	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERIA** **ESCUELA DE COMPUTACION**
CICLO I	***GUIA DE LABORATORIO # 10*** **Tema de la Practica:** Creación de Funciones **Lugar de Ejecución:** Centro de computo **Tiempo Estimado:** 2 horas y 30 minutos **MATERIA:** Programación de Algoritmos (PAL404)

I. OBJETIVOS

Que el estudiante:

- Comprenda las ventajas del diseño de una aplicación distribuida con funciones
- Aplique el concepto de "modularización de programas"
- Realice la declaración, llamado y definición de cualquier función
- Identifique las diferencias entre "paso por valor" y "paso por referencia".

II. INTRODUCCION TEORICA

¿Qué es una Función?

Una función es un módulo / sección de código independiente y separado del cuerpo principal, que realiza una tarea específica y que opcionalmente puede regresar un valor a la parte principal del programa u otra función o procedimiento que la llame.

Una función tiene las siguientes características:

1. Tiene un único nombre. Al utilizar este nombre en otras partes del programa se pueden ejecutar los enunciados contenidos en la función. A esto se le conoce como llamar a la función. Una función puede ser llamada desde otra función.
2. Una función es independiente. Una función puede ejecutar sus tareas sin interferencia en otras partes del programa.
3. la función ejecuta una **tarea específica**. Una **tarea** es un trabajo discreto que su programa debe ejecutar como parte de su desempeño general, por ejemplo: *mandar una línea de texto a impresión, ordenar una lista o hacer un cálculo matemático*.
4. Cuando un programa llama a una función, el programa puede enviar hacia la función información en la forma de uno o más argumentos. Los **argumentos** son datos que la función puede necesitar para realizar su trabajo. Cuando la función termina su trabajo, la ejecución del programa regresa a la línea siguiente a la llamada a la función. Las funciones pueden enviar información al programa en la forma de un valor devuelto.
5. Cuando un programa llama a una función, la ejecución del código salta a la posición inicial definida por el nombre de la función.

   El programa ejecuta los enunciados contenidos en ésta y posteriormente el programa que llamó a la función

continúa en la línea de código siguiente. Una función bien diseñada ejecuta una tarea específica y fácil de entender. Es recomendable que ante una tarea complicada, dividirla en múltiples funciones y entonces llamarlas a su debido momento.

6. Una función (opcionalmente) devuelve un valor al programa que la llamó. Cuando un programa llama a una función se ejecutan los enunciados contenidos en ésta, si su programa lo requiere, éstos enunciados pueden devolver un valor al programa que llamó a la función.

Existen 2 tipos de funciones que se pueden utilizar dentro de un programa creado en C:

a) Función incluida dentro de una librería (archivo extensión .h) de C
b) Función personalizada, creada por el programador

A. Funciones incluidas en las bibliotecas de C

Las funciones de la biblioteca de C son útiles para resolver cierto grupo de operaciones o cálculos de uso común. Hemos utilizado en guías de práctica anteriores funciones de la biblioteca math.h, stdio.h, stdlib.h, etc. Todas estas librerías permiten al programador el uso de funciones específicas para realizar tareas, y le evitan tener que desarrollar o implementar él mismo esas tareas.

Lenguaje C consta de una serie de funciones almacenadas en las librerías o bibliotecas (archivos .h). A estos archivos de bibliotecas de funciones también se les llama **Archivos de Cabecera**.

Cuando se desea invocar a una función, debe utilizarse su nombre y proporcionarse los argumentos para los parámetros especificados en su definición o prototipo, y esta devuelve un valor que puede asignarse a una variable.

Funciones incluidas en librerías estándar de C.

Algunos de los archivos de cabecera más comunes de C / C++ son los siguientes:

Archivo de cabecera	Descripción
`iostream.h`	Rutinas de control de flujo de entrada y salida para C++
`conio.h`	Entrada y salida de la consola y puertos
`ctype.h`	Funciones para manejo de caracteres
`math.h`	Funciones matemáticas
`string.h`	Funciones para manejo de cadenas de caracteres

Función personalizada, definida por el Programador

En el caso de las funciones definidas por el programador, hay que recordar que su objetivo principal al escribirlas es: **"Divide un programa complejo en un cierto número de módulos más pequeños, y cada uno realizará una tarea específica más sencilla"**.

DECLARACIÓN DE UNA FUNCION (PROTOTIPO)

Para utilizar una función en un programa se requiere en primer lugar declararla y después definirla.

A la declaración de una función se le llama también el **prototipo de la función** y como todo enunciado ejecutable en un programa C++ debe terminarse con un símbolo de punto y coma (**;**). La declaración de la función le indica al compilador:

a) El nombre de la función a crear

b) Cada uno de los parámetros pasados a la misma.

c) El tipo de dato devuelto al finalizar las tareas de la función

```
Tipodato Nombrefunción(lista de parámetros formales);
```

DEFINICIÓN DE LA FUNCION

La definición de una función es en sí la función misma. Está compuesta en su primera línea de código por el encabezado, el cual debe ser idéntico al prototipo de la función, pero no se utiliza el punto y coma.

Enseguida del encabezado se redacta el **Cuerpo de la Función**, encerrados entre llaves { }, el cual contiene el código C para definir los pasos a ejecutar por la función. Si la función devuelve SOLAMENTE un valor, éste se debe especificar al final del cuerpo de la función.

La sintaxis del Cuerpo de una función es la siguiente:

```
Tipodato | void Nombrefunción (lista de parámetros formales)
{
Cuerpo de Instrucciones;
return [dato | var | expresión];
}
```

En donde:

a) **Tipodato** especifica el tipo de dato (Ej.: *int*, *float*, *char* y otros) que regresara la función cuando termine sus tareas.

Si la función no regresa valor al finalizar, se debe usar la palabra clave ***void***, que indica significa "vació".

b) La **lista de parámetros formales** es una lista de declaración de variables separadas por comas (,) que almacenan los valores que recibe la función desde el lugar de llamada de la misma. Estas variables actúan como locales dentro del cuerpo de la función. Si no necesita parámetros, se indica con **void** entre los ( ).

c) La instrucción **return** le indica a la función que finalice su ejecución, y que debe regresar uno y solo un dato a la parte del programa que la llamo. Se puede regresar un valor (2, -7.453, 'r', etc.), una variable o una expresión algebraica, tal como lo muestran los ejemplos siguientes:

```
return (23.56);      return (area);      return (x+15/2);
```

Dentro del cuerpo de la función puede haber una o varias instrucciones `return`, pero tenga en cuenta que cuando se alcanza uno de estos `return`, la función finaliza su trabajo, ignorando el conjunto restante de líneas que falten.

INVOCANDO A UNA FUNCION

Una vez redactado el Prototipo y la Definición de una función, se puede mandar a llamar desde un bloque de código de su programa (de igual forma que las funciones incluidas en las librerías de C).

Cuando se llame a una función, lo hará en una nueva línea, teniendo en cuenta lo siguiente:

+ Si la función no devuelve un valor, basta con escribir el nombre de la función. Ejemplo:

```
Saludar( );
```

+ Si la función devuelve un valor, debe escribir una sentencia de asignación, en la cual, una variable reciba el valor devuelto por la función, ejemplo:

```
resultado = Nombrefuncion ();
```

+ A la derecha del nombre de la función invocada debe escribir una pareja de paréntesis ( ). Si la función necesita de argumentos de entrada-salida, debe reemplazar cada parámetro por un argumento (valor fijo o variable) que sea del mismo tipo de dato.

PARÁMETROS POR VALOR Y POR REFERENCIA

A. En C++ el **paso por valor** significa que al compilar la función y al código que la llama, ésta recibe en sus parámetros a una copia de los valores que se le pasan como argumentos. Las variables reales no se pasan a la función, sino sólo copias de su valor.

B. Cuando una función debe modificar el valor de la variable pasada como argumento y que esta modificación retorne a la función llamadora, se debe **pasar con un parámetro por referencia**.

   En este método, el compilador no pasa una copia del valor del argumento; en su lugar, pasa una referencia, que indica a la función dónde existe la variable en memoria. La referencia que una función recibe es la dirección de la variable. Es decir, pasar un argumento por referencia es, simplemente, indicarle al compilador que pase la dirección del argumento.

   Una limitación del método de paso por referencia es que se pueden pasar sólo variables a la función. No se pueden utilizar constantes ni expresiones en la línea de llamada a la misma.

ALCANCE DE LAS VARIABLES.

Una variable puede ser declarada de acuerdo a un tipo de dato, por ejemplo: variable entera (int), entero largo (long), punto decimal flotante (float), punto decimal de doble alcance (double), carácter (char) y otros.

Una vez vistas la definición y uso de funciones, también se puede clasificar a las variables de acuerdo al lugar donde se declaran dentro del código fuente. A este concepto se le denomina ***Alcance de la Variable***.

Existen 2 tipos de alcance de variables:

a) **Alcance Global**: cuando una variable se declara justo debajo de la última sentencia #include (utilizadas para incluir archivos de cabeceras .H).

b) **Alcance Local**: la variable se declara como un argumento de una función X o si no, se declara dentro del cuerpo de la función X.

Una variable global puede ser utilizada desde cualquiera de las funciones definidas en el código fuente, ya sea `main` (la función principal) u otras funciones.

En cambio, una variable local puede ser utilizada dentro del cuerpo de la función X en donde se halla declarado. El resto de funciones no la reconocerán y generara un error de "variable no declarada".

Tenga en cuenta los siguientes aspectos al usar variables locales y/o globales:

- ✓ Se pueden declarar variables locales que tengan el mismo nombre, pero cada declaración se realiza en funciones diferentes dentro del código fuente. Cada variable puede ser de tipos de datos diferentes, y lenguaje C usará la definición de variable correcta de acuerdo a la función que se invoque en la llamada.
- ✓ Si se declara una variable global y otra con el mismo nombre de alcance local (dentro de una función X), la variable local ocultará a la variable global cuando se quieran hacer operaciones dentro de la función X.

CONSEJO FINAL:

Para crear un código fuente en el cual defina y utilice funciones personalizadas, debe seguir estos 2 pasos:

1. Primero escriba la definición (Cuerpo) de la función.

2. Una vez finalizado el cuerpo de la función, solo copie el encabezado de la misma para formar así el Prototipo, agregándole un punto y coma (;) al final del enunciado.

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Memoria USB	1
2	DEV-C++	1

IV. PROCEDIMIENTO

1. Cree una carpeta denominada **PAL_Practica10**, en la cual guardara los diferentes códigos fuentes solicitados a continuación.

Ejemplo1.cpp

Creación de una función que retorna el mayor de 3 números enteros recibidos en sus parámetros.

```
#include<iostream>
using namespace std;
#include<conio.h>

//Prototipo de la funcion mayorde
int mayorde(int a,int b,int c);

//Definiciones de funciones

main( ){
   int a,b,c;//variables locales de funcionmain
   cout<<"Ingrese tres valores enteros (a,b y c):"<<endl;
   cin>>a>>b>>c;
//Llamada de la funcion mayorde, directamente en el flujo salida de cout
cout<<"El numero mayor es: "<<mayorde(a,b,c)<<endl;

   getch();
}//fin de main()

//Definicion de la funcion mayorde
int mayorde(int a,int b,int c){

    if((a>b)&&(a>c)) return(a);
    if((b>a)&&(b>c)) return(b);
    if((c>a)&&(c>b)) return(c);
    if((a==b)&&(b==c)){
       cout<<"Los 3 numeros son iguales entre si"<<endl;
       return(a);
    }
}// fin funcion mayorde
```

2. Pruebe el cod. anterior con varias combinaciones de números de entrada.

Ejemplo2.cpp

Desarrolle una aplicación que solicite 2 numeros enteros al usuario, para que le muestre la resta y la división entre ambos valores.

Solucion

La solución compleja de este problema se divide en 3 funciones mas simples de implementar, las cuales se describen en la documentación del siguiente código fuente:

```
#include<iostream>
using namespace std;
#include<stdlib.h>

// Prototipos de funciones a implementar
void SolicitarNumeros(void);
void RestarNumeros (int a, int b);
void DividirNumeros (int a, int b);

// Definiciones del Cuerpo de cada funcion

main(){// Función principal main
   cout<<"Prog. para calculo de Resta y Division de 2 numeros\n";
   SolicitarNumeros(); //invoca a funcion

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

void SolicitarNumeros(void){ // Función NotaMateria
/*
No requiere parámetros.
Hace a usuario la solicitud de los 2 numeros a operar.
Luego invoca al resto de las funciones
*/
    int n1, n2; //2 numeros que ingresara usuario

    cout<<"\nDigite 2 numeros cualquiera:\n";
    cin>>n1>>n2;
 //Invoca a cada funcion, enviando los 2 numeros como sus argumentos
   RestarNumeros(n1,n2);
   DividirNumeros(n1,n2);

}//fin funcion SolicitarNumeros

void RestarNumeros (int a, int b){
/*
recibe en 2 parámetros a los 2 numeros que restara. Y muestra el resultado a usuario
*/
   int r;
   r= a-b;
   cout<<"\nResta de "<<a<<" y "<<b<<" es "<<r;
}//fin funcion RestaNumeros

void DividirNumeros (int a, int b){
/*
recibe como parámetros a los 2 numeros que dividira.
```

```
Y luego, muestra el resultado a usuario o indica del error en la operacion
*/
   float d;
   if(b!=0){ //si denominador no es cero
      //hace la division de a / b
      d= (float) a / (float) b;
      cout<<"\nCociente de "<<a<<" y "<<b<<
         " es igual a "<<d;
   }else
      cout<<"\nError, division por cero";
} //fin funcion DividirNumeros
```

3. Compile y ejecute el cod. fuente **Practica9_ejemplo2.cpp** anterior.
   En cada prueba, ingrese 2 numeros enteros cualquiera y confirme que se obtienen las 2 respuestas correctas.

Ejemplo3.cpp

Elabore un programa para registro académico de la UDB, que solicite el nombre de una materia y sus correspondientes notas de periodo. Al final, se muestra la nota final y si aprobó o reprobo el curso.

Tome en cuenta que si la materia no tiene laboratorio, solo se toman 3 notas de periodo, de lo contrario, se solicitan 6 notas (3 notas de teoría y el resto de notas de laboratorio).

```
#include<iostream>
using namespace std;
#include<stdlib.h>
#include<conio.h>

// Prototipos de funciones a implementar
void IngresarMateria(void);
float MateriaSinLab(char nommateria[ ]);
float MateriaConLab(char nommateria[ ]);
void evaluarNotaFinal(float notaf);
// Definiciones del Cuerpo de cada funcion

main(){// Función principal main
   cout<<"Prog. de administracion academica\n";
   IngresarMateria();
   system("pause>nil");
}//fin main

void IngresarMateria(void){
   char nommat[20];
   char tienelab; //  tiene ('s') o no ('n') lab.
   bool listo; //hace repetir indefinidamente al ciclo do-while
   float notafinal;

   cout<<"\nDigite el nombre de la materia: ";
   cin.getline (nommat,20);

   cout<<"\nLa materia "<<nommat<<" tiene laboratorio ?? (Si o No):";

   //asume que la preg. anterior no ha sido respondida
   listo=false;
```

```
   do{ //repite continuamente hasta que indique S o N
      tienelab=getch();
      switch(tienelab){
         case 's': case 'S':
            notafinal=MateriaConLab(nommat);
            listo=true; //modifica bandera
            break;
         case 'n': case 'N':
            notafinal=MateriaSinLab(nommat);
            listo=true; //modifica bandera
      }//fin switch lab
   }while(!listo);

   //Muestra nota e indica si fue aprobada o no
   evaluarNotaFinal(notafinal);
}//fin funcion IngresarMateria

float MateriaSinLab(char nommateria[]){
   float nt1,nt2,nt3;
   float nf;
   system("cls");
   cout<<"\nMateria sin Laboratorio: "<<nommateria;
   cout<<"\n\nIngrese las 3 notas de periodo";
   cout<<"\nPeriodo 1 ? "; cin>>nt1;
   cout<<"\nPeriodo 2 ? "; cin>>nt2;
   cout<<"\nPeriodo 3 ? "; cin>>nt3;
   nf=0.3*nt1+0.35*(nt2+nt3);
   return(nf);
}

float MateriaConLab(char nommateria[]){
   float nt1,nt2,nt3;
   float np1, np2, np3;
   float nf;
   system("cls");
   cout<<"\nMateria con Laboratorio: "<<nommateria;
   cout<<"\n\nIngrese 3 notas de teoria y luego 3 de practicas";
   cout<<"\nPeriodo 1 Teo ? "; cin>>nt1;
   cout<<"\nPeriodo 2 Teo ? "; cin>>nt2;
   cout<<"\nPeriodo 3 Teo ? "; cin>>nt3;
   cout<<"\nPeriodo 1 Lab ? "; cin>>np1;
   cout<<"\nPeriodo 2 Lab ? "; cin>>np2;
   cout<<"\nPeriodo 3 Lab ? "; cin>>np3;
   nf=0.1*nt1+0.15*(nt2+nt3)+0.2*(np1+np2+np3);
   return(nf);
}

void evaluarNotaFinal(float notaf){
   cout<<"\nNota final = "<<notaf;
   if(notaf>=6.0)
      cout<<" (APROBADA)";
   else
      cout<<" (REPROBADA)";
}
```

4. Compile y ejecute el cod. fuente **Ejemplo3.cpp**. Haga pruebas diferentes, de materias con y sin laboratorio.
5. Cree un nuevo cod. fuente, bajo el nombre **Ejemplo4.cpp**.

Ejemplo4.cpp

Crear por medio de funciones la solución al siguiente problema:

Calcula el sueldo final (SF) a pagar a cada uno de los 8 empleados de la empresa LA CONSTANCIA SA.

Cada empleado tiene un Nivel (0, 1, 2, 3) y un Sueldo Base (SB).

Para calcular el SF de un empleado se le aplican los siguientes descuentos y Bono a su SB:

1. Se le aplican los descuentos de Ley: ISSS (3.1%) y Renta (9.3%).

2. Si el Nivel de empleado es 2 o 3, se le descuenta el 11.4% del SB en concepto de Seguro de Vida

3. Según el Nivel del empleado se le calcula un Bono (por el esfuerzo hecho) sobre el SB así:

- Nivel 0: $7.0 exactamente
- Nivel 1: 6.4% del SB
- Nivel 2: 13.94% del SB
- Nivel 3: 21.04% del SB

Al final se muestra los totales ($) en conceptos de:

a) Descuentos del ISSS.

b) Planilla a pagar a los empleados.

El programa en C++ debe tener una función para cada una de estas tareas diferentes:

- **Ingreso del sueldo base y el Nivel de un empleado como parámetros. Luego calcula los 2 descuentos de ley y el seguro de vida (si se aplica), para devolver cada uno de estos resultados en el resto de los argumentos de la función.**
- **Recibe el sueldo base y el nivel del empleado como parámetros, para retornar el Bono.**
- **Mostrar en pantalla el valor de cada descuento, bono y el sueldo final de un empleado. Retornar el Sueldo final.**

```
#include <iostream>
using namespace std;
#include <iomanip>
#include <conio.h>
 //prototipos de funciones
void descuentos(float sb,int nivel, float &isss, float &renta,float &seguro);
float calculobono(float sb, int nivel);
float verempleado(float sb, int nivel, float isss, float renta,float seguro, float bono);

// Definiciones de funciones

main(){// Función principal
 int te=0;//total empleados ingresados
 float sueldob;//sueldo base de un empleado
```

```
 int niv;//nivel de un empleado
 float disss,drenta,dseguro;//cada descuento retenido
 float sbono;

 float montoisss,planilla;

 cout<<"Aplicacion Contable de LA CONSTANCIA SA\n";
 cout<<"ingrese el sueldo base de cada uno de sus 4 empleados.\n";

 montoisss=0.0;
 planilla=0.0;
 for(te=1;te<=4;te++){
  cout<<"\nEmpleado # "<<te<<endl;
  cout<<"\nSueldo base ?? $";cin>>sueldob;
  cout<<"\nNivel (entre 0 a 4) ??";cin>>niv;
  //invoca a funcion que calcula y retorna descuentos
  descuentos(sueldob,niv,disss,drenta,dseguro);
  //invoca funcion que retorna valor del bono
  sbono=calculobono(sueldob,niv);
  montoisss+=disss;
  planilla=planilla+verempleado(sueldob,niv,disss,drenta,dseguro,sbono);

 }//fin for te
 cout<<"\nMonto retenido en concepto de ISSS: $"<<montoisss;
 cout<<"\nPlanilla a pagar: $"<<planilla;

 getch();
}//fin funcionmain

// Definición de Función NotaMateria
void descuentos(float sb,int nivel,float &isss,float &renta,float &seguro){
 //calcula y retorna los descuentos de ley en parametros salida
 isss=0.031*sb;
 renta=0.093*sb;
 seguro=0.0;
 switch(nivel){
  case2:case3:
   seguro=0.114*sb;
 }//fin switch nivel
}//fin funcion descuentos

float calculobono(float sb, int nivel){
 switch(nivel){
  case0:return(7);
```

```
 case1:return(0.064*sb);
 case2:return(0.1394*sb);
 case3:return(0.2104*sb);
 }
}//fin funcioncalculobono

float verempleado(float sb,int nivel,float isss,float renta,float seguro,float bono){
 float sf;
 sf=sb-(isss+renta+seguro)+bono;
 cout<<"\nSueldo Base $"<<setw(8)<<sb;
 cout<<"\nDescuento de: ISSS $"<<setw(5)<<isss<<"\tRenta $"<<setw(5)<<renta<<
  "\tSeguro $"<<setw(5)<<seguro<<endl;
 cout<<"Bono $"<<setw(5)<<bono<<endl;
 cout<<"Sueldoliquido $"<<setw(8)<<sf<<endl;
 return(sf);
}//fin funcionverempleado
```

V. ANALISIS DE RESULTADOS

Elabore un codigo fuente de C++ que solucione a cada uno de los siguientes problemas:

PROBLEMA 1:

Realice una función llamada **TablaPotencias(N)**, que genere en pantalla la tabla de potencias del numero decimal N recibido como parámetro.

Observe a la derecha un ejemplo del resultado de esta funcion.

Recuerde los siguientes principios del calculo de una potencia B^e:

$$B^e = B^e . B^{e-1} \qquad B^0 = 1$$

Restricciones:

- La base N a recibir en el parámetro solo puede variar entre 1 a 4.
- Ante cualquier numero fuera de este rango, la función solo mostrara en pantalla al mensaje:
  `"ERROR, imposible generar tabla de potencia requerida"`
- No puede usar a la librería math.h
- Solo se puede usar operadores de suma (+), resta (-) o los operadores cortos (++ o --). No puede usar al operador de producto (*)
- Puede crear otras funciones complementarias a la función solicitada.
- En caso de necesitar ciclos en cualquiera de las funciones a desarrollar, solo puede utilizar como máximo a un ciclo en su interior.

Tabla de potencias del numero 2:

2 ^ 0 = 1
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
2 ^ 5 = 32
2 ^ 6 = 64
2 ^ 7 = 128
2 ^ 8 = 256
2 ^ 9 = 512
2 ^ 10 = 1024

PROBLEMA 2:

Modifique el Ejemplo 2 del procedimiento, para agregar las siguientes funciones:

- **Calculo de una potencia B^N**
  La base B y el exponente N deben ser enteros. Ambos valores pueden ser positivos, negativos o ceros.
- **Total de combinaciones** $\binom{N}{M}$
  Determine el total de combinaciones que se pueden realizar entre N valores tomados en grupos de M valores.
  Por ejemplo, si se tienen 5 digitos y quieren formarse números formados por 2 digitos, el total de combinaciones sera de $\binom{5}{2}$, retornando un resultado de *60 combinaciones*.

Restricciones:

- No se puede agregar a la librería **math.h** en su solucion.
- En ambas funciones, debe tener en cuenta los casos especiales de solucion y los casos en donde es imposible calcular un resultado, generando el mensaje de justificación apropiado al usuario.

PROBLEMA 3:

Realice un programa en C++ que muestre un menú con las 4 opciones siguientes:

1. Area de un Triangulo equilatero
2. Area de un Rombo (basado en las medidas de su diagonal mayor y diagonal menor)
3. Volumen de un cubo.
4. Salir

Para cada una de las 3 primeras opciones del menú anterior, debe crear una función diferente para resolver el cálculo correspondiente.

Restricciones:

- Cada una de las funciones debe borrar el contenido de la pantalla y solicitar los datos apropiados al usuario según la medida elegida en el menu, asi como mostrar la respuesta.
- Luego que usuario seleccione una opcion y se le muestre la respuesta, debe limpiarse la pantalla y mostrar de nuevo el menu inicial.
- El usuario no podrá salir del programa hasta que haya seleccionado la opción 4.

PROBLEMA 4:

La función trigonométrica Seno de un anguloX se puede calcular por la sumatoria de la siguiente serie infinita de términos:

En donde el angulo x de la formula debe ser proporcionado en grados radianes.

$$Sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$$

Desarrolle una función denominada **Seno( )**, la cual contara con la siguiente pareja de parámetros:

- el primer parámetro indica la medida del angulo.
- el segundo parámetro dice que tipo de medida de angulo (**'s'**: grado sexagesimal o **'r'**: grado radian) se utilizara en el 1er parámetro.

Demuestre desde main que la función Seno( ) es correcta, invocándola con un angulo medido en grados sexagesimales y luego otra llamada pero con su equivalente en grados radianes. Ambos resultados deberán dar el mismo resultado.

Por ejemplo, el valor del seno de 30 grados sexagesimales es idéntico al seno de un angulo de 0.5235 radianes.

Restricciones:

- Para obtener un resultado con una alta precisión, debe utilizar una sumatoria de 12 términos de la serie dada y utilizar tipos de datos decimales de doble precisión (double).
- Solo pueden definirse variables de alcance local.
- No puede utilizar la librería math.h
- Puede crear otras funciones complementarias a la función Seno solicitada.
- En cualquiera de las funciones a desarrollar: en caso de necesitar ciclos, solo puede utilizar como máximo a un ciclo en su interior.

PROBLEMA 5:

Desarrolle una aplicación en C en la cual se defina el cuerpo para la siguiente pareja de prototipos.

Y cada definición debe ajustarse a la descripción brindada.

Prototipo de función a desarrollar
`char capturartecla(char listateclas[ ]);`
Descripcion del funcionamiento: Genera una pausa, la cual termina solamente cuando usuario presione una tecla existente en el arreglo `listateclas` recibido en su parametro. La función retorna el carácter que presiono usuario y sin imprimirla en pantalla.

Ejemplos de llamadas y resultados obtenido:
Ejemplo 1:
`char letra = capturartecla("abcde");`
La ejecución se suspende hasta que usuario presione cualquiera de las letras de la cadena del argumento `"abcde"`.
Ejemplo 2:
`cout<<capturartecla("123gGtT");`
La ejecución se suspende hasta que usuario presione solamente a una de las letras de la cadena del argumento "123gGtT". Cuando presiona una tecla valida, la función `capturartecla()` retorna el carácter presionado y `cout` lo mostrara en pantalla.

Prototipo de función a desarrollar
`bool buscartecla(char tecla, char teclas[ ]);`
Descripcion del funcionamiento: Retorna `true` solamente si el carácter recibido en parámetro (`tecla`) se encuentra dentro del vector recibido en el parámetro 2 (`teclas[]`). De lo contrario, retorna `false`.

Ejemplos de llamadas y resultados obtenido:
Ejemplo 1:
`buscartecla('p',"abcde" );`
Retorna `false`, porque el carácter 'p' no está dentro de la cadena "abcde".
Ejemplo 2:
`char lista[ ]="12345mn";`
`buscartecla('4', lista);`
Retorna `true`, porque carácter '4' se encuentra dentro del arreglo `lista`.

Finalmente, demuestre el uso de ambas funciones, haciendo que el programa solicite el nombre del usuario y luego genere un menú para que usuario elija que se le salude en uno de 4 idiomas diferentes.

PROBLEMA 6:

Desarrolle una funcion llamada `ListaPrimos`, que genere en pantalla al listado de numeros primos ubicados entre 1 hasta un numero X (recibido como parametro).

Por ejemplo, si la funcion recibe en su parametro a 30, esta debe mostrar en pantalla el siguiente resultado:

```
Lista de primos entre 1 a 30 es >>   1   2   3   5   7   11   13   17   19   23   29
```

Restricciones:

- Puede crear otras funciones complementarias a la función principal (ListaPrimos) solicitada.
- Solo pueden definirse variables de alcance local
- En caso que cualquiera de las funciones requiera el uso de ciclos, solo puede utilizar como máximo a un ciclo en su interior.

RUBRICA DE EVALUACION

Actividad a evaluar: ANALISIS DE RESULTADOS

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

Instructor seleccionara 2 problemas de la DISCUSION DE RESULTADOS para ser resueltos. Luego, su instructor indicara la metodología de entrega de la solución final.

Lista de Integrantes:

CARNET 1	CARNET 2	CARNET 3	CARNET 4	CARNET 5

Problemas a resolver:

Criterio a evaluar	¿Prob 1?	¿Prob 2?	PROM.
(25%) No se utilizan variables globales Cód. fuente se logra compilar y se obtiene a c/u de los resultados solicitados			
(50%) Se crea la(s) función(es) solicitada(s), con el/los nombre(s) y los parámetros requeridos en el problema Se cumple cada una de las restricciones dadas en el problema			
(25%) Aplica un lenguaje conciso para comunicarse con usuario Documenta el código fuente de manera apropiada			