Міністерство освіти і науки України ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА Факультет прикладної математики та інформатики

Кафедра програмування

ЛАБОРАТОРНА РОБОТА № 8 **Алгоритм Маркова**

з курсу "Алгоритми та структури даних"

Виконав:

Студент групи ПМІ-16

Бевз Маркіян Юрійович

Мета: Ознайомлення з принципом роботи алгоритму Маркова та його застосуванням у заміні символів у рядку згідно заданих правил.

Принцип роботи алгоритму Маркова:

- 1. **Запис правил:** Правила для заміни символів у рядку задаються у вигляді пар (ліва частина, права частина), де ліва частина символ або послідовність символів, які потрібно замінити, а права частина символ або послідовність символів, на які потрібно замінити.
- 2. Застосування правил: Програма зчитує вхідний рядок та послідовно застосовує задані правила до символів у цьому рядку.
- 3. **Заміна символів:** Коли зустрічається символ, який відповідає лівій частині якого-небудь правила, він замінюється на праву частину цього правила.
- 4. **Повторення:** Процес заміни символів повторюється, поки в рядку залишаються символи, які можна замінити згідно заданих правил.
- 5. Вихід: Замінений рядок повертається як результат роботи алгоритму.

Цей принцип дозволяє ефективно заміняти символи у рядку з використанням заданих правил, що може бути корисним у різних областях програмування.

Принциповою частиною реалізації алгоритму Маркова ϵ функція **apply_rules**, яка застосовує задані правила до вхідного рядка. Цей підхід дозволяє автоматизувати процес заміни символів у великих текстових даних згідно заданих правил.

Хід роботи: Після вивчення принципу роботи алгоритму Маркова я реалізував структуру правил та саму функцію застосування. Також було створено 4 тести для перевірки правильності роботи функції для різних випадків. Нижче буде прикріплено результат виконання програми, тестів, та їх код.

```
MyString apply_rules(const MyString& input, const vector<Rule>& rules)
{
    MyString output = input;
    bool rule_applied = true;

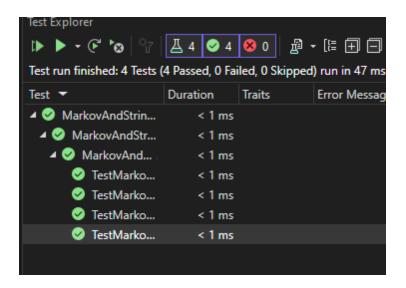
    while (rule_applied) {
        rule_applied = false;

        for (const auto& rule : rules) {
            size_t pos = output.find(rule.left);
            if (pos != string::npos) {
                  output.replace(pos, rule.left.length(), rule.right);
                  rule_applied = true;
                  break;
        }
    }
}
return output;
```

```
Original string: |||||-||
Result after applying Markov algorithm: |||-
Original string: abcirkrtc
Result after applying Markov algorithm: I love you
Original string: |||||+||
Result after applying Markov algorithm: ||||||
```

```
TEST_CLASS(MarkovAndStringTest)
public:
   TEST_METHOD(TestMarkovAlgorithm)
       // Тести для алгоритму Маркова
       vector<Rule> rules = {
           {"+|", "|"},
           // Додаткові правила
       };
       string input("|||+||");
       string result = apply_rules(input, rules);
       Assert::AreEqual(string("||||"), result);
   TEST_METHOD(TestMarkovAlgorithm2)
       vector<Rule> rules = {
           {"|-|", "-"},
           // Додаткові правила
       };
       string input("||-||");
       string result = apply_rules(input, rules);
       Assert::AreEqual(string("|-"), result);
```

```
TEST_METHOD(TestMarkovAlgorithmMultRul)
    vector<Rule> rules = {
        {"abc", "I "},
        {"irk", "love "},
        {"rtc", "you"}
       // Додаткові правила
    };
    string input("abcirkrtc");
    string result = apply_rules(input, rules);
    Assert::AreEqual(string("I love you"), result);
TEST_METHOD(TestMarkovAlgorithmNoRuleApplied)
    // Тест для алгоритму Маркова, коли жодне правило не застосовано
    vector<Rule> rules = {
        {"+|", "|+"},
{"-|", " "},
        {"abc", "def"},
    };
    string input("xyz");
    string result = apply_rules(input, rules);
    Assert::AreEqual(string("xyz"), result); // Очікується, що вихід буде таким же
```



Висновок: на цій лабораторні роботі я ознайомився з принципом роботи алгоритму Маркова та його застосуванням у заміні символів у рядку згідно заданих правил.