Analyse et Explication de Scripts C++

Aouanet Bahaeddine
Filière: 2 Génie Informatique

January 19, 2025

Contents

1 Introduction

Ce document présente une analyse détaillée de plusieurs scripts C++ couvrant divers concepts, notamment la recherche de motifs dans une chaîne, la reconnaissance d'entiers, d'identificateurs et d'autres structures syntaxiques. Chaque script est accompagné d'une explication détaillée pour en faciliter la compréhension.

2 Recherche de Motif dans une Chaîne

2.1 Code

```cpp
#include <iostream>
#include <string>
#include <vector>

std::vector<int> Recherche_motif(const std::string& motif, const std::string& texte) {
    std::vector<int> positions;
    size_t pos = texte.find(motif, 0); // Cherche le motif    partir du d but
    while (pos != std::string::npos) {
        positions.push_back(pos); // Ajoute la position trouv e
        pos = texte.find(motif, pos + 1); // Continue la recherche apr s la position actuelle
    }
    return positions;
}

int main() {
    std::string texte = "if x>2 : if x<5 : print(x) else: print(x-2)";
    std::string motif = "if";

    std::vector<int> positions = Recherche_motif(motif, texte);
    std::cout << "Positions du motif \"" << motif << "\": ";
    for (int pos : positions) {
        std::cout << pos << " ";
    }
    return 0;
}
```

Listing 1: Recherche de motif

2.2 Explication

Ce script utilise la fonction `std::string::find` pour rechercher un motif donné dans une chaîne de texte. Les positions trouvées sont stockées dans un vecteur, qui est ensuite affiché. Il permet d'explorer les occurrences multiples d'un motif dans une chaîne.

3 Vérification de Conformité de Chaînes

3.1 Script 1 : Appartenance à un Motif Spécifique

```cpp
#include <iostream>
#include <string>

bool Appartient(const std::string& u) {
    int n = u.size();
    if (n < 4) // La cha ne doit contenir au moins "abab"
        return false;

    int i = 0;

    // V rifier les 'a' au d but
    while (i < n && u[i] == 'a') {
        i++;
    }

    // Il doit y avoir au moins un 'a'
    if (i == 0)
        return false;

    // V rifier 'b'
    if (i < n && u[i] == 'b') {
        i++;
    } else {
        return false;
    }

    // V rifier 'a'
    if (i < n && u[i] == 'a') {
        i++;
    } else {
        return false;
    }

    // V rifier les 'b' restants
    while (i < n && u[i] == 'b') {
        i++;
    }

    // Il doit y avoir au moins un 'b' apr s 'a'
    if (i == n - 1 && u[n - 1] == 'b') {
        return true;
    }

    return (i == n);
}

int main() {
    std::string mot1 = "aaababb";
    std::string mot2 = "aabbb";

    std::cout << "Appartient(\"" << mot1 << "\"): " << (Appartient(mot1
    ) ? "True" : "False") << std::endl;
    std::cout << "Appartient(\"" << mot2 << "\"): " << (Appartient(mot2
    ) ? "True" : "False") << std::endl;

    return 0;
```

```
}
```

Listing 2: Appartenance à un motif spécifique

3.2 Explication

Ce script vérifie si une chaîne suit un motif spécifique composé de caractères `a` et `b`. Les transitions sont basées sur une séquence définie où les `a` et `b` apparaissent selon des règles bien précises.

4 Reconnaissance d'Entiers et Réels

4.1 Script 2 : Reconnaissance d'Entiers

```cpp
#include <iostream>
#include <string>

bool ReconnaitreEntier(const std::string& chaine) {
    if (chaine.empty()) // Une cha ne vide n'est pas un entier
        return false;

    int i = 0;

    // V rifier le signe (+ ou -)
    if (chaine[0] == '+' || chaine[0] == '-') {
        i++;
    }

    // V rifier que les caract res restants sont des chiffres
    for (; i < chaine.size(); i++) {
        if (!isdigit(chaine[i])) {
            return false;
        }
    }

    return true;
}

int main() {
    std::string tests[] = {"123", "-456", "12a3", "+789", "-", ""};
    for (const auto& test : tests) {
        std::cout << "Reconna treEntier(\"" << test << "\"): "
                  << (ReconnaitreEntier(test) ? "True" : "False") <<
   std::endl;
    }

    return 0;
}
```

Listing 3: Reconnaissance d'entiers

4.2 Explication

Ce programme identifie si une chaîne est un entier valide en vérifiant la présence optionnelle d'un signe au début, suivi exclusivement de chiffres.

5 Reconnaissance d'Identificateurs

5.1 Script 3 : Vérification d'Identificateurs

```cpp
#include <iostream>
#include <string>

bool ReconnaitreIdentificateur(const std::string& chaine) {
    int n = chaine.size();
    if (n == 0) // Une cha ne vide n'est pas un identificateur valide
        return false;

    // V rifier le premier caract re : il doit  tre  une lettre ou un underscore
    if (!((chaine[0] >= 'a' && chaine[0] <= 'z') ||
          (chaine[0] >= 'A' && chaine[0] <= 'Z') ||
          (chaine[0] == '_'))) {
        return false;
    }

    // V rifier les caract res restants : ils doivent  tre  des lettres, des chiffres ou des underscores
    for (int i = 1; i < n; i++) {
        if (!((chaine[i] >= 'a' && chaine[i] <= 'z') ||
              (chaine[i] >= 'A' && chaine[i] <= 'Z') ||
              (chaine[i] >= '0' && chaine[i] <= '9') ||
              (chaine[i] == '_'))) {
            return false;
        }
    }

    return true;
}

int main() {
    std::string tests[] = {"_valid", "2invalid", "also_valid", "not-valid"};
    for (const auto& test : tests) {
        std::cout << "ReconnaitreIdentificateur(\"" << test << "\"): "
                  << (ReconnaitreIdentificateur(test) ? "True" : "False") << std::endl;
    }

    return 0;
}
```

Listing 4: Reconnaissance d'identificateurs

5.2 Explication

Ce programme détermine si une chaîne est un identificateur valide. Les identificateurs doivent commencer par une lettre ou un underscore et peuvent inclure des lettres, des chiffres et des underscores dans les positions suivantes.