Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 13

По дисциплине «Конструирование программного обеспечения»

Выполнил:

Студент 1 курса 9 группы

Бондарик Никита

Преподаватель: Панченко Ольга Леонидовна

2024, Минск

Вариант 2

Регулярное выражение:



# 1) Примеры цепочек символов:

1. open + write + read + seek + close
2. open + write + read + close
3. open + read + write + read + close
4. open + write + seek + write + read + close
5. open + seek + write + read + close
6. open + write + write + read + close
7. open + read + read + seek + close

# 2) Диаграмма мгновенных состояний конечного автомата для цепочки open + write + read + close:

1. Состояние q0 - Начальное состояние. Чтение open переводит в состояние q1.
2. Состояние q1 - Чтение write переводит в состояние q2.
3. Состояние q2 - Чтение read переводит в состояние q3.
4. Состояние q3 - Чтение close переводит в конечное состояние q4.
5. Состояние q4 - Конечное состояние (автомат принимает цепочку).

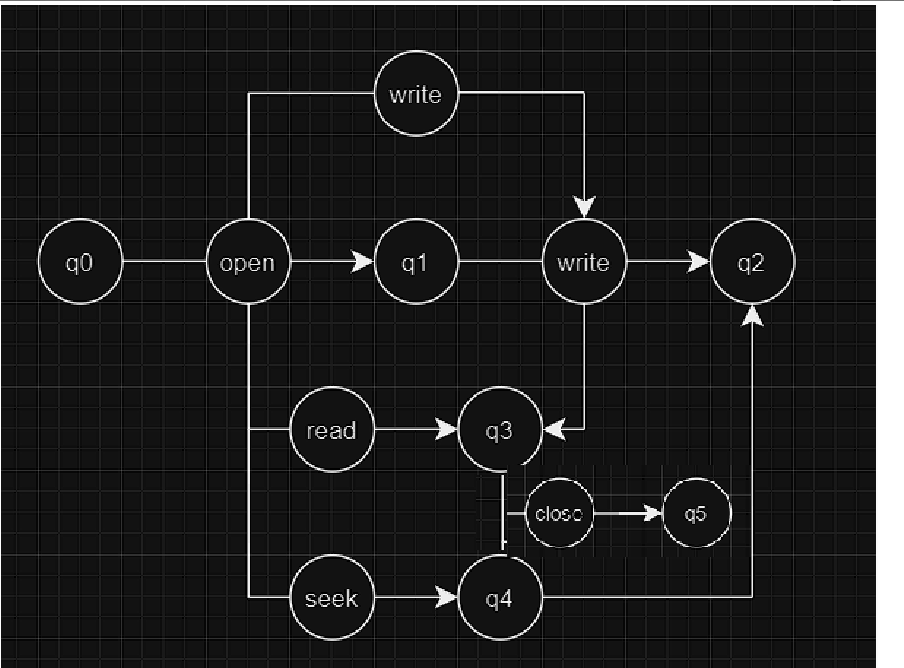
|  |
| --- |
| q0 --open--> q1 --write--> q2 --read--> q3 --close--> q4 |

# 3) Граф конечного автомата, распознающего цепочки, описанные регулярным выражением:

Мы можем построить конечный автомат, используя состояния и переходы для описанных цепочек. Вот автомат:

* Состояние q0: Начальное состояние. open переводит в состояние q1.
* Состояние q1: write переводит в состояние q2. read переводит в состояние q3. seek переводит в состояние q4.
* Состояние q2: write переводит в состояние q2 (петля). read переводит в состояние q3. seek переводит в состояние q4.
* Состояние q3: write переводит в состояние q2. read переводит в состояние q3 (петля). seek переводит в состояние q4.
* Состояние q4: write переводит в состояние q2. read переводит в состояние q3. seek переводит в состояние q4 (петля).
* Состояние q3 и q4: close переводит в конечное состояние q5.
* Состояние q5: Конечное состояние (автомат принимает цепочку).

Диаграмма конечного автомата:



Этот график конечного автомата распознает цепочки, описанные регулярным выражением open( (write | read | seek) + )+ close.

## 1. Что такое алфавит I?

Алфавит I — это конечное множество символов, используемых для построения цепочек (слов) в языках.

## 2. Поясните обозначения λ, ∗, +:

λ — пустая строка.

∗∗ — оператор Клини, означающий ноль или более повторений символа или подмножества.

++ — один или более повторений символа или подмножества.

## 3. Что такое язык L(I) над алфавитом I?

Язык 𝐿(𝐼)L(I) над алфавитом 𝐼I — это множество всех возможных цепочек, составленных из символов алфавита 𝐼I.

## 4. Определение формальной грамматики G:

Формальная грамматика G определяется как четверка G=(N,T,P,S), где:

* N — конечное множество нетерминалов.
* T — конечное множество терминалов (алфавит).
* P — конечное множество продукций (правил вывода).
* S — начальный символ (аксиома).

## 5. Поясните обозначения 𝛼⇒𝛽α⇒β и 𝛼⇒∗𝛽α⇒∗β:

* 𝛼⇒𝛽α⇒β означает, что цепочка β может быть непосредственно получена из цепочки α с помощью одного применения правила вывода.
* 𝛼⇒∗𝛽α⇒∗β означает, что цепочка β может быть получена из цепочки α с помощью конечного числа применений правил вывода.

## 6. Что такое язык L(G), порождаемый грамматикой G?

Язык L(G), порождаемый грамматикой G — это множество всех цепочек, которые могут быть выведены из начального символа S с помощью правил грамматики P.

## 7. Что такое форма Бэкуса-Наура?

Форма Бэкуса-Наура (BNF) — это нотация для формального описания синтаксиса языков программирования и других формальных языков, использующая рекурсивные правила.

## 8. Понятие «регулярная грамматика»:

Регулярная грамматика — это грамматика, в которой все правила вывода имеют одну из следующих форм: 𝐴→𝑎𝐵A→aB или 𝐴→𝑎A→a, где 𝐴A и 𝐵B — нетерминалы, а 𝑎a — терминал.

## 9. Понятие «регулярное множество»:

Регулярное множество — это множество, которое может быть представлено регулярным выражением.

## 10. Понятие «регулярный язык»:

Регулярный язык — это язык, который может быть распознан конечным автоматом или описан регулярной грамматикой или регулярным выражением.

## 11. Что такое лексический анализ?

Лексический анализ — это процесс разбора входного текста на токены (лексемы), которые являются элементарными единицами смысла, используемыми для синтаксического анализа.

## 12. Что такое лексический анализатор?

Лексический анализатор (или сканер) — это программа или модуль, который выполняет лексический анализ, преобразуя входной текст в последовательность токенов.

## 13. Входная и выходная информация для лексического анализатора:

* Входная информация: текст исходного кода или другой текстовый ввод.
* Выходная информация: последовательность токенов.

## 14. Различия между последовательным и параллельным лексическими анализаторами:

* Последовательный лексический анализатор обрабатывает входной текст линейно, шаг за шагом.
* Параллельный лексический анализатор может использовать многопоточность или распараллеливание для ускорения процесса анализа.

## 15. Определение регулярного выражения над алфавитом I:

Регулярное выражение над алфавитом I — это формальное выражение, описывающее множество строк, которые можно построить из символов алфавита 𝐼I с использованием операций объединения, конкатенации и замыкания Клини.

## 16. Определение конечного автомата M=(S,I,δ,s0​,F):

Конечный автомат M определяется как пятерка:

* S — конечное множество состояний.
* I — алфавит (множество входных символов).
* 𝛿 — функция переходов 𝛿:𝑆×𝐼→𝑆.
* 𝑠0​ — начальное состояние.
* F — множество конечных состояний.

## 17. Отличие между детерминированным и недетерминированным автоматом:

* Детерминированный конечный автомат (DFA) имеет одну функцию перехода 𝛿 для каждой пары (состояние, символ).
* Недетерминированный конечный автомат (NFA) может иметь множество переходов для одной пары (состояние, символ), включая переходы без входного символа (ε-переходы).

## 18. Мгновенное состояние конечного автомата:

Мгновенное состояние конечного автомата — это пара (𝑠,𝑤), где s — текущее состояние, а w — оставшаяся часть входной строки.

## 19. Поясните обозначения (𝑠,𝑎𝑤)⊢(𝑠′,𝑤)(s,aw)⊢(s′,w) и (𝑠𝑖,𝑤)⊢∗(𝑠𝑘,𝑤𝑘)(si​,w)⊢∗(sk​,wk​):

* (𝑠,𝑎𝑤)⊢(𝑠′,𝑤)(s,aw)⊢(s′,w) означает, что автомат переходит из состояния 𝑠s в состояние 𝑠′s′ при обработке символа 𝑎a, оставляя 𝑤w как оставшуюся строку.
* (𝑠𝑖,𝑤)⊢∗(𝑠𝑘,𝑤𝑘)(si​,w)⊢∗(sk​,wk​) означает, что автомат может перейти из состояния 𝑠𝑖si​ в состояние 𝑠𝑘sk​ за несколько шагов, обрабатывая строку 𝑤w.

## 20. Соотношение между регулярной грамматикой, регулярным языком, регулярным выражением, конечным автоматом, графом состояний конечного автомата:

* Регулярная грамматика, регулярное выражение, конечный автомат и граф состояний конечного автомата — эквивалентные способы описания регулярных языков.
* Регулярный язык может быть описан любой из этих формализаций: регулярной грамматикой, регулярным выражением, конечным автоматом или графом состояний автомата.