TALLER 01

Nombre: Doménica Camila Sánchez **Curso:** GR1SW

Fecha: 28-01-2024

CURSO STANFORD PROGRAMA 03

Objetivos

- Entender la estructura del lenguaje Cool mediante el desarrollo de un programa
- Realizar un programa list.cl para manipular la estructura de datos de forma eficiente
- Comprender la diferencia entre list.cl y list.s para el desarrollo, compilación e interpretación del programa.

Desarrollo

Realizar el 3er programa video curso Compiladores StanfordOnline

En esta práctica se desarrolló el 3er programa del curso de Stanford donde se realizó un programa que manipuló una estructura de datos en el lenguaje Cool con el fin de entender cómo se realiza la traducción del código fuente de alto nivel a instrucciones ejecutables de un compilador.

List

En esta primera sección se creó un archivo "**list.cl"** con el fin de dar una introducción básica sobre la manipulación de datos, aquí se definió la estructura principal de un "Hello World! ", pero, de una forma poco usual, con una estructura similar a la de una lista (pero no está definida como lista) . Sin embargo, esta estructura sirvió de base para la realización de una lista abstracta.

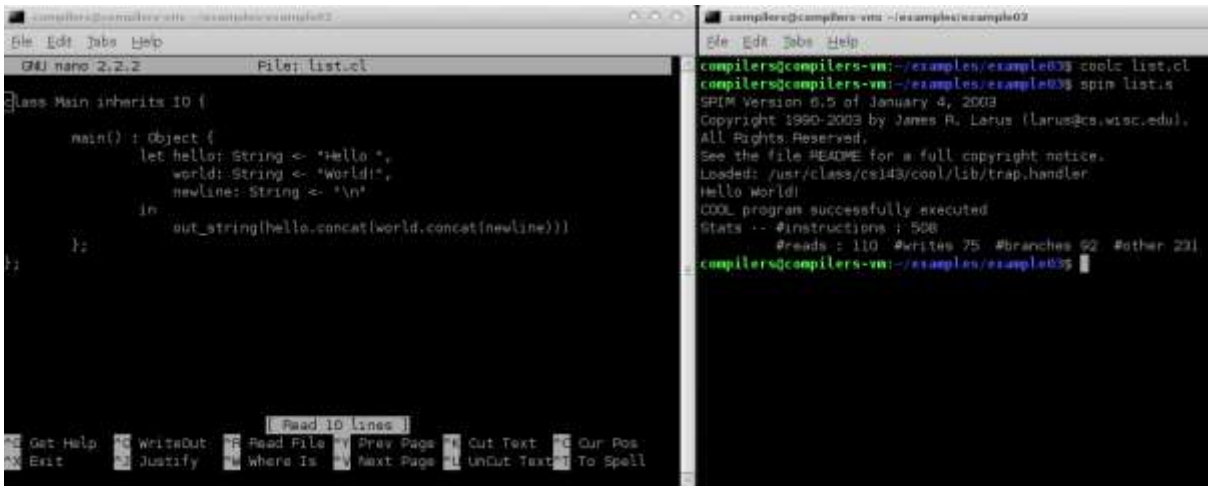

Fig 01. Sección 01 – "Hello Word!" en una List

Como se puede apreciar en la Fig 01 se heredó una clase Input/Output → IO para la gestión de la entrada y salida de datos además se declaró un bloque de variables locales tipo String ("hello"," world", "newline").

Ya con esta estructura se implementó en el programa listas abstractas.

Creando una lista abstracta "List abstraction"

El implementar una lista abstracta sirvió para representar con eficiencia la información ingresada (expresiones-cadenas de texto) durante el proceso de compilación.

Fig 02. List abstract – list.cl

Ahora, lo que se realizó en esta sección fue implementar la clase list que hereda de A2I y tiene dos atributos importantes que son:

1. **Item:** elemento actual de la lista
2. **Next:** referencia la siguiente lista.

Además, con el método init se creó un objeto List que tiene un elemento inicial → i y una referencia a la siguiente lista → n. El elemento Object es flexible porque permite almacenar varios tipos de objetos en la lista.

También se implementó el método flatten que se encarga de convertir una lista en una cadena (String) de acuerdo a los distintos tipos de elementos en este caso Int, String, y Object.

- Si item es Int se convierte en una cadena de caracteres i2a(i) donde, i es intem
- Si es una cadena string se mantiene.
- Si es un un Object se termina de forma abrupta y se devuelve una cadena vacía

Luego, se concatenan los elementos de la lista actual y la lista siguiente con concat. Pero, antes debe verificar que el siguiente elemento "next" no sea nulo.

Finalmente, en el main se imprime el saludo con un nombre: "Hello World! and Camila Sanchez ", esto fue posible con declarar varias cadenas y lista anidadas donde, el bloque:

```
in
   out_string(list.flatten())
};
```

Fig 03. Bloque in

Se ejecuta el método "flatten" en la lista que se definió para que se imprima toda la cadena en forma estándar y legible para el humano.

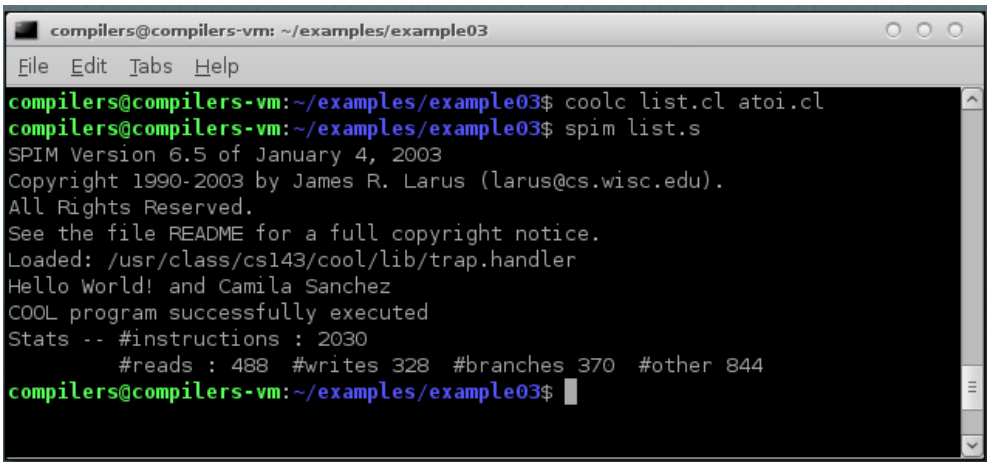

Fig 04. Resultado de crear la List abstract y concatenar varias cadenas

En la figura 03 se observa el resultado de la compilación de ejecutar ***"list.s"***.

Hay que tomar en cuenta que ***"list.s"*** es el código ensamblador que se genera del código fuente ***"list.cl"*** cuando se ejecuta el comando -coolc list.cl atoi.cl- ; asimismo ***"atoi.cl"*** se usa para ejecutar la herencia que se realizó en la clase List de inherits A2I, si es que no se llamaba a este archivo entonces, el código no se podría ejecutar.

Conclusión

En conclusión, el lograr desarrollar y ejecutar el programa *"list.cl"* demuestra cómo se puede manipular estructuras de datos de alto nivel a implementar un código ensamblador que se generó del código fuente, *"list.s"* (ensamblador). Este proceso permitió ejecutar con eficiencia el programa. El programa **"list.cl"** muestra la construcción de una lista abstracta que se enlaza y luego se aplana para que se imprima. Entonces, con la elaboración de este programa se logró comprender como funciona la abstracción para entender cómo funciona la estructura de un compilador.

Referencias

Course. (s/f). Edx.org. Recuperado el 31 de enero de 2024, de https://learning.edx.org/course/course-v1:StanfordOnline+SOE.YCSCS1+3T2020/block-v1:StanfordOnline+SOE.YCSCS1+3T2020+type@sequential+block@01a7248aaffb4f07b4708e11f16c3cfe/block-v1:StanfordOnline+SOE.YCSCS1+3T2020+type@vertical+block@16dbc2207b804703974c1bfc31620eff?_gl=1%2A5kqvm8%2A_ga%2AODM4NDkxNzI0LjE3MDQzNDM4NjU.%2A_ga_D3KS4KMDT0%2AMTcwNjY2MzM2MS4xMS4wLjE3MDY2NjMzNjEuNjAuMC4w