GENERALIDADES DE LOS LENGUAJES DE PROGRAMACIÓN

AUTOR: WALTER MADRIGAL CHAVES

NOVIEMBRE: 2020

San Marcos

Contenido

Introducción

La creciente demanda de sistemas informáticos hace que más personas se dediquen al desarrollo de sistemas en busca de nichos de trabajo, sin embargo, no todos se capacitan lo necesario para asegurar la creación sistemas de calidad. El aprendizaje de lenguaje especifico no debería ser un punto de partida, si no un eslabón más en una larga cadenas de conocimientos.

El mercado ofrece distintos tipos de lenguajes de programación, cada uno con sus diferentes características, virtudes y desaciertos. Todos convergen en la premisa de que un manejo adecuado de la memoria da como resultado sistemas más eficientes y una mejor administración de los recursos.

En la siguiente lectura se analizarán diferentes generalidades de los lenguajes de programación, se ahondará en temas como: manejo de memoria, tipos de memoria, depuradores de errores lógicos y de sintaxis y los métodos con sus diferentes variantes.

Lenguajes de programación

Los lenguajes de programación son un tipo de software que tiene la virtud de crear nuevos sistemas, su función es dar una interfaz con sintaxis comprensibles para los seres humanos, para posteriormente trasladar las sentencias creadas a un código entendible por las computadoras. Es un intermediario entre el programador y equipo que ejecuta el código.

Uso de memoria

La memoria es un espacio lógico para guardar información en la computadora, retienen los datos por periodos de tiempo y su correcta administración es muy importante para rendimiento del equipo y por consiguiente de los sistemas que corren en él.

A nivel de software debe haber una buena gestión de la memoria, esto implica utilizar los recursos de forma justa, declarar variables adecuadas al dato que se almacenara y utilizar estructuras de almacenamiento adecuadas al caso. Existen dos tipos estructura de administración interna de los datos, las estáticas y los dinámicos.

Memoria estática

Este tipo de memoria se aplica cuando se declaran estructuras en las cuales su tamaño no varía durante la ejecución del programa, por ejemplo, cuando se declaran vectores y matrices es necesario definir el tamaño que tendrá desde un inicio, en ese momento es cuando se reserva en memoria un espacio para ese arreglo. Las siguientes son algunas consideraciones de utilizar este tipo de memoria:

- Puede ocurrir errores en tiempo de ejecución de índice fuera del rango: Esto sucede cuando se intenta almacenar información en una estructura que utiliza memoria estática y está lleno, al no poder aumentar su

capacidad no hay donde guardar los datos.

- Se debe conocer con anticipación el tamaño de la estructura: como no puede aumentar su espacio en memoria durante la ejecución, es necesario pronosticar su tamaño desde un inicio.

- Algunas de las estructuras que utilizan memoria estática son: vectores, matrices, cubos, registros y archivos.

- Dentro de las ventajas que tienen estas estructuras están: velocidad de lectura, utilización de lógica simple y fácil de codificar.

- Algunas desventajas son: no se puede modificar el tamaño de la estructura en tiempo de ejecución, no es óptimo con grandes cantidades de datos, hay desperdicio de memoria cuando no se utiliza en su totalidad del tamaño y menor capacidad, debido a que cada celda de almacenamiento requiere más transistores.

Memoria dinámica

Este tipo de memoria no puede ser definida en un inicio, esto debido a que se desconoce o no se tiene idea del número de las variables a considerarse, la memoria dinámica permite a los elementos solicitar más memoria según necesidad, así se utiliza justo lo necesario.

Una información interesante es que este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del sistema, impidiendo así perder datos o desperdiciar memoria, en otras palabras, hace un uso justo de los recursos.

Según (Rosas, 2020) cuando se crea un programa en el que es necesario manejar memoria dinámica el sistema operativo divide el programa en cuatro partes que son: texto, datos (estáticos), pila y una zona libre o heap. En la última

parte es donde queda la memoria libre para poder utilizarla de forma dinámica.

Las siguientes son algunas consideraciones de utilizar este tipo de memoria:

- Es memoria que se reserva en tiempo de ejecución. Su tamaño puede crecer o disminuir durante la ejecución del programa, haciendo uso estrictamente de los recursos necesarios.
- Siempre utiliza punteros.
- Su implementación es más compleja y tiende a dar errores si no tiene un manejo experto.
- Es mucho más lento un programa que la implemente pues ocupa más instrucciones. La memoria dinámica puede afectar el rendimiento.
- Hay que programar muchas más instrucciones y realizar varias tareas, como buscar un bloque de memoria libre y almacenar la posición y tamaño de la memoria asignada, de manera que pueda ser liberada más adelante. Todo esto representa una carga adicional, aunque esto depende de la implementación y hay técnicas para reducir su impacto.

Tipos de memoria en el lenguaje de programación

Al momento de ejecutar un programa se involucran diferentes tipos de memorias o formas de ella, estas son:

- **Memoria de Código**: después de compilado el código fuente de un software, se genera el código máquina, que tambien suele llamarse ejecutable, cuando este se carga a la computadora para su ejecución está haciendo uso de memoria de código.

- **Memoria de datos estáticos:** un ejemplo de utilización de este tipo de memoria son las variables static que tienen la particularidad de que viven durante toda la ejecución del programa. Es decir, que se les reserva memoria durante el inicio del programa y solo se desocupa esa memoria, al terminar el programa.

- **Memoria de Pila:** para ejemplificar este tipo de memoria están los parámetros de las funciones, estos variables solo viven durante la ejecución de la función, es decir que nacen al ingresar el llamado a la función y mueren al retornar el control a la función que la llamó.

- **Memoria Heap:** Este tipo de memoria es dinámica y se reserva durante la ejecución del sistema, puede crecer o decrecer según la necesidad.

Depurador

En términos genérales el concepto de depurar se refiere a limpiar algo, a nivel de programación es un término muy utilizado para mostrar la limpieza del código en el caso que posea errores. Un depurador es un sistema que permite detectar y diagnosticar fallos en programas informáticos.

El objetivo de los depuradores es certificar, que el software funcione en todos los dispositivos y plataformas para los que está pensado. Por este motivo, muchos depuradores no solo analizan el código fuente del programa, sino también su interacción con el sistema operativo que lo ejecuta y con los elementos de hardware.

Cuando se depura un programa se hace un seguimiento del funcionamiento de dicho programa y se van estudiando los valores de las distintas variables, así como los resultados obtenidos en las operaciones. Se dice que un programa esta depurado cuando está libre de errores.

Bug

Es el nombre técnico que reciben los errores en un programa. La aparición de un bug no siempre implica que un programa se caiga o se cierre repentinamente, a veces simplemente significa que el resultado no es el esperado o que el rendimiento es muy pobre.

La técnica llamada debugging (depuración), realiza el proceso de encontrar y eliminar los errores que pueden impedir que los códigos funcionen de forma adecuada.

Cada herramienta de desarrollo tiene su propio instrumento para hacer debugging al código. A través de ellas, se puede crear puntos de parada, conocidos como break points, para verificar el estado en el momento de la aplicación.

Métodos

Un método es una subrutina que tiene una serie de sentencias para llevar a cabo una acción, está ligado a un objeto o clase, posee parámetros que permiten la entrada de datos y generan un valor de salida. Una clase puede declarar cualquier número de métodos que lleven a cabo operaciones con los objetos.

Métodos de clase con objetos

Gracias a los métodos se pueden escribir programas modulares y se consigue la reutilización de código, es importante cuando se trabaja con métodos tener en cuenta los siguientes puntos:

- Nos es posible crear un método sin una clase.
- No existe el concepto de anidar, un método no se puede escribir dentro de otro, pero si invocar a otro
- Todo programa en java tiene un método llamado main, que es el más

importante de todos, la ejecución del programa empieza en este método.

- Los métodos tienen un único punto de inicio y de fin. El cuerpo es encerrado entre llaves.
- La palabra reservada return puede aparecer en cualquier lugar dentro del método, no tiene que estar necesariamente al final.
- Cuando un método finaliza, la ejecución del programa continúa a partir del punto donde se produjo la llamada al método.

Imagen 1 Métodos en Java

```
1   import java.util.Scanner;
2
3   public class Metodos1 {
4
5       public static void main(String[] args) {
6
7           Scanner sc = new Scanner(System.in);
8           int numero1, numero2, resultado;
9           System.out.print("Introduce primer número: ");
10          numero1 = sc.nextInt();
11          System.out.print("Introduce segundo número: ");
12          numero2 = sc.nextInt();
13          resultado = sumar(numero1, numero2);
14          System.out.println("Suma: " + resultado);
15      }
16
17      //método sumar
18      public static int sumar(int a, int b){
19              int c;
20              c = a + b;
21              return c;
22      }
23
24  }
```

Fuente: http://puntocomnoesunlenguaje.blogspot.com/201/04/metodos.html

En el ejemplo anterior se crean dos métodos, el main (el principal) y el llamado sumar, este último toma dos números obtenidos por parámetro y devuelve su suma. Como se aprecia en la línea 13 se invoca desde el método main al que se llama sumar.

Los métodos se clasifican en dos grupos: los de instancia y los de clase.

Métodos de instancia

Un método de instancia es el que se invoca o se usa siempre sobre una instancia (objeto) de una clase. Por ejemplo, en Java se aplica este concepto cuando se incluye un método en una definición de una clase sin utilizar la palabra clave static.

Cuando se invoca o llama a un método de instancia a través de un objeto, si este método referencia a variables de instancia de la clase, en realidad se están referenciando variables de instancia específicas del objeto específico que se está invocando. Los métodos de instancia tienen acceso tanto a las variables de instancia como a las variables estáticas.

La llamada a los métodos de instancia en Java se realiza utilizando el nombre del objeto, el operador punto y el nombre del método.

```
miObjeto.miMetodoDeInstancia();
```

Métodos de clase

Son tambien llamadas estáticos, en su declaración se utiliza la palabra reservada static, en este tipo de estructuras pueden ser invocados sin necesidad de instanciar ningún objeto de la clase. Estos métodos se pueden invocar de la siguiente manera:

```
MiClase.miMetodoDeClase();
```

Los métodos de clase trabajan solamente con variables estáticas, no tienen acceso a las variables de instancia, analicemos el siguiente ejemplo:

Imagen 2 Métodos

```
class persona {
    static int identificador = 10;
    int edad;


    static void add_edad() {
       edad++; // esto no funciona
    }


    static void modifica_identificador( int i ) {
        identificador++;        // esto si funciona
    }
}
```

Fuente: Elaboración propia.

Hay dos variables la primera de tipo static y la segunda es normal, al intentar modificar la variable edad desde un método del tipo clase o static, no funcionara, se estaría violando las reglas de acceso. El segundo método no tendría problemas pues ambos son de tipo static.

Conclusiones y recomendaciones

- El manejo de memoria es un tema que no se puede pasar por alto, un adecuado uso de los recursos es tema primordial para los programadores.

- Algunas veces se debe decidir durante el desarrollo del sistema que es más importante economizar, en recursos usando estructuras de memoria dinámicas o velocidad usando estructuras de datos estáticas.

- La eliminación de errores de sintaxis y lógica durante la concepción de un sistema es un trabajo muy importante que lo realiza los depuradores, por supuesto no se debe dar toda la responsabilidad a este instrumento, los programadores al final tienen la obligación de entregar un programa limpio.

- Los métodos son grandes herramientas e indispensables (como el método main en java). Es importante hacer un uso adecuado de ellos para aprovechar todas las ventajas que ofrecen.

Referencias bibliográficas

Buriticá, O. T. (2017). *Lógica de programación.* Bogotá: Ediciones de la U.

Herrera Morales, J., Gutiérrez Posada, J., & Pulgarín Giraldo, R. (2017). *Introducción a la lógica de programación.* Armenia: ElIZCOM S.A.A.

Rosas, O. (01 de 12 de 2020). *Compilando conocimiento*. Obtenido de https://compilandoconocimiento.com/2016/12/2/memoria-dinamica/

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica