Estructura de Datos

Trabajo Práctico Grupal

Profesor: Martín Santoro

Comisión: “B”

Integrantes:

-Marcela Orozco Somoza

-Manuel Navarro

-Matías Armándola

-Raúl Martínez

-Facundo Nahuel Rojas

1. ¿Qué es un IDE? Cite ejemplos

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word. Los IDEs ofrecen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc.

Ejemplos:

Eclipse: Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar proyectos.

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.

Geany: Es un IDE que hasta hace poco 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.

CodeRun: Es un IDE que te permitirá programar en línea varios lenguajes, entre ellos PHP, Ajax, C#, CSS, JavaScript y HTML. Funciona perfectamente, aunque está en inglés, es útil para quién no disponga de un buen editor a mano.

2. ¿Qué es programación Estructurada?

Es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa.

Esta forma de programar (paradigma) se basa en un famoso teorema, desarrollado por Edsger Dikjstra, que demuestra que todo programa puede escribirse utilizando únicamente las tres estructuras básicas de control:

• Secuencia: el bloque secuencial de instrucciones, ejecutadas sucesivamente, una detrás de otra.

• Selección: la instrucción condicional con doble alternativa, de la forma “if condición then instrucción-1 else instrucción 2”.

• Iteración: el bucle condicional “while condición do instrucción”, que ejecuta la instrucción repetidamente mientras la condición se cumpla.

​Este paradigma llevado a un enfoque moderno, agrega un punto muy importante: segmentación. Generando las siguientes ventajas:

• El coste de resolver varios subproblemas de forma aislada es con frecuencia menor que el abordar el problema global.

• Facilita el trabajo simultáneo en paralelo de distintos grupos de programadores.

• Posibilita en mayor grado la reutilización de código en futuras aplicaciones.

3. ¿Qué es programación orientada a Objetos?

La programación Orientada a objetos se define como un paradigma de la programación, una manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Al programar orientado a objetos tenemos que aprender a pensar cómo resolver los problemas de una manera distinta a como se realizaba anteriormente, en la programación estructurada. Ahora tendremos que escribir nuestros programas en términos de clases, objetos, propiedades y métodos

4. ¿Qué es un lenguaje de bajo nivel?

Es aquel en el que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan. El uso de la palabra bajo en su denominación no implica que el lenguaje sea menos potente que un lenguaje de alto nivel, sino que se refiere a la reducida abstracción entre el lenguaje y el hardware. Por ejemplo, este tipo de lenguajes se utiliza para programar tareas críticas de los sistemas operativos, de aplicaciones en tiempo real o controladores de dispositivos.

5. ¿Qué es un lenguaje de alto nivel?

Se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas. Estos lenguajes permiten una máxima flexibilidad al programador a la hora de abstraerse o de ser literal. Permiten un camino bidireccional entre el lenguaje máquina y una expresión casi oral entre la escritura del programa y su posterior compilación. Por lo general suelen estar orientados a objetos, a eventos o a funciones, pudiendo estos combinarse. Asimismo, pueden ser compilados o interpretados. Algunos ejemplos son: Java, PHP, Python, Javascript, C++.

6. ¿Qué es un intérprete?

Un intérprete es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así hasta completar el programa origen.Los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción

7. ¿Qué es un compilador?

Un compilador es un programa que convierte o traduce el código fuente de un programa hecho en lenguaje de alto nivel, a un lenguaje de bajo nivel (lenguaje de máquina).

Básicamente, un compilador cuenta con dos partes fundamentales: El Front End y el Back End. Estas partes se complementan para lograr el objetivo del compilador.

Front End: El Front End es la parte del compilador que interactúa con el usuario y por lo general, es independiente de la plataforma en la que se trabaja. Se encarga de realizar el análisis del código fuente a compilar y verifica que este sea válido, así como también, genera el árbol de derivación y rellena los valores de la tabla de símbolos.

Back End: Esta parte del compilador es la encargada de generar el código en formato de máquina, a partir del trabajo hecho por el Front End.