

Profesor:

Davian Aguilar

Alumno:

Marjorie Yosibel Blandón Aguilar

Carrera:

Ingeniería en sistemas

Segundo cuatrimestre

Materia:

Programación II

2016

Diferencias entre C, C# Y C++

* C

1. C es un lenguaje libre estandarizado por ISO MUY PEQUEÑO que admite programación estructurada (la de toda la vida), nada más. Útil en programación de microchips, sistemas operativos, drivers y programación web... Poco más.
2. C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.
3. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.
4. Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel.
5. La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas.

* C++

1. es un lenguaje libre estandarizado por ISO MUY GRANDE, que admite: Programación estructurada (la de toda la vida), la totalidad de la POO (objetos, herencia simple, herencia múltiple, polimorfismo, upcasting, downcasting, RTTI, interfaces, clases abstractas, clases amigas, operadores, sobrecarga... y mil cosas más avanzadas)
2. es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup.
3. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.
4. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.
5. Antes se había usado el nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

* C#

1. C# es un invento de Microsoft (lenguaje propietario) que mezcla las características básicas de C++ (no las avanzadas) simplificándolas al estilo Java y ofreciendo un framework. El problema es que es .Net, y deja de ser código nativo/portable. Eso sí, el framework provee bastante facilidad de programación de tareas comunes, al igual que Java. Por ello se genera el debate ¿Java o C#? Su funcionalidad viene a ser parecida. .Net es más nativo y Java más virtual.
2. es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.
3. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.
4. El nombre C Sharp fue inspirado por la notación musical, donde ♯ (sostenido, en inglés sharp) indica que la nota (C es la nota do en inglés) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' viene de cuatro '+' pegados.
5. Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

GIT

¿Qué es un Git?

Git es un software para el control de versiones desarrollado en un principio por Linus Torvalds(sí, el creador del kernel Linux!) y se ha convertido en una herramienta indispensable para el desarrollo de software, incluido el propio kernel de Linux. Ya había otros programas que más o menos realizaban estas funcionas.

Pero al señor Torvalds no le convencían, así que se aplicó el lema de GNU/Linux. Si no existe siempre lo puedes crear. Y se puso manos a la obra y desarrolló este software. Y como es “lema de la casa” del creador del kernel Linux, es software abierto y colaborativo, por lo que el propio software de Git es desarrollado y mantenido por la comunidad en todo el mundo.

Herramientas o comandos de Git

Comandos Básicos de Git

a) 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 ultimo 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

i) Etiquetas:

32. git tag name [revision]

    Crea una etiqueta para la revisión referida

      -s : Firma la etiqueta con su llave privada usando GPG

      -l [patrón] : Imprime etiquetas y opcionalmente los registros que concuerden con el patrón de búsqueda

j) Banderas de Estado de los Archivos:

M (modified) : El archivo ha sido modificado

C (copy-edit) : El archivo ha sido copiado y modificado

R (rename-edit) : El archivo ha sido renombrado y modificado

A (added) : El archivo ha sido añadido

D (deleted) : El archivo ha sido eliminado

U (unmerged) : El archivo presenta conflictos después de ser guardado en el servidor (merge)

GUIHUB

¿Que es GitHub?

Esta web es un lugar donde nos proporcionan espacio para almacenar nuestros repositorios. Es un lugar para alojar nuestros proyectos utilizando el sistema de control de versiones Git, desarrollado por Linus Torvalds.  Así no tienes que preocuparte de gestionar un servidor, etc. Ellos hacen el trabajo lo que resulta muy cómodo. Existen otras opciones como Gitorius, y otras. Pero hablaré de esta que es la que he utilizado.

Para trabajar con repositorios de otros en los que colaborar con tu trabajo. Debes abrirte una cuenta en la página de GitHub. Rellenar un sencillo formulario y un certificado digital para que cuando subas material el servidor te identifique y autentifique que realmente eres quien dices ser

Si vas a trabajar en repositorios de otros, como veremos en este ejemplo, es necesario que el mantenedor o encargado del repositorio te autorice a ti, a tu usuario de GitHub a escribir en el repo, de lo contrario sólo podrás leer, pero no podrías subir tus aportes, así que pídele derechos de lectura/escritura.

Como usar GitHub

Antes de empezar, es importante tener en cuenta conocimiento básico de los términos utilizados en Git: empujar (push), jalar (pull), guardar los cambios (commit), repositorio (repository), etc. También requiere el registro previo en [GitHub](https://github.com/).

¿Cómo crear un repositorio?

Para crear un repositorio en GitHub, solo hay que seleccionar el botón *“Create a New Repo”*, de la barra de herramientas, habiendo entrado a [GitHub](https://www.github.com/) con tu cuenta: Ahora habrá que llenar dos datos:

1. Nombre del repositorio
2. Descripción del repositorio (opcional)

Ademas en github se pueden crear y subir proyectos e incluso colaborar en un proyecto ajeno.