Cuestionario N°1

Materia: Estructura de Datos

Profesor: Martín Santoro[[1]](#footnote-0)

1. **¿Qué es un IDE? Cite ejemplos.**

Un IDE (“Integrated Development Environment”, o “Entorno de Desarrollo Integrado”)

es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador.

La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense).

Algunos IDE contienen un compilador, un intérprete, o ambos (NetBeans y Eclipse); otros no (SharpDevelop y Lazarus). Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos.

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Estos, presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software.

**Diferentes ejemplos de IDE:**

1. **Eclipse**: Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados , como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). También se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus. En Eclipse se pueden usar diferentes lenguajes de programación como: Java, ANCI C, C++, JSP, sh, perl, php, sed.
2. **NetBeans**: Programa que sirve como IDE que permite programar en distintos lenguajes, es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), además ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. La IDE de NetBeans es perfecta y muy comoda para los programadores. Tiene un excelente balance entre una interfaz con múltiples opciones y un aceptable completamiento de código.
3. **Geany**: Es un IDE que hasta hace bien poquito sólo estaba disponible para sistemas Linux, Mac OS X y BSD, pero ya está disponible para Windows. Este entorno es muy sencillo, pero proporciona las funcionalidades necesarias para desarrollar aplicaciones sin problemas. Su interfaz está dividida en tres zonas: panel lateral con el árbol de carpetas y documentos abiertos, sección principal para el código y panel inferior para los mensajes de la aplicación, compilación, etc. Este IDE permite programar en diferentes lenguajes como: C, [C++](https://www.ecured.cu/C%2B%2B), [Java](https://www.ecured.cu/Java), Python, Pascal, [SQL](https://www.ecured.cu/SQL) o HTML
4. **MS Visual Studio:** Microsoft **Visual Studio** es un entorno de desarrollo integrado (**IDE**, por sus siglas en inglés) para sistemas operativos Windows. ... **Visual Studio** permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET
5. **¿Qué es Programación Estructurada?**

La programación estructurada es un [paradigma de programación](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) orientado a mejorar la claridad, calidad y tiempo de desarrollo de un [programa de computadora](https://es.wikipedia.org/wiki/Programa_de_computadora) recurriendo únicamente a [subrutinas](https://es.wikipedia.org/wiki/Subrutina) y tres estructuras básicas: [secuencia](https://es.wikipedia.org/wiki/Bloque_de_c%C3%B3digo), selección ([if](https://es.wikipedia.org/wiki/If) y [switch](https://es.wikipedia.org/wiki/Switch)) e [iteración](https://es.wikipedia.org/wiki/Iteraci%C3%B3n) ([bucles](https://es.wikipedia.org/wiki/Bucle_(programaci%C3%B3n)) [for](https://es.wikipedia.org/wiki/Bucle_for) y [while](https://es.wikipedia.org/wiki/Bucle_while)).

El [teorema del programa estructurado](https://es.wikipedia.org/wiki/Teorema_del_programa_estructurado), propuesto por Böhm-Jacopini, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

* Secuencia.
* Instrucción condicional.
* Iteración (bucle de instrucciones) con condición inicial.

Solamente con estas tres estructuras se pueden escribir todos los [programas](https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico) y aplicaciones posibles. Si bien los [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) tienen un mayor repertorio de [estructuras de control](https://es.wikipedia.org/wiki/Estructuras_de_control), estas pueden ser construidas mediante las tres básicas citadas.

1. **¿Qué es programación Orientada a Objetos?**

La programación orientada a objetos (POO, u OOP según sus siglas en inglés) es un [paradigma de programación](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Muchos de los objetos pre-diseñados de los [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguajes_de_programaci%C3%B3n) actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

Está basada en varias técnicas, incluyendo [herencia](https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)), [cohesión](https://es.wikipedia.org/wiki/Cohesi%C3%B3n_(inform%C3%A1tica)), [abstracción](https://es.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)), [polimorfismo](https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)), [acoplamiento](https://es.wikipedia.org/wiki/Acoplamiento_(inform%C3%A1tica)) y [encapsulamiento](https://es.wikipedia.org/wiki/Encapsulamiento_(inform%C3%A1tica)).

La programación orientada a objetos difiere de la [programación estructurada](https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada) tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos.

Los programadores que emplean POO, primero definen objetos para luego enviarles mensajes solicitandoles que realicen sus métodos por sí mismos.

1. **¿Qué es un lenguaje de bajo nivel?**

Los lenguajes de bajo nivel, son lenguajes totalmente dependientes de la máquina, es decir que el programa que se realiza con este tipo de lenguajes no se pueden migrar o utilizar en otras máquinas. Al estar prácticamente diseñados a medida del hardware, aprovechan al máximo las características del mismo.

Dentro de este grupo se encuentran:

1. **El lenguaje maquina:** Este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de 0 y 1 para formar las ordenes entendibles por el hardware de la máquina. Este lenguaje es mucho más rápido que los lenguajes de alto nivel. La desventaja es que son bastantes difíciles de manejar y usar, además de tener códigos fuente enormes donde encontrar un fallo es casi imposible.
2. **El lenguaje ensamblador:** Es un derivado del lenguaje máquina y está formado por abreviaturas de letras y números llamadas mnemotécnicos.   
   Con la aparición de este lenguaje se crearon los programas traductores para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Como ventaja con respecto al código máquina es que los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria. Las desventajas de este lenguaje siguen siendo prácticamente las mismas que las del lenguaje ensamblador, añadiendo la dificultad de tener que aprender un nuevo lenguaje difícil de probar y mantener.
3. **¿Qué es un lenguaje de alto nivel?**

Un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) de alto nivel se caracteriza por expresar los [algoritmos](https://es.wikipedia.org/wiki/Algoritmo) de una manera adecuada para la comprensión humana. En lugar de tratar con registros, direcciones de memoria y las pilas de llamadas, los lenguajes de alto nivel se refieren a las variables, matrices, objetos, aritmética compleja o expresiones booleanas, subrutinas y funciones, bucles, hilos, cierres y otros conceptos de informática abstracta, con un enfoque en la facilidad de uso sobre la eficiencia óptima del programa.

**Existen tres modos de ejecución de estos lenguajes:**

- [Interpretado](http://www.alegsa.com.ar/Dic/interprete.php):

Cuando un código en un lenguaje es interpretado, su sintaxis es leída y luego ejecutada directamente, sin fase de compilación. Un programa llamado intérprete lee cada instrucción del programa, siguiendo el flujo de programa, y decide qué hacer y lo hace. Los intérpretes son la forma más simple de implementación del comportamiento de un lenguaje, comparado con las otras dos variantes listadas aquí.

- [Compilado](http://www.alegsa.com.ar/Dic/compilador.php):

Cuando un código en un lenguaje es compilado, su sintaxis es primero transformada a una forma ejecutable antes de correrlo. Existen dos tipos de compilación: generación de código máquina (compilador que pasa del código fuente directamente al código máquina) y representación intermedia (se compila una representación que puede ser optimizada o guardada para ejecutar luego sin la necesidad de releer el código fuente. Cuando esta representación se guardada puede ser en forma de [bytecode](http://www.alegsa.com.ar/Dic/bytecode.php)).

- Traducción fuente a fuente (Source-to-Source Translated o Trans-compiling):

El código escrito en un lenguaje puede ser traducido a términos de un lenguajes de programación para los cuales los compiladores de código nativo ya están ampliamente disponibles. Los lenguajes JavaScript y C son usuales para estas traducciones. Por ejemplo, CoffeeScript, Chicken Scheme y Eiffel.

Algunos ejemplos de lenguajes de programación de alto nivel:

- [Ada](http://www.alegsa.com.ar/Dic/ada.php)

- [Algol](http://www.alegsa.com.ar/Dic/algol.php)

- [BASIC](http://www.alegsa.com.ar/Dic/basic.php)

- [COBO](http://www.alegsa.com.ar/Dic/cobol.php)L

- [Delphi](http://www.alegsa.com.ar/Dic/delphi.php)

- [FORTRAN](http://www.alegsa.com.ar/Dic/fortran.php)

- [Java](http://www.alegsa.com.ar/Dic/java.php)

- [LISP](http://www.alegsa.com.ar/Dic/lisp.php)

- [Pascal](http://www.alegsa.com.ar/Dic/pascal.php)

- [Perl](http://www.alegsa.com.ar/Dic/perl.php)

- [php](http://www.alegsa.com.ar/Dic/php.php)

- [Prolog](http://www.alegsa.com.ar/Dic/prolog.php)

- [Python](http://www.alegsa.com.ar/Dic/python.php)

- [Visual Basic .NET](http://www.alegsa.com.ar/Dic/visual%20basic%20.net.php)

1. **¿Qué es un Intérprete?**Un **intérprete** es un programa que ejecuta línea a línea las instrucciones de un programa de alto nivel. El intérprete carga el código fuente y traduce las instrucciones a un lenguaje intermedio que puede luego ser ejecutado.  
   Los intérpretes generan un código binario que se interpreta cada vez que se ejecuta el programa a diferencia del compilador que crea un archivo ejecutable. Ejemplos de programas que utilizan intérpretes son BASIC, y Visual Basic. En VB puedes compilar todo el programa para crear un ejecutable o puedes ejecutar un formulario o módulo a la vez, en cuyo caso interviene un intérprete.  
   Los lenguajes interpretados son más fáciles de aprender que los lenguajes compilados. Un intérprete le permite al programador saber de inmediato cuándo y dónde existen problemas en el código mientras que los programas compilados hacer esperar al programador hasta que el programa se ha completado.  
     
   **Las ventajas de utilizar un intérprete son:**

* Al ejecutarse línea a línea, resulta ser más fácil de depurar y comprobar errores.
* Si el programa es muy largo, un intérprete resulta es más rápido que un compilador porque no tiene que traducir todo el programa para ejecutarlo.

**Las desventajas de utilizar un intérprete son:**

* El código fuente se requiere para poder ejecutar un programa
* Los intérpretes son más lentos que los programas compilados o ejecutables.

1. **¿Qué es un compilador?**Un compilador es un programa especial que procesa las instrucciones escritas en un lenguaje de programación de alto nivel y las convierte a lenguaje de máquina. Un programa que traduce a partir de un lenguaje de bajo nivel a un lenguaje de nivel superior se conoce como descompilador.  
   Los compiladores son programas muy grandes que permiten la comprobación de errores. Algunos compiladores pueden traducir lenguaje de alto nivel a un lenguaje ensamblador intermedio, que luego se traduce a código de máquina por un programa ensamblador. Otros compiladores generan código de máquina directamente.  
   Todo lenguaje de alto nivel tiene un compilador incorporado. Básicamente, el compilador es la parte central de dicho lenguaje, porque define y traduce las instrucciones recibidas.  
     
   **Las ventajas de utilizar un compilador son:**

* El programa ejecutable no incluye el código fuente, por lo tanto, el código compilado es más seguro.
* Un programa compilado se ejecutan más rápidamente que un programa que está siendo interpretado al momento de ejecutarse

**Las desventajas de utilizar un compilador son:**

* El código objeto necesita crearse para obtener el archivo ejecutable final, si el programa es muy largo, el proceso de compilación puede ser muy lento.
* El código fuente debe estar libre de errores para que el archivo ejecutable se genere.

1. [↑](#footnote-ref-0)