

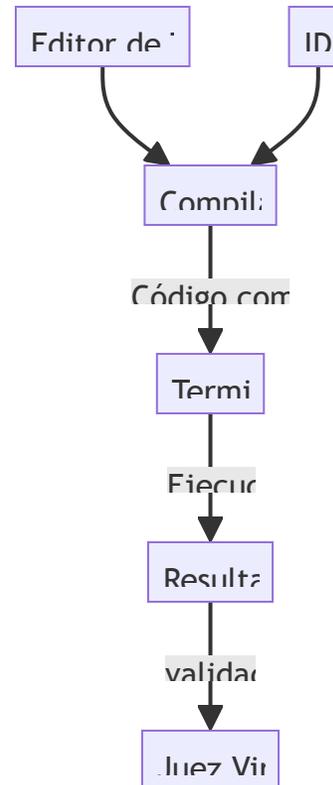
[Γα=Ω5]

# Entorno de Desarrollo

Para la programación competitiva en C++

Por Ariel Parra

# ¿Qué se necesita para la programación competitiva?



# Codeforces

está es la plataforma que estaremos usando para el curso, por lo que ocupamos crear una cuenta a través de <https://codeforces.com/register> ↑ . La razón por la cual usamos esta plataforma es que tiene múltiples problemas de distintas categorías y dificultades, donde al subir cada problema se evaluara con un juez virtual (virtual judge) que juzgara si nuestro código soluciona el problema o en que caso/s no lo cumple.

Para el acompañamiento del curso, se puede recurrir al  Junior Training Sheet V7.0 ↑ , donde hay múltiples problemas ordenados por dificultad con links directos a codeforces.

# IDEs

Un IDE (Integrated Development Environment) es una aplicación que combina múltiples herramientas para el desarrollo de software en una sola interfaz. Generalmente, un IDE cuenta con las siguientes características:

- **Editor de código:** editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico para el lenguaje y la comprobación de errores a medida que se escribe el código.
- **Automatización de compilaciones:** herramientas que automatizan las tareas sencillas y repetitivas como parte de la creación de una compilación local del software para que use el desarrollador, como la compilación del código fuente de la computadora en código binario, el empaquetado de ese código y la ejecución de pruebas automatizadas.
- **Depurador (Debugger):** programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

# DEV-C++



Este es el IDE favorito de los profesores. El principal problema es que la versión original incluye una versión antigua de GCC (4.9.2) aparte solo estar disponible en Windows.

Si se quiere usar este IDE, mi recomendación es usar el DEV-C++ "Rojo" (embarcadero) ↑ ya que este incluye un compilador más moderno (TDM-GCC 9.2.0), pero aun así solo sigue estando disponible en Windows.

```
bool LoadFile(HWND hEdit, LPSTR pszFileName)
{
    HANDLE hFile;
    BOOL bSuccess = FALSE;

    hFile = CreateFile(pszFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
        OPEN_EXISTING, 0, 0);
    if (hFile != INVALID_HANDLE_VALUE)
    {
        DWORD dwFileSize;
        dwFileSize = GetFileSize(hFile, NULL);
        if (dwFileSize != 0xFFFFFFFF)
        {
            LPSTR pszFileText;
            pszFileText = (LPSTR)GlobalAlloc(GPTR, dwFileSize + 1);
            if (pszFileText != NULL)
            {
                DWORD dwRead;
                if (ReadFile(hFile, pszFileText, dwFileSize, &dwRead, NULL))
                {
                    pszFileText[dwFileSize] = 0; // Null terminator
                }
            }
        }
    }
}
```

```
switch(Message) {
    // Upon destruction, tell the main thread to stop */
    case WM_DESTROY: {
        PostQuitMessage(0);
        break;
    }
    // All other messages (a lot of them) are processed using default procedures */
    default:
        return DefWindowProc(hwnd, Message, wParam, lParam);
}
return 0;
}

/* The 'main' function of most GUI programs: this is where execution starts */
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASSEX wc; /* A properties struct of our window */
    MSG msg; /* A 'message', hence the M, or a pointer to our window */
    /* msg: /* A temporary location for all messages */

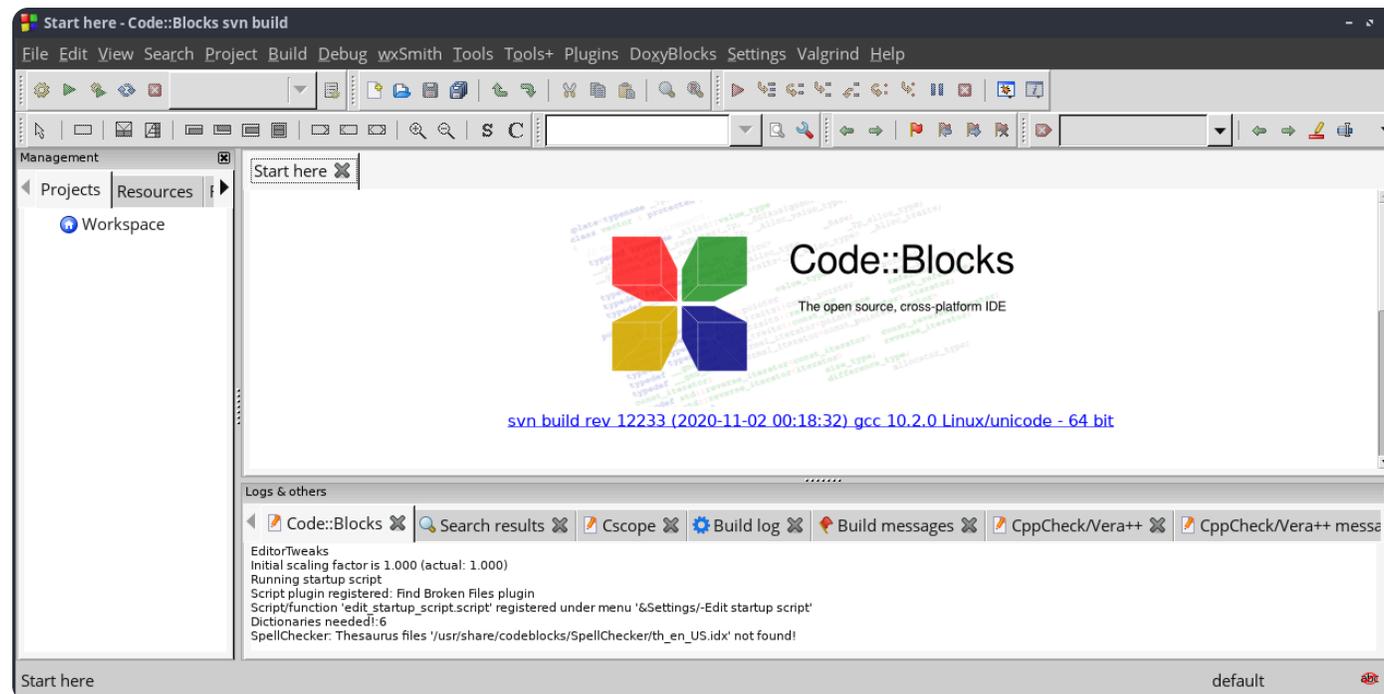
    /* Zero out the struct and set the stuff we want to modify */
    memset(&wc, 0, sizeof(wc));
    wc.cbSize = sizeof(WNDCLASSEX);
    wc.lpfnWndProc = WndProc; /* This is where we will send messages to */
    wc.hInstance = hInstance;
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);

    /* Assign COLOR_WINDOW to put a window for a system color, try Ctrl+Clicking it */
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW);
    wc.lpszClassName = "windowClass";
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION); /* Load a standard icon */
    wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION); /* use the name "M" to use the project icon */
}
```

# Code::Blocks

Este es uno de los IDEs más populares entre las competencias de programación, debido a que es de código abierto, no pesa mucho, incluye compilador para Windows y ser compatible con Linux y MacOS.

💡 Si lo descargas para Windows, descarga la versión que diga 'mingw' para que incluya el compilador



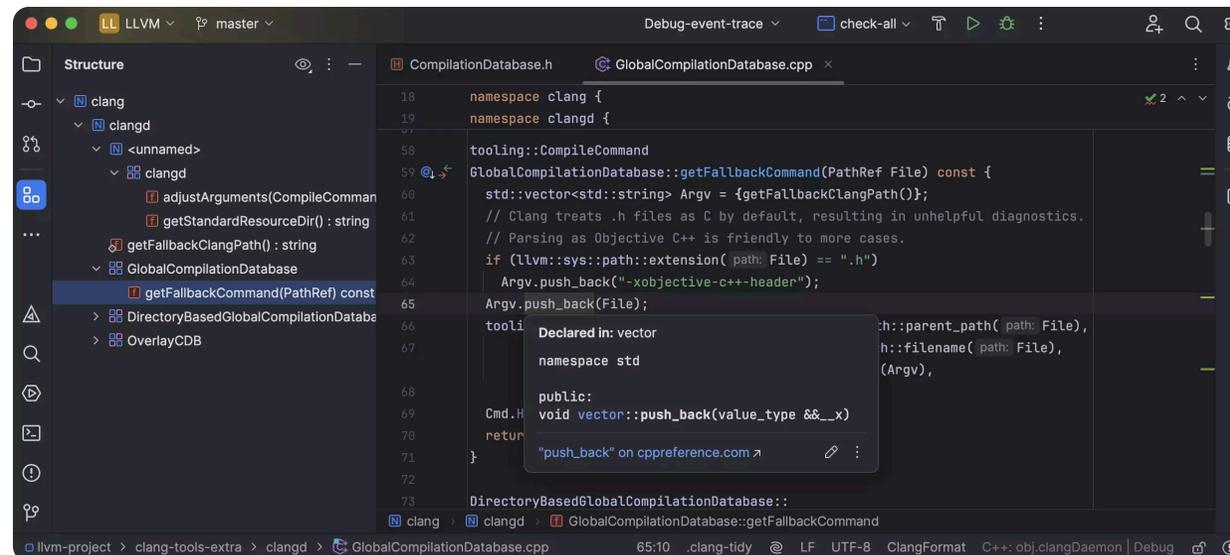
# JetBrains CLion

Este es el IDE de C++ por excelencia en la industria, aunque sea propietario es usado por múltiples empresas como Google, Nasa, BMW, etc.

Este IDE es expansible a través de plugins, todas las herramientas necesarias para desarrollar en C++ y aparte es multiplataforma siendo compatible en Windows, MacOS y Linux.

Las desventajas serian su amplio uso de recursos y su elevado costo anual de más de \$100 USD.

 Pero este es gratis  mientras seas un estudiante y al egresar tendrás un 40% de descuento!



# VScode

No confundir con Visual Studio . VScode es el editor por excelencia del Club, esto debido a ser de código abierto, multiplataforma al estar desarrollado con Electron (básicamente es una app web), tener editor en línea, su implementación con git, y principalmente por su amplio catalogo de extensiones con los cuales podemos desarrollar en cualquier lenguaje y de múltiples maneras. La única desventaja es que no incluye el compilador de C++, por lo que tendremos que instalarlo manualmente.

Para instalarlo podemos hacerlo desde su página web o usando el gestor de paquetes integrado en Windows `winget` a través de una terminal (Powershell ) , con el comando:

```
winget install -e --id Microsoft.VisualStudioCode --override '/SILENT /mergetasks="!runcode,addcontextmenufiles,addcontextmenufolders"'
```

# Extensiones para VScode

-  C/C++ ↗ está es la extensión necesaria para poder correr código de C++ dentro de VScode
-  .run code runner ↗ está es una extensión opcional que acelera el proceso de compilación en VScode
-  Error lens ↗ está extensión nos muestra los errores que va dando el IDE o el compilador.
-  Prettier ↗ está extensión le da formato a nuestro código al presionar la combinación de teclas `shift + alt + F`
-  Competitive Programming Helper ↗ está es la extensión para poder, probar los códigos antes de publicar y también podremos subir directamente los códigos a codeforces de una manera muy rápida

## Temas recomendados:

-  Material Icon Theme ↗,  nord ↗,  catppuccin ↗,  GruvBox ↗,  Monokai Pro ↗ o  dracula ↗

# Instalar el compilador de C++ en Windows

💡 Para otros sistemas o dispositivos, visite el blog: [https://cpc-gallos.github.io/blog/Entorno\\_Desarrollo/](https://cpc-gallos.github.io/blog/Entorno_Desarrollo/) ↗

1. Descargar el gestor de paquetes `msys2` desde su página oficial [www.msys2.org/](http://www.msys2.org/) ↗ o desde el gestor de paquetes integrado en Windows `winget` a través de una terminal (Powershell ) , con el comando:

```
winget install -e --id MSYS2.MSYS2
```

2. Ya instalado, abrimos la terminal de MSYS2 UCRT64  y ejecutamos este comando:

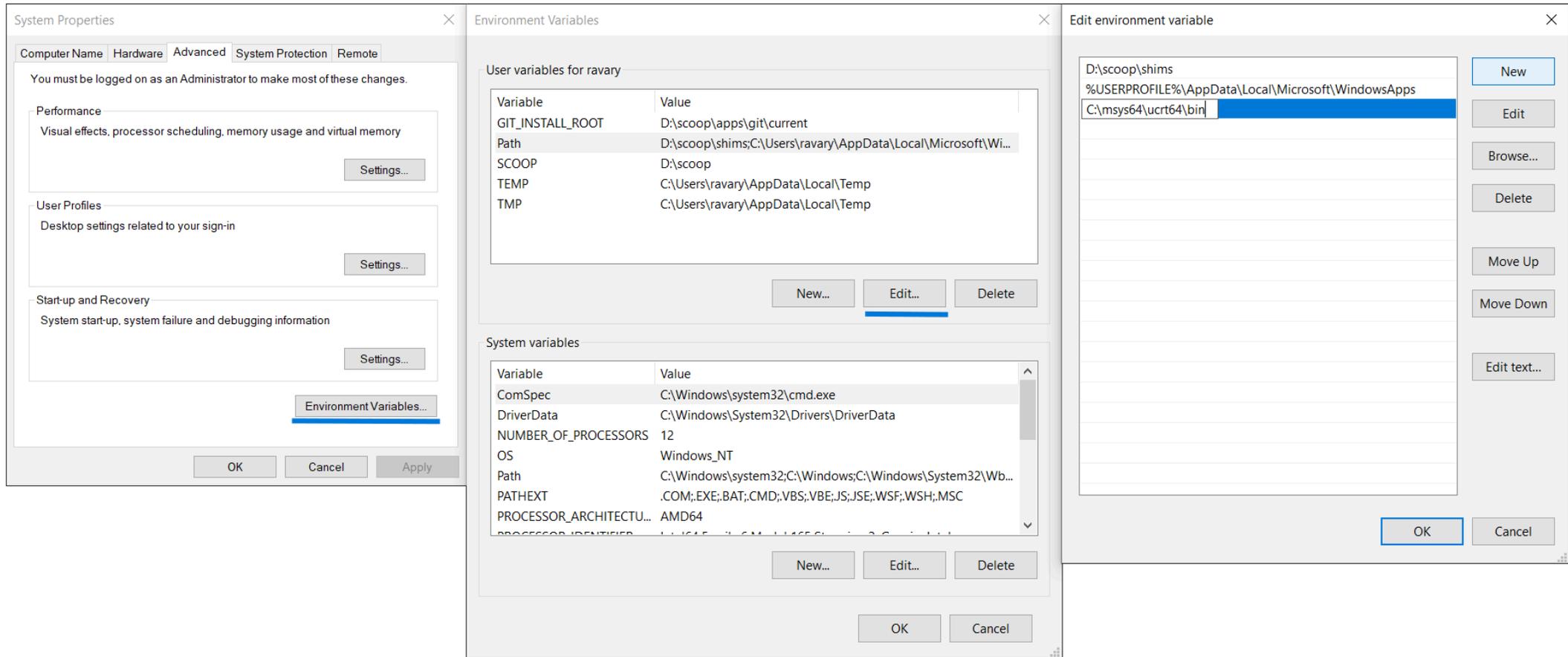
```
pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain
```

Acepta presionando `Enter` 2 veces.

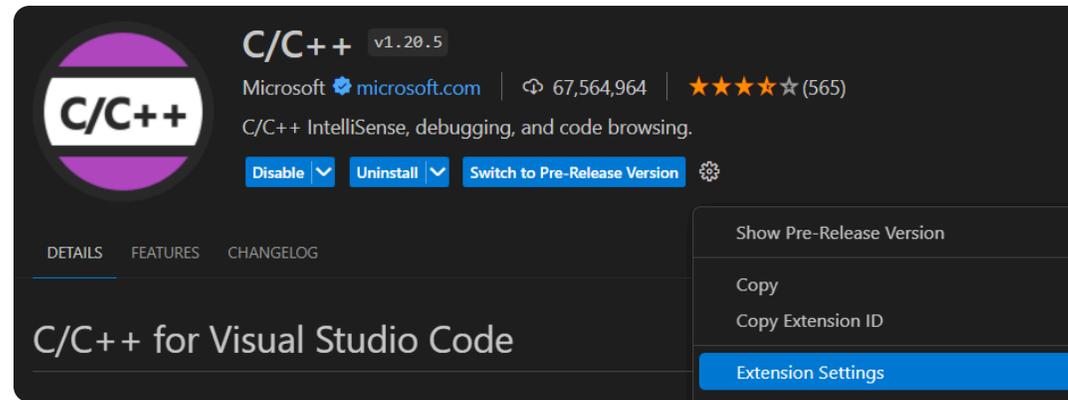
```
$ pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain
warning: base-devel-2022.12-2 is up to date -- skipping
:: There are 19 members in group mingw-w64-ucrt-x86_64-toolchain:
:: Repository ucrt64
 1) mingw-w64-ucrt-x86_64-binutils  2) mingw-w64-ucrt-x86_64-crt-git
 3) mingw-w64-ucrt-x86_64-gcc      4) mingw-w64-ucrt-x86_64-gcc-ada
 5) mingw-w64-ucrt-x86_64-gcc-fortran 6) mingw-w64-ucrt-x86_64-gcc-libgfortran
 7) mingw-w64-ucrt-x86_64-gcc-libs  8) mingw-w64-ucrt-x86_64-gcc-objc
 9) mingw-w64-ucrt-x86_64-gdb     10) mingw-w64-ucrt-x86_64-gdb-multiarch
11) mingw-w64-ucrt-x86_64-headers-git 12) mingw-w64-ucrt-x86_64-libgccjit
13) mingw-w64-ucrt-x86_64-libmangle-git 14) mingw-w64-ucrt-x86_64-libwinpthread-git
15) mingw-w64-ucrt-x86_64-make     16) mingw-w64-ucrt-x86_64-pkgconf
17) mingw-w64-ucrt-x86_64-tools-git 18) mingw-w64-ucrt-x86_64-winpthread-git
19) mingw-w64-ucrt-x86_64-winstorecompat-git

Enter a selection (default=all):
```

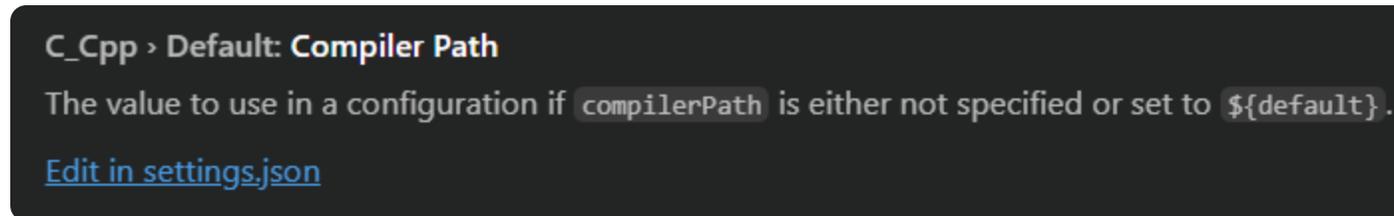
3. Ya terminada la instalación del compilador, procederemos a agregar el compilador al `path`. Para ello, buscamos alguna de estas palabras clave: "variables", "path", "variables del sistema" en el buscador de Windows  y abriremos la aplicación con icono de una computadora con una palomita , allí abriremos las variables y editaremos el path agregando la dirección: `C:\msys64\ucrt64\bin`.



4. ya instalada la extensión del compilador C/C++ en VScode iremos hacia las configuraciones de la extensión.



5. Ahi buscamos la opción que diga "compiler path", la cual nos mandara nos manda al archivo `settings.json`



6. Agregaremos estas líneas a los corchetes internos, teniendo cuidado de no borrar otras configuraciones.

```
"C_Cpp.default.compilerPath": "C:/msys64/ucrt64/bin",  
"C_Cpp.clang_format_path": "C:/msys64/ucrt64/bin",  
"code-runner.executorMap": {  
  "cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt -g && $dir$fileNameWithoutExt"  
}
```

7. Con esto ya podremos compilar y ejecutar el código dentro de VScode dándole click al símbolo de reproducir ▶ en la parte superior derecha del editor o con la extensión code runner ↕ presionando las teclas `ctrl + alt + N`.

# Integrar Codeforces con VScode

1. Primero en VScode tenemos que tener la extensión  Competitive Programming Helper (CPH) 
2. Después cambiaremos el lenguaje por defecto a C++ y agregaremos la ruta (ubicación) de la plantilla del club  a la configuración de la extensión Competitive Programming Helper (CPH) en VScode
3. Para  firefox: Agrega las extensiones:  competitive companion  y  CPH Submit 
4. Para navegadores basados en  chromium ( Google Chrome,  Microsoft Edge,  Opera,  Brave, etc.):  
Agrega las extensiones:  competitive companion  y  CPH Submit 

Para usarlas, ten tu sesión de codeforces activa y también ten abierto VScode, cuando abras un problema en codeforces haz click en , después cámbiate a VScode donde veras unas opciones donde eligieras cpp, y listo podrás probar y publicar tu código desde vscode.

## Git y Github

Git (git-scm) es una herramienta de código abierto para el control de versión distribuido mientras que GitHub es una plataforma y servicio basado en la nube, propiedad de Microsoft, para el desarrollo de software y el control de versiones utilizando Git, básicamente un servidor donde almacenas tus proyectos y los controlas con Git.

En el club recomendamos usar git para guardar tus códigos en GitHub, esto es útil no solo para las actividades del club, sino que podrás usar estas habilidades para tus próximos proyectos escolares o profesionales.

# Para conectar Git con Github

1. Primero crearemos una cuenta de GitHub en <https://github.com/signup> ↗
2. Después creamos un repositorio (proyecto) en <https://github.com/new> ↗
3. Luego descargamos git desde la página oficial <https://git-scm.com/downloads> ↗ o con el gestor de paquetes integrado en Windows `winget` a través de una terminal, con el comando:

```
winget install -e --id Git.Git
```

4. Si Windows no está en la última versión, comprobamos que está instalado el cliente de OpenSSH ↗.
5. En una terminal (Powershell  o git-bash ) creamos nuestro usuario local de git con los comandos:

```
git config --global user.name "Usuario"  
git config --global user.email "correo@ejemplo.com"
```

6. comprobamos nuestros datos con el comando:

```
git config --list
```

7. En la misma terminal creamos una llave pública de ssh:

```
ssh-keygen -t ed25519 -C "correo@ejemplo.com"
```

8. Copiamos los datos del archivo generado en `C:\Users\%USERNAME%\ssh\id_ed25519.pub` (en Windows tiene icono de Microsoft publisher  y se puede abrir con el bloc de notas) y lo conectamos con nuestra cuenta de github através de <https://github.com/settings/ssh/new> , dándole un nombre (puede ser el modelo de tu computadora) y pegando la los datos copiados (llave).

9. Por último descargamos nuestro repositorio, en una terminal navega hacia tu carpeta de destino, yo recomiendo la carpeta de Documentos (Documents) o Escritorio (Desktop).

💡 Mucho ojo 👁 en Windows, estas carpetas suelen estar bajo 'C:\Users\%USERNAME%\One Drive\'

```
git clone git@github.com:"Usuario de GitHub"/"Proyecto".git
```

10. Una vez ya tengamos conectado nuestra cuenta y hallamos descargado el proyecto, los tres comandos principales de git que usaremos (ya están integrados en VScode 🐙) son:

```
git commit -m "mensaje del commit" # cambios locales a "pushear"  
git push # pusheas los cambios junto con el commit hacia GitHub  
git pull # descargas los cambios actuales desde GitHub
```

# Documentación del lenguaje C++

- Tabla de valores ASCII ↗ al trabajar con `char` y `string` es útil ver esta tabla
- Microsoft Learn ↗ aunque está enfocado en su compilador MSVC tiene información muy valiosa sobre el lenguaje
- cplusplus ↗ vienen clases por librería y ejemplos de cada una de las funciones
- CPPReference ↗ muy similar a cplusplus
- W3school ↗ tutoriales interactivos para aprender C++
- GeeksForGeeks ↗ una de las mejores comunidades de programadores proporcionan estos blogs.
- CPPbyExample ↗ ejemplos muy útiles de varios temas de C++
- GNU libstdc++ ↗ esta página es muy técnica, pero te muestra las implementaciones de la STL de C++

# Referencias

- GitHub. (s.f). *Generating a new SSH key and adding it to the ssh-agent*. Recuperado de <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?platform=windows> ↑
- Josal. (2024). *What is an IDE? – Integrated Development Environment*. Recuperado de <https://www.geeksforgeeks.org/what-is-ide/> ↑
- RedHat. (2023). *¿Qué es y para qué sirve un IDE?*. Recuperado de <https://www.redhat.com/es/topics/middleware/what-is-ide> ↑ .
- VScode. (s.f). *Using GCC with MinGW*. Recuperado de <https://code.visualstudio.com/docs/cpp/config-mingw> ↑