Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа №13

По дисциплине «Конструирование программного обеспечения»

На тему: «Подготовка к разработке лексического распознавателя»

Выполнила:

Студентка1 курса 9 группы  
Волосюк Арина Вадимовна

Преподаватель: Панченко О. Л.

2024, Минск

Вариант 6:

proc(□)+((abs()|sin(x+));)\* □+end

Примеры цепочек символов:  
1. procabs(□); sin(x+); □end

2. procsin(x+); sin(x+); □end

3. procabs(□); sin(x+);abs();sin(x+); □end

4. procabs(□); □end

5. proc(□); sin(x+);sin(x+);sin(x+); □end

6. procabs(□); sin(x+);sin(x+);sin(x+); □end

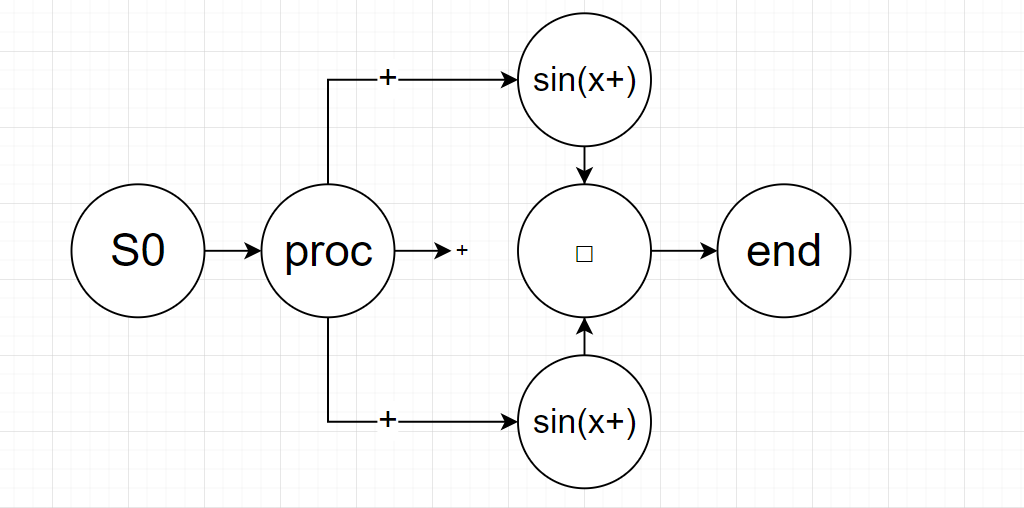
7. procabs(□); abs();sin(x+);sin(x+); □end

Диаграмма мгновенный состояний конечного автомата для данной цепочки:

procsin(x+); sin(x+); □end

|  |  |  |  |
| --- | --- | --- | --- |
| procsin(x+) | sin(x+) | □ | end |
| a | b | c | d |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | a | b | c | d |
| S0 | S1 | ∅ | ∅ | ∅ |
| S1 | ∅ | S2 | ∅ | ∅ |
| S2 | ∅ | ∅ | S3 | ∅ |
| S3 | ∅ | ∅ | ∅ | S4 |
| S4 | ∅ | ∅ | ∅ | ∅ |

***Граф конечного автомата:***

### 1. Что такое алфавит I?

Алфавит \( I \) – это конечное множество символов, используемых для построения цепочек (слов).

### 2. Обозначения \(\lambda\), \(I^\*\), \(I^+\)

- \(\lambda\): Пустая строка, то есть строка длины ноль.

- \(I^\*\): Множество всех возможных строк, которые можно составить из символов алфавита \( I \), включая пустую строку.

- \(I^+\): Множество всех возможных строк, которые можно составить из символов алфавита \( I \), за исключением пустой строки.

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

Язык \( L(I) \) над алфавитом \( I \) – это множество строк (слов), составленных из символов алфавита \(

### 4. Определение формальной грамматики \( G \)

Формальная грамматика \( G \) определяется как четверка \( (N, \Sigma, P, S) \), где:

- \( N \) – конечное множество нетерминальных символов.

- \( \Sigma \) – конечное множество терминальных символов (алфавит), \( N \cap \Sigma = \emptyset \).

- \( P \) – конечное множество правил продукций вида \( \alpha \rightarrow \beta \), где \( \alpha \) и \( \beta \) – строки из \( (N \cup \Sigma)^\* \).

- \( S \) – стартовый нетерминал из \( N \).

### 5. Обозначения \(\alpha \Rightarrow \beta\) и \(\alpha \Rightarrow^\* \beta\)

- \(\alpha \Rightarrow \beta\): Цепочка \( \beta \) может быть получена из цепочки \( \alpha \) применением одного правила продукции.

- \(\alpha \Rightarrow^\* \beta\): Цепочка \( \beta \) может быть получена из цепочки \( \alpha \) применением конечного числа правил продукции (возможно, нуля).

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

Язык \( L(G) \), порождаемый грамматикой \( G \) – это множество всех строк из \( \Sigma^\* \), которые могут быть получены из стартового символа \( S \) путём конечного числа применений правил продукций грамматики \( G \).

### 7. Что такое форма Бэкуса-Наура?

Форма Бэкуса-Наура (BNF) – это нотация для спецификации синтаксиса языков программирования и формальных языков. В BNF используются следующие элементы:

- Нетерминалы заключены в угловые скобки \( \langle \rangle \).

- Терминалы пишутся как есть.

- Правила продукции имеют вид \( \langle A \rangle ::= \alpha \), где \( \langle A \rangle \) – нетерминал, а \( \alpha \) – последовательность терминалов и/или нетерминалов.

### 8. Регулярная грамматика

Регулярная грамматика – это грамматика, в которой все правила продукций имеют один из двух видов:

### 9. Регулярное множество

Регулярное множество – это множество строк, описываемое регулярным выражением

### 10. Регулярный язык

Регулярный язык – это язык, который может быть распознан детерминированным или недетерминированным конечным автоматом, либо описан регулярной грамматикой или регулярным выражением.

### 11. Лексический анализ

Лексический анализ – это процесс преобразования входного потока символов в последовательность токенов (лексем),

### 12. Лексический анализатор

Лексический анализатор (сканер) – это программа или часть компилятора, которая выполняет лексический анализ, разбивая исходный код на токены.

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

- Входная информация: Исходный текст программы.

- Выходная информация: Последовательность токенов, которая будет передана синтаксическому анализатору.

### 14. Последовательные и параллельные лексические анализаторы

- Последовательный лексический анализатор обрабатывает входной поток символов последовательно, один за другим.

- Параллельный лексический анализатор может одновременно обрабатывать разные части входного потока

### 15. Регулярное выражение над алфавитом \( I \)

Регулярное выражение над алфавитом \( I \) – это выражение, описывающее множество строк, составленных из символов \( I \), с использованием операций объединения (\(|\)), конкатенации и замыкания (\(\*\)).

### 16. Конечный автомат \( M = (S, I, \delta, s\_0, F) \)

Конечный автомат определяется как пятёрка:

- \( S \) – конечное множество состояний.

- \( I \) – алфавит (множество входных символов).

- \( \delta \) – функция переходов \( \delta: S \times I \rightarrow S \).

- \( s\_0 \) – начальное состояние \( s\_0 \in S \).

- \( F \) – множество заключительных (допустимых) состояний \( F \subseteq S \).

### 17. Отличие между детерминированным и недетерминированным автоматом

- Детерминированный конечный автомат (DFA): Для каждого состояния и каждого символа входного алфавита имеется ровно один переход.

- Недетерминированный конечный автомат (NFA): Для каждого состояния и каждого символа входного алфавита может быть несколько переходов, включая отсутствие перехода.

### 18. Мгновенное состояние конечного автомата

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

### 19. Обозначения \((s, aw) \phi (s', w)\) и \((s\_i, w\_i) \phi^\* (s\_k, w\_k)\)

- \((s, aw) \phi (s', w)\): Из состояния \( s \), прочитав символ \( a \), автомат переходит в состояние \( s' \), оставив \( w \) как оставшуюся часть входной строки.

- \((s\_i, w\_i) \phi^\* (s\_k, w\_k)\): За несколько шагов автомат переходит из состояния \( s\_i \), имея строку \( w\_i \), в состояние \( s\_k \), оставив строку \( w\_k \).

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

- Регулярная грамматика, регулярный язык, регулярное выражение, конечный автомат и граф состояний конечного автомата – это взаимосвязанные концепции.

- Регулярные выражения описывают регулярные языки.

- Регулярные грамматики порождают регулярные языки.

- Конечные автоматы распознают регулярные языки.

- Граф состояний конечного автомата визуально представляет переходы между состояниями в конечном автомате.

Эти элементы теории формальных языков и автоматов тесно связаны и часто используются взаимозаменяемо для анализа и описания языков.