконаної лабораторної роботи і проект у Git-репозиторії https://github.com/odorenskyi/student\_name.

10.1

Лістинг:

void count\_symb(const string& infilename, const string& outfilename1) {

system("chcp 1251 & cls");

ifstream in(infilename);

if (!in.is\_open()) {

cerr << "Не вдалося відкрити вхідний файл: " << infilename << endl;

return;

}

ofstream out(outfilename1, ios::app);

if (!out.is\_open()) {

cerr << "Не вдалося відкрити вихідний файл: " << outfilename1 << endl;

return;

}

string text;

char ch, check = ':';

int count = 0;

while (in.get(ch)) {

if (ch != 'А' && ch != 'а' && ch != 'О' && ch != 'о' && ch != 'Е' && ch != 'е' && ch != 'Ї' && ch != 'ї' && ch != 'Є' && ch != 'є' && ch != 'У' && ch != 'у' && ch != 'И' && ch != 'и' && ch != 'І' && ch != 'і' && ch != 'Я' && ch != 'я') {

text += ch;

}

if (check == ch) {

count++;

}

}

out << "------------------------------------------------------\n"

<< "| Tsarenko Stas, CUNTU, Ukraine, Kropyvnytskyi, 2024 |\n"

<< "------------------ © All rights reserved -------------\n"

<< "Кількість символів ( : ) = " << count << "\nТекст без голосних = " << text << endl;

}

10.2

Лістинг:

void transliteration(const string& inputFileName) {

system("chcp 1251 & cls");

string text;

char ch;

time\_t now = time(nullptr);

tm current\_time;

localtime\_s(&current\_time, &now);

ifstream in(inputFileName);

if (in.is\_open()) {

while (in.get(ch))

text += ch;

}

in.close();

unordered\_map<char, string> translit\_map = {

{'А', "A"}, {'а', "a"},

{'Б', "B"}, {'б', "b"},

{'В', "V"}, {'в', "v"},

{'Г', "H"}, {'г', "h"},

{'Ґ', "G"}, {'ґ', "g"},

{'Д', "D"}, {'д', "d"},

{'Е', "E"}, {'е', "e"},

{'Є', "Ye"}, {'є', "ie"},

{'Ж', "Zh"}, {'ж', "zh"},

{'З', "Z"}, {'з', "z"},

{'И', "Y"}, {'и', "y"},

{'І', "I"}, {'і', "i"},

{'Ї', "Yi"}, {'ї', "i"},

{'Й', "Y"}, {'й', "i"},

{'К', "K"}, {'к', "k"},

{'Л', "L"}, {'л', "l"},

{'М', "M"}, {'м', "m"},

{'Н', "N"}, {'н', "n"},

{'О', "O"}, {'о', "o"},

{'П', "P"}, {'п', "p"},

{'Р', "R"}, {'р', "r"},

{'С', "S"}, {'с', "s"},

{'Т', "T"}, {'т', "t"},

{'У', "U"}, {'у', "u"},

{'Ф', "F"}, {'ф', "f"},

{'Х', "Kh"}, {'х', "kh"},

{'Ц', "Ts"}, {'ц', "ts"},

{'Ч', "Ch"}, {'ч', "ch"},

{'Ш', "Sh"}, {'ш', "sh"},

{'Щ', "Shch"}, {'щ', "shch"},

{'Ь', ""}, {'ь', ""},

{'Ю', "Yu"}, {'ю', "iu"},

{'Я', "Ya"}, {'я', "ia"}

};

string transliterated\_text;

for (char ch : text) {

auto it = translit\_map.find(ch);

if (it != translit\_map.end()) {

transliterated\_text += it->second;

}

else {

transliterated\_text += ch;

}

}

ofstream out(inputFileName, ios::app);

if (out.is\_open()) {

out << "\nТекст після транслітерації: " << transliterated\_text << "\nДата дозапису: " << current\_time.tm\_mday << "/" << (current\_time.tm\_mon + 1) << "/" << (current\_time.tm\_year + 1900) << endl;

}

out.close();

}

10.3  
Лістинг:

void calc(int x, int y, int z, int b, const string& outputFileName) {

system("chcp 1251 & cls");

double result;

result = s\_calculation(x, y, z);

string binary = "";

while (b > 0) {

binary = to\_string(b % 2) + binary;

b /= 2;

}

ofstream out(outputFileName, ios::app);

if (out.is\_open()) {

out << "Результат s\_calculation: " << result << "\nПеретворення b в двійковий код: " << binary << endl;

} else {

cerr << "Не вдалося відкрити вихідний файл: " << outputFileName << endl;

}

out.close();

}

Лістинг TestDriver:

#include <iostream>

#include"ModulesTsarenko.h"

using namespace std;

int main()

{

count\_symb("input.txt", "output\_10\_1.txt");

transliteration("input.txt");

calc(2, 3, 4, 5, "output\_10\_2.txt");

}

**Контрольні запитання і завдання**

1. Яке призначення та синтаксис запису блоку-контроля try - throw - catch у мові програмування С++?

У мові програмування C++, блоки try-catch використовуються для обробки винятків.

try {

Код, в якому може виникнути виняток

}

catch (ExceptionType1 e1) {

Обробка винятку типу ExceptionType1

}

catch (ExceptionType2 e2) {

Обробка винятку типу ExceptionType2

}

1. Наведіть приклад опису й використання міжмодульної змінної.

int globalVariable; - оголошення міжмодульної змінної

void setGlobalVariable(int value) {

globalVariable = value;

}

1. Яку область видимості матимуть об’єкти (змінні, типи, константи тощо), описані в тілі функції main С++?

Область видимості об'єктів (змінних, типів, констант і т. д.), описаних у тілі функції main у C++, обмежена самою функцією main. Це означає, що об'єкти, оголошені всередині main, будуть доступні тільки всередині цієї функції.

1. Здійсніть порівняльний аналіз змінної типу enum та масиву.

enum (перерахування) визначає набір іменованих константних цілих значень. Вони зазвичай використовуються для позначення набору можливих варіантів для одного концепту.

enum Days { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY };

Days today = MONDAY;

Масив є упорядкованою колекцією одного типу даних. Елементи масиву можна ідентифікувати за допомогою їх індексів.

int numbers[5] = {1, 2, 3, 4, 5};

int x = numbers[2]; - x буде 3

Основні відмінності:

* enum використовується для набору іменованих констант, тоді як масив використовується для зберігання даних одного типу.
* У enum кожному значенню призначається ціле число за замовчуванням (починаючи з 0), тоді як у масиву елементи індексуються цілими числами.
* enum дозволяє легше читати та зрозуміліші програми, коли є обмежене число варіантів, а масиви частіше використовуються для зберігання інформації великих масивів даних.

**Висновок**У ході виконання лабораторної роботи № 10 “” Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням ” треба Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище лабораторних робіт №8–9) та реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.

Виконання цієї роботи дозволило краще розуміти архітектуру програмного забезпечення, зокрема, створення модулів, їх збірку у бібліотеки та тестування.  
Використання Git для управління версіями програмного коду є важливою навичкою, яка дозволяє ефективно співпрацювати над проектами та зберігати їх історію.   
У цілому, ця лабораторна робота допомогла розвинути свої навички у програмуванні та розумінні процесу розробки програмного забезпечення з використанням файлового введення/виведення та модульного тестування.