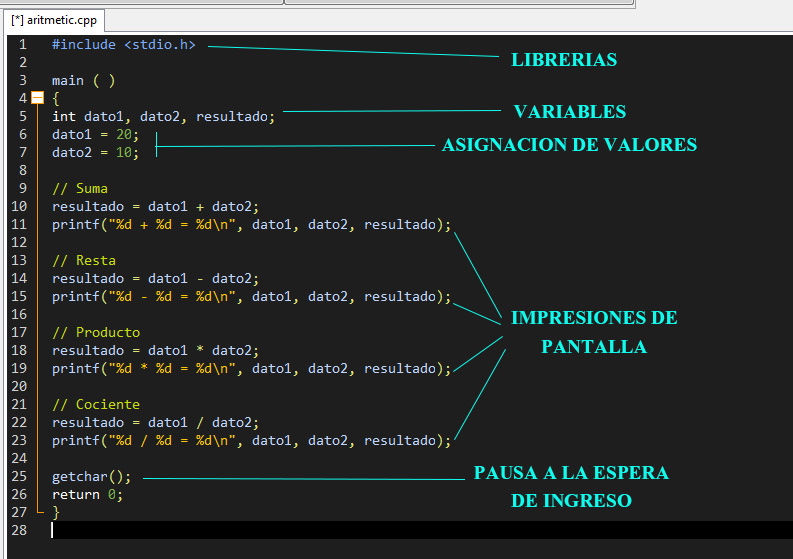
Estimad@s Alumn@s, en la clase pasada pudimos echarle un pequeño vistazo con el "Hola Mundo" a lo que se refiere la programación en C++, comenzaremos en esta clase a analizar con el manual en mano el ejemplo Simple del TEMA 3 y continuaremos con el TEMA 4 dispuesto en la bibliografía, para ver cuáles son las posibilidades que nos brinda este lenguaje con respecto a variables y parámetros.

          Como pudimos ver en la clase pasada, la llamada al argumento ***printf***nos permite mostrar por pantalla una cadena de caracteres del tipo ***string***, y visto en el ejemplo de aritmetic.cpp también puede mostrar resultados de números enteros que estén almacenados en una variable.

           Analicemos el ejemplo:



         Se observa que hay diferencias entre la sintaxis que vemos en el manual y la que se desarrolla en CLASE, esto se debe a que la idea principal de la materia es que utilicen ambas herramientas para la construcción del aprendizaje, por tal motivo van a encontrar errores comunes falta de pausas, puntuaciones, etc.., los cuales podremos compartir en el foro y sobre los cuales podremos dialogar para así llegar a una comprensión completa y correcta del funcionamiento del lenguaje.

         Primero: como ya habrán leído en el primer documento de la materia lo que va a tener nuestro programa es o son **las librerías** denominadas con el símbolo "#" y acompañado de la sigla "include".

         Segundo: el encabezado del programa "main ()" y la apertura "{ " así como el cierre al final " }".

         Tercero: **ASIGNACION DE VARIABLES** -> en este punto es donde ingresaremos las variables (en nuestro caso variables del tipo entero "dato1" "dato2" y "resultado”) que utilizara nuestro programa para resolver y trabajar.

         Cuarto: **IMPRESIONES DE PANTALLA** -> de la Suma, Resta, Producto y Cociente de los valores, fíjense como calculamos el **resultado**y le asignamos un valor y luego en la representación por pantalla utilizamos el printf y en la declaración de texto la expresión "**%d**" la cual es para mostrar al usuario un numero entero (int) que contiene cada variable, en otros casos se usaría un "**%s**" si fuera string o "**%f**" para una variable del tipo float, etc, etc., como lo explica el cuadro1 más abajo. Después de la coma, se colocan las variables que se reemplazaran a los especificadores en el orden que se leen *es decir el primer %d para el primer dato1 y así sucesivamente.*

**CUADRO 1:**



         Como vemos en la bibliografía esta subido el **TEMA 4** los tipos de Variables puede ser de los siguientes tipos:

Tipo "char" o carácter:

         Es el tipo básico alfanumérico, es decir que puede contener un carácter, un dígito numérico o un signo de puntuación. Desde el punto de vista del ordenador, todos esos valores son caracteres. En C++ este tipo siempre contiene un único carácter del código ASCII. El tamaño de memoria es de 1 byte u octeto. Hay que notar que **en C un carácter es tratado** en todo **como un número**, de hecho, habrás observado que puede ser declarado con y sin signo. Si no se especifica el modificador de signo, se asume que es con signo.

Tipo "int" o entero:

         Las variables enteras almacenan números enteros dentro de los límites de cada uno de sus tamaños. A su vez, esos tamaños dependen de la plataforma, del compilador, y del número de bits que use por palabra de memoria: 8, 16, 32... No hay reglas fijas para saber el tamaño, y, por lo tanto, el mayor número que podemos almacenar en cada tipo entero: **short int**, **int** o **long int**; depende en gran medida del compilador y del sistema operativo. Sólo podemos estar seguros de que el tamaño de un **short int** es menor o igual que el de un **int**, y éste a su vez es menor o igual que el de un **long int**. Veremos cómo averiguar estos valores cuando estudiemos los operadores.

Tipo "long long":

         Este tipo ocupa el siguiente puesto en cuanto a tamaño, después de **long int**. Como en los otros casos, su tamaño no está definido, pero sí sabemos que será mayor o igual que el de **long int**.

Tipo "float" o coma flotante:

         Las variables de este tipo almacenan números en formato de coma flotante, esto es, contienen un valor de mantisa y otro de exponente, que, para entendernos, codifican números con decimales.

         Aunque el formato en que se almacenan estos números en un ordenador es binario, podemos ver cómo es posible almacenar números muy grandes o muy pequeños mediante dos enteros relativamente pequeños, usando potencias en base 10. Por ejemplo, tenemos para la mantisa un valor entero, *m*, entre -0.99 y 0.99, y para el exponente un valor, *e* entre -9 y 9.

Tipo "bool" o Booleano:

         Las variables de este tipo sólo pueden tomar dos valores **true** (verdadero) o **false** (falso). Sirven para evaluar expresiones lógicas. Este tipo de variables se puede usar para almacenar respuestas, por ejemplo: ¿Posees carné de conducir? O para almacenar informaciones que sólo pueden tomar dos valores, por ejemplo: qué mano usas para escribir. En estos casos debemos acuñar una regla, en este ejemplo, podría ser diestro->**true**, zurdo->**false**.

Tipo "double" o coma flotante de doble precisión:

         Las variables de este tipo almacenan números en formato de coma flotante, mantisa y exponente, al igual que **float**, pero usan una precisión mayor, a costa de usar más memoria, claro. Son aptos para variables de tipo real. Usaremos estas variables cuando trabajemos con números grandes, pero también necesitemos gran precisión. El mayor espacio para almacenar el número se usa tanto para ampliar el rango de la mantisa como el del exponente, de modo que no sólo se gana en precisión, sino también en tamaño.

         Al igual que pasaba con los números enteros, no existe un tamaño predefinido para cada tipo en coma flotante. Lo que sí sabemos es que el tamaño de **double** es mayor o igual que el de **float** y el de **long double** mayor o igual que el de **double**.

Tipo "void" o sin tipo:

**void** es un tipo especial que indica la ausencia de tipo. Se usa para indicar el tipo del valor de retorno en funciones que no devuelven ningún valor, y también para indicar la ausencia de parámetros en funciones que no los requieren, (aunque este uso sólo es obligatorio en C, y opcional en C++), también se usará en la declaración de punteros genéricos, aunque **esto lo veremos más adelante detalladamente**.

Tipo "enum" o enumerado:

Este tipo nos permite definir conjuntos de constantes enteras, llamados datos de tipo enumerado. Las variables declaradas de este tipo sólo podrán tomar valores dentro del dominio definido en la declaración, lo veremos más adelante detalladamente.

Estos son los ejemplos de alguna de las variables disponibles en el Lenguaje de programación C++, igualmente las iremos viendo paso a paso a medida que vayamos avanzando en el cuatrimestre a no desesperar.

Continuamos con la explicación del programa:

Quinto: **ASIGNACION DE VALORES ->**a los enteros dato1 y dato2 se le asignan los valores 20 y 10 respectivamente para resolver las ecuaciones.

         Sexto: **MUESTREO POR PANTALLA DE LOS RESULTADOS ->** utilizaremos la función "printf" para mostrar por pantalla los resultados obtenidos de las ecuaciones aritméticas, el desarrollo de la sintaxis es el siguiente todo lo que está dentro de las comillas ->" " es tenido en cuenta como texto salvo aquellas secuencias que este antecedidas por los símbolos ( y %) dichas secuencias y argumentos van acompañado de una letra que caracterizan las siguientes funciones:

         Séptimo: la función getchar() que nos permite hacer una pausa en pantalla para poder ver los resultados a la espera de que presionemos una tecla cualquiera.

Con esto terminamos la explicación del ejercicio realizado como ejemplo en la clase anterior y que nos sirve de muestra para las siguientes ejercitaciones.

           \* Por favor copien la sintaxis que utilizamos en la ejercitación de la imagen de arriba y ejecútenlo en el DEV C++ que instalamos en la clase pasada si aún no lo hicieron, de esta forma el programa nos permitirá observar los resultados que se muestran por pantalla, OJO hay que copiar todos los signos y no olvidarse ningún "%" "/" o ";" ya que de otra forma el programa nos mostraría un error.

           Para realizar un programa nuevo recuerden que abren el DEV C++ se van a Archivos -> Nuevo -> Archivo Fuente o presionan (Ctrl+N). Una vez finalizada la programación vamos a compilar yendo a Ejecutar -> Compilar o presionan la tecla (f9) y por último Ejecutar -> Depurar o presionan la tecla (f5). Pueden también cambiarles los valores asignados a dato1 y dato2 para que vean que resultados les va dando.

           \*Observaron como utilizamos la función getchar(); para la pausa? en realidad getchar es una función que permite el ingreso de un carácter y queda a la espera del mismo, es decir que hasta que no se presione una tecla y no se obtenga un ingreso el programa queda en stand by, pero entonces que paso con system("PAUSE"); como en el ejercicio no agregamos la librería   #include <stdlib.h> si pusiéramos este comando en la depuración nos mostraría un [Error] 'system' was not declared in this scope. Incluyamos entonces stdlib a nuestras librerias de funciones y agreguemos en la ejercitación un system ("PAUSE"); en vez de getchar para que nos permitirá hacer una pausa de la forma correcta.

           Ahora utilizaremos los siguientes ejercicios con ***ERRORES***para resolver y probar con el Dev-C++. Esta ejercitación **no es obligatoria** y **no es necesario entregarla** pero sirve para que practiquemos errores comunes de sintaxis que suelen darse al realizar programas y es de sana practica realizarlos sobre el compilador para probar como se deben escribir correctamente las líneas de código (recuerden que si hay errores en " ; " o faltas de " }" el programa no les va a compilar).

EJERCITACION:

**Los siguientes ejemplos contienen errores encuentre los problemas de sintaxis y resuélvalos, para compilar los ejercicios correctamente. Recuerden que las "" y los; son parte de la sintaxis y que se deben realizar pausas para poder observar todo por pantalla.**

**1) Se espera mostrar por pantalla el valor de la variable entera "numero".**

#include <stdio.h>  
main()

{    int numero;    numero = 2;    printf( "El valor es %i" Numero );

**2) Se espera mostrar por pantalla el valor de las variables enteras "a" y "c".**

#include <stdio.h>

main()

{ int a, c; a = 5; c += a + 5; prnt("Elll valor de c= %d",c);}

**3) Se espera mostrar por pantalla el valor de las variables enteras "a", "b" y "c".**

include <conio.h>

#include <stdio.h>

main()

{ int a, b, c; a = 5; b = ++a; \*/++a significa que le suma uno al valor contenido en la variable a/\* c = ( a + 5 \* 2 ) \* ( b + 6 / 2 ) + ( a \* 2 ); printf( "%i; %i, %i", a, b, &c ); }

           Con esto damos por concluido todo lo que respecta a printf y enunciado de variables. A continuacion vamos a leer el tema 4 de la bibliografia, para comenzar a trabajar con declaraciones del tipo char, signed, unsigned, short, enum, float, doubles, etc..

\*\*Para aclarar dudas y dialogar sobre como será la materia me gustaría que nos encontráramos el próximo lunes 3 de abril a las 20:00 hs. a 20:30 hs. en un meet (por favor sean puntuales):

meet.google.com/fyi-zyxz-xho

El encuentro **NO**es de carácter obligatorio, pero si servirá para que nos podamos conocer y además puedan entender la dinámica de la materia. Les dejo un saludo y seguimos en contacto. Recuerden que hay un FORO DE CONSULTAS donde pueden dejar mensajes o también puede escribirme a mi correo eshimoyama@esbabarrionorte.edu.ar.