|  |  |
| --- | --- |
| ***Instituto Tecnológico de Costa Rica***  ***Sede Regional San Carlos*** | Tercer Tarea Programada  Compiladores e Intérpretes |
| Prof. Oscar Mario Víquez Acuña | ***Generación de Código*** |

**Descripción:**

La generación de código es el punto máximo en el desarrollo del compilador. Es en esta fase donde a partir de los resultados correctos de las fases anteriores se comienza con la tarea de generar código de máquina.

Para realizar el generador de código se hará uso de la librería Reflection de .NET y por tanto será fundamental que antes y durante la generación del código, el estudiante conozca los conceptos relacionados con la estructura de representación (en este caso el AST) y las técnicas de organización y generación de código de .NET, con el objetivo de diseñar plantillas de código que se adecuen a estos aspectos.

Ahora bien el objetivo fundamental de esta tarea será crear código ejecutable funcional para .NET. Todos los estudiantes deberán generar código de máquina abstracta para .NET (CIL – Common Intermediate Language) y luego generar el código ejecutable utilizando la librería indicada.

Para esto es necesario estudiar, analizar y dominar el set de instrucciones del lenguaje de máquina de .NET (anexo) y la funcionalidad del paquete Reflection.Emit de .NET, con la finalidad de conocer aspectos relacionados propiamente con las instrucciones de máquina, la forma de almacenamiento (uso de pila, etc), la creación de ensamblajes dinámicos y otras.

La técnica para la generación de código será el recorrido mediante “visitors” de manera que el código se irá creando según sean las llamadas en el árbol de los visitors. La representación del código objeto intermedio la define la librería Reflection pero se solicitará que al finalizar el programa, se genere un archivo de texto con el CIL.

Para esta fase se generará código para la mayor parte de la sintaxis y semántica definidas para el lenguaje que estamos implementando, exceptuando los siguientes elementos del lenguaje mini CSharp:

* Importaciones de otros archivos utilizando “using”
* Clases internas con elementos que puedan tener cualquier otro tipo de datos que no sean los tipos simples definidos
* Definición de métodos internos a otros métodos (solo se crearán métodos de nivel global)
* Casting y métodos “add” y “del” para listas (solo “len” será implementado).
* “Write” sin cantidad de repeticiones (*[“,” number]*)
* Usos de new para otras cosas que no sean para crear instancias de clase e instancias de arreglos/listas

El manejo de memoria deberá realizarse siguiendo las especificaciones y restricciones de la librería Reflection. Recordamos que al ser máquina de pila, el manejo de la misma es clave para que funcionen adecuadamente los programas generados.

Como resultado de esta tarea deberá mostrarse en pantalla el código generado a partir de una acción en interfaz que lo indique explícitamente. Así mismo deberá poder correrse el código ejecutable mediante otra acción definida en la interfaz (tipo play común de otros lenguajes de programación – debe existir una opción para compilar y otra para correr). Se debe generar un archivo .exe que pueda ejecutarse de forma independiente.

**Documentación:**

La documentación deberá incluir las siguientes partes:

* Portada formal.
* Soluciones e implementación. Se debe presentar un esquema completo de las principales plantillas de código necesarias para las reglas gramaticales y semánticas (no código generado por algún ejemplo sino el machote para cualquier código). Por ejemplo, en caso de un “while” cuál será la plantilla genérica para cualquier “while” que se presente en el texto fuente.
* Resultados obtenidos. Una completa revisión de aquellos puntos terminados y de aquellos que no se lograran terminar (si fuese el caso). Se debe hacer énfasis especialmente en aquellos segmentos de código que se hayan generado o que no funcionen correctamente.
* Concusiones del trabajo.
* Manual de pruebas. Donde se especifiquen las pruebas necesarias para cuantificar la eficiencia de la generación de código (dos partes: código miniC# y código ensamblador de IL). RECUERDEN LA IMPORTANCIA EN EL CONTENIDO DE LAS PRUEBAS.
* Bibliografía.

**Aspectos Administrativos:**

* La tarea se desarrollará en los mismos grupos con quienes se ha venido trabajando
* La fecha de entrega será el **Lunes 5 de Junio** de 2023 antes de las **10:00** de la noche, en el **TEC\_DIGITAL**.
* Cualquier intento de plagio, copias totales o parciales de otras personas o de Internet, serán castigados con nota de 0.