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

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

ВИКОНАЛА
студентка академічної групи
КІ-22-2
Тітарова А. А.

ПЕРЕВІРИВ
викладач
кафедри кібербезпеки
та програмного забезпечення
Собінов Олександр
Георгійович

Кропивницький – 2023

ТЕМА: РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕННЯМ

Мета роботи полягає у набутті грунтовних вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування C++ 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.

Варіант №18

https://github.com/odorenskyi/

ВАРІАНТ 18

—— *ВХІДНИЙ ТЕКСТ - ВМІСТ ВХІДНОГО ТЕКСТОВОГО ФАЙЛУ* ——

Довільне слово українською мовою.

—— *ЗАДАЧА 10.1* ——

У ***вихідний*** текстовий файл записати:

- авторську інформацію: ім'я й прізвище розробника модуля, установа/організація, місто, країна, рік розробки;
- кількість приголосних літер у слові із вхідного файла;
- повідомлення, чи є слово із вхідного файла у наступній краплинці Віталія Іващенка:

Про себе не кажи недобрих слів,
Бо має сказане таємну силу.
Кажи: «Я сильний, впевнений, щасливий!»
І буде саме так, як ти хотів!

—— *ЗАДАЧА 10.2* ——

У ***вхідний*** текстовий файл дописати:

- кількість символів у файлі та дату дозапису інформації.

—— *ЗАДАЧА 10.3* ——

Вхідні дані – числові значення x, y, z та натуральне число b. У ***вихідний*** текстовий файл дописати:

- результати виконання функцій із заголовкового файлу Modules*Прізвище*.h `s_calculation` з аргументами x, y, z;
- число b у двійковому коді.

- Мова повідомлень – українська (*наприклад*, якщо у вихідний файл записується кількість символів у вхідному файлі, то модуль повинен сформувати й записати/дописати повноцінне речення: "У файлі `ВхФайл.txt` міститься 257 символів.").
- Вхідний файл `*.txt` створюється користувачем, у який за допомогою текстового редактора (у Windows – Блокнот) записується вхідний текст відповідно до завдання; вихідний файл створюється програмним модулем; імена вхідного й вихідного файлів є параметрами відповідного модуля.
- Перед читання/записом з/у файловий потік слід реалізувати перевірку його відкриття; після завершення – закрити всі відкриті файлові потоки.
- Оброблення текстових файлів рекомендовано реалізувати за допомогою файлових потоків `ofstream` та `ifstream <fstream>` C++.
- Для отримання локальної дати й часу ОС можна використати стандартні функції `time`, `ctime`, `localtime`, `asctime`, реалізовані у `ctime` / `time.h`.

Завдання 1

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>
using namespace std;

int countConsonants(const string& word) {
    int count = 0;
    string vowels = "aeiouAEIOU";
    for (char c : word) {
        if (isalpha(c) && vowels.find(c) == string::npos) {
            count++;
        }
    }
    return count;
}

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    string inputFile = "input.txt";
    string outputFile = "output.txt";
    string name = "Анастасія Тітарова";
    string city = "Кропивницький";
    string country = "Україна";
    string university = "ЦНТУ";
    int year = 2023;
```

```
    ifstream input(inputFile);
    ofstream output(outputFile, ios::app);
    if (input.is_open() && output.is_open()) {
        string inputText;
        getline(input, inputText);


        output << "Автор: " << name << endl;
        output << "Місто: " << city << endl;
        output << "Країна: " << country << endl;
        output << "ЦНТУ: " << university << endl;
        output << "Рік розробки: " << year << endl;


        int consonantCount = countConsonants(inputText);
        output << "Кількість приголосних літер у слові: " << consonantCount << endl;


        string vitaliyText = "Про себе не кажи недобрих слів, Бо має сказане таємну силу. Кажи: «Я сильний, впевнений, щасливий!» І буде саме так, як ти хотів!";
        if (vitaliyText.find(inputText) != string::npos) {
            output << "Слово є у краплинці Віталія Іващенка." << endl;
        } else {
            output << "Слово не знайдено у краплинці Віталія Іващенка." << endl;
        }


        input.close();
        output.close();
        cout << "Запис успішно виконано." << endl;
    } else {
        cout << "Помилка при відкритті файлів." << endl;
    }
```

```
    return 0;
}
```

Завдання 2

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <windows.h>
using namespace std;

int main() {
    SetConsoleOutputCP(1251)
    SetConsoleCP(1251)
    string inputFile = "input.txt";
    string outputFile = "output.txt";

    ifstream input(inputFile);
    ofstream output(outputFile, ios::app);
    if (input.is_open() && output.is_open()) {
        string inputText;
        getline(input, inputText);

        output << "Кількість символів у файлі: " << inputText.length() << endl;

        time_t now = time(0);
        string date = ctime(&now);
        output << "Дата дозапису: " << date;

        input.close();
```

```
        output.close();
        cout << "Дозапис успішно виконано." << endl;
    } else {
        cout << "Помилка при відкритті файлів." << endl;
    }

    return 0;
}
```

Завдання 3

```
#include <iostream>
#include <fstream>
#include <windows.h>
#include <string>
#include "ModulesTitarova.h" // Підключення заголовкового файлу з функціями

using namespace std;

int main() {
    SetConsoleOutputCP(1251)
    SetConsoleCP(1251)
    string outputFile = "output.txt";

    double x, y, z;
    int b;
    // Отримання вхідних даних x, y, z, b...

    ofstream output(outputFile, ios::app);
    if (output.is_open()) {
```

```
        output << "Результати виконання функцій:" << endl;
        output << "s_calculation(x, y, z) = " << s_calculation(x, y, z) << endl;
        output << "s_calculation(x, y) = " << s_calculation(x, y) << endl;
        output << "s_calculation(x, b) = " << s_calculation(x, b) << endl;

        output << "Число b у двійковому коді: " << decToBinary(b) << endl;

        output.close();
        cout << "Дозапис успішно виконано." << endl;
    } else {
        cout << "Помилка при відкритті файлу." << endl;
    }

    return 0;
}
```

ModulesTitarova.h

```
#ifndef MODULESTITAROVA_H
#define MODULESTITAROVA_H

double s_calculation(double x, double y, double z) {
}

std::string decToBinary(int number) {
}

#endif // MODULESTITAROVA_H
```

ModulesTitarova.cpp

```
#include "ModulesTitarova.h"

double s_calculation(double x, double y, double z) {

std::string decToBinary(int number) {
}
```

ВІДПОВІДІ НА КОНТРОЛЬНІ ЗАПИТАННЯ

1. Призначення та синтаксис блоку-контроля try-throw-catch у мові програмування C++:

- Призначення: Блок try-throw-catch використовується для обробки винятків (exceptions) у програмі. Він дозволяє визначити блок коду, в якому можуть виникати виключні ситуації, та обробити їх у відповідному блоку catch.

- Синтаксис:

```cpp
try {
    // Блок коду, в якому можуть виникати виключні ситуації
}
catch (тип_виключення1 об'єкт1) {
    // Обробка виключення типу_виключення1
}
catch (тип_виключення2 об'єкт2) {
    // Обробка виключення типу_виключення2
}
// ...
catch (тип_виключенняN об'єктN) {
    // Обробка виключення типу_виключенняN
}
```

- У блоку try розміщується код, який може викликати виключні ситуації. Якщо виникає виключна ситуація, вона "кидається" (throw) за допомогою оператора throw.

- В блоках catch вказуються типи виключень, які можуть бути оброблені. Коли відбувається виняток, система спробує знайти відповідний блок catch для обробки цього виключення. Якщо відповідний блок catch знайдений, виконується код у цьому блоку, а потім програма продовжує виконання після блоку try-catch.

2. Приклад опису й використання міжмодульної змінної:

В одному модулі (файлі) змінна оголошується з ключовим словом `extern`, а в іншому модулі змінна визначається без ключового слова `extern`. Наприклад:

У файлі "Module1.cpp":

```cpp
// Module1.cpp
extern int globalVariable;  // Оголошення міжмодульної змінної

void function1() {
    globalVariable = 10;  // Вик
```

ористання міжмодульної змінної

```
}
```

У файлі "Module2.cpp":

```cpp
// Module2.cpp
int globalVariable;  // Визначення міжмодульної змінної

void function2() {
    int value = globalVariable;  // Використання міжмодульної змінної
```

```
    // Інші дії зі змінною
}
```

У даному прикладі `globalVariable` є міжмодульною змінною. Вона оголошується з ключовим словом `extern` у файлі "Module1.cpp", а потім визначається без ключового слова `extern` у файлі "Module2.cpp". Таким чином, змінна стає доступною для використання в обох модулях.

3. Об'єкти (змінні, типи, константи тощо), описані в тілі функції `main` в C++, матимуть локальну область видимості. Це означає, що їх можна використовувати лише всередині функції `main`. Наприклад:

```cpp
#include <iostream>

int main() {
    int localVariable = 10;  // Локальна змінна

    std::cout << localVariable << std::endl;  // Використання локальної змінної

    return 0;
}
```

У даному прикладі змінна `localVariable` є локальною для функції `main` і доступна лише всередині цієї функції.

4. Порівняльний аналіз змінної типу enum та масиву:

- Змінна типу enum:

- Визначення: Змінна типу enum (перерахування) є змінною, яка може приймати одне з певного набору значень, заданих у перерахуванні.

- Синтаксис оголошення: `enum EnumName { Value1, Value2, ..., ValueN };`

- Приклад:

```cpp
enum Days { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };

Days currentDay = Tuesday;
```

- Масив:

- Визначення: Масив є упорядкованою послідовністю елементів одного типу, розташованих у пам'яті.

- Синтаксис оголошення:

`тип_елемента ім'я_масиву[розмір];`

- Приклад:

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

Порівняння:

- Тип: Змінна типу enum може приймати одне з обмеженого набору значень, визначених у перерахуванні, тоді як масив може зберігати послідовність елементів одного типу.

- Доступ до елементів: В змінній типу enum доступ до значення здійснюється через ім'я змінної, наприклад `currentDay`, тоді як у масиві доступ до елементів здійснюється за допомогою індексів, наприклад `numbers[0]`.

- Розмір: Змінна типу enum має фіксований розмір, який залежить від розмірності базового цілочисельного типу (зазвичай 4 байти), тоді як розмір масиву може бути визначений під час оголошення або динамічно виділятись у пам'яті.

- Індексація: Змінна типу enum не підтримує індексацію, тоді як масив може бути індексований для доступу до окремих елементів за їхніми індексами.

- Застосування: Змінна типу enum зручно використовується для представлення набору можливих значень (наприклад, днів тижня), тоді як масиви широко використовуються для зберігання та обробки групи елем**ентів одного типу.**