**Вариант-1**

Вход: , , , // грамматика, распознающая любое слово длины

Выход: если распознает слово , и иначе

**МТ решатель**

М-1. Проверить, что КС-грамматика находится в НФХ

М-2. Если нет, то привести к НФХ

М-3. Алгоритм 1

**Алгоритм 1 (динамическое программирование)**

1. Если //

2. Если содержит правило то // КС-грамматика содержит правило для вывода пустой строки

3. Вернуть

4. Конец алгоритма

5. Иначе

6. Вернуть

7. Конец алгоритма

8. Для всех // для всех нетерминальных состояний

9. Для всех от до

10. Если содержит правило то

11.

12. Иначе

13.

14. Для всех от до

15. Для всех от до

16. Для всех от до

17. Для всех вида

18.

19. Для всех

20. Если то

21. Вернуть // КС-грамматика распознает язык

22. Конец алгоритма

23. Вернуть

**Пример 1**

ДКА, распознающий язык :

Изображение выглядит как текст, седзи, часы

Автоматически созданное описание

Соответствующая КС-грамматика:

Соответствующая КС-грамматика в НФХ:

Вход: ,

1.

23.

**Пример 2**

ДКА, распознающий язык :

Изображение выглядит как седзи, часы

Автоматически созданное описание

Соответствующая КС-грамматика:

Соответствующая КС-грамматика в НФХ:

Вход: , ,

1.

8.

9. // 1-й символ в произвольной строке

11.

9. // 2-й символ в произвольной строке

11.

14. // рассматриваем строку из 2-х первых символов

15. // начинаем с 1-го символа

16. // строка разбивается на 1 + 1 символов

17.

18.

23.

**Пример 3**

ДКА, распознающий язык :

Изображение выглядит как текст, седзи, стена, часы

Автоматически созданное описание

Соответствующая КС-грамматика:

Соответствующая КС-грамматика в НФХ:

Вход: , ,

1.

8.

9.

11.

9.

11.

8.

9.

11.

9.

11.

14.

15.

16.

17.

18.

20.

**Пример 4**

ДКА, распознающий язык :

Изображение выглядит как стена, седзи, в помещении, часы

Автоматически созданное описание

КС-грамматика, распознающая язык :

Соответствующая КС-грамматика в НФХ:

Вход: ,

1.

8.

9.

11.

9.

11.

9.

11.

8.

9.

11.

9.

11.

9.

11.

14. // рассматриваем строку из первых 2-х символов

15. // начинаем с 1-го

16. // разбиваем строку на 1 + 1 символа

17.

18.

17.

18.

15. // начинаем со 2-го символа

16. // разбиваем на 1 + 1 подстрок

17.

18.

17.

18.

14. // рассматриваем строку из первых 3-х символов

15. // начинаем с 1-го

16. // разбиваем строку на 1 + 2 символа

17.

18.

17.

18.

16. // разбиваем строку на 2 + 1 символа

17.

18.

17.

18.

20.