### Ejercicio 1



Termina con el mismo valor numérico.

Aritmética: int

El operador if trabaja con int por lo que el casteo de la variable es innecesario, produciendo los mismos resultados.



Terminan con distinto valor. x = 5/3 y x = 1.0.

Aritmética 1: float Artimética 2: int

En el primer caso, ‘y’ se castea a float y luego se realiza la divisón, porque uno de los operandos es float.

En el segundo caso, primero realiza la división de dos enteros, y el resultado se castea a float.



Terminan con distinto valor. x = 5/3 y x = 1.

Aritmética 1: float Aritmética 2: float y luego se castea a int.

En el primer caso, ‘y’ se castea float y luego se realiza la división, porque uno de los operandos es float. Se almacena en float.

En el segundo caso, la operación se realiza en float, pero se guarda en un tipo int.



Terminan con el mismo valor numérico. x = 1.

Aritmética: float y se almacena en variable tipo int.

Las operaciones se realizan en float pero luego el resultado se almacena en un tipo int.



Terminan con el mismo valor numérico. x = 5.

Aritmética: double.

Las operaciones se realizan en double. En el primer caso se almacena en variable tipo int y en el segundo caso se almacena en variable tipo float.



Terminan con el mismo valor numérico. x = 6.

Aritmética: int.

Se convierten los valores a int al ser almacenados en x (que es int) y luego se opera en int y se almacena en int.



Terminan con el mismo valor numérico. x = 40 000.

Aritmética: int.

En el primer caso se almacena en int y en el segundo se almacena en unsigned int.



Terminan con el mismo valor numérico. c = 120.

Aritmética: char.

En el primer caso se almacena en char, y en el segundo en unsigned char.



En el primer caso puede ser 200 o un valor negativo dependiendo de si el compilador toma por defecto signed o unsigned. En el segundo caso el valor es 200.

Aritmética: int.

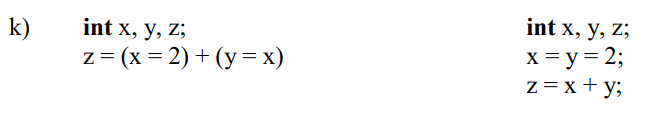


El resultado numérico es diferente.

Aritmética: int.

En el primer caso, al ser un char agarra el último byte y lo asigna a x por lo que perdemos información.

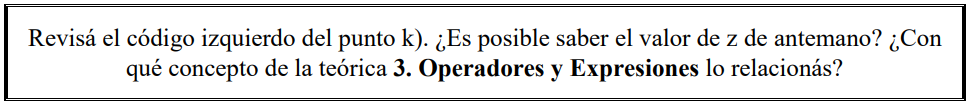
En el segundo caso, se almacena 300 en un tipo int y funciona de acuerdo a lo deseado.



En el primer caso no se puede determinar un valor debido a que depende del orden en el que el compilador haya realizado las asignaciones que están entre los paréntesis.

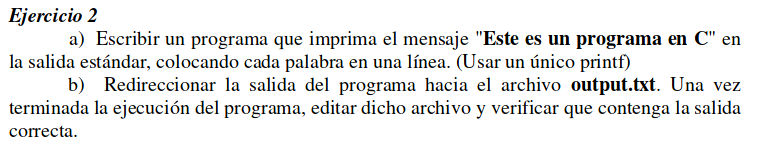
En el segundo caso, solo se suman los valores asignados anteriormente por lo que su resultado es 4.

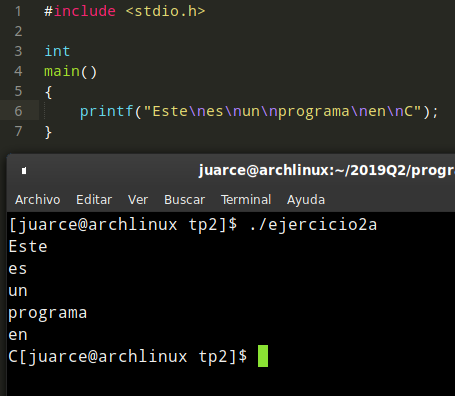
Aritmética: int

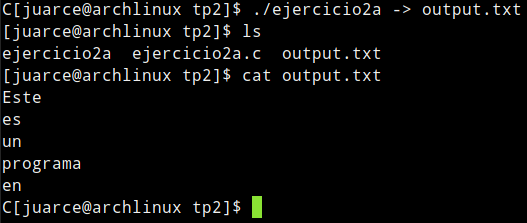


No es posible obtener el valor de z de antemano porque el compilador no especifica el orden en que se realizan las asignaciones.

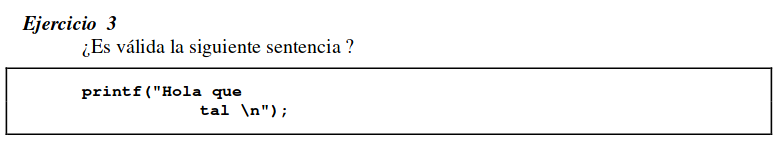
### Ejercicio 2





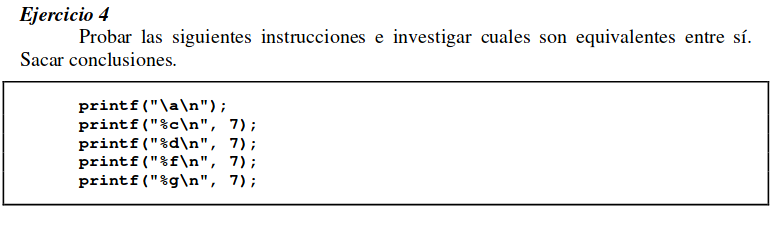


### Ejercicio 3



No es válida, se debe escribir toda la cadena en una sola línea.

### Ejercicio 4



printf("\a\n"); Imprime un beep a salida estándar. Es un sonido.

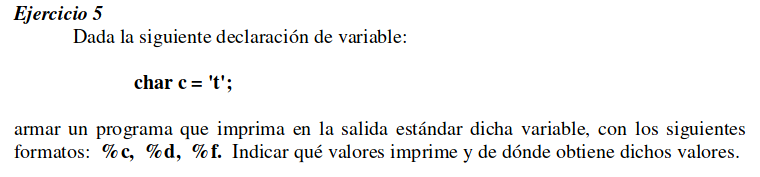
printf("%c\n", 7); Imprime un beep a salida estándar. Es un sonido.

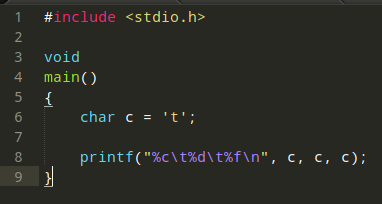
printf("%d\n", 7); Imprime 7.

printf("%f\n", 7); Imprime 0 porque busca en el registro de double.

printf("%g\n", 7); Imprime 0 porque busca en el registro de float.

### Ejercicio 5



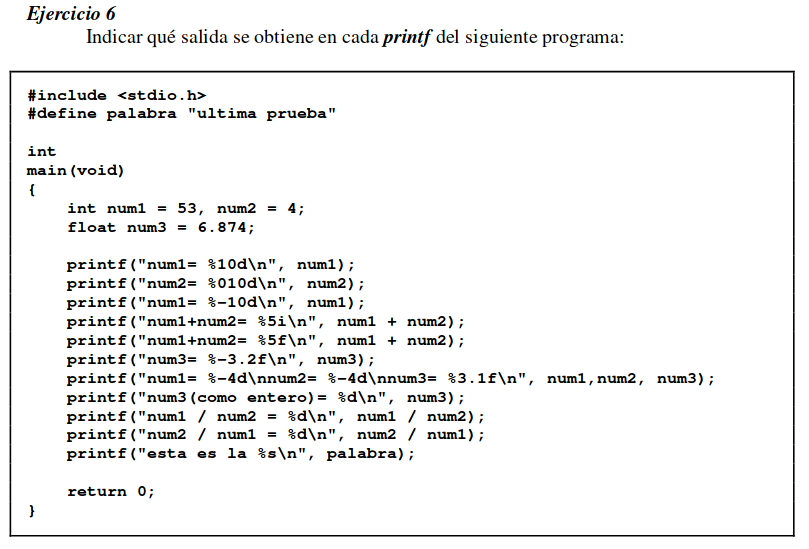


%c Imprime el carácter

%d Imprime el valor ascii

%f Imprime cero porque busca en el registro de float.

### Ejercicio 6



%10d\n Completa con blancos a la izquierda. En este caso 10 en total (entre el número y los espacios.

%010d\n Completa con ceros a la izquierda hasta completar los 10 dígitos.

%-10d\n El – alinea el número al lado izquierdo y deja 10 espacios en blanco.

%5i\n Completa con blancos a la izquierda. La letra i funciona igual que d.

%5f\n Completa con blancos a la izquierda. Muestra cero porque f indica que debe entregar un doble, pero en el registro no hay nada ya que la suma se almacenó en el registro de int.

%-3.2f\n Alineado a la izquierda. Muestra dos decimales.

%-4d\n Alineado a la izquierda. Completa con blancos a la derecha.

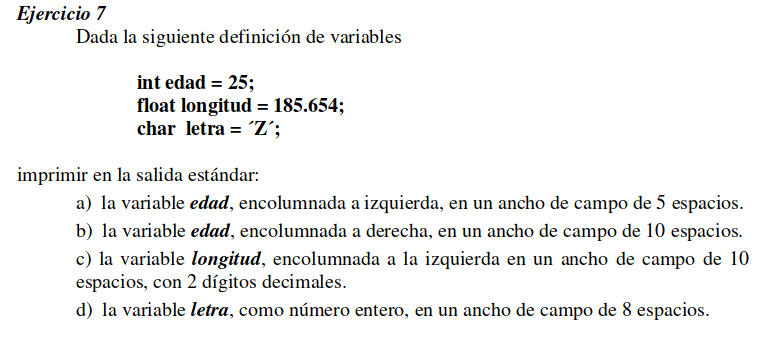
%3.1f\n Muestra un solo decimal redondeado.

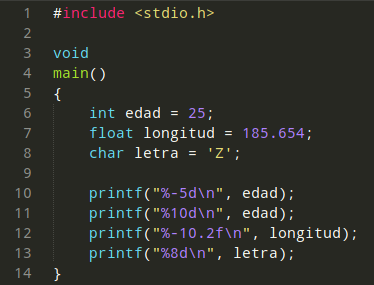
%d\n Muestra el número float como entero en base 10. Muestra cualquier valor.

%d\n Muestra la división de los enteros como entero en base 10 truncado.

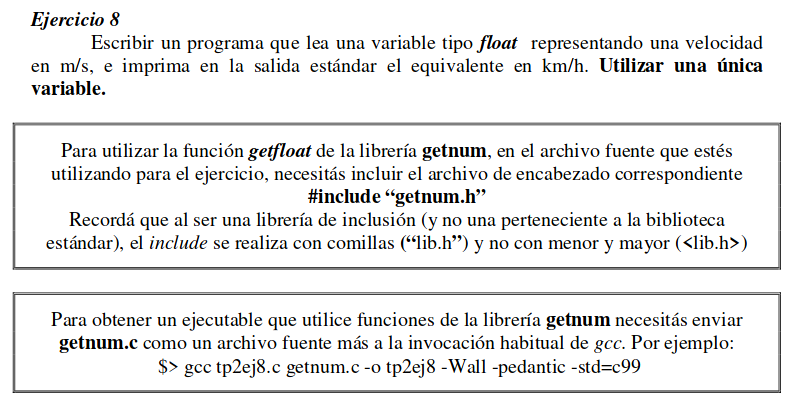
%s Muestra la cadena de caracteres.

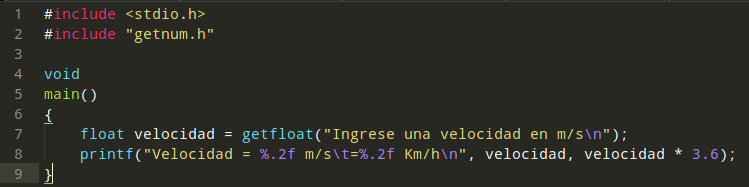
### Ejercicio 7



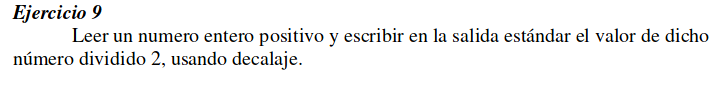


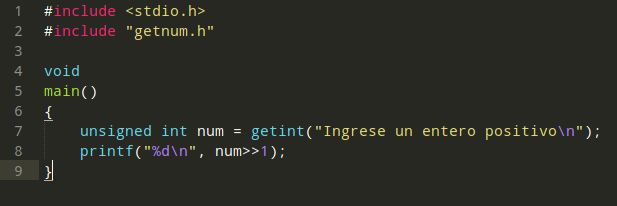
### Ejercicio 8



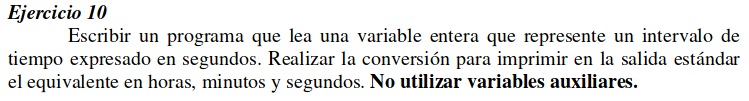


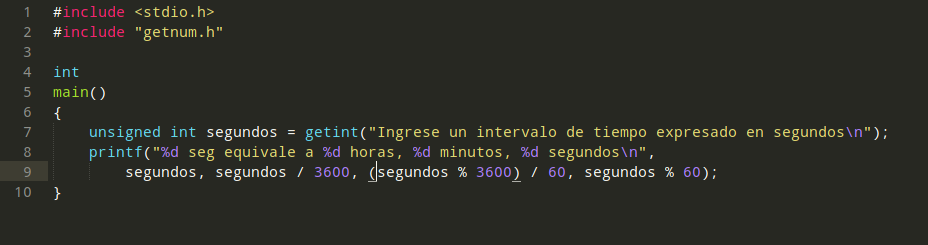
### Ejercicio 9



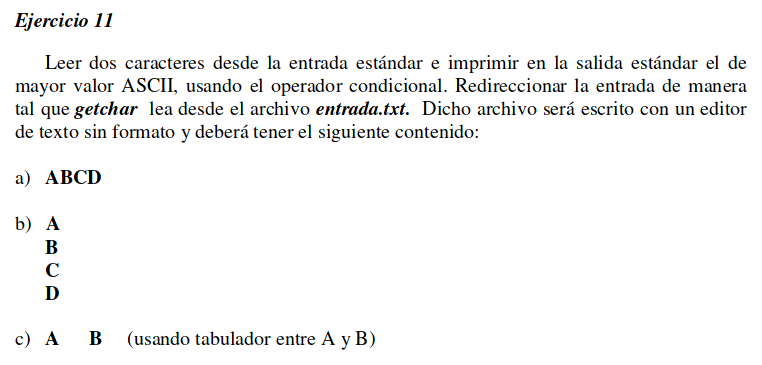


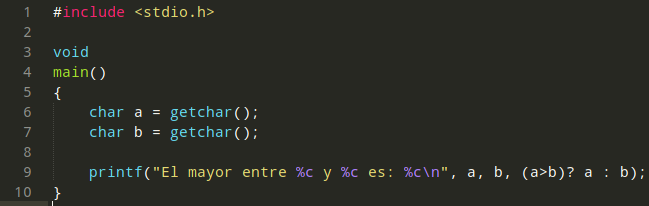
### Ejercicio 10

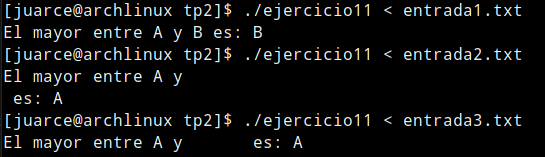




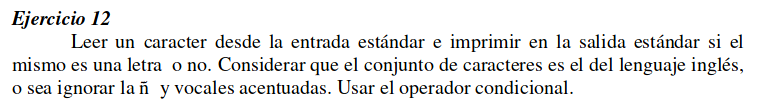
### Ejercicio 11

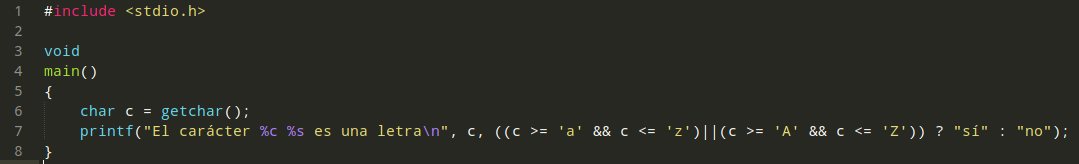




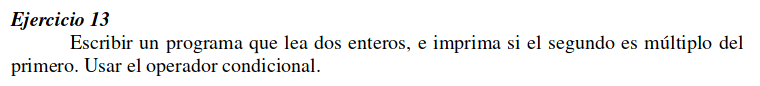


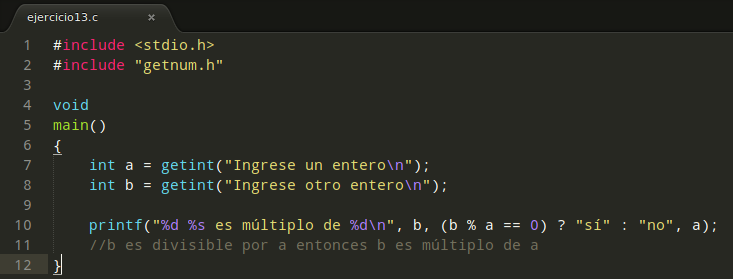
### Ejercicio 12



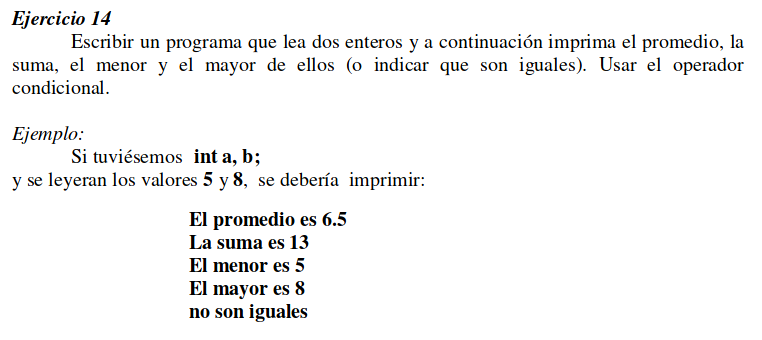


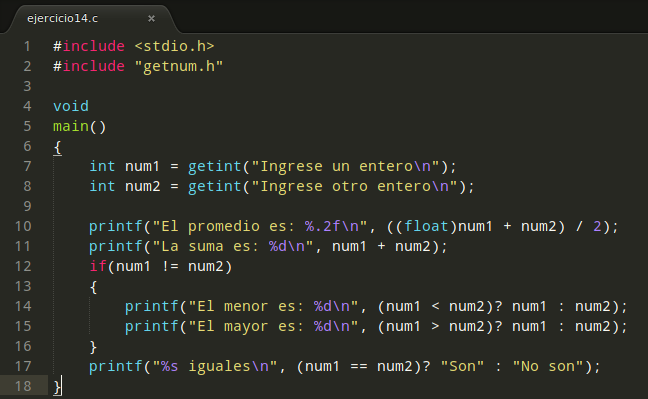
### Ejercicio 13

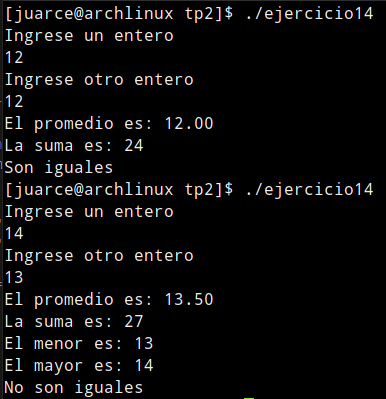




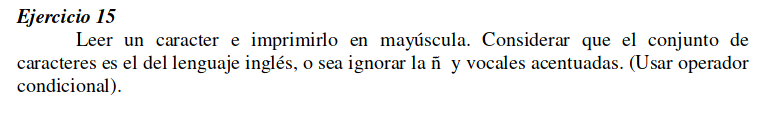
### Ejercicio 14

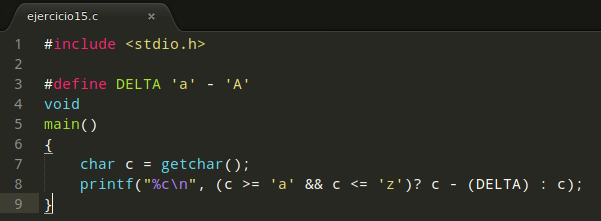




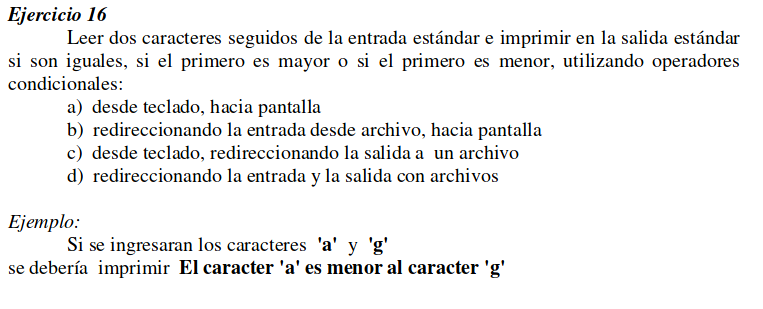


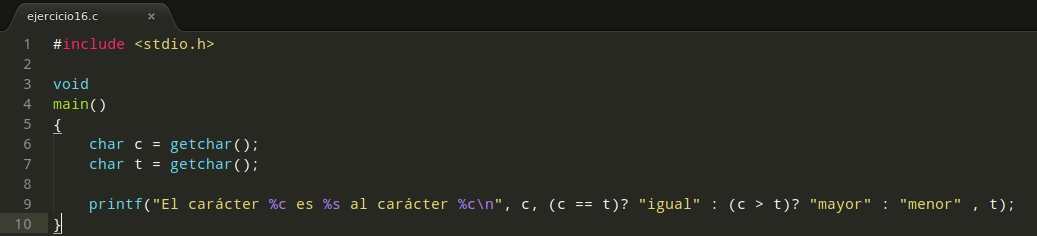
### Ejercicio 15



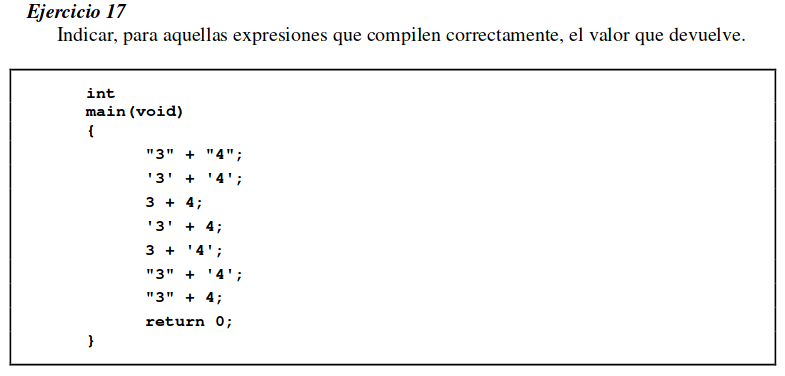


### Ejercicio 16





### Ejercicio 17



**‘Número‘** Devuelve el valor ASCII

**“Número“** String. No usa el valor numérico

**Número** Usa el valor numérico correspondiente

“3” + “4” No compila.

‘3’ + ‘4’ Devuelve la suma del ASCII 3 y 4.

3 + 4 Devuelve 7

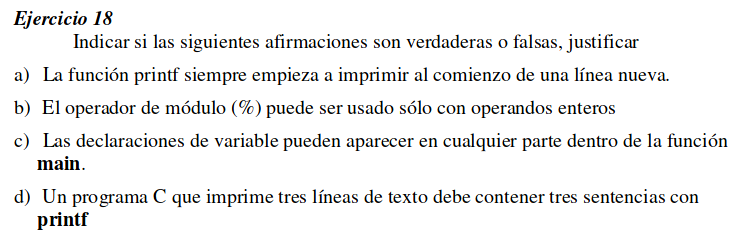
‘3’ + 4 Al ASCII del 3 le suma 4 “posiciones”

3 + ‘4’ Al ASCII del 4 le suma 4 “posiciones”

“3” + ‘4’ No compila

“3” + 4 No compila

### Ejercicio 18



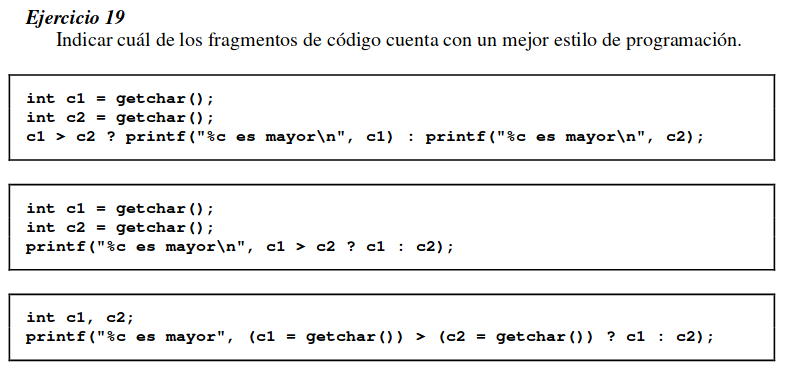
Falso. Para que imprima en una nueva línea se debe colocar \n.

Verdadero. Debido a que el operador % calcula el resto de una división entera.

Falso. Las declaraciones de variables deben estar antes o en el momento de ser invocadas para realizar alguna operación.

Falso. Con un solo printf se pueden imprimir tres líneas de texto, a través de \n.

### Ejercicio 19

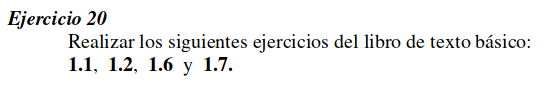


El fragmento que cuenta con el mejor estilo de programación es el segundo.

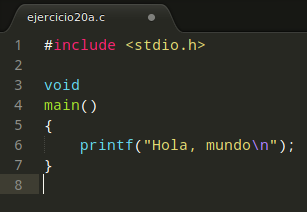
Se realizan las asignaciones directamente en la declaración de las variables.

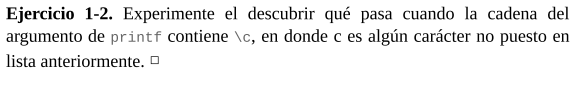
En el condicional solo aparece la condición y los retornos haciendo más legible y entendible esa línea.

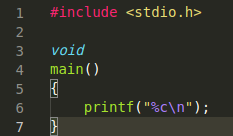
### Ejercicio 20

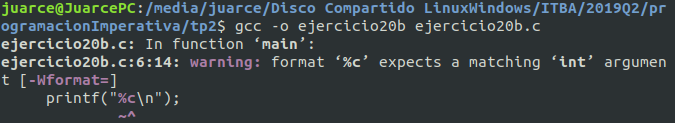




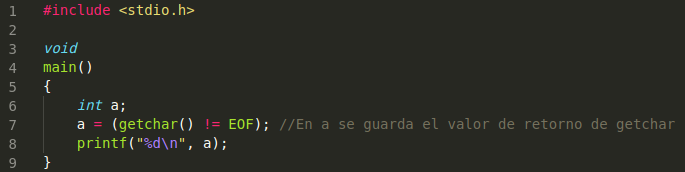






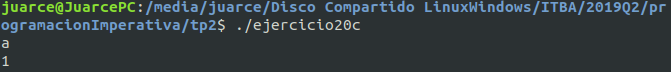






En a se guarda el valor de retorno de la comparación de getchar() con EOF. Este es 1 si getchar obtiene un carácter distinto de EOF. 0 en caso contrario. A continuación se muestra un ejemplo.

Caso a = 1



En este caso, getchar obtuvo la letra ‘a’ por entrada estándar, por lo que es distinto de EOF y devuelve un 1.

Caso a = EOF.



Para introducir EOF por entrada estándar, redirecciono la entrada estándar a un texto vacío, por lo que el getchar encuentra EOF. Esto produce que la variable a valga 0.



