























# Manual de instalación Programación Competitiva

# Contenido

Elija la sección de instalación 1 o 2 dependiendo de su Sistema Operativo (Windows o Linux)

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	











# Contenido

<b>1. Windows</b>	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
<b>2. Linux</b>	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	



# Windows

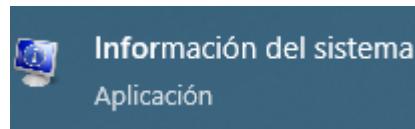
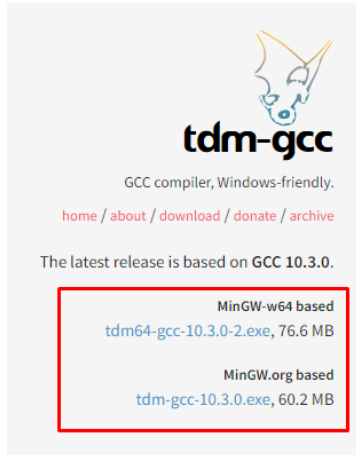
# Contenido

1. Windows	
<b>1.1. Instalación de C++</b>	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

# 1.1. Instalación de C++

Utilizaremos principalmente el lenguaje de programación C++. Podrá instalarlo siguiendo los siguientes pasos.

1. Ingrese al siguiente [link](#) para instalar MinGW (compilador de C++ para Windows)
2. En la barra lateral izquierda tendrá 2 opciones como se muestra en la imagen. Se recomienda descargar la primera, a excepción de si su computadora es de 32 bits, en cuyo caso descargue la 2da. En caso no sepa si su computadora es de 32 o 64 bits, puede ir a “Información del sistema” y revisar “Tipo de Sistema” – x64 es de 64 bits y x86 de 32 bits.

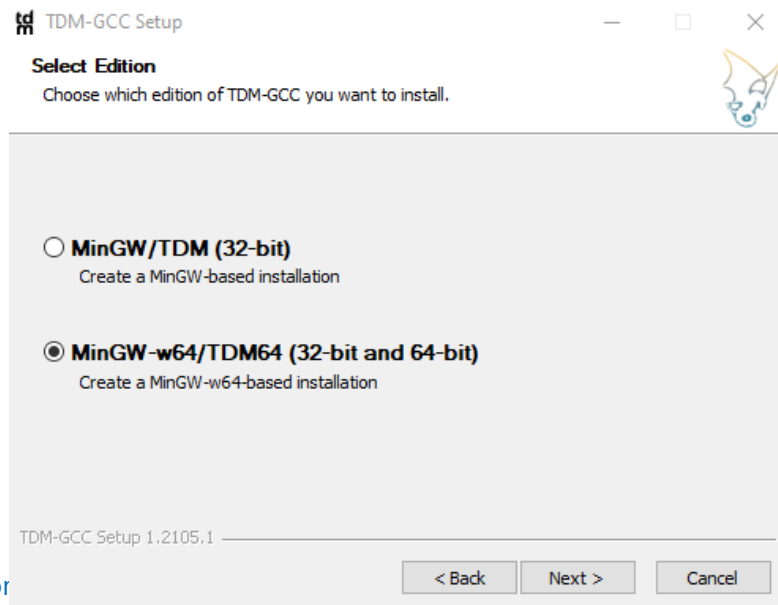
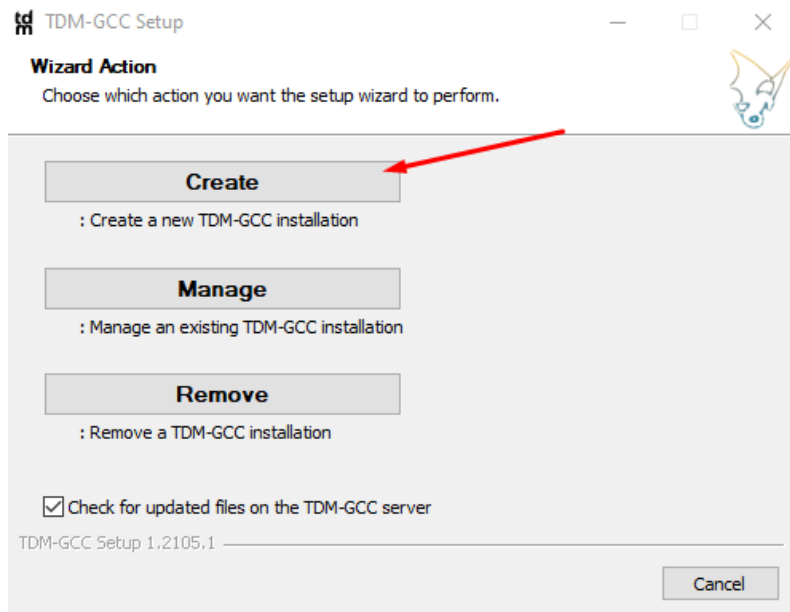


Tipo de sistema	PC basado en x64

# 1.1. Instalación de C++

3. Una vez descargado, abra el ejecutable y presione **Create** y a continuación seleccione la opción de 32 o 64 bits dependiendo de su computadora. Luego deje siempre las opciones por defecto dando click en *next* o *install*.

**Nota:** Trate de recordar en qué dirección se guardará MinGW. Probablemente sea en *C:/TDM-GCC-64*













# 1.1. Instalación de C++

4. Para comprobar que la instalación fue exitosa abra un *cmd*, escriba *g++* y presione enter. Le debe salir un mensaje como el siguiente.

```
D:\>g++  
g++: error fatal: no hay ficheros de entrada  
compilación terminada.
```



# Contenido

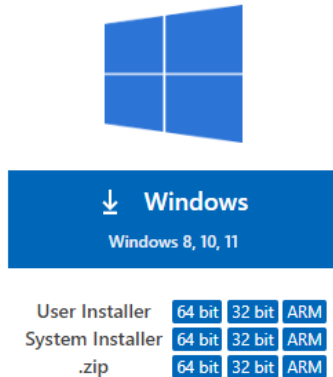
1. Windows	
1.1. Instalación de C++	
<b>1.2. Instalación del IDE</b>	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

## 1.2. Instalación del IDE



Para poder programar de forma más sencilla, utilizaremos un IDE, el cual es un término que hace referencia a una aplicación que ayuda en el desarrollo de programas. Existen muchos IDEs que puede usar, como por ejemplo VSCode o Geany. En esta guía les mencionaremos como instalar VSCode.

1. Ingrese al siguiente [link](#) para instalar VSCode y escoja la opción de Windows dependiendo de si su computadora es 64 o 32 bits y corra el ejecutable descargado.



# 1.2. Instalación del IDE

2. Elija las opciones por defecto hasta llegar a la siguiente pantalla donde puede seleccionar todas las opciones y luego darle a *Siguiente* y finalmente a *Instalar*.

## Seleccione las Tareas Adicionales

¿Qué tareas adicionales deben realizarse?



Seleccione las tareas adicionales que desea que se realicen durante la instalación de Visual Studio Code y haga clic en Siguiente.

Accesos directos adicionales:

☒ Crear un acceso directo en el escritorio

Otros:











☒ Agregar la acción "Abrir con Code" al menú contextual de archivo del Explorador de Windows

☒ Agregar la acción "Abrir con Code" al menú contextual de directorio del Explorador de Windows

☒ Registrar Code como editor para tipos de archivo admitidos

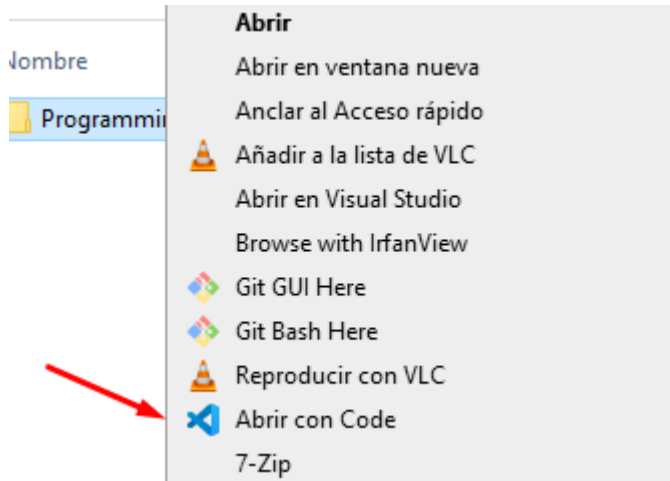
☒ Agregar a PATH (disponible después de reiniciar)

# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
<b>1.3. Configuración del IDE</b>	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

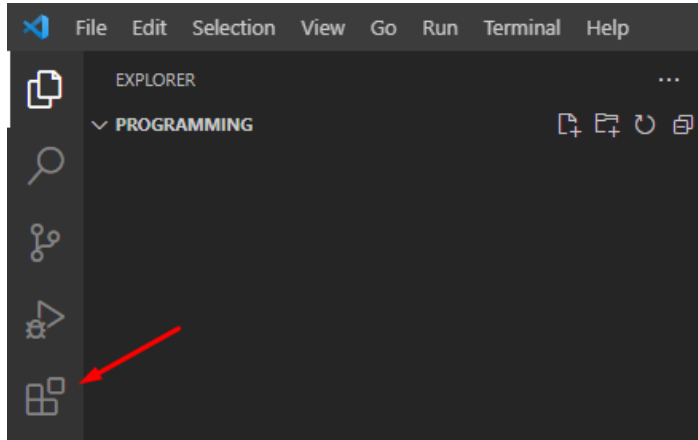
# 1.3. Configuración del IDE

1. Es recomendable que cree una carpeta en donde se desarrollen los futuros programas que realizará en el grupo de Programación Competitiva. Supongamos que crea una carpeta en su escritorio llamada *Programming*.
2. Abra VSCode. Puede abrirlo para su carpeta directamente haciendo click derecho en ella y luego dándole a “Abrir con Code”

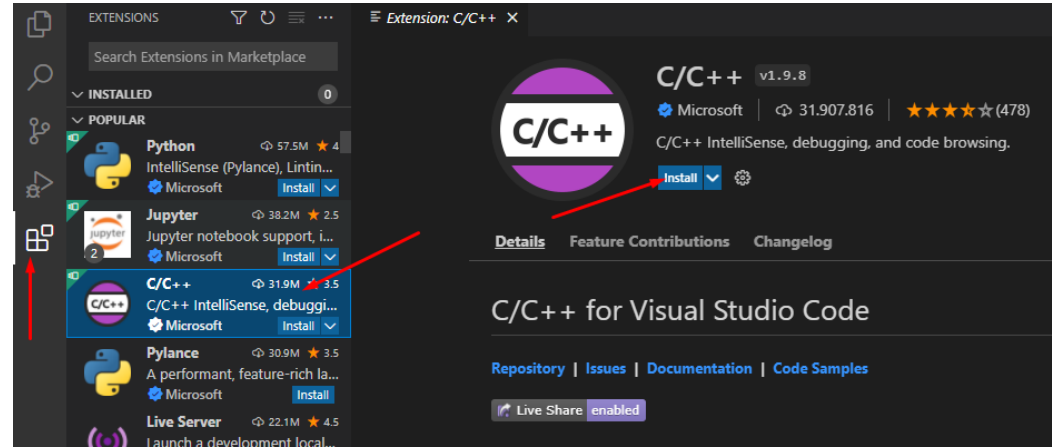


# 1.3. Configuración del IDE

3. Abra VSCode y en la parte de la izquierda elige el último botón ir al panel de extensiones.

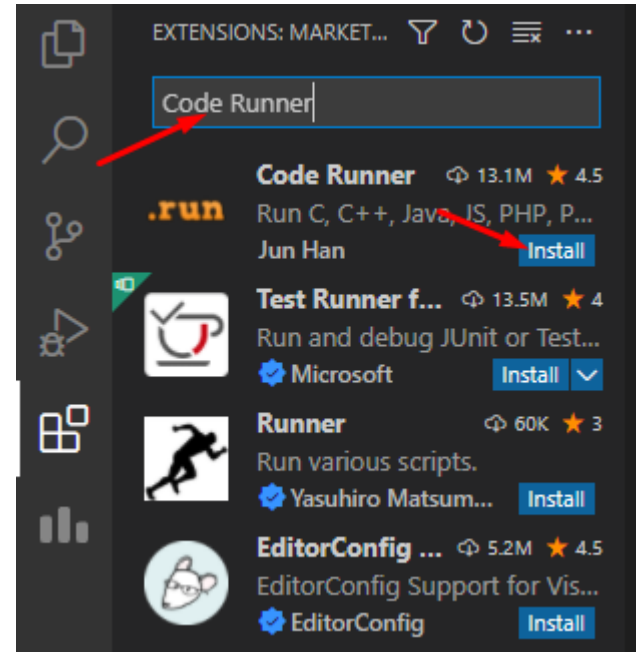
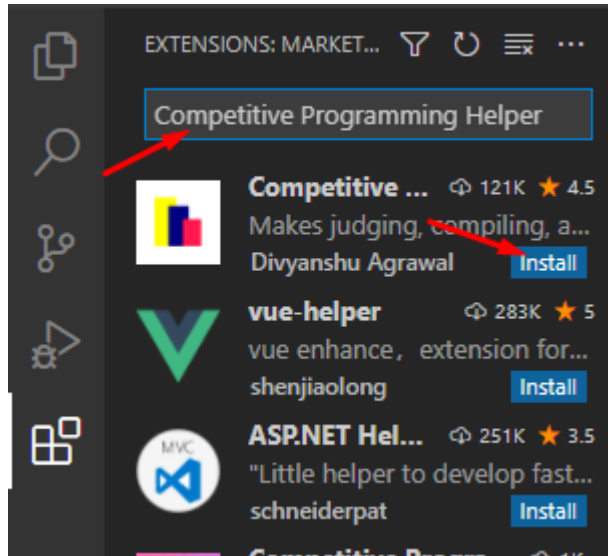


4. Instale la extensión de C/C++



# 1.3. Configuración del IDE

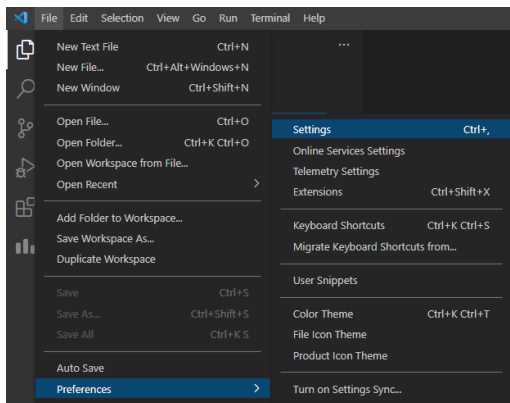
5. En el mismo panel busca e instala la extensión *Competitive Programming Helper*
6. En el mismo panel busca e instala la *Code Runner*



# 1.3. Configuración del IDE

Para que las herramientas instaladas funcionen correctamente, configuraremos algunas opciones recomendadas.

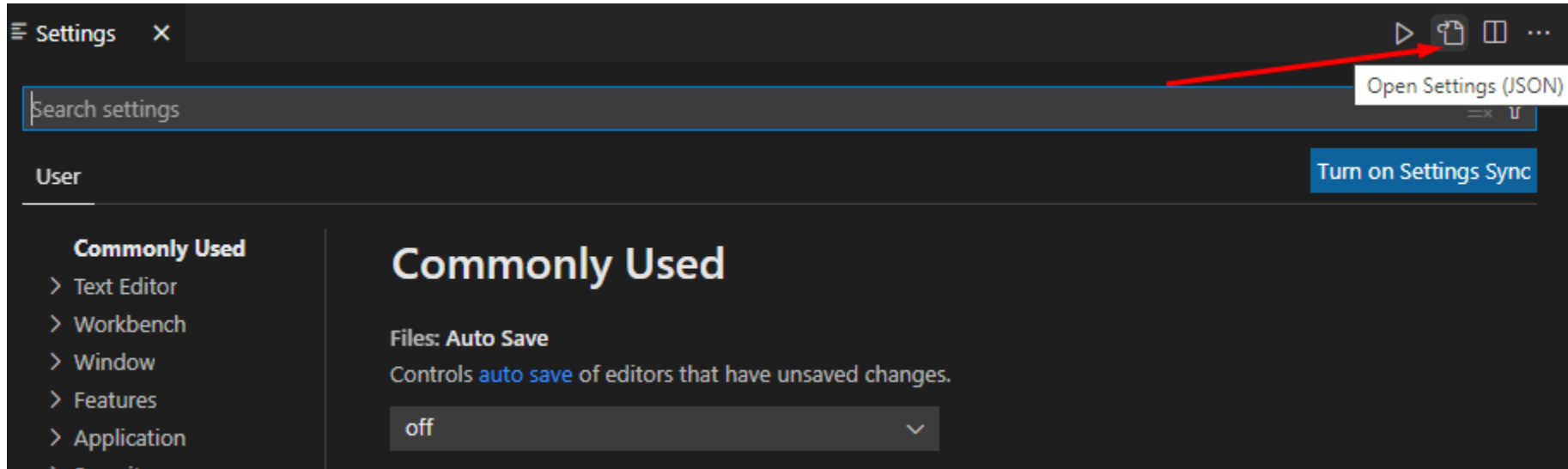
7. Una de las herramientas requiere guardar ciertos datos “temporales” en alguna carpeta para poder hacer bien su trabajo. Por lo tanto, cree una carpeta vacía con el nombre *cph-files* en el lugar que usted crea conveniente. Por ejemplo supongamos que usted la crea dentro de su carpeta *Programming*. La ubicación probablemente sea algo como *C:/Users/User/Escritorio/Programming/cph-files*. Esta ubicación será importante para un paso posterior.
8. Vaya a *File* → *Preferences* → *Settings*





# 1.3. Configuración del IDE

7. En *Settings*, en la parte superior derecha haga click al siguiente botón que dice *Open Settings (JSON)*



# 1.3. Configuración del IDE











8. Debe encontrarse en un archivo llamado *settings.json*. En ese archivo copie el contenido de este [link](#) y péguelo. Note que puede ser que deba cambiar el “**C\_Cpp.default.compilerPath**” dependiendo de la ubicación de MinGW y “**cph.general.saveLocation**” dependiendo de dónde creo la carpeta del paso anterior para archivos temporales. Le debe quedar algo similar a la imagen.

```
1 {
2   "C_Cpp.default.compilerPath": "C:/TDM-GCC-64/bin/g++.exe",
3   "C_Cpp.default.cppStandard": "c++17",
4   "C_Cpp.clang_format_fallbackStyle": "{BasedOnStyle: Google, IndentWidth: 4, SpacesBeforeTrailingComments: 1, All...
5   "C_Cpp.autocomplete": "Disabled",
6   "editor.formatOnSave": true,
7   "[cpp]": {
8     "editor.wordBasedSuggestions": true,
9     "editor.quickSuggestionsDelay": 500,
10  },
11  "cph.general.saveLocation": "C:/Users/User/Escritorio/Programming/cph-files",
12  "cph.language.cpp.Args": "-Wl,--stack=268435456 -O2 -std=c++17",
13  "cph.general.timeOut": 5000,
14  "code-runner.runInTerminal": true,
15  "code-runner.executorMap": {
16    "cpp": "g++ -Wl,--stack=268435456 -O2 -std=c++17 $fileName -o $fileNameWithoutExt && $fileNameWithoutExt",
17  },
18  "code-runner.fileDirectoryAsCwd": true,
19  "code-runner.saveFileBeforeRun": true,
20  "code-runner.ignoreSelection": true,
21  "terminal.integrated.defaultProfile.windows": "Command Prompt",
22 }
```

La ruta en donde instaló  
MingW seguido de  
**/bin/g++.exe**

La ruta en donde creó  
la carpeta para guardar  
archivos temporales

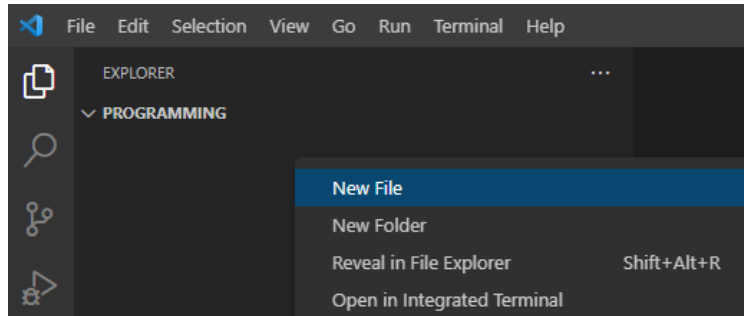
# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
<b>1.4. Comprobar el funcionamiento del IDE</b>	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

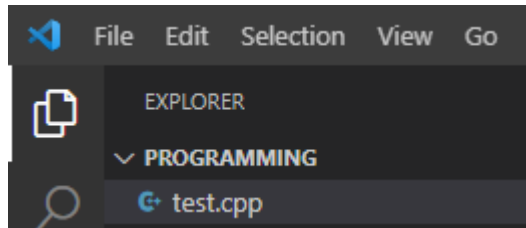
# 1.4. Comprobar el funcionamiento del IDE

Para comprobar que VSCode y las herramientas instaladas funcionan correctamente haga lo siguiente

1. Cierre VSCode y vuélvalo a abrir desde su carpeta *Programming*.
2. Cree una nuevo archivo dando click derecho y luego *New File*.



3. Póngale de nombre *test.cpp*



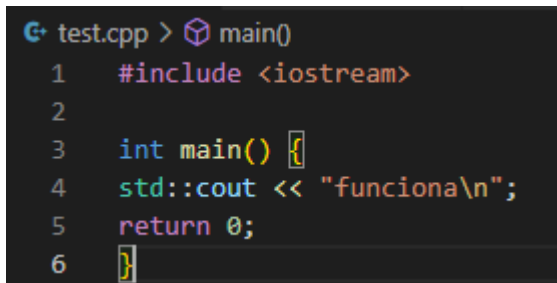
# 1.4. Comprobar el funcionamiento del IDE

4. Copie el siguiente texto y péguelo en su archivo

```
#include <iostream>

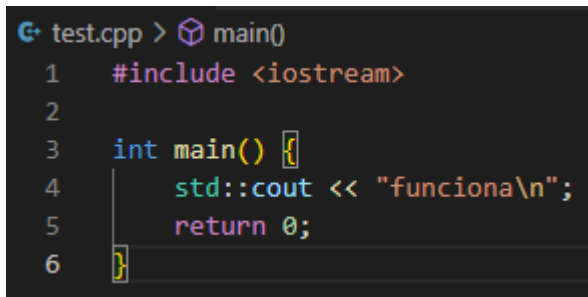
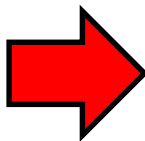
int main() {
std::cout << "funciona\n";
return 0;
}
```

5. Presione *CTRL* + *S* para guardar los cambios. Su archivo debe formatearse automáticamente con unos espacios de tabulación en la línea 4 y 5. Esto valida que el formatter funciona. ✓



test.cpp > main()

```
1  #include <iostream>
2
3  int main() {
4  std::cout << "funciona\n";
5  return 0;
6  }
```

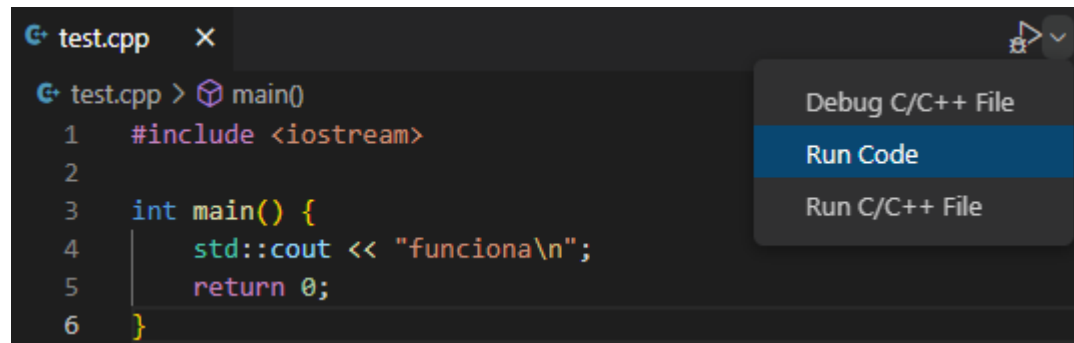


test.cpp > main()

```
1  #include <iostream>
2
3  int main() {
4      std::cout << "funciona\n";
5      return 0;
6  }
```

# 1.4. Comprobar el funcionamiento del IDE

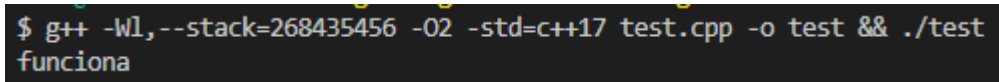
6. En la parte superior derecha encontrará un botón donde podrá darle a la flecha y luego a *Run Code*. Se le debe abrir una terminal en la parte inferior donde se ejecute el código y salga el mensaje “funciona”. Esto valida que el runner funciona. ✓✓



The screenshot shows an IDE window with a file named `test.cpp`. The code in the editor is:

```
test.cpp > main()
1  #include <iostream>
2
3  int main() {
4      std::cout << "funciona\n";
5      return 0;
6  }
```

A context menu is open over the Run button (a play icon) in the top right corner of the editor. The menu contains three options: "Debug C/C++ File", "Run Code" (which is highlighted in blue), and "Run C/C++ File".

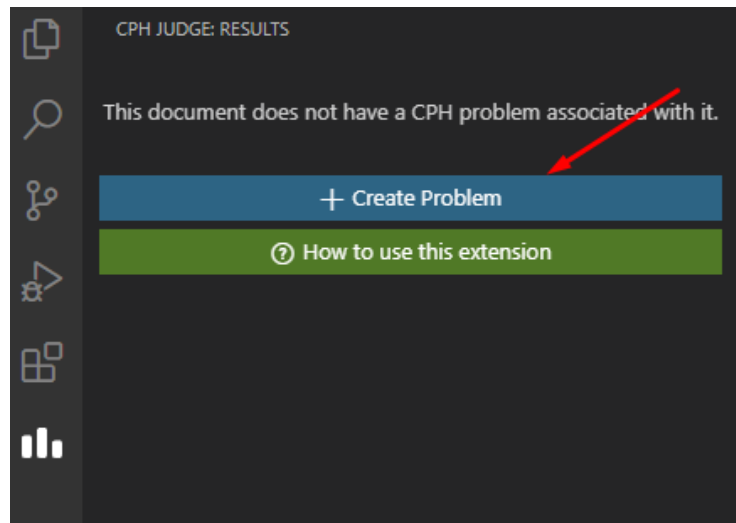
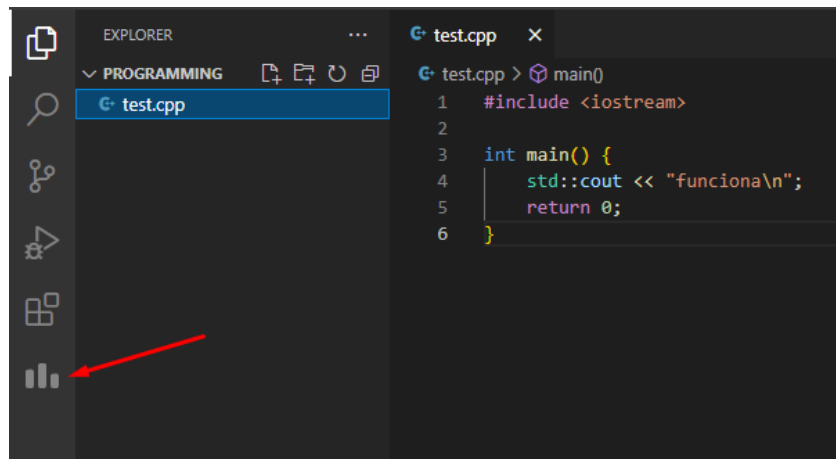


The screenshot shows a terminal window with the following command and output:

```
$ g++ -Wl,--stack=268435456 -O2 -std=c++17 test.cpp -o test && ./test
funciona
```

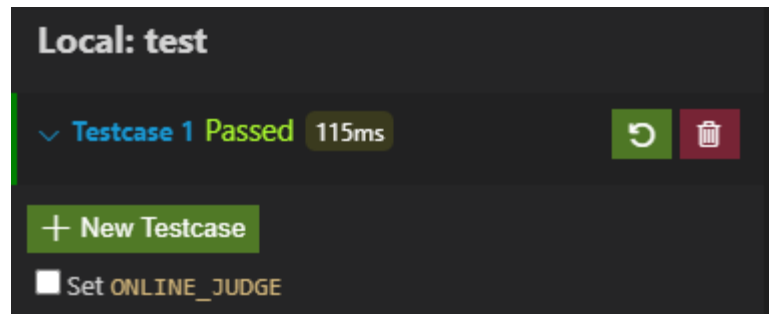
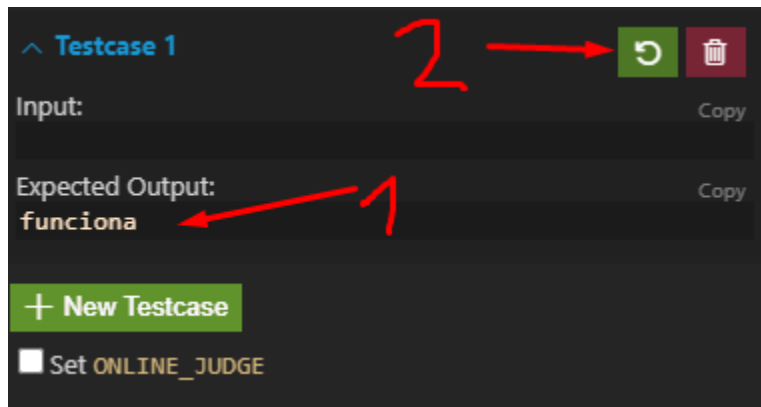
# 1.4. Comprobar el funcionamiento del IDE

7. En la barra de la izquierda, haga click en el último ícono. Se le abrirá un panel a la izquierda (agrándelo de ser necesario). Haga click en *Create Problem*.













# 1.4. Comprobar el funcionamiento del IDE

8. Dentro del panel del paso anterior, en *Expected Output* escriba “funciona” y luego aprete el botón de ejecutar en la parte superior derecha. Le debe salir *Passed*. Esto valida que el helper funciona. ✓✓✓















# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
<b>2. Linux</b>	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	



Linux

# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
<b>2.1. Instalación de C++</b>	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

## 2.1. Instalación de C++











Utilizaremos principalmente el lenguaje de programación C++. Podrá instalarlo ejecutando el siguiente comando.

```
sudo apt-get install g++
```

Para comprobar que la instalación fue exitosa en la terminal escriba `g++`. Debe aparecerle algo como el siguiente mensaje.

```
@PC001:~$ g++  
g++: fatal error: no input files  
compilation terminated.
```

# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
<b>2.2. Instalación del IDE</b>	
2.3. Configuración del IDE	
2.4. Comprobar el funcionamiento del IDE	

## 2.2. Instalación del IDE













Para poder programar de forma más sencilla, utilizaremos un IDE, el cual es un término que hace referencia a una aplicación que ayuda en el desarrollo de programas. Existen muchos IDEs que puede usar, como por ejemplo VSCode o Geany. En esta guía les mencionaremos como instalar VSCode.

Visite el siguiente [link](#) y busque la forma de instalación correcta según su distribución de Linux. Una de las formas más sencillas es con snap (en caso lo tenga instalado), bajo el siguiente comando.

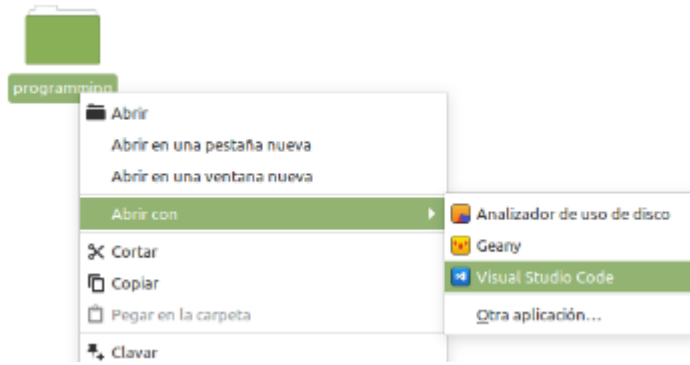
```
sudo snap install code --classic
```

# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
<b>2.3. Configuración del IDE</b>	
2.4. Comprobar el funcionamiento del IDE	

## 2.3. Configuración del IDE

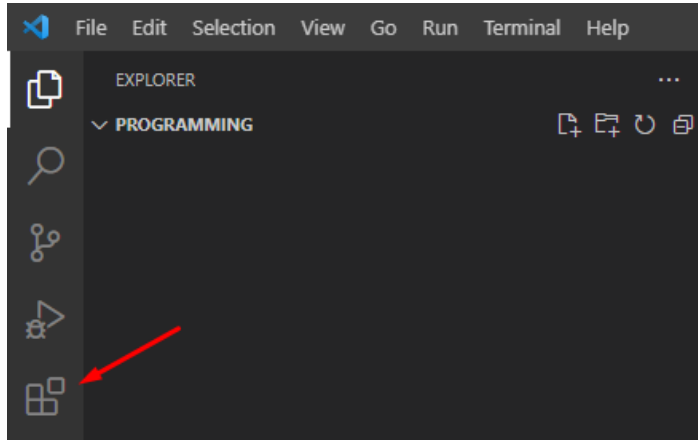
1. Es recomendable que cree una carpeta en donde se desarrollen los futuros programas que realizará en el grupo de Programación Competitiva. Supongamos que crea una carpeta en su escritorio llamada *programming*.
2. Abra VSCode. Puede abrirlo para su carpeta directamente haciendo click derecho en ella y luego dándole a “Abrir con Visual Studio Code”



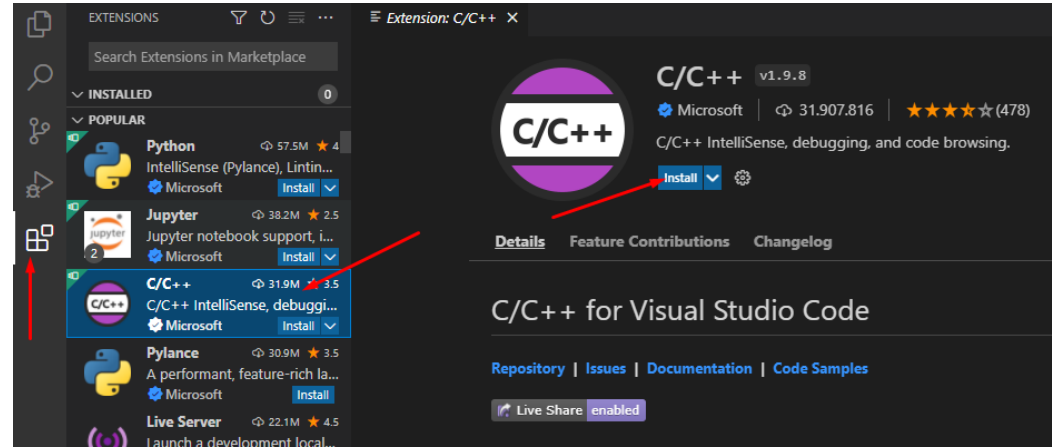


## 2.3. Configuración del IDE

3. Abra VSCode y en la parte de la izquierda elige el último botón ir al panel de extensiones.

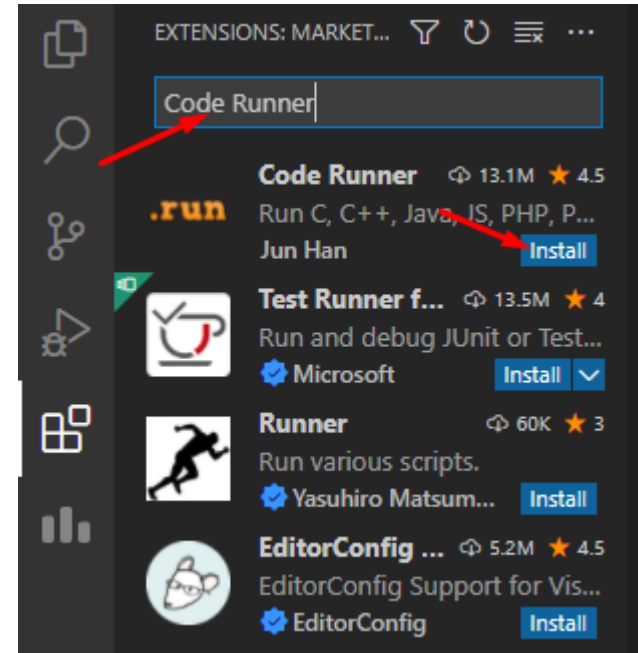
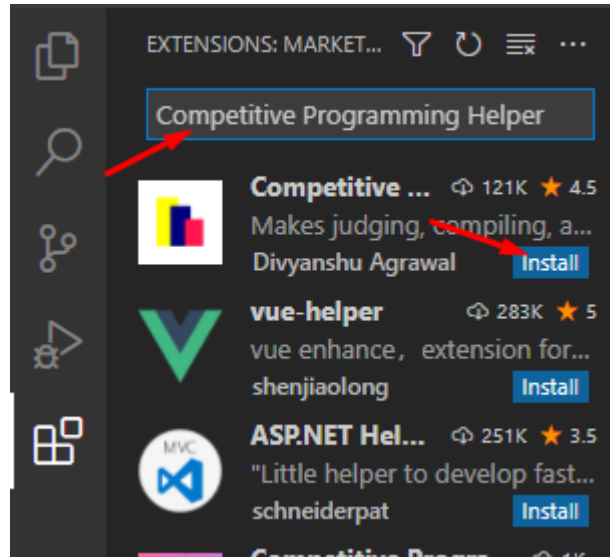


4. Instale la extensión de C/C++



## 2.3. Configuración del IDE

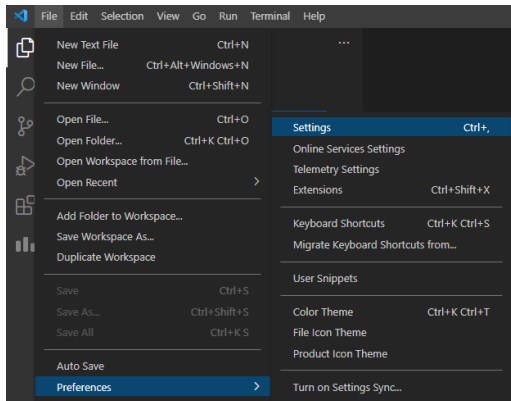
5. En el mismo panel busca e instala la extensión *Competitive Programming Helper*
6. En el mismo panel busca e instala la *Code Runner*



## 2.3. Configuración del IDE

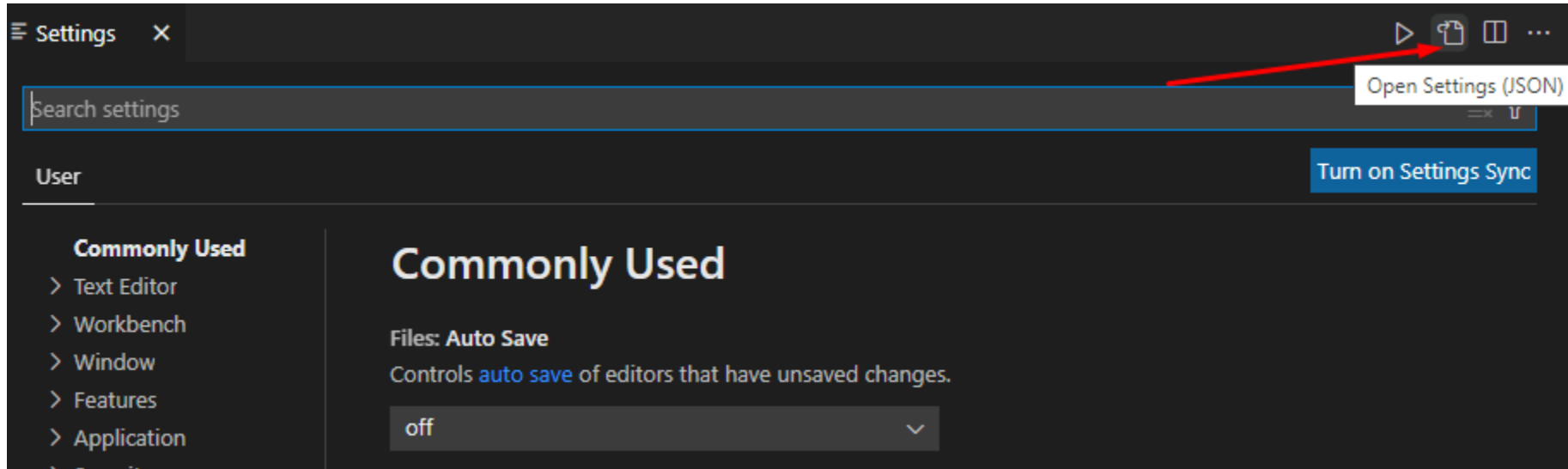
Para que las herramientas instaladas funcionen correctamente, configuraremos algunas opciones recomendadas.

- Una de las herramientas requiere guardar ciertos datos “temporales” en alguna carpeta para poder hacer bien su trabajo. Por lo tanto, cree una carpeta vacía con el nombre *cph-files* en el lugar que usted crea conveniente. Por ejemplo supongamos que usted la crea dentro de su carpeta *programming*. La ubicación probablemente sea algo como */home/user/desktop/programming*. Esta ubicación será importante para un paso posterior.
- Vaya a *File* → *Preferences* → *Settings*



## 2.3. Configuración del IDE

7. En *Settings*, en la parte superior derecha haga click al siguiente botón que dice *Open Settings (JSON)*



## 2.3. Configuración del IDE

8. Debe encontrarse en un archivo llamado *settings.json*. En ese archivo copie el contenido de este [link](#) y péguelo. Note que puede ser que deba cambiar el “**C\_Cpp.default.compilerPath**” dependiendo de la ubicación de su compilador de C++ y “**cph.general.saveLocation**” dependiendo de dónde creo la carpeta del paso anterior para archivos temporales. Le debe quedar algo similar a la imagen.











```
1  {
2    "C_Cpp.default.compilerPath": "/usr/bin/g++",
3    "C_Cpp.default.cppStandard": "c++17",
4    "C_Cpp.clang_format_fallbackStyle": "{BasedOnStyle: Google, IndentWidth: 4, SpacesBeforeTrailingComments: 1}",
5    "C_Cpp.autocomplete": "Disabled",
6    "editor.formatOnSave": true,
7    "[cpp]": {
8      "editor.wordBasedSuggestions": true,
9      "editor.quickSuggestionsDelay": 500,
10   },
11   "cph.general.saveLocation": "/home/graphtronco/Escritorio/Programming/cph-files",
12   "cph.language.cpp.Args": "-O2 -std=c++17",
13   "cph.general.timeOut": 5000,
14   "code-runner.runInTerminal": true,
15   "code-runner.executorMap": {
16     "cpp": "g++ -O2 -std=c++17 $fileName -o $fileNameWithoutExt && ./$fileNameWithoutExt",
17   },
18   "code-runner.fileDirectoryAsCwd": true,
19   "code-runner.saveFileBeforeRun": true,
20 }
```

Para saber donde está su compilador de C++ ejecute el comando **which g++** en la terminal

```
@PC001:~$ which g++
/usr/bin/g++
```

La ruta en donde creó la carpeta para guardar archivos temporales

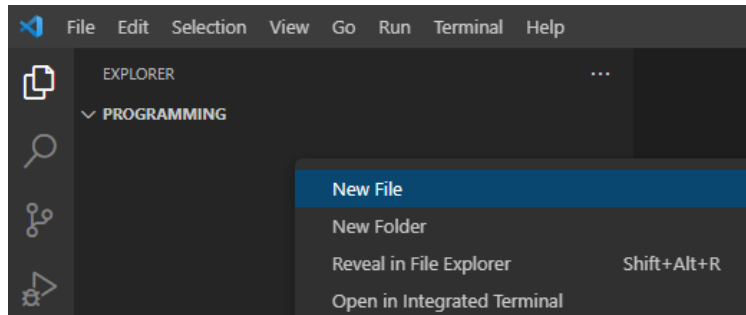
# Contenido

1. Windows	
1.1. Instalación de C++	
1.2. Instalación del IDE	
1.3. Configuración del IDE	
1.4. Comprobar el funcionamiento del IDE	
2. Linux	
2.1. Instalación de C++	
2.2. Instalación del IDE	
2.3. Configuración del IDE	
<b>2.4. Comprobar el funcionamiento del IDE</b>	

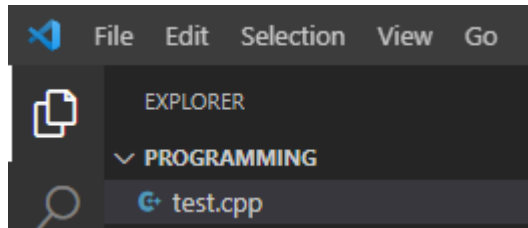
## 2.4. Comprobar el funcionamiento del IDE

Para comprobar que VSCode y las herramientas instaladas funcionan correctamente haga lo siguiente

1. Cierre VSCode y vuélvalo a abrir desde su carpeta *Programming*.
2. Cree una nuevo archivo dando click derecho y luego *New File*.



3. Póngale de nombre *test.cpp*



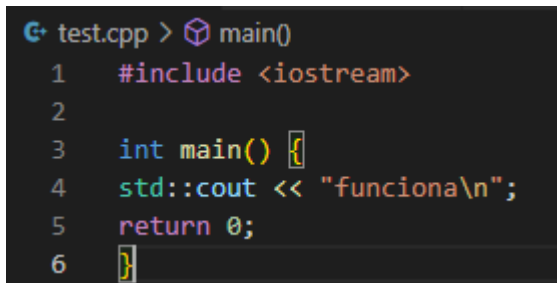
## 2.4. Comprobar el funcionamiento del IDE

4. Copie el siguiente texto y péguelo en su archivo

```
#include <iostream>

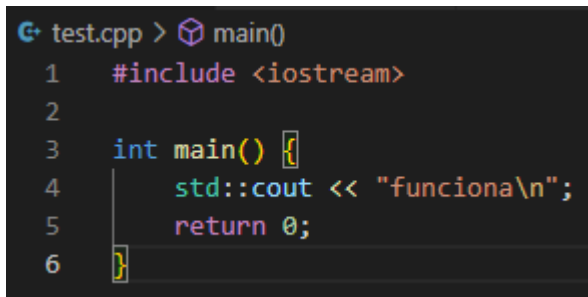
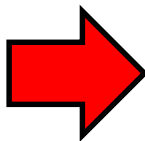
int main() {
std::cout << "funciona\n";
return 0;
}
```

5. Presione *CTRL* + *S* para guardar los cambios. Su archivo debe formatearse automáticamente con unos espacios de tabulación en la línea 4 y 5. Esto valida que el formatter funciona. ✓



test.cpp > main()

```
1  #include <iostream>
2
3  int main() {
4  std::cout << "funciona\n";
5  return 0;
6  }
```



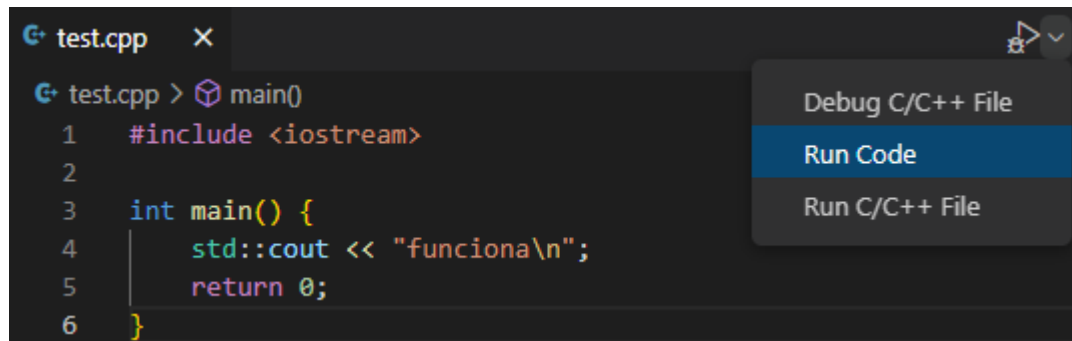
test.cpp > main()

```
1  #include <iostream>
2
3  int main() {
4      std::cout << "funciona\n";
5      return 0;
6  }
```



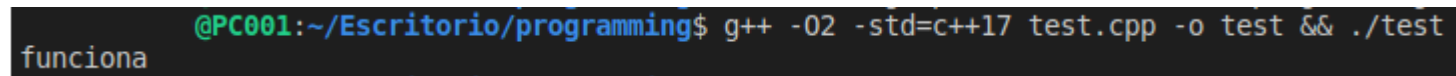
## 2.4. Comprobar el funcionamiento del IDE

6. En la parte superior derecha encontrará un botón donde podrá darle a la flecha y luego a *Run Code*. Se le debe abrir una terminal en la parte inferior donde se ejecute el código y salga el mensaje “funciona”. Esto valida que el runner funciona. ✓✓



The screenshot shows a code editor with a file named `test.cpp`. The code contains a simple C++ program that prints "funciona\n" and returns 0. A context menu is open over the code, showing options: "Debug C/C++ File", "Run Code" (highlighted), and "Run C/C++ File".

```
test.cpp > main()
1  #include <iostream>
2
3  int main() {
4      std::cout << "funciona\n";
5      return 0;
6  }
```

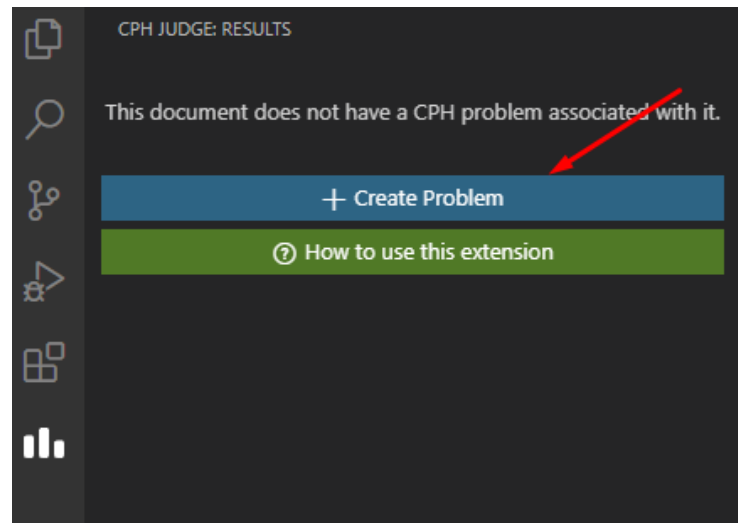
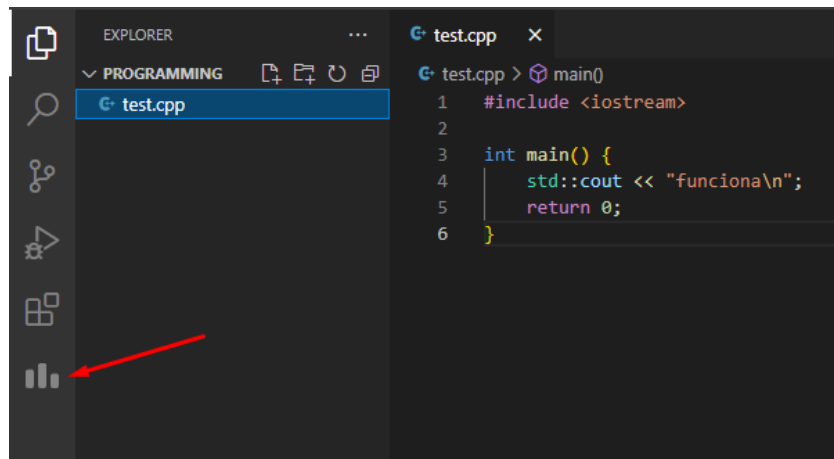


The screenshot shows a terminal window with the command `g++ -O2 -std=c++17 test.cpp -o test && ./test` being executed. The output of the program is "funciona".

```
@PC001:~/Escritorio/programming$ g++ -O2 -std=c++17 test.cpp -o test && ./test
funciona
```

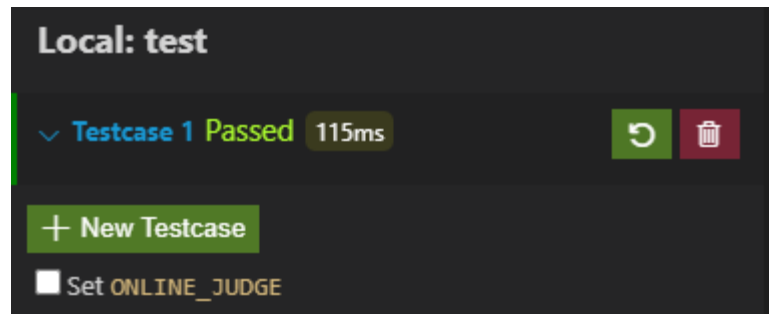
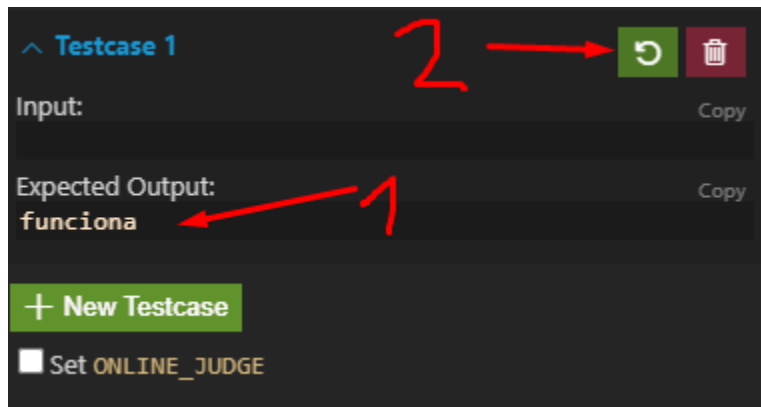
## 2.4. Comprobar el funcionamiento del IDE

7. En la barra de la izquierda, haga click en el último ícono. Se le abrirá un panel a la izquierda (agrándelo de ser necesario). Haga click en *Create Problem*.



## 2.4. Comprobar el funcionamiento del IDE

8. Dentro del panel del paso anterior, en *Expected Output* escriba “funciona” y luego aprete el botón de ejecutar en la parte superior derecha. Le debe salir *Passed*. Esto valida que el helper funciona. ✓✓✓



# ¡Gracias por su atención!

