

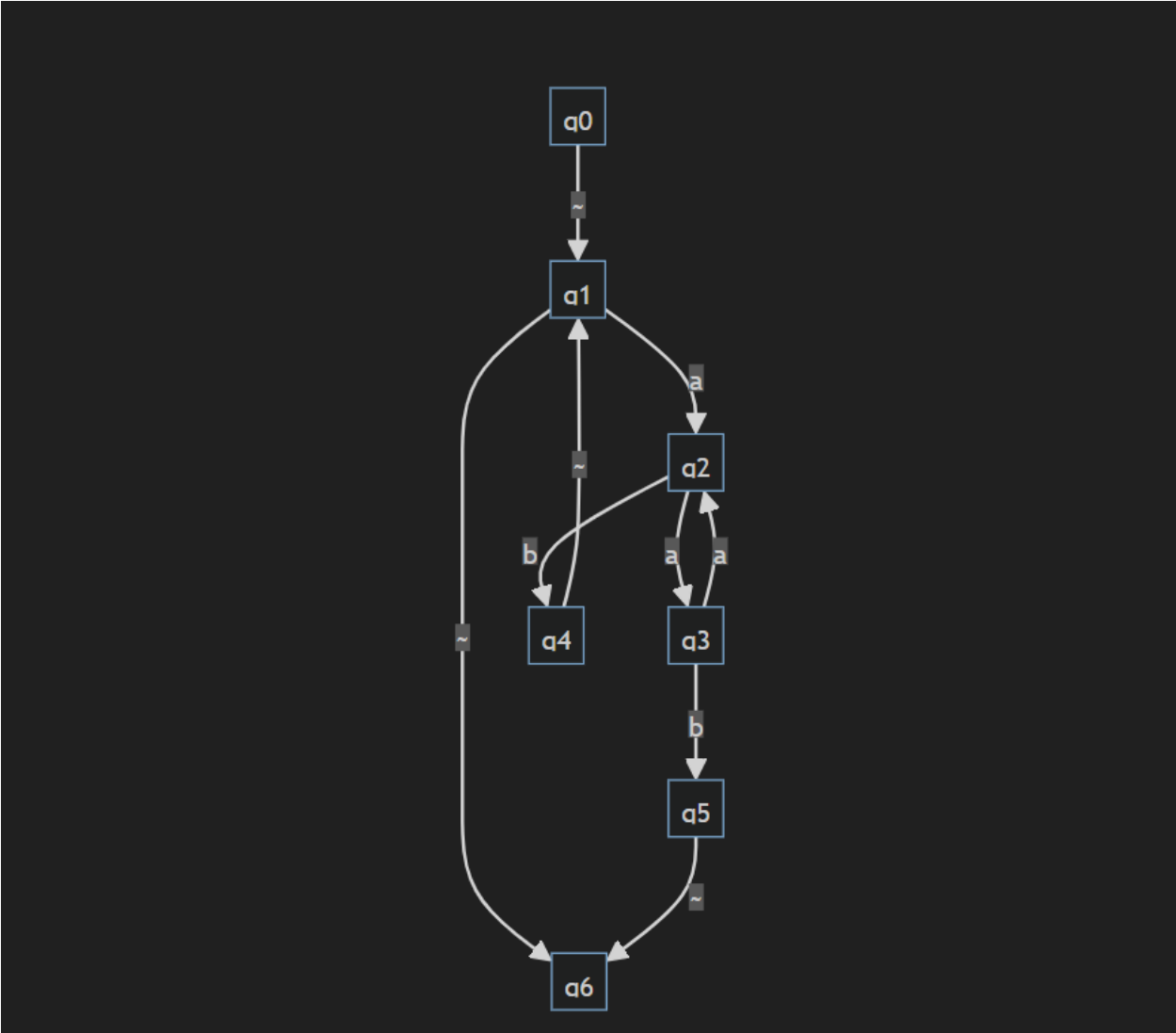
# Реализация КА для регулярного языка

Кручинин Евгений Владимирович

22.Б12-пу

**Задание:** Разработать распознающий КА для цепочек начинающихся с префикса ~ и заканчивающихся суффиксом ~, между которыми располагается чётное число подряд идущих a, за которыми следует один символ b, или нечетное число подряд идущих a, за которыми следует два символа b.

**Диаграмма переходов:**




**Таблица переходов:**

	~	a	b
→ q0	q1	∅	∅
q1	q6	q2	∅
q2	∅	q3	q4
q3	∅	q2	q5

	~	a	b
q4	∅	∅	q1
q5	q6	q2	∅
*q6	∅	∅	∅
∅	∅	∅	∅

Регулярное выражение:



~((aa)+b|a+bb)\*~

Листинги программ:

```
#include <iostream>
#include <string>

//Таблица переходов      ~   a   b
static const int table[7][3] = { {1, -1, -1}, //q0
                                  {6, 2, -1}, //q1
                                  {-1, 3, 4}, //q2
                                  {-1, 2, 5}, //q3
                                  {-1, -1, 1}, //q4
                                  {6, 2, -1}, //q5
                                  {-1, -1, -1} }; //q6

void find(const std::string& str) {
    bool check = false; //проверяем, что хоть один вход был
    for (int index = 0; index < str.length(); ++index) //идем циклом по всей строке
    {
        std::string result; //будем записывать сюда подходящую цепочку
        int ind = index; //заводим еще один индекс, чтобы итерироваться по строке
        int state = 0; //состояние
        while (state != -1) //пока не перешли в пустое состояние
        {
            //проверяем на нужные символы, иначе ловим else
            if (str[ind] == '~' || str[ind] == 'a' || str[ind] == 'b')
                state = table[state][static_cast<int>(str[ind]) % 3]; //задаем состояние
            result += str[ind]; //добавляем текущий элемент к цепочке
            ++ind; //увеличиваем ind, двигаясь по строке
            if (state == 6) //проверяем, что получили может правильную
            {
                //выводим индекс первого символа и саму цепочку
                std::cout << index + 1 << ": " << result << std::endl;
                check = true; //хоть одна цепочка есть
            }
        }
    }
    else {
        state = -1; //переходим в пустое состояние
    }
}

//проверка, что могли не найти ни одной цепочки
if (!check) {
    std::cout << "цепочек не найдено!" << std::endl;
}

int main() {
    std::string input; //заводим переменную для нашей строки

    std::cout << "Введите строку:" << std::endl; //просим пользователя ввести строку
```

```
std::cin >> input; //принимаем строку

find(input); //передаем строку на обработку

return 0;
}
```

**Краткое описание работы подпрограммы:**

- Будем передавать строку в функцию. Введем переменную состояния(0- начальное) и таблицу переходов. Будем идти циклом, пока не кончится строка, и проверять принадлежность. По таблице переходов мы будем менять начальное состояние, которое изначально равно 0. Если получаем состояние 6, то значит перейдем в **финальное состояние** , после этого у нас выведется индекс первого элемента и сама строка.
- При определение состояния пользуюсь остатком от деления на 3, чтобы итерироваться по столбцам таблицы, значения символов '~' = 126, 'a' = 97, 'b' = 98, остаток от деления будет соответственно 0, 1, 2.
- Так как нам надо проверить еще, что хоть одна строка была, я использую переменную check, изначально она false, но при первом же нахождение нужной цепочки она меняется на true. Если за всю строку мы не нашли ни одной нужной цепочки, то будет выведено **“цепочек не найдено!”**

**Таблица тестирования программы:**

Номер теста	Ввод	Вывод
1.	~abbb~	цепочек не найдено!
2.	~aabb~	цепочек не найдено!
3.	~abbb~abb~	6: ~abb~
4.	~abb~	1: ~abb~
5.	~aab~abb~	1: ~aab~; 5: ~abb~
6.	~abbb~ab~	цепочек не найдено!
7.	~a~	цепочек не найдено!
8.	~abbaab~	1: ~abbaab~
9.	~abb~abbb~	1: ~abb~
10.	~abbaab~abb~	1: ~abbaab~; 8: ~abb~
11.	~b~	цепочек не найдено!
12.	~abb~a~	1: ~abb~
13.	~aab~b~	1: ~aab~
14.	~abbb~b~	цепочек не найдено!
15.	~~abb~	1: ~~; 2: ~abb~