ITEDES

Educación Digital

Módulo:

Fundamentos de Ingeniería de Software

Segmento:

Elementos Informáticos

Tema:

Introducción a las Comunicaciones

Prof. Germán C. Basisty
german.basisty@itedes.com

Índice de Contenidos

Herramientas de Desarrollo

IDEs y Editores de Texto

En el mundo del desarrollo de software, la herramienta principal del programador es el **editor de texto**. Estos, a diferencia de sus pares de oficina, no estan pensados para mejorar la estética del contenido en función de una carta, curruculum, informe, etc, sino que estan orientados a hacer más sencillo el proceso de escritura de código. Dentro de sus características fundamentales, podemos citar:

- Coloreo contextual del código
- Sugerencias del lenguaje en el que se esté desarrollando
- Potentes motores de búsqueda y reemplazo de cadenas de carácteres.
- Etc.

Algunos ejemplos (los que vamos a utilizar) son:

- Microsoft Visual Studio Code
- Atom
- Kate
- VIm (linea de comandos)

Por lo general, los editores de texto son multilenguaje y sus funcionalidades pueden extenderse a través de plugins.

Los **IDEs** (del inglés, *Integrated Development Environment*) son herramientas más sofisticadas orientadas a un único lenguaje / tecnología, y además de un editor de textos potente y específico, con todas las funcionalidades arriba mencionadas, suelen incorporar además herramientas de generación de código y compilación.

Algunos ejemplos son:

- Eclipse (Java)
- NetBeans (Java)
- Microsoft Visual Studio (Suite de Desarrollo Microsoft)
- Code::Blocks (C++)
- Android Studio (Java para Android)

Tipos de Lenguaje de Programación

Existen lenguajes de programación

- **Compilados**: El código (texto) que escribe el desarrollador se traduce a lenguaje de máquina mediante un proceso llamado Compilación. Ejemplos: C, C++, Go, etc. Son muy eficientes ya que son ejecutados directamente por el sistema operativo y están muy ligados a la plataforma para la que han sido desarrollados.

- **Compilados a máquina virtual**: El código que escribe el desarrollador es traducido a algo parecido al lenguaje de máquina, pero en vez de ser ejecutado por el propio sistema operativo, se ejecuta por una máquina virtual (o entorno de ejecución), y son bastante eficientes (aunque no tanto como los compilados). Ejemplos: Java, dotNet Core. Son multiplataforma. Esto quiere decir que los programas se ejecutarán en aquellas plataformas para las cuales esté disponible el entorno de ejecución del lenguaje.

- **Interpretados**: El código que escribe el desarrollador se va traduciendo a lenguaje de máquina línea por línea a medida que el programa se va ejecutando. Para poder funcionar, la plataforma necesita tener instalado el intérprete del lenguaje en cuestión. Su desempeño es inferior a los dos anteriores, aunque son completamente multiplataforma. Ejemplos: python, perl, JavaScript (en el navegador, o en el servidor via node.js).

Compiladores, Depuradores e Intérpretes

- Los **compiladores** son herramientas que traducen el código fuente de un programa a lenguaje de máquina o a lenguaje de entorno de ejecucion (para el caso de los lenguajes compilados a máquina virtual).

- Los **depuradores** son herramientas que nos permiten hacer un seguimiento de los programas a medida que se van ejecutando y son sumamente útiles a la hora de encontrar errores.

- Los **intérpretes** son herramientas que permiten ejecutar un programa del tipo interpretado.

Herramientas de Control de Versiones

El código fuente va cambiando permanentemente a medida que se va desarrollando una aplicación. Por lo tanto, es necesario darle seguimiento a dicha evolución para poder gestionar los cambios apropiadamente.

Actualmente, de forma predominante se utilizan 2 tecnologías:

- GIT
- SVN

Comparativa de soluciones:

	SVN	GIT
Control de versiones	Centralizada	Distribuida
Repositorio	Un repositorio central donde se generan copias de trabajo	Copias locales del repositorio en las que se trabaja directamente
Autorización de acceso	Dependiendo de la ruta de acceso	Para la totalidad del directorio
Control de cambios	Basado en archivos	Basado en contenido
Historial de cambios	Solo en el repositorio completo, las copias de trabajo incluyen únicamente la versión más reciente	Tanto el repositorio como las copias de trabajo individuales incluyen el historial completo
Conectividad de red	Con cada acceso	Solo necesario para la sincronización

Criterio de elección de GIT

- No se desea depender de una conexión de red permanente, pues se puede trabajar en el proyecto desde cualquier lugar.
- Seguridad en caso de fallo o pérdida de los repositorios principales.
- No se necesita contar con permisos especiales de lectura y escritura para los diferentes directorios (aunque, de ser así, será posible y complejo implementarlo).
- La transmisión rápida de los cambios es una de las prioridades.

SVN será la opción indicada si

- Se necesita otorgar permisos de acceso basados en rutas para las diferentes áreas del proyecto.
- Se desea agrupar todo el trabajo en un solo lugar.
- Se trabaja con numerosos archivos binarios de gran tamaño.
- También se quiere guardar la estructura de los directorios vacíos (estos son rechazados por GIT, debido a que no contienen ningún tipo de contenido).

Teoría de la Programación

Que es un algoritmo

En matemáticas, lógica, ciencias de la computación y disciplinas relacionadas, un algoritmo es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.

Que es un programa

Un programa informático o programa de computadora es una secuencia de instrucciones, escritas para realizar una tarea específica en una computadora. Es la expresión de uno o varios algoritmos en un lenguaje formal.

Que es programar

La programación de computadoras es el proceso iterativo de escribir o editar código fuente. Dicha edición implica probar, analizar y perfeccionar, y, a veces, coordinar con otros programadores, en el caso de un programa desarrollado en conjunto.

Que es un paradigma de programación

Un paradigma de programación es una propuesta tecnológica adoptada por una comunidad de programadores y desarrolladores que representa un enfoque particular o filosofía para diseñar soluciones. Los paradigmas difieren unos de otros, en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran su solución del problema, en otras palabras, el cómputo.

Ejemplos de Paradigmas de Programación:

- **Programación Monolítica o Top-Down**: El paradigma más antiguo de todos. Consiste en un bloque de código que va ejecutándose de arriba hacia abajo, utilizando saltos entre líneas para desviar el flujo del programa. El código desarrollado de esta forma es poco reutilizable y difícil de mantener.

- **Programación Estructurada**: Más moderno que el anterior. Divide el problema total en subproblemas más acotados y fáciles de manejar. Incorpora el concepto de función, que es un bloque de código que recibe parámetros, procesa y devuelve un resultado. El control del flujo del programa se transfiere entre las diversas funciones que componen el programa para ir solucionando el problema computacional por parte. Existe cierta capacidad de reutilización de código a través de las bibliotecas de funciones.

- **Programación Orientada a Objetos**: Es el paradigma de programación actualmente más utilizado, en donde se entiende la representación de los problemas del mundo real en el medio computacional a través de objetos que tienen propiedades y acciones (métodos), y las interacciones que se producen entre estos objetos.

Componentes de un programa

- **Variables**: Una variable es una porción de la memoria que se reserva durante la ejecución de un programa para guardar un dato o valor que puede ir modificándose a medida que el programa fluye. Las variables tienen 2 características principales: nombre y tipo de dato. Toda variable debe tener un nombre, que debe estar compuesto por letras y números, y, dependiendo del lenguaje, algunos símbolos (como _ por ejemplo). Los tipos de datos posibles también dependen del lenguaje. En pseudocódigo son válidos: enteros, decimales, texto y booleanos.

- **Constantes**: Las constantes son representaciones de datos o valores que no van a cambiar durante la ejecución de un programa, y se utilizan para mejorar la legibilidad del código fuente.

- **Vectores**: Son variables que se encuentran dispuestas de forma adyacente en la memoria del ordenador, y tienen nombre y tipo de dato en común. Los componentes de un vector se diferencian entre sí mediante un índice. Los vectores tienen dimensiones y cada dimensión tiene una magnitud. Este tema se desarrollará en profundidad más adelante.

- **Estructuras de programación**: Son las estructuras que hacen al flujo del programa. Las hay de decisión (simple, doble, múltiple) y de repetición (controladas por contador, controladas por centinela).

Operaciones con variables

Dependiendo del tipo de dato existen diversas operaciones que pueden realizarse, a saber:

- Universal
  - Asignación. Se utiliza el símbolo “=”.

- Numéricas
  - Suma. Se utiliza el símbolo “+”.
  - Resta. Se utiliza el símbolo “-”.
  - Multiplicación. Se utiliza el símbolo “*”.
  - División. Se utiliza el símbolo “/”.
  - Módulo. Se utiliza el símbolo “%”.

- Texto
  - Concatenación. Se utiliza el símbolo “+” (dependiendo del lenguaje).

- Booleanos
  - Igualdad. Se utiliza el símbolo “==” (dependiendo del lenguaje).
  - Distinto. Se utiliza el símbolo “!=” (dependiendo del lenguaje).
  - Mayor. Se utiliza el símbolo “>”.
  - Mayor o igual. Se utiliza el símbolo “>=”.
  - Menor. Se utiliza el símbolo “<”.
  - Menor o igual. Se utiliza el símbolo “<=”.

- Además, existen los conectores lógicos AND “&&” y OR “||” para unir dos o más expresiones.

NOTA:

Python es un lenguaje de programación interpretado, débilmente tipado, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Bash es un intérprete de comandos del entorno **UNIX** que permite el desarrollo de scripts (pequeños programas orientados principalmente a la automatización de tareas), y será utilizado durante esta materia, junto con Python, para realizar prácticas de programación.

Java es un lenguaje de programación compilado a máquina virtual y fuertemente tipado, cuyo paradigma de programación es el orientado a objetos puro.

Ejemplos

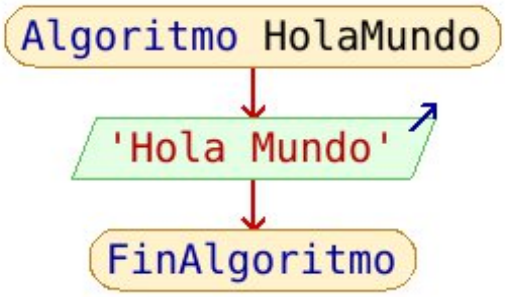
Ejemplo 1

Desarrollar un programa que imprima por pantalla “Hola Mundo”

Pseudocódigo

```
Algoritmo HolaMundo
        Escribir 'Hola Mundo'
FinAlgoritmo
```

Diagrama de Flujo



BASH

```
#!/bin/bash

echo “Hola Mundo”
exit 0
```

Python

```
print(“Hola Mundo!”)
```

Java

```
public class HolaMundo {
        public static void main(String args[]) {
                System.out.println("Hola Mundo");
        }
}
```

Ejercicio 2

Desarrollar un programa que le pregunte el nombre y la edad al usuario, y luego lo salude.

Pseudocódigo

```
Algoritmo Saludador
	Escribir "Ingrese su nombre: "
	Leer nombre
	Escribir "Ingrese su edad: "
	Leer edad
	Escribir "Hola ", nombre, " Usted tiene ", edad, " años"
FinAlgoritmo
```

Diagrama de flujo

BASH

```
#!/bin/bash

declare name=””
read -p “Ingrese su nombre: “ name

declare age
read -p “Ingrese su edad: “ age

echo “Hola $name Usted tiene $age años.”

exit 0
```

Python

```
name = input(“Ingrese su nombre: “)
age = int(input(“Ingrese su edad: “))

print(“Hola “ + name + “! Usted tiene “ + str(age) + “ años”)
```

Java

```
import java.util.Scanner;

public class Ejercicio2 {
	public static void main(String args[]) {
		Scanner keyboard = new Scanner(System.in);

		System.out.print("Ingrese su Nombre: ");
		String name = keyboard.next();

		System.out.print("Ingrese Su edad: ");
		Integer age = keyboard.nextInt();
	}
}
```

Ejemplo 3

Desarrollar un software que dados 2 números determine el resultado de la suma.

Pseudocódigo

```
Algoritmo Sumador
        Escribir "Ingrese un número:"
        Leer numero1

        Escribir "Ingrese otro número:"
        Leer numero2

        resultado = numero1 + numero2

        Escribir "El resultado de la suma es: ", resultado
FinAlgoritmo
```

Diagrama de Flujo

BASH

```
#!/bin/bash

declare -i number1=0
read -p “Ingrese un número: “ number1

declare -i number2=0
read -p “ingrese otro número: “ number2

declare -i result=$((number1 + number2))

echo “El resultado de la suma es: $result”

exit 0
```

Python

```
number1 = int(input(“Ingrese un número: “))
number2 = int(input(“Ingrese otro número: “))

result = number1 + number2

print(“El resultado de la suma es: “, result)
```

Java

```
import java.util.Scanner;

public class Ejercicio3 {
	public static void main(String args[]) {
		Scanner keyboard = new Scanner(System.in);

		System.out.print("Ingrese un Número: ");
		Integer number1 = keyboard.nextInt();

		System.out.print("Ingrese otro Número: ");
		Integer number2 = keyboard.nextInt();

		Integer result = number1 + number2;
		System.out.println("El resultado de la suma es: " + result.toString());
	}
}
```

Tips de desarrollo

El código se escribe una vez, y se lee cien veces. Por eso debemos escribir código exquisito, para que sea perfectamente legible y entendible para quien lo lea. Esto se conoce como código limpio.

Respecto a las variables:

- Las variables deben tener nombres que representen su contenido.
- No utilizar variables de una sola letra (a excepción de i, j y k en casos puntuales que estudiaremos más adelante).
- Los nombres de las variables deben ser pronunciables.
- Los nombres de las variables deben estar en inglés (en pseudocódigo no es necesario).
- Los nombres de las variables siempre se escriben en minúscula.
- En el caso de necesitar utilizar 2 o más palabras para nombrar una variable, distinguirlas utilizando CAMEL CASE, esto es escribir en mayúscula la primera letra a partir de la segunda palabra, por ejemplo firstNumber, sureName, currentDate, etc.
- Si bien en BASH no es obligatorio declarar las variables (se explicará más adelante), es una buena práctica y conviene hacerlo, utilizando el comando declare con los modificadores -i para enteros y -a para vectores.
- Inicialice siempre las variables, utilizando algún valor nulo o etéreo para evitar errores en tiempo de ejecición.

Respecto a los bloques de código:

- Agrupar visualmente aquellas líneas de código que tengan relación entre sí. Separarlas utilizando líneas vacías.
- Agrupar los bloques según su función, por ejemplo, entrada, proceso y salida.
- Indentar adecuadamente los bloques de código en función de su anidación, para que la lectura resulte más comprensible.

Ejercitación

1) Desarrollar un software que le solicite al usuario que ingrese el nombre de su barrio, la dirección postal de la comisaría más cercana, y las entrecalles. Luego mostrar un mensaje por pantalla similar a “La comisaría de Banfield se encuentra en la calle Maipú 133 entre Alsina y Belgrano”. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

2) Desarrollar un software que le solicite al usuario sus datos personales, a saber: DNI, nombre y apellido, dirección y teléfono. Mostrar los datos por pantalla. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

3) Desarrollar un software que le solicite al usuario el ingreso de 5 números, y mostrarlos en el orden inverso al ingresado. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

4) Desarrollar un software que le solicite al usuario el ingreso de 2 números y le muestre al usuario el resultado de la suma de esos números. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

5) Desarrollar un software que le solicite al usuario el ingreso de 2 números y le muestre al usuario el resultado de la resta de esos números. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

6) Desarrollar un software que le solicite al usuario el ingreso de 2 números y le muestre al usuario la multiplicación de esos números. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.

7) Desarrollar un software que le solicite al usuario el ingreso de 2 números y le muestre al usuario el resultado de la división de esos números. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Java y Python.