Programación I

Unidad 01 – Semana 02

Conceptos básicos de programación en C

Unidad 01 – Semana 02
Conceptos básicos de programación

Temario

- Origen del C.
- Conceptos básicos de programación en el Lenguaje C++
  1. Tipos de datos, variables y constantes
  2. Operadores:
     a) De asignación (=)
     b) Aritméticos (+, -, *, %, /)
     c) De incremento y decremento (++, --, +=, -=, *=, /=)
     d) De relación (>, <, ==, <=, >=, !=)
     e) Lógicos (&&, ||, !)
  3. Estructura de un programa
  4. Operaciones de entrada y salida de datos

Origen del Lenguaje C

"C es un lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. **C** es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas. En la práctica, los programadores suelen usar elementos no-portátiles dependientes del compilador o del sistema operativo." (1)

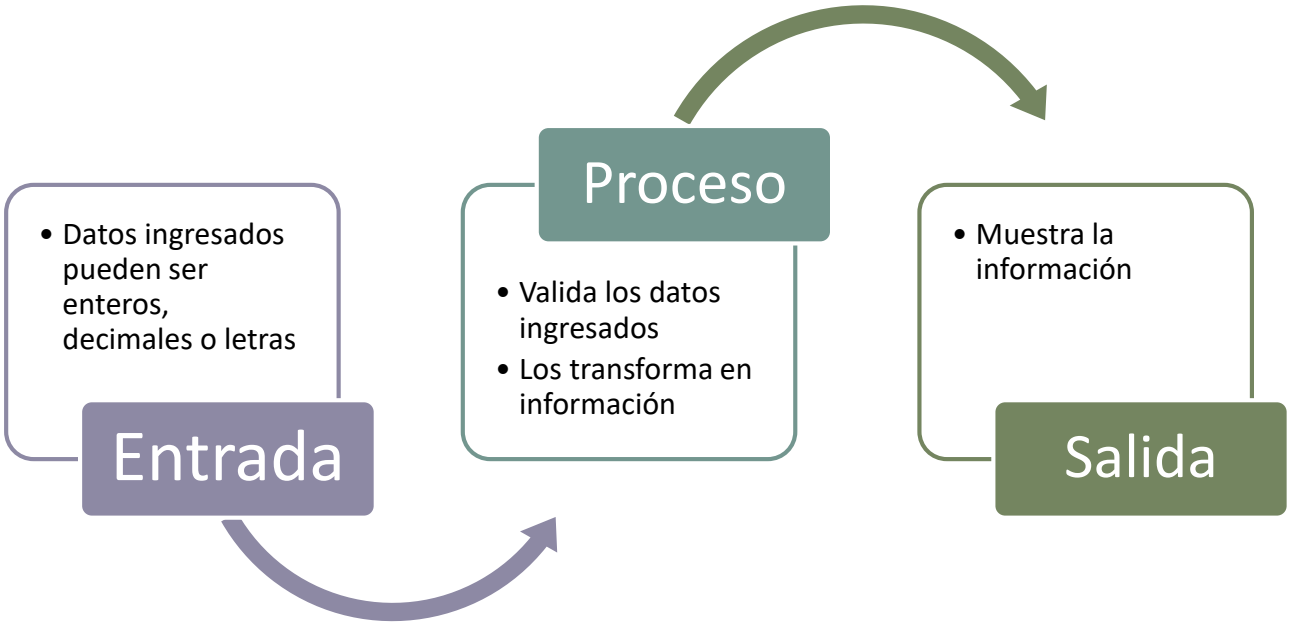
(1) https://es.wikipedia.org/wiki/C_(lenguaje_de_programación)

Conceptos básicos de programación

Tipos de datos

Tipos de datos

La computadora siempre procesa datos y los transforma en información.



Tipos de datos

- La computadora siempre procesa datos y los transforma en información.

- Los datos que procesa la computadora, son básicamente de tres tipos: enteros, decimales y letras.

- En C existen tipos de datos básicos y también da la posibilidad de crear tipos de datos definidos por el usuario.

Tipos de datos - Analogía

- Un tipo de datos es la propiedad de un valor que determina su dominio (qué valores puede tomar), qué operaciones se le pueden aplicar y cómo es representado internamente por el computador.

- Por ejemplo, si tenemos que hablar acerca de una persona, debemos saber su nombre, apellido, edad y estatura.
  - El nombre y apellido son palabras.
  - La edad es un número entero.
  - La estatura un número decimal.

- Como podemos apreciar tenemos tres tipos de datos, palabras, números enteros y números decimales.

Tipos de datos en C

Tipo	Descripción	Rango	Ejemplo
bool	Valor binario verdadero o falso.	true False	bool dato = false; dato = true;
char	Valor entero que representa un carácter de la tabla ASCII	–128 a 127 ó 0 a 255 compilado con /J	char letra = 'A'; letra = '\n'; letra = 65;
short	Valor entero de 2 bytes	–32,768 a 32,767	short x = 94; x = -54;
int	Valor entero de 4 bytes	–2,147,483,648 a 2,147,483,647	int x = 1598; x = -988574;
unsigned int	Valor entero positivo de 4 bytes	0 a 4,294,967,295	unsigned int x = 9887; x = 98745;
long long	Valor entero de 8 bytes	–9,223,372,036,854,775,808 a 9,223,372,036,854,775,807	long long x = 684574; x = -998564;
float	Valor decimal de 4 bytes	3.4E +/- 38 (7 dígitos)	float x = 45.6; x = -98.58;
double	Valor decimal de 8 bytes	1.7E +/- 308 (15 dígitos)	double x = 9878.568; x = -98745.668;
void	Tipo de dato nulo. Representa la ausencia de valor.		

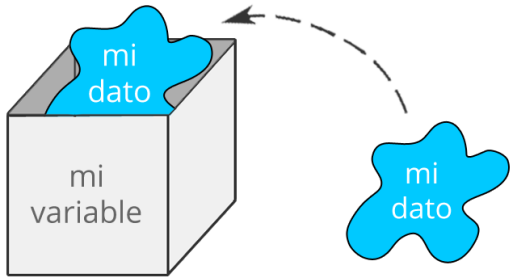
Variables y constantes

Variables y constantes

Si se desea imprimir los resultados de multiplicar un número fijo por otro que adopta valores entre 0 y 9, la forma normal de programar esto sería crear una **constante** para el primer número y un par de **variables** para el segundo y para el resultado del producto.

Para trabajar con los datos ingresados, la computadora necesita guardarlos.

El lugar que usa la computadora para almacenar los datos es la memoria.



Variables y constantes en C++

- Las variables son espacios en memoria que permiten almacenar y modificar un valor en cualquier punto del programa.

- Las constantes son espacios en memoria que permiten almacenar un valor pero no modificarlo.

- Las variables y constantes tienen que especificar a que tipo de dato pertenecen para que el compilador sepa como interpretarlos.

- Las variables y constantes deben poseer un nombre único dentro del programa y es **obligatorio declararlas** antes de poder utilizarlas.

Nombres de variables y constantes

- El primer carácter debe ser una letra o el signo de subrayado “ _ ”.
- Los caracteres restantes pueden ser letras, el signo del subrayado o dígitos.
- **NO** se pueden utilizar caracteres especiales en el nombre como por ejemplo (ñ, ?, , , %, etc.)
- Debe ser distinto de las palabras reservadas por el lenguaje, por ejemplo int, float, if, else, etc.
- El Lenguaje C, hace diferencia entre mayúsculas y minúsculas por lo que una variable llamada **Sueldo** es distinta de una variable llamada **sueldo**.

Nombres de variables y constantes Ejemplos

- Sueldo, SUELDO, sueldo ***son 3 identificadores distintos***
- _Area
- Ordena_un_arreglo
- Leer_Datos
- Valor3
- D
- Z5

Tipos de una variable en C++

Las variables pueden ser de dos tipos:

- Estáticas
- Dinámicas

Las variables, que reciben el nombre de estáticas, tienen un tamaño asignado desde el momento en que se crea el programa.

Las variables dinámicas, son aquellas para las cuales se reserva espacio en el momento de ejecución del programa, sólo en la cantidad necesaria.

Definición de una variable en C++

Para crear una variable en un lugar determinado del programa escribimos primero el tipo de la variable y luego el identificador con el que queremos nombrar la variable, seguido todo de un ';'.

Las variables se pueden inicializar, es decir, establecer un valor inicial, en el momento de creación. Para ello, detrás del identificador ponemos el carácter '=' seguido del valor inicial.

Los valores iniciales pueden ser cualquier constante válida para el tipo de variable que vamos a crear.

Definición de una variable en C++

Existen tres sitios donde se pueden definir variables:

- fuera de todas las funciones (variables globales).
- dentro de las funciones (variables locales)
- en la definición de parámetros de funciones (parámetros formales).

Variable Global	**Variable Local**

#include <stdio.h>
int x;
void main()
{
}
	#include <stdio.h>
	void main()
	{
	  int x;
	}

Definición de una variable en C++

Estática

<Tipo de dato> <Nombre variable>;

Por ejemplo:

```
int variableEntera;
bool _esVerdadero_o_no;
char Mi_1_era_letra;
long long EsteEsUnEnteroLargo;
unsigned int No_Acepta_Negativos;
int x1, x2, x3, x4;
```

Definición de una variable en C++

Dinámica

<Tipo de dato> * <Nombre variable>;

Por ejemplo:

```
int *variableEntera;
    variableEntera = new int;
double *Estatura;
    Estatura = new double,
bool *_esVerdadero_o_no;
char *Mi_1_era_letra;
long long *EsteEsUnEnteroLargo;
unsigned int *No_Acepta_Negativos;
int *x1, *x2, *x3, *x4;
```

Definición de una constante en C++

const <Tipo de dato> <Nombre constante> = <valor>**;**

Por ejemplo:

```
const int ValorMaximo = 50;
const double IGV = 18.00;
const char PrimeraLetra = 'A';
```

Comentarios

- Los comentarios sirven para incluir aclaraciones en el código del programa.
- Se permite dos tipos de comentarios:
  - **// Comentarios de una línea**
  - **/* Comentarios de varias líneas*/**
- Incluya comentarios que expliquen lo que hace el programa.
- Los comentarios deben ayudar al lector del código en las partes difíciles.

Constantes especiales - MACROS

Constantes especiales - MACROS

- Existe una forma de colocar valores constantes en los programas llamadas MACROS.

- Los MACROS a diferencia de las constantes no tienen un tipo de dato, simplemente tienen un valor que será reemplazado cada vez que se encuentre el nombre de la MACRO.

- Por ejemplo si una MACRO se llama DATO con valor 10, el compilador reemplazará la palabra DATO por 10 cada vez que la encuentre dentro de mi programa.

Definición de una MACRO en C++

#define <Nombre MACRO> <valor>

Por ejemplo:

#define IGV 18.00

#define PrimerMes 3

#define PalabraClave “Universidad”

Definición de una MACRO en C++

Si mi programa tiene el siguiente código:

```
#define IGV 18.00
#define MENSAJE “Hola como estas”
int main() {
  double x;
  x = 94 + IGV / 100 + IGV * 45;
  cout<<MENSAJE;
}
```

El compilador lo convertirá a:

```
int main() {
  double x;
  x = 94 + 18.00 / 100 + 18.00 * 45;
  cout<<“Hola como estas”;
}
```

Operadores

- Asignación
- Aritméticos
- Incremento y decremento
- Relación
- Lógicos

Operador de asignación

- El operador de asignación permite colocarle un valor a una variable o constante.
- En el caso de las constantes solo se puede utilizar este operador cuando se define la constante.
- El símbolo de igual (=) es el operador de asignación.
- Tenga en cuenta que la asignación siempre se realiza de derecha a izquierda.

Ejemplos

```
int x;
x = 198.54;

double valor;
valor = 98.55684;
valor = valor * 2.0;

char letra;
letra = ‘A’;
letra = letra + 2;
letra = ‘X’;
letra = 65;
```

Tenga en cuenta que cuando asignamos un valor decimal a una variable entera, solamente se almacena la parte entera.

Operadores aritméticos

Símbolo	Operación
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (Solo de números enteros)

A excepción de la operación de modulo %, que se aplica a datos enteros, todas las operaciones dan resultados:

- Del mismo tipo que los operandos si ambos son de mismo tipo ó
- Del tipo de mayor rango si los operandos son de tipos distintos. (1).

(1) Maynard Kong

Operadores aritméticos - Ejemplos

Operación	Igual a	Resultado
15 / 2 + 3.0 * 2	7 + 6.0	13.0
15 / 2.0 + 3.0 * 2	7.5 + 6.0	13.5
(3 – 4.7) * 5	-1.7 * 5	-8.5
1 + 7 % 3	1 + 1	2

- Se puede emplear paréntesis para agrupar datos y especificar un cálculo.
- El orden en que se evalúa la expresión se puede especificar utilizando paréntesis, o se asume el orden de precedencia que se indica a continuación:
  - Paréntesis
  - *, /, %
  - +, -
- Si un operador tiene la misma precedencia que otro operador en la misma expresión, se evaluará de izquierda a derecha.

Operador de incremento ++

Incrementa el valor actual de la variable en 1 y puede ser usado de dos formas:

- Pre Fijo (++ variable)
  - Suma 1 a variable y devuelve el resultado.
- Post Fijo (variable ++)
  - Suma 1 a variable y devuelve el valor de variable antes de la suma.

Operador de decremento --

Reduce el valor actual de la variable en 1 y puede ser usado de dos formas:

- Pre Fijo (-- variable)
  - Resta 1 a variable y devuelve el resultado.
- Post Fijo (variable --)
  - Resta 1 a variable y devuelve el valor de variable antes de la resta.

Operadores de incremento y decremento

Operación	Ejemplo	Equivalencia
++	a++; ++a;	a = a + 1;
--	a--; --a;	a = a - 1;
+=	a += 10;	a = a + 10;
-=	a -= 10;	a = a - 10;
*=	a *= 10;	a = a * 10;
/=	a /= 10;	a = a / 10;

Ejemplo de operadores ++ y --

```
int x; int y;
// Operador de incremento
 x = 1;
 y = ++x; // x es ahora 2, y es también 2
 y = x++; // x es ahora 3, y es 2
 // Operador de decremento
 x = 3;
 y = x--; // x es ahora 2, y es 3
y = --x; // x es ahora 1, y es también 1
```

Operador de moldeado o typecasting

Es el proceso mediante el cual convierto de un tipo de dato a otro.

Este proceso se puede realizar siempre y cuando los tipos de datos sean compatibles entre sí. Los tipos de datos genéricos en C++ son compatibles entre sí.

Para realizarlo debe colocar lo siguiente:

(<tipo_de_dato>) <expresión>

Ejemplo:

- **int** DatoEntero;
- DatoEntero= 1.6 + 1.7; *// Almacena 3*
- DatoEntero = (**int**)1.6 + (**int**)1.7; *// Almacena 2*

Función sizeof

La función **sizeof** indica la cantidad de bytes que ocupa un tipo de dato. Esta función **sólo** se puede utilizar con tipos de dato para que funcione correctamente.

Ejemplo:

- **int** tamanioDeInt ;
- tamanioDeInt = sizeof(**int**);

- **int** tamanioDeDouble:
- tamanioDeDouble = sizeof(**double**);

Operadores de relación

Operador	Descripción	Ejemplo	Respuesta
>	Mayor	10 > 20 30 > 20 20 > 20	F V F
<	Menor	10 < 20 30 < 20 20 < 20	V F F
>=	Mayor o igual	10 >= 20 30 >= 20 20 >= 20	F V V
<=	Menor o igual	10 <= 20 30 <= 20 20 <= 20	V F V
==	Igual	10 == 10 20 == 10	V F
!=	Diferente	10 != 10 20 != 10	F V

Operadores de relación

Tenga cuidado de no confundir el operador de asignación con el operador de comparación “es igual a”

Ejemplo:

Entero = 3; *// asigna el valor 3 a la variable Entero*

Entero == 3; *// comprueba si Entero tiene el valor 3.*

¿Qué es verdad?

- Esta pregunta se la han formulado a filósofos de todas las épocas. Nosotros nos daremos el gusto de contestarla, al menos en lo que respecta al Lenguaje C.

- Recuerde que cada expresión en C, siempre tiene un valor.

- Analice los siguientes ejemplos.

(1) M. Waite; Programación en C, Introducción y conceptos avanzados

¿Qué es verdad?

```
int cierto, falso;
cierto = (10>2);      //-- cierto es 1
falso  = (10==2);     //-- falso es 0

bool cierto2, falso2;
cierto2 = (10>2);     //-- cierto2 es true
falso2  = (10==2);    //-- falso2 es false
```

¿Qué es verdad?

```
bool valor1, valor2, valor3, valor4;

valor1 = (bool) 77;
valor2 = (bool) -33;
valor3 = (bool) 1;
valor4 = (bool) 0;
```

Solamente el valor4 es falso ya que el lenguaje C++ solo considera el valor 0 como falso, cualquier otra cosa es considerada verdadera.

Operadores lógicos

Operador	Significado
&&	and (y)
||	or (o)
!	not (no)

A	B	A && B	A || B	! A	! B
V	V	V	V	F	F
V	F	F	V	F	V
F	V	F	V	V	F
F	F	F	F	V	V

Operadores lógicos - Ejemplos

Expresión	Valor	Explicación
(5 > 2) && (4 > 7)	Falso	(5 > 2) = V (4 > 7) = F (V) && (F) = F
(5 > 2) || (4 > 7)	Verdadero	(5 > 2) = V (4 > 7) = F (V) || (F) = V
! (4 > 7)	Verdadero	(4 > 7) = F !(F) = V

Expresiones

Expresión	Valor
‘A’ > ‘C’	
(100 > 3) && (‘A’ > ‘C’)	
!(100 > 3)	
	Numero es igual o mayor que 1 pero menor que 9
	Numero está entre 1 y 9, pero no es 5
	Numero no está comprendido entre 1 y 9

Expresiones - Respuestas

Expresión	Valor
‘A’ > ‘C’	**Falso**
(100 > 3) && (‘A’ > ‘C’)	**Falso**
!(100 > 3)	**Falso**
(Numero >= 1) && (Numero < 9)	Numero es igual o mayor que 1 pero menor que 9
(Numero > 1) && (Numero < 9) && (Numero != 5)	Numero está entre 1 y 9, pero no es 5
(Numero < 1) || (Numero > 9)	Numero no está comprendido entre 1 y 9

Estructura de un programa

```
// P1.cpp : main project file.

#include <iostream>
#include <conio.h>

using namespace System;
using namespace std;

int main()
{
  int L,A;
  cout << "Largo, Ancho :";
  cin >> L >> A;

  Console::Clear();
  Console::SetCursorPosition(40,12);
  Console::BackgroundColor = ConsoleColor::DarkCyan;
  Console::ForegroundColor = ConsoleColor::Black;

  cout << "El area es " << L * A;
  Console::Beep();

  _getch();
  return 0;
}
```

Directivas del preprocesador

Nombres de espacio

Un programa en C++, puede tener varias funciones, pero siempre debe estar presente la función main()

Operaciones de entrada y salida de datos

Entrada y Salida de datos

- Para recibir datos desde el teclado utilizaremos el objeto **cin**.

- Para mostrar datos en la pantalla utilizaremos el objeto **cout**.

- Ambos están definidos en la biblioteca **iostream**, la cuál puede ser agregada a nuestro programa utilizando el comando **#include <iostream>**

cout

- El objeto cout puede ser utilizado de 2 formas:

1. cout << “Texto a mostrar”;
2. cout << “Texto” << valores;

- Ejemplos:

```
cout << “UPC…, protagonistas del cambio!!!”;
cout << “Hola como estas”;
cout << “La suma de A + B es: ”  << 10 + 5;
cout << “La suma de ” << 10 << ”+ ” << 5 << ”es: ” << 10 + 5;
```

Secuencias de escape

- Las secuencias de escape se utilizan para imprimir caracteres especiales.

Secuencia	Explicación	Ejemplo
\n	Salto de línea	cout << “Primera Linea\nSegunda Linea”;
\a	Alarma	cout << “Suena ahora\a…Listo”;
\b	Retroceso	cout << “Error do\be tipeo”;
\t	Tabulación	cout << “Columna1\tColumna2\tColumna3”;
\\	Barra inclinada	cout <<“ El salto de linea es con \\n solamente”;
\”	Comillas dobles	cout << “Me dijo: \”Salta ahora\” y nada más”;

Indicadores de formato

- Los indicadores de formato son tres funciones miembro (width, precision y fill) que fijan formato de anchura, precisión y carácter de relleno.
- Es necesario fijar la anchura, precisión y carácter de relleno antes de cada sentencia de escritura.

Indicadores de formato

- ***ANCHO*:** cout.width(ancho);

  Establece la anchura de campo mínima para un dato de salida.
- ***DECIMALES*:** cout.precision(nº digitos);

  Establece el número de cifras para un dato de salida.
- ***RELLENO*:** cout.fill('carácter');

  Establece el carácter de relleno para un dato de salida.

Indicadores de formato - Ejemplo

```
double numero;
numero = 123.1234567;
cout<< "hola" <<"\n";
cout.width(15);
cout<< "hola" <<"\n";
cout.width(15);
cout.fill('*');
cout<< "hola"<<"\n";
cout<< numero <<"\n";
cout.precision(4);
cout<< numero <<"\n";
cout.precision(10);
cout<< numero;
```

cin

- El objeto cin puede ser utilizado de la siguiente forma:

  cin >> variable [>> variable];

- Ejemplos:

```
int  entero,  entero1,  entero2;
float  flotante;
char   caracter;
```

  - cin >> entero;
  - cin >> caracter;
  - cin >> flotante;
  - cin >> entero1 >> entero2;

cin

Ejemplo 1:

```
int  L,  A;
cout << "Ingrese el largo: ";
cin >>  L;
cout  << "Ingrese el ancho: ";
cin >>  A;
cout << "El area es: " <<  L *  A;
```

```
Ingrese el largo: 10
Ingrese el ancho: 5
El area es: 50
```

Ejemplo 2:

```
int  L,  A;
cout << "Ingrese Largo y Ancho: ";
cin >>  L  >>  A;
cout << "El area es: " <<  L *  A;
```

```
Ingrese el Largo y Ancho: 10  5
El area es: 50
```

Ejercicios

Ejercicio 1

Escriba un programa en C++ que permita solicitar un número entero y mostrar el valor, el doble del valor y el triple del valor.

```
Ingrese un numero entero: 10
El valor es: 10
El doble es: 20
El triple es: 30
```

Ejercicio 2

Escriba un programa en C++ que permita ingresar 2 números enteros y mostrar, la suma, resta, multiplicación, división y módulo de los números ingresados.

```
Ingrese el primer numero: 10
Ingrese el segundo numero: 4

Suma: 14
Resta: 6
Multiplicacion: 40
Division: 2.5
Modulo: 2
```

Ejercicio 3

Escriba un programa en C++ que permita ingresar 1 solo número entero de 3 dígitos y mostrar la suma de los dígitos. Puede considerar que el usuario siempre ingresará un número de 3 dígitos.

```
Ingrese un número de 3 digitos: 492

La suma de los digitos es:
4 + 9 + 2 = 15
```

Ejercicio 4

Escriba un programa en C++ que permita solicitar 1 caracter y mostrar los 3 caracteres siguientes según la tabla ASCII.

```
Ingrese un carácter: F
Las letras que le siguen a F son: GHI
```

Ejercicio 5

Escriba un programa en C++ que permita ingresar un número de 4 dígitos y mostrar el número invertido. Puede considerar que el usuario siempre ingresará un número de 4 dígitos.

```
Ingrese un número de 4 digitos: 4987

El número invertido es:
7894
```

Ejercicio 6

Escriba un programa en C++ que permita ingresar un número decimal y mostrar de forma separada la parte entera y la parte decimal.

```
Ingrese un número decimal: 98.567

La parte entera es: 98
La parte decimal es: 0.567
```

Ejercicio 7

Escriba un programa en C++, que permita hallar el área y el volumen de una esfera. El usuario deberá ingresar el valor del radio.

```
Ingrese el radio: 3

El area de la esfera es: 113.094

El volumen de la esfera es: 113.094
```

La fórmulas son:

$$\text{Área} = 4\ \Pi\ r^2$$

$$\text{Volumen} = \frac{4\ \Pi\ r^3}{3}$$

Ejercicio 8

Escriba un programa en C++, que permita convertir grados Farenheit a Grados Centígrados.

```
Ingrese los grados Farenheit: 98

98°F = 36.67°C
```

La fórmula es:

Centigrados = (Farenheit - 32) * (5 / 9)

Ejercicio 9

Un distribuidor de material eléctrico vende alambre en rollos de 500, 300 y 75 metros. Escriba un programa en C++ que pida al usuario, la longitud total de alambre en metros que se requiere e imprima la **menor cantidad de rollos de alambre** de 500, 300 y 75 metros y el número de metros de alambre que tendría el ultimo rollo.

```
Ingrese la cantidad de alambre requerido: 376

0 rollos de 500 metros
1 rollo de 300 metros
1 rollo de 75 metros
El último rollo tendrá 1 metro.
```