**Variable estática**

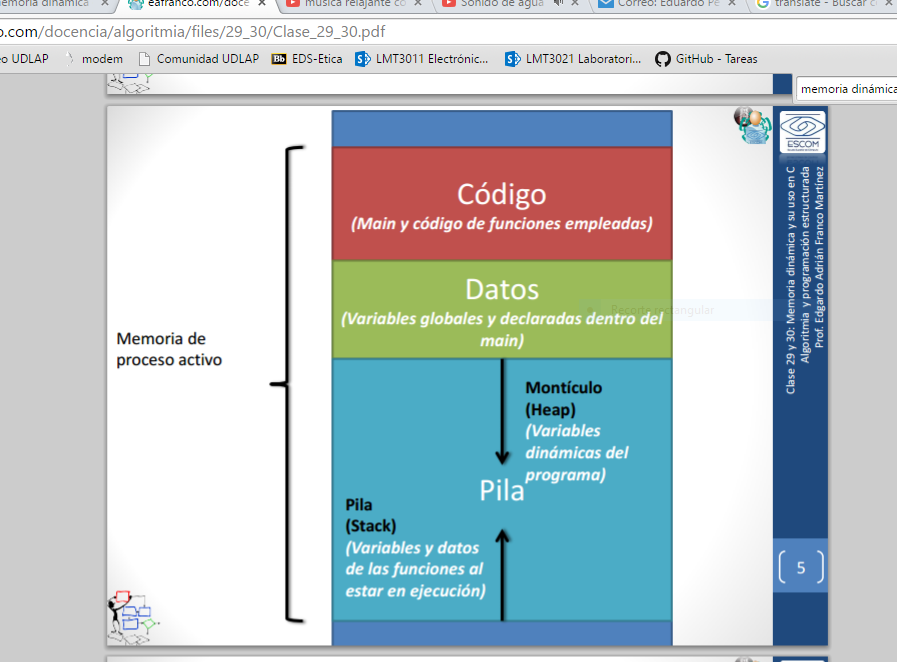
Una variable estática se caracteriza por mantener un tiempo de vida extendido durante toda la ejecución del programa. Otro termino más común para definir una variable es mencionarla como una variable global(que todo el programa conoce).

Una variable estática prácticamente es un mensaje para todo el contexto de programa, por ejemplo en el lenguaje java la clase principal o de prueba contiene el método, *public static void main* este método contiene el control de todo el programa dando órdenes a cada elemento del programa.

**Ciclo de vida de las variables**

En cuanto al ciclo de vida de las variables depende el tipo de cada una. Por ejemplo: existen variables de instancia(objeto) se crean cuando el objeto es creado, se destruyen cuando el recolector de basura de Java no encuentra referencias activas para el objeto. Además cabe mencionar que Java es un lenguaje que utiliza el paradigma orientado a objetos. Por otro lado las variables estáticas(o de clase), se crean cuando la clase se usa por primera vez, suelen existir para el resto del programa(salvo que no esté cargado). Agregando las variables locales(o de bloque) son creadas en la sentencia en la que están definidas, estas pueden adquirir diferentes valores durante la ejecución del programa. Sin embargo son destruidas al salir del bloque(en la llave final).

**Memoria dinámica (lenguaje C)**

En cuanto al manejo de memoria dinámica es la base del poder del lenguaje C y le da la capacidad de crear programas complejos que emplean grandes cantidades de memoria y los maneja de manera eficiente.

Las variables globales y del programa (main) se almacenan en posiciones fijas de la memoria llamada *memoria de datos.*

La ilustración muestra gráficamente como está constituida la memoria dinámica.

**Clase**

La clase contiene un nombre para identificarla, atributos que son las características de la misma clase, en otros términos los atributos son variables que contendrá la clase. Cabe mencionar que el paradigma orientado a objetos se caracteriza por la encapsulación obligando al programador que los atributos de clase tengan una visibilidad que solo pueda ser accedida por la misma clase o en dado caso este protegida pero en ningún caso debe ser accedida por cualquier otra clase diferente. La clase también tiene contiene métodos que son acciones que desempeña una clase en la ejecución programa, a diferencia de los atributos los métodos deben de ser visibles para todo el programa. Por otro lado un método puede ser visto como una función que recibe argumentos para trabajar o en dado caso simplemente puede ser invocado. Además un método puede o no regresar un resultado.

**Objeto**

En el paradigma orientado a objetos, un objeto puede ser relacionado con el mundo exterior cuya característica principal es que tiene comportamiento diferente durante toda la ejecución de programa, por ejemplo en determinado tiempo puede recibir un dato pero si este sobrepasa un parámetro puede rechazar la entrada de datos. Otra característica importante es que los objetos interactúan entre ellos durante la ejecución del programa, esto se puede apreciar en el diagrama de clases del programa implementado.

**Instanciación**

En el [paradigma](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) [orientado a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), una instancia se refiere a una creación específica de una clase. En otros términos una instancia se refiere a la creación e inicialización de un objeto.

**Herencia**

En el paradigma orientado a objetos el concepto herencia hace referencia a la creación de una nueva clase a partir de una clase existente. Además en la herencia se puede observar el funcionamiento de los atributos protegidos. Cuando una clase hereda sus atributos y métodos, esta clase es conocida como superclase, en cuanto a las clases que adquieren las características de la superclase, estas clases son conocidas como subclases de la superclase.

**Sobrecarga**

La sobrecarga en el paradigma orientado a objetos hace referencia a utilizar varios métodos y constructores con el mismo identificador pero con diferentes parámetros, es decir cuando se invoca algún método con ‘n’ parámetros, el programa buscara el método adecuado para que puedan encajar los datos recibidos. De misma manera funciona con los constructores de objetos.

**Ensombrecimiento**

El termino ensombrecimiento se refiere a la existencia de dos variables con el mismo nombre pero que son vistas desde diferentes puntos de vista, es decir puede existir una variable con el mismo nombre pero puede ser local(estar dentro de una función) y al mismo tiempo existir otra con el mismo nombre pero ser global (visible para todo el programa). El compilador permite este tipo de declaración pero cuando se ejecuta la función se produce el ensombrecimiento, la variable global no puede ser accedida, pues ese nombre se refiere a la variable local a la función. Al terminar la función, la variable global ya no está “ensombrecida” en consecuencia vuele a ser accesible para el programa.

**Referencias**

* Universidad Carlos III de Madrid.(sin año).*Capitulo3. Declaración de variables.* Recuperado el 30 de septiembre de 2015 de: http://www.it.uc3m.es/labas/course\_notes/variables\_es.html
* Bel Puchol, Antonio.(2001)*.Clases – Miembros estáticos.* Recuperado el 30 de septiembre de 2015 de: http://www.arrakis.es/~abelp/ApuntesJava/ClasesIII.htm
* *Ámbito y acceso*. Recuperado el 30 de septiembre de 2015 de: <http://mit.ocw.universia.net/1.00/s02/class-sessions/lecture-11/lecture-11.pdf>
* *Compresión del ciclo de vida de las variables*. Recuperado el 30 de septiembre de 2015 de: https://msdn.microsoft.com/es-es/library/office/gg278427.aspx
* Franco, Edgardo . (sin año). *Clase 29 y 30: Memoria dinámica y uso en C*. Recuperado el 30 de septiembre de 2015 de:http://eafranco.com/docencia/algoritmia/files/29\_30/Clase\_29\_30.pdfG