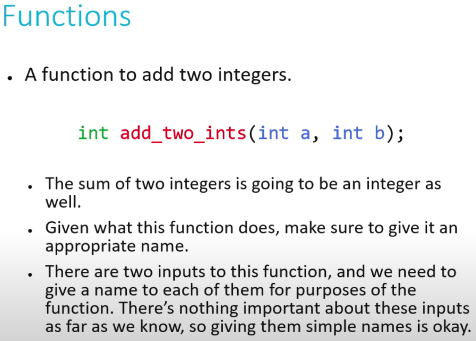
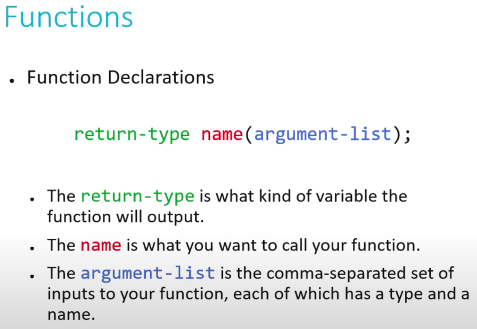
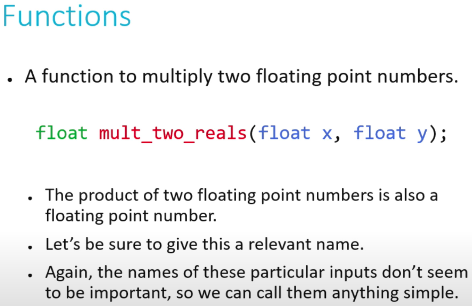
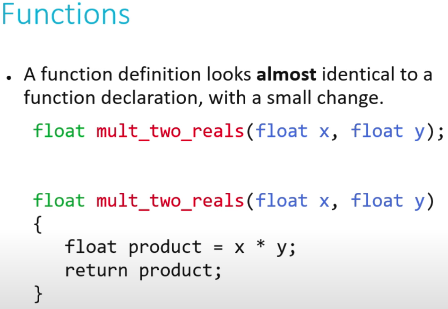
***Sobre las funciones***

***Consejo:*** *si dentro de las librerias o bibliotecas a las que usted tiene acceso en su archivo originario cuenta ya con una función, cuál sea, que sea de su interés parcial; entonces, no es necesario diseñar una función personalizable en caso que deseara optimizar su proceso de programación por medio de una: simplemente se llama a la función ya existente que posee una libreria (también ya diseñada) de su acceso en cuestión. Por el contrario, si le urge simplificar y dividir las etapas de su programa por medio de una función que se dedique especificamente a una tarea; pero resulta que, esa tarea (función) no la contempla ninguna de las librerias a las que tiene acceso; entonces ahí, sí se hace necesario quizás crear una función personalizable.*

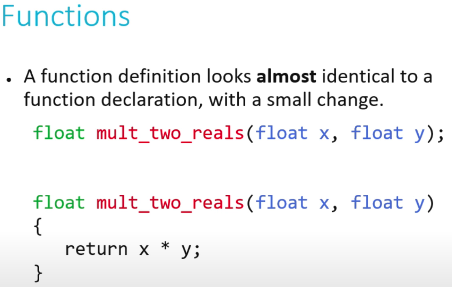
*Las funciones son muy importantes, nos permiten depurar con mayor facilidad nuestro programa, dividirlo en partes para no hacer el código tan complejo en su comprensión (y en caso de eventuales errores pescarlos más rápidos). Incluso, podríamos exportar todas las funciones personalizables que creemos por medio de una libreria (que supone uno también deberiamos crear). Así es como, por ejemplo, la función* ***printf*** *sólo fue necesario que se escribiera una vez y ahora cualquier programador tiene acceso a ella si incluye dentro de su programa la libreria o biblioteca* ***stdio. Hace más de 40 años se escribio, por una sola vez, la función printf y desde entonces sólo se ha reciclado a través del tiempo.***

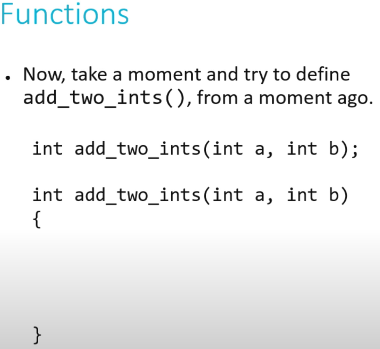


*Esto...*



*Es lo mismo que esto...*

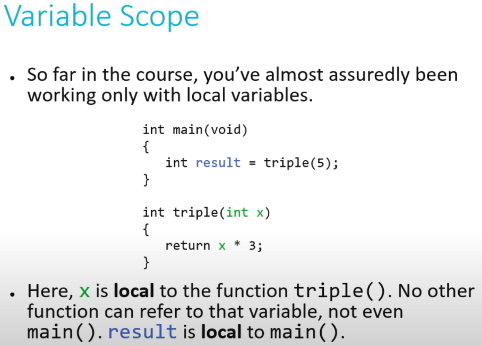


***ahora, haz este ejercicio:***

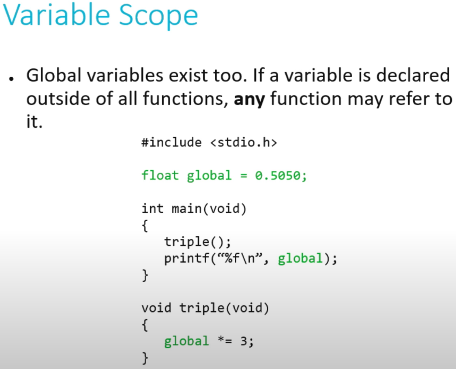
***Sobre las variables***

*El alcance de una variable (variable scope) evalúa los límites, en términos de funcionalidad, de una variable. Si puede operar en todas las funciones de nuestro archivo orginario o sólo dentro de una función en especifica. Eso nos obliga a distinguir entre dos tipos de variables:*

***Variables locales (local variables):*** *A estas variables sólo se puede acceder desde las funciones particualres donde fueron creadas (no importa si se trata de funciones personalizables o si se trata de la función principal). Dicho lo anterior, no se puede acceder a ellas desde cualquier otra función que exista en su programa; sólo desde la función donde fueron creadas.*



***Variables globales (global variables):*** *Estas variables tienen la facultad de ser ejecutadas desde cualquier función dentro del programa, sea principal o una función personalizable. La razón de porque se hacen o son variables globales es porque sencillamente no fueron creadas dentro de ninguna función en particular: antes se declaran por fuera de todas las funciones (incluso por fuera de la función principal).*



*En el anterior ejemplo nuestra variable global es:* ***“float global = 0.5050”.*** *De hecho, si te das cuenta, nos referimos a la variable “global” tanto en la función “main” como en la función “triple”. Ahora “global” no imprimirá 0.5050, sino 1.515 (sabe por qué me imagino).*

*En las variables globales, si manipulamos una variable global en una función, el efecto sobre esa función se extiende sobre todas las demás funciones. Pero con las variables locales, eso no es cierto.*

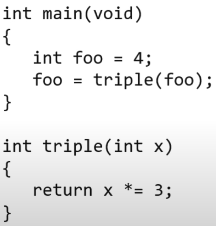
***Passed by value (pasado por valor)***

*Muchas veces se nos ha enseñado solamente a trabajar con variables locales; pero, además de eso, a manipularlas entre sí dentro de nuestro programa.* ***Si usted revisa el apartado 2, del bloque 4 dentro del subdirectorio “Week 1, C” de la carpeta de “CS50’s Introduction to Computer Science”****; siempre estamos hablando de “pasar, a partir de una función independiente o personalizable, toda su información, en términos de condiciones de dicha función, a nuestra función principal”.*

***Deciamos:*** *“... Es decir, al pasar a trabajar con las variables "a" y "b" de la función "maximo" (una función personalizable cualquiera) a las variables "x" e "y" respectivamente dentro de la función "main" (porque estamos trabajando dentro de esta última función y no con otra); entonces, la información y las condiciones estipuladas dentro de la variable "aux" (que ha sido retornada: “return aux;”) de la función "maximo" serán importadas dentro de la función "main"; mas no sus variables ni mucho menos los valores de dichas variables; sólo las condiciones; en este caso, elegir el número mayor entre una variable y otra. OJO.”*

*La función “máximo”, almacenda en la declaración “return aux;”, que determinaba inicialmente qué número era mayor entre la variable “a” y “b”; fue llamada en la función “main”: lo que quiere decir entonces es que, ahora se hará la elección del número mayor entre dos variables (porque para eso fue escrita la función, para llevar acabo esa tarea) pero, ojo, entre dos variables de la función “main” y no “maximo”, porque es en la función principal que ahora está siendo llamada la función “máximo”, sus condiciones y no sus variables (viendo también que se trata de variables locales, es así como funciona).* ***A esta dinamica, de pasar condiciones de una función personalizable a otra (y trabajar ahora con las variables locales de esa otra función), se le conoce como: “Passed vy value” o, en español, “pasado por valor”.***

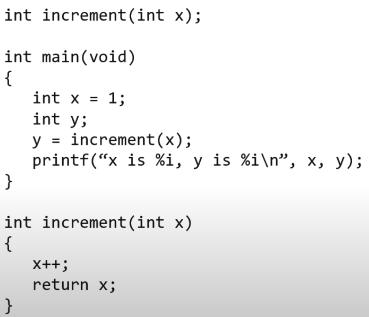
***Un ejemplo:***



***La variable de tipo entero “foo” será equivalente a: 12. Le hemos pasado las condiciones de la función personalizable “triple”, reflejadas en “return x\*= 3;”, a nuestra función principal. Esto quiere decir entonces que, dentro de la función “main”, estaremos trabajando propiamente con las variables de “main” (y no con otras) a la hora de llamar a la función “triple” (y sus condiciones) dentro de la función “main”.***

*En “main” no estariamos hablando entonces de:* ***“return x\*= 3;****,**sino de,* ***return foo\*= 3;****, lo cuál daría como resultado:* ***12****... ciertamente.*

*Ahora, un concejo que quizás ya di: ya sabemos que no podemos trabajar con la variable local de una función dentro de otra función, de hecho ni se reconocen entre sí (por lo menos ahora); pero, evitese de todas maneras, ponerles nombres iguales a sus variables locales entre una función y otra (se complicará la vista si lo hace). Veamos este ejemplo:*



***Debe saber qué valores se imprimen al final de la función principal.***