МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Ахо-Корасик

Студентка гр. 8383 Ишанина Л.Н.

Преподаватель Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Задание.

Вариант 4.

Реализовать режим поиска, при котором все найденные образцы не пересекаются в строке поиска (т.е. некоторые вхождения не будут найдены; решение задачи неоднозначно)..

Вход:
Первая строка содержит текст ($T, 1 \leq |T| \leq 100000$).
Вторая - число n ($1 \leq n \leq 3000$), каждая следующая из n строк содержит шаблон из набора $P = \{p1,\ldots,pn\}$ $1 \leq |pi| \leq 75$
Все строки содержат символы из алфавита {A,C,G,T,N}
Выход:
Все вхождения образцов из P в T.
Каждое вхождение образца в текст представить в виде двух чисел - i p
Где i - позиция в тексте (нумерация начинается с 1), с которой начинается вхождение образца с номером p
(нумерация образцов начинается с 1).
Строки выхода должны быть отсортированы по возрастанию, сначала номера позиции, затем номера шаблона.

Описание структуры данных.

В данной программе используются векторы.

Вектор vector<std::string> strings – данный контейнер нужен для записи в него строк-шаблонов, из которых потом и будет составляться бор.

Вектор vector<std::pair<int, int>> vectorAnswers – это вектор для записи в него ответа.

Описание используемого класса.

Класс вершин бора(class Top):

Класс вершин бора состоит из следующих свойств:

- vector <int> nextV – вектор вершин, которые являются ребенком для текущей;
- int numberString – номер строки-шаблона;
- bool isTerminal – булевская переменная для обозначения терминальных вершин;
- int suffLink – переменная для суффиксной ссылки;
- vector <int> autoMove – вектор для перехода по состояниям;
- int parent – переменная для родителя обозначающая номер его вершины;

- char numberSymbol – символ для перехода;
- vector <Top> bor - бор;
- char name – переменная для вывода имени символа, по которому переходим;

Класс вершин бора содержит следующие конструкторы:

- Top(int parent, char numberSymbol) – конструктор, принимающий номер вершины родителя и символ по которому от него переходят. Конструктор инициализирует все поля класса.
- Top() – конструктор по умолчанию, он создает начальную вершину бора и добавляет её в бор. Данный конструктор вызывается в самом начале программы.

Описание алгоритма1.

Алгоритм строит конечный автомат, которому затем передаёт строку поиска. Автомат получает по очереди все символы строки и переходит по соответствующим рёбрам. Если автомат пришёл в конечное состояние, соответствующая строка словаря присутствует в строке поиска.

Самым первым шагом алгоритм строит бор, то есть такую структуру данных для хранения набора строк, представляющую из себя подвешенное дерево с символами на рёбрах. Строки получаются последовательной записью всех символов, хранящихся на рёбрах между корнем бора и терминальной вершиной. Размер бора линейно зависит от суммы длин всех строк, а поиск в бору занимает время, пропорциональное длине образца.

Далее алгоритм осуществляет посимвольный проход по тексту, начинается этот обход с вершины бора. Если оттуда есть переход по текущему символу, то переходим по нему, если нет, то переходим по суффиксной ссылке. Если после перехода по суффиксной ссылке снова нет ребра с текущим символом, то алгоритм проходит по суффиксной ссылке ещё и ещё раз, пока либо не пройдёт по корневой ссылке-петле, либо не найдёт ребро, нагруженное нужным символом. Важно, что для корневого узла суффиксная ссылка — петля.

Для реализации режима поиска, при котором все найденные образцы не

пересекаются в строке поиска, суффиксные ссылки на терминальных вершинах переносят на начало бора.

Описание main ( ) :

В функции прописан ввод текста, в котором будет идти поиск и строк-шаблонов. Также вызов функции обработки текста, сортировка вектора ответа, а также вывод ответа.

Кроме того, для удобства, во время работы алгоритма, происходит вывод промежуточной информации.

Описание дополнительных функций.

Функция void addString(std::vector<Top>* bor, std::string* string, int numberString) принимает на вход бор, который изначально состоит из начальной вершины, строку-шаблон и номер шаблона. Данная функция добавляет строку-шаблон в бор.

Функция int getSuff(int top, std::vector<Top>* bor) принимает на вход вершину и бор. Данная функция предназначена для получения суффиксной ссылки вершины. Для вершин, которые являются детьми начальной вершины бора, суффиксная ссылка устанавливается на начало. Если вершина является терминальной, то суффиксная ссылка устанавливается также на начало бора, именно благодаря этому выполняется условие варианта: все найденные образцы не пересекаются в строке поиска.

Функция int getAutoMove(int top, int token, std::vector<Top>* bor) принимает на вход номер вершины, её символ и бор. Данная функция осуществляет переход к ребенку вершины или, если перехода по символу нет, то осуществляет переход по суффиксной ссылке.

Функция void function(std::string& text, std::vector<std::pair<int, int>>* vectorAnswers, std::vector<Top>* bor, std::vector<std::string>* strings) принимает на вход текст, в котором будет идти поиск, вектор ответа, который состоит из пары чисел(номер вхождения и номер строки-шаблона), бор и вектор шаблонов. Данная

функция осуществляет саму обработку текста, с помощью функции getAutoMove и вызывает функцию writeResponse(), которая осуществляет проверку и записывает ответ в вектор ответа.

Функция void writeResponse(int vert, int ch, std::vector<std::pair<int, int>>* vectorAnswers, std::vector<Top>* bor, std::vector<std::string>* strings) принимает номер вершины, вектор ответа, бор и вектор шаблонов. Данная функция записывает позицию в тексте найденного шаблона и его номер в вектор ответа.

Сложность алгоритма1.

Сложность алгоритма по времени: O (m*k + n + g), m – длина образа, n – длина текста, g – общая длина совпадений, так как алгоритм проходит всю строку и в случае совпадения, осуществляет сравнение текста со строкой шаблоном.

Сложность алгоритма по памяти: O (m*k), m – длина всех образов, а k – размер алфавита, так как программа хранит шаблоны, которые считываются в самом начале.

Описание алгоритма2.

Данный алгоритм решает задачу точного поиска для одного образца с *джокером*. Для того чтобы найти все вхождения в текст заданного шаблона с масками Q, необходимо обнаружить вхождения в текст всех его безмасочных кусков. То есть изначально, после считывания шаблона, вызываются функции разделения шаблона на подстроки. Затем, полученные подстроки добавляются в бор, устанавливаются суффиксные ссылки и начинается поиск вхождений подстрок в шаблоне. Например, шаблон AG$$C$ содержит две подстроки без масок AG и C и их стартовые позиции соответственно 1 и 5. Для алгоритма используется массив array, где array[i] — количество встретившихся в тексте безмасочных подстрок шаблона, который начинается в тексте на позиции i. Тогда появление подстроки в тексте на позиции j будет означать возможное появление шаблона на позиции j−стартовая позиция подстрок из шаблона+1.

Используя алгоритм Ахо-Корасик, находятся безмасочные подстроки шаблона. Поиск подстрок заданного шаблона с помощью алгоритма Ахо-Корасик выполняется за время O(m+n+a), где n — суммарная длина подстрок, то есть длина шаблона, m — длина текста, a — количество появлений подстрок шаблона. Далее просто надо пробежаться по массиву array и просмотреть значения ячеек за время O(m).

Для индивидуализации варианта в конце происходит проверка на пересечение шаблона с самим собой.

Сложность алгоритма2.

Сложность алгоритма по времени: (m+n+a), где n — суммарная длина подстрок, то есть длина шаблона, m — длина текста, a — количество появлений подстрок шаблона.

Сложность алгоритма по памяти: O (m), m – длина образов, так как программа хранит строку-образ, которая считывается в самом начале.

Тестирование алгоритма1.

Входные данные:

TAGTAGAGG

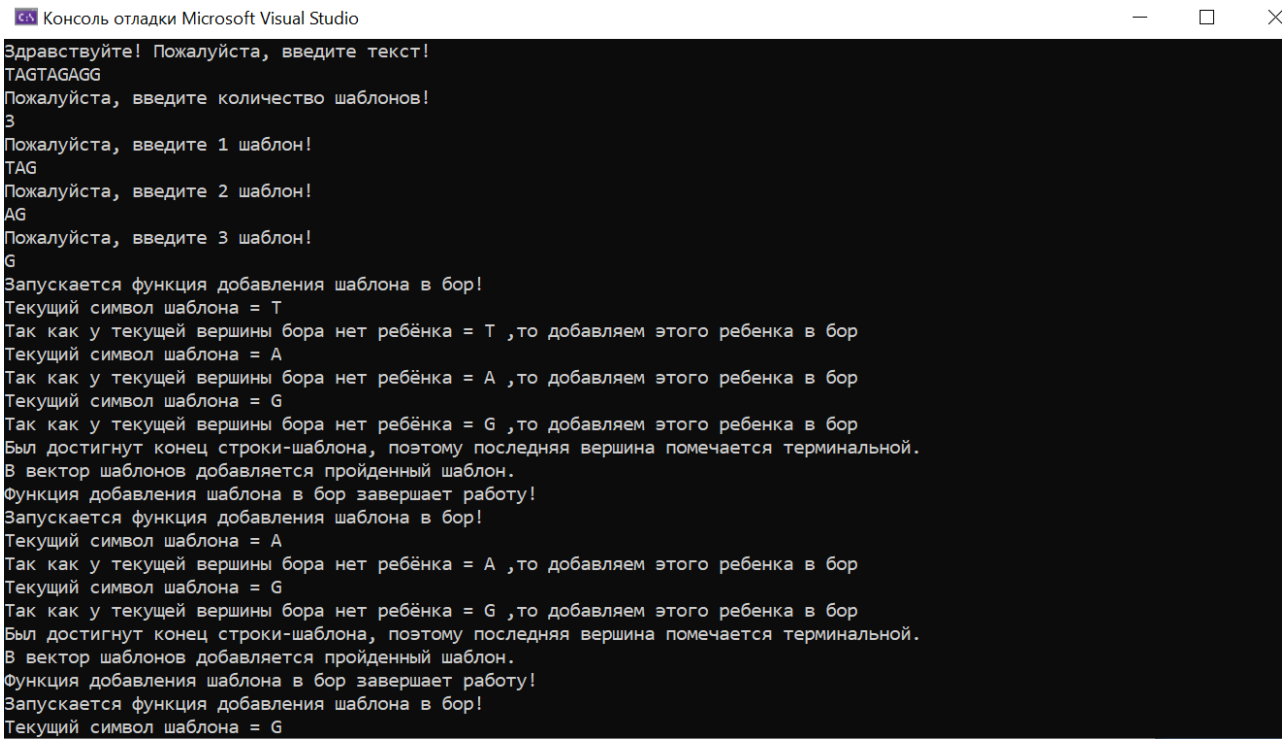
3

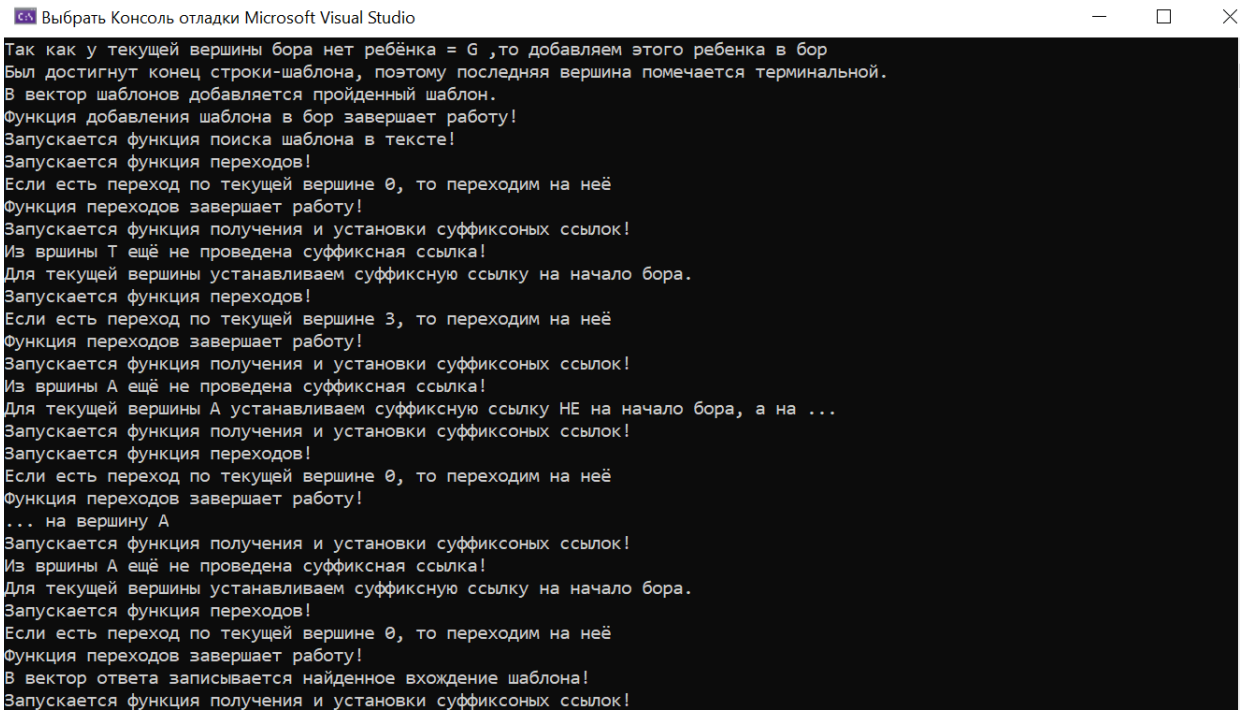
TAG

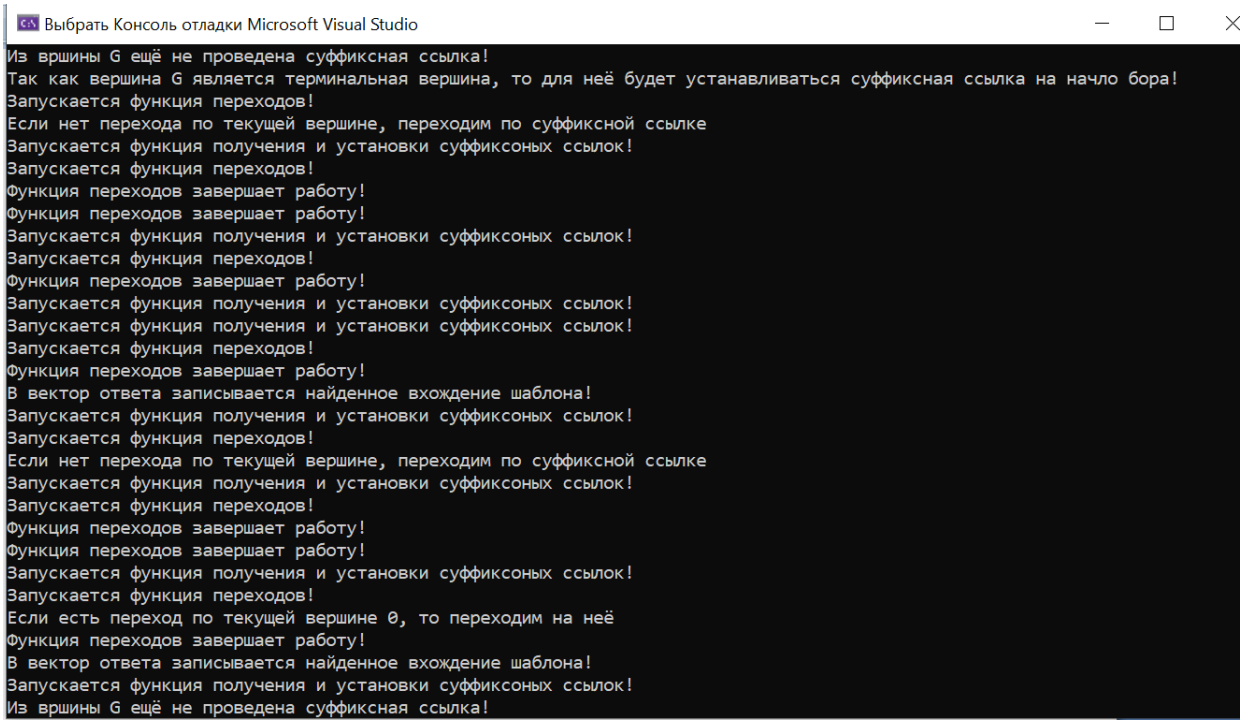
AG

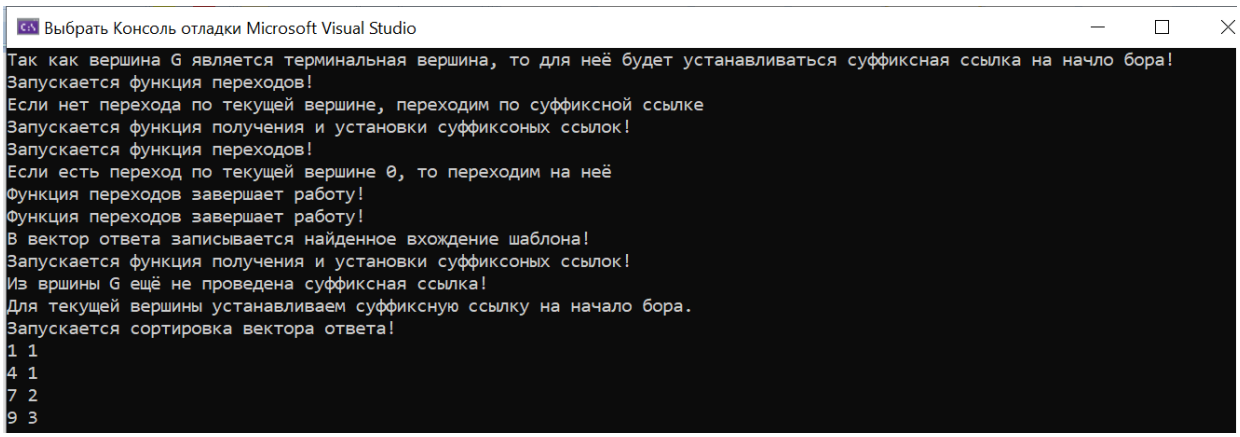
G

Вывод:


Консоль отладки Microsoft Visual Studio
Здравствуйте! Пожалуйста, введите текст!
TAGTAGAGG
Пожалуйста, введите количество шаблонов!
3
Пожалуйста, введите 1 шаблон!
TAG
Пожалуйста, введите 2 шаблон!
AG
Пожалуйста, введите 3 шаблон!
G
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = G
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = G
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = G


Выбрать Консоль отладки Microsoft Visual Studio
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция поиска шаблона в тексте!
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины T ещё не проведена суффиксная ссылка!
Для текущей вершины устанавливаем суффиксную ссылку на начало бора.
Запускается функция переходов!
Если есть переход по текущей вершине 3, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!
Для текущей вершины A устанавливаем суффиксную ссылку НЕ на начало бора, а на ...
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
... на вершину A
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!
Для текущей вершины устанавливаем суффиксную ссылку на начало бора.
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
В вектор ответа записывается найденное вхождение шаблона!
Запускается функция получения и установки суффиксоных ссылок!





Тест№2

Входные данные:

GATTA_TATTA_CAT

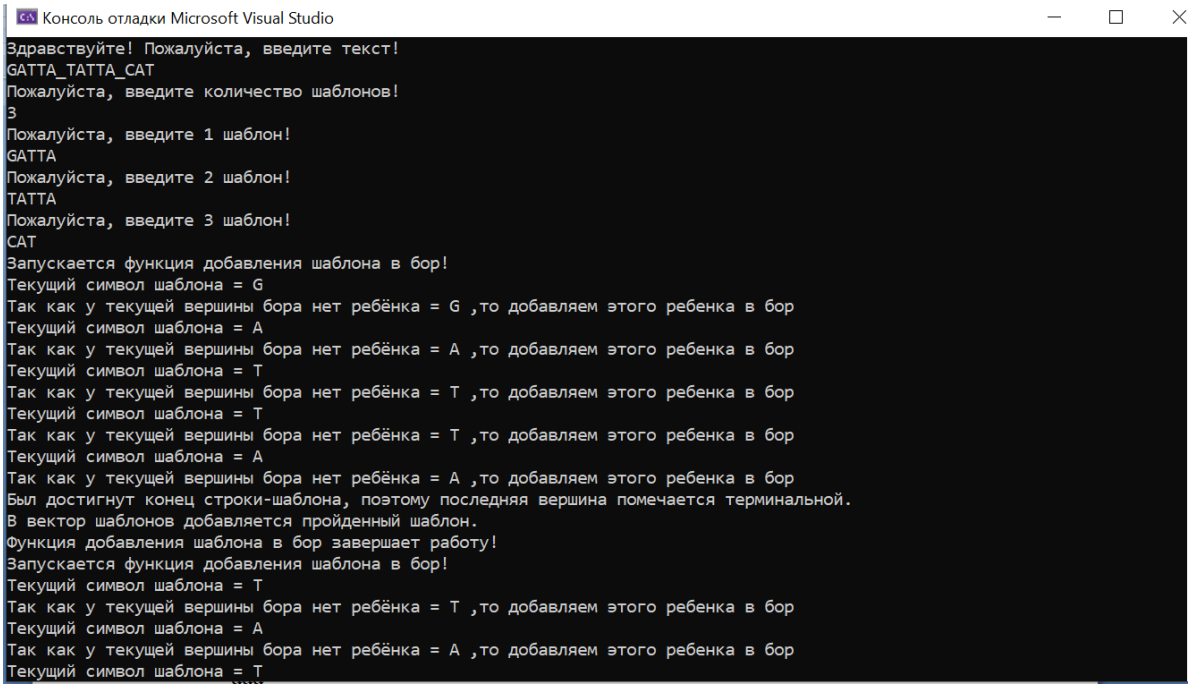
3

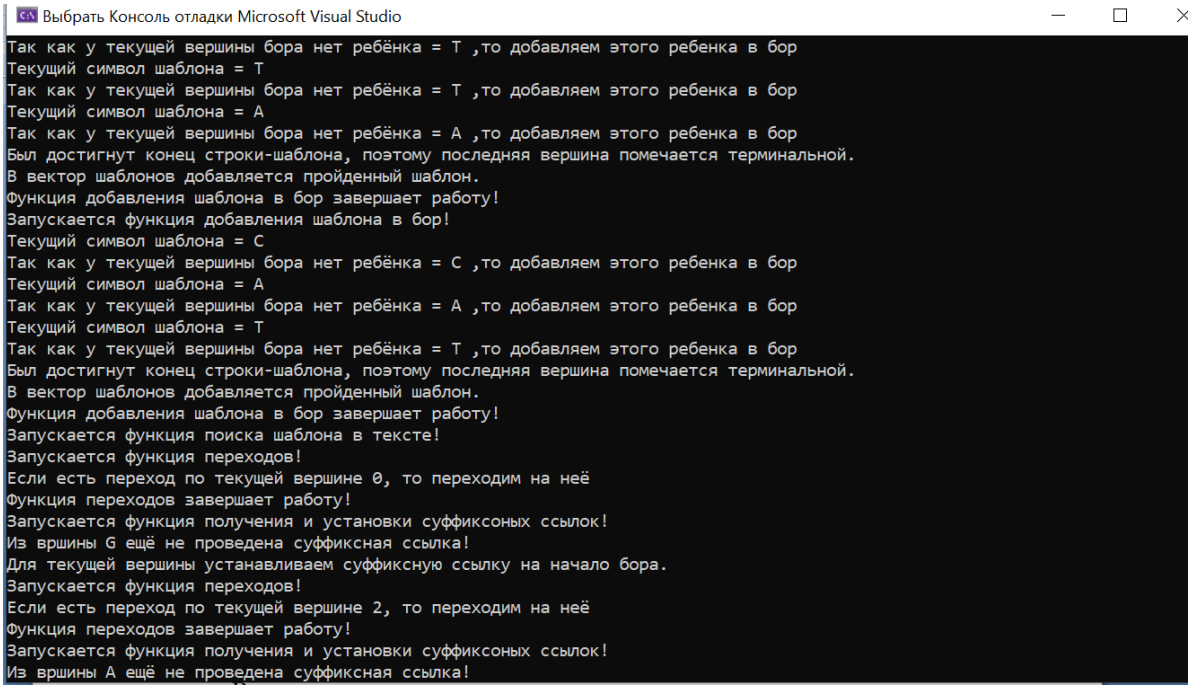
GATTA

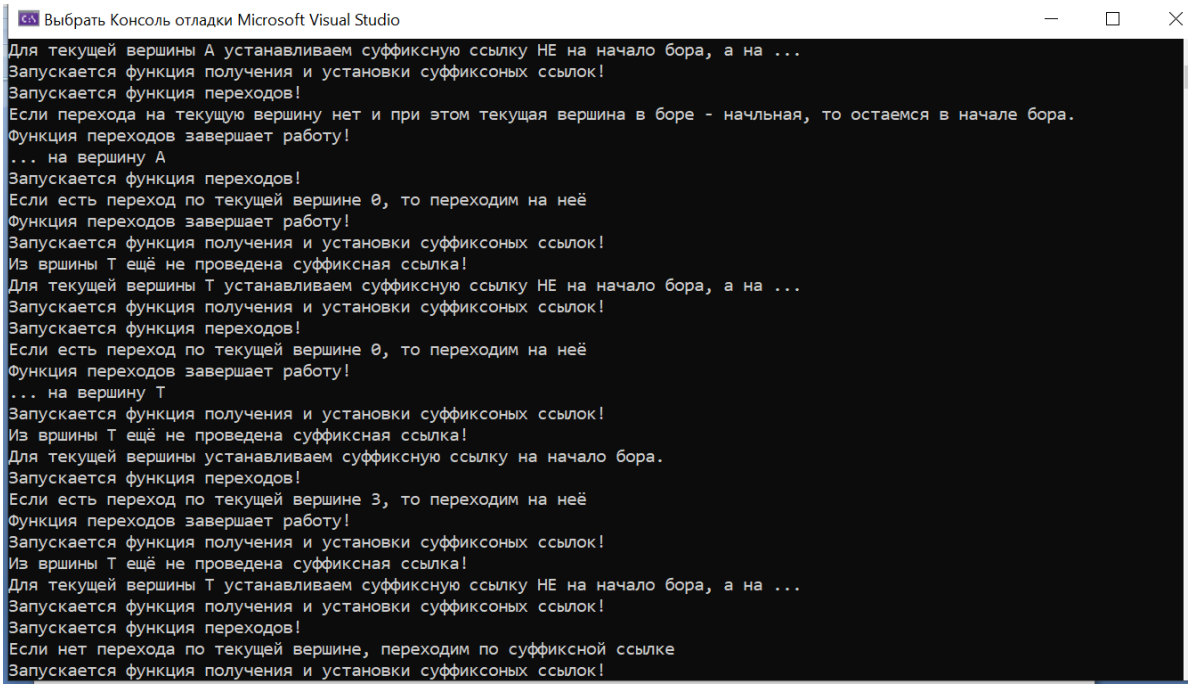
TATTA

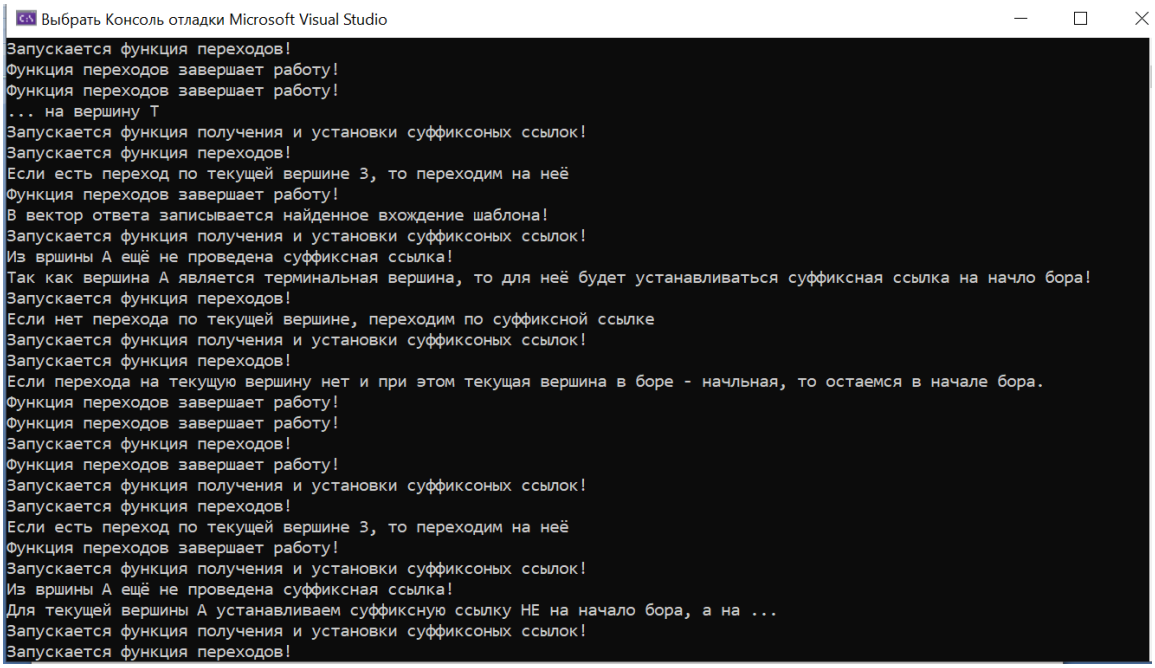
CAT

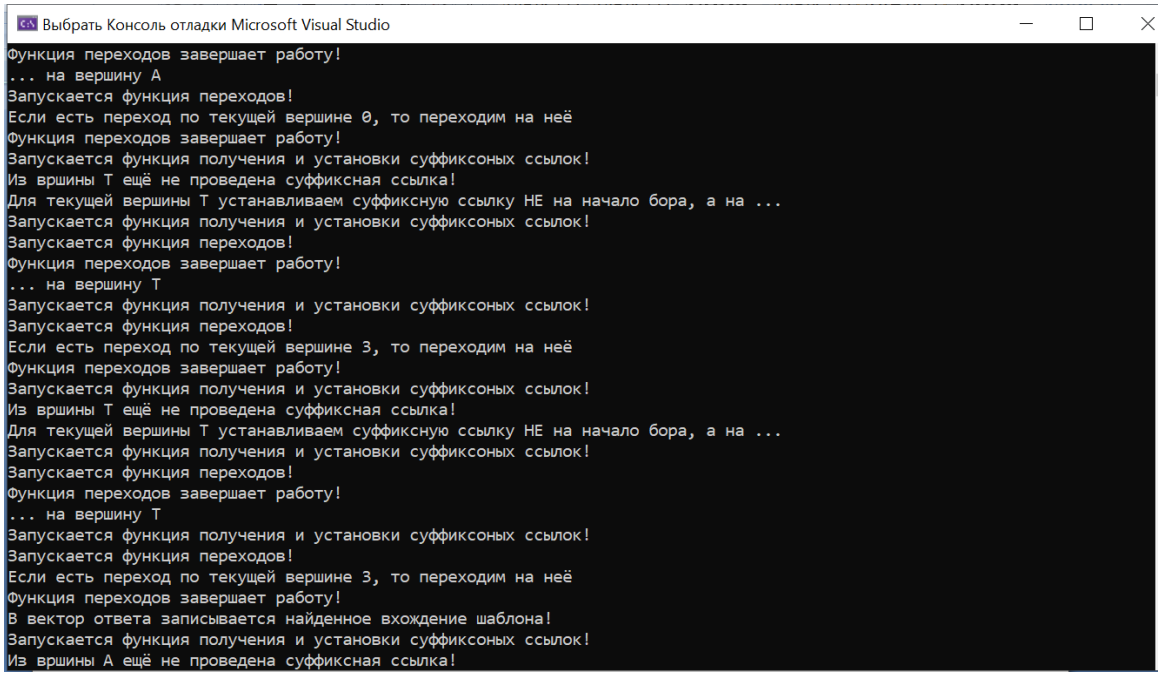
Вывод:

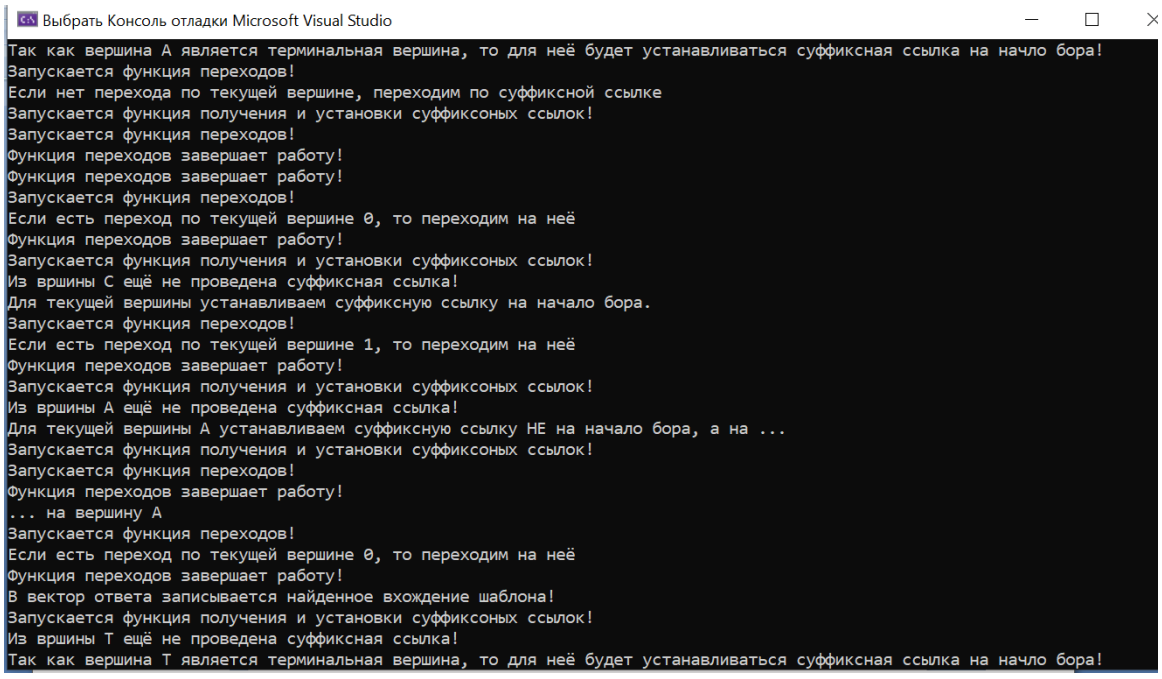

Консоль отладки Microsoft Visual Studio
Здравствуйте! Пожалуйста, введите текст!
GATTA_TATTA_CAT
Пожалуйста, введите количество шаблонов!
3
Пожалуйста, введите 1 шаблон!
GATTA
Пожалуйста, введите 2 шаблон!
TATTA
Пожалуйста, введите 3 шаблон!
CAT
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = G
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = T


Выбрать Консоль отладки Microsoft Visual Studio
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = C
Так как у текущей вершины бора нет ребёнка = C ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = T
Так как у текущей вершины бора нет ребёнка = T ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция поиска шаблона в тексте!
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины G ещё не проведена суффиксная ссылка!
Для текущей вершины устанавливаем суффиксную ссылку на начало бора.
Запускается функция переходов!
Если есть переход по текущей вершине 2, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!


Выбрать Консоль отладки Microsoft Visual Studio
Для текущей вершины A устанавливаем суффиксную ссылку НЕ на начало бора, а на ...
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если перехода на текущую вершину нет и при этом текущая вершина в боре - начльная, то остаемся в начале бора.
Функция переходов завершает работу!
... на вершину A
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины T ещё не проведена суффиксная ссылка!
Для текущей вершины T устанавливаем суффиксную ссылку НЕ на начало бора, а на ...
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
... на вершину T
Запускается функция получения и установки суффиксоных ссылок!
Из вршины T ещё не проведена суффиксная ссылка!
Для текущей вершины устанавливаем суффиксную ссылку на начало бора.
Запускается функция переходов!
Если есть переход по текущей вершине 3, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины T ещё не проведена суффиксная ссылка!
Для текущей вершины T устанавливаем суффиксную ссылку НЕ на начало бора, а на ...
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если нет перехода по текущей вершине, переходим по суффиксной ссылке
Запускается функция получения и установки суффиксоных ссылок!


Выбрать Консоль отладки Microsoft Visual Studio
Запускается функция переходов!
Функция переходов завершает работу!
Функция переходов завершает работу!
... на вершину T
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если есть переход по текущей вершине 3, то переходим на неё
Функция переходов завершает работу!
В вектор ответа записывается найденное вхождение шаблона!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!
Так как вершина A является терминальная вершина, то для неё будет устанавливаться суффиксная ссылка на начло бора!
Запускается функция переходов!
Если нет перехода по текущей вершине, переходим по суффиксной ссылке
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если перехода на текущую вершину нет и при этом текущая вершина в боре - начльная, то остаемся в начале бора.
Функция переходов завершает работу!
Функция переходов завершает работу!
Запускается функция переходов!
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Если есть переход по текущей вершине 3, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!
Для текущей вершины A устанавливаем суффиксную ссылку НЕ на начало бора, а на ...
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!





Тест№3

Входные данные:

AGAGAGAAAGG

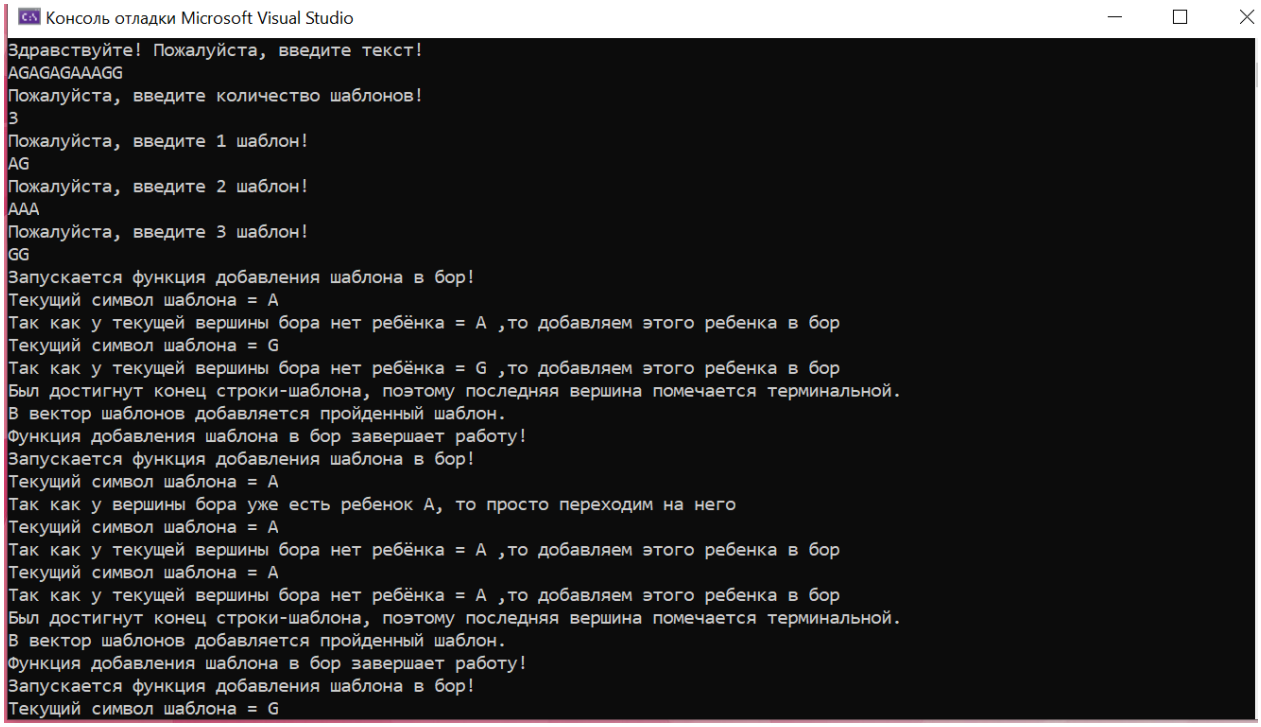
3

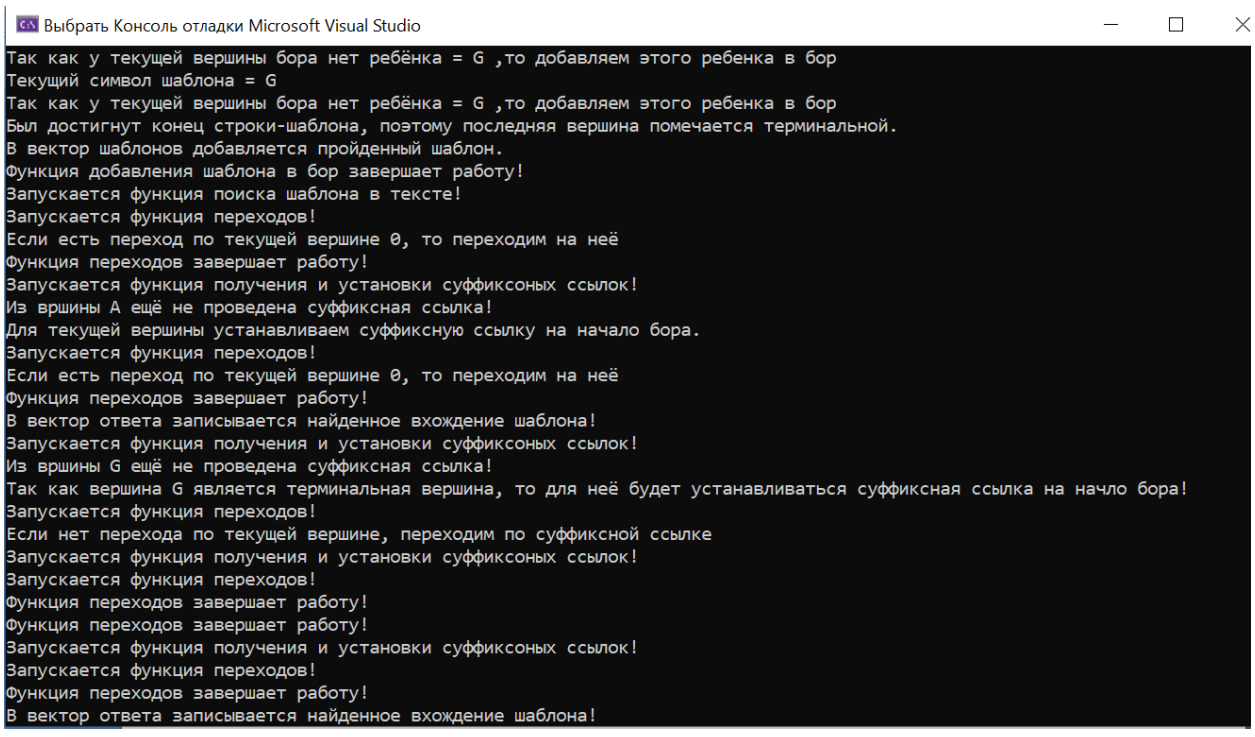
AG

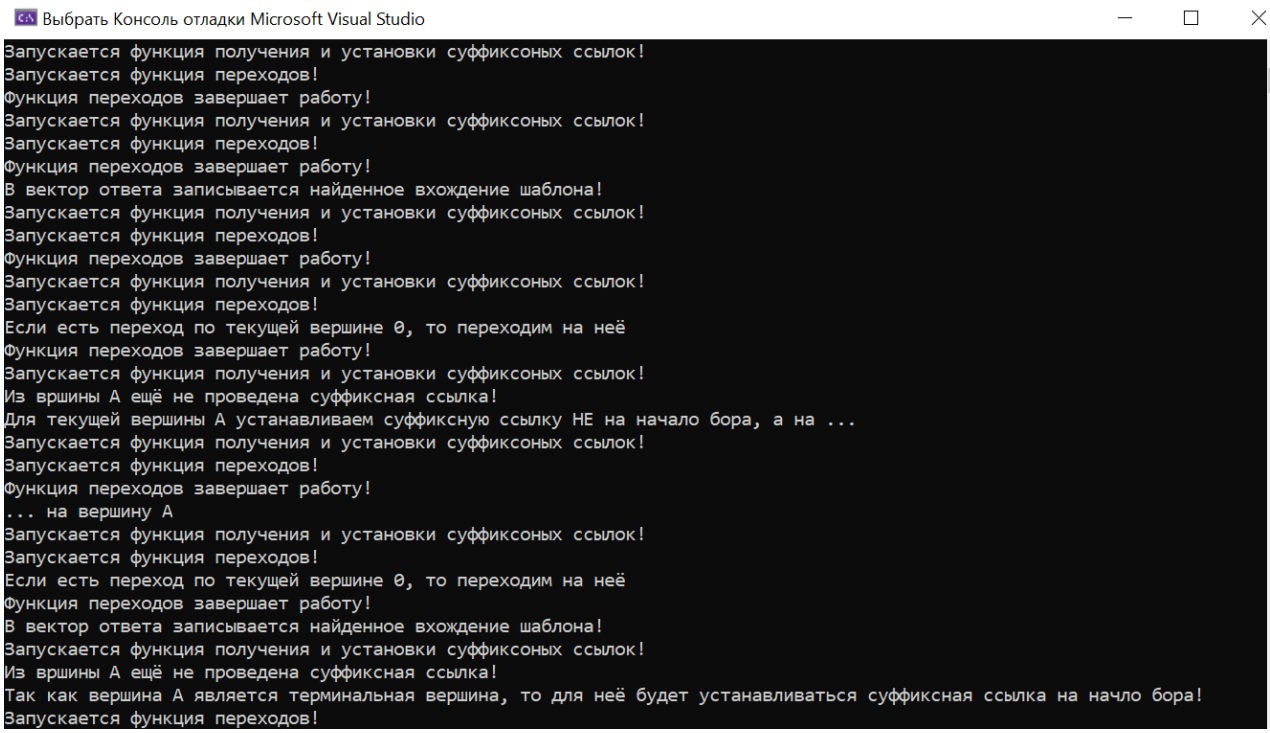
AAA

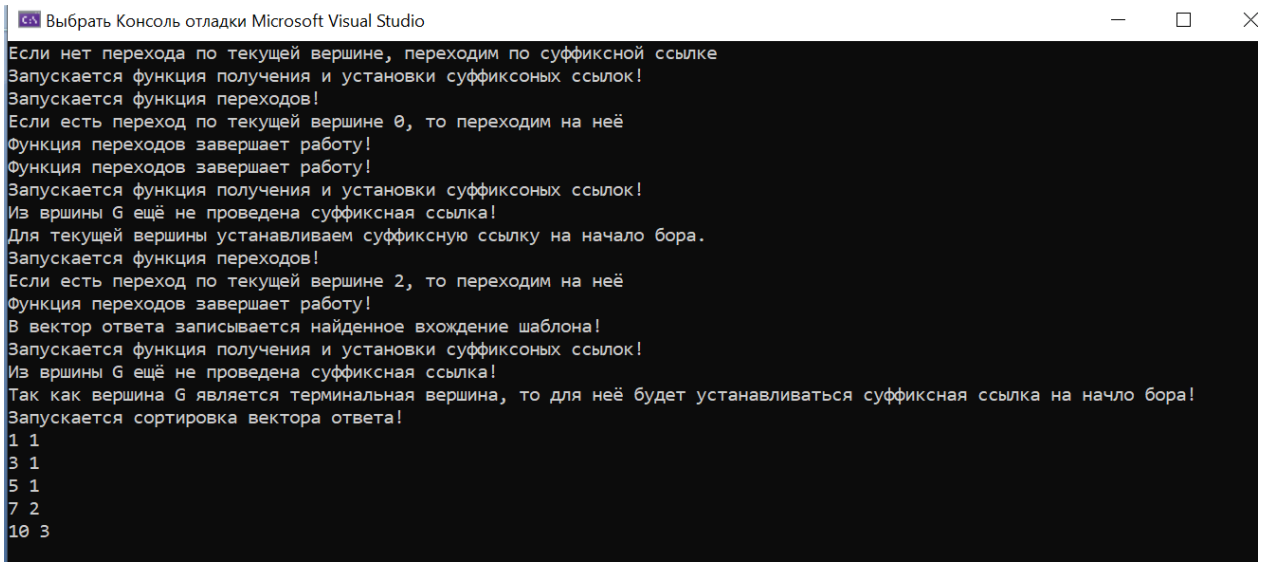
GG

Вывод:


Консоль отладки Microsoft Visual Studio
Здравствуйте! Пожалуйста, введите текст!
AGAGAGAAAGG
Пожалуйста, введите количество шаблонов!
3
Пожалуйста, введите 1 шаблон!
AG
Пожалуйста, введите 2 шаблон!
AAA
Пожалуйста, введите 3 шаблон!
GG
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = G
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = A
Так как у вершины бора уже есть ребенок A, то просто переходим на него
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Текущий символ шаблона = A
Так как у текущей вершины бора нет ребёнка = A ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция добавления шаблона в бор!
Текущий символ шаблона = G


Выбрать Консоль отладки Microsoft Visual Studio
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Текущий символ шаблона = G
Так как у текущей вершины бора нет ребёнка = G ,то добавляем этого ребенка в бор
Был достигнут конец строки-шаблона, поэтому последняя вершина помечается терминальной.
В вектор шаблонов добавляется пройденный шаблон.
Функция добавления шаблона в бор завершает работу!
Запускается функция поиска шаблона в тексте!
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины A ещё не проведена суффиксная ссылка!
Для текущей вершины устанавливаем суффиксную ссылку на начало бора.
Запускается функция переходов!
Если есть переход по текущей вершине 0, то переходим на неё
Функция переходов завершает работу!
В вектор ответа записывается найденное вхождение шаблона!
Запускается функция получения и установки суффиксоных ссылок!
Из вршины G ещё не проведена суффиксная ссылка!
Так как вершина G является терминальная вершина, то для неё будет устанавливаться суффиксная ссылка на начло бора!
Запускается функция переходов!
Если нет перехода по текущей вершине, переходим по суффиксной ссылке
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Функция переходов завершает работу!
Функция переходов завершает работу!
Запускается функция получения и установки суффиксоных ссылок!
Запускается функция переходов!
Функция переходов завершает работу!
В вектор ответа записывается найденное вхождение шаблона!





Тест№4

Входные данные:

ILYILYLYY

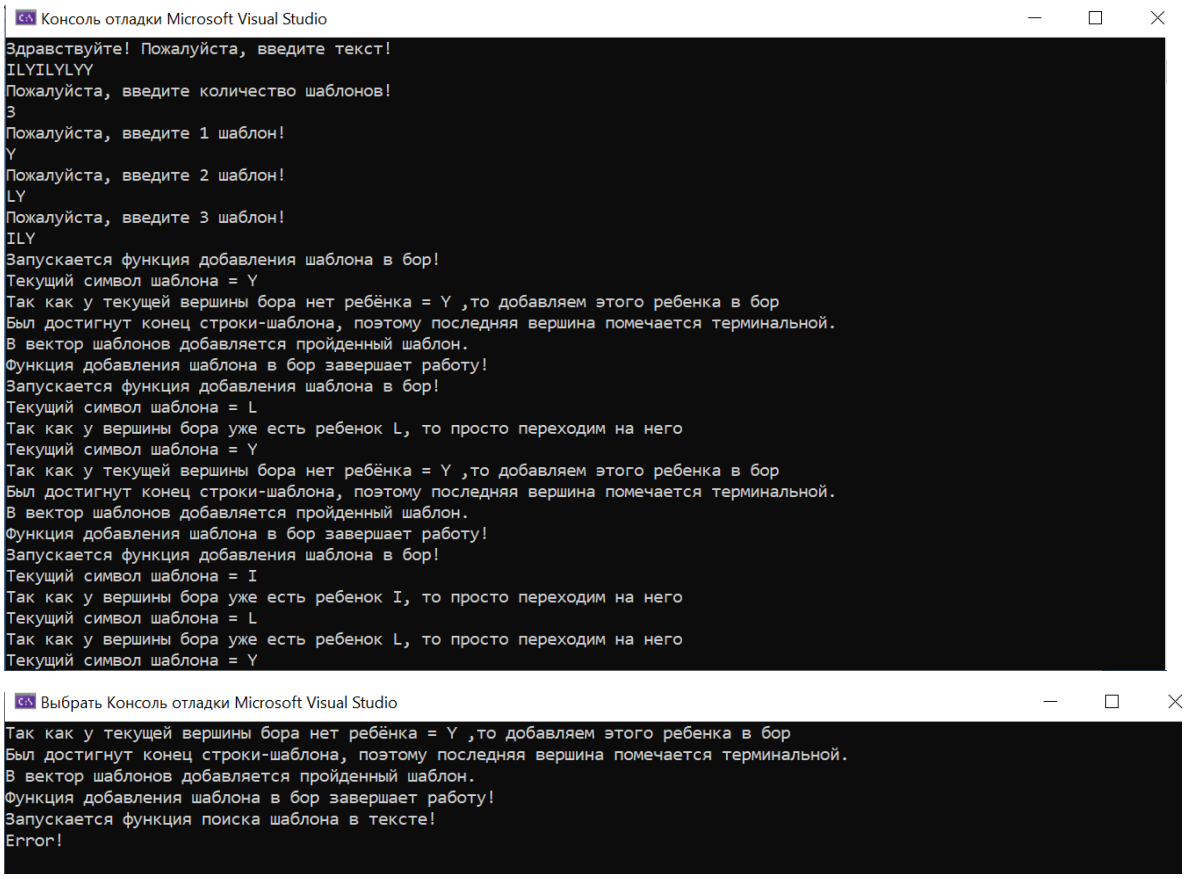
3

Y

LY

ILY

Вывод:



Тестирование алгоритма2.

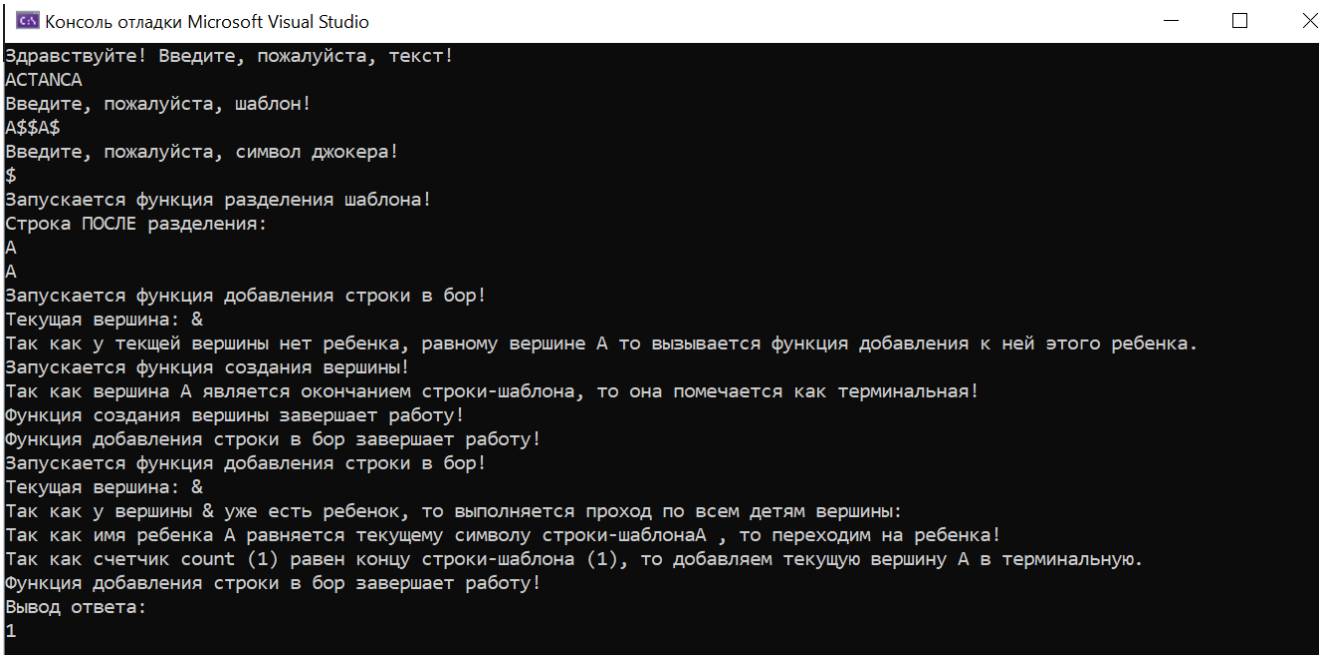
Входные данные:

ACTANCA

A$$A$

$

Вывод:



Тестирование алгоритма2.

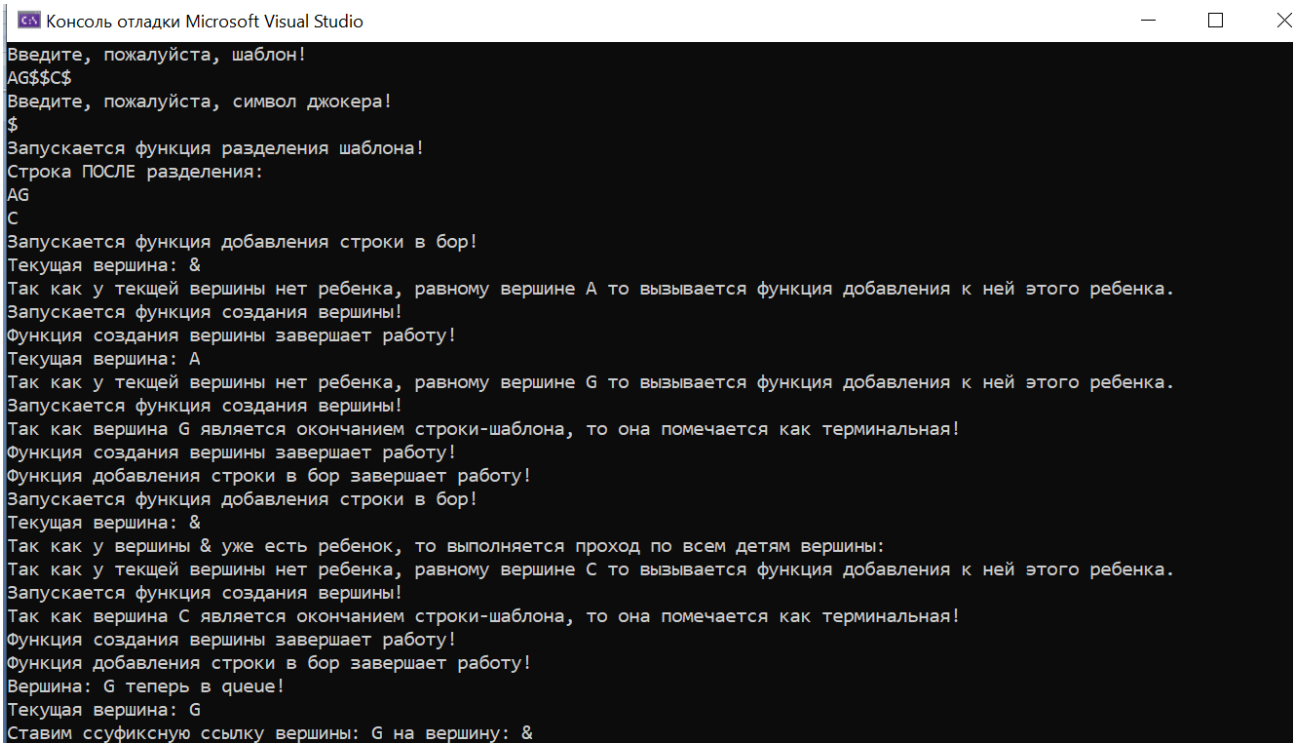
Входные данные:

NAGNNCAGAGCAN

AG$$C$

$

Вывод:



Пояснение к результату: так как конец первого вхождения шаблона является началом ещё одного его вхождения, то из-за их пересечения, вхождение второй раз не учитывается.

Вывод.

В ходе работы был построен и анализирован алгоритм Ахо-Корасик. Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД

```
#include <iostream>
#include <vector>
#include <algorithm>


class Top
{
public:
    std::vector <int> nextV;
    int numberString;
    bool isTerminal;
    int suffLink;
    std::vector <int> autoMove;
    int parent;
    char numberSymbol;
    std::vector <Top> bor;
    char name;

    Top(int parent, char numberSymbol) : parent(parent), numberSymbol(numberSymbol)
    {
        nextV = { -1,-1,-1,-1,-1,-1 };
        autoMove = { -1,-1,-1,-1,-1,-1 };
        suffLink = -1;
        numberString = 0;
        isTerminal = false;

        if (numberSymbol == 0)
            name = 'A';
        else if (numberSymbol == 1)
            name = 'C';
        else if (numberSymbol == 2)
            name = 'G';
        else if (numberSymbol == 3)
            name = 'T';
        else if (numberSymbol == 4)
            name = 'N';
        else
            name = '#';
    }

    Top()
    {
        bor.emplace_back(Top(0, 0));
    }
};

void writeResponse(int vert, int ch, std::vector<std::pair<int, int>>* vectorAnswers,
std::vector<Top>* bor, std::vector<std::string>* strings);
int getAutoMove(int vert, int ch, std::vector<Top>* bor);

void addString(std::vector<Top>* bor, std::string* string, int numberString)
{
    std::cout << "Запускается функция добавления шаблона в бор!" << std::endl;
    int num = 0;
    bool flag = false;
    for (int i = 0; i < string->length(); i++)
    {
        int token = ' ';
        if (string->at(i) == 'A')
            token = 0;
```

```
        else if (string->at(i) == 'C')
            token = 1;
        else if (string->at(i) == 'G')
            token = 2;
        else if (string->at(i) == 'T')
            token = 3;
        else if (string->at(i) == 'N')
            token = 4;
        else
            token = 5;
        std::cout << "Текущий символ шаблона = " << string->at(i) << std::endl;
        if (bor->at(num).nextV[token] == -1)
        {
            flag = true;
            std::cout << "Так как у текущей вершины бора нет ребёнка = " << string->at(i) << "
,то добавляем этого ребенка в бор" << std::endl;
            bor->emplace_back(Top(num, token));
            bor->at(num).nextV[token] = bor->size() - 1;

        }
        else if (!flag)
            std::cout << "Так как у вершины бора уже есть ребенок " << string->at(i) << ", то
просто переходим на него" << std::endl;
        num = bor->at(num).nextV[token];

    }
    std::cout << "Был достигнут конец строки-шаблона, поэтому последняя вершина помечается
терминальной." << std::endl;
    bor->at(num).isTerminal = true;
    std::cout << "В вектор шаблонов добавляется пройденный шаблон." << std::endl;
    bor->at(num).numberString = numberString;
    std::cout << "Функция добавления шаблона в бор завершает работу!" << std::endl;
}

int getSuff(int top, std::vector<Top>* bor)
{
    std::cout << "Запускается функция получения и установки суффиксоных ссылок!" << std::endl;
    if (bor->at(top).suffLink == -1)
    {
        std::cout << "Из вршины " << bor->at(top).name << " ещё не проведена суффиксная
ссылка!" << std::endl;
        if (top == 0 || bor->at(top).parent == 0)
        {
            std::cout << "Для текущей вершины устанавливаем суффиксную ссылку на начало бора."
<< std::endl;
            bor->at(top).suffLink = 0;
        }
        else
        {
            if (bor->at(top).isTerminal)
            {
                std::cout << "Так как вершина " << bor->at(top).name << " является терминальная
вершина, то для неё будет устанавливаться суффиксная ссылка на начло бора!" << std::endl;
                bor->at(top).suffLink = 0;
            }
            else
            {
                std::cout << "Для текущей вершины " << bor->at(top).name << " устанавливаем
суффиксную ссылку НЕ на начало бора, а на ..." << std::endl;
                bor->at(top).suffLink = getAutoMove(getSuff(bor->at(top).parent, bor), bor-
>at(top).numberSymbol, bor);
                std::cout << "... на вершину " << bor->at(bor->at(top).suffLink).name <<
std::endl;
            }
```

```
        }
    }
    return bor->at(top).suffLink;
}

int getAutoMove(int top, int token, std::vector<Top>* bor)
{
    std::cout << "Запускается функция переходов!" << std::endl;
    if (bor->at(top).autoMove[token] == -1)
    {
        if (bor->at(top).nextV[token] != -1)
        {
            std::cout << "Если есть переход по текущей вершине " << int(bor-
>at(top).numberSymbol) << ", то переходим на неё" << std::endl;
            bor->at(top).autoMove[token] = bor->at(top).nextV[token];
        }
        else if (top == 0)
        {
            std::cout << "Если перехода на текущую вершину нет и при этом текущая вершина в
боре - начльная, то остаемся в начале бора." << std::endl;
            bor->at(top).autoMove[token] = 0;
        }
        else
        {
            std::cout << "Если нет перехода по текущей вершине, переходим по суффиксной ссылке"
<< std::endl;
            bor->at(top).autoMove[token] = getAutoMove(getSuff(top, bor), token, bor);
        }
    }
    std::cout << "Функция переходов завершает работу!" << std::endl;
    return bor->at(top).autoMove[token];
}

void function(std::string& text, std::vector<std::pair<int, int>>* vectorAnswers,
std::vector<Top>* bor, std::vector<std::string>* strings)
{
    std::cout << "Запускается функция поиска шаблона в тексте!" << std::endl;
    int t = 0;
    for (int i = 0; i < text.length(); i++)
    {
        int token;
        if (text[i] == 'A')
            token = 0;
        else if (text[i] == 'C')
            token = 1;
        else if (text[i] == 'G')
            token = 2;
        else if (text[i] == 'T')
            token = 3;
        else if (text[i] == 'N')
            token = 4;
        else
        {
            token = 5;
            std::cout << "Error!" << std::endl;
            break;
        }
        //std::cout << "Так как текущий символ текста " << text[i] << " равен возможным
вариантам алфавита, то..." << std::endl;

        t = getAutoMove(t, token, bor);
        //std::cout << "Выполняется переход на вершину" << bor->at(t).name << std::endl;
        writeResponse(t, i + 1, vectorAnswers, bor, strings);
```

```
    }
}

void writeResponse(int vert, int token, std::vector<std::pair<int, int>>* vectorAnswers,
std::vector<Top>* bor, std::vector<std::string>* strings)
{

    for (int i = vert; i != 0; i = getSuff(i, bor))
    {
        if (bor->at(i).isTerminal)
        {
            std::cout << "В вектор ответа записывается найденное вхождение шаблона!" <<
std::endl;
            vectorAnswers->emplace_back(std::make_pair(token - strings->at(bor-
>at(i).numberString).length() + 1, bor->at(i).numberString + 1));

        }
    }
}


int main()
{
    setlocale(LC_ALL, "Russian");
    std::string text;
    std::vector<std::string> strings;
    strings.resize(0);
    int countStrings = 0;
    int depth = 0;
    std::cout << "Здравствуйте! Пожалуйста, введите текст!" << std::endl;
    std::cin >> text;
    std::cout << "Пожалуйста, введите количество шаблонов!" << std::endl;
    std::cin >> countStrings;
    for (int i = 0; i < countStrings; ++i) {
        std::string string;
        std::cout << "Пожалуйста, введите " << i + 1 << " шаблон!" << std::endl;
        std::cin >> string;
        strings.emplace_back(string);
    }

    std::vector <Top> bor;
    bor.resize(0);
    bor.emplace_back(Top(0, 0));


    std::vector<std::pair<int, int>> vectorAnswers;
    vectorAnswers.resize(0);

    for (int j = 0; j < countStrings; ++j) {
        addString(&bor, &strings[j], j);
    }

    function(text, &vectorAnswers, &bor, &strings);
    //std::cout << "Запускается сортировка вектора ответа!" << std::endl;
    std::sort(vectorAnswers.begin(), vectorAnswers.end(),
        [](std::pair<int, int> a, std::pair<int, int> b)
        {
            if (a.first == b.first) {
                return a.second < b.second;
            }
            else {
                return a.first < b.first;
            }
        });
```

```
    //std::sort(vectorAnswers.begin(), vectorAnswers.end(), comp);
    for (auto answer : vectorAnswers) {
        std::cout << answer.first << " " << answer.second << std::endl;
    }
}
```

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
#include <sstream>

class Top {

private:
    char name;
    std::vector<Top*> childs;
    Top* sufLink;
    Top* parent;
    bool isTerminal;
    bool isFirstElement;
    std::vector<int> numberStrings;

public:

    Top(char name, Top* parent) : name(name), parent(parent) {
        sufLink = nullptr;
        childs.resize(0);
        numberStrings.resize(0);
        isTerminal = false;
        isFirstElement = false;
    }

    Top() {
        name = '&';
        parent = nullptr;
        sufLink = this;
        childs.resize(0);
        numberStrings.resize(0);
        isTerminal = false;
        isFirstElement = true;
    }

    void addChild(Top* child) {
        childs.emplace_back(child);
    }

    void setTerminal(int numberString) {
        isTerminal = true;
        numberStrings.emplace_back(numberString);
    }

    const std::vector<Top*>& getChilds() const {
```

```
        return childs;
    }

    char getName() const {
        return name;
    }

    Top* getSufLink() const {
        return sufLink;
    }

    Top* getParent() const {
        return parent;
    }

    void setSufLink(Top* sufLink) {
        Top::sufLink = sufLink;
    }

    bool isFirstElement1() const {
        return isFirstElement;
    }

    bool isTerminal1() const {
        return isTerminal;
    }

    const std::vector<int>& getNumberStrings() const {
        return numberStrings;
    }


};

Top* createTop(Top* parent, std::string* string, int countElement, int numberString) {//данная
функция создает ребенка для текущей вершины
    std::cout << "Запускается функция создания вершины!" << std::endl;
    auto* element = new Top(string->at(countElement), parent);
    if (countElement == string->length() - 1) {
        std::cout << "Так как вершина " << element->getName() << " является окончанием строки-
шаблона, то она помечается как терминальная!" << std::endl;
        element->setTerminal(numberString);
    }
    parent->addChild(element);
    std::cout << "Функция создания вершины завершает работу!" << std::endl;
    return element;//возвращает ребенка
}

void addString(Top* top, std::string* string, int numberString) {//функция добавляет строку в
бор
    std::cout << "Запускается функция добавления строки в бор!" << std::endl;
    int count = 0;
    Top* cur = top;//изначально cur указывает на начало бора
    while (count < string->length()) {
        char name = cur->getName();
        std::cout << "Текущая вершина: " << name << std::endl;
        bool isFind = false;
        std::vector<Top*> childs = cur->getChilds();
        for (auto child : childs) {
            std::cout << "Так как у вершины " << name << " уже есть ребенок, то выполняется
проход по всем детям вершины:" << std::endl;
            if (child->getName() == string->at(count)) {
                std::cout << "Так как имя ребенка " << child->getName() << " равняется текущему
символу строки-шаблона" << string->at(count) << " , то переходим на ребенка!" << std::endl;
```

```
                cur = child;
                count++;
                isFind = true;
                if (count == string->length()) {
                    std::cout << "Так как счетчик count (" << count << ") равен концу строки-
шаблона (" << string->length() << "), то";
                    std::cout << " добавляем текущую вершину " << cur->getName() << " в
терминальную." << std::endl;
                    cur->setTerminal(numberString);
                }
                break;
            }
        }

        if (!isFind) {
            std::cout << "Так как у текщей вершины нет ребенка, равному вершине " << string-
>at(count) << " то вызывается функция добавления к ней этого ребенка." << std::endl;
            cur = createTop(cur, string, count, numberString);
            count++;
        }
    }
    char name = cur->getName();
    std::cout << "Функция добавления строки в бор завершает работу!" << std::endl;
}


std::vector<std::string>& split(const std::string& s, char delim, std::vector<std::string>&
elems) {//разрезает строку джокера
    std::stringstream ss(s);
    std::string item;
    while (std::getline(ss, item, delim)) {
        if (item.length() > 0) {
            elems.push_back(item);
        }
    }
    return elems;
}

std::vector<std::string> split(const std::string& s, char delim) {//формирует вектор строк
    std::vector<std::string> elems;
    split(s, delim, elems);
    return elems;
}

void createAnswer(Top* bor, std::string text, std::vector<std::string>* strings,
    std::vector<std::pair<int, int>>* vectorAnswers) {

    int count = 0;
    Top* cur = bor;
    while (count < text.length()) {
        bool isExistX = false;
        for (auto child : cur->getChilds()) {
            if (child->getName() == text[count]) {
                cur = child;
                isExistX = true;
                count++;
                if (cur->isTerminal1()) {

                    for (auto numberString : cur->getNumberStrings()) {
                        vectorAnswers->emplace_back(
                            std::make_pair(count - strings->at(numberString).length() + 1,
numberString));
                    }
                }
```

```
                break;
            }
        }
        if (!isExistX) {
            if (cur->isFirstElement1()) {
                count++;
            }
            cur = cur->getSufLink();
        }
    }

}

int main() {
    setlocale(LC_ALL, "Russian");
    std::string text;
    std::vector<std::string> strings;
    strings.resize(0);
    int countStrings = 0;


    std::string my_str;
    char symbol;

    std::cout << "Здравствуйте! Введите, пожалуйста, текст!" << std::endl;
    std::cin >> text;
    std::cout << "Введите, пожалуйста, шаблон!" << std::endl;
    std::cin >> my_str;
    std::cout << "Введите, пожалуйста, символ джокера!" << std::endl;
    std::cin >> symbol;
    std::cout << "Запускается функция разделения шаблона!" << std::endl;
    strings = split(my_str, symbol);//разделяем строку
    std::cout << "Строка ПОСЛЕ разделения: " << std::endl;

    countStrings = strings.size();
    for (int j = 0; j < countStrings; ++j) {
        std::cout << strings[j] << std::endl;
    }
    auto* bor = new Top;

    for (int j = 0; j < countStrings; ++j) {
        addString(bor, &strings[j], j);
    }

    std::queue<Top*> queue;//создаем очередь
    for (auto local : bor->getChilds()) {//проход по всем детям бора
        local->setSufLink(bor);//ставим суффиксную ссылку на начало бора(&)
        for (auto local2 : local->getChilds()) {
            std::cout << "Вершина: " << local2->getName() << " теперь в queue!" << std::endl;
            queue.push(local2);
        }

    }

    while (!queue.empty()) {
        char name = queue.front()->getName();
        std::cout << "Текущая вершина: " << name << std::endl;
        Top* element = queue.front()->getParent();
        bool isExistX = false;//флаг окончания прохода

        while (!isExistX) {
            element = element->getSufLink();//делаем шаг назад(отходим на предыдущую вершину)
            for (auto child : element->getChilds()) {//проходимся по детям вершины
```

```
                if (child->getName() == name) {//если есть совпадение с текущей вершиной из
очереди
                    isExistX = true;
                    element = child;
                    break;
                }
            }
            if (element->isFirstElement1() && !isExistX) {//если не было совпадений с вершиной
из очереди
                isExistX = true;
            }



        }
        queue.front()->setSufLink(element);//ставим суффиксную ссылку на element
        std::cout << "Ставим ссуфиксную ссылку вершины: " << name << " на вершину: " <<
element->getName() << std::endl;
        if (queue.front()->getSufLink()->isTerminal1()) {//проверка если она терминальная
            for (int numberString : queue.front()->getSufLink()->getNumberStrings()) {

                queue.front()->setTerminal(numberString);
            }
        }
        //char dfd = queue.front()->getSufLink()->getName();
        for (auto child : queue.front()->getChilds()) {//если есть ребенок у текущей вершины из
очереди, то записываем его в неё
            std::cout << "Добавляем вершину " << child->getName() << " в очередь queue!" <<
std::endl;
            queue.push(child);
        }
        std::cout << "Удаляем верхнюю вершину " << name << " из очереди queue!" << std::endl;
        queue.pop();
    }


    std::vector<std::pair<int, int>> vectorAnswers;
    vectorAnswers.resize(0);

    std::vector<std::pair<int, int>> vectorAnswers2;
    vectorAnswers.resize(0);

    int count2 = 0;

    for (int i = 0; i < my_str.length(); ++i) {//находим вхождение подстрок в шаблоне
        if (my_str[i] == strings[count2][0]) {
            int count = 0;
            for (int j = 0; j < strings[count2].length(); ++j) {
                if (strings[count2][j] == my_str[i + count]) {
                    count++;
                }
            }
            if (count == strings[count2].length()) {
                vectorAnswers2.emplace_back(i, count2);//
                i = i + count;
                if (count2 < strings.size() - 1) {
                    count2++;
                }
            }
        }
    }

    createAnswer(bor, text, &strings, &vectorAnswers);
```

```
    int* array = new int[text.size()];//создаем массив и заполняем его нулями
    for (int i = 0; i < text.size(); ++i) {
        array[i] = 0;
    }

    for (auto answer : vectorAnswers) {

        int number = answer.first - vectorAnswers2[answer.second].first - 1;
        if (number >= 0) {
            array[number]++;
        }
    }

    std::vector<int> finalAnswers;
    for (int i = 0; i < text.size(); ++i) {
        if (array[i] == strings.size()) {
            if (i + my_str.size() <= text.size()) {
                //std::cout << i + 1 << "\n";
                finalAnswers.emplace_back(i+1);//массив вхождения шаблонов текста
            }
        }

    }

    int cur = 0;//номер текущего вхождения
    int lenghtShablon = my_str.size();//длина шаблона
    for (int k = 1; k < finalAnswers.size(); k++)
    {
        if ((finalAnswers[cur] + lenghtShablon-1 >= finalAnswers[k]) && finalAnswers[k] != -1)
            finalAnswers[k] = -1;
        else
        {
            cur = k;
        }

    }


    std::cout << "Вывод ответа:" << std::endl;

    for (int h = 0; h < finalAnswers.size(); h++)
    {
        if (finalAnswers[h] != -1)
        {
            std::cout << finalAnswers[h] << std::endl;
        }

    }



    delete[] array;

    return 0;
}
```