	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERIA** **ESCUELA DE COMPUTACIÓN**
Ciclo II	**Programación de Algoritmos** **Guía de Laboratorio No. 4** **Introduccion a C++**

I. RESULTADOS DE APRENDIZAJE

Que el estudiante:

- Se familiarice con el entorno de desarrollo del compilador de Lenguaje C++
- Identifique la estructura general de un programa en C y los tipos de datos de las variables.
- Declare variables y constantes dentro de un código fuente de C.
- Escriba nuevos programas (software) con C++ para solucionar problemas.

II. INTRODUCCIÓN

Historia del Lenguaje C

El lenguaje C fue desarrollado por Dennis Ritchie en 1972. Este lenguaje se considera un lenguaje débilmente tipificado de nivel medio, pero con muchas características de bajo nivel.

Dispone de las estructuras típicas de los lenguajes de alto nivel porque puede ocultar los detalles de la arquitectura de la computadora y por tanto incrementar la eficiencia en la programación.

Pero, a su vez, dispone de construcciones del lenguaje que permite un control a muy bajo nivel, tanto así, que algunos compiladores ofrecen extensiones del lenguaje que permiten mezclar código en lenguaje ensamblador con código en C.

Lenguaje C++

El lenguaje C++ se desarrolló como un súper conjunto del lenguaje C y fue desarrollado por el Dr. Bjarne Stroustrup entre 1983 y 1987.

C++ mantiene todas las características del lenguaje C, pero además proporciona la capacidad de desarrollar programas orientados a objetos (POO), almacenando variables y funciones en módulos llamados clases.

¿Cómo crear un programa utilizando C++?

Para digitar los códigos de estos programas se requiere un editor de C/C++ (o un simple editor de texto), y para ejecutar un programa escrito en lenguaje C/C++ se necesita de un Compilador. En la actualidad,

existe una gran cantidad de compiladores y editores para ejecutar programas en C/C++, ejemplos de ellos son: Borland C, Turbo C, Dev C++, GCC, Visual C++, etc.

Un programa en C++, casi siempre, pasa a través de 6 fases para su ejecución, las cuales son: **editar**, **pre procesar**, **compilar**, **enlazar**, **cargar** y **ejecutar**. Estas fases se describen a continuación:

a) La edición es el proceso en el que el programador digita el código de programa en un editor para C++. A este código se le conoce como código fuente.

b) El pre procesamiento consiste en modificar el código fuente de C según una serie de instrucciones conocidas como directivas de pre procesamiento con el propósito de simplificar el trabajo del compilador.

c) La compilación es el proceso de pasar el código fuente ya pre procesado a código objeto.

d) La fase de enlace consiste en unir los códigos objeto de los distintos módulos y bibliotecas externas (bibliotecas de sistema) con el fin de generar el programa ejecutable final.

e) Finalmente, una vez generado el código ejecutable se carga y se ejecuta.

Estructura de un programa en C++

Para elaborar un programa básico en lenguaje C++, se necesita definir 2 partes fundamentales (Ver Tabla 4.1):

a) **Encabezado de programa**:

En esta parte se definen los archivos extensión .h, los cuales contienen las librerías de funciones que C necesita para ejecutar diferentes tareas/cálculos dentro del programa a crear.

b) **Cuerpo del programa**:

Se define con la palabra **main ( )** y una pareja de llaves **{ }**.

Entre estas llaves se definen las variables y las instrucciones utilizadas para entradas de datos, cálculos y salidas de resultados.

Encabezado del programa (archivos de cabecera)	`#include <iostream.h>` `using namespace std:`

Cuerpo del programa (función main)	`main()`
	`{`
	`  cout<<"Hola Mundo";`
	`  return(0);`
	`}`

Tabla 4.1: Partes básicas que componen la estructura de un programa fuente de C++

De un modo más explícito, un programa C puede incluir:

- Directivas de preprocesador;
- Declaraciones globales

- La función main ( )
- Funciones definidas por el usuario
- Comentarios del programa (utilizados en su totalidad).

Palabras reservadas

Existen una serie de indicadores reservados, con una finalidad específica dentro del compilador de C++, y que no pueden utilizarse como identificadores. En la tabla 4.2 se listan algunas de las palabras reservadas mas utilizadas.

_asm	auto	else	private	try
_based	bool	enum	protected	typedef
_cdecl	break	explicit	public	typeid
_dec1spec	case	extern	register	typename
_except	catch	false	Throw	union
_fastcall	char	float	return	unsigned
_finally	class	for	Short	using
_inheritance	const	friend	Signed	using
_inline	const-cast	goto	inheritance	uuid
_int32	continue	if	sizeof	virtual
_int64	declaration	inline	static	void
_int8	default	int	static-cast	volatile
_intl6	delete	long	struct	wchar_t
_leave	directive	man	switch	while
_multiple	dl1export	mutable	template	wmain
_stdcall	dllimport	naked	this	xalloc
_try	do	namespace	thread	
_uuidof	double	new		

Tabla 4.2: Algunas de las palabras reservadas de C++

Bibliotecas / Librerías

C/C++ ofrece un conjunto de funciones estándar que dan soporte a las operaciones que se utilizan con más frecuencia. Estas funciones están agrupadas en **Bibliotecas**, también conocidas como **Librerias**.

Para utilizar cualquiera de las funciones que forman parte de las bibliotecas estándar de C, sólo hace falta realizar una llamada a dicha función.

Las funciones que forman parte de la biblioteca estándar de C, funciones estándar o predefinidas, están divididas en grupos.

Los grupos de funciones estándar más comunes son: entrada/salida estándar, matemáticas, de conversión, diagnóstico, de manipulación de memoria, control de proceso, ordenación, directorios, fecha y hora, cadenas, gráficas, etc.

Todas las funciones que pertenecen a un grupo o libreria, se redactan en un **fichero de cabecera**.

Directivas de procesador

Todas las directivas del preprocesador comienzan con el signo de almohadilla (#), que indica al compilador que lea las directivas antes de compilar la parte (función) principal del programa.

Las dos directivas más usuales son **#include** y **#define**. Observe ejemplos de su uso en la Tabla 4.3

`#include`	Incluye el contenido del archivo nombrado. Estos son usualmente llamados archivos de cabecera (header). Por ejemplo: `#include <math.h>` -- Archivo de la biblioteca estándar de matemáticas. `#include <iostream>` -- Archivo de la biblioteca de Entrada/Salida de C++
`#define`	Define un nombre simbólico o constante. Sustitución de macros. `#define TAM_MAX_ARREGLO 100`

Tabla 4.3: Algunas de las palabras reservadas de C++

Archivos de cabecera / librerias

Existen archivos de cabecera estándar que se utilizan ampliamente, tales como stdio.h, math.h, string.h, iostream.h y se utilizarán otros archivos de cabecera definidos por el usuario para diseño estructurado.

Los archivos de cabecera (archivos con extensión **.h** contienen código fuente C) se sitúan en un programa C mediante la directiva del preprocesador **#include** con una instrucción que tiene el siguiente formato:

`#include<nombreArch. h>` o también `#include "nombreArch.h"`

Control de entrada y salida de información (iostream.h)

La entrada y salida (E/S) de información se gestiona en C++ mediante la funcionalidad proporcionada por la biblioteca iostream.h. La palabra stream significa "flujo" o "corriente" en inglés, para dar una indicación de cómo funcionan los aspectos de E/S:

- Los dispositivos de salida (ej.: monitor) se modelan como canales/tuberías a los que va llegando un flujo de información.
- Del mismo modo, los dispositivos de entrada (el teclado) son fuentes de las que surge un flujo de información.

Tanto en un caso como en otro, la librería iostream brinda los operadores para insertar información en el flujo de salida, o extraer información del flujo de entrada.

Sobre dichos flujos se aplican las operaciones de extracción (operador >>) y de inserción (operador <<).

Función *cout*

El flujo de salida (asociado a la pantalla) se administra con el operador cout. La sintaxis general para el uso de cout es:

<Flujo de salida>**cout<<** (***<expresión>*** | **endl**) {**<<** (***<expresión>*** | **endl**)}**;**

El operador cout imprime una cadena de caracteres sobre la pantalla del ordenador, formada por expresiones (variables o cadenas de caracteres) y saldos de línea (generado con operador endl).

Operador cin

La biblioteca iostream dispone automáticamente del flujo asociado a la entrada a través de teclado con el operador cin. Su sintaxis de uso es:

<Flujo de entrada>**cin>>*<ident>*** {***>><identX>***}**;**

Con cin se leen datos de la entrada estándar y se almacenan en variables que recibe como argumentos (ident). Por Ej.: **cin>>variable1>>variable2>>variable3**;

C++ proporciona otros mecanismos de entrada/salida menos sofisticados que permiten leer o escribir simplemente un carácter por la entrada/salida estándar. Esto se realiza mediante las funciones **cin.get(variable_char)** y **cout.put(variable_char)**.

Tipos de Datos

C no soporta un gran número de tipos de datos predefinidos, pero tiene la capacidad para crear sus propios tipos de datos. En la Tabla 4.4, se presentan los principales tipos de datos básicos, sus tamaños en bytes y el rango de valores que puede almacenar.

Nombre del tipo de dato	Bytes de Memoria	Rango de valores
`int`	*	Depende del sistema
`unsigned int`	*	Depende del sistema
`_int8`	1	-128 hasta 127
`_int16`	2	-32,768 hasta 32,767
`_int32`	4	-2,147,483,648 hasta 2,147,483,647
`_int64`	8	-9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807
`char`	1	-128 hasta 127
`unsigned char`	1	0 hasta 255
`short`	2	-32,768 hasta 32,767
`long`	4	-2,147,483,648 hasta 2,147,483,647
`float`	4	+/-3.4E+/-38(7 dígitos)
`double`	8	+/-1.7E+/-308(15 dígitos)

Tabla 4.4: Tipos de datos de C++ y los rangos de valores que soportan.

Los tipos de datos básicos/fundamentales en C++ son:

- Enteros: (números completos y sus negativos), de tipo int.
- Variantes de enteros: tipos short, long y unsigned.
- Reales: números decimales, tipos float, double o long double.
- Caracteres: letras, dígitos, símbolos y signos de puntuación, tipo char.

Las palabras `char`, `int`, `float` y `double` son reservadas y especifican ***Tipos de datos***. Cada tipo de dato tiene su propia lista de atributos que definen las características del tipo y pueden variar de una máquina a otra.

Los tipos char, int y double tienen variaciones o modificadores de tipos de datos, tales como `short`, `long`, `signed` y `unsigned`, para permitir un uso más eficiente de los tipos de datos.

Variables y Constantes

Variables

Son las posiciones de memoria en donde se almacenan cada uno de los datos a utilizar por el programa y estas pueden ser de un tipo de dato particular.

Constantes

Si un dato no va a cambiar su valor durante la ejecución del programa, generalmente se declara como un valor constante (con la sentencia const).

¿Cómo crear Identificadores de variables y constantes?

Hemos visto que una **variable** es un lugar en la memoria de una PC, en donde se almacena un dato en nuestro programa, y esta puede ser de un tipo de dato particular.

Cada variable tiene un nombre (identificador), el cual debe cumplir estas restricciones:

1. Se forma por una secuencia de letras y dígitos, aunque también acepta el carácter de subrayado.
2. No acepta los acentos, el espacio en blanco, ni la ñ / Ñ.
3. El primer carácter no puede ser un número (digito), sino que debe ser una letra o el símbolo _ .
4. C++ hace distinción entre mayúsculas y minúsculas, es Case Sensitive.

Para declarar una variable debe hacer uso de uno de los tipos de datos mencionados en la tabla 4.4 (`int`, `char`, `float`, etc.) en el área del cuerpo del programa siguiendo esta sintaxis:

NombreTipoDato ListadeVariables;

En donde ListadeVariables es el nombre de una o muchas variables, cada una de las cuales se les define un Tipo de dato en común (NombreTipoDato). Si se declaran 2 o más variables, estas se separan por comas (,).

También puede indicar un "valor inicial" que tendrá una variable al declararla, agregando el operador de asignación (=) y luego el valor. Cualquier variable sin un valor inicial, tendrá un valor aleatorio asignado por la PC cuando se ejecute el programa final. Para terminar una línea de declaración de variables se utiliza un (;)

Ejemplos de declaraciones de variables:

```
int A, B, C; //declaras 3 variables enteras sin valor inicial
//Crea una variable entera llamada numX y tendrá un valor inicial de -8
int numX=-8;
/*Se crean 4 variables decimales.
Sólo a las variables H y num3 se les define un valor inicial*/
float H=4.6, num2, R, num3=0.08;
```

Una vez declarada una variable, la puede utilizar en su programa, teniendo en cuenta las restricciones del rango y que se pueden hacer con ella, gracias a su tipo de dato definido.

Declarar el identificador de una constante

Si un dato no va a cambiar su valor durante la ejecución completa del programa, generalmente se declara como un valor constante. Lenguaje C++ permite 2 maneras de definir una constante:

Método 1: sentencia #define

Los pasos para definir una constante bajo el lenguaje C original, son los siguientes:

1	Ubica el cursor en una línea vacia, justo donde declara las librerías (con #include), dentro del encabezado (inicio) del programa.	
2	Escribe la directiva #define, el identificador de la constante y su valor fijo (valor constante), con esta sintaxis: `#define<identificador><valor>`	Ejemplo: `#define PI 3.141592`

3	Si necesita más constantes, escribe las otras constantes en las líneas siguientes, cuidando de escribir solo una definición por línea, por ejemplo:	Ejemplo:
		`#define PI 3.141592`
		`#define Dolar 8.75`

Método2: sentencia const

En lenguaje C++, los identificadores se declaran constantes por medio de la palabra reservada `const`.

Observe los siguientes ejemplos:

```
const double PI = 3.1416;
const char BLANCO = ' ';
const double PI_EG = PI;
const double DOBLE_PI = 2*PI;
```

Tipos de Operadores

Un operador es un símbolo que expresa que ha de realizarse una operación específica a un valor o a una pareja de valores, retornando un resultado único.

Al igual que los operadores utilizados al elaborar el pseudocódigo y/o diagrama de flujo, lenguaje C tiene una serie de categorías de operadores, los cuales se explican a continuación:

- ✓ Operadores de asignación
- ✓ Operador de agrupación
- ✓ Operadores aritméticos
- ✓ Operadores lógicos
- ✓ Operadores relacionales

Formato de los datos impresos en pantalla

Se puede alterar el formato de impresión de los datos en pantalla generados por cout mediante el uso de la funcionalidad que proporciona la biblioteca **iomanip**. Mediante la misma es posible por ej.: indicar el número de decimales de precisión con el que se quiere escribir un número en punto flotante, el número de espacios que vamos a emplear para escribir un dato, caracteres de relleno, etc.

Esto se realiza mediante la inserción en el flujo de salida de modificadores, que afectarán a los datos que se introduzcan a continuación.

Alguno de estos modificadores son los siguientes:

- ✓ **setprecision( )**: indica el número de dígitos significativos en un dato en punto flotante. Afecta a todos los datos que se introduzcan con posterioridad.
- ✓ **setw( )**: define el número de espacios que se emplearán para escribir un dato, alineando al mismo a la derecha dentro de dicho espacio. Si el espacio requerido es mayor que el indicado, el modificador se ignora. Sólo afecta al dato que se indica a continuación.
- ✓ **setfill( )**: especifica el carácter de relleno que se empleará para los espacios no usados al escribir un dato, según un modificador setw().

Redaccion de Comentarios

Usted puede decirle al compilador de C que ignore a propósito, determinados segmentos de líneas de su programa, para que así, utilice estos bloques para redactar ayudas, explicaciones o comentarios sobre determinados bloques del código fuente de su programa.

Hay 2 formas de crear comentarios sobre segmentos de su código fuente de C, las cuales son:

a) Con dos plecas (//) se ignora lo escrito hacia la derecha de la línea actual, comenzando por el inicio //.

b) Con 2 parejas de símbolos / y un * utilizados de esta forma: `/* comentario */`.

Se ignora todo el código (de una o muchas líneas) escrito entre el `/*` (de apertura) y el `*/` (de cierre del comentario).

III. MATERIALES Y EQUIPO

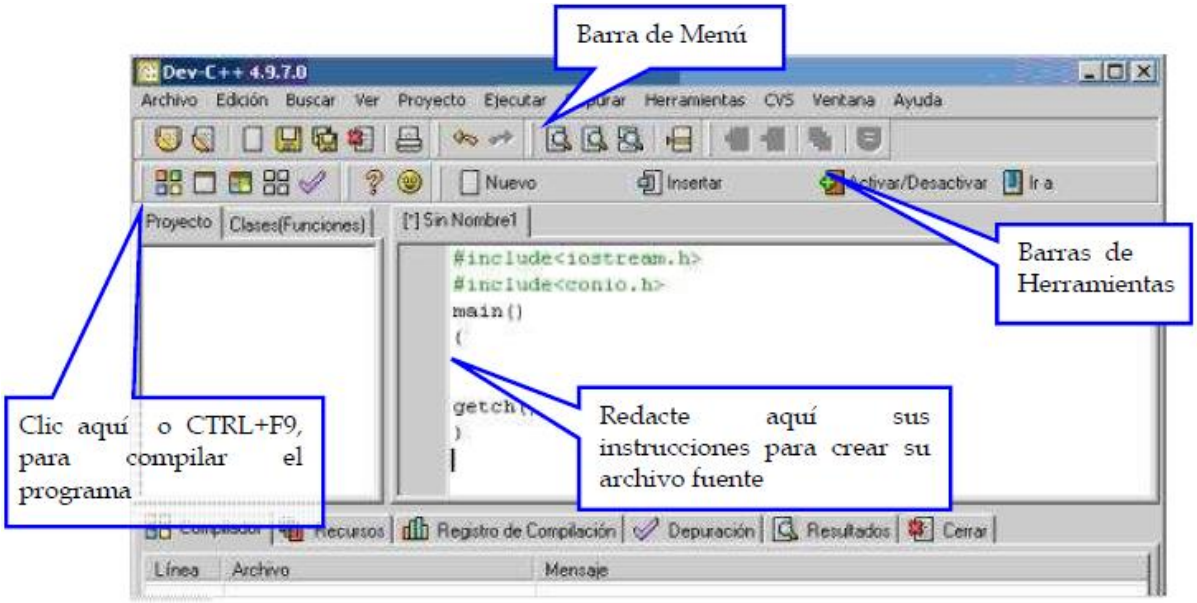
No.	Requerimientos	Cantidad
1	Memoria USB	1
2	Compilador y editor de C++	1

IV. PROCEDIMIENTO

Parte 1: Creando los códigos fuente (archivos .CPP) de los programas

1. Para cargar el compilador Dev-C++ siga la siguiente secuencia de pasos:
   a. Haga clic en botón Inicio de Windows, luego opción *Todos los Programas*
   b. Seleccione programa **Bloodshed Dev C++**.
   c. Dev C++ se ejecutan abriendo el entorno de desarrollo. Luego aparecerá la ventana del compilador vacía

2. A continuación se describe la ventana principal del programa C++ que será nuestro entorno de trabajo para el resto del curso:



3. Para crear un nuevo programa (código fuente), haga clic en Archivo y luego en opción Nuevo, y por último, haga clic en Archivo fuente.
4. Cree una carpeta llamada **PALguia04de*Carnet*** (reemplace ***Carnet*** por su correspondiente número de carnet). Aquí guardará todos sus programas a crear durante la práctica.

5. Ahora proceda a crear cada uno de los siguientes programas:

1. Nombre de programa:Ejercicio1

Código fuente C de un programa para mostrar el clásico mensaje "Hola Mundo", utilizando el flujo de salida administrado por el operador **cout** de la biblioteca de E/S (**iostream.h**) de C++

```
#include<iostream>
using namespace std;

#include<stdlib.h>
/*Este es un programa en C para mostrar en pantalla
al mensaje Hola Mundo */
main(){
   cout<<"hola Mundo\n";
   system("pause");
}//fin main
```

6. Guarde su primer archivo de código fuente con el nombre **Ejercicio1.cpp** en su carpeta de trabajo ya sea por medio de opción ***Archivo*** y luego ***Guardar*** o también, busque el respectivo icono en la barra de herramientas

Parte 2: Compilar un código fuente (.CPP) y ejecutarlo

7. Hasta aquí ya tiene su código fuente (archivo .CPP) almacenado. Ahora deberá compilar el archivo .CPP para que el Lenguaje C cree el programa (Ejercicio1.exe) a ejecutar.
8. Pero debe tomar en cuenta que si el cod. presenta errores de sintaxis, este no será compilado ni se podrá ejecutar, por lo que se le recomienda:
   a) Revisar bien la sintaxis, cada elemento es fundamental que se encuentre en la secuencia correcta.
   b) Ser cuidadoso con el uso apropiado de mayúsculas y minúsculas, debido a que C++ es CaseSensitive.
   c) No puede usar palabras reservadas de C como nombres de variables.

9. Para generar la aplicación .exe de su código fuente, ejecute una de las siguientes 3 alternativas:
   - ✓ dé clic en icono 
   - ✓ desde el menú principal, seleccione Ejecutar y busque la opción compilar, o también
   - ✓ presione la tecla (F9)

10. El programa será compilado (traducción a lenguaje de máquina). Luego debe ejecutar el programa, para ello haga clic en el icono  .
11. El mensaje al final es dado por la función **system ( )**. Solo debe presionar una tecla y el programa terminara su ejecución.
12. Modifique el texto del argumento usado por system ( ) con este texto: **"pause>nil"**
13. Vuelva a compilar el código fuente y ejecútelo. La función system ( ) siempre espera que presione una tecla para terminar, pero ya no muestra el mensaje explícito como antes.

Parte 3: Declaración de variables y asignación de sus valores

14. Cree un nuevo código fuente vacío. Para ello, nuevamente haga clic en Archivo, luego Nuevo, y por último, clic en Archivo fuente.
15. En este código fuente, hará uso de las 2 maneras para declarar variables en C++, determinar el valor que C++ le asigna al inicio y como poder alterarlo.

16. En la ventana vacía del nuevo código fuente, digite el siguiente código:

Programa: Ejercicio2.cpp

#	Se demuestra cómo declarar variables y determinar el total de bytes de la RAM que reserva C para c/variable. Finalmente, cómo asignar un valor o dejar su valor predeterminado y la manera como asignar nuevos valores a las mismas.

```
1  #include<iostream>
2  using namespace std;
   #include<stdlib.h>
   int main(){
      //declaracion de variables
      char A;//tipo caracter
      float B;//tipo decimal flotante
      double C;//decimal doble flotante

   cout<<"Cuanto espacio me memoria (bytes) ocupa variable A ? "<<sizeof(A);
9     cout<<"\nCuantos bytes se reserva para variable B ? "<<sizeof(B);
      cout<<"\nCuantos bytes de memoria para variable C ? "<<sizeof(C);

      cout<<"\n\nValor actual de variables es el siguiente:";
      cout<<"\nvariable A = "<<A;
      cout<<"\nvariable B = "<<B;
      cout<<"\nvariable C = "<<C;
      //despues, asigna valores solo a variable B y C
   cout<<"\n\nDime un nuevo valor decimal para variable B :";
      cin>>B;//solicita valor a usuario
      C=6.02214e23;//programador asigna valor a C
19
      cout<<"\n\nLuego del cambio, las variables guardan estos valores:";
      cout<<"\nvariable A = "<<A;
      cout<<"\nvariable B = "<<B;
         cout<<"\nvariable c = "<<C;
         system("pause>nil");//genera una pausa en la ejecucion
   }
```

17. Guarde los cambios anteriores con el nombre de archivo **Ejercicio2** y compile este nuevo código.
    Si hay problemas de compilación, compare línea por línea, carácter por carácter del cod. anterior y el digitado en el editor, para corregir los errores en la sintaxis esperada por C++.

18. Ahora, del código fuente anterior, analice los siguientes detalles

- ✓ Compare la declaración de cada variable definida en el código fuente y el correspondiente valor que almacena posteriormente.
- ✓ Observe que la variable **b**, almacena un valor dado directamente por usuario del programa.
- ✓ **Línea 10**: utiliza la función **`sizeof( )`**, para saber la cantidad de bytes que reserva C++ para definir a una variable en la aplicación. La cantidad de bytes depende del tipo de dato de la variable.
- ✓ **Línea 19**: El valor resaltado se conoce como el "número de avogadro". Es un valor muy grande, por lo que se asigna en notación científica.

  Consulte su valor real en internet y compare su escritura normal con la sintaxis usada en el código fuente.

Parte 4: Modificadores de formato de impresión de operador cout

19. Prepare un nuevo archivo de código fuente en blanco. Aquí digite el siguiente código:

#	**Programa: Ejercicio3.cpp** Solicitar 3 números decimales, para mostrar el proceso de suma de los mismos, todos impresos con solamente 2 cifras decimales.
1	`#include <iostream>`
2	`using namespace std;`
3	`#include <stdlib.h>`
	`main(){`
	`float a, b, c, res;`
	`cout<<"digita 3 numeros reales para asi sumarlos:\n";`
	`cin>> a >> b >> c;`
	`res = a + b + c;`
	`//Se formatea la salida con 2 cifras decimales`
10	`//cout.setf(ios::fixed);`
11	`//cout.precision(2);`
	`cout<<"\nSuma resultante es: \n";`
13	`//cout.setf(ios::right);`
14	`//cout.width(10);`
	`cout<< a <<endl;`
16	`//cout.width(10);`
	`cout<< b <<endl;`
18	`//cout.width(10);`
	`cout<< c <<"+"<<endl;`

```
20  //cout.width(11); cout.fill('-');
    cout<<".";
22  //cout.width(10); cout.fill(' ');
    cout<<endl<< res <<"\n\n";
    system("pause");
    } //fin main
```

20. Compilar y ejecutar el programa anterior. Ingresar 3 números decimales cualquiera, con diferentes cifras decimales y evalúe como se presenta cada número y el resultado.

    Observe que no queda claro que tan grande o pequeño es cada número respecto al resto de números y el valor de la respuesta.
21. Desplace el cursor hasta el inicio de la línea 10 y reactive la línea de código, borrando solamente la pareja de plecas (//)
22. Vuelva a compilar y ejecutar el programa. Ingrese los mismos 3 valores de la prueba anterior.

    Analice como se presentan los números almacenados en las variables, gracias a la activación de la bandera **ios::fixed**
23. Active la línea 11, vuelva a compilar el programa y digite 3 valores numéricos decimales.

    Identifique el cambio que genera la función **precision( )** del operador cout.
24. Active las líneas 13 y 14, recompile y pruebe el programa nuevamente. Determine ¿Cuál es la función de esta pareja de líneas en el código fuente?
25. Active las líneas (16, 18, 20 y 22) del código fuente y compile nuevamente.

    Determine la función hecha por la función width y luego de fill del operador cout.
26. Guarde los cambios del archivo.

Parte 5: Resolviendo un problema real

27. Prepare un nuevo archivo de código fuente y guardarlo bajo el nombre **Ejercicio4.cpp**
28. A continuación, le resolverá el siguiente problema a un estudiante de Bachillerato.

Problema: "Calcule el área total de un cono cualquiera, cuyas medidas de altura y radio son dadas en centímetros y la respuesta se obtiene en centímetros cuadrados"

29. Para comenzar la solución, en la Imagen 4.1, se ofrece la información correspondiente al cálculo del area total de un cono.

30. En la ventana de código del nuevo programa, digite el bloque de código indicado en la tabla del **Ejercicio4.cpp**, teniendo en cuenta las siguientes aclaraciones:
    - A lo largo de todo el código, existen una serie de comentarios de programador. Cada uno indica las sentencias de código que usted debe implementaren las siguientes líneas.

Por ejemplo, observe las líneas (7 y 8). Aquí ya se implementó el código que solicita el comentario.

De manera similar, redacte el código faltante, comenzando desde el comentario de la línea 4.

- Observe las ecuaciones mostradas en la Imagen 4.1. Identifique el nombre de cada una de las variables y constantes indicadas en las ecuaciones.

  Estos mismos nombres los usará al definir las variables y constantes en su código fuente, respetando el uso de mayúsculas/minúsculas.

- C++ no admite el operador (`^`) para el cálculo de potencias, por lo que se incluye a la librería `math.h` en el encabezado del código fuente y luego se invoca a la función `pow( )`

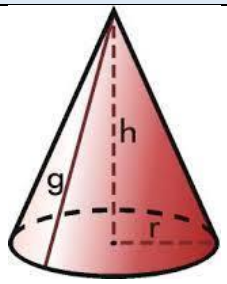
Figura geométrica	Conceptos a tomar en cuenta
 En donde: Altura (**h**) radio de la base (**r**) generatriz (**g**)	Las formulas a tomar en cuenta son las siguientes: $g = \sqrt[2]{r^2 + h^2}$ $AreaTotal = AreaBase + AreaLateral$ $AreaBase = Pi.r^2$ $AreaLateral = Pi.r.g$

Imagen 4.1: Cálculo del área de un cono

#	Programa: Ejercicio4.cpp
1	`#include<iostream>`
	`usingnamespace std;`
3	`#include<stdlib.h>`
4	`//declare una constante llamada Pi (con la sintaxis de C)`
5	
6	`main(){`
7	`//defina aqui a cada una de las variables de entrada`
8	`float r,h;//medida de radio y altura del cono`
	`//declare a continuacion a las variables de salida`

```
//ahora declare a cada una de las variables de procesos

cout<<"\nCalculo de area total de un cono\n\n";
/*
Redacte el codigo para solicitar al usuario el valor de cada dato de
entrada
  (indicando que las unidades de medida seran en centimetros)
  */

//Redacte los calculos para determinar el area total del cono


//Borre al contenido de la pantalla


/*
  Muestre el valor de area final, cumpliendo el formato
de presentacion descrito a continuacion

         Medidas de cono:

  Radio: 2.00 cm, Altura: 4.00 cm

  Area total: 40.67 cm. cuadrados

  */

  system("pause>nil");
}//fin main
```

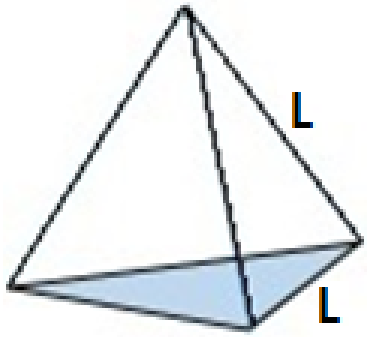
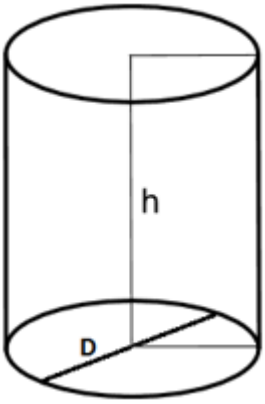
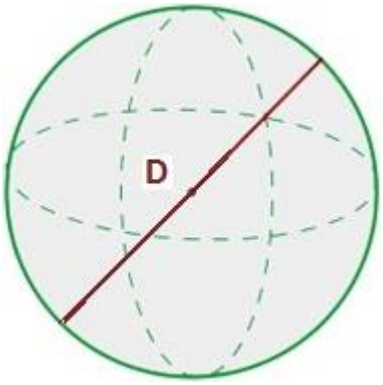
31. Guarde los cambios, compile el programa y compruebe que se obtienen los resultados esperados por cada combinación de datos de entrada.
32. Una vez desarrollado los ejemplos anteriores, ahora proceda a crear un código fuente diferente para soluciona a cada uno de los problemas listados a continuación.
33. Al finalizar la redacción y comprobación de las soluciones, entregue la carpeta de trabajo a su instructor, de acuerdo a como este lo indique.

V. ANÁLISIS DE RESULTADOS

A continuación, se ofrece un listado de problemas, cuya solución se debe implementar bajo archivos de códigos fuentes de Lenguaje C++.

PROBLEMA 1

Determine las áreas totales de cada uno de los siguientes cuerpos geométricos:

Tetraedro	**Cilindro**	**Esfera**
		
Se necesita la medida de una de las aristas (L).	Requiere el valor del diametro (D) de la base y la altura (h)	Diametro (D)

Para cada cuerpo geométrico, debe solicitar solamente a las medidas descritas en la tabla anterior.

PROBLEMA 2

La velocidad de impacto (en metros/seg) de una pelota de hule soltada verticalmente desde una altura determinada (en metros), está dada por la fórmula:

$$velocidad = \sqrt{2.Gravedad.Altura}$$

Esta pelota rebota a 2/3 de la altura anterior desde la cual se suelta.

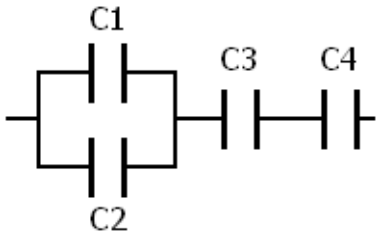
Si este experimento se realizara en nuestra Luna Terrestre, elabore una aplicación en C que determine la siguiente información:

- Calcule la velocidad de impacto de los primeros 2 rebotes
- La altura que alcanza en el 2do rebote.

Recuerde que las medidas de gravedad de planetas o lunas se consideran valores fijos/constantes.

PROBLEMA 3

Ayude a un estudiante de electrónica a obtener el capacitor equivalente que reemplazaría al conjunto de capacitores mostrados en el siguiente diagrama:



Los diferentes capacitores se miden en microfaradios.

El capacitor C43 del circuito tiene un valor fijo de 50 microfaradios.

PROBLEMA 4

Una empresa distribuidora de combustibles almacena el combustible Diesel y de Gasolina de una sus estaciones de venta en 2 tanques subterráneos idénticos (ver diagrama); un tanque diferente por cada tipo de combustible.

Determine:

- El volumen máximo (en litros) de Diesel que puede ser almacenado en el 1er tanque.
- Luego, obtenga la cantidad límite de Gasolina (en Kilogramos) que puede ser almacenada en el 2do tanque.

Cada tanque es cilíndrico y sus medidas están dadas en metros.

PROBLEMA 5

Realice una aplicación que genere la factura por la venta de 3 productos diferentes.

De cada producto, se conoce su nombre, el precio por unidad y la cantidad de unidades vendidas. A la venta final se debe aplicar un impuesto al valor agregado (IVA) del 13%

Muestre el detalle de venta e indique el monto total a pagar por concepto de IVA, cumpliendo el siguiente ejemplo de presentación de factura:

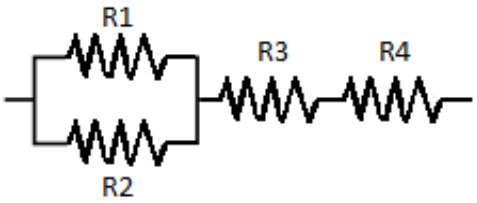
```
FACTURA

PRODUCTO       | unit ($)   |   cantidad   | costo ($)  |
cd             |       0.85 |            3 |       2.55 |
calculadora    |      23.10 |            2 |      46.20 |
TV plasma      |     141.99 |            1 |     141.99 |

Monto sin iva    $   190.74
+ IVA            $    24.80
FACTURA A PAGAR  $   215.54
```

PROBLEMA 6

Ayude a un estudiante de electrónica a obtener el resistor equivalente que reemplazaría al conjunto de resistencias mostrados en el siguiente diagrama:

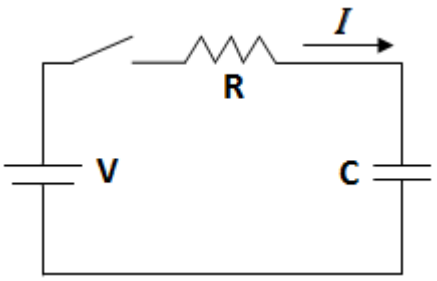


Las diferentes resistencias se miden en kilo-ohmios y R3 tiene un valor fijo de 60 kilo-ohmios.

PROBLEMA 7

La corriente eléctrica **I** (en amperios), que fluye a través del circuito mostrado aquí, esta dado por la siguiente ecuación:

$$I = \frac{V.e^{\frac{-t}{R.C}}}{R}$$



En donde **V** es el voltaje de la batería (en voltios), **R** es la resistencia (en ohmios), **C** es el valor del capacitor (en faradios), **t** es el tiempo (en segundos) luego de cerrar el interruptor y **e** es la constante de Euler (2.718281...).

Escriba un código fuente que solicite a usuario los valores necesarios para calcular y mostrar en pantalla a la corriente (I) que circula por la resistencia. La medida de la corriente debe ser presentada en miliamperios. Por ejemplo:

Cuando el voltaje de la batería es de 20v, la resistencia de 8100 ohmios, el capacitor de 0.000018 faradios y transcurren 0.31 segundos luego de cerrar el interruptor, por la resistencia circulan 2.46914 miliamperios.

PROBLEMA 8

Determinar los "porcentajes de votos" alcanzados en una elección del municipio de Aleluya, en la que participaron 3 candidatos. De cada candidato participante en la elección se conoce su nombre completo y el total de votos obtenidos en la elección.

Los resultados solicitados se deben presentar bajo el siguiente formato de ejemplo:

```
Resultados finales

|CANDIDATO          | VOTOS | PORCENTAJE |
|Iris C             |  1200 |    32.5 %  |
|Raquel B           |  1504 |    40.8 %  |
|Ivania M           |   985 |    26.7 %  |

                      3689     100.0 %
```

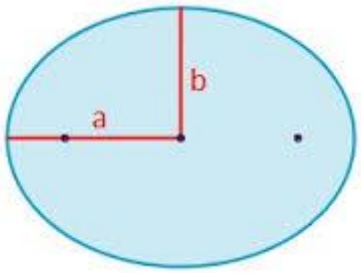
Los porcentajes solicitados se deberán presentar con una exactitud de 1 cifra decimal.

Para manejar texto, investigue cómo funciona el método getline del operador cin: `cin.getline()` y luego el uso de la función `fflush()`

PROBLEMA 9

La siguiente formula, creada por el científico indio Ramanujan, permite determinar un valor muy aproximado del perímetro de una elipse:

$$perimetro \approx \pi \left[3(a+b) - \sqrt{(3a+b)(3b+a)}\right]$$



En donde a y b son el radio mayor y menor, respectivamente. Ambos radios son medidos en centimetros.

Determine la longitud del perímetro de una elipse y muestre su medida en centimetros y tambien en pulgadas.

Guía de Laboratorio No. 4:
RÚBRICA DE EVALUACIÓN

Actividad a evaluar: ANÁLISIS DE RESULTADOS

Formar grupos entre 3 a 5 estudiantes, llenar esta hoja de evaluación y entregarla a su docente.

Su instructor seleccionará 3 problemas del análisis de resultados, para ser resueltos apropiadamente por el grupo

Lista de Integrantes:

CARNET 1	CARNET 2	CARNET 3	CARNET 4	CARNET 5

Problemas a resolver:

Criterio a evaluar	¿Prob 1?	¿Prob 2?	¿Prob 3?	¿Prob 4?	PROM.	Puntaje
(20%) Define las variables de entrada y salida esperadas. Define y usa al menos un valor constante.						
(20%) Define las expresiones correctas para operar a los datos de entrada.						
(20%) Se cumplen cada una de las restricciones descritas en el problema						
(15%) Diálogo de la aplicación con usuario es apropiado. Se documento internamente al código fuente.						
(25%) Se obtiene a c/u de los resultados solicitados						
					Nota:	