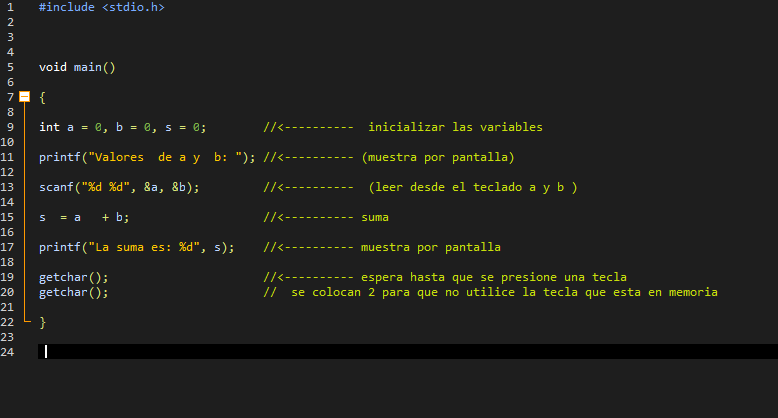
Alumn@s sean bienvenidos a una nueva semana de Herramientas de Programación, ya estamos avanzados en la comprensión de sentencias y de estructuras de los programas que se escriben en C.

     En esta ocasión continuaremos hasta finalizar el **Tema IV**donde podremos ver cómo utilizar diferentes directivas para ordenar, definir e incluir variables, archivos o incluso realizar llamadas externas.

     En la ejercitación vista hasta el momento pudimos realizar varios pequeños programas, ahora empezaremos a utilizar la sentencia "scanf" que nos permite realizar inclusiones de entradas al programa por el teclado y en de diversos formatos de variables.

     por ejemplo, en:



     En este caso gracias a la sentencia ***scanf***pudimos asignarle un valor a las variables "a" y "b" anteponiendo el simbolo "**&**" (notese que se puede realizar mas de un ingreso en la misma secuencia), ingresamos 2 valores enteros para las varibles "a" y "b" (linea 13) y luego los sumamos dichos valores y los asignamos a la variable "**s**" (linea 15) que luego mostraremos por pantalla con la conocida funcion **printf**(linea 17).

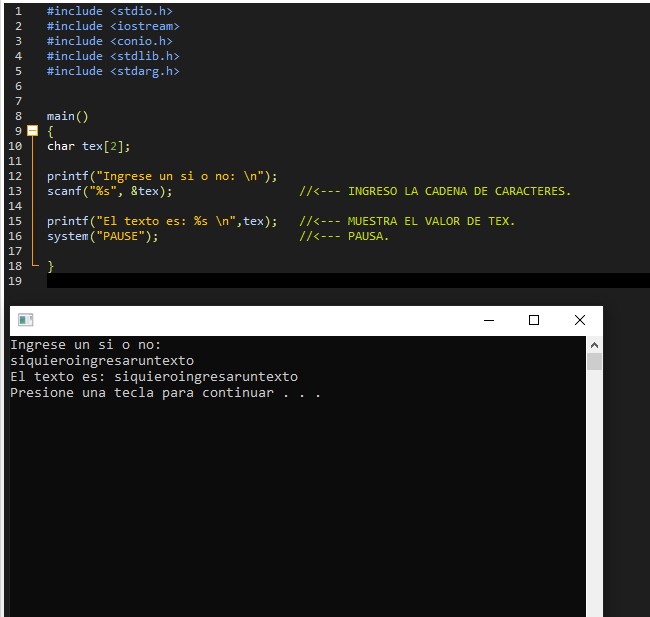
     Algunos de los ingresos que se pueden realizar con scanf:

          scanf("%s", &variable\_cadena); //Para string \_ cadena  
          scanf("%d", &variable\_int);        //Para entero  
          scanf("%ld", &variable\_long);    //Para long  
          scanf("%c", &variable\_char);     //Para caracter  
          scanf("%f", &variable\_float);      //Para float  
          scanf("%lf", &variable\_double); //Para double (long float)

     Scanf no solo nos permite leer los datos de entrada, sino que anteponiendo diferentes siglas como "f" o "s" podremos

* La función scanf() lee los datos de entrada en el **stdin** ([flujo de entrada estándar](https://es.wikipedia.org/wiki/Entrada_est%C3%A1ndar)).
* La función fscanf() (*file-scanf*) lee en un flujo de entrada dado, por lo general un [fichero](https://es.wikipedia.org/wiki/Archivo_inform%C3%A1tico) (file) abierto para lectura.
* La función sscanf() (*string-scanf*) obtiene la entrada que se va a analizar de una [cadena de caracteres](https://es.wikipedia.org/wiki/Cadena_de_caracteres) dada (*string*).

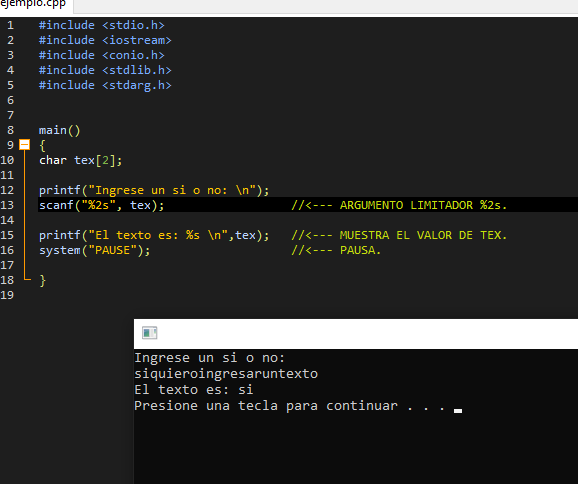
Veamos algunas curiosidades:



     En el anterior ejemplo se crea una variable del tipo char (caracter) que va a contener una cadena de 2 letras "tex[2]", pero que ocurre cuando realizo el ingreso con scanf, al darle la orden de que ***scanf***va a trabajar con un "%s" (string), ***scanf***toma todo el texto que escriba no importa las limitaciones que tenga la variable "tex", por esto a la hora de mostrar por pantalla el valor del ingreso nos va a mostrar toda la linea de caracteres anteriormente ingresado.

     Para que esto no ocurra lo que vamos a utilizar es el argumento "%2(n\_cant\_caracteres)s" y se lo vamos a asignar a "tex" **sin utilizar** &, de esta forma utilizando el limitador de cantidad ingresada lo que va a tomar scanf en nuestro caso van a ser los 2 primeros caracteres ingresados, OJO podemos utilizar n\_cantidad de caracteres si ponemos un 3 o un 4 nos tomaria esa cantidad.

     Es recomendable entonces que escriban en el compilador el siguiente ejemplo y jueguen un poco con el, dandole diferentes valores y probando colocar mas scanf de distinto tipos para comprender correctamente su funcionamiento, recuerde que para poder ver los resultados por pantalla siempre es necesario un printf que enseñe el valor.



**Algunos EJEMPLOS:**

**Guarda un número en n:**  
int nu;  
printf("Introduce un numero: ");  <---- muestra por pantalla.  
scanf("%d",&nu);                          <---- asigna a traves de un ingreso un valor a la variable "nu".  
   
   
**Guarda un caracter en x:**  
char x;  
printf("Introduce un caracter: ");   <---- muestra por pantalla.  
scanf("%c",&x);                               <---- asigna a traves de un ingreso un valor a la variable "x".  
   
   
**Guarda una cadena de caracteres (solamente una palabra) en cad.**  
char cad[20];                              <---- creamos una cadena de n caracteres (en este caso 20).  
printf("Introduce una palabra: ");  <---- muestra por pantalla.  
scanf("%s", &cad);                      <---- utilizamos %s para identificar una cadena de tipo string.  
printf("Introduce una palabra: ");  <---- muestra por pantalla.  
scanf("%10s",cad);                      <---- lee máximo 10 caracteres y le concatena el caracter cero.

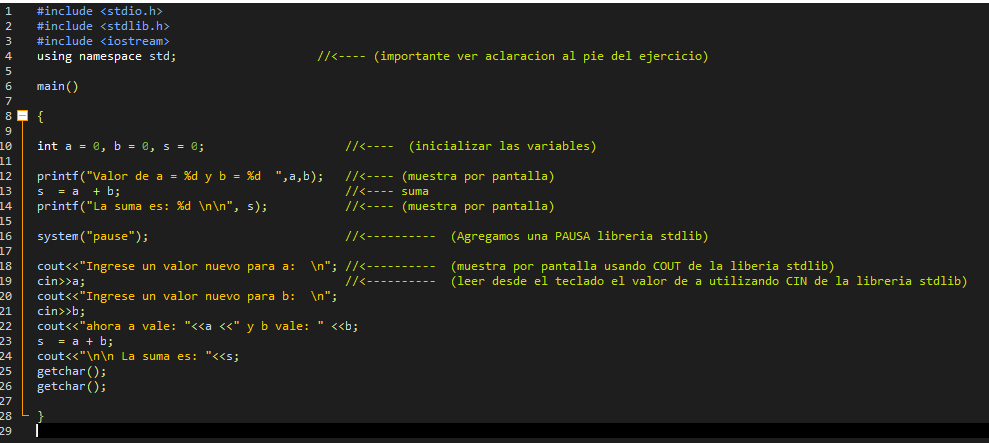
     Pueden probar cualquiera de estos ejercicios siempre teniendo en cuenta que hay que incluir la libreria y correr el programa como un main.

**CIN - COUT**

     Otro formato para poder realizar ingresos es utilizando la sentencia **CIN** que esta disponible en la libreria #include <iostream>:

     Leer datos por teclado en C++ se hace usando el comando *cin >>* es importante notar el uso de los dos signos *>>* que son usados para controlar el flujo de datos. Cada vez que utilicemos *cin* debes agregarle *>>* para quedar con un *cin>>*. Una manera muy sencilla de recordar esta instrucción es que **in**significa entrar y como estamos programando en C++ le añadimos la letra C al comienzo quedando así *cin>>* (sin olvidar los *>>*).

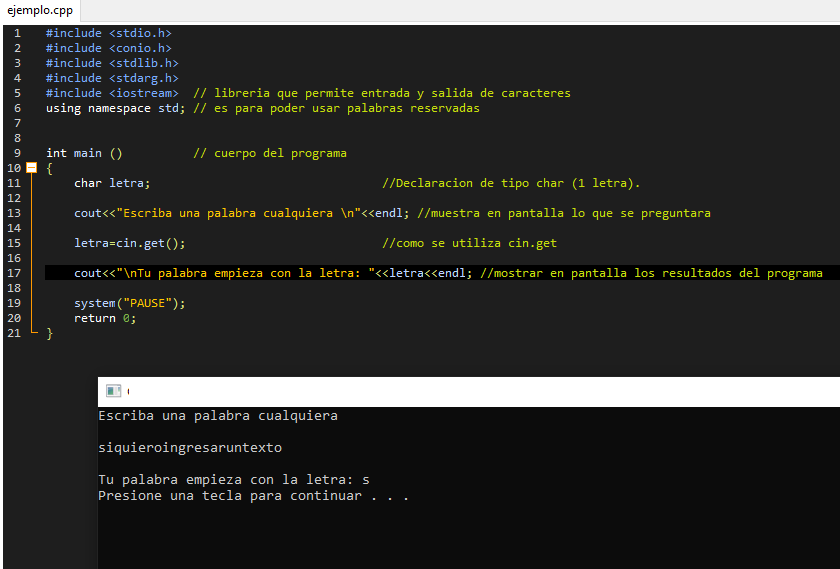
     Veamos unos ejemplos simples:



     En este ultimo ejercicio utilizamos un mix de printf y cout, asi como utilizamos el cin para realizar los ingresos. Hay varias cosas a tener en cuenta, de las mas importantes son:

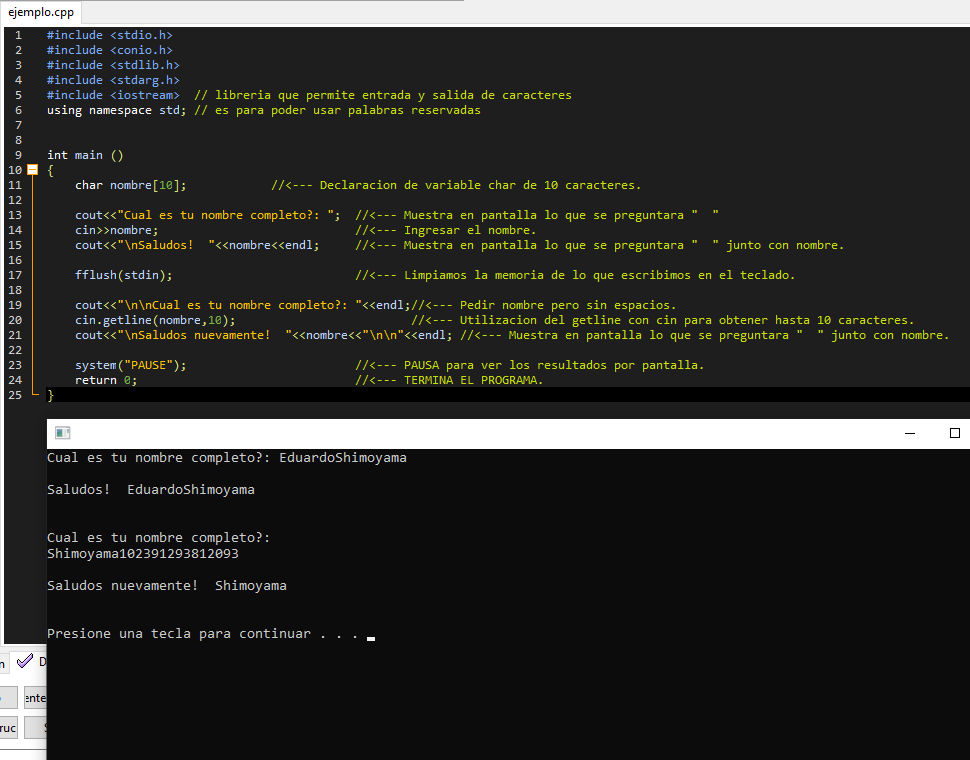
1) Al ser C++ una ampliación del lenguaje C, es necesario agregar nuevas palabras reservadas. Éstas palabras reservadas están en un “namespace” (espacio de nombres). Para ser mas específicos, las palabras reservadas cout y cin están el namespace std (standard) es por ello que si los tuvieramos que utilizar deberiamos de declararlo en el programa de la siguiente forma: std::cout<<"hola mundo"; para evitar utilizar una y otra vez esta larga sentencia le indicamos al programa que utilice directamente las palabras reservadas agregando **using namespace std;** al comienzo del programa, de esta forma cada vez que utilicemos cout o cin lo nombraremos directamente en la linea de codigo.  
  
2) Notese que la forma de ingreso y de salida del sistema es diferente a scanf y printf respectivamente, cin permite ingresos tanto de numeros como de strings sin restricciones, no requerimos el uso de %d u otra forma de nomenclatura para asignarle un valor, **cin**detecta que tipo de variable es y le asigna el valor correspondiente esto es muy util.

     Asi como vimos algunas particularidades de scanf ahora vamos a ver como funciona **cin**por ejemplo supongamos que queremos capturar una linea de caracteres en cin:



     Como podemos observar en la imagen de arriba tenemos una variabla "letra" a la cual le asignamos la funcion cin.get(); esto es muy similar al getchar y nos permite obtener lo que escribimos por teclado pero solo la primera tecla, por dicho motivo lo que se obtiene como resultado es la letra o caracter (char) "s".

     Vamos a mostrar una linea de texto:



     En este caso utilizaremos 1 variable char[10] es decir que puede almacenar hasta 10 caracteres, pero que pasa cuando le damos ingresos de caracteres con cin? pues asi como ocurria con scanf este metodo captura absolutamente todos los caracteres ingresados no importa si son mas que 10, por tal motivo se puede leer "EduardoShimoyama" por mas que sean mas de 10 caracteres. Para limitar la cantidad de caracteres ingresados entonces utilizamos la funcion getline, cin.getline va a tomar lo que escribimos por teclado y por mas que sean mas de 10 caracteres va a limitar la lectura de nombre a (n\_caracteres) es nuestro caso **10** fijense que en la segunda linea solo se lee Shimoyama no importa cuantos caracteres mas agreguemos a la cadena. Si contamos S H I M O Y A M A, tiene 9 letras entonces porque solo muestra esa cantidad? esto se debe a que nosotros colocamos el tope de la linea en el punto 10 es decir que si quiero mostrar 10 caracteres completos tendria que poner el tope en 11 y asi sucesivamente.

**Algunos EJEMPLOS:**

     En el caso de un *valor númerico*, ya sea int, float, double, etc. veremos las diferencias de ingresar datos con scanf o con cin, y de mostrar la salida con printf o con cout.

si en -------> scanf ("%.2f", &a);                         /        en -------> cin>>a;  
si en -------> printf("El valor de a es = %.2f ",a);   /       en ------->cout<<"El valor de a es = "<<a;

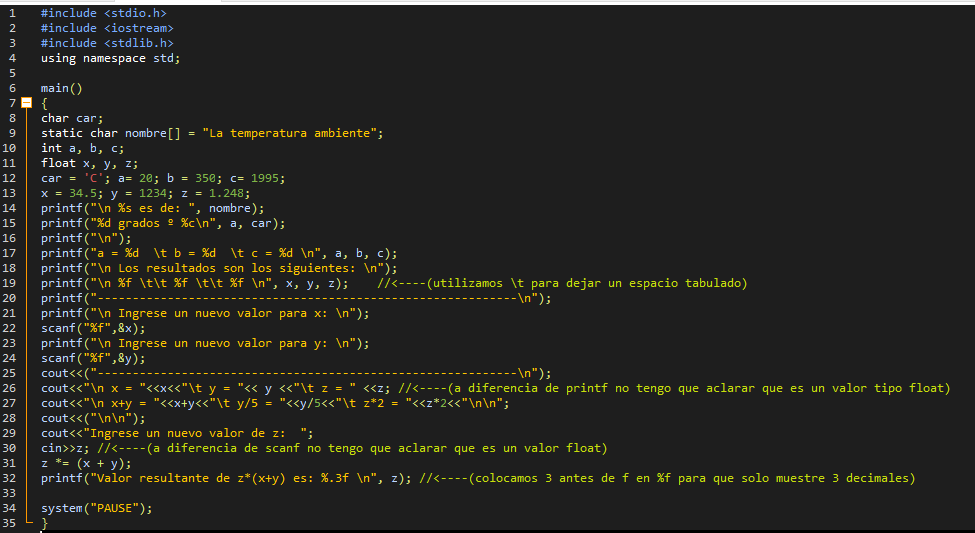
En el caso del ***texto tipo string*** es mas complicado aun, para **scanf**primero debemos crear una cadena de caracteres.

char palabra[10];          //<---- se crea la cadena o vector de 10 caracteres.  
scanf("%s", &palabra);  //<---- se utiliza %s para ingresar datos a una variable string.

     Pero si utilizamos **cin**

string palabra;         //<---- declaramos la variabla palabra  
cin>>palabra;         //<---- le asignamos el valor string que queremos sin restricciones a la variable "palabra".

     Probemos como quedaria estéticamente un ejercicio que respeta y utiliza todos los parámetros de entrada y salida aprendidos hasta el momento.



**EJERCITACION / TAREA:**

     Ahora como ejercitación van a copiar cada línea de código que esta en el ejercicio de arriba denominado por nombre ^TEMPERATURA, van a compilarlo y ejecutarlo, lógicamente les va a solicitar tal y como se muestra en las entradas y salidas asignarles nuevos valores a las variables "X, Y, Z"  realicen una captura de pantalla a ver como luce y que lo utilizaremos mas adelante.

**ATENCION!! RECUERDEN REVISAR CADA SEMANA LA SECCION DE AVISOS! YA QUE EL VIERNES DE ESTA SEMANA SE CIERRA LA INSCRIPCION A LOS EQUIPOS DE TRABAJO.**