Домашнее задание №3.

Проектирование и реализация нормального алгоритма Маркова.

Вариант 3.8

Выполнил: Кравченко Александр Андреевич
КТбо1-6

1. Постановка задачи:

1. Разработать нормальный алгоритм Маркова, удаляющий из слова все последовательности, представляющие собой чередование согласных и гласных букв, начинающиеся и заканчивающиеся согласными.
2. Разработать программу, реализующую данный нормальный алгоритм Маркова.

2. Словесное описание алгоритма решения задачи с помощью НАМ:

Идея работы НАМ состоит в том, что минимальная длина последовательности, подходящей под условие – 3(например ‘bab’). Следовательно, первое и главное действие – замена последовательностей вида Согласная/Гласная/Согласная на дополнительный знак ‘+’. Это делается по причине того, что последовательности могут быть длинее 3 букв. Таким образом, после последовательностей длиной 3, заменяются последовательности вида +Гласная/Согласная или Согласная/Гласная+. То есть, знак ‘+’ означает то, что перед/после него была последовательность, подходящая под условие. Таким образом возможно распознавать и удалять последовательности любой длины. Затем знак ‘+’ меняется на ‘.’, что является завершающей заменой. Однако из-за неё невозможно удалить больше 1 знака ‘+’ без завершения работы алгоритма. В следствии этого имеется большое количество вспомогательных замен, которые удаляют ‘+’ из смежных с ними букв, но при этом не ‘ломают’ слово, то есть, не создают новые подпоследовательности, которых не существовало в исходном слове. Также, если в слове два ‘+’ идут подряд, то они заменяются на пустоту.

3. Используемый алфавит:

Используемый в НАМ алфавит: {a,b,c,e,+}
Входной алфавит, совпадает с выходным: {a,b,c,e}
Вспомогательный алфавит: {+}

4. Система подстановок НАМ:

Система подстановок НАМ представлена в таблице:

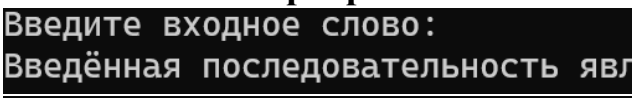
Образец	Замена
+ec	+
+eb	+
+ac	+
+ab	+
ce+	+
be+	+
ca+	+
ba+	+
b+b	bb
c+b	cb
b+c	bc
c+c	cc
a+b	ab
e+b	eb
b+a	ba
b+e	be
a+c	ac
e+c	ec
c+e	ce
c+a	ca
aa+	aa
ee+	ee
ae+	ae
ea+	ea
bab	+b
bac	+
beb	+
bec	+
cac	+
cab	+
cec	+
ceb	+
+a+	a
+b+	b
+c+	c
+e+	e
++	
+	.

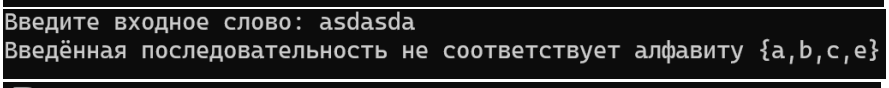
5. Набор тестов, охватыющих все режимы работы алгоритма, а также особые случаи

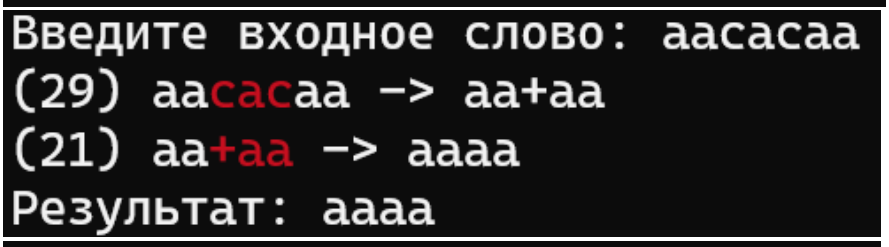
1. aacacaa -> aaaa

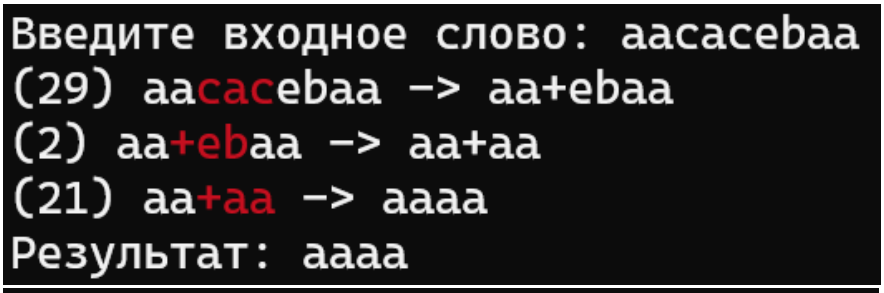
2. aacacebaa -> aaaa
3. caccaccac -> (пустая строка)
4. bbabaabab -> baa
5. cacacaccebbebaaca -> aaca
6. babceabbecbabbbbaecb -> ceabbbbaecb
7. aaacacebecebaaa -> aaaaaa
8. babbababecbbababab -> b
9. babbababecbbabaabab -> baa
10. cebabecabe -> e
11. becabecbebcabeb -> (пустая строка)
12. becabecbbecabec -> b
13. bcbcaeaeae -> bcbcaeaeae
14. b->b
15. abcebacebceb -> ab
16. bcaeabcbebeca -> bcaeabca
17. cbeaaceb -> cbeaa

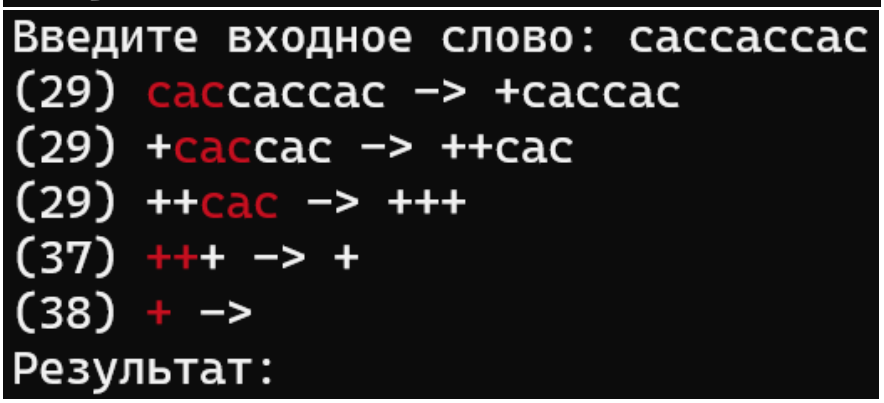
6. Скриншоты выполнения программы на тестовых примерах

1. 

2. 

3. 

4. 

5. 

6.
```
Введите входное слово: bbabaabab
(25) bbabaabab -> b+aabab
(13) b+aabab -> baabab
(25) baabab -> baa+
(38) baa+ -> baa
Результат: baa
```
7.
```
Введите входное слово: cacacaccebbebaaca
(27) cacacaccebbebaaca -> cacacacceb+aaca
(13) cacacacceb+aaca -> cacacaccebaaca
(29) cacacaccebaaca -> +acaccebaaca
(3) +acaccebaaca -> +accebaaca
(3) +accebaaca -> +cebaaca
(32) +cebaaca -> ++aaca
(21) ++aaca -> +aaca
(21) +aaca -> aaca
Результат: aaca
```
8.
```
Введите входное слово: babceabbecbabbbbaecb
(25) babceabbecbabbbbaecb -> +ceabbecbabbbbaecb
(25) +ceabbecbabbbbaecb -> +ceabbec+bbbaecb
(10) +ceabbec+bbbaecb -> +ceabbecbbbaecb
(28) +ceabbecbbbaecb -> +ceab+bbbaecb
(9) +ceab+bbbaecb -> +ceabbbbaecb
(38) +ceabbbbaecb -> ceabbbbaecb
Результат: ceabbbbaecb
```
9.
```
Введите входное слово: aaacacebecebaaa
(28) aaacacebecebaaa -> aaacace+ebaaa
(2) aaacace+ebaaa -> aaacace+aaa
(5) aaacace+aaa -> aaaca+aaa
(7) aaaca+aaa -> aaa+aaa
(21) aaa+aaa -> aaaaaa
Результат: aaaaaa
```
10.
```
Введите входное слово: babbababecbbabababab
(25) babbababecbbabababab -> +bababecbbabababab
(25) +babababecbbabababab -> ++abecbbabababab
(4) ++abecbbabababab -> ++ecbbabababab
(1) ++ecbbabababab -> ++bbabababab
(25) ++bbabababab -> ++b+abab
(4) ++b+ababab -> ++b+abab
(4) ++b+abab -> ++b+ab
(34) ++b+ab -> ++b+
(34) ++b+ -> +b
(38) +b -> b
Результат: b
```

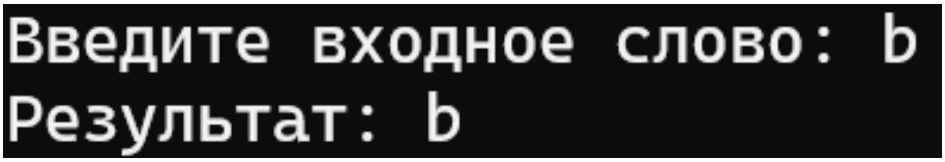
11.
```
Введите входное слово: babbababecbbabaabab
(25) babbababecbbabaabab -> +bababecbbabaabab
(25) +bababecbbabaabab -> ++abecbbabaabab
(4) ++abecbbabaabab -> ++ecbbabaabab
(1) ++ecbbabaabab -> ++bbabaabab
(25) ++bbabaabab -> ++b+aabab
(13) ++b+aabab -> ++baabab
(25) ++baabab -> ++baa+
(37) ++baa+ -> baa+
(38) baa+ -> baa
Результат: baa
```

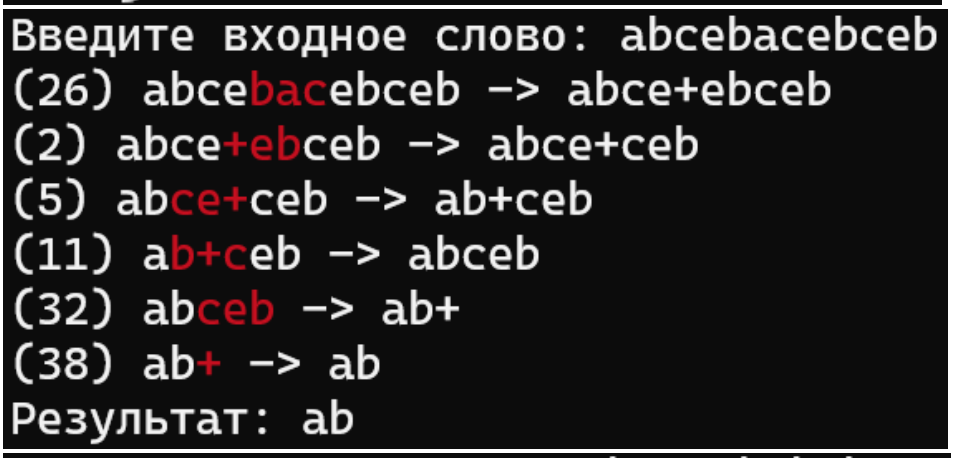
12.
```
Введите входное слово: cebabecabe
(25) cebabecabe -> ce+ecabe
(1) ce+ecabe -> ce+abe
(4) ce+abe -> ce+e
(5) ce+e -> +e
(38) +e -> e
Результат: e
```

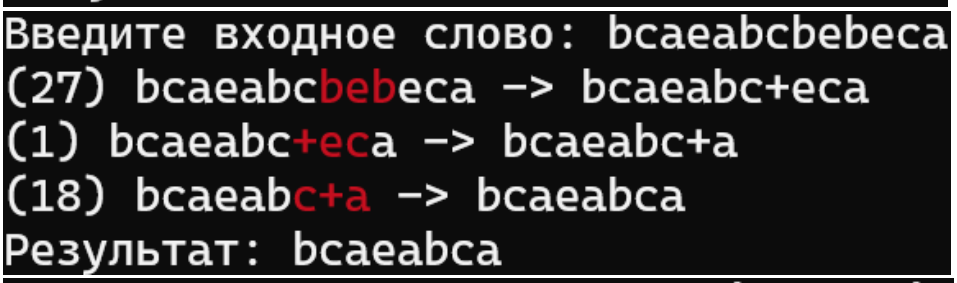
13.
```
Введите входное слово: becabecbebcabeb
(27) becabecbebcabeb -> becabec+cabeb
(12) becabec+cabeb -> becabeccabeb
(27) becabeccabeb -> becabecca+
(7) becabecca+ -> becabec+
(28) becabec+ -> +abec+
(4) +abec+ -> +ec+
(1) +ec+ -> ++
(37) ++ ->
Результат:
```

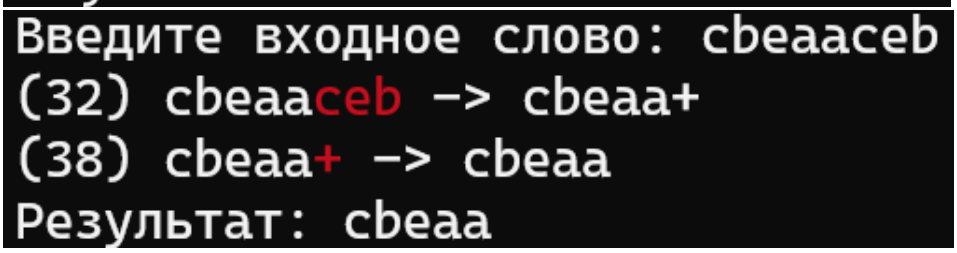
14.
```
Введите входное слово: becabecbbecabec
(28) becabecbbecabec -> +abecbbecabec
(4) +abecbbecabec -> +ecbbecabec
(1) +ecbbecabec -> +bbecabec
(28) +bbecabec -> +b+abec
(4) +b+abec -> +b+ec
(1) +b+ec -> +b+
(34) +b+ -> b
Результат: b
```

15.
```
Введите входное слово: bcbcaeaeae
Результат: bcbcaeaeae
```

16. 

17. 

18. 

19. 

7. Описание структуры данных, используемой в программе:

В программе используется структура с тремя компонентами:
Sample – образец для замены;
Replacement – замена для образца
endPoint – булевая переменная, указывает на то, является ли состояние завершающим.

8. Словесное описание одного шага программы:

Программная реализация одного шага программы работает следующим образом. Первым шагом является приоритетный поиск образца в заданной строке (проход циклом по таблице подстановок, пока не будет найдена первая подходящая подстановка). Если такой образец найден – он сменяется заменой, соответствующей ему, а также происходит проверка того, является ли это состояние завершающим(по результату ставится ставится флаг, указывающий на то, что алгоритм завершён). Затем посимвольно выводится изначальная строка, в которой образец окрашивается в красный цвет. После этого выводится и возвращается строка, полученная после замены. Если образец не был найден – это означает то, что

алгоритм завершается, соответственно ставится флаг, указывающий на то, что алгоритм завершён, а также возвращается полученная строка.

9. Листинг програмы:

```
//ЮФУ, ИКТИБ, МОП ЭВМ
//Программирование и основы теории алгоритмов
//ДЗ3 - Проектирование и реализация нормального алгоритма Маркова.
//КТбо1-6, Кравченко Александр Андреевич
//25.04.2024
#include <iostream>
#include <string>
#include <conio.h>
#include <windows.h>
#include <regex>

struct SubstTable {
    std::string Sample;
    std::string Replacement;
    bool endPoint;
};

// Функция проверки входных данных
// Получает на вход строку
// Возвращает значения:
// 0 - введённая строка соответствует критериям
// 1 - введённая строка не соответствует алфавиту
// 2 - введённая строка пуста
int inputValid(std::string);

// Функция выбора и реализации подстановки для текущего слова и вывода результата шага
// Получает на вход: введённую строку, таблицу подстановок, флаг окончания работы алгоритма
// Проходит по каждому состоянию таблицы сверху вниз. Проверяет введённую строку на наличие образца.
// Если образец найден - меняет его на подстановку, проверяет является ли подстановка конечной,
// выводит номер подстановки, окрашивает образец в строке, выводит результат подстановки.
// Если не найден - ставит флаг остановки в 1
// В любом случае возвращает строку
std::string Substitution(std::string, const SubstTable*, bool&);

int main() {
    setlocale(LC_ALL, "Russian");

    SubstTable STable[38] = {
        {"+ec","+",0},
        {"+eb","+",0},
        {"+ac","+",0},
        {"+ab","+",0},
        {"ce+","+",0},
        {"be+","+",0},
        {"ca+","+",0},
        {"ba+","+",0},
        {"b+b","bb",0},
        {"c+b","cb",0},
        {"b+c","bc",0},
        {"c+c","cc",0},
        {"b+a","ba",0},
        {"b+e","be",0},
        {"a+c","ac",0},
```

```
        {"e+c","ec",0},
        {"c+e","ce",0},
        {"c+a","ca",0},
        {"a+b","ab",0},
        {"e+b","eb",0},
        {"+aa","aa",0},
        {"+ee","ee",0},
        {"+ae","ae",0},
        {"+ea","ea",0},
        {"bab","+",0},
        {"bac","+",0},
        {"beb","+",0},
        {"bec","+",0},
        {"cac","+",0},
        {"cab","+",0},
        {"cec","+",0},
        {"ceb","+",0},
        {"+a+","a",0},
        {"+b+","b",0},
        {"+c+","c",0},
        {"+e+","e",0},
        {"++","",0},
        {"+","",1},
    };

    std::string input;
    char cont = -1;

    do {
        bool stop = 0;
        if (cont != 1)
            std::cout << "Введите входное слово: ";
        std::getline(std::cin, input);
        std::transform(input.begin(), input.end(), input.begin(), [](unsigned char
c) { return std::tolower(c); });

        int action = inputValid(input);
        if (!action) {
            while (stop != 1) {
                input = Substitution(input, STable, stop);
            }
            std::cout << "Результат: " << input;
            std::cout << "\nНажмите любую клавишу, чтобы продолжить\nДля выхода
нажите клавишу ESC на клавиатуре";
            cont = _getch();
            std::cout << "\n\n";
        }
        else if (action == 1) {
            cont = 1;
            std::cout << "Введённая последовательность не соответствует алфавиту
{a,b,c,e}\n\nПопробуйте ввести другую последовательность: ";
        }
        else if (action == 2) {
            cont = 1;
            std::cout << "Введённая последовательность является пустой\n\nПопробуйте
ввести другую последовательность: ";
        }
    } while (cont != 27);
    std::cout << "Спасибо за использование!";
    return 0;
}

int inputValid(std::string input) {
    const std::regex validSymbols("^(a|b|c|e)+$");
    if (input.empty()) return 2;
```

```
        if (!regex_match(input, validSymbols)) return 1;
        return 0;
    }

    std::string Substitution(std::string input, const SubstTable* STable, bool& stop) {
        HANDLE hConsoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);

        std::string noSub = input;

        for (int i = 0; i < 38; i++)
        {
            size_t replacePos = input.find(STable[i].Sample);
            if (replacePos != std::string::npos) {
                stop = STable[i].endPoint;
                input.replace(replacePos, STable[i].Sample.size(),
    STable[i].Replacement);

                std::cout << "(" << i + 1 << ") ";
                for (int j = 0; j < noSub.size(); j++)
                {
                    SetConsoleTextAttribute(hConsoleHandle, (j >= replacePos && j <
    replacePos + STable[i].Sample.size()) ? 4 : 15);
                    std::cout << noSub[j];
                }
                SetConsoleTextAttribute(hConsoleHandle, 15);
                std::cout << " -> " << input << "\n";
                return input;
            }
        }
        stop = 1;
        return input;
}
```