Unidad 8

Intérpretes

Contenido

- Introducción
- Ventajas y Desventajas
- Estructura de un Intérprete
- Gestión y Liberación de Memoria
- Tipos de Intérpretes

Introducción

Definición de Intérprete

- Un Intérprete analiza un programa que está escrito en algún lenguaje de alto nivel y lo ejecuta de manera directa en el lenguaje de la máquina en que se está ejecutando el intérprete.
- Cuando se quiere ejecutar nuevamente el programa, es necesario interpretarlo otra vez.

Ejemplo

- Java es un lenguaje manejado por este tipo de software, primero se traduce por un compilador-intérprete para generar algo conocido como BYTECODE y posteriormente se interpreta por la máquina virtual de Java.

Compilador - Intérprete

- Existe un Compilador-Intérprete que traduce el programa fuente a un formato o lenguaje intermedio que después se interpreta

- La mayoría de los Intérpretes son en realidad Compiladores-Intérpretes y su función se realiza en dos pasos o etapas:
- Etapa de Compilación o de Introducción del Programa.
- Etapa de Interpretación o de Ejecución del Programa.

Etapa de Compilación

- El código fuente de origen se compilar y se traduce a un formato o lenguaje intermedio.
- Este lenguaje normalmente no coincide con algún lenguaje simbólico de alto nivel
- La etapa de Compilación normalmente se realiza solamente una vez

Etapa de Interpretación

- El programa compilado a lenguaje intermedio se interpreta y se ejecuta, esta operación se realizará tantas veces como se desee ejecutar el programa

Necesidad de Uso de Intérpretes

- Entre las razones para el uso de Intérpretes se tienen las siguientes:
- Se ha eliminado del lenguaje la gestión dinámica de memoria dejándole este manejo al Intérprete y quitándole la responsabilidad al programador
- En algunos lenguajes se ha eliminado la declaración de variables siendo esto un procedimiento implícito
- Se desea que un lenguaje sea capaz de ejecutarse en cualquier plataforma o sistema operativo, por ejemplo Java puede ejecutarse incluso en navegadores web gracias a un Intérprete de bytecode en los navegadores

Ventajas y Desventajas

Ventajas de los Intérpretes

- Existen diversas ventajas que presentan los Intérpretes respecto a los Compiladores
- Flexibilidad
- Facilidad de Depuración

Flexibilidad

- Los lenguajes que utilizan un Intérprete normalmente son más flexibles y pueden realizar algunas acciones que podrían resultar complejas para un compilador, por ejemplo:
- No necesitar de declaraciones, lo que reduce la carga de trabajo del programador que no se preocupa por declarar variables, aunque esto en ocasiones dificulta el proceso de Depuración.

Flexibilidad

- Manejar de manera más sencilla el uso de memoria dinámica ya que este manejo se le pasa al Intérprete
- En algunos casos facilita el manejo de un lenguaje orientado a objetos ya que el Intérprete tiene acceso a la Tabla de Símbolos en el momento de la ejecución

Ciclo de Desarrollo con un Intérprete

- Es común realizar diversas modificaciones durante el desarrollo de un programa
- En un lenguaje compilado, es necesario volver a compilar para crear el ejecutable y ver los cambios
- Como un intérprete solo debe generar un código intermedio, solo es cuestión de guardar los cambios para que se vean reflejados en la ejecución

Facilidad de Depuración

- En un lenguaje que utiliza un Intérprete, el proceso de depuración es más sencillo, ya que permite interrumpir el proceso en cualquier momento (cosa que con un compilador también es posible), pero lo que un compilador no puede es reiniciar la depuración en el punto en donde se interrumpió, cosa que un Intérprete no puede realizar.

Facilidad de Depuración

- Cuando en un programa compilado se detecta un error, se interrumpe la depuración, se corrige y se genera un nuevo ejecutable
- Con un lenguaje interpretado, se detecta un error, se corrige y se puede comenzar la depuración a partir del punto en donde se había encontrado el error

Desventajas de los Intérpretes

- También existen algunas desventajas de los Intérpretes respecto a los Compiladores
- Velocidad de los Programas Ejecutables
- Tamaño del Programa Objeto

Velocidad de Programas Ejecutables

- Los programas interpretados suelen ser más lentos que los compilados, esto se debe a que un programa compilado realiza varios análisis necesarios para el procesos solo una vez y genera código ejecutable en la máquina.
- Un Intérprete cada que se ejecuta debe realizar un determinado análisis al momento de la ejecución por lo que siempre se produce una carga de trabajo extra.

Tamaño del Programa Objeto

- Lo que podría considerarse como programa compilado en un entorno de Intérpretes, se construye agregando una parte del mismo Intérprete al programa fuente, por lo que se agrega un tamaño considerablemente mayor al del mismo código en un lenguaje compilado.

Estructura de un Intérprete

Elementos de un Intérprete

- La estructura de un Intérprete es muy parecida a la de un compilador, contiene varios elementos similares: analizadores morfológicos, sintáctico y semántico además de una tabla de símbolos o de identificadores.

- Cuenta también con una sección para manejar la memoria y otra para procesar errores.

- La principal diferencia es que un Intérprete no contiene una etapa de generación de código que se sustituye por una componente de ejecución de código ya que un Intérprete no genera código.

Traductor a Representación Interna

- Es el encargado de convertir el código fuente de entrada y convertirlo a una representación adecuada para el Intérprete.

Tabla de Símbolos

- Almacena información relacionada con los símbolos que van apareciendo
- Entre la información que se almacena se encuentran etiquetas, identificadores u otro tipo de información dependiendo del lenguaje utilizado

Evaluador de Representación Interna

- Es la sección encargada de realizar las operaciones a partir de los datos de entrada y de la salida del Traductor de la Representación Interna

Recuperación de Errores

- El manejo de errores en un Intérprete se puede ver desde dos enfoques, cuando se ejecuta bajo control total del Intérprete y cuando se empaqueta una parte del Intérprete junto con el código para su ejecución.

Intérprete Encapsulado

- Es en este caso en donde se debe tener un control más complejo de los errores que puedan aparecer, en estos casos lo que se hace es manejar una aplicación independiente que tiene un procesador de errores más simple y restringido que el del Intérprete completo.

Control Total de un Intérprete

- Si un intérprete detecta un error cuando tiene el control total de la ejecución, lo más conveniente es detener la ejecución e indicar el error detectado con un mensaje para permitir al programador que se tome alguna acción.
- Algunos lenguajes que funcionan con Intérpretes tienen la ventaja de que ofrecen la secuencia de instrucciones que produjeron el error, lo que hace más sencillo rastrearlos y corregirlos.

Gestión y Liberación de Memoria

Gestión y Liberación de Memoria

- La memoria que se asigna a un programa objeto se maneja como memoria dinámica en una zona denominada ***Heap***, tomándose memoria para cada variable cuando es necesaria y liberándola cuando deja de ser utilizada.

- Cómo el manejo de memoria lo realiza el mismo Intérprete, el programador no tiene que preocuparse de este manejo.

- La liberación de memoria de manera dinámica puede provocar que esta se fragmente, por lo que se debe realizar de manera correcta, eso se realiza con algo denominado Recolector de Basura

El Recolector de Basura

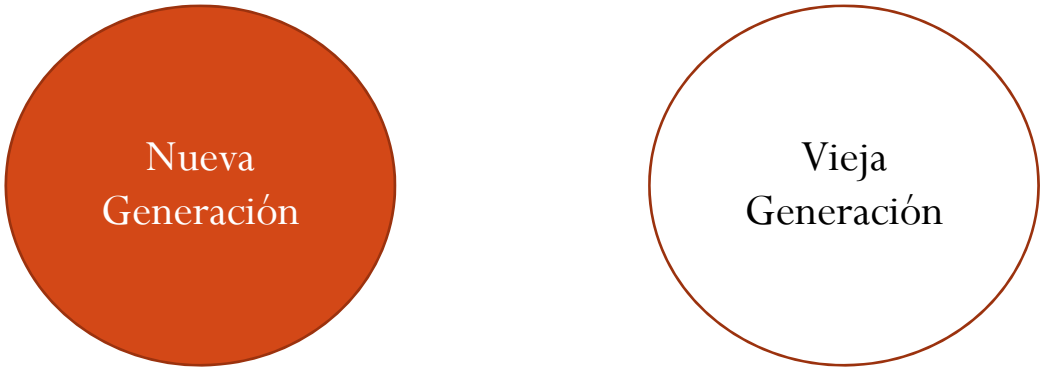
- Un recolector de basura es un mecanismo para el manejo de la memoria en un lenguaje interpretado
- Esta liberación puede ser de dos tipos:
  - Explícita. Siendo el programador el responsable de la liberación
  - Implícita. Cuando es un mecanismo del intérprete el que hace la liberación
- Un objeto se dice elegible por el Recolector cuando se pierde la referencia hacia él

Pérdida de Referencias

- Existen varias formas en las que se pierde la referencia de un objeto:
  - Asignarle un valor nulo (*null*) al objeto
  - Cuando a un objeto se le asigna uno nuevo, el valor del primero pasa a ser inaccesible y puede ser removido
  - Cuando solo es accesible durante un momento específico, por ejemplo, cuando se crea en una estructura condicional o iterativa
- Considerar que si un objeto A se asigna a un objeto B y luego A se asigna como nulo, también puede ser removido

Recolector de Basura en Java

- Cuando se ejecuta un programa en Java, se crean dos bloques de memoria principales



Nueva Generación

- En esta zona de memoria se almacenan los objetos que han sido creados recientemente
- Esta zona está dividida en tres:
  - Edén
  - Supervivencia S0
  - Supervivencia S1

Vieja Generación

- La zona de la Vieja Generación se divide en dos secciones:
  - Generación Titular
  - Generación Permanente

- En la zona de Generación Permanente, se encuentran las clases de Java necesarias para la ejecución de la JVM

El proceso de Recolección

- Cuando se crea un objeto, se almacena en la sección Edén de la Nueva Generación
- Cuando el recolector revisa la memoria, si un elemento en la sección Edén ya no tiene referencias, es colocado en el nivel S0 de Supervivencia
- Después de un tiempo, el recolector ahora colocará los elementos del nivel S0 en el nivel S1 de Supervivencia
- En una nueva pasada por la memoria, moverá los elementos que estén en S1 al nivel de Generación Titular
- Los objetos ahí se eliminarán solo si el espacio en memoria ya es insuficiente

Algoritmos de Recolección

Tipos de Intérpretes

Clasificación de los Intérpretes

- Los intérpretes se clasifican según su estructura interna y se cuenta con las siguientes clasificaciones:
  - Intérpretes puros
  - Intérpretes avanzados

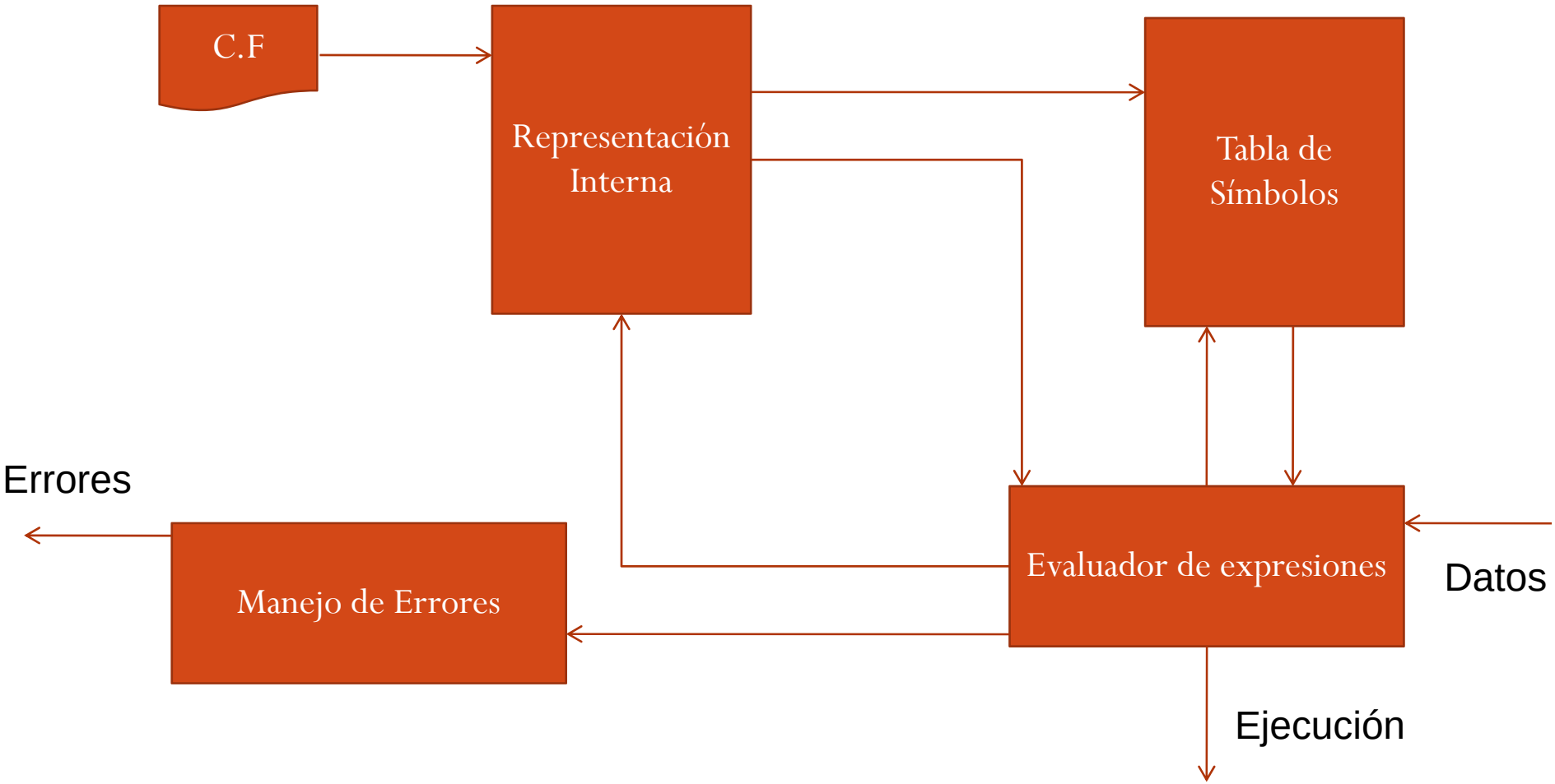
Intérpretes Puros

- Son aquellos que analizan y ejecutan sentencia por sentencia todo el código fuente, siguen un modelo de interpretación iterativa y se usan para lenguajes simples

- El código fuente se traduce a una representación interna que se almacena ya sea en la memoria o en el disco

- Al mismo tiempo se construye la tabla de símbolos y al finalizar se comienza con la ejecución de la primer instrucción la cuál es evaluada por el Evaluador de Instrucciones que la ejecuta y determina la siguiente instrucción a ejecutar

Estructura de un Intérprete Puro



Intérpretes Avanzados

- Son los más utilizados e incluyen un primer paso en donde se analiza todo el código fuente, posteriormente se genera un lenguaje intermedio

- Esto permite que si hay errores de sintaxis no se pase a la etapa de ejecución

- Se utilizan para lenguajes más avanzados ya que permiten analizar de manera más detallada el código fuente

Estructura de un Intérprete Avanzado