College of Computing

Université Mohammed VI Polytechnique

TP5 - Manipulation avancée des chaînes de caractères : String

Imad Kissami

16 Février 2025

Instructions

- Créer un `Makefile` pour compiler tous les exercices
- Chaque exercice devrait être sauvegarder dans exonumero_exercice.cpp
- L'exécution se fait avec : `./exo1` (pour l'exercice 1).
- Tous les fichiers doivent être regroupés dans un dossier **TP5_Nom_Prénom**.

Exercice 1 : Détection de palindrome (`std::string`, `std::reverse`)

Objectif :

- Vérifier si une chaîne est un **palindrome** (se lit de la même façon dans les deux sens).
- Ignorer la **casse** et les **espaces**.

Code attendu :

```
bool is_palindrome(const std::string& text);
```

Entrée :

```
Entrez un mot : "Engage le jeu que je le gagne"
```

Sortie attendue :

```
"Engage le jeu que je le gagne" est un palindrome.
```

Exercice 2 : Fréquence des mots dans un texte (`std::unordered_map`, `std::istringstream`)

Objectif :

- Lire un texte et compter la fréquence des mots avec `std::unordered_map` :
  - Example : `std::unordered_map<std::string, int> freq;`

Code attendu :

```
void word_frequency(const std::string& text);
```

Entrée :

```
Entrez une phrase : "C++ est puissant est rapide est efficace"
```

Sortie attendue :

```
Fréquence des mots :
C++ -> 1
est -> 3
puissant -> 1
rapide -> 1
efficace -> 1
```

Exercice 3 : Compression de texte (Encodage RLE)

Objectif :

— Implémenter **Run-Length Encoding (RLE)**, une compression simple remplaçant les séquences répétées par un nombre.

Code attendu :

```cpp
std::string encode_rle(const std::string& text);
```

Entrée :

```
Texte : "aaabbccccddee"
```

Sortie attendue :

```
Encodé : "a3b2c4d2e2"
```

Exercice 4 : Extraction des mots les plus fréquents (`std::map`)

Objectif :

— Trouver les **3 mots les plus fréquents** dans un texte.

— Use this vector `sorted_words`

```cpp
std::vector<std::pair<std::string, int>> sorted_words(freq.begin(), freq.end());
std::sort(sorted_words.begin(), sorted_words.end(),
        [](const auto& a, const auto& b) { return a.second > b.second; });
```

Code attendu :

```cpp
void top_frequent_words(const std::string& text, int n);
```

Entrée :

```
Texte : "C++ est rapide, C++ est puissant, C++ est utilisé"
```

Sortie attendue :

```
Top 3 mots les plus fréquents :
1. C++ -> 3
2. est -> 3
3. rapide -> 1
```

Exercice 5 : Trie des mots par longueur (`std::multimap`)

Objectif :

— Trier les mots d'une phrase **par longueur croissante** avec `std::multimap`.

```cpp
std::multimap<int, std::string> word_map;
```

Code attendu :

```cpp
void sort_by_length(const std::string& text);
```

Entrée :

```
Phrase : "Le langage C++ est rapide et efficace"
```

Sortie attendue :

```
Trie par longueur : Le C++ et est rapide efficace langage
```

Exercice 6 : Recherche avancée avec expressions régulières (`std::regex`)

Objectif :

— Vérifier si une phrase **contient une adresse email**.
— Utiliser **`std::regex`** pour la validation.

Code attendu :

```cpp
bool contains_email(const std::string& text);
```

Entrée :

```
Phrase : "Mon email est user@example.com"
```

Sortie attendue :

```
Adresse email détectée : user@example.com
```