Trabajo Práctico Especial:

Autómatas, Teoría de lenguajes y Compiladores.

Lenguaje Mana.

Integrantes:

- Ana Tarantino Pedrosa - 59147
- Martina Rodriguez - 59036

Índice

Objetivo del lenguaje

Mana fue creado con el objetivo de enseñar de una forma muy básica a programar. Está dirigido principalmente a aquellas personas que tienen muy poco conocimiento en el tema. Intenta evitar formalismos propios que los lenguajes tienen para facilitar su uso.

Consideraciones realizadas

Para contemplar que no se redefinan variables, se utilizó una lista con la información necesaria de cada variable definida hasta el momento.
A su vez, se decidió que el compilador genere un archivo de C para luego compilarlo con gcc y contemplar cualquier error que este genere.

Desarrollo del lenguaje

Se utilizó LEX y YACC. El primero fue útil para generar el análisis léxico del lenguaje mientras que con YACC se generaron las producciones de la gramática.
Como el lenguaje apunta a usuarios que recién comienzan a programar, se intentó que el lenguaje sea lo más parecido a la escritura corriente para que se convierta intuitivo. Es por esto que se evitó que el usuario tenga que lidiar con tipos de datos técnicos como ocurre en C. Por ejemplo, la manera de imprimir un texto es la siguiente:

print text: “texto a imprimir”.

O bien,

print text: variable_de_texto.

Mientras que en C se utiliza:

printf(“texto a imprimir”);
printf(“%s”,variable_de_texto);

Para quien recién comienza a programar le puede resultar confuso que función utilizar, de qué manera y el uso de los paréntesis. A su vez, el ‘%s’ y los diferentes tipos de datos que utiliza C para la impresión suelen ser tediosos para quien los desconoce.
Otra consideración fue que se decidió terminar las instrucciones con un punto ya que es lo más similar a la escritura cotidiana.

Como se mencionó anteriormente, se creó una lista de variables del programa con la cual se buscan las variables existentes hasta el momento y se evitan colisiones. La misma está construida a partir de nodos que contienen el nombre de la variable, su tipo de dato y un puntero al nodo siguiente. Con esto se intentó conseguir un mayor control de la compilación ya que de esta manera es posible abortar la misma cuando se está utilizando una variable que nunca fue definida, se está re definiendo una variable o se está utilizando de manera equívoca. Por ejemplo, no se puede tratar a un número como una lista de texto.
Se incorporó al lenguaje la posibilidad de crear dos tipos de listas: de números y de texto. De esta manera, el usuario puede hacer uso de ellas de manera muy simple. Su implementación permite agregar y borrar elementos de la misma e imprimirla. Esta característica permite a los nuevos programadores tener un acercamiento a los arreglos, los cuales son un pilar de la programación.
Se crearon cinco programas de ejemplo para introducir al lenguaje. Los mismos son:

- **hello:** un programa que intenta ser una introducción al lenguaje. El mismo le pide al usuario que ingrese su nombre y luego pide un número para generar la serie fibonacci hasta ese término. Con este programa se intenta mostrar al usuario la potencia matemática que tiene la programación.
- **factorial:** pide un número al usuario y calcula su factorial.
- **greaterNum:** pide dos números al usuario y devuelve cuál es el mayor de ellos o si son iguales.
- **textList:** crea una lista de textos inicializada y le pide al usuario ingresar datos para completarla.
- **numList:** crea una lista de números y hace un ciclo para rellenarla. Luego le pide al usuario un número para removerlo.

Gramática

La gramática empleada para este lenguaje sigue con una lógica parecida a la de C. Sus características son las siguientes:

- ➔ Para iniciar un programa se necesita necesita la palabra *start* y para finalizarlo *end*.
- ➔ Dentro del programa se pueden combinar instrucciones, permitiendo la anidación.
- ➔ Se puede declarar, asignar, imprimir, leer desde línea de comandos y operar sobre listas, todas las instrucciones deben terminar con '.'. En un principio se había elegido ';' como delimitador final por el lenguaje C pero se creyó que no era tan intuitivo para alguien que recién empieza a programar. La gramática para estos casos es:

Declaracion:

Variables:

new [TYPE] [VAR_NAME]. (TYPE puede ser de tipo TEXT o NUMBER).

Listas:

new [LIST_TYPE] [LIST_NAME] *starts with* [INIT_VALUE].

Asignación:

[VAR_NAME] = [EXP | CTE | OTHER_VAR].

Impresión:

print [TYPE]: [CTE | VAR_NAME].

Leer información de teclado:

read [TYPE] *and save inside* [VAR_NAME].

Operaciones de listas:

add [CTE | VAR_NAME] *to* [LIST_TYPE] *list* [LIST_NAME].

delete [CTE | VAR_NAME] *from* [LIST_TYPE] *list* [LIST_NAME].

- ➔ Sentencias de control permitidas:

if [EXP] *then* [INSTRUCTIONS] *else* [INSTRUCTIONS] *end if* (else opcional).

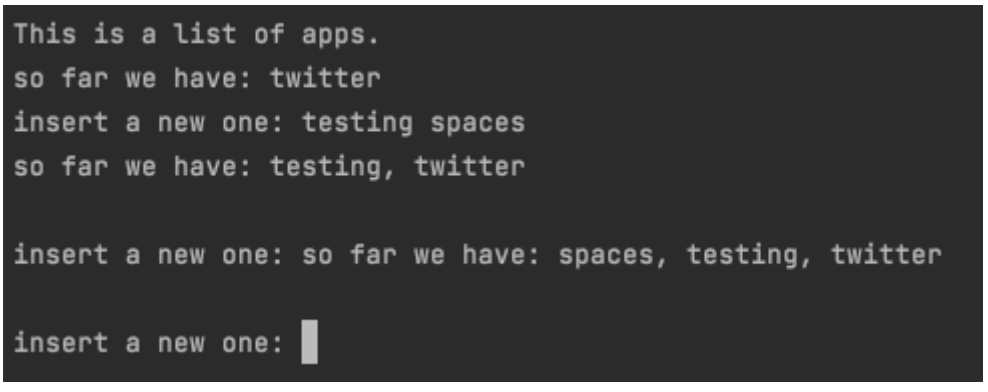
repeat: [INSTRUCTIONS] *while* [EXP].

➔ Las expresiones pueden contener operadores aritméticos, relacionales o lógicos. Se utilizaron símbolos para empezar a acostumbrar al usuario, y aproximarlos más a lo que es el lenguaje de programación C.
- ➢ [EXP] > [EXP]
- ➢ [EXP] < [EXP]
- ➢ [EXP] + [EXP]
- ➢ [EXP] - [EXP]
- ➢ [EXP] * [EXP]
- ➢ [EXP] / [EXP]
- ➢ [EXP] || [EXP]
- ➢ [EXP] && [EXP]
- ➢ [EXP] *is equal to* [EXP]
- ➢ [EXP] *is not equal to* [EXP]

En los únicos que se utilizaron palabras para facilitar la comprensión fueron los de comparación porque eran un poco menos intuitivos.

Dificultades encontradas

La mayor dificultad se encontró al momento de manejar listas. Había problemas para inicializarlas correctamente. Por eso se decidió poder crear listas pero que nunca estén vacías, es decir al crear una esta debe estar inicializada. Otro inconveniente encontrado fue que cuando se quiere agregar un elemento con un espacio en el medio, en vez de agregarse una vez en la lista como era lo esperado, se agrega como dos elementos distintos.
Por ejemplo:



Futuras extensiones

Mana si bien es un buen arranque para empezar a familiarizarse con la programación, podría en un futuro ofrecer más cosas para poder hacer que la experiencia de usuario sea aún mejor.
Principalmente el objetivo es terminar de pulir todas las dificultades encontradas.
Se podrían ofrecer distintos niveles de dificultad, para aquellos que ya saben un poco sobre programación, agregar nuevas funcionalidades como manejo de vectores, uso de for, funciones, nuevos tipos de datos (ejemplo: números con coma), entre otras. El objetivo de este lenguaje es que todos los que utilicen Mana puedan desarrollar un programa en C básico sin problemas.

Referencias

Para la correcta comprensión de Lex y Yacc se utilizó todo lo provisto por la cátedra, tanto clases como apuntes.