### Описание темы (идея языка)

**Концепция**: Это простой язык для выполнения математических вычислений, работы с массивами и базовыми управляющими конструкциями

**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

**1.1 Лексемы для языка SimpleCalc**

| **Номер** | **Лексема** | **Тип** |
| --- | --- | --- |
| 1 | if | Ключевое слово |
| 2 | else | Ключевое слово |
| 3 | while | Ключевое слово |
| 4 | input | Ключевое слово |
| 5 | print | Ключевое слово |
| 6 | sqrt | Ключевое слово |
| 7 | sin | Ключевое слово |
| 8 | cos | Ключевое слово |
| 9 | + | Оператор |
| 10 | - | Оператор |
| 11 | \* | Оператор |
| 12 | / | Оператор |
| 13 | = | Оператор |
| 14 | == | Оператор сравнения |
| 15 | != | Оператор сравнения |
| 16 | < | Оператор сравнения |
| 17 | > | Оператор сравнения |
| 18 | <= | Оператор сравнения |
| 19 | >= | Оператор сравнения |
| 20 | ( | Скобка |
| 21 | ) | Скобка |
| 22 | [ | Скобка (массив) |
| 23 | ] | Скобка (массив) |
| 24 | { | Скобка (тело) |
| 25 | } | Скобка (тело) |
| 26 | ; | Разделитель |
| 27 | , | Разделитель |
| 28 | ID | Идентификатор |
| 29 | INT | Целая константа |
| 30 | FLOAT | Вещественная константа |
| 31 | STRING | Строковая константа |

**1.2 Таблица переходов автомата**

Конечный автомат лексического анализатора имеет следующие состояния:

* S0 — начальное состояние.
* S1 — обработка целых чисел.
* S2 — обработка идентификаторов.
* S3 — обработка операторов и операторов сравнения.
* S4 — обработка вещественных чисел (после точки).
* S5 — обработка строковых литералов.

Таблица переходов:

| **Текущее состояние** | **Входной символ** | **Следующее состояние** | **Действие** |
| --- | --- | --- | --- |
| S0 | Буква | S2 | Начать идентификатор |
| S0 | Цифра | S1 | Начать число |
| S0 | +, -, \*, / | S3 | Начать оператор |
| S0 | = | S3 | Проверить на == |
| S0 | <, >, ! | S3 | Проверить на <=, >=, != |
| S0 | (, ), [, ], {, }, ;, , | S0 | Обработать скобку или разделитель |
| S0 | " | S5 | Начать строку |
| S1 | Цифра | S1 | Продолжить число |
| S1 | . | S4 | Перейти к вещественному числу |
| S1 | Не цифра | S0 | Завершить число |
| S2 | Буква или цифра | S2 | Продолжить идентификатор |
| S2 | Не буква/цифра | S0 | Завершить идентификатор |
| S3 | = (если предыдущий <, >, !, =) | S0 | Обработать оператор сравнения |
| S3 | Другой символ | S0 | Обработать одиночный оператор |
| S4 | Цифра | S4 | Продолжить вещественное число |
| S4 | Не цифра | S0 | Завершить вещественное число |
| S5 | Любой, кроме " | S5 | Продолжить строку |
| S5 | " | S0 | Завершить строку |

### 2 КС-грамматика языка

A → Q ; A | λ

Q → E | C | L | I | O

E → ID = S | ID [ S ] = S

S → T S'

S' → + T S' | - T S' | λ

T → F T'

T' → \* F T' | / F T' | λ

F → ( S ) | ID | ID [ S ] | INT | FLOAT | - F | sqrt ( S ) | sin ( S ) | cos ( S )

C → if ( V ) { A } C'

C' → else { A } | λ

L → while ( V ) { A }

V → S W S

W → == | != | < | > | <= | >=

I → input ID

O → print ( S ) | print ( STRING )

### 3. КС-грамматика в нестрогой форме Грейбах

Преобразованная грамматика, где правила начинаются с терминала:

S → T S'

S' → + T S' | - T S' | λ

T → F T'

T' → \* F T' | / F T' | λ

F → ( S ) | ID | ID [ S ] | INT | FLOAT | - F | sqrt ( S ) | sin ( S ) | cos ( S )

C → if ( V ) { A } C'

C' → else { A } | λ

L → while ( V ) { A }

V → S W S

### 4. Семантические действия для генерации ОПС

Используются следующие соглашения:

* gen(op) — добавить операцию op в ОПС.
* push(val) — загрузить значение/переменную в стек.
* new\_label() — создать уникальную метку для переходов.

Таблица семантических действий:

| **Правило** | **Действие** |
| --- | --- |
| A → Q ; A | Сгенерировать код для Q, затем для A. |
| A → λ | Ничего не генерировать. |
| Q → E | Сгенерировать код для E. |
| E → ID = S | Вычислить S, затем gen("STORE ID"). |
| E → ID [ S ] = S | Вычислить первый S (индекс), затем второй S, затем gen("STORE\_ARR ID"). |
| S → T S' | Сгенерировать код для T, затем для S'. |
| S' → + T S' | Сгенерировать код для T, gen("ADD"), затем для S'. |
| S' → - T S' | Сгенерировать код для T, gen("SUB"), затем для S'. |
| S' → λ | Ничего не генерировать. |
| T → F T' | Сгенерировать код для F, затем для T'. |
| T' → \* F T' | Сгенерировать код для F, gen("MUL"), затем для T'. |
| T' → / F T' | Сгенерировать код для F, gen("DIV"), затем для T'. |
| T' → λ | Ничего не генерировать. |
| F → ( S ) | Сгенерировать код для S. |
| F → ID | gen("LOAD ID"). |
| F → ID [ S ] | Вычислить S (индекс), gen("LOAD\_ARR ID"). |
| F → INT | gen("PUSH INT"). |
| F → FLOAT | gen("PUSH FLOAT"). |
| F → - F | Сгенерировать код для F, gen("NEG"). |
| F → sqrt ( S ) | Сгенерировать код для S, gen("SQRT"). |
| C → if ( V ) { A } C' | Создать метки L1, L2. Вычислить V, gen("JMP\_IF\_FALSE L1"), сгенерировать код для A, gen("JMP L2"), gen("LABEL L1"), сгенерировать код для C', gen("LABEL L2"). |
| C' → else { A } | Сгенерировать код для A. |
| C' → λ | Ничего не генерировать. |
| L → while ( V ) { A } | Создать метки L1, L2, gen("LABEL L1"), вычислить V, gen("JMP\_IF\_FALSE L2"), сгенерировать код для A, gen("JMP L1"), gen("LABEL L2"). |
| V → S W S | Вычислить оба S, gen("CMP W"). |
| I → input ID | gen("IN ID"). |
| O → print ( S ) | Вычислить S, gen("OUT"). |
| O → print ( STRING ) | gen("PUSH STRING"), gen("OUT"). |

### 5. Список операций ОПС

Обратная польская запись (ОПС) для языка SimpleCalc включает следующие операции:

| **Код операции** | **Описание** |
| --- | --- |
| LOAD a | Загрузить переменную a в стек. |
| STORE a | Сохранить значение из стека в a. |
| LOAD\_ARR a | Загрузить элемент массива a[индекс]. |
| STORE\_ARR a | Сохранить значение в a[индекс]. |
| PUSH k | Положить константу k в стек. |
| ADD | Сложение: a + b. |
| SUB | Вычитание: a - b. |
| MUL | Умножение: a \* b. |
| DIV | Деление: a / b. |
| NEG | Унарный минус: -a. |
| CMP op | Сравнить a и b (op: ==, !=, <, >, <=, >=). |
| JMP\_IF\_FALSE L | Перейти к метке L, если условие ложно. |
| JMP L | Безусловный переход к метке L. |
| LABEL L | Объявление метки L. |
| IN a | Ввод значения в переменную a. |
| OUT | Вывод значения из стека. |
| SQRT | Вычислить квадратный корень. |
| SIN | Вычислить синус. |
| COS | Вычислить косинус. |

### 6. Формат ОПС

**Каждая команда записывается в виде:**

<КОД\_ОПЕРАЦИИ> [ОПЕРАНД]

**Пример для выражения x = 5 + 3.14;**

PUSH 5

PUSH 3.14

ADD

STORE x