Реализация конечного автомата для регулярного языка

Выполнила:

Штоколова Елизавета Викторовна.

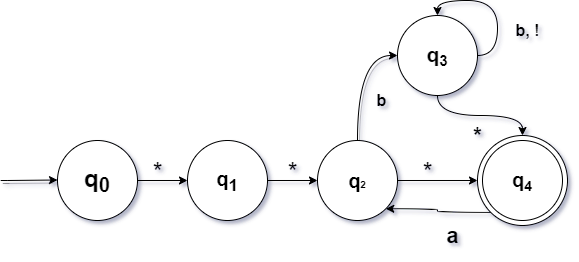
233 группа.

1. **Задание**

Строка символов a,b,\*,!,начинающаяся с префикса \*\*и заканчивающаяся суффиксом \*, между которыми располагается последовательность символов a,b(может быть и пустой),в которой после каждого символа b следует последовательность символов !(возможно пустая),а каждый символ a предваряется символом \*(пример:\*\*\*или \*\*b!\*a\*abb!!\*a\*).

1. **Диаграмма переходов и таблица переходов КА**

**Диаграмма переходов.**



**Ваш автомат допустит цепочку \*\*a\* (q0-q1-q2-q4-q5), но она не принадлежит языку Вашего задания.**

**Таблица переходов. Если Вы построили ДКА, то таблица переходов не полная!, а если НКА, то значением функции перехода должны быть множества.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | \* | a | b | ! |
| → **q0** | q1 | ∅ | ∅ | ∅ |
| q1 | q2 | ∅ | ∅ | ∅ |
| q2 | q4 | ∅ | q3 | ∅ |
| q3 | q4 | ∅ | q3 | q3 |
| \*q4 | ∅ | q2 | ∅ | ∅ |
| ∅ | ∅ | ∅ | ∅ | ∅ |

1. **Регулярное выражение.**

Обозначим символ \* за ×.

× × (bb\*!\* ×+×)a

**4,5. Листинги программ (с обязательными комментариями!).**

**Краткое описание работы подпрограммы, реализующий КА, и программы поиска цепочек языка в последовательности символов**

**Задание:**  
1.Разработать распознающий КА для цепочек определенного в задании регулярного языка над заданным алфавитами реализовать его в виде подпрограммы.

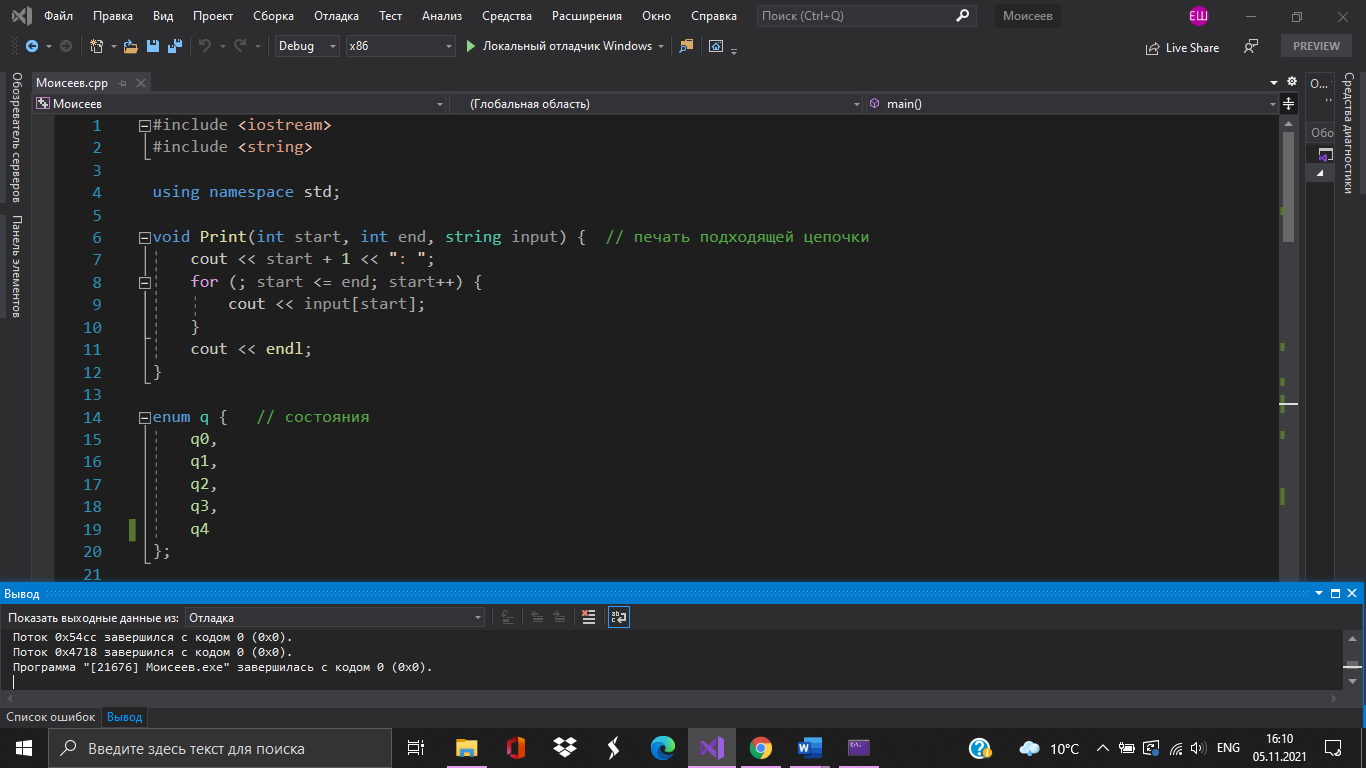
2.Разработать программу поиска цепочек этого языка в последовательности символов, используя подпрограмму пункта 1.

* Вход: последовательность символов алфавита регулярного языка задания
* Выход: последовательность строк формата номер: цепочка номер–номер позиции начала цепочки языка в последовательности символов цепочка–найденная цепочка языка в последовательности символов. Если цепочек языка не найдено, то на выходе отображается «цепочек не найдено»

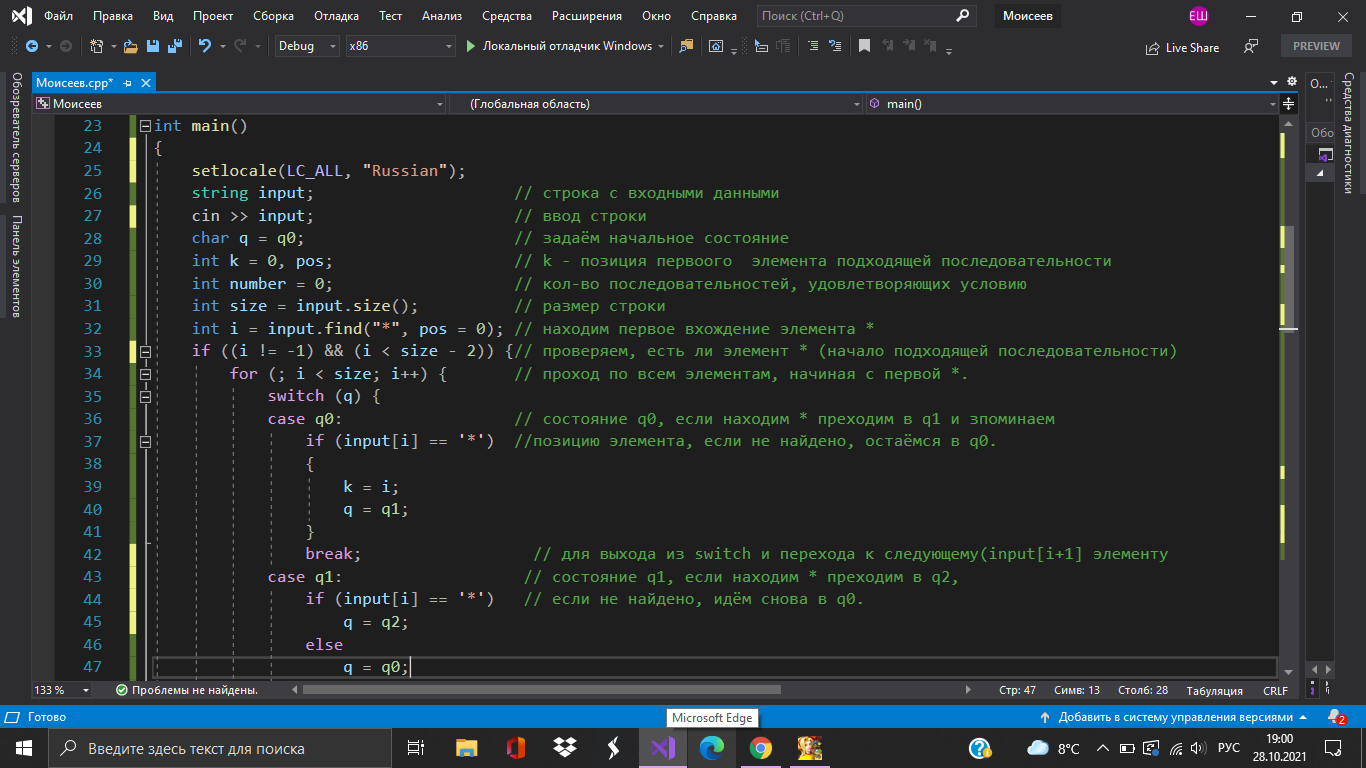
**Решение:**

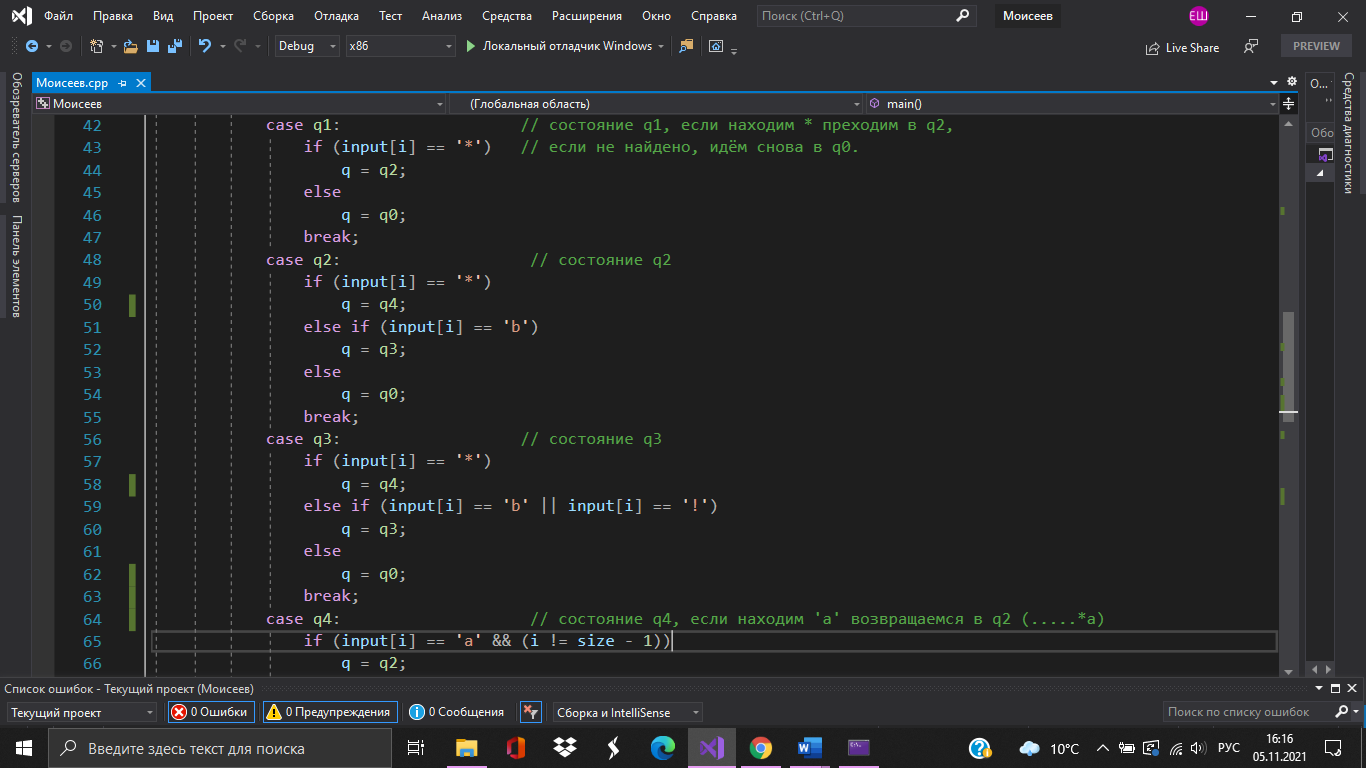
Программа на языке с++.

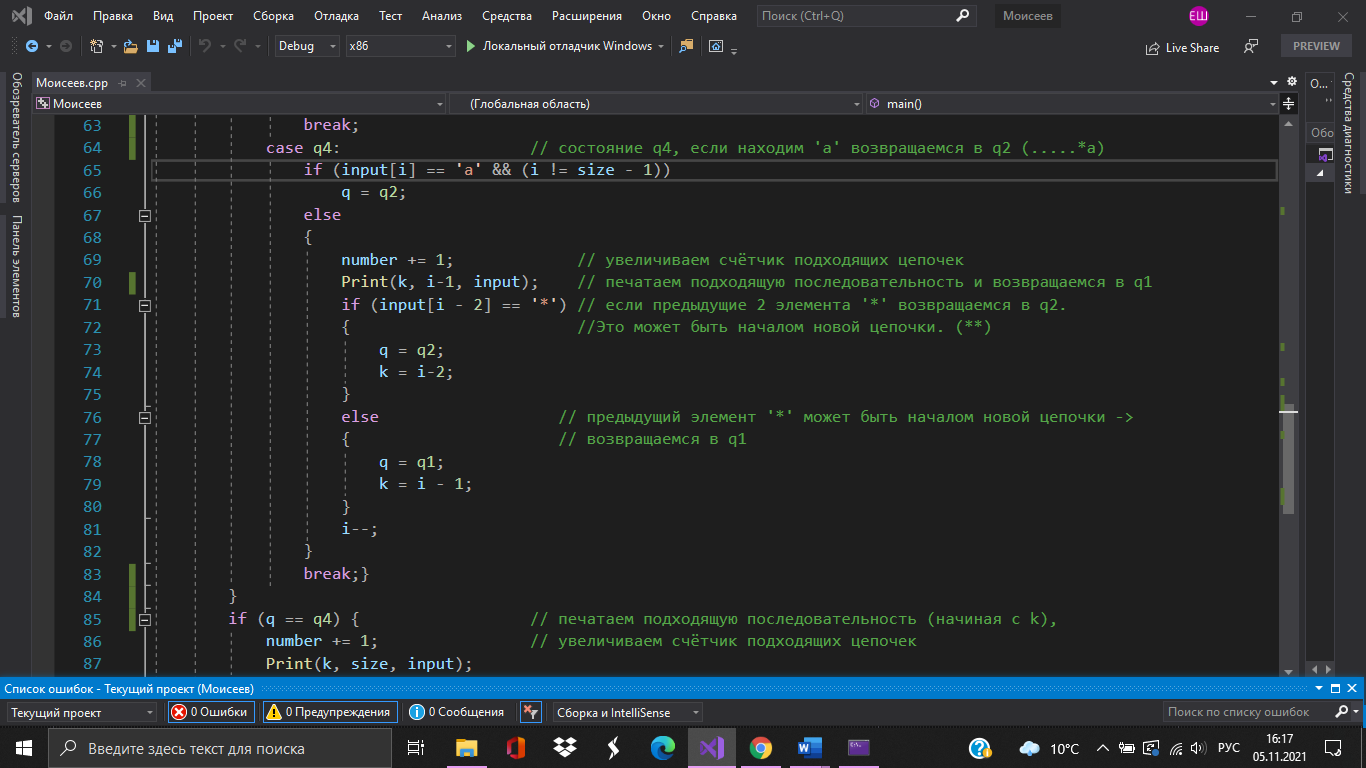
Перед функцией int main() реализуем функцию Print, которая печатает подходящие цепочки, объявим в enum q {} состояния (согласно таблице переходов), чтобы далее использовать их в switch при реализации автомата:



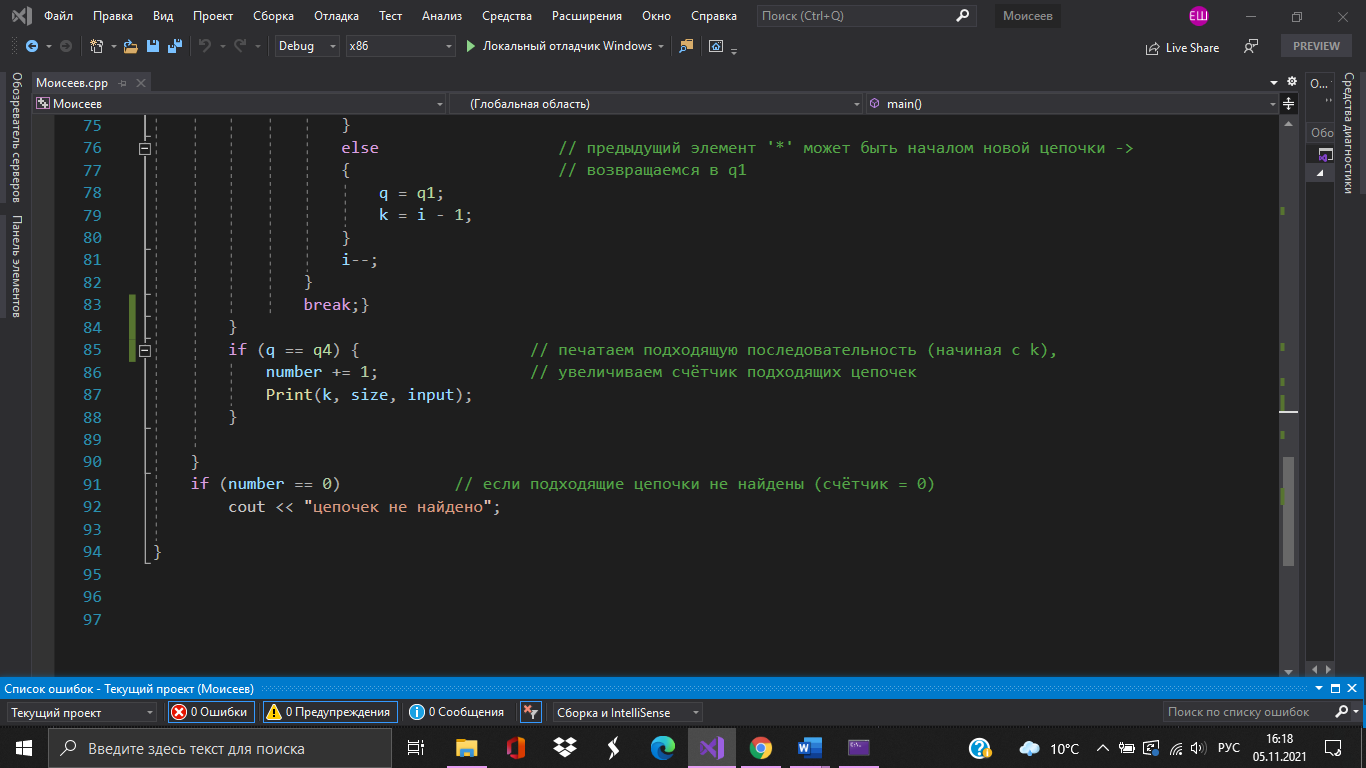
Далее получим входные данные, реализуем автомат и выведем нужные цепочки на экран.

Входные данные записываем в переменную типа string. Значение переменных и пояснения расписаны в комментариях. С помощью switch реализуется работа автомата. Case реализует каждое состояние (q0,q1 и т.д.) согласно таблице переходов (т.е. первую строку таблицы переходов). В каждом case рассматриваются строки таблицы переходов для конкретного состояния с помощью if, if else, else. Если условия выполняются, переходим в следующее состояние согласно таблице (например, текущий элемент: “\*”, из текущего состояния q1 автомат переходит в q2). Если попадаем на ∅ в таблице, возвращаемся в q0, данная цепочка нам не подходит. 





Обрабатываем ситуацию, когда цепочка находится в конце или не найдена:



**6. Таблица тестирования программы (10-15 тестов): вход и выход**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **\*\*\*** | **1: \*\*\*** |
| **saghgdgh** | **цепочек не найдено** |
| **\*\*a\*** | **цепочек не найдено** |
| **\*\*\*b!\*a\*a\*** | **1: \*\*\***  **2: \*\*b!\*a\*a\*** |
| **\*\*\*\*** | **1: \*\*\***  **2: \*\*\*** |
| **\*\*!b\*a\*a\*** | **цепочек не найдено** |
| **\*b!\*a\*** | **цепочек не найдено** |
| **\*\*b!a\*** | **цепочек не найдено** |
| **\*\*\*\*b!\*\*a\*a\*abbb!!\*** | **1: \*\*\***  **2: \*\*\***  **3: \*\*b!\*** |
| **\*\*sdudya\*** | **цепочек не найдено** |
| **\*\*a\*ab!\*a** | **цепочек не найдено** |
| **\*\*\*ab!\*a\*b** | **1: \*\*\*ab!\*a\*** |