EBook Gratis

APRENDIZAJE vscode

Free unaffiliated eBook created from
Stack Overflow contributors.

#vscode

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vscode

It is an unofficial and free vscode ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vscode.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con vscode

Observaciones

Esta sección proporciona una descripción general de qué es vscode, y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de vscode, y vincular a los temas relacionados. Dado que la Documentación para vscode es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Versión	Fecha de lanzamiento
0.10.1-extensionbuilders	2015-11-13
0.10.1	2015-11-17
0.10.2	2015-11-24
0.10.3	2015-11-26
0.10.5	2015-12-17
0.10.6	2015-12-19
0.10.7-insiders	2016-01-29
0.10.8	2016-02-05
0.10.8-insiders	2016-02-08
0.10.9	2016-02-17
0.10.10-insiders	2016-02-26
0.10.10	2016-03-11
0.10.11	2016-03-11
0.10.11-insiders	2016-03-11
0.10.12-insiders	2016-03-20
0.10.13-insiders	2016-03-29
0.10.14-insiders	2016-04-04

Versión	Fecha de lanzamiento
0.10.15-insiders	2016-04-11
1.0.0	**2016-04-14**
1.1.0-insider	2016-05-02
1.1.0	2016-05-09
1.1.1	2016-05-16
1.2.0	2016-06-01
1.2.1	2016-06-14
1.3.0	2016-07-07
1.3.1	2016-07-12
1.4.0	2016-08-03
Traducción / 20160817.01	2016-08-17
Traducción / 20160826.01	2016-08-26
Traducción / 20160902.01	2016-09-02
1.5.0	2016-09-08
1.5.1	2016-09-08
1.5.2	2016-09-14
1.6.0	2016-10-10
1.6.1	2016-10-13
traducción / 20161014.01	2016-10-14
traducción / 20161028.01	2016-10-28
1.7.0	2016-11-01
1.7.1	2016-11-03
traducción / 20161111.01	2016-11-12
traducción / 20161118.01	2016-11-19
1.7.2	2016-11-22

Versión	Fecha de lanzamiento
traducción / 20161125.01	2016-11-26
traducción / 20161209.01	2016-12-09
1.8.0	2016-12-14
1.8.1	2016-12-20
Traducción / 20170123.01	2017-01-23
Traducción / 20172701.01	2017-01-27
1.9.0	2017-02-02
Traducción / 20170127.01	2017-02-03
Traducción / 20170203.01	2017-02-03
1.9.1	2017-02-09
Traducción / 20170217.01	2017-02-18
Traducción / 20170227.01	2017-02-27
1.10.0	2017-03-01
1.10.1	2017-03-02
1.10.2	2017-03-08
Traducción / 20170311.01	2017-03-11
Traducción / 20170317.01	2017-03-18
Traducción / 20170324.01	2017-03-25
Traducción / 20170331.01	2017-03-31
1.11.0	2017-04-06
1.11.1	2017-04-06
Traducción / 20170407.01	2017-04-07
1.11.2	2017-04-13

Examples

En Windows

- Descarga el instalador de Visual Studio Code para Windows.
- Una vez que se descargue, ejecute el instalador (VSCodeSetup-version.exe). Esto solo tomará un minuto.

De forma predeterminada, el Código VS se instala en C: \ Archivos de programa (x86) \ Código VS de Microsoft para una máquina de 64 bits.

Nota: .NET Framework 4.5.2 es necesario para el Código VS. Si está utilizando Windows 7, asegúrese de que .NET Framework 4.5.2 esté instalado.

Consejo: El programa de instalación agregará opcionalmente el código de Visual Studio a su% PATH%, por lo que desde la consola puede escribir 'código'. para abrir VS Code en esa carpeta. Deberá reiniciar la consola después de la instalación para que el cambio a la variable ambiental% PATH% tenga efecto.

En mac

- Descargar Visual Studio Code para Mac.
- Haga doble clic en el archivo descargado para expandir los contenidos.
- Arrastre Visual Studio Code.app a la carpeta Aplicaciones, haciéndolo disponible en el Launchpad.
- Agregue VS Code a su Dock haciendo clic derecho en el ícono y seleccionando Opciones, Mantener en el Dock.

También puede ejecutar VS Code desde el terminal escribiendo 'código' después de agregarlo a la ruta:

- Lanzar VS Code.
- Abra la paleta de comandos (Ctrl + Shift + P) y escriba 'comando de shell' para encontrar el comando de shell: comando 'código' en el comando PATH.

Reinicie el terminal para que el nuevo valor de $ PATH tenga efecto. Podrás escribir 'código'. en cualquier carpeta para comenzar a editar archivos en esa carpeta.

Nota: si aún tiene el alias de código antiguo en su perfil .bash (o equivalente) de una versión anterior de Código VS, elimínelo y reemplácelo ejecutando el comando Shell: Instale el comando 'código' en el comando PATH.

En linux

Distribuciones basadas en Debian y Ubuntu.

La forma más fácil de instalar para las distribuciones basadas en Debian / Ubuntu es descargar e instalar el paquete .deb (64 bits) a través del centro de software gráfico si está disponible o a través de la línea de comandos con:

```
sudo dpkg -i <file>.deb
sudo apt-get install -f # Install dependencies
```

La instalación del paquete .deb instalará automáticamente el repositorio de apt y la clave de firma para habilitar la actualización automática utilizando el mecanismo normal del sistema. Tenga en cuenta que los binarios de 32 bits y .tar.gz también están disponibles en la página de descarga.

El repositorio y la clave también se pueden instalar manualmente con el siguiente script:

```
curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.gpg
sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main" >
/etc/apt/sources.list.d/vscode.list'
```

Luego actualice el paquete de caché e instale el paquete usando:

```
sudo apt-get update
sudo apt-get install code # or code-insiders for insiders build
```

Distribuciones basadas en RHEL, Fedora y CentOS.

Actualmente enviamos el Código VS estable de 64 bits en un repositorio yum, la siguiente secuencia de comandos instalará la clave y el repositorio:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio
Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ngpgcheck=1\ngpgkey=https://pac
> /etc/yum.repos.d/vscode.repo'
```

Luego actualice el paquete de caché e instale el paquete usando dnf (Fedora 22 y superior):

```
dnf check-update
sudo dnf install code
```

O en versiones anteriores usando yum:

```
yum check-update
sudo yum install code
```

Distribuciones basadas en openSUSE y SLE

El repositorio yum anterior también funciona para sistemas basados en openSUSE y SLE, la siguiente secuencia de comandos instalará la clave y el repositorio:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio
Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ntype=rpm-
md\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/zypp/repos.d/vscode.repo'
```

Luego actualice el paquete de caché e instale el paquete usando:

```
sudo zypper refresh
sudo zypper install code
```

Paquete AUR para Arch Linux

Existe un paquete de Repositorio de usuarios arch (AUR) mantenido por la comunidad para el Código VS.

Instalación del paquete .rpm manualmente El paquete .rpm (64 bits) también se puede descargar e instalar manualmente, sin embargo, la actualización automática no funcionará a menos que se instale el repositorio anterior. Una vez descargado, se puede instalar utilizando su gestor de paquetes, por ejemplo con dnf:

```
sudo dnf install <file>.rpm
```

Tenga en cuenta que los binarios de 32 bits y .tar.gz también están disponibles en la página de descarga .

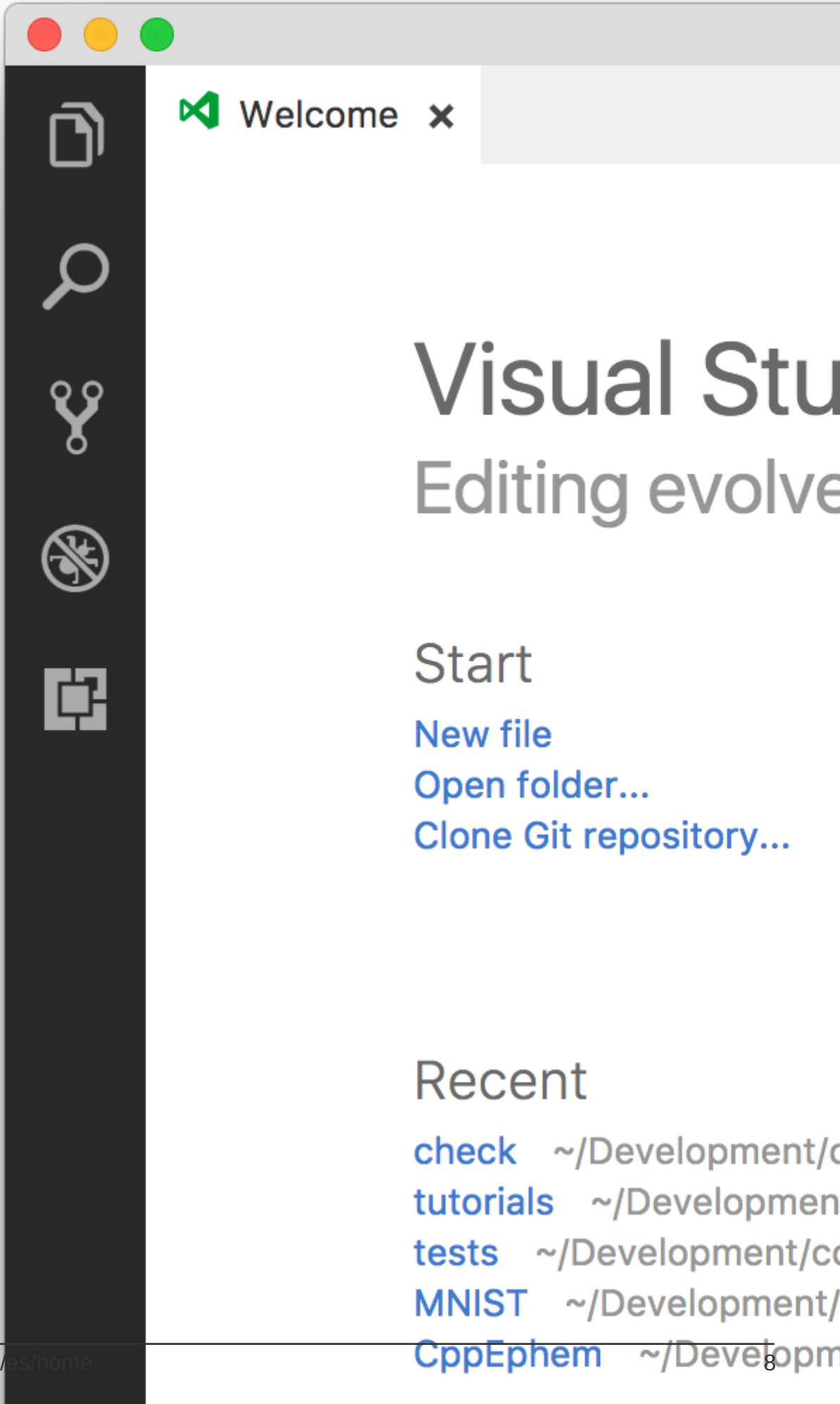
Primeros pasos (C ++): HelloWorld.cpp

El primer programa que se escribe normalmente en cualquier idioma es el script "hola mundo". Este ejemplo muestra cómo escribir este programa y depurarlo usando el código de Visual Studio (de ahora en adelante me referiré al código de Visual Studio como código VS).

Crear el proyecto

El paso 1 será crear un nuevo proyecto. Esto se puede hacer de varias maneras. La primera forma es directamente desde la interfaz de usuario.

1. Programa abierto de código VS. Recibirá una bienvenida con la pantalla de bienvenida estándar (tenga en cuenta que las imágenes se toman mientras trabaja en una Mac, pero deberían ser similares a su instalación):


Welcome
Visual Stu
Editing evolve
Start
New file
Open folder...
Clone Git repository...
Recent
check ~/Development/c
tutorials ~/Developmen
tests ~/Development/c
MNIST ~/Development/
CppEphem ~/Developm

. Esto abrirá una nueva ventana de edición donde podremos comenzar a construir nuestro script. Continúe y guarde este archivo (puede usar el menú *Archivo* > *Guardar* para hacer esto). Para este ejemplo, llamaremos al archivo **HelloWorld.cpp** y lo **colocaremos** en un nuevo directorio al que llamaremos **VSC_HelloWorld /** .

3. Escribe el programa. Esto debería ser bastante sencillo, pero siéntase libre de copiar lo siguiente en el archivo:

```
#include <iostream>

int main() {
   std::cout << "Hello world!" << std::endl;
   return 0;
}
```

Ejecutar el código

A continuación queremos ejecutar el script y comprobar su salida. hay muchas maneras de hacer esto. Lo más sencillo es abrir una terminal y navegar al directorio que creamos. Ahora puede compilar el script y ejecutarlo con gcc escribiendo:

```
$ g++ HelloWorld.cpp -o helloworld
$ ./helloworld
Hello World!
```

¡Yay, el programa funcionó! Pero esto no es realmente lo que queremos. Sería mucho mejor si pudiéramos ejecutar el programa desde el propio VSCode. Aunque estamos de suerte! VSCode tiene un terminal incorporado al que podemos acceder a través del menú " **Ver** "> " **Terminal integrado** ". Esto abrirá un terminal en la mitad inferior de la ventana desde la que puede navegar al directorio **VSC_HelloWorld** y ejecutar los comandos anteriores.

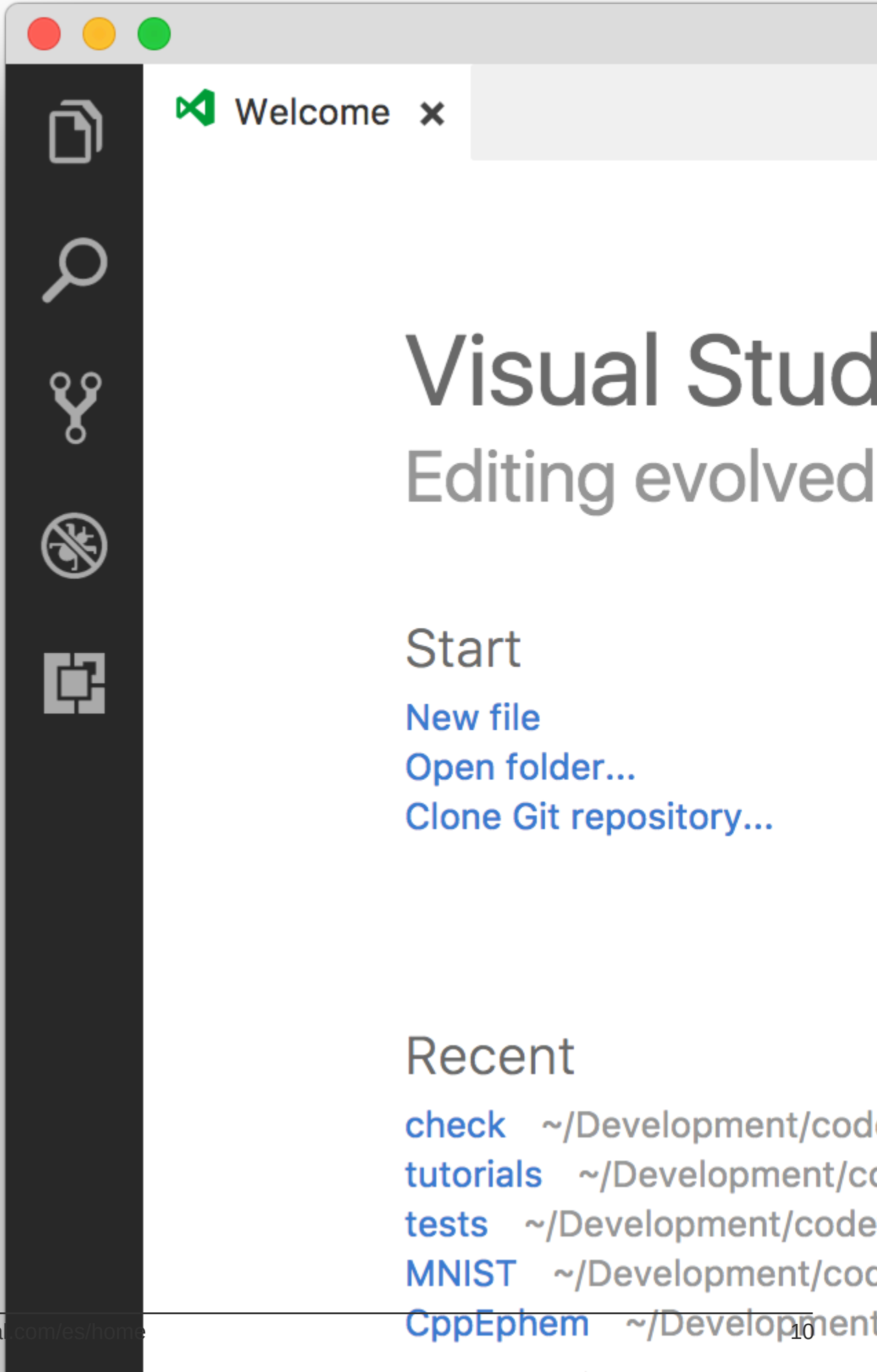
Normalmente hacemos esto ejecutando una *tarea de ejecución* . En el menú, seleccione " **Tareas** "> " **Ejecutar tarea ...** ". Notará que aparece una pequeña ventana emergente cerca de la parte superior de la ventana con un mensaje de error (algo en la línea de

Primer programa (C ++): Hello World.cpp

Este ejemplo lo introduce a la funcionalidad básica de VS Code al demostrar cómo escribir un programa "hola mundo" en C ++. Antes de continuar, asegúrese de tener instalada la extensión " **ms-vscode.cpptools** ".

Inicializar el proyecto

El primer paso es crear un nuevo proyecto. Para ello, cargue el programa VS Code. Debería ser recibido con la típica pantalla de bienvenida:


Welcome
Visual Stud
Editing evolved
Start
New file
Open folder...
Clone Git repository...
Recent
check ~/Development/cod
tutorials ~/Development/c
tests ~/Development/code
MNIST ~/Development/co
CppEphem ~/Developmen

" en la pantalla de bienvenida. Esto abrirá una nueva ventana de archivo. Continúe y guarde el archivo (" **Archivo** "> " **Guardar** ") en un nuevo directorio. Puede nombrar el directorio como desee, pero este ejemplo llamará al directorio " **VSC_HelloWorld** " y al archivo " **HelloWorld.cpp** ".

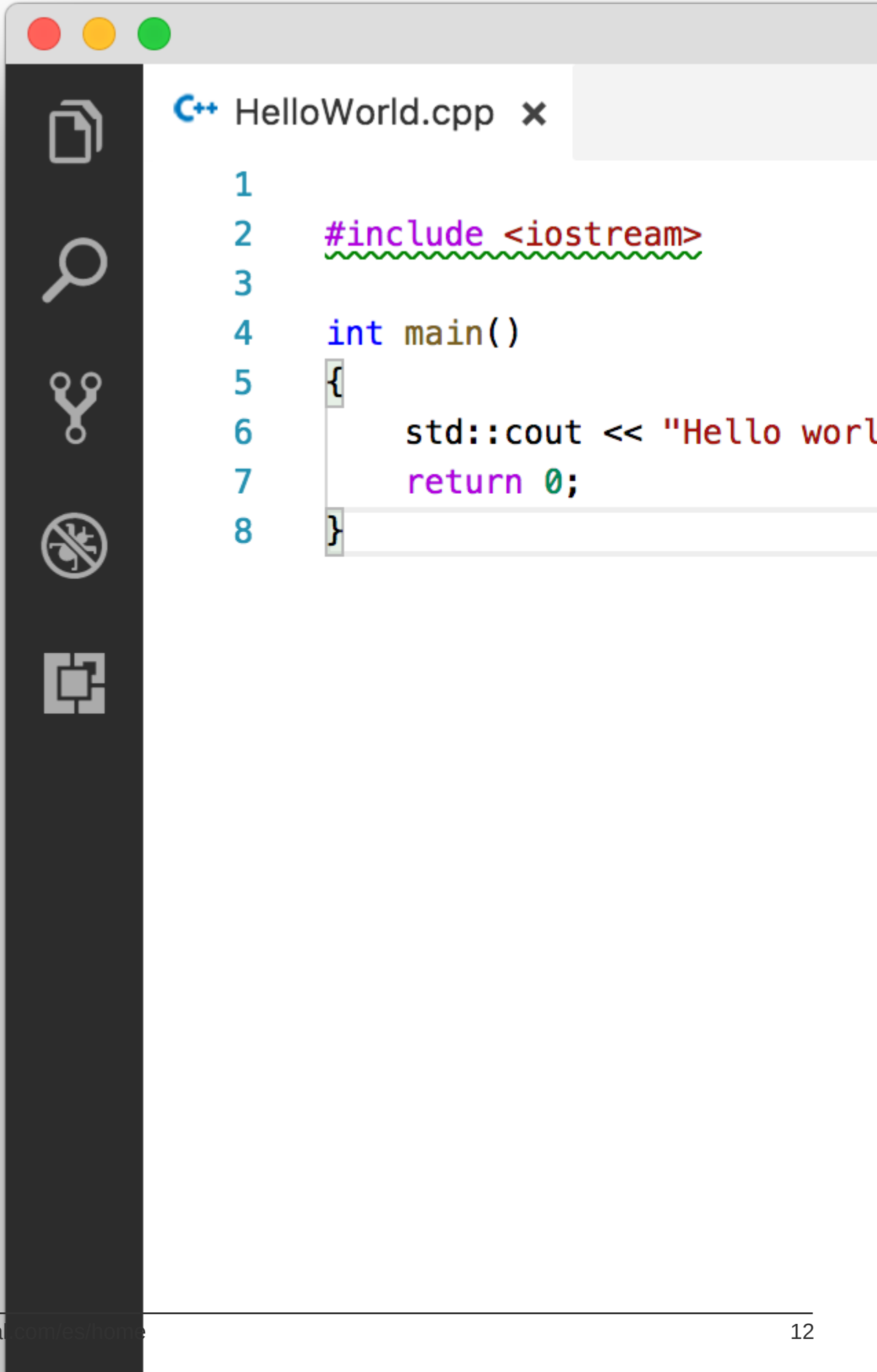
Ahora escriba el programa real (siéntase libre de copiar el siguiente texto):

```
#include <iostream>

int main()
{
    // Output the hello world text
    std::cout << "Hello world!" << std::endl;
    return 0;
}
```

¡Genial! También notará que, debido a que ha instalado la extensión " **ms-vscode.cpptools** ", también tiene un destacado resaltado de código. Ahora pasemos a ejecutar el código.

Ejecutando el Script (básico)

Podemos ejecutar " **HelloWorld.cpp** " desde dentro del propio Código VS. La forma más sencilla de ejecutar dicho programa es abrir el terminal integrado (" **Ver** "> " **Terminal integrado** "). Esto abre una ventana de terminal en la parte inferior de la vista. Desde dentro de este terminal podemos navegar a nuestro directorio creado, construir y ejecutar el script que hemos escrito.


HelloWorld.cpp
1
2 #include <iostream>
3
4 int main()
5 {
6 std::cout << "Hello worl
7 return 0;
8 }

Aquí hemos usado los siguientes comandos para compilar y ejecutar el código:

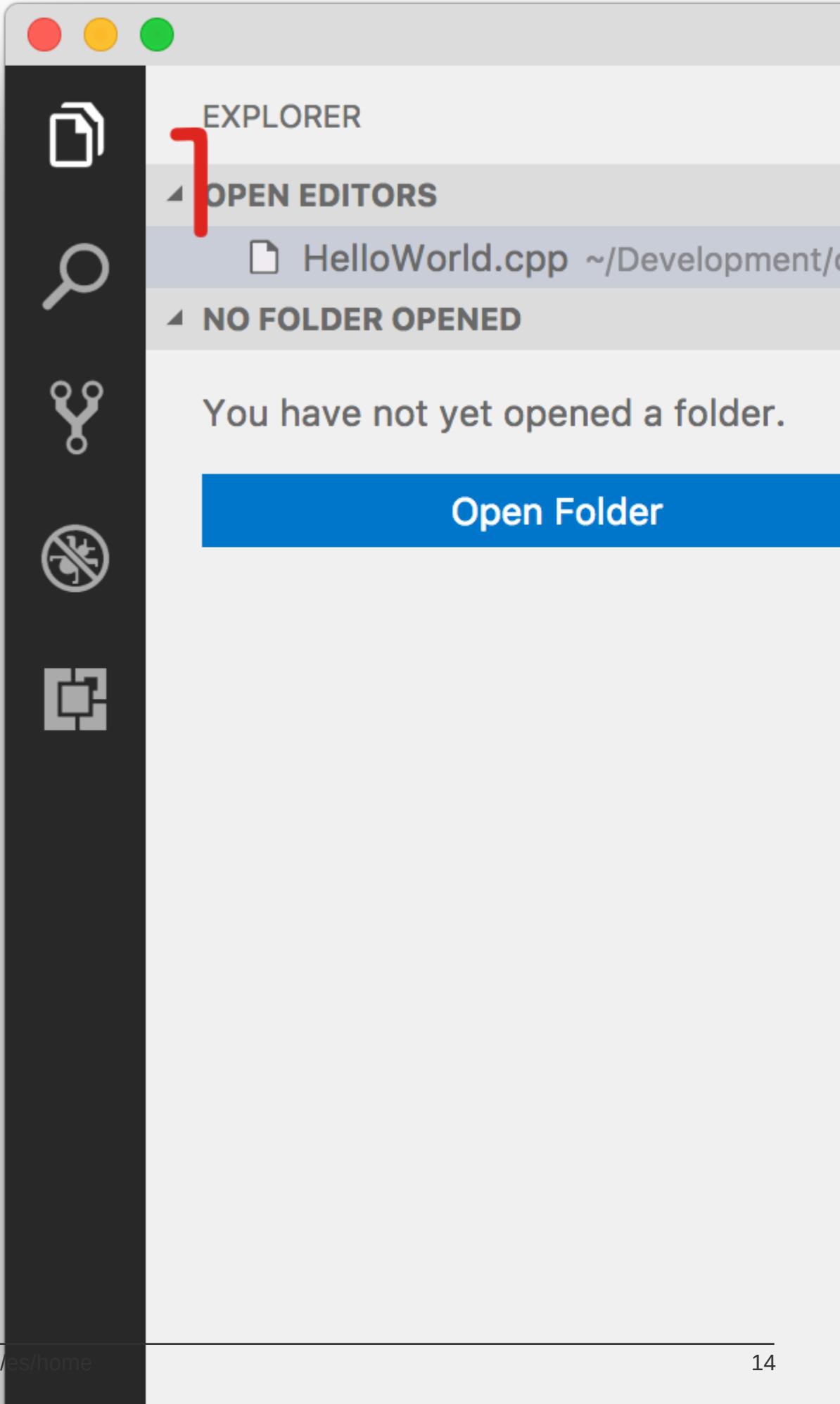
```
$ g++ HelloWorld.cpp -o hellowold
$ ./hellowold
```

Tenga en cuenta que conseguimos el esperado `Hello World!` salida.

Ejecutando el Script (un poco más avanzado)

Genial, pero podemos usar el Código VS directamente para compilar y ejecutar el código también. Para esto, primero debemos convertir el directorio " **VSC_HelloWorld** " en un espacio de trabajo. Esto se puede hacer por:

1. Abrir el menú del *Explorador* (el elemento más superior en el menú vertical en el extremo izquierdo)
2. Seleccione el botón *Abrir carpeta*
3. Seleccione el directorio " **VSC_HelloWorld** " en el que hemos estado trabajando.


EXPLORER
OPEN EDITORS
HelloWorld.cpp ~/Development/
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder

*Nota: si abre un directorio dentro de Código VS (por ejemplo, usando " **Archivo** "> " **Abrir ...** ") ya estará en un área de trabajo.*

El menú del *explorador* ahora muestra el contenido del directorio.

A continuación, queremos definir las tareas reales que queremos que ejecute VS Code. Para hacer esto, seleccione " **Tareas** "> " **Configurar tarea de compilación predeterminada** ". En el menú desplegable, seleccione " **Otro** ". Esto abre un nuevo archivo llamado " **tareas.json** " que contiene algunos valores predeterminados para una tarea. Necesitamos cambiar estos valores. Actualice este archivo para que contenga lo siguiente y guárdelo:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "taskName": "build",
            "type": "shell",
            "command": "g++ HelloWorld.cpp -o helloworld"
        },
        {
            "taskName": "run",
            "type": "shell",
            "command": "${workspaceRoot}/helloworld"
        }
    ]
}
```

*Tenga en cuenta que lo anterior también crea un directorio oculto **.vscode** dentro de nuestro directorio de trabajo. Aquí es donde VS Code coloca los archivos de configuración, incluidos los archivos de configuración específicos del proyecto. Puede encontrar más información sobre las tareas aquí .*

En el ejemplo anterior, `${workspaceRoot}` referencia al directorio de nivel superior de nuestro espacio de trabajo, que es nuestro directorio " **VSC_HelloWorld** ". Ahora, para compilar el proyecto desde el método, seleccione " **Tareas** "> " **Ejecutar tarea de compilación ...** " y seleccione nuestra tarea de " **compilación** " creada y " *Continuar sin escanear el resultado de la tarea* " en los menús desplegables que aparecen. Luego, podemos ejecutar el ejecutable utilizando " **Tareas** "> " **Ejecutar tarea ...** " y seleccionando la tarea " **ejecutar** " que creamos. Si tiene el terminal integrado abierto, notará que el "¡Hola mundo!" El texto será impreso allí.

Es posible que el terminal se cierre antes de que pueda ver la salida. Si esto sucede, puede insertar una línea de código como esta `int i; std::cin >> i;` justo antes de la declaración de retorno al final de la función `main()` . Luego, puede finalizar el script escribiendo cualquier número y presionando <*Entrar*> .

¡Y eso es! Ahora puede comenzar a escribir y ejecutar sus scripts C ++ desde dentro de VS Code.

Lea Empezando con vscode en línea: https://riptutorial.com/es/vscode/topic/4489/empezando-con-vscode

Capítulo 2: Múltiples proyectos establecidos.

Observaciones

Las pruebas unitarias del proyecto configuradas actualmente se pueden encontrar aquí.

Examples

Referencia a proyectos locales.

No hay cosas como los archivos `.sln` y `.proj` .
En lugar de ellos, las **carpetas** se utilizan en Visual Studio Code.
Cada carpeta de proyecto debe tener un archivo `project.json` separado.

```
/MyProject.Core
   SourceFile.cs
   project.json

/MyProject.Web
   /Controllers
   /Views
   project.json
```

Para hacer referencia a `MyProject.Core` desde el proyecto `MyProject.Web` , edite el archivo `MyProject.Web\project.json` y agregue la dependencia:

```
// MyProject.Web/project.json
{
   "dependencies": {
      "MyProject.Core": {"target": "project"},
     ...
   }
   "buildOptions": {
      "emitEntryPoint": true
   }
}
```

La línea `"emitEntryPoint": true` dice que `MyProject.Web` es un proyecto de inicio para la solución. `MyProject.Core` debe tener este indicador deshabilitado en su archivo `project.json` :

```
 // MyProject.Core/project.json
 {
   ...
   "buildOptions": {
      "emitEntryPoint": false
   }
}
```

Cree el proyecto (Mac: ⌘ + `Shift` + `B` , Windows: `Ctrl` + `Shift` + `B`) y cada proyecto debe tener sus propias carpetas `\bin` y `\obj` con nuevos archivos `.dll` .

Estructura de la solución

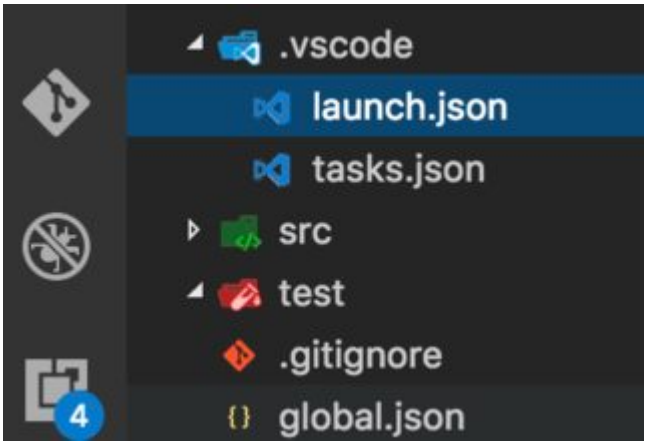
Es muy común agrupar proyectos, por ejemplo, colocar proyectos de prueba en la carpeta `/test` y proyectos de origen en la carpeta `/src` . Agregue el archivo `global.json` y haga una estructura similar:

```
global.json
/src/
    /MyProject.Core/
       SourceFile.cs
       project.json

    /MyProject.Web/
       /Controllers
       /Views
       project.json

/test/
    /MyProject.Core.UnitTests/
       SourceFileTest.cs
       project.json

    /MyProject.Web.UnitTests/
       /Controllers
       /Views
       project.json
```

Edite el archivo `global.json` vacío y especifique los grupos de proyectos:

```
{
    "projects":["src", "test"]
}
```

VS Code usa `tasks.json` para ejecutar tareas (por ejemplo, construir una solución) y `launch.json` para iniciar un proyecto (por ejemplo, depurar). Si no puede encontrar estos archivos, intente iniciar la depuración presionando `F5` e ignore los errores, VS Code generará en la carpeta raíz `.vscode` carpeta con los archivos.



Edite el archivo `launch.json` y especifique la ruta a su biblioteca de inicio, cambie `MyProject.Web` con el nombre de su proyecto:

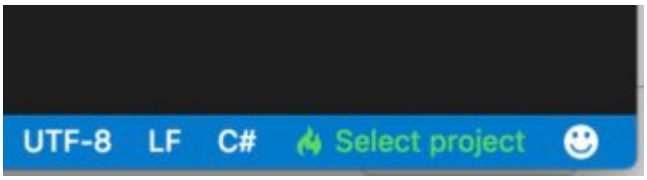
```
{
    "configurations": [
    {
        ...
        "program":
"${workspaceRoot}/src/MyProject.Web/bin/Debug/netcoreapp1.0/MyProject.Web.dll",
        "args": [],
        "cwd": "${workspaceRoot}/src/Washita.Web",
        ...
   }
}
```

Edite el archivo `tasks.json` y especifique la ruta a su biblioteca de inicio, cambie `MyProject.Web` con el nombre de su proyecto:

```
{
    "tasks": [
        {
        "taskName": "build",
            "args": [
                "${workspaceRoot}/src/MyProject.Web"
           ],
           "isBuildCommand": true,
           "problemMatcher": "$msCompile"
       }
   ]
}
```

Ahora debería poder construir y depurar archivos de origen .NET.

Sin embargo, Intellisense desaparecerá debido a la configuración de múltiples proyectos. Para solucionarlo, abra cualquier archivo `.cs` y cambie al proyecto apropiado (project.json) `Select project` en la esquina inferior derecha:



Lea Múltiples proyectos establecidos. en línea: https://riptutorial.com/es/vscode/topic/7717/multiples-proyectos-establecidos-

Creditos

S. No	Capítulos	Contributors
1	Empezando con vscode	Community, H. Pauwelyn, Jvinniec, Kronos, RamenChef, Sahan Serasinghe
2	Múltiples proyectos establecidos.	Artru