Introducción al lenguaje de programación C para programadores de Python

Rodrigo Alexander Castro Campos
https://racc.mx
https://omegaup.com/profile/rcc

Última revisión: 16 de julio de 2022

1. La filosofía detrás de los lenguajes de programación C y Python

Los lenguajes de programación C y Python son, al día de hoy, los dos lenguajes de programación más populares del mundo [1]. Sin embargo, la filosofía detrás de los mismos es diametralmente distinta. Por su parte, el lenguaje C fue diseñado en 1972 por Dennis Ritchie para facilitar la implementación de sistemas operativos; es decir, C fue diseñado para programar aplicaciones que deban ser de alto rendimiento o que interactúen directamente con el hardware. En cuanto a Python, éste fue diseñado y lanzado en 1991 por Guido van Rossum, con un énfasis es su simplicidad y facilidad de uso. En ese sentido, la creciente popularidad de Python es fácil de explicar: Python es amigable para aquellas personas que apenas comienzan a programar, o bien, para aquellas personas que no tienen formación de ingenieros pero que necesitan programar de vez en cuando. Ciertamente, Python es mejor opción que C para programas cortos que se limiten a usar bibliotecas escritas por otros (muchas veces llamados programas de *scripting*). Sin embargo, al día de hoy *todo programa termina ejecutando código originalmente escrito en C o C++* (siendo el último un lenguaje derivado de C), ya sea de una u otra forma. A continuación se listan algunas aplicaciones que están escritas principalmente en C y C++.

- Todos los sistemas operativos modernos (UNIX, Linux, Windows, Android, IOS, macOS).
- Los compiladores o intérpretes de los lenguajes de programación C, C++, Java, Python, PHP, etc.
- Los navegadores de internet Chrome, Firefox, Safari, Edge y Opera.
- Los buscadores de internet Google y Bing.
- Los motores de bases de datos como MySQL, SQL Server y Oracle Database.
- Casi la totalidad de los videojuegos de consolas Xbox, Playstation y consolas de Nintendo.
- Editores de audio, video e imágenes como Photoshop, Blender y Audacity.
- Controladores de dispositivos de hardware, desde tarjetas de video hasta el vehículo Falcon 9 de SpaceX.
- Mucha de la infraestructura de las redes de telecomunicaciones y energía eléctrica.

El lenguaje C ofrece simultáneamente 1) madurez, al estar estandarizados ante la Organización Internacional de Normalización desde 1989; 2) tiempos de ejecución cortos y consumo de memoria mínimo, al ofrecer una contraparte de alto nivel de lo que el hardware real puede hacer; 3) ubicuidad, al estar disponible en toda plataforma de cómputo. En otras palabras, el lenguaje C constituye el soporte tecnológico para prácticamente todo lo demás. Es por esto que la formación de un ingeniero en computación o en electrónica debe incluir *necesariamente* la enseñanza del lenguaje C en su plan de estudios.

Cambios recientes y sumamente desafortunados en los planes de estudio de la UAM Azcapotzalco abrieron la posibilidad de que los estudiantes de las ingenierías mencionadas anteriormente aprueben el

[1] https://www.tiobe.com/tiobe-index/

curso de "Programación Estructurada" sin haber aprendido C. Esto es un error, porque aunque al resto de las ingenierías les beneficie más ver Python, el curso de "Programación Estructurada" es común para todos; los alumnos no están separados por ingeniería. Peor aún, el curso de programación que sigue debe impartirse en C, C++, Java o C#, según el programa de estudios [2]. Los últimos tres lenguajes son derivados de C, así que es viable que un alumno que aprendió C los entienda sin demasiados inconvenientes. Por su parte, el alumno que aprendió Python en "Programación Estructurada" está en una franca desventaja.

Este documento presenta una introducción breve al lenguaje C para programadores que aprendieron Python y los códigos de ejemplo se pueden ejecutar en `http://callix.azc.uam.mx/rcc/ide/`. El lenguaje C *es* más difícil de usar que Python, en parte por cuestiones históricas y en parte porque C no intenta ocultar demasiado el hardware sobre el cual se ejecuta el programa. Por otra parte, C también es más flexible que Python en muchas situaciones y además consume hasta 100 veces menos recursos (tiempo y memoria) que Python. Más importante que ser más eficiente que Python, es que el rendimiento de C *es* el rendimiento del hardware; no hay factores externos artificiales.

2. Variables, funciones y comentarios

Los códigos escritos en C y Python son visualmente muy distintos e incluso difieren en las formas de escribir comentarios de una o múltiples líneas. Para comentarios de una línea, Python usa `#` y C usa `//`. Para comentarios multilínea, Python usa literales de cadena multilínea delimitadas por `'''` y C usa comentarios de bloque que comienzan en `/*` y terminan en `*/`. Por ejemplo:

Código en Python	Código en C

`# comentario 1`
`# comentario 2`
`'''comentario`
`  comentario`
`  comentario'''`
	`// comentario 1`
	`// comentario 2`
	`/* comentario`
	`  comentario`
	`  comentario */`

En Python, una función se declara con la palabra reservada `def` seguida del nombre de la función, la lista de parámetros y un `:` como puntuación, además de que el código de la función debe aparecer correctamente indentado. Los parámetros de una función de Python sólo nombran identificadores y la función puede recibir o regresar valores de cualquier tipo. En C, una función debe especificar explícitamente el tipo del valor que se regresará (antes del nombre de la función) y los tipos de sus parámetros. Además, en C la indentación es irrelevante porque el cuerpo de la función se delimita entre llaves. Dos palabras reservadas de C que especifican tipos de dato son `int` para enteros y `float` para reales. En C se usan terminadores explícitos (como ; en un return) porque los saltos de línea son irrelevantes.

Código en Python	Código en C

`def suma(a, b):`
`  return a + b`
	`int suma(int a, int b) {`
	`  return a + b;`
	`}`

El siguiente código, aunque nada estético y no recomendado, es una forma válida de escribir el código en C del ejemplo anterior. El tener que usar { } y ; da flexibilidad en cuanto al espaciado en blanco.

Código en C

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

[2] https://dcbi.azc.uam.mx/media/Licenciaturas/Computacion/PlanesProgEstudio/TBP/1151042.pdf

El lenguaje C debe usar la palabra reservada `void` para especificar que una función no regresa un valor.

Código en Python	Código en C

`def nada( ):`
`  return`
	`void nada( ) {`
	`  return;`
	`}`

En Python, una variable puede cambiar su tipo con tan sólo reasignarle otro valor. En C una variable no cambia su tipo y éste se debe especificar en su primer uso (pero no en usos posteriores).

Código en Python	Código en C

`def f(a, b):`
`  c = a + b`
`  c = 5 - c`
`  return c`
	`int f(int a, int b) {`
	`  int c = a + b;`
	`  c = 5 - c;`
	`  return c;`
	`}`

En Python, el uso cómodo de las utilidades de la biblioteca requieren el uso de sentencias `import`. Por su parte, la biblioteca estándar de C está dividida en archivos que deben incluirse con la notación `#include <archivo>` y es necesario hacer una inclusión para poder leer o imprimir valores. Las rutinas de lectura y escritura de la biblioteca de C están disponibles en el archivo `stdio.h`.

Código en Python	Código en C

`def saluda( ):`
`  print("hola")`
`  print("adios")`
	`#include <stdio.h>`
	
	`void saluda( ) {`
	`  puts("hola");`
	`  puts("adios");`
	`}`

En Python, la construcción del lenguaje `print` agrega un salto de línea implícito, al igual que lo hace `puts` de C. Para evitarlo, Python usa la función `sys.stdout.write` del módulo `sys` (el cual debe importarse) mientras que C usa la función `printf`. En ambos lenguajes, `\n` es un salto de línea explícito. Los siguientes ejemplos imprimirán `1234` con un salto de línea al final.

Código en Python	Código en C

`import sys`
 	
`def f( ):`
`  sys.stdout.write("12")`
`  sys.stdout.write("34\n")`
	`#include <stdio.h>`
	
	`void f( ) {`
	`  printf("12");`
	`  printf("34\n");`
	`}`

La ejecución de un programa en Python comienza en el ámbito global, mientras que la ejecución de un programa en C comienza a partir de la función `main`. La función `main` debe regresar un entero (generalmente con valor 0). Al igual que en Python, es un error llamar a una función antes de declararla.

Código en Python	Código en C

`def saluda( ):`
`  print("hola")`
 	
`saluda( )`
	`#include <stdio.h>`
	
	`void saluda( ) {`
	`  puts("hola");`
	`}`
	
	`int main( ) {`
	`  saluda( );`
	`  return 0;`
	`}`

3. Aritmética, lectura y escritura de variables

Los operadores aritméticos (`+`, `-`, `*`, `/` y `%`) de Python y C son similares, siendo la división la única diferencia notable. En Python 3, el operador `/` realiza la división real y `5 / 2` es 2.5, mientras que en Python 2 y en C el resultado se trunca cuando ambos operandos son enteros y `5 / 2` es 2. Python 3 dispone de un operador específico para la división con truncamiento, el cual no existe en C. Además, guardar un real en una variable entera truncará la parte decimal.

> A partir de este momento y salvo que se especifique lo contrario, los ejemplos en C deben escribirse dentro de `main` y deben incluirse los archivos de biblioteca necesarios (como `stdio.h`) para ejecutar el programa.

Código en Python

```
a = 5
b = 2
x = 3.14

t1 = a + b    # 7
t2 = a + x    # 8.14
t3 = a / b    # 2.5
t4 = a / x    # 1.59
```

Código en C

```
int a = 5;
int b = 2;
float x = 3.14;

// con enteros
int i1 = a + b;    // 7
int i2 = a + x     // 8
int i3 = a / b;    // 2
int i4 = a / x;    // 1

// con reales
float r1 = a + b;  // 7
float r2 = a + x;  // 8.14
float r3 = a / b;  // 2 (operandos enteros)
float r4 = a / x;  // 1.59
```

Los moldeados o *casts* también existen en C, pero el tipo destino se escribe dentro de paréntesis. Por otra parte, C no tiene un operador de potencia como `**` de Python, pero existe la función `pow` de `math.h`.

Código en Python

```
a = int(8 ** 0.5)    # 2
```

Código en C

```
float a = (int)pow(8, 0.5);    // 2
```

Tanto en Python como en C, es posible declarar o asignar más de una variable en la misma línea de código, pero la notación difiere. Al declarar varias variables en la misma línea, el lenguaje C usa el mismo tipo base para ambas. Más allá de eso, ambos lenguajes comparten las asignaciones compuestas, tales como `+=`, `-=`, etcétera. En C también se usan los operadores `++` y `--` para sumar o restar 1.

Código en Python

```
a, b = 5, 2
a, b = 6, 3
a += 1
```

Código en C

```
int a = 5, b = 2;
a = 6, b = 3;
a += 1;    // alternativa: a++;
```

En Python existen muchas formas de imprimir variables: enviar varios argumentos a `print`, concatenar variables en una cadena, o usar valores interpolados en una literal de cadena. En C, la función `printf` es muy similar a la última opción mencionada para Python. En ambos lenguajes, los especificadores de formato `%d` y `%f` permiten imprimir enteros y reales, respectivamente.

Código en Python

```
a = 5
x = 3.14

print(a, x)
print(str(a) + " " + str(x))
print("%d %f" % (a, x))
```

Código en C

```
int a = 5;
float x = 3.14;

printf("%d %f\n", a, x);
```

La lectura de variables en C se hace con la función `scanf` y se usa la misma idea, excepto que se debe usar el operador `&` prefijo sobre las variables para que `scanf` pueda escribir en ellas. Una ventaja de `scanf` es que la entrada está orientada a *tokens* y no a líneas, como en Python. Por esta razón, el espaciado de la entrada es irrelevante. Las variables se deben declarar antes de leerlas, pero no es necesario inicializarlas.

Código en C	Entrada	Salida

`int a;`
`float x;`
`scanf("%d%f", &a, &x);`
`printf("%d %f, a, x);`
	`5`
	
	`3.14`
		`5 3.14`

A diferencia de Python, el rango de valores que se puede representar en los tipos enteros del lenguaje C es limitado. Por ejemplo, el tipo `int` generalmente sólo puede representar enteros del -2147483648 al 2147483647. El lenguaje C disponible de un tipo entero `long long` que tiene un rango mucho mayor, pero finito de todos modos. El tipo `long long` debe leerse e imprimirse con el especificador de formato `%lld`.

4. Condiciones y sentencias de decisión

Tanto Python como C permiten evaluar condiciones y decidir qué acción tomar dependiendo de su resultado. Lo anterior es posible tanto a nivel de expresión como a nivel de sentencia en ambos lenguajes. En principio, los lenguajes comparten los operadores relacionales `<`, `<=`, `>`, `>=`, `==` y `!=`. A su vez, las expresiones que permiten elegir uno u otro valor dependiendo del valor de una condición se llaman expresiones condicionales en Python y expresiones ternarias en C. La principal diferencia es su notación.

Código en Python	Código en C

`a, b = 5, 2`
`c = 10 if a < b else -8`
`print(c)  # -8`
	`int a = 5, b = 2;`
	`int c = (a < b ? 10 : -8);`
	`printf("%d\n", c);  // -8`

Las sentencias `if` y `else` también están disponibles en ambos lenguajes. Mientras que Python usa la indentación para determinar el bloque de código que depende del `if`, el lenguaje C usa llaves. En C, la condición debe aparecer entre paréntesis y no se usa `:` como puntuación. La otra diferencia importante es que Python usa `elif` como contracción de `else if`, pero dicha contracción no existe en C.

Código en Python	Código en C

`a, b = 5, 2`
`if a < b:`
`  print("a es menor")`
`elif a > b:`
`  print("a es mayor")`
`else`
`  print("a es igual")`
	`int a = 5, b = 2;`
	`if (a < b) {`
	`  puts("a es menor");`
	`} else if (a > b) {`
	`  puts("a es mayor");`
	`} else {`
	`  puts("a es igual");`
	`}`

Los operadores lógicos `and`, `or` y `not` también existen en C y se pueden escribir tal cual si se incluye el archivo de biblioteca `iso646.h`. Sin embargo, es mucho más común escribirlos en C como `&&`, `||` y `!`, respectivamente, lo que además no necesita ninguna inclusión.

Código en Python	Código en C

`a = 6`
`if a > 5 and a % 2 == 0:`
`  print("a es mayor que 5 y par")`
	`int a = 6;`
	`if (a > 5 && a % 2 == 0) {`
	`  printf("a es mayor que 5 y par");`
	`}`

Las comparaciones encadenadas de Python, tales como `a < b < c`, desafortunadamente no funcionan correctamente en C, a pesar de que el lenguaje aparentemente las acepta sin problemas. En C, dichas expresiones deben escribirse como comparaciones individuales conectadas por operadores lógicos.

Código en Python	Código en C

`a = 5`
`if 2 < a < 7:`
`  print("a está entre 2 y 7")`
	`int a = 5;`
	`if (2 < a && a < 7) {`
	`  printf("a está entre 2 y 7");`
	`}`

5. Sentencias de repetición

La sentencia `while` de Python es prácticamente idéntica a la del lenguaje C, salvo por las diferencias usuales en la notación. Incluso están disponibles las sentencias `break` y `continue` para romper el ciclo completo o sólo la iteración actual, respectivamente.

Código en Python	Código en C

`a = 0`
`while a < 10:`
`  a += 1`
`  if a % 5 == 0:`
`  continue`
`  elif a == 7:`
`  break`
`  print(a)`
	`int a = 0;`
	`while (a < 10) {`
	`  a += 1;`
	`  if (a % 5 == 0) {`
	`  continue;`
	`  } else if (a == 7) {`
	`  break;`
	`  }`
	`  printf("%d\n", a);`
	`}`

Desafortunadamente, las similitudes en los ciclos de Python y C terminan ahí. Python no provee un tipo de ciclo llamado comúnmente `do while`, el cual existe en C y es similar al `while` pero donde el bloque se ejecuta al menos una vez porque la condición se evalúa al final del bloque y no al inicio.

Código en C	Salida

`int n = 0;`
`do {`
`  printf("%d\n", n);`
`  n += 1;`
`} while (n > 5);`
	`0`

El ciclo `for` de Python está basado en generadores, los cuales no existen en C. En C, el ciclo `for` es principalmente una forma abreviada de escribir un ciclo `while` que tiene inicialización, condición, actualización y cuerpo. El siguiente código imprime los enteros del 0 al 4 en ambos lenguajes.

Código en Python	Código en C

`for i in range(0, 5):`
`  print(i)`
	`for (int i = 0; i < 5; i++) {`
	`  printf("%d\n", i);`
	`}`
	
	`/* int i = 0;`
	`  while (i < 5) {`
	`  printf("%d\n", i);`
	`  i += i;`
	`  }`
	`*/`

En ambos lenguajes, un `continue` dentro de un `for` aplica la actualización sobre la variable controladora del ciclo. Además, C permite controlar un ciclo `for` con más de una variable.

Código en C	Salida

`for (int i = 0, j = 0; i < 5; i++, j--) {`
`  printf("%d %d\n", i, j);`
`}`
	`0 0`
	`1 -1`
	`2 -2`
	`3 -3`
	`4 -4`

6. Arreglos y matrices

En Python, una lista es una forma de guardar una secuencia modificable de valores dentro de una variable. Los elementos de la lista reciben posiciones enteras numeradas implícitamente a partir de 0. Un concepto similar en C es el de arreglo, el cual tiene las restricciones adicionales de que no puede cambiar su tamaño inicial y de que todos los valores de la secuencia deben tener el mismo tipo. La declaración de una lista de Python y de un arreglo de C difieren en cuanto a notación, pero el acceso a los elementos de la secuencia es muy similar. Un arreglo en C debe declararse con un tamaño fijo, aunque éste puede inferirse del inicializador. En ambos lenguajes es un error accesar a una posición que no existe.

Código en Python	Código en C
`a = [ 5, 8, 6 ]`	`int a[] = { 5, 8, 6 };  // tamaño 3`
`print(a[0])  # 5`	`printf("%d\n", a[0]);  // 5`
`print(a[1])  # 8`	`printf("%d\n", a[1]);  // 8`
`print(a[2])  # 6`	`printf("%d\n", a[2]);  // 6`
`a[2] = 4`	`a[2] = 4;`
`print(a[2])  # 4`	`printf("%d\n", a[2]);  // 4`

Mientras que en Python existe la función `len` para consultar el tamaño de una lista, desafortunadamente no existe una manera cómoda o general en C de preguntarle al lenguaje el tamaño de un arreglo. El programador generalmente usa una constante o anota esta información en una variable adicional.

En Python, la forma de crear una lista de un tamaño específico sin listar manualmente todos sus valores es algo rara, mientras que en C basta con declarar el tamaño del arreglo dentro de los corchetes de la declaración. Si el arreglo se inicializa con llaves vacías, entonces todos los elementos reciben el valor 0 del tipo, aunque desafortunadamente no existe forma de inicializarlo implícitamente con otro valor.

Código en Python	Código en C
`a = 10 * [ 0 ]`	`int a[10] = { };`

Una matriz o arreglo multidimensional se representa en Python mediante una lista de listas en Python, mientras que en C se usa un arreglo de arreglos del mismo tipo y tamaño. Si la matriz se inicializa, C infiere sus dimensiones. El siguiente ejemplo crea una matriz de dos filas y tres columnas.

Código en Python	Código en C
`a = [ [ 1, 2, 3 ], [ 4, 5, 6 ] ]`	`int a[][] = { { 1, 2, 3 }, { 4, 5, 6 } };`
`print(a[0][0])  # 1`	`printf("%d\n", a[0][0]);  // 1`
`print(a[1][2])  # 6`	`printf("%d\n", a[1][2]);  // 6`

En C no existen rutinas generales de lectura o escritura de arreglos o matrices. Lo común es emplear ciclos (casi siempre ciclos `for`) para visitar y procesar cada elemento del arreglo o matriz, uno por uno.

Código en C

```
int n;
scanf("%d", &n);
int arr[n];       // declarar un arreglo de n enteros

for (int i = 0; i < n; i++) {
   scanf("%d", &arr[i]);      // leer cada elemento del arreglo
}
```

7. Caracteres y cadenas

El manejo de caracteres y cadenas de Python difiere casi por completo con el de C. En principio, un caracter en Python no es mas que una cadena de longitud 1 y no hay diferencia entre usar comillas dobles o sencillas al escribir literales de cadena. Por su parte, C tiene un tipo específico llamado `char` para manipular caracteres. En C, una literal de caracter usa comillas sencillas y una literal de cadena usa comillas dobles. Además, en C una cadena es simplemente un arreglo de `char`.

Código en Python	Código en C
`print("hola")`	`puts("hola");`
`a = '@'  # cadena de longitud 1`	`char a = '@';  // caracter individual`
`b = "@"  # cadena de longitud 1`	`char b[] = "@";  // cadena de longitud 1`

En Python, las cadenas no se pueden modificar y es necesario convertirlas en listas para hacerles algún cambio (y luego es necesario volver a convertirlas en cadenas en caso de requerirlo). Como las cadenas de C ya son arreglos, estos pasos intermedios son innecesarios.

Código en Python	Código en C
`a = "casa"`	`char a[] = "casa";`
`t = list(a)`	`a[1] = 'o';`
`t[1] = "o"`	
`a = "".join(t)  # "cosa"`	

En C, la lectura de caracteres y cadenas se puede hacer con los especificadores `%c` y `%s`, respectivamente. La lectura de cadenas requiere que le indiquemos la posición del arreglo a partir de la cual comenzará a guardar la cadena leída. El especificador `%s` está orientado a *tokens*, por lo que leerá sólo una palabra. Si se quiere leer una línea completa, se puede usar el especificador de formato `%[^\n]`.

Código en Python	Código en C
`# leer un caracter`	`char c;`
`c = sys.stdin.read(1)`	`char s1[100], s2[100];`
`# leer una línea`	
`s = input( )`	`// leer un caracter`
`# leer una palabra es difícil en Python`	`scanf("%c", &c);`
	`// leer una línea`
	`scanf("%[^\n]", &s1[0]);`
	`// leer una palabra`
	`scanf("%s", &s2[0]);`

En Python, el tamaño de la entrada es irrelevante porque Python usa cadenas que son lo suficientemente grandes como para guardar todo lo leído. Como las cadenas de C son arreglos que tienen un tamaño especificado antes de hacer la lectura, es importante que el tamaño sea suficiente para guardar lo que `scanf` leerá. Por su fuera poco, Las cadenas de C siempre guardan un caracter implícito al final de la misma, el cual se denomina caracter nulo. Eso quiere decir que si vamos a leer una cadena de a lo mucho n caracteres, debemos apartar un arreglo de $n + 1$ de capacidad.

Código en C	Entrada

`// necesitamos al menos 5 de capacidad`
`char s[4 + 1];`
`scanf("%s", &s[0]);`
	`hola`

Lo normal en C es apartar arreglos que sean un poco más grandes que la cantidad máxima de caracteres que vayamos a leer o guardar. Una vez guardada la cadena, podemos preguntar la longitud verdadera de la misma con la función `strlen` que está en el archivo de biblioteca `string.h`.

Código en Python	Código en C
`s = "hola"`	`char s[10] = "hola";`
`t = len(s)  # 4`	`int t = strlen(s);  // 4`

8. Registros o estructuras

Los registros o estructuras son tipos compuestos. Cada valor miembro de la estructura tiene asociado un identificador y no todos los valores necesitan ser del mismo tipo. En Python es muy común usar diccionarios, los cuales se pueden inicializar y posteriormente se les pueden agregar o eliminar miembros. En C, el concepto más cercano es el de tipo `struct`, pero un `struct` tiene una estructura rígida que no cambia con el tiempo y que además necesita ser declarado con anticipación. La declaración de un `struct` en C puede ser algo fastidiosa y rara notacionalmente, pero su uso posterior es sencillo. El acceso a los miembros de un `struct` en C se hace con el operador `.` seguido de un identificador y no con corchetes.

Código en Python

```
r1 = { "nombre": "pablo", "edad": 5 }
r2 = dict(nombre = "juan", edad = 22)
print(r1["nombre"], r2["edad"])
```

Código en C

```
#include <stdio.h>

typedef struct {
   char nombre[10 + 1];
   int edad;
} persona;

int main( ) {
   persona r1 = { "pablo", 5 };
   persona r2 = { "juan", 22 };
   printf("%s %d\n", r1.nombre, r2.edad);
   return 0;
}
```

9. Problemas de programación en Python y C

A continuación se muestran una serie de problemas de programación y códigos que los resuelven tanto en Python como en C. Las notas de curso de la UEA "Programación Estructurada" disponibles en el sitio `https://sites.google.com/site/rccuam` abordan dicho curso usando C y no Python, por lo que son una buena referencia para conocer un poco más de este lenguaje.

1. Resuelve el problema disponible en `https://omegaup.com/arena/problem/Detectando-el-orden`, el cual consiste en leer tres enteros y determinar si está en orden ascendente, en orden descendente, si todos los enteros son iguales o si están desordenados.

Solución en Python 3:

```
a, b, c = map(int, input( ).split( ))
if a == b == c:
   print("I")
elif a <= b <= c:
   print("C")
elif a >= b >= c:
   print("D")
else:
   print("X")
```

Solución en C:

```
#include <stdio.h>
```

```
int main( ) {
   int a, b, c;
   scanf("%d%d%d", &a, &b, &c);

   if (a == b && b == c) {
      puts("I");
   } else if (a <= b && b <= c) {
      puts("C");
   } else if (a >= b && b >= c) {
      puts("D");
   } else {
      puts("X");
   }

   return 0;
}
```

2. Resuelve el problema disponible en `https://omegaup.com/arena/problem/Divisores-positivos`, el cual consiste en leer un entero positivo y contar cuántos divisores enteros tiene.

Solución en Python 3:

```
n, d = int(input( )), 0
for i in range(1, n + 1):
   if n % i == 0:
      d += 1
print(d)
```

Solución en C:

```
#include <stdio.h>

int main( ) {
   int n, d = 0;
   scanf("%d", &n);

   for (int i = 1; i <= n; ++i) {
      if (n % i == 0) {
         d += 1;
      }
   }

   printf("%d", d);
   return 0;
}
```

3. Resuelve el problema disponible en `https://omegaup.com/arena/problem/Buscar-y-contar`, el cual consiste en leer una secuencia de enteros (de la cual lo primero que se da es su tamaño) y un entero a buscar. El programa debe contar cuántas veces aparece el entero en la secuencia.

Solución en Python 3:

```
n = int(input( ))
arr = map(int, input( ).split( ))
b, res = int(input( )), 0
for actual in arr:
   if actual == b:
      res += 1
print(res)
```

Solución en C:

```
#include <stdio.h>

int main( ) {
   int n;
   scanf("%d", &n);

   int arr[n];
   for (int i = 0; i < n; ++i) {
      scanf("%d", &arr[i]);
   }

   int b;
   scanf("%d", &b);

   int res = 0;
   for (int i = 0; i < n; ++i) {
      if (arr[i] == b) {
         res += 1;
      }
   }

   printf("%d", res);
   return 0;
}
```