FUNCIONES CON NUMERO VARIABLE DE ARGUMENTOS

[https://github.com/upiitacodelamberto/SOTR/UNIDADES/U0/01\_00FuncionesConNumeroVariable](https://github.com/upiitacodelamberto/SOTR/UNIDADES/U0/01_00FuncionesConNumeroVAriable)DeArgumentos/

En el lenguaje C existe la posibilidad de declarar funciones con un número de argumentos variable, número que no es conocido hasta el moemto en que se llama a la función. Un caso muy conocido en C de este tipo de funciones es la función printf, que puede imprimir un número variable de argumentos con el formato que se especifique en el primer argumento de la función.

Una de las formas en las en las que se puede programar este tipo de funciones es utilizando funciones y macros predefinidas en el archivo de cabecera <stdarg.h>, las cuales proporcionan facilidades para recorrer la lista de argumentos variables aun cuando sean de tipo y número desconocido.

Para ilustrar el uso de funciones con un número variable de argumentos, defínanse dos funciones que admiten un número vatriable de argumentos y hacen operaciones de suma y multiplicación sobre dichos argumentos.

int sumar(int cantidad,...);

int multiplicar(int catidad,...);

El detalle característico de estas funciones es que tienen los puntos suspensivos. Dichos puntos, que deben aparecer obligatoriamente al final de la lista de argumentos conocidos, indican que la función puede tener además un número variable de argumentos. Para acceder a dicha lista de argumentos dentro de la función, se usa el tipo va\_list y tres macros que operan sobre objetos de tipo lista de argumentos: va\_start, va\_arg y va\_end (todas ellas definidas en <stdarg.h>).

El tipo va\_list depende de la máquina en que se ejecuta la aplicación, pero habitualmente se define como un vector o un apuntador. Para trabajar con los argumentos, es necesario declarar una variable de este tipo dentro de la función con número de argumentos variable:

va\_list ap; /\* lista de argumentos \*/

Una vez declarada esa variable, y antes de hacer algo con ella, es necesario inicializarla con la lista (o su dirección) de argumentos variable. Dicha inicialización se hace usando la macro predefinida va\_start de la siguiente manera:

va\_start(ap,cantidad);

Es decir se guarda en la variable ap la dirección de inicio de la lista de argumentos variables, o lo que es lo mismo, la posición del último argumento conocido (en este caso cantidad). Observe que esto obliga a que en cualquier función con número de argumentos variable exista al menos un argumento con nombre en la definición de la función. En caso contrario, sería imposible obtener la dirección de comienzo para la lista de los restantes argumentos.

Una vez inicializada la variable con la lista de argumentos, es necesario recorrer dicha lista para extraer todos los argumentos existentes. Para ello se usa la macro predefinida va\_arg, función que extrae el siguiente argumento de la lista. El prototipo de dicha función exige conocer el tipo del argumento que se espera obtener de la lista para poder extraerlo. Por esta razón, entre otras cosas, se usan los especificadores de conversión en la función printf.

type va\_arg(va\_list args,type);

Así, en la función sumar, donde todos los argumentos son enteros, un argumento se extrae de la siguiente manera:

arg=va\_arg(ap,int);

Para extraer todos los elementos de la lista, es necesrio recorrerla secuencialmente y llamar a la macro va\_arg en cada paso. Finalmente, cuando todos los argumentos han sido extraidos, en necesario llamar a la macro va\_end para limpiar de la pila de llamada de la función todos los elementos variables. Esta limpieza es necesaria para asegurar que la función con número de argumentos variable termina normalmente y vuelve a la dirección de retorno incluida en la pila cuando la función fue llamada. La función de limpieza, se llama llama en la función sumar de la siguiente manera:

va\_end(ap);

Obsérvese que la macro va\_arg solo funciona correctamente cuando se conoce el tipo de dato a extraer y si dicho tipo se puede convertir facilmente a un apuntador añadiendo el operador de indirección, \*. Los tipos básicos, como char o int, funcionan sin ninguna dificultad. Si los argumentos son objetos más complejos, como vectores o estructuras hay que pasarlos de forma que la conversión a apuntador sea correcta. Por ejemplo, para un vector de enteros, int[], la forma correcta de uso es int \*, y no int[], ya que int[]\* no es un tipo de dato válido en C.

El número de argumentos en funciones de este tipo debe ser igual o mayor que el especificado en los parámetros con nombre. Si se sobrepasa el número definido, los parámetros excedentes no serán tenidos en cuenta. Si hay menos parámetros de los indicados se pueden obtener resultados impredecibles dependiendo de la implementación de va\_arg y va\_end.