Министерство высшего образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

**Лабораторная работа № 1.1**

**«Нормальный алгоритм Маркова»**

Выполнил студент гр. РИС-24-2б

Меньшиков Арсений Андреевич

Проверил:

Доц. каф. ИТАС

Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Пермь, 2024

**1. Удалить первый символ слова**

A={a,b}. Удалить из непустого слова P его первый символ. Пустое слово не менять.

**Анализ**

* Модель алгоритма Маркова преобразует исходные слова с помощью заданных правил, которые могут заменять часть исходной строки.
* Слово – это строка из символов.
* Кроме правил в алгоритмах Маркова используют приоритеты их выполнения:

1. Правило проходит первое своё вхождение в исходной строке и заменяет его на правую часть(замена);
2. Правила выполняются по порядку их номеров;
3. Если правило не применяется к исходной строке, то алгоритм переходит к проверке следующего по порядку правила;
4. Если правило выполнено, то алгоритм возвращается к проверке слова, начиная с начала списка;
5. Алгоритм завершается, если ни одно правило неприменимо.

* Виды подстановок:

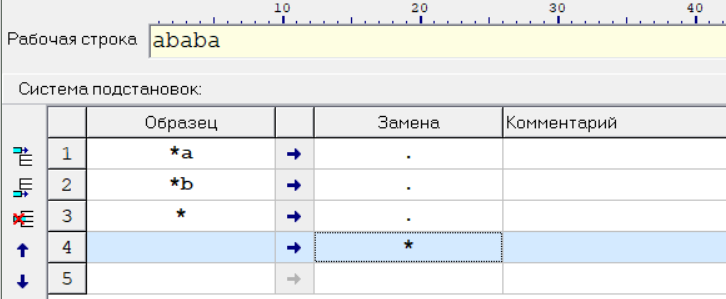
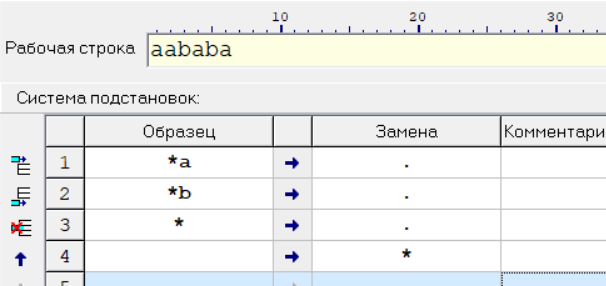
1. Нетерминальные – подстановка, которая позволяет выполнение других правил, после выполнения данной;
2. Терминальная – если подстановка терминальная, то применяется замена и программа принудительно завершается.

* Если правая часть(замена) правила пустая строка, то левая часть правила исчезает из строки.
* Если в левой части правила стоит пустая строка, то в начало слова добавляется правая часть(замена) этого правила.

**Алгоритм**

1. \*a |→ (Удаляем 1 символ)
2. \*b |→ (Удаляем 1 символ)
3. \* |→ (Чтобы в пустом слове алгоритм не зациклился, удаляем \*)
4. → \* (Перед первым символом ставим \*, чтобы пометить 1 символ)

Скриншот программы



**2. Удаление и замена символов в слове**

A={a,b,c,d}. В слове P требуется удалить все вхождения символа c, а затем заменить первое вхождение подслова bb на ddd.

Например, abbcabbca → adddabba

**Анализ**

* Модель алгоритма Маркова преобразует исходные слова с помощью заданных правил, которые могут заменять часть исходной строки.
* Слово – это строка из символов.
* Кроме правил в алгоритмах Маркова используют приоритеты их выполнения:

1. Правило проходит первое своё вхождение в исходной строке и заменяет его на правую часть(замена);
2. Правила выполняются по порядку их номеров;
3. Если правило не применяется к исходной строке, то алгоритм переходит к проверке следующего по порядку правила;
4. Если правило выполнено, то алгоритм возвращается к проверке слова, начиная с начала списка;
5. Алгоритм завершается, если ни одно правило неприменимо.

* Виды подстановок:

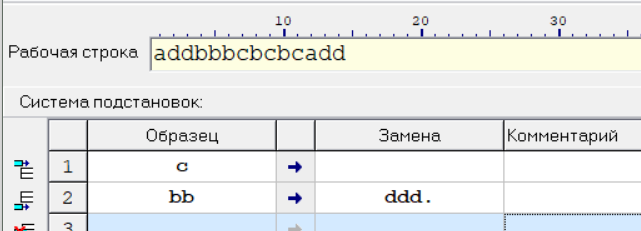
1. Нетерминальные – подстановка, которая позволяет выполнение других правил, после выполнения данной;
2. Терминальная – если подстановка терминальная, то применяется замена и программа принудительно завершается.

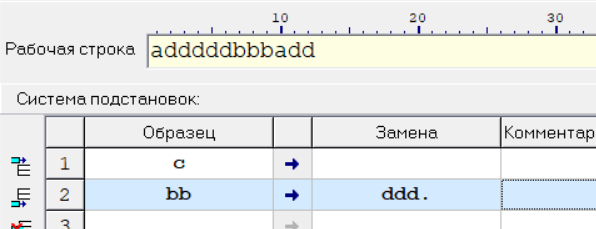
* Если правая часть(замена) правила пустая строка, то левая часть правила исчезает из строки.
* Если в левой части правила стоит пустая строка, то в начало слова добавляется правая часть(замена) этого правила.

**Алгоритм**

1. c → (Удаляем символ c)
2. bb |→ ddd (Заменяем первое вхождение bb на ddd)

Скриншот программы





**3. Приписать символ a к концу слова**

A={a,b}. Требуется приписать символ a к концу слову P. Например, bbab → bbaba.

**Анализ**

* Модель алгоритма Маркова преобразует исходные слова с помощью заданных правил, которые могут заменять часть исходной строки.
* Слово – это строка из символов.
* Кроме правил в алгоритмах Маркова используют приоритеты их выполнения:

1. Правило проходит первое своё вхождение в исходной строке и заменяет его на правую часть(замена);
2. Правила выполняются по порядку их номеров;
3. Если правило не применяется к исходной строке, то алгоритм переходит к проверке следующего по порядку правила;
4. Если правило выполнено, то алгоритм возвращается к проверке слова, начиная с начала списка;
5. Алгоритм завершается, если ни одно правило неприменимо.

* Виды подстановок:

1. Нетерминальные – подстановка, которая позволяет выполнение других правил, после выполнения данной;
2. Терминальная – если подстановка терминальная, то применяется замена и программа принудительно завершается.

* Если правая часть(замена) правила пустая строка, то левая часть правила исчезает из строки.
* Если в левой части правила стоит пустая строка, то в начало слова добавляется правая часть(замена) этого правила.

**Алгоритм**

1. \*a → a\* (Переставляем символ и \* местами, чтобы \* стала ближе к концу слова)
2. \*b → b\* (Переставляем символ и \* местами, чтобы \* стала ближе к концу слова)
3. \* |→ a (Когда звезда в конце, заменяем её на a)
4. → \* (Поставим звезду в начале, чтобы дойти до последнего символа)

Скриншот программы

