-SENTENCIA WHILE-

     La sentencia **while** ejecuta un ciclo dada una condición simple o compuesta, tantas veces como sea necesario, hasta que la condicion se deje de cumplir. Resulta de gran importancia recordar que el ciclo unicamente se inicia si se cumple con X condicion/condiciones determinadas.

     Su sintaxis es:

**while** (condición)

{

Sentencia1;

Sentencia2;

…..

}

     Donde condición se refiere a cualquier expresión numérica, relacional o lógica y dentro del ciclo la/s "sentencia/s" se refiere a una conjugación de líneas a ejecutar.

     La ejecución de la sentencia **while** sucede así:

1. Se evalúa la condición.
2. Si el resultado de la evaluación es cero (falso), la sentencia no se ejecuta y se pasa el control a la siguiente sentencia en el programa.
3. Si el resultado de la evaluación es distinto de cero (verdadero), se ejecuta la sentencia y el proceso descrito se repite desde el punto 1.

     Al igual que el if en **while** tenemos los operadores lógicos **|| (or) && (and) ! (not)**

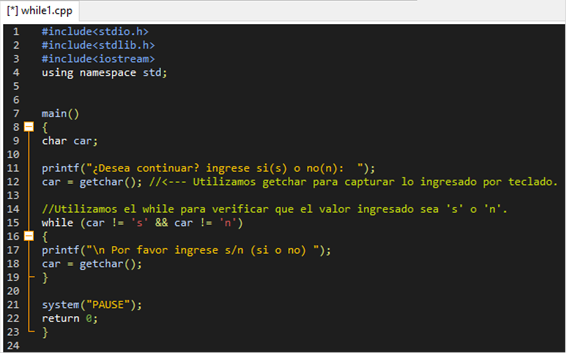


     El operador “&&” equivale al “AND” o “Y”; devuelve true sólo si los dos operandos son true o lo que es equivalente, distintas de cero. En cualquier otro caso el resultado es false. Es decir que ej: while ( a==0 && b==1 ) se cumple solo cuando las dos variables tengan esos valores en el caso que a=0; y b=5; no ingresa al ciclo.

     El operador “||” equivale al “OR” u “O inclusivo”; devuelve true si cualquiera de las expresiones evaluadas es true, o distinta de cero, en caso contrario devuelve false. Es decir que ej: while ( a==0 || a==1 ) se cumple SIEMPRE y cuando la variable "a" tenga el valor 0 o 1 tambien podria ser que while ( a==0 || b==1 ) en este caso podremos ingresar al ciclo siempre y cuando se cumpla alguna de las 2 condiciones, es decir a debe valer 0 sin importar cuanto valga b o bien b debe valer 1 sin importar cuanto valga a.

     El operador “!” es equivalente al “NOT”, o “NO”, y devuelve true cuando la expresión evaluada es false o cero, en caso contrario devuelve false. Es decir while       ( a!=0 ) ingresa al ciclo siempre y cuando el valor de a NO sea igual a 0, con cualquier OTRO valor puede ingresar.

     Por ejemplo, la rutina siguiente solicita obligatoriamente que unas de las dos respuestas posibles sean: s/n (sí o no).



Observe que antes de ejecutarse la sentencia while se visualiza el mensaje "Desea continuar ingrese si(s) o no(n). ¿De que se trata este while? Este ciclo comprobara que el valor que ingresemos por teclado sea ‘s’ o ‘n’ y rechazara cualquier otro valor volviendo a ingresar al ciclo. Esta comprobación es súper útil cuando nosotros esperamos obtener un resultado exacto y permite **validar** el ingreso realizado.

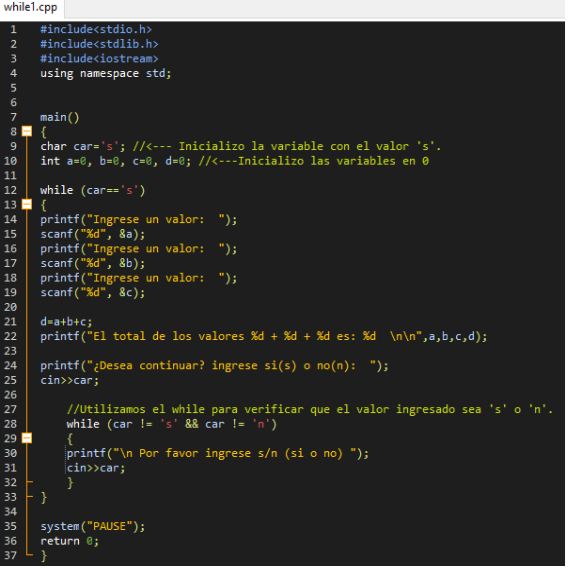
La sentencia while se interpreta de la forma siguiente: mientras el valor de car no sea igual (!=) ni al carácter 's' ni al carácter 'n', visualizar el mensaje "Por favor ingrese s/n (sí o no)" y luego getchar asigna el valor a car. Esto obliga al usuario a escribir el carácter 's' o 'n'. Es decir que en este caso en particular controlaremos todos los resultados posibles y optaremos por tomar una ruta de funcionamiento solo cuando este sea s o n.

¿Por qué utilizamos getchar? En este caso decidí utilizar esta función para que Uds. estimados alumnos comprendan el funcionamiento del mismo, por ejemplo, cuando nos pida ingresar s o n jueguen ingresando una serie de caracteres como ser: auto, casa, arpon, etc. Y vean que sucede.

**While ANIDADO:**

     Además de los bucles while, la sentencia puede ser utilizada también para realizar estímulos del tipo anidado, es decir cuando se incluye una sentencia while dentro de otra sentencia while, en general una sentencia while, do, o for dentro de otra de ellas, estamos en el caso de bucles anidados.

**\*Ejemplo 1: while anidado.**



     En este caso utilizamos el primer while para volver a ejecutar el ingreso de valores y la suma (d) y el segundo while para controlar los valores ‘s’ o ‘n’ de la variable car.

     Debemos observar como "fuerzo" el primer ingreso al while (líneas de código 12 a 33) inicializando la variable **car** con el valor ‘s’ de esta forma me aseguro se ejecute el ingreso al primer ciclo.

     Luego inicializo las variables enteras "a, b...." = 0; para que no quede residuo en memoria, esto es una vieja práctica de la buena programacion, que es muy útil ya que algunas veces por algún motivo de los procesos de programacion los valores en memoria producen que las sumas no se resuelvan correctamente. También se sigue utilizando para cuando busco el número mayor o el número menor en una ecuación.

     El ***segundo while*** (líneas de código 28 a 32) verifica que lo ingresado sea un ‘s’ o ‘n’. Cualquier otro valor me pedirá que vuelva a realizar un ingreso que sea igual a s o n.

Copien el ejercicio y comprueben los resultados.

**SENTENCIA "do while":**

     do while es similar a while, ¿cual es la diferencia? Mientras que **while** se ejecuta solo cuando se cumple la condición, **do while** se ejecuta y luego condiciona la salida del ciclo. Su sintaxis es la siguiente:

     Su sintaxis es:

**do**

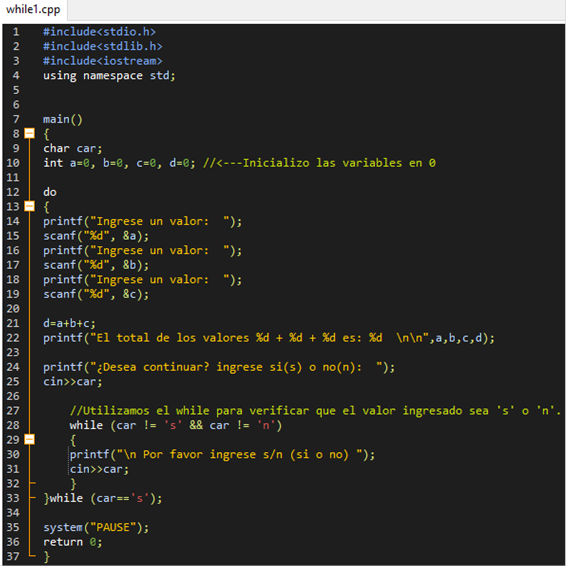
{  
Sentencia1;  
Sentencia 2;  
…..  
} **while**(condición);

\*Observe que la estructura do - while finaliza con un punto y coma.

     La ejecución de una sentencia do sucede de la siguiente forma:

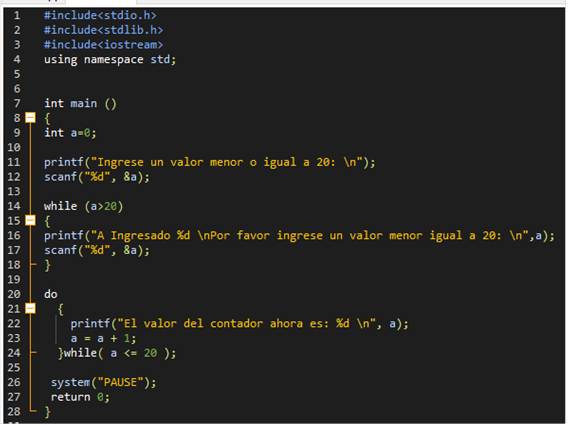
1. Se ejecuta el bloque (sentencia simple o compuesta) de do.
2. Se evalúa la expresión correspondiente a la condición de finalización del bucle.
3. Si el resultado de la evaluación es cero (falso), se pasa el control a la siguiente sentencia en el programa.
4. Si el resultado de la evaluación es distinto de cero (verdadero), el proceso descrito se repite desde el punto 1.

**\*Ejemplo 1bis: do while anidado.**



         Observemos el siguiente ejemplo, en esencia es el mismo ejercicio que el caso anidado anterior, pero utilizando **do while** no requiero inicializar la variable **car**, ya que el **do** me obliga a ingresar al ciclo y el **while** recién al final verifica la condición.

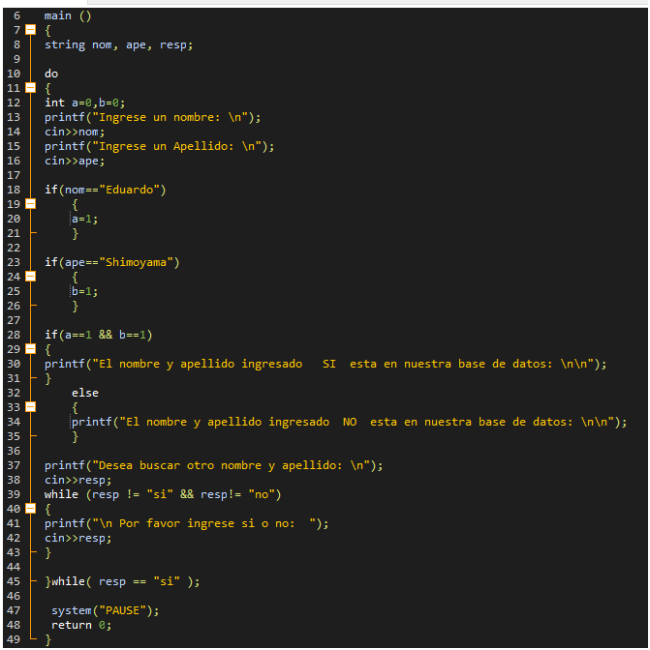
**\*Ejemplo 2: while valor.**



     Para este ejemplo utilizamos el primer while para comprobar que el valor asignado a la variable a no sea mayor a 20. Si fuera mayor nos pide que ingresemos nuevamente el valor hasta que este sea menor, igual a 20.

     Luego utilizaremos el do while como una especie de for es decir que la función realice bucles mientras no se termine de cumplir la condición de "while", entonces nuestro programa tomara el valor inicial de la variable *a* y realizara el primer ciclo, mientras no se cumpla la condición de que *a* sea menos o igual a 20, el do va a imprimir por pantalla los valores uno abajo del otro hasta llegar a 20.

**\*Ejemplo 3: while, do while, if.**



     En este ejercicio comenzamos a utilizar todos los conocimientos que fuimos adquiriendo hasta el momento, notaran que utilizo los ciclos do while y while, lo mas interesante que esencialmente me gustaría que observen en esta ejercitación es lo siguiente, prácticamente es como funciona una base de datos, ingresamos un nombre a la variable tipo string nom y un apellido a la variable ape, luego comparamos los datos ingresados con un if, este if va a  validar la igualdad entre el nombre y el apellido, si el dato ingresado es igual le asignara un valor =1 a las variables enteras a y b.

     Habiendo realizado todo esto se utiliza un tercer if para comprobar directamente el valor de a y b, y con ese resultado imprimo una pantalla “ ….SI esta en nuestra base de datos” o la otra “….NO esta en la base de datos”. Reflexionemos por favor sobre lo siguiente, en principio la comparación parece algo más complejo que directamente hacer 1 if que compare nom y ape y arroje el resultado, pero en una base de datos de gran escala es mucho más fácil para la memoria y para el procesamiento de datos comparar valores a 1 o 0 que a “Eduardo”.

     Finalmente utilizo un último while para comprobar que la variable string resp sea un “si” o un “no”, asegurándome que por la respuesta “si” vuelva a comenzar el ciclo do while y que por “no”, salga del sistema.

     Copien los ejercicios y cambien los valores de las distintas variables para observar los resultados y comprender su funcionamiento.

\*\*Les recuerdo que todos deben estar anotados en un equipo de trabajo para poder presentar los TP's de la materia y para presentar el PARCIAL.

\*\*\*No se olvide de revisar semanalmente el sector de "**AVISOS**" donde se encuentran las novedades de la materia y para cualquier pregunta o consulta se encuentra disponible mi mail eshimoyama@esbabarrionorte.edu.ar o el foro "**CONSULTAS**" donde podremos intercambiar ideas, situaciones que se les pueda haber presentado o incluso compartir conclusiones con todos. Saludos y que tengan una maravillosa semana.