

**Reporte de proyecto**

Compiladores y desarrollo de librerías

Alumno:

* Jose Jonathan Ramirez Valdovinos

Maestro:

* Gustavo Guzmán

Compiladores:

Los compiladores son programas de computadora que traducen un lenguaje a otro. Un compilador toma como su entrada un programa escrito en su lenguaje fuente y produce un programa equivalente escrito en su lenguaje fuente y produce un programa equivalente escrito en su lenguaje objetivo.

Proceso de traducción:

Un compilador se compone internamente de varias etapas:

Código Fuente:

Manejador de errores

Tabla de literales

Tabla de símbolos

Código objetivo

Optimizador de código objetivo

Código objetivo

Generador de código

Código Intermedio

Optimizador de código fuente

Árbol con anotaciones

Analizador

Semántico

Árbol Sintáctico

Analizador

Sintáctico

Tokens

Analizador

Léxico

Analizador léxico:

Efectúa la lectura real del código fuente y recolecta secuencias de caracteres en unidades significativas denominadas Tokens, las cuales son como las palabras de un lenguaje natural.

Analizador sintáctico:

Recibe el código fuente en la forma de Tokens proveniente del analizador léxico y determina la estructura del programa. Esto es semejante a realizar el análisis gramatical sobre una frase en un lenguaje natural. Los resultados se presentan como un árbol de análisis gramatical o un árbol sintáctico.

Analizador semántico:

Determina su comportamiento durante el tiempo de ejecución, pero la mayoría de los lenguajes de programación tienen características que se pueden determinar antes de la ejecución e incluso no se pueden expresar de manera adecuada como sintaxis y analizarse mediante el analizador sintáctico.

Optimizador de código fuente:

El punto más anticipado en el que la mayoría de las etapas de optimización se pueden realizar es precisamente después del análisis semántico, y puede haber posibilidades para el mejoramiento del código que dependerán solo del código fuente.

Generador de código:

El generador de código toma el código intermedio y genera el código para la maquina objetivo. En esta fase de la compilación en la que las propiedades de la maquina objetivo se convierten en el factor principal. No solo es necesario emplear instrucciones que existan en la maquina objetivo, sino que las decisiones respecto a la representación de los datos desempeñaran ahora también un papel principal, tal como cuantos bytes o palabras de memoria ocuparan las variables de tipos de datos enteros y de punto flotante.

Optimizador de código objetivo:

En esta fase el compilador intenta mejorar el código objetivo generado por el generador de código. Dichas mejoras incluyen la selección de modos de direccionamiento para mejorar el rendimiento. Reemplazando las instrucciones lentas por otras rápidas, y eliminando las operaciones redundantes o innecesarias.

Estructura de datos:

Tokens:

Cuando el analizador léxico reúne los caracteres de un token, generalmente representa el token de manera simbólica, como un valor de un tipo de datos enumerado que representa el conjunto de Tokens del lenguaje fuente.

Árbol sintáctico:

Si el analizador sintáctico genera un árbol sintáctico, por lo regular se construye como una estructura estándar basada en un apuntador que se asigna de manera dinámica a medida que se efectúa el análisis sintáctico. El árbol entero puede entonces conservarse como una variable simple que apunta al nodo raíz. Cada nodo en la estructura es un registro cuyos campos representan la información recolectada tanto por el analizador sintáctico como, por el analizador semántico.

Tabla de símbolos:

Esta estructura de datos mantiene la información asociada con los identificadores: funciones, variables, constantes y tipos de datos. La tabla de símbolos interactúa con casi todas las fases del compilador: analizador léxico, analizador sintáctico o el analizador semántico puede introducir identificadores dentro de la tabla; el analizador semántico agregara tipos de datos y otra información; y las fases de optimización y generación de código utilizaran la información proporcionada por la tabla de símbolos para efectuar selecciones apropiadas de código objeto.

Tabla de literales:

La búsqueda y la inserción rápida son esenciales también para la tabla de literales, la cual almacena constantes y cadenas utilizadas en el programa. Sin embargo una tabla de literales necesita impedir las eliminaciones porque sus datos se aplican globalmente al programa y una constante o cadena aparecerá solo una vez en la tabla. La tabla de literales es importante en la reducción del tamaño de un programa en la memoria, al permitir la reutilización de constantes y cadenas. También es necesaria para que el generador de código construya direcciones simbólicas para literales y para introducir definiciones de datos en el archivo de código objetivo.

Código intermedio

De acuerdo con la clase de código intermedio y de las clases de optimizaciones, este código puede conservarse como un arreglo de cadenas de texto, un archivo de texto temporal o bien una lista de estructuras ligadas.

Lenguaje TINY:

Es una secuencia de sentencias separadas mediante signos de punto y coma. No hay procedimientos ni declaraciones. Todas las variables son enteras, y las variables son declaradas simplemente al asignar valores a las mismas. Tiene dos sentencias de control: “if” y “repeat”, y ambas pueden tener secuencias de sentencias. Una secuencia “if” tiene una parte opcional “else” y debe terminarse con la palabra reservada “**end**”.También existen sentencias de lectura y escritura que realizan entrada/salida. Los comentarios se permiten dentro de llaves. Solo permite expresiones aritméticas enteras y booleanas.

{Programa de muestra}

**read x;**

**if x > 0 then**

**fact := 1;**

**repeat**

**fact := fact \* x;**

**x := x – 1;**

**until x = 0;**

**write fact**

**end**

Compilador TINY:

Se compone de los siguientes archivos en C, donde enumeramos los archivos de cabera y los archivos de código.

globals.h main.c

util.h util.c

scan.h scan.c

parse.h parse.c

symtab.h symtab.c

analyze.h analyze.c

code.h code.c

cgen.h cgen.c

El archivo main.c contiene el programa principal que controla el compilador y asigna e inicializa las variables globales.

Los archivos scan, parse, analyze y cgen corresponden exactamente a las fases del analizador léxico, analizador sintáctico, analizador semántico y generador de código.

Bibliografía:

Construcción de compiladores principios y practica (Louden, 2004)