### 1. Описание темы (идея языка)

**Концепция**: Это простой язык для выполнения математических вычислений, работы с массивами и базовыми управляющими конструкциями. Он будет понятным, но достаточно мощным для демонстрации всех требуемых возможностей

**Особенности языка**:

1. **Типы данных**:
   * Целые числа (int) и вещественные числа (float) — для переменных и констант
   * Одномерные статические массивы (например, arr[10])
   * Дополнительно: поддержка стандартных математических функций, таких как sqrt, sin, cos
2. **Операторы**:
   * Присваивание: =
   * Арифметические операции: +, -, \*, / (с приоритетами: \*, / выше, чем +, -)
   * Скобки () для группировки выражений
   * Условные операторы: if (условие) { ... } else { ... }
   * Циклы: while (условие) { ... }
   * Операции сравнения: ==, !=, <, >, <=, >=
3. **Ввод/вывод**:
   * input(x) — для ввода значения переменной x
   * print(x) — для вывода значения переменной, массива или выражения
4. **Дополнительно (для бригады из 3 человек)**:
   * Поддержка функций sqrt, sin, cos в выражениях, чтобы показать работу с математическими операциями
   * Простая диагностика ошибок с указанием строки и позиции символа

**2. Лексемы для языка SimpleCalc**

| **№** | **Лексема** | **Описание** | **Пример** |
| --- | --- | --- | --- |
| 1 | ID | Идентификатор (имя переменной) | x, y, arr |
| 2 | NUMBER | Целое или вещественное число | 5, 3.14 |
| 3 | ASSIGN | Оператор присваивания | = |
| 4 | PLUS | Сложение | + |
| 5 | MINUS | Вычитание | - |
| 6 | MUL | Умножение | \* |
| 7 | DIV | Деление | / |
| 8 | LPAREN | Левая скобка | ( |
| 9 | RPAREN | Правая скобка | ) |
| 10 | SEMICOLON | Точка с запятой | ; |
| 11 | LBRACE | Левая фигурная скобка | { |
| 12 | RBRACE | Правая фигурная скобка | } |
| 13 | IF | Ключевое слово if | if |
| 14 | ELSE | Ключевое слово else | else |
| 15 | WHILE | Ключевое слово while | while |
| 16 | INPUT | Ключевое слово ввода | input |
| 17 | PRINT | Ключевое слово вывода | print |
| 18 | EQ | Равенство | == |
| 19 | NEQ | Неравенство | != |
| 20 | LT | Меньше | < |
| 21 | GT | Больше | > |
| 22 | LE | Меньше или равно | <= |
| 23 | GE | Больше или равно | >= |
| 24 | LBRACKET | Левая квадратная скобка | [ |
| 25 | RBRACKET | Правая квадратная скобка | ] |
| 26 | COMMA | Запятая | , |
| 27 | SQRT | Функция квадратного корня | sqrt |
| 28 | SIN | Функция синуса | sin |
| 29 | COS | Функция косинуса | cos |
| 30 | STRING | Строковый литерал (для print) | "z is small" |
| 31 | EOF | Конец файла | - |

### 3. Конечный автомат для лексического анализатора

**Основные состояния автомата:**

* **START**: Начальное состояние, где мы решаем, что за лексема начинается
* **ID**: Чтение идентификатора (буквы и цифры)
* **NUMBER**: Чтение числа (цифры, возможно точка для вещественных чисел)
* **OPERATOR**: Чтение операторов (=, +, -, \*, /, ==, != и т.д.)
* **STRING**: Чтение строкового литерала (например, "z is small")
* **ERROR**: Обнаружена ошибка (например, недопустимый символ)

Таблица переходов (упрощённая):

| **Текущее состояние** | **Входной символ** | **Новое состояние** | **Действие** |
| --- | --- | --- | --- |
| START | Буква (a-z, A-Z) | ID | Начать чтение идентификатора |
| START | Цифра (0-9) | NUMBER | Начать чтение числа |
| START | = | OPERATOR | Зафиксировать ASSIGN или проверить == |
| START | + | OPERATOR | Зафиксировать PLUS |
| START | - | OPERATOR | Зафиксировать MINUS |
| START | \* | OPERATOR | Зафиксировать MUL |
| START | / | OPERATOR | Зафиксировать DIV |
| START | ( | OPERATOR | Зафиксировать LPAREN |
| START | ) | OPERATOR | Зафиксировать RPAREN |
| START | ; | OPERATOR | Зафиксировать SEMICOLON |
| START | { | OPERATOR | Зафиксировать LBRACE |
| START | } | OPERATOR | Зафиксировать RBRACE |
| START | " | STRING | Начать чтение строки |
| START | Пробел | START | Пропустить пробел |
| START | Неверный символ | ERROR | Выдать ошибку |
| ID | Буква, цифра | ID | Добавить символ к идентификатору |
| ID | Другой символ | START | Завершить идентификатор, вернуть токен |
| NUMBER | Цифра | NUMBER | Добавить цифру к числу |
| NUMBER | . | NUMBER | Перейти к чтению дробной части |
| NUMBER | Другой символ | START | Завершить число, вернуть токен |
| STRING | Любой, кроме " | STRING | Добавить символ к строке |
| STRING | " | START | Завершить строку, вернуть токен |

### 4. Таблица переходов для строки x = 5 + 3.14;

| **Шаг** | **Текущее состояние** | **Входной символ** | **Новое состояние** | **Действие** | **Строка** | **Столбец** | **Созданный токен** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | START | x | ID | Начать чтение идентификатора | 1 | 1 | - |
| 2 | ID | (пробел) | START | Завершить идентификатор, создать токен | 1 | 2 | Token(TokenType.ID, 'x', 1, 1) |
| 3 | START | = | OPERATOR | Зафиксировать оператор | 1 | 3 | - |
| 4 | OPERATOR | (пробел) | START | Создать токен = | 1 | 4 | Token(TokenType.ASSIGN, '=', 1, 3) |
| 5 | START | 5 | NUMBER | Начать чтение числа | 1 | 5 | - |
| 6 | NUMBER | (пробел) | START | Завершить число, создать токен | 1 | 6 | Token(TokenType.NUMBER, '5', 1, 5) |
| 7 | START | + | OPERATOR | Зафиксировать оператор | 1 | 7 | - |
| 8 | OPERATOR | (пробел) | START | Создать токен + | 1 | 8 | Token(TokenType.PLUS, '+', 1, 7) |
| 9 | START | 3 | NUMBER | Начать чтение числа | 1 | 9 | - |
| 10 | NUMBER | . | NUMBER | Продолжить чтение (вещественное число) | 1 | 10 | - |
| 11 | NUMBER | 1 | NUMBER | Продолжить чтение числа | 1 | 11 | - |
| 12 | NUMBER | 4 | NUMBER | Продолжить чтение числа | 1 | 12 | - |
| 13 | NUMBER | ; | START | Завершить число, создать токен | 1 | 13 | Token(TokenType.NUMBER, '3.14', 1, 9) |
| 14 | START | ; | OPERATOR | Зафиксировать оператор | 1 | 13 | - |
| 15 | OPERATOR | (конец) | START | Создать токен ; | 1 | 14 | Token(TokenType.SEMICOLON, ';', 1, 13) |
| 16 | START | (конец) | DONE | Создать токен EOF | 1 | 14 | Token(TokenType.EOF, '', 1, 14) |