Лабораторная работа №1. Лексический анализатор: распознавание чисел и идентификаторов

Имеется несколько типов токенов в любом формальном языке. Символьное имя, заданное пользователем для некоторых частей программы, таких как функции и переменные, называется идентификатор. Затем идут ключевые слова, например, в языке C ++ известны такого рода слова, как cout, cin, if, else, for, break, continue и т. д. Знаки пунктуации используются для построения выражений и утверждений. Они используются в операторе языка вместе с идентификаторами или ключевыми словами. Операторы предназначены для выполнения реальных операций с данными, таких как арифметические, логические операции, операции сдвига и.т.д. Литералы - это постоянные данные, атомы, с которыми должны работать программы, например, из литералов складываются числа, слова, идентификаторы или прочие комбинации символов над заданным алфавитом.

Цель лабораторной работы

состоит в написании и программной реализации лексического анализатора, который разделяет входной поток на токены и классифицирует их в поиске чисел и идентификаторов. Попутно лексический анализатор распознает ключевые слова, знаки операций и символы пунктуации.

Вход:

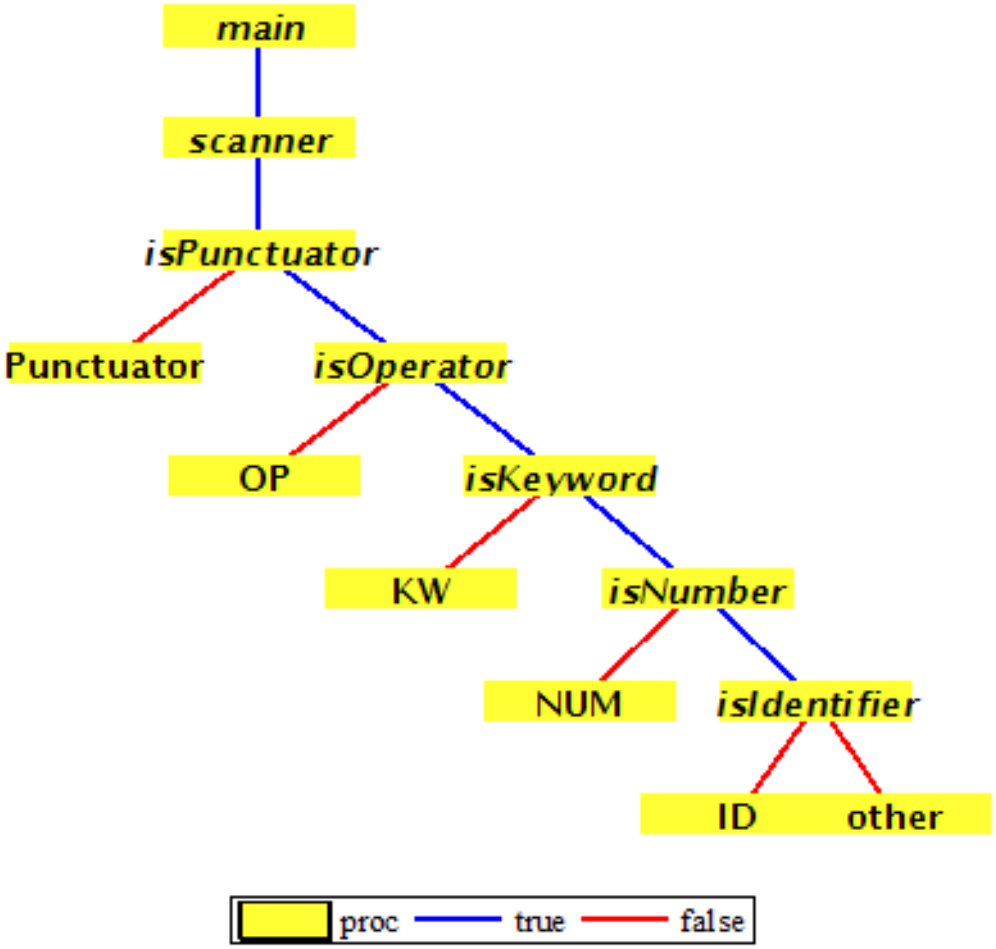
входной поток символов.

Выход:

список распознанных идентификаторов, чисел, ключевых слов, знаков операций и символов пунктуации.

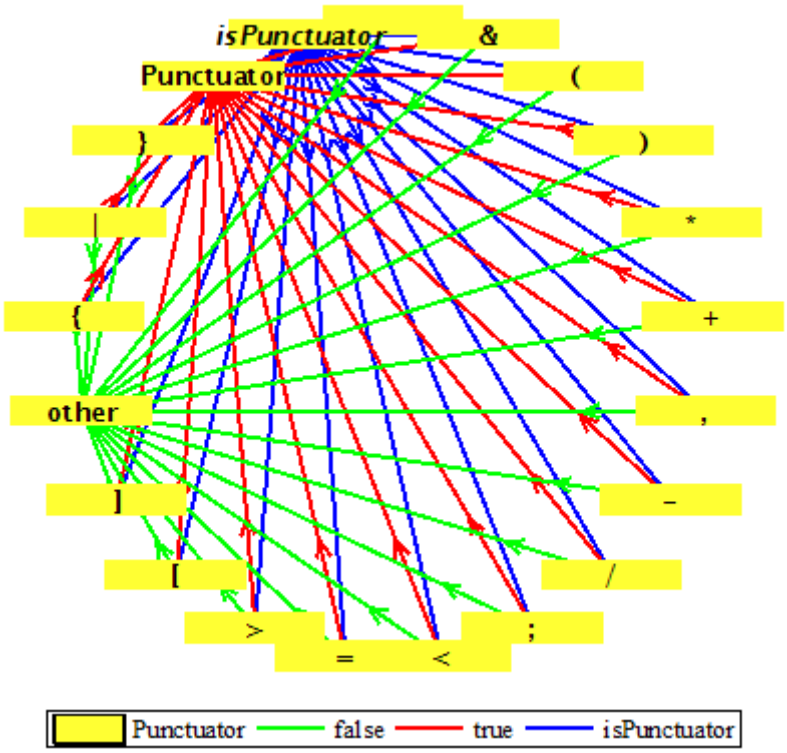
Метод:

безоткатное сканирование входного потока для распознавания и классификации токенов. Программный алгоритм лексического анализатора содержит булевы процедуры или предикаты, отвечающие на вопрос: является ли текущий токен идентификатором, числом, ключевым словом, знаком операции или символом пунктуации. Алгоритм сканирования представляется деревом, с программным корнем **main**, который передает управление диспетчеру программы **main**-> **scanner**.



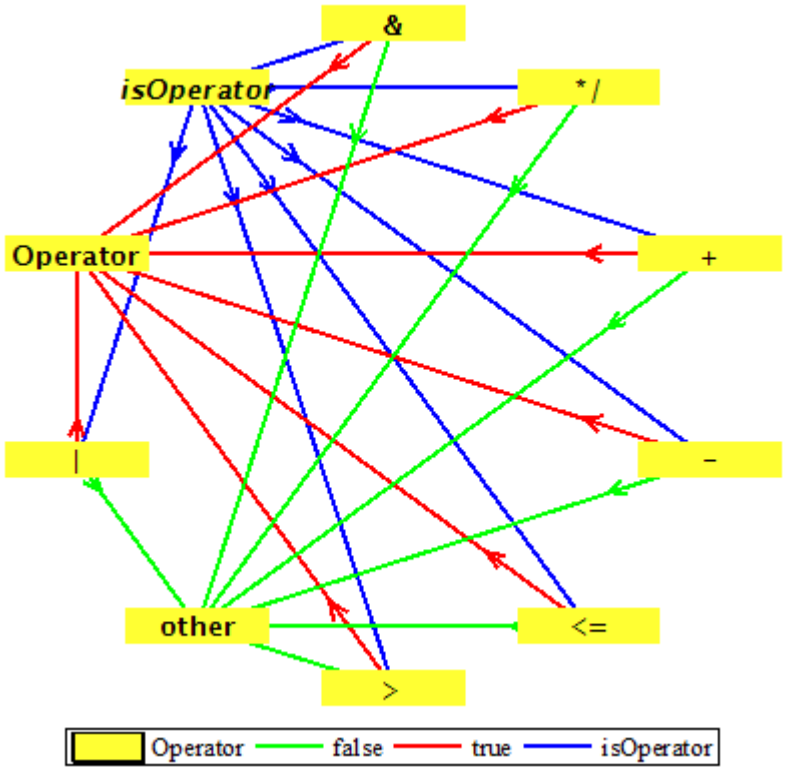
Диспетчер **scanner** организует последовательную проверку предикатов по пути ->**isPunctuator**->**isOperator**->**isKeyword**->**isNumber**->**validIdentifier**. Предикат **validIdentifier**,-в процессе разбора, вызывает предикат **isPunctuator**. Остальные предикаты **isPunctuator**, **isOperator**, **isKeyword**, **isNumber** работают автономно.

Предикат **isPunctuator** проверяет, является ли текущий символ пунктуацией: " ", "+", "-", "*", "/", ",", ";", ">", "<", "=", "(", ")", "[", "]", "{", "}", "&", "|" или нет.



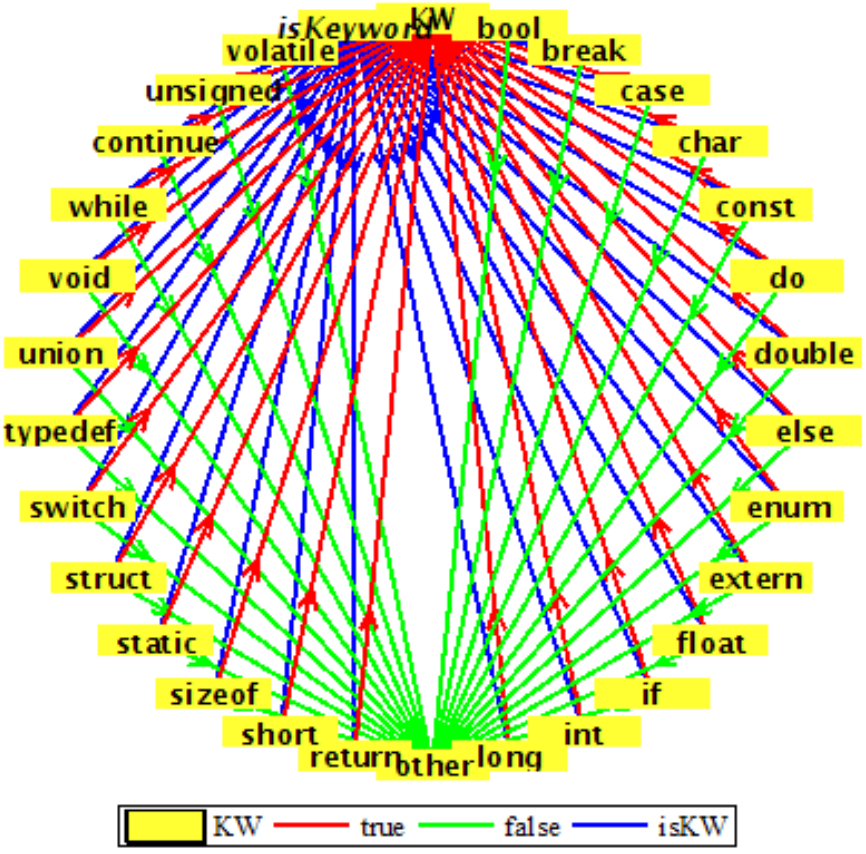
Предикат **validIdentifier** проверяет, действителен ли данный идентификатор – отвечает на поставленный вопрос ответом «нет», если первый символ токена является цифрой "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" или специальным символом пунктуации.

Предикат **isOperator** проверяет, является ли данный символ оператором: "+", "-", "*" "/", ">", "<" "=", "|", "&" или нет.



я

Предикат **isKeyword**: проверяет, является ли данная подстрока ключевым словом "if", "else", "while", "do", "break", "continue", "int", "double", "float", "return", "char", "case", "long", "short", "typedef", "switch", "unsigned", "void", "static", "struct", "sizeof", !strcmp(str, "long", "volatile", "typedef", "enum", "const", "union", "extern", "bool" или нет



Предикат **isNumber**: проверяет, является ли данная подстрока числом или нет.

Результат

Результатом отладки программы лексического анализатора является исполняемый файл и файл входных данных, содержащий тестируемую цепочку.

Выводы

Написание лексического анализатора *ab initio* – достаточно трудоемкая работа, и, более того, неэффективная в профессиональной деятельности. Использование регулярных выражений и регулярных определений существенно сокращает объем работы и автоматизирует ее с одновременным повышением качества распознавания. Этот вопрос изучается в следующей лабораторной работе.

Литература и ссылки на источник информации

Тема: Конечные автоматы и лексические анализаторы	Технологии создания и реализация лексических

		анализаторов
	https://favtutor.com/blogs/lexical-analyzer-cpp https://github.com/amirhakimnejad/Scanner-for-lexical-analyzer-in-cpp https://en.wikipedia.org/wiki/Lexical_analysis	Примеры программ