Universidad Hispanoamericana



INGENIERÍA EN SISTEMAS

PROGRAMACION II

**TAREA# 1**

ING. JORGE DAVIÁN AGUILAR OVIEDO, LIC

**ESTUDIANTE**

María Eduarte Núñez.

II Cuatrimestre, 2016

**Diferencias entre el lenguaje C, C++ y C#**

**C**

* Cuando una función no toma parámetros tienes que añadir el voit.
* En C se tienen que declarar las variables al principio del código.
* No es un leguaje orientado a objetos, solo implementa procedimientos y funciones.
* No es necesario utilizar prototipos de las funciones al migrar código.
* No admite declaración de variables locales.
* Utiliza bibliotecas (Run-Time).

**C++**

* C++ nació como una evolución de C.
* Implementa el paradigma de programación orientada a objetos.
* El mantenimiento de versiones es más sencillo.
* Utiliza punteros.
* Admite la declaración de variables locales dentro de sentencias.
* Cuando una función no toma parámetros es opcional añadir el voit.

**C#**

* No se utilizan punteros. En su lugar se utilizan referencias a objetos.
* No se utilizan los operadores -> ni ::. El único operador de acceso a métodos de objetos que se utiliza es el operador punto ‘.’.
* C# no admite campos de bits.
* No existe una librería de tiempo de ejecución (Runtime).
* No se permite herencia múltiple en clases pero sí en interfaces.
* Es una implementación propia de Microsoft.

**Git**

Es un software de control de versiones diseñado por Linus Torvalds. Fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida.

Algunas de las características más importantes de Git son:

* Rapidez en la gestión de ramas, debido a que indica cuando un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
* Gestión distribuida: Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
* Gestión eficiente de proyectos grandes.
* Realmacenamiento periódico en paquetes.

**Uso de Git**

Se trabaja en un directorio donde se encuentran los archivos, que se modifican, borran, crean nuevos, entre otros.

Ese directorio es lo que se llama "Directorio de trabajo", puede contener otros directorios, de hecho es el que contiene el directorio .git .

Se crea un repositorio el cual se controla y se modifica de acuerdo a la necesidad de lo que se está requiriendo y de acuerdo a la manipulación de los comandos para controlar y darle mantenimiento de los repositorios.

**Principales comandos**

**Buscando Ayuda:**

1. **git help** *comando* ó **git** *comando* --help

    Muestra la ayuda para ese comando

b) **Creación de un repositorio:**

2. **git init**

    Crea un repositorio en el directorio actual

3. **git clone** *url*

    Clona un repositorio remoto dentro de un directorio

c) **Operaciones sobre Archivos:**

4. **git add** *path*

    Adiciona un archivo o un directorio de manera recursiva

5. **git rm** *ruta*

    Remueve un archivo o directorio del árbol de trabajo

      -f : Fuerza la eliminación de un archivo del repositorio

6. **git mv** *origen* *destino*

    Mueve el archivo o directorio a una nueva ruta

      -f : Sobre-escribe los archivos existentes en la ruta destino

7. **git checkout** *[rev]* *archivo*

    Recupera un archivo desde la rama o revisión actual

      -f : Sobre-escribe los cambios locales no guardados

d) **Trabajando sobre el código:**

8. **git status**

    Imprime un reporte del estado actual del árbol de trabajo local

9. **git diff** *[ruta]*

    Muestra la diferencia entre los cambios en el árbol de trabajo local

10. **git diff** HEAD *ruta*

    Muestra las diferencias entre los cambios registrados y los no registrados

11. **git add** *path*

    Selecciona el archivo para que sea incluido en el próximo commit

12. **git reset** HEAD *ruta*

    Marca el archivo para que no sea incluido en el próximo commit

13. **git commit**

    Realiza el commit de los archivos que han sido registrados (con git-add)

      -a : Automáticamente registra todos los archivos modificados

14. **git reset** --soft HEAD^

    Deshace commit & conserva los cambios en el árbol de trabajo local

15. **git reset** --hard HEAD^

    Restablece el árbol de trabajo local a la versión del último commit

16. **git clean**

    Elimina archivos desconocidos del árbol de trabajo local

e) **Examinando el histórico:**

17. **git log** *[ruta]*

    Muestra el log del commit, opcionalmente de la ruta especifica

18. **git log** *[desde [..hasta]]*

    Muestra el log del commit para un rango de revisiones dado

      --stat : Lista el reporte de diferencias de cada revisión

      -S'pattern' : Busca el historial de cambios que concuerden con el patrón de búsqueda

19. **git blame** *[archivo]*

    Muestra el archivo relacionado con las modificaciones realizadas

f) **Repositorios remotos:**

20. **git fetch** *[remote]*

    Trae los cambios desde un repositorio remoto

21. **git pull** *[remote]*

    Descarga y guarda los cambios realizados desde un repositorio remoto

22. **git push** *[remote]*

    Guarda los cambios en un repositorio remoto

23. **git remote**

    Lista los repositorios remotos

24. **git remote add** *remote* *url*

    Añade un repositorio remoto a la lista de repositorios registrados

g) **Ramas:**

25. **git checkout** *rama*

    Cambia el árbol de trabajo local a la rama indicada

      -b *rama* : Crea la rama antes de cambiar el árbol de trabajo local a dicha rama

26. **git branch**

    Lista las ramas locales

27. **git branch** -f *rama* *rev*

    Sobre-escribe la rama existente y comienza desde la revisión

28. **git merge** *rama*

    Guarda los cambios desde la rama

h) **Exportando e importando:**

29. **git apply** - < *archivo*

    Aplica el parche desde consola (stdin)

30. **git format-patch** *desde [..hasta]*

    Formatea un parche con un mensaje de log y un reporte de diferencias (diffstat)

31. **git archive** *rev* > *archivo*

    Exporta resumen de la revisión (snapshot) a un archivo

      --prefix=*dir*/ : Anida todos los archivos del snapshot en el directorio

      --format=*[tar|zip]* : Especifica el formato de archivo a utilizar: *tar* or *zip.*

**Github**

Es un servicio de alojamiento de repositorios de software que ofrece opciones muy interesantes para trabajar en equipo. GitHub está basado en Git, un sistema de control de versiones muy popular entre los desarrolladores.

**Su Uso**

GitHub aloja tu repositorio de código y brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además,  contribuye a mejorar el software de los demás.

**Algunos de sus beneficios es que cuenta con:**

* Una wiki para el mantenimiento de las distintas versiones de las páginas.
* Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
* Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
* Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

**Bibliografía**

**Introducción a Git y Github.** <http://www.desarrolloweb.com/articulos/introduccion-git-github.html>. Consultada (15-05-2016).

**Uso Básico de Git.** <https://github.com/oslugr/curso-git/blob/master/texto/uso_basico.md>**.**

Consultada (15-05-2016).

**Aprendiendo a Usar Github.** <http://conociendogithub.readthedocs.io/en/latest/data/dinamica-de-uso/>**.** Consultada (15-05-2016).

**Introducción al Lenguaje C#.** <http://www.ehu.eus/mrodriguez/archivos/csharppdf/Lenguaje/Introduccion.pdf>. Consultada (15-05-2016).

**Diferencia entre C, C++ y C#.** http://blog.nativehex.com/diferencia-entre-c-cpp-y-cs. Consultada (15-05-2016).