MoeScript

**Introducción al Lenguaje**

**MoeScript** es un lenguaje de programación imperativo con un enfoque lúdico, fuertemente inspirado en la cultura anime. Su sintaxis mezcla conceptos tradicionales de lenguajes como C con palabras reservadas y símbolos que evocan humor y originalidad.

Ideal para proyectos educativos, compiladores experimentales o simplemente como una declaración de amor a la creatividad en la programación.

**Herramientas Utilizadas: FLEX y BISON**

**FLEX (Generador de Analizadores Léxicos)**

* Utiliza autómatas finitos deterministas (AFD) para reconocer patrones
* Opera en dos fases:
  + Escaneo: Divide el input en tokens usando reglas regex.
  + Acciones semánticas: Ejecuta código C asociado a cada Token (ej: almacenar valores literales).

**Características avanzadas:**

* + Modos de inicio (%x): Permite manejar contextos complejos (ej: dentro de comentarios multilínea).
  + Buffer reentrante: Útil para analizar múltiples flujos de entrada simultáneos.

**BISON (Generador de Analizadores Sintácticos)**

* Implementa parsers LALR(1) con posibilidad de manejo de conflictos (shift/reduce, reduce/reduce).
* Soporta atributos sintetizados y heredados para análisis semántico.

**Características avanzadas:**

* + Manejo de precedencia: %left, %right, %nonasoc para operadores.
  + Valores semánticos (%union): Permite múltiplos tipos de datos en el AST.
  + Traza de parsing (YYDEBUG): Depuración detallada del proceso de análisis.

**Integración FLEX-BISON**

Flujo de trabajo:

FLEX envía tokens a BISON mediante yylex().

BISON:

* + Solicita tokens llamando a yylex().
  + Construye el AST usando acciones semánticas.
  + Maneja la tabla de símbolos (implementada externamente en C).

**Aplicación en Creación de Lenguajes**

**Proceso típico:**

* + Diseño gramatical:
    - Definir Token (FLEX).
    - Especificar gramática libre de contexto (BISON).
  + Análisis semántico:
    - Atributos en reglas de BISON para verificación de tipos.
    - Tabla de símbolos para manejo de ámbitos.
  + Generación de código:
    - Recorrido del AST para emitir código intermedio o final.

**Gramática**

**Características principales:**

* Tipado estático con declaraciones con términos de anime como:
  + Senpai (entero).
  + Kohai (flotante).
  + Tsundere/Yandere (booleano, True/False respectivamente)
* Sintaxis única, donde las sentencias terminan con **-chan** en lugar del tradicional ;.
* Palabras clave temáticas como:
  + Moshi (if).
  + Kawaii (while).
  + Baka (print).

**Convenciones y Estilo:**

* Las palabras reservadas pueden escribirse en minúsculas o mayúsculas (moshi o MOSHI)
* Los identificadores pueden contener letras, números y guiones bajos
* Los literales numéricos tienen límites de tamaño (enteros entre -99999 y 99999)

**Manual de Usuario de MoeScript:**

**instalación y ejecución.**

**Sintaxis Básica**

Comentarios.

|  |
| --- |
| /\* Esto es un comentario en MoeScript \*/ |

Declaración de Variables

|  |  |  |
| --- | --- | --- |
| Tipo | Palabra Clave | Ejemplo |
| Entero | Senpai | senpai edad = 17-chan |
| Flotante | Kohai | kohai precio = 99.99-chan |
| Booleano | Tsundere | tsundere esMayor = true-chan |
| Cadena | String | string saludo = “Konichiwa”-chan |

Operadores

* **Aritméticos:** +, -, \*, /
* **Comparación:** ==, !=, >, <, >=, <=
* **Lógicos:** && (and), || (or), ! (not)

Estructuras de Control

Condicional if / else (moshi / nani respectivamente)

Bucle while (kawaii):

Bucle for (douka):

**Entrada y Salida**

Imprimir por pantalla: baka “Hola mundo”-chan

Leer entrada: yomu variable-chan

**Crear Objeto/Función, return**

Crear objetos(doki): doki Persona p -chan

Crear Funcion(sugoi): sugoi senpai crearpersona() {}

Return (uwu): uwu resultado

**Errores Comunes y Soluciones**

|  |  |  |
| --- | --- | --- |
| Error | Causa Probable | solución |
| Error: Falta ‘-chan’ | Olvidar terminar una sentencia | Anadir -chan al final. |
| Error: Variable no declarada | Usar una variable sin definirla | Declarar con senpai, kohai, etc. |
| Error: Tipo incorrecto | Asignar un valor de tipo distinto | Verificar tipos de datos. |

**Ejemplo de Ejecución:**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Descripción del proceso de generación de código**

Se transforma del código de MoeScript a C

Declaracion de variable:  
 int num = 0;

Impresión de mensaje inicial:

printf(“ingresa un numero: “)

Lectura desde consola:

scanf(“%d”, &num);

Condicional anidado:

if (num != 0) {

if (num > 0) {

printf (“El número es positivo”)

}

if (num < 0) {

printf (“El número es negativo”)

}

} else {

printf(“El número es 0”)

}

**AST del ejemplo de ejecución**

**Diagrama

El contenido generado por IA puede ser incorrecto.**

**Imágenes de otro ejemplo de ejecución**

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Conclusión**

MoeScript es un lenguaje ideal para quienes buscan aprender programación de una manera entretenida. Con su sintaxis temática y reglas simples, permite escribir código claro y expresivo.

Si quieres descargar y probar MoeScript, puedes hacerlo desde el siguiente enlace.

<https://github.com/JOT4JOT4/MoeScript>

Ahora estás listo para programar como un verdadero otaku.