**EBook Gratis**

# APRENDIZAJE vala

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#vala

# Tabla de contenido

# Acerca de

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

It is an unofficial and free vala 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 vala.



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 vala

## Observaciones

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

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

## Versiones

| Versión | Fecha de lanzamiento |
|---|---|
| 0.36.4 | 2017-06-26 |
| 0.36.3 | 2017-05-02 |
| 0.36.2 | 2017-04-25 |
| 0.36.1 | 2017-04-03 |
| 0.36.0 | 2017-03-18 |
| 0.34.6 | 2017-03-02 |
| 0.34.5 | 2017-03-02 |
| 0.34.4 | 2016-12-05 |
| 0.34.3 | 2016-11-22 |
| 0.34.2 | 2016-10-23 |
| 0.34.1 | 2016-10-09 |
| 0.34.0 | 2016-09-19 |
| 0.32.1 | 2016-06-20 |
| 0.32.0 | 2016-03-21 |
| 0.31.1 | 2016-02-07 |
| 0.30.2 | 2016-06-20 |
| 0.30.1 | 2016-01-31 |

| Versión | Fecha de lanzamiento |
| --- | --- |
| 0.30.0 | 2015-09-18 |
| 0.29.3 | 2015-08-11 |
| 0.29.2 | 2015-06-22 |
| 0.29.1 | 2015-05-27 |
| 0.28.1 | 2015-08-11 |
| 0.28.0 | 2015-03-22 |
| 0.27.2 | 2015-03-18 |
| 0.27.1 | 2015-01-12 |
| 0.26.2 | 2015-01-12 |
| 0.26.1 | 2014-10-13 |
| 0.26.0 | 2014-09-22 |
| 0.25.4 | 2014-09-15 |
| 0.25.3 | 2014-09-01 |
| 0.25.2 | 2014-08-24 |
| 0.25.1 | 2014-07-23 |
| 0.24.0 | 2014-03-24 |
| 0.23.3 | 2014-02-18 |
| 0.23.2 | 2014-02-05 |
| 0.23.1 | 2013-12-22 |
| 0.22.1 | 2013-11-13 |
| 0.22.0 | 2013-09-23 |
| 0.21.2 | 2013-09-13 |
| 0.21.1 | 2013-08-02 |
| 0.20.1 | 2013-04-08 |
| 0.20.0 | 2013-03-26 |

| Versión | Fecha de lanzamiento |
| --- | --- |
| 0.19.0 | 2013-02-20 |
| 0.18.1 | 2012-11-13 |
| 0.18.0 | 2012-09-24 |
| 0.17.7 | 2012-09-16 |
| 0.17.6 | 2012-09-03 |
| 0.17.5 | 2012-08-20 |
| 0.17.4 | 2012-08-06 |
| 0.17.3 | 2012-07-16 |
| 0.17.2 | 2012-06-24 |
| 0.17.1 | 2012-06-02 |
| 0.17.0 | 2012-04-28 |
| 0.16.1 | 2012-06-23 |
| 0.16.0 | 2012-03-26 |
| 0.15.2 | 2012-02-25 |
| 0.15.1 | 2012-01-26 |
| 0.15.0 | 2011-12-05 |
| 0.14.2 | 2012-01-31 |
| 0.14.1 | 2011-11-30 |
| 0.14.0 | 2011-09-17 |
| 0.13.4 | 2011-09-07 |
| 0.13.3 | 2011-08-22 |
| 0.13.2 | 2011-08-16 |
| 0.13.1 | 2011-07-06 |
| 0.13.0 | 2011-06-17 |
| 0.12.1 | 2011-06-01 |

| Versión | Fecha de lanzamiento |
| --- | --- |
| 0.12.0 | 2011-04-03 |
| 0.11.7 | 2011-03-16 |
| 0.11.6 | 2011-02-14 |
| 0.11.5 | 2011-01-21 |
| 0.11.4 | 2011-01-15 |
| 0.11.3 | 2011-01-05 |
| 0.11.2 | 2010-11-08 |
| 0.11.1 | 2010-10-25 |
| 0.11.0 | 2010-10-04 |
| 0.10.4 | 2011-03-12 |
| 0.10.3 | 2011-01-22 |
| 0.10.2 | 2010-12-28 |
| 0.10.1 | 2010-10-26 |
| 0.10.0 | 2010-09-18 |
| 0.9.8 | 2010-09-04 |
| 0.9.7 | 2010-08-19 |
| 0.9.6 | 2010-08-18 |
| 0.9.5 | 2010-08-09 |
| 0.9.4 | 2010-07-27 |
| 0.9.3 | 2010-07-14 |
| 0.9.2 | 2010-06-20 |
| 0.9.1 | 2010-06-07 |
| 0.8.1 | 2010-04-21 |
| 0.8.0 | 2010-03-31 |
| 0.7.10 | 2010-02-04 |

| Versión | Fecha de lanzamiento |
|---|---|
| 0.7.9 | 2009-12-19 |
| 0.7.8 | 2009-11-04 |
| 0.7.7 | 2009-09-27 |
| 0.7.6 | 2009-09-18 |
| 0.7.5 | 2009-08-02 |
| 0.7.4 | 2009-06-28 |
| 0.7.3 | 2009-05-26 |
| 0.7.2 | 2009-05-07 |
| 0.7.1 | 2009-04-20 |
| 0.7.0 | 2009-04-05 |
| 0.6.1 | 2009-04-12 |
| 0.6.0 | 2009-03-30 |
| 0.5.7 | 2009-02-20 |
| 0.5.6 | 2009-01-18 |
| 0.5.5 | 2009-01-10 |
| 0.5.4 | 2009-01-07 |
| 0.5.3 | 2008-12-16 |
| 0.5.2 | 2008-12-01 |
| 0.5.1 | 2008-11-03 |
| 0.4.0 | 2008-10-20 |

# Examples

## Instalación o configuración

La forma más sencilla de instalar Vala es instalar su paquete específico de distribución.

En Ubuntu:

```
sudo apt install valac
```

En Fedora:

```
sudo dnf install vala
```

En arco

```
sudo pacman -S vala
```

En OS X, con Homebrew:

```
brew install vala
```

En Windows, puede obtener un instalador para la última versión aquí .

También puede compilarlo desde fuentes, pero necesitará instalar `pkg-config` , un compilador de C, una biblioteca de C estándar y GLib 2 antes:

```
wget https://download.gnome.org/sources/vala/0.34/vala-0.34.4.tar.xz
tar xvf vala-0.34.4.tar.xz
cd vala-0.34.4
./configure
make
sudo make install
```

## ¡Hola Mundo!

En `foo.vala` :

```
void main (string[] args) {
    stdout.printf ("Hello world!");
}
```

Para compilar la fuente en el binario `foo` :

```
valac foo.vala
```

Para compilar y ejecutar la fuente:

```
vala foo.vala
```

Lea Empezando con vala en línea: https://riptutorial.com/es/vala/topic/9067/empezando-con-vala

# Capítulo 2: Asíncrono y rendimiento

## Introducción

Vala proporciona dos construcciones de sintaxis para tratar operaciones `async` : función `async` y declaración de `yield` .

## Examples

### Declarar una función asíncrona

```
public async int call_async () {
    return 1;
}

call_async.begin ((obj, res) => {
    var ret = call_async.end (res);
});
```

Para llamar a funciones asíncronas desde un contexto síncrono, use el método `begin` y pase una devolución de llamada para recibir el resultado. Los dos argumentos son:

- `obj` es un objeto `GLib.Object` si esta llamada se definió en una clase
- `res` es un `GLib.AsyncResult` contiene el resultado de la operación asíncrona

El método `end` extrae el resultado de la operación.

### Uso de GLib.Task para realizar operaciones asíncronas

`GLib.Task` proporciona una API de bajo nivel para realizar operaciones asíncronas.

```
var task = new GLib.Task (null, null, (obj, result) => {
    try {
        var ret = result.propagate_boolean ();
    } catch (Error err) {
        // handler err...
    }
});
```

Más tarde en un hilo o una devolución de llamada:

```
task.return_boolean (true);
```

Para utilizar el `GLib.Task` subprocesos interno `GLib.Task` :

```
task.run_in_thread (() => {
    task.return_boolean (true);
});
```

### Rendimiento de una función asíncrona

Para poder encadenar operaciones asíncronas y evitar un infierno de devolución de llamada, Vala admite la declaración de `yield` .

Utilizado con una invocación asíncrona, pausará la rutina actual hasta que se complete la llamada y extraiga el resultado.

Si se usa solo, `yield` pausa a la corriente actual hasta que se active al invocar su devolución de llamada de origen.

```
public async int foo_async () {
    yield; // pause the coroutine
    Timeout.add_seconds (5, bar_async.callback); // wakeup in 5 seconds
    return ret + 10;
}

public async int bar_async () {
    var ret = yield foo_async ();
}
```

Lea Asíncrono y rendimiento en línea: https://riptutorial.com/es/vala/topic/9281/asincrono-y-rendimiento

# Capítulo 3: Funciones

## Introducción

Las funciones son piezas de código que pueden ser ejecutadas por otras funciones de su programa.

Tu programa siempre comienza con la función `main` .

Ver también funciones asíncronas .

## Observaciones

Los métodos son exactamente iguales a la función, pero actúan en una instancia de objeto.

## Examples

### Funciones básicas

Una función se define al menos por su tipo de retorno y un nombre único.

```
void say_hello () {
    print ("Hello, world!\n");
}
```

Luego, para llamarlo, simplemente use el nombre de la función seguido de un parentesco.

```
say_hello ();
```

---

Las funciones también pueden tener parámetros entre paréntesis, definidos por sus tipos y nombres y separados por comas. Entonces puedes usarlas como variables normales en tu función.

```
int greet (string name, string family_name) {
    print ("Hello, %s %s!\n", name, family_name);
}
```

Para llamar a una función con parámetros, simplemente coloque una variable o un valor entre los paréntesis.

```
string name = "John";
greet (name, "Doe");
```

---

También puede devolver un valor que se puede asignar a una variable con la palabra clave `return`

.

```
int add (int a, int b) {
    return a + b;
}

int sum = add (24, 18);
```

Toda la ruta del código debe terminar con una declaración de `return` . Por ejemplo, el siguiente código no es válido.

```
int positive_sub (int a, int b) {
    if (a >= b) {
        return a - b;
    } else {
        // Nothing is returned in this case.
        print ("%d\n", b - a);
    }
}
```

## Parámetros opcionales

Los parámetros se pueden marcar como opcionales dándoles un valor predeterminado. Los parámetros opcionales se pueden omitir al llamar a la función.

```
string greet (string name, string language = "English") {
    if (language == "English") {
        return @"Hello, $name!";
    } else {
        return @"Sorry $name, I don't speak $language";
    }
}

greet ("John");
greet ("Jane", "Italian");
```

## Parámetros de salida y referencia

Los tipos de valor (estructuras y enumeraciones) se pasan por valor a las funciones: se dará una copia a la función, no una referencia a la variable. Así que la siguiente función no hará nada.

```
void add_three (int x) {
    x += 3;
}

int a = 39;
add_three (a);
assert (a == 39); // a is still 39
```

Para cambiar este comportamiento puede utilizar la palabra clave `ref` .

```
// Add it to the function declaration
void add_three (ref int x) {
```

```
    x += 3;
}

int a = 39;
add_three (ref a); // And when you call it
assert (a == 42); // It works!
```

`out` funciona de la misma manera, pero se ve obligado a establecer un valor para esta variable antes del final de la función.

```
string content;
FileUtils.get_contents ("file.txt", out content);

// OK even if content was not initialized, because
// we are sure that it got a value in the function above.
print (content);
```

## Programación de contratos

Puede afirmar que los parámetros tienen ciertos valores con los `requires` .

```
int fib (int i) requires (i > 0) {
    if (i == 1) {
        return i;
    } else {
        return fib (i - 1) + fib (i - 2);
    }
}

fib (-1);
```

No recibirá ningún error durante la compilación, pero obtendrá un error al ejecutar su programa y la función no se ejecutará.

También puede afirmar que el valor de retorno coincide con una determinada condición con `ensures`

```
int add (int a, int b) ensures (result >= a && result >= b) {
    return a + b;
}
```

Puede tener tantos `requires` y `ensures` como desee.

## Argumentos variables

```
int sum (int x, ...) {
    int result = x;
    va_list list = va_list ();
    for (int? y = list.arg<int?> (); y != null; y = list.arg<int?> ()) {
        result += y;
    }
    return result;
}
```

```
int a = sum (1, 2, 3, 36);
```

Con esta función, puedes pasar tantos int como quieras. Si pasa algo más, obtendrá un valor inesperado o un error de segmentación.

Lea Funciones en línea: https://riptutorial.com/es/vala/topic/9319/funciones

# Capítulo 4: Las clases

## Introducción

Vala soporta varios sabores de clases.

## Observaciones

Se requieren las dependencias `glib-2.0` y `gobject-2.0` menos que se `--nostdpkg` explícitamente `--nostdpkg` .

## Examples

### Clase de objeto

```
public class Foo : Object {
    public string prop { construct; get; }
}
```

Está destinado a la API interospectable mediante la introspección de GObject. Esta es la forma recomendada para declarar clases.

### Clase sencilla

```
public class Foo {
    public string prop { construct; get; }
}
```

Pure-Vala y clase liviana. Esto es útil si necesita un compromiso entre la eficiencia de una `struct` y la característica de una clase GObject completa.

### Clase compacta

```
[Compact]
public class Foo {
    public string prop;
}
```

Se utiliza principalmente para escribir enlaces con gestión de memoria específica.

Lea Las clases en línea: https://riptutorial.com/es/vala/topic/9076/las-clases

# Capítulo 5: Mesón

## Introducción

Meson es un sistema de compilación de próxima generación diseñado con simplicidad y claridad en mente.

## Examples

### Proyecto basico

```
project('Vala Project')

glib_dep = dependency('glib-2.0')
gobject_dep = dependency('gobject-2.0')

executable('foo', 'foo.vala', dependencies: [glib_dep, gobject_dep])
```

Nota: se requieren las dependencias `glib-2.0` y `gobject-2.0` menos que se `--nostdpkg` explícitamente `--nostdpkg` .

### Proyecto basado en Posix (sin GLib o GObject)

```
project('Posix-based Project', 'vala')

add_project_arguments(['--nostdpkg'], language: 'vala')

posix_dep = meson.get_compiler('vala').find_library('posix')

executable('foo', 'foo.vala', dependencies: [posix_dep])
```

### Fuentes mixtas

```
project('Mixed sources Project', 'vala')

glib_dep = dependency('glib-2.0')
gobject_dep = dependency('gobject-2.0')

executable('foo', 'foo.vala', 'bar.c', dependencies: [glib_dep, gobject_dep])
```

En `foo.vala` :

```
namespace Foo {
    public extern int bar ();

    public int main (string[] args) {
        return bar ();
    }
}
```

En `bar.c` :

```
int
bar ()
{
    return 0;
}
```

Lea Mesón en línea: https://riptutorial.com/es/vala/topic/9074/meson

# Capítulo 6: Propiedad

## Observaciones

Tenga en cuenta que el compilador no le impedirá usar la variable para la cual se transfirió su valor de propiedad.

## Examples

### Transferir propiedad

```
var foo = new uint8[12];
var bar = (owned) foo;
assert (foo == null);
```

La variable de `bar` será propietaria del valor que anteriormente era propiedad de `foo` .

### Copia implícita

```
var foo = new uint8[12];
var bar = foo;
assert (foo != bar);
```

En este ejemplo, tanto `foo` como `bar` poseen una referencia sólida, pero como `uint8[]` solo es compatible con propiedad única, se realiza una copia.

Lea Propiedad en línea: https://riptutorial.com/es/vala/topic/9075/propiedad

# Capítulo 7: Señales

## Examples

### Señal basica

Las señales solo están disponibles para las clases de GObject. Solo pueden ser públicos, lo que significa que cualquier parte del código puede conectar controladores y activarlos.

```
public class Emitter : Object {
    // A signal is declared like a method,
    // but with the signal keyword.
    public signal void my_signal ();

    public void send_signal () {
        this.my_signal (); // Send a signal by calling it like a method.
    }
}

void main () {
    var emitter = new Emitter ();
    // Use the connect method of the signal to add an handler.
    emitter.my_signal.connect (() => {
        print ("Received the signal.\n");
    });
    emitter.send_signal ();
    emitter.my_signal (); // You can send a signal from anywhere.
}
```

También puede usar las funciones normales como manejadores si tienen la misma firma que la señal.

```
void main () {
    var emitter = new Emitter ();
    emitter.connect (my_handler);
    emitter.my_signal ();
}

void my_handler () {
    print ("Received the signal.\n");
}
```

### Señal detallada

Puede escribir señales detalladas con el atributo `[Signal (detailed = true)]` .

```
public class Emitter : Object {
    [Signal (detailed = true)]
    public signal void detailed_signal ();

    public void emit_with_detail (string detail) {
        this.detailed_signal[detail] ();
```

```
    }
}

void main () {
    var emitter = new Emitter ();

    // Connect only when the detail is "foo".
    emitter.detailed_signal["foo"].connect (() => {
       print ("Received the signal with 'foo'.\n");
    });

    // Connect to the signal, whatever is the detail.
    emitter.detailed_signal.connect (() => {
        print ("Received the signal.\n");
    });

    emitter.emit_with_detail ("foo"); // Both handlers will be triggered.
    emitter.emit_with_detail ("bar"); // Only the general handler will be triggered.
}
```

Esta característica se usa a menudo con la señal de `notify` , que tiene cualquier clase basada en `Object` , y que se envía cuando cambia una propiedad. El detalle aquí es el nombre de la propiedad, por lo que puede elegir conectarse a esta señal solo para algunos de ellos.

```
public class Person : Object {
    public string name { get; set; }
    public int age { get; set; }
}

void main () {
    var john = new Person () { name = "John", age = 42 });
    john.notify["age"].connect (() => {
        print ("Happy birthday!");
    });
    john.age++;
}
```

## Controlador por defecto y connect_after

Las señales pueden tener un controlador predeterminado. Todo lo que necesitas hacer es darle un cuerpo cuando lo declaras.

```
public class Emitter : Object {
    public signal void my_signal () {
        print ("Hello from the default handler!\n");
    }
}
```

Este controlador siempre se llamará después de los `connect` . Pero puede usar `connect_after` lugar de `connect` si desea agregar un controlador después del predeterminado.

```
var emitter = new Emitter ();
emitter.my_signal.connect_after (() => {
    print ("After the default handler!\n");
});
```

```
emitter.my_signal ();
```

Lea Señales en línea: https://riptutorial.com/es/vala/topic/9791/senales

# Capítulo 8: Usando Vala en Windows

## Introducción

Este tema se centra en cómo ejecutar valac en Windows.

## Examples

### Utilizando msys2 (64 Bit)

1. Instalar msys2 ( http://www.msys2.org/)
2. Instale los requisitos previos requeridos para Vala

   ```
   pacman -S mingw64/mingw-w64-x86_64-gcc
   pacman -S mingw64/mingw-w64-x86_64-pkg-config
   pacman -S mingw64/mingw-w64-x86_64-vala
   ```

   y todos los paquetes adicionales que su código requiere, es decir,

   ```
   pacman -S mingw64/mingw-w64-x86_64-libgee
   ...
   ```

3. Ejecutar el shell msys2 correcto

   ```
   C:\msys64\mingw64.exe
   ```

4. Compruebe las variables de entorno `MSYSTEM` y `PKG_CONFIG_PATH`

   ```
   $ echo $MSYSTEM
   MINGW64

   $ echo $PKG_CONFIG_PATH
   /mingw64/lib/pkgconfig:/mingw64/share/pkgconfig
   ```

5. Ejecute `valac` como siempre, pero asegúrese de trabajar siempre en el entorno correcto (vea los pasos 3 y 4)

   Por ejemplo, vamos a construir el primer GeeSample aquí:

   ```
   $ valac gee-list.vala --pkg gee-0.8
   ```

Lea Usando Vala en Windows en línea: https://riptutorial.com/es/vala/topic/9899/usando-vala-en-windows

# Capítulo 9: Utilizando GLib.Value

## Examples

### ¿Cómo inicializarlo?

Para inicializar la estructura, puedes hacer esto:

```
public static void main (string[] args) {
    Value val = Value (typeof (int));
    val.set_int (33);
}
```

Pero Vala trae otra forma de inicializar valores:

```
public static void main (string[] args) {
    Value val = 33;
}
```

Su valor se inicializa con el tipo 'int' y contiene '33' valor int.

### Cómo usarlo ?

Use uno de los métodos de obtención de valor de GLib.Value ( consulte la documentación de valadoc ) o emita su valor con el tipo de su valor:

```
public static void main (string[] args) {
    Value val = 33;
    int i = val.get_int();
    int j = (int)val;
}
```

Nota: si su valor actual no contiene el tipo deseado, el sistema GObject arrojará un error crítico:

```
public static void main (string[] args) {
    Value val = 33;
    string s = (string)val;
}
```

```
(process:5725): GLib-GObject-CRITICAL **: g_value_get_string: assertion 'G_VALUE_HOLDS_STRING
(value)' failed
```

### Usar GLib.Value en parámetros de función

Este ejemplo muestra cómo puede pasar varios tipos en parámetros de función:

```
static void print_value (Value val) {
    print ("value-type : %s\n", val.type().name());
```

```
    print ("value-content : %s\n\n", val.strdup_contents());
}

public static void main (string[] args) {
    print_value (33);
    print_value (24.46);
    print_value ("string");
}
```

```
value-type : gint
value-content : 33

value-type : gdouble
value-content : 24.460000

value-type : gchararray
value-content : "string"
```

Nota: si GObject puede transformar su valor con el tipo 'string' (gchararray), 'strdup_contents' devuelve el valor convertido, en lugar de la dirección del puntero

```
static void print_value (Value val) {
    print ("value-type : %s\n", val.type().name());
    print ("value-content : %s\n\n", val.strdup_contents());
}

public static void main (string[] args) {
    print_value (new DateTime.now_local());
}
```

```
value-type : GDateTime
value-content : ((GDateTime*) 0x560337def040)
```

## Tipos de registro para GLib.Value

En el ejemplo anterior, Value.strdup_contents imprime GLib.DateTime como dirección de puntero. Puede registrar funciones que transformarán el valor al tipo deseado. Primero, crea una función que tendrá esta firma:

```
static void datetime_to_string (Value src_value, ref Value dest_value) {
    DateTime dt = (DateTime)src_value;
    dest_value.set_string (dt.to_string());
}
```

luego registre esta función con Value.register_transform_func :

```
Value.register_transform_func (typeof (DateTime), typeof (string), datetime_to_string);
```

ahora GObject puede convertir cualquier objeto DateTime en valor de cadena.

el ejemplo completo:

```
static void datetime_to_string (Value src_value, ref Value dest_value) {
    DateTime dt = (DateTime)src_value;
    dest_value.set_string (dt.to_string());
}

static void print_value (Value val) {
    print ("value-type : %s\n", val.type().name());
    print ("value-content : %s\n\n", val.strdup_contents());
}

public static void main (string[] args) {
    print_value (new DateTime.now_local());
    Value.register_transform_func (typeof (DateTime), typeof (string), datetime_to_string);
    print_value (new DateTime.now_local());
}
```

```
value-type : GDateTime
value-content : ((GDateTime*) 0x560337def040)

value-type : GDateTime
value-content : 2017-04-20T18:40:20+0200
```

Lea Utilizando GLib.Value en línea: https://riptutorial.com/es/vala/topic/9777/utilizando-glib-value

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con vala | Adrià Arrufat, arteymix, avojak, Community, Günther Wutz |
| 2 | Asíncrono y rendimiento | arteymix |
| 3 | Funciones | Community |
| 4 | Las clases | AlThomas, arteymix, Community, Günther Wutz |
| 5 | Mesón | arteymix |
| 6 | Propiedad | arteymix |
| 7 | Señales | Community |
| 8 | Usando Vala en Windows | Jens Mühlenhoff |
| 9 | Utilizando GLib.Value | yannick inizan |