

Informática II

Programación en C bajo GNU/Linux

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2021 –

Programar en lenguaje C

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```



Programar en lenguaje C

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

► ¿Qué representa el texto de arriba? 🖱️

Programar en lenguaje C



```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

- ▶ ¿Qué representa el texto de arriba? 
- ▶ ¿Qué hay que hacer para obtener los sig. en la terminal? 

```
Hola mundo.
```

Programar en lenguaje C

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

- ▶ ¿Qué representa el texto de arriba? 
- ▶ ¿Qué hay que hacer para obtener los sig. en la terminal? 

```
Hola mundo.
```

- ▶ ¿Qué herramienta utilizan?

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software.

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- Resaltado de sintaxis

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida
- ▶ Etc.

IDE – Integrated Development Environment

Entorno de desarrollo integrado

Incluye un conjunto de herramientas informáticas para facilitar el desarrollo de software. Normalmente consiste en:

1. Editor de código fuente
2. Herramientas de construcción
3. Depurador (debugger)

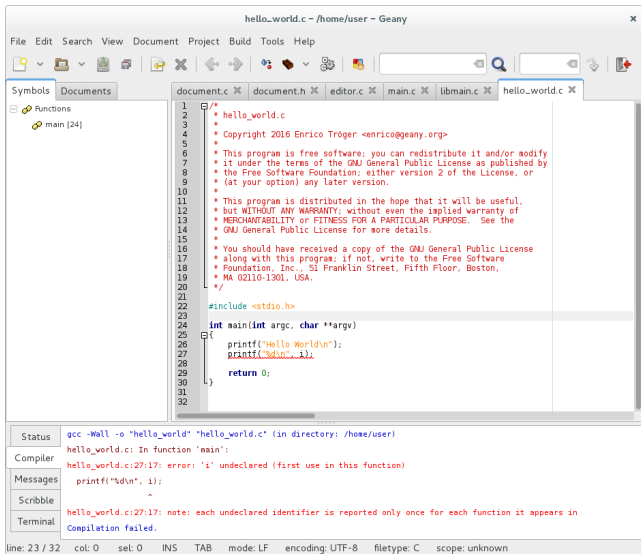
Algunas características de los IDE

- ▶ Resaltado de sintaxis
- ▶ Indentado automático
- ▶ Plegado de código
- ▶ Autocompletado
- ▶ Administración de proyecto
- ▶ Terminal embebida
- ▶ Etc.

¿Cuáles conocen?

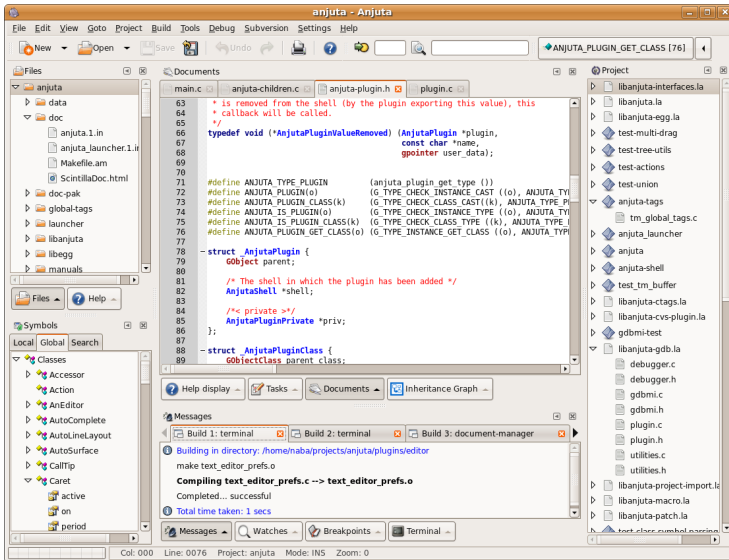
IDE – Integrated Development Environment

Geany



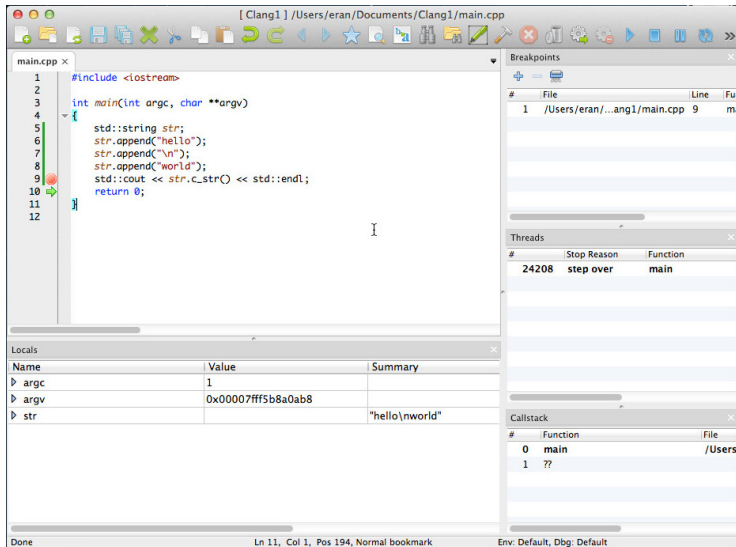
IDE – Integrated Development Environment

Anjuta



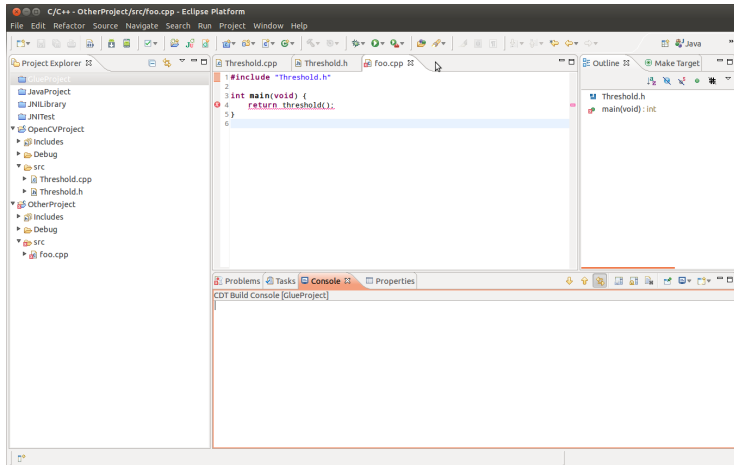
IDE – Integrated Development Environment

CodeLite



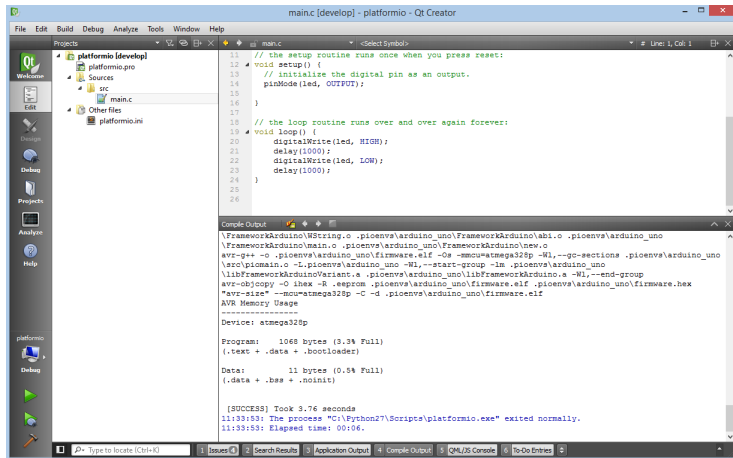
IDE – Integrated Development Environment

Eclipse CDT



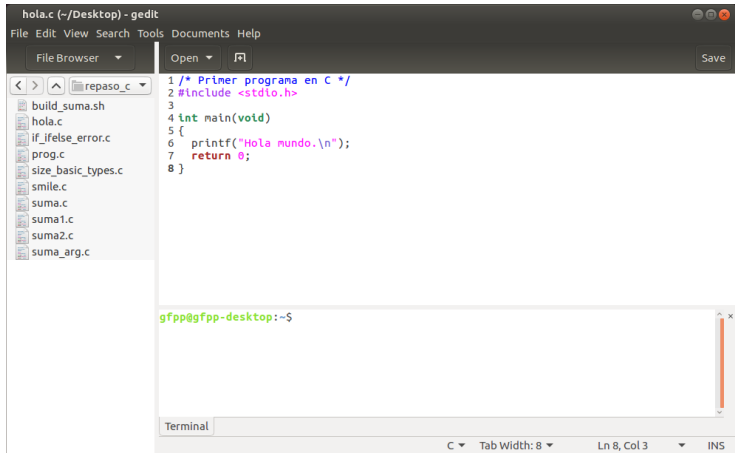
IDE – Integrated Development Environment

Qt Creator

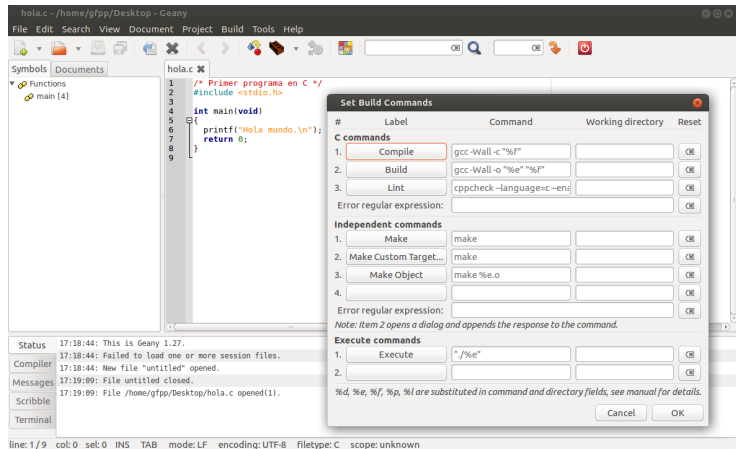


IDE – Integrated Development Environment

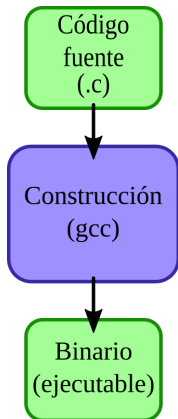
gedit



Ejemplo de configuración del IDE Geany

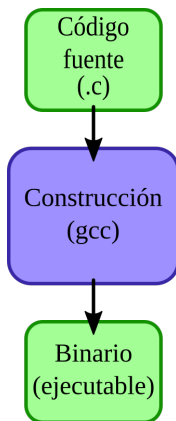


Construcción de un programa



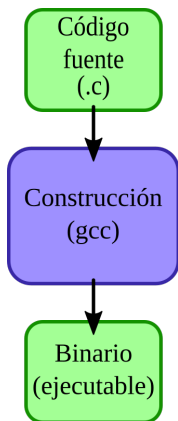
Construcción de un programa

1. Escribir el código fuente



```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

Construcción de un programa

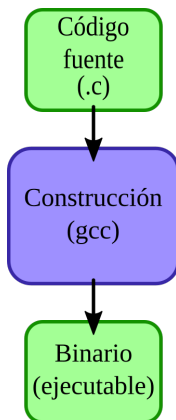


1. Escribir el código fuente

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

Construcción de un programa



1. Escribir el código fuente

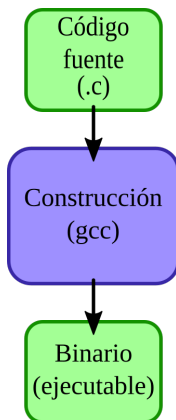
```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

3. Construir el programa

```
> gcc hola.c
```

Construcción de un programa



1. Escribir el código fuente

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

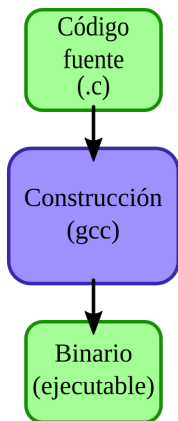
3. Construir el programa

```
> gcc hola.c
```

4. Ejecutar el programa

```
> ./a.out
Hola mundo.
```


Construcción de un programa



1. Escribir el código fuente

```
1 /* Primer programa en C */  
2 #include <stdio.h>  
3  
4 int main(void)  
5 {  
6     printf("Hola mundo.\n");  
7     return 0;  
8 }
```

2. Guardar con extensión .c (nombre sin espacios), p.e. hola.c

3. Construir el programa

```
> gcc hola.c
```

4. Ejecutar el programa

```
> ./a.out  
Hola mundo.
```

En la cátedra se utilizará este procedimiento de compilación

Introducción a gcc

hola.c

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

Introducción a gcc

hola.c

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

Compilación

```
> gcc hola.c
```

Introducción a gcc

hola.c

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

Compilación

```
> gcc hola.c
```

(salida a.out)

Introducción a gcc

hola.c

```
1  /* Primer programa en C */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("Hola mundo.\n");
7      return 0;
8  }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

```
> gcc -Wall hola.c -o hola
```


Introducción a gcc

hola.c

```
1 /* Primer programa en C */
2 #include <stdio.h>
3
4 int main(void)
5 {
6     printf("Hola mundo.\n");
7     return 0;
8 }
```

Compilación

```
> gcc hola.c
```

(salida a.out) Cómo se ejecuta? > ./a.out

Cómo cambiar el nombre a binario?

```
> gcc hola.c -o hola
```

```
> gcc -Wall hola.c -o hola
```

(habilita todas las advertencias/warnings)

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4);
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Error de compilación (-Wall)

```
mal.c:5:10: warning: format '%f'
expects argument of type 'double',
but argument 2 has type 'int'
[-Wformat=]
```

Introducción a gcc

mal.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Dos y dos son %f\n", 4.); // CORREGIDO
6     return 0;
7 }
```

¿Qué se imprime al ejecutar el programa?

Compilar y ejecutar

```
> gcc mal.c -o mal
> ./mal
Dos y dos son 0.000000
```

Error de compilación (-Wall)

```
mal.c:5:10: warning: format '%f'
expects argument of type 'double',
but argument 2 has type 'int'
[-Wformat=]
```

Introducción a gcc

wall1.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a = 2, b = 3;
6     printf("Dos más tres son %d\n", 5);
7     return 0;
8 }
```

Introducción a gcc

wall1.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a = 2, b = 3;
6     printf("Dos más tres son %d\n", 5);
7     return 0;
8 }
```

wall2.c

```
1 int main(void)
2 {
3     int x;
4     x == 2 + 2;
5
6     x = 3;
7
8     if(x = 4)
9         x = 6;
10    return x;
11 }
```

Introducción a gcc

- ▶ Si el código fuente presenta errores de sintaxis el compilador (C o C++) reportará los errores y no producirá el archivo binario de salida.

Introducción a gcc

- ▶ Si el código fuente presenta errores de sintaxis el compilador (C o C++) reportará los errores y no producirá el archivo binario de salida.
- ▶ Los compiladores tiene la capacidad de advertirnos sobre situaciones inesperadas en el código fuente.

Introducción a gcc

- ▶ Si el código fuente presenta errores de sintaxis el compilador (C o C++) reportará los errores y no producirá el archivo binario de salida.
- ▶ Los compiladores tiene la capacidad de advertirnos sobre situaciones inesperadas en el código fuente.
- ▶ Se puede modificar este comportamiento utilizando *flags* de compilación.

Introducción a gcc

- ▶ Si el código fuente presenta errores de sintaxis el compilador (C o C++) reportará los errores y no producirá el archivo binario de salida.
- ▶ Los compiladores tiene la capacidad de advertirnos sobre situaciones inesperadas en el código fuente.
- ▶ Se puede modificar este comportamiento utilizando *flags* de compilación.
- ▶ El compilador puede considerarse como una herramienta útil para verificar el código fuente y aprender.

Introducción a gcc

- ▶ Si el código fuente presenta errores de sintaxis el compilador (C o C++) reportará los errores y no producirá el archivo binario de salida.
- ▶ Los compiladores tiene la capacidad de advertirnos sobre situaciones inesperadas en el código fuente.
- ▶ Se puede modificar este comportamiento utilizando *flags* de compilación.
- ▶ El compilador puede considerarse como una herramienta útil para verificar el código fuente y aprender.
- ▶ Los *flags* (banderas) del compilador permiten cambiar o alterar el comportamiento del compilador.

