НИЯУ МИФИ
Кафедра 22 «Кибернетики»

Лабораторная работа

на тему: «Разработка компактного языка для вероятностного моделирования»
(язык сценариев ProbabilityScript)

Выполнил: _____ (ФИО студента)

Руководитель: _____ (ФИО преподавателя)

Москва – 2025

Введение

Цель проекта – разработка компактного предметно-ориентированного языка **ProbabilityScript** для вероятностного моделирования. Язык сценариев предназначен для упрощения моделирования случайных процессов и статистических вычислений. В рамках проекта реализован интерпретатор, позволяющий задавать последовательность случайных экспериментов, собирать результаты и рассчитывать основные статистические показатели.

Представленная документация описывает синтаксис и возможности языка ProbabilityScript, архитектуру реализации интерпретатора, пример использования языка, а также содержит заключение с выводами и возможными направлениями развития. В приложении приведён полный листинг демонстрационного сценария `demo_basic.psc`, иллюстрирующего основные возможности языка.

1 Синтаксис языка ProbabilityScript

Данный раздел описывает синтаксис и ключевые возможности языка сценариев ProbabilityScript. Язык имеет минималистичный синтаксис и предназначен для генерации случайных чисел, повторения серии экспериментов, условий, а также вычислений и сбора статистики. Основной (и единственный) тип данных в языке – вещественное число с плавающей запятой (floating point).

1.1 Типы данных

В языке ProbabilityScript используется один базовый тип данных:

- **Вещественные числа:** все переменные и выражения являются числовыми (тип с плавающей запятой, обычно двойной точности). Целочисленные значения не выделяются отдельно, а представляют собой частный случай вещественного.

Переменные в сценарии не объявляются заранее – они неявно создаются при первом присваивании. Имена переменных могут состоять из букв латинского алфавита, цифр и символа подчёркивания, начинаются с буквы или подчёркивания. Примеры допустимых имён: `x`, `value1`, `sum_2`.

1.2 Выражения и операторы

Выражения используются в присваиваниях, `print`, `collect`, условиях `if` и аргументах функций. Поддерживаются:

- **Числа:** `1`, `3.14`.
- **Переменные:** `x`, `sum`.
- **Вызовы функций:** `uniform(0, 1)`, `mean()`.
- **Унарный минус:** `-x`, `-(1 + 2)`.
- **Скобки:** `( ... )` для явного задания порядка вычислений.

Арифметические операторы:

$$+,\ -,\ *,\ /$$

Они вычисляют результат как обычные арифметические операции над вещественными числами.

Операторы сравнения:

$$>,\ <,\ ==,\ !=$$

Результат сравнения является числом: `1.0`, если условие истинно, и `0.0`, если ложно. Это важно для конструкции `if`, которая проверяет «ненулевость» результата.

Приоритеты операций (от более высокого к более низкому):

1. унарный минус `-`
2. умножение/деление `*`, `/`
3. сложение/вычитание `+`, `-`
4. операции сравнения `>`, `<`, `==`, `!=`

1.3 Конструкции языка

Язык сценариев предоставляет небольшой набор управляющих конструкций и инструкций:

- **Присваивание.** Оператор `=` используется для присваивания значения переменной: `x = 5`, `y = x + 1.5`.
- **Цикл `repeat`.** Синтаксис: `repeat` E `{ ... }`, где E – числовое выражение (ожидается неотрицательное значение; фактически используется целая часть). Блок в фигурных скобках выполняется указанное число раз. Вложенные блоки и вложенные `repeat` допускаются.
- **Условие `if`.** Синтаксис: `if` E `{ ... }`, где E – числовое выражение. Если $E \neq 0$, блок выполняется, иначе пропускается. Обычно в условии используются сравнения, например: `if x > 0 { ... }`.
- **Инструкция `collect`.** `collect` E вычисляет выражение E и добавляет значение в выборку (по умолчанию в единственную текущую выборку).
- **Вывод `print`.** `print` E выводит значение выражения E в стандартный вывод. Команда печатает число и перевод строки.
- **Вывод статистики `print_stat`.** `print_stat("name")` печатает **одну** статистику по текущей выборке. Поддерживаются: `"mean"`, `"variance"` (или `"var"`), `"stddev"` (или `"std"`), `"median"`, `"count"`.

Блоки кода ограничиваются фигурными скобками `{}`. Каждая инструкция записывается с новой строки (точка с запятой не требуется). Поддерживаются однострочные комментарии `// ...`.

1.4 Встроенные функции

Для организации вероятностных экспериментов и статистической обработки в языке ProbabilityScript доступны следующие встроенные функции:

- `uniform()`: равномерное распределение по умолчанию $U(0, 1)$.

- `uniform(`b`)`: равномерное распределение $U(0, b)$.
- `uniform(`a`, `b`)`: равномерное распределение $U(a, b)$, где должно выполняться $a < b$.
- `normal()`: нормальное распределение по умолчанию $N(0, 1)$.
- `normal(`μ`)`: нормальное распределение $N(\mu, 1)$.
- `normal(`μ`, `σ`)`: нормальное распределение $N(\mu, \sigma)$, где $\sigma > 0$.
- `mean()` – среднее по текущей выборке.
- `variance()` (или `var()`) – дисперсия по текущей выборке.
- `stddev()` (или `std()`) – стандартное отклонение по текущей выборке.
- `median()` – медиана по текущей выборке.
- `count()` – размер текущей выборки.

Функции `mean`, `variance`, `stddev`, `median`, `count` не принимают аргументов. Если выборка пуста, вызов статистики считается ошибкой исполнения (в текущей реализации).

1.5 Примеры сценариев

Ниже приведены примеры использования ProbabilityScript.

Пример 1. Оценка характеристик равномерного распределения.

```
N = 100
repeat N {
    x = uniform(0, 1)
    collect x
}
print_stat("count")
print_stat("mean")
print_stat("variance")
print_stat("stddev")
print_stat("median")
```

Листинг 1. Оценка характеристик равномерного распределения

Пример 2. Условие `if` и операторы сравнения.

```
x = 1 + 2
if x > 2 {
    print 42
}
if x == 0 {
    print 99
}
```

Листинг 2. Условие if и операторы сравнения

В этом примере первое условие истинно (печатается `42`), второе – ложно (блок не выполняется).

Пример 3. Оценка характеристик нормального распределения.

```
N = 50
repeat N {
    y = normal(0, 1)
    collect y
}
print mean()
print stddev()
```

Листинг 3. Оценка характеристик нормального распределения

2 Архитектура реализации

Реализация интерпретатора ProbabilityScript основана на классической структуре интерпретируемого языка: входной сценарий последовательно проходит этапы лексического анализа, синтаксического анализа (разбора), построения внутреннего представления и непосредственного исполнения.

Основные компоненты системы: лексер, парсер, дерево разбора (AST), интерпретатор, среда выполнения и модуль статистики.

2.1 Модули интерпретатора

- **Лексер (Lexer).** Преобразует исходный текст в поток токенов: ключевые слова (`repeat`, `collect`, `print`, `if`), идентификаторы,

числа, строковые литералы (для `print_stat("...")`), знаки операций и скобки. Поддерживаются однострочные комментарии `// ...`.

- **Парсер (Parser).** Строит AST по потоку токенов. Реализованы правила для операторов (`repeat`, `if`, `collect`, `print`, `print_stat`, присваивание), а также выражений с поддержкой арифметики и сравнений.
- **AST.** Представляет программу как дерево узлов выражений и операторов (в т.ч. `IfStmt` и `BinaryExpr` для сравнений).
- **Интерпретатор.** Выполняет AST. Сравнения возвращают `1.0/0.0`, а `if` проверяет условие как «значение не равно нулю».
- **Среда выполнения (Environment).** Хранит переменные и собранную выборку значений (для `collect`).
- **Модуль статистики.** Содержит расчёт статистик по выборке: `mean`, `variance`, `stddev`, `median`, `count`.

2.2 Путь от сценария до исполнения

После запуска интерпретатора с указанием сценария ProbabilityScript выполнение происходит по следующим этапам:

1. Чтение файла сценария (`.psc`) в строку.
2. Лексический анализ: текст преобразуется в токены.
3. Синтаксический анализ и построение AST.
4. Исполнение AST интерпретатором: вычисление выражений, выполнение `repeat` и `if`, сбор данных через `collect`, вывод через `print` и `print_stat`.
5. Завершение работы (или сообщение об ошибке при некорректном сценарии/ошибке исполнения).

3 Пример использования

1. **Подготовка сценария.** Сценарий хранится в файле с расширением `.psc`.
2. **Запуск интерпретатора.** Пример:

   ```
   $ psc demo_basic.psc
   ```

3. **Фиксация seed (воспроизводимость).** Пример:

   ```
   $ psc --seed=42 demo_basic.psc
   ```

Заключение

В рамках проекта разработан и реализован простой интерпретируемый язык сценариев **ProbabilityScript** для вероятностного моделирования. Язык поддерживает генерацию случайных чисел из базовых распределений (равномерного и нормального), повторение серий экспериментов с накоплением результатов, вычисление основных статистических характеристик выборки, условный оператор `if` на базе числовой семантики истинности, а также выражения с арифметическими операциями и операторами сравнения.

Дальнейшее развитие может включать добавление `else`, логических операций (`&&`, `||`), новых распределений, поддержку нескольких именованных выборок, расширение типов данных и более удобные средства вывода/экспорта результатов.

Приложение: Полный листинг сценария `demo_basic.psc`

```
// demo_basic.psc

N = 100
repeat N {
    x = uniform(0, 1)
    collect x
}
```

```
print_stat("count")
print_stat("mean")
print_stat("variance")
print_stat("stddev")
print_stat("median")
```

Листинг 4. Сценарий demo_basic.psc