Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра ПЗ

Лабораторна робота №8, варіант № 19
з дисципліни «Основи програмування»

Виконала: ст. 1ПІ-25Б Семенов В.О. .
Перевірив: доцент Решетнік О.О.

Вінниця – 2025

Тема: Файли та робота з ними

Мета: Опанувати введення/виведення даних у файли, робота з текстовими та бінарними файлами.

Завдання 19. Дано текстовий файл F1. Переписати його вміст у файл F2, розбивши на рядки таким чином, щоб кожен рядок, або містив не більше 20 символів, якщо це можливо додати пробіли між словами рядка, щоб текст був вирівняний по ширині рядка.

Декомпозиція завдання

1. Відкриття файлів

- Відкрити вхідний файл для читання.
- Відкрити вихідний файл для запису.
- Перевірити успішність відкриття.

2. Зчитування слів

- Читати текст слово за словом.
- Зберігати слова в тимчасовий масив поточного рядка.
- Підраховувати сумарну довжину слів без пробілів.

3. Формування рядка

- Перевіряти, чи вміститься наступне слово у рядок.
- Якщо слово вміщується – додати його до поточного рядка.
- Якщо слово не вміщується – завершити поточний рядок і почати новий.

4. Вирівнювання рядка по ширині

- Визначити загальну кількість пробілів, яку треба додати, щоб рядок дорівнював максимальній ширині.
- Розділити пробіли на проміжки між словами рівномірно.
- Додати залишок пробілів зліва направо, щоб рядок був точно потрібної довжини.

5. Запис рядка у файл

- Якщо рядок містить одне слово – просто записати його.
- Якщо рядок довший за ширину або вирівнювання не потрібно – з’єднати слова одним пробілом.
- Якщо рядок коротший – додати пробіли для вирівнювання.

6. Обробка довгих слів

- Якщо слово довше максимальної ширини, розбити його на частини.
- Записати частини окремими рядками.

7. Запис останнього рядка

- Після завершення зчитування файлу перевірити, чи залишилися слова в тимчасовому масиві.
- Записати їх у файл із вирівнюванням або без нього.

8. Закриття файлів

- Закрити вхідний та вихідний файли після завершення роботи.

Код програми показано на лістингу.

Приклади роботи програми можна побачити на рисунку 1

Лістинг 1 – Програма для роботи з файлами

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <limits>

const int LINE_WIDTH = 20;

std::string justifyLine(const std::vector<std::string>& words,
int wordsLen)
{
    if (words.empty()) return "";
    if (words.size() == 1) return words[0];

    int spacesNeeded = LINE_WIDTH - wordsLen;
    int gaps = static_cast<int>(words.size()) - 1;
    int baseSpaces = spacesNeeded / gaps;
    int extraSpaces = spacesNeeded % gaps;

    std::string line;
    for (size_t i = 0; i < words.size(); ++i)
    {
        line += words[i];
        if (i < static_cast<size_t>(gaps))
        {
```

Продовження лістингу 1

```
            int thisGap = baseSpaces + (extraSpaces-- > 0 ? 1 :
0);
            line.append(thisGap, ' ');
        }
    }
    return line;
}

std::string joinWithSingleSpaces(const std::vector<std::string>&
words)
{
    if (words.empty()) return "";
    std::string line = words[0];
    for (size_t i = 1; i < words.size(); ++i)
    {
        line += ' ';
        line += words[i];
    }
    return line;
}

int main() {
    std::ifstream fin("F1.txt");
    std::ofstream fout("F2.txt");

    if (!fin.is_open() || !fout.is_open()) {
        std::cerr << "File open error\n";
        return 1;
    }

    std::string word;
    std::vector<std::string> words;
    int wordsLen = 0;
    int currentLineLen = 0;

    while (fin >> word)
    {
        if (words.empty())
        {
            words.push_back(word);
            wordsLen = static_cast<int>(word.length());
            currentLineLen = wordsLen;
        }
        else
        {
            if (currentLineLen + 1 +
static_cast<int>(word.length()) <= LINE_WIDTH)
            {
                words.push_back(word);
                wordsLen += static_cast<int>(word.length());
```

Продовження лістингу 1

```
                currentLineLen += 1 +
static_cast<int>(word.length());
            }
            else
            {
                if (words.size() == 1)
                {
                    fout << words[0] << '\n';
                }
                else
                {
                    if (wordsLen < LINE_WIDTH) fout <<
justifyLine(words, wordsLen) << '\n';
                    else fout << joinWithSingleSpaces(words) <<
'\n';
                }
                words.clear();
                words.push_back(word);
                wordsLen = static_cast<int>(word.length());
                currentLineLen = wordsLen;
            }
        }
    }

    if (!words.empty())
    {
        if (words.size() == 1)
        {
            fout << words[0] << '\n';
        }
        else
        {
            if (wordsLen < LINE_WIDTH)
            {
                fout << justifyLine(words, wordsLen) << '\n';
            }
            else
            {
                fout << joinWithSingleSpaces(words) << '\n';

            }
        }
    }

    fin.close();
    fout.close();
    return 0;
}
```

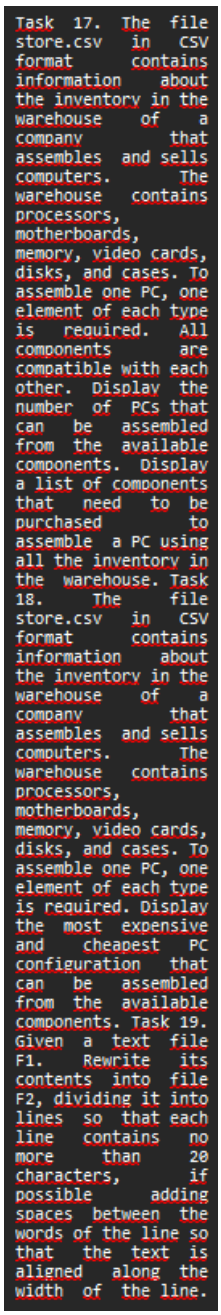

Рисунок 1 – файл F2 після переносу і фрматування

Контрольні запитання:

7. Якими засобами можна реалізувати багато файлову програму?

- Розділити код на кілька файлів (.cpp, .h).
- Використовувати заголовочні файли для оголошень.
- Застосовувати компілятор для окремої компіляції файлів і лінкер для об’єднання.

8. Поясніть реалізацію багато файлової програми за допомогою директиви #include.

- Заголовочний файл (.h) містить оголошення функцій і структур.
- У файлі .cpp підключаємо заголовочний через #include.
- Це дозволяє використовувати функції та структури з інших файлів без повторного оголошення.

9. Поясніть процес формування і виконання файлу проекта.

- Компіляція: кожен файл .cpp перетворюється у об’єктний файл (.obj).
- Лінкування: об’єктні файли об’єднуються в один виконуваний файл (.exe).
- Виконання: операційна система завантажує .exe у пам’ять і запускає програму.

Висновок: Під час роботи з файлами студент освоїв способи введення та виведення даних у текстові та бінарні файли, навчились відкривати, читати, записувати та закривати файли, а також організовувати збереження та обробку інформації в зовнішніх джерелах.