Введение

Books | Coffee | Two Rooks SL — интерпретируемый язык программирования с поддержкой базовых конструкций, ООП, работы с файлами и сетевыми запросами.
Основные цели:

- Простота изучения.
- Минималистичный синтаксис.
- Расширяемость.

Установка и запуск

1. Требования: .NET Framework 4.8 или выше.
2. Скомпилируйте код интерпретатора в Visual Studio.
3. Запустите AdvancedInterpreter.exe.
4. Вводите команды в интерактивном режиме.

Синтаксис

- Команды пишутся в одну строку.
- Регистр команд не учитывается (например, PRINT и print эквивалентны).
- Многострочные блоки завершаются ключевым словом END.

Типы данных

- **Числа**: 10, 3.14.
- **Строки**: "Hello, World!".
- **Массивы**: ARRAY numbers 1 2 3.
- **Объекты**: Экземпляры классов.
- **Логические значения**: 1 (истина), 0 (ложь).

Переменные

- **Объявление:** LET <имя> = <значение>.
- **Присваивание:** LET x = 10.
- **Доступ к элементам массива:** numbers[0].
  Пример:

```
LET name = "Alice"
ARRAY values 10 20 30
 PRINT values[1]  # Выведет 20
```

Операторы

- **Арифметические**: +, -, *, /.
- **Сравнения**: =, <, >.
- **Логические**: AND, OR, NOT.

Управляющие конструкции

Условные операторы

Синтаксис:

```
IF <условие> THEN
 <блок_кода>
[ELSE <блок_кода>]
END
```

Пример:

```
IF x > 10 THEN
 PRINT "x > 10"
ELSE
PRINT "x <= 10"
 END
```

Циклы

Цикл FOR:

```
FOR <переменная> = <начало> TO <конец> [STEP <шаг>] <блок_кода> END
```

Пример:

```
FOR i = 1 TO 5 PRINT i END
```

Цикл WHILE:

```
WHILE <условие> DO <блок_кода> END
```

Пример:

```
LET x = 0 WHILE x < 3 DO PRINT x LET x = x + 1 END
```

Функции

- **Объявление**: DEFINE FUNCTION <имя> AS <выражение>.
- **Вызов**: <имя>(аргументы).
  Пример:

```
DEFINE FUNCTION SQUARE AS x * x PRINT SQUARE(5)  # 25
```

Встроенные функции

- LEN(s) — длина строки.
- SUBSTR(s, start, length) — подстрока.
- TO_UPPER(s) — строка в верхнем регистре.
  Классы и объекты

- **Объявление класса**: CLASS <имя>.
- **Создание объекта**: LET obj = NEW <имя_класса>.
  Пример:

```
CLASS Person LET p = NEW Person
```

Работа с файлами

- **Чтение**: FILE READ "file.txt".
- **Запись**: FILE WRITE "file.txt" "Текст".
  Пример:

```
FILE WRITE "test.txt" "Hello!" FILE READ "test.txt" PRINT FILE_CONTENT  # Hello!
```

Сетевые операции

HTTP-запросы:

```
HTTP GET "https://api.example.com/data"
```

Пример:

```
HTTP GET "https://jsonplaceholder.typicode.com/todos/1" # Выведет JSON-ответ
```

Многопоточность

Запуск кода в потоке:

```
THREAD <блок_кода>
```

Пример:

```
THREAD PRINT "Это выполняется в фоне"
```

Обработка ошибок

Блок TRY/CATCH:

```
TRY <блок_кода> CATCH <обработка_ошибки> END
```

Пример:

```
TRY LET x = 10 / 0 CATCH PRINT "Ошибка: деление на ноль" END
```

Примеры программ

Калькулятор:

```
DEFINE FUNCTION ADD AS a + b PRINT ADD(5, 3)  # 8
```

Чтение файла:

```
FILE READ "data.txt" PRINT FILE_CONTENT
```

Известные ограничения

5. Нет строгой проверки типов.
6. Ограниченная поддержка ООП (нет наследования).
7. Многострочные блоки требуют END.
8. Низкая производительность для сложных задач.

Заключение

Язык подходит для обучения основам программирования и автоматизации простых задач. Для проектов уровня производства рекомендуется использовать более мощные языки (Python, C#).