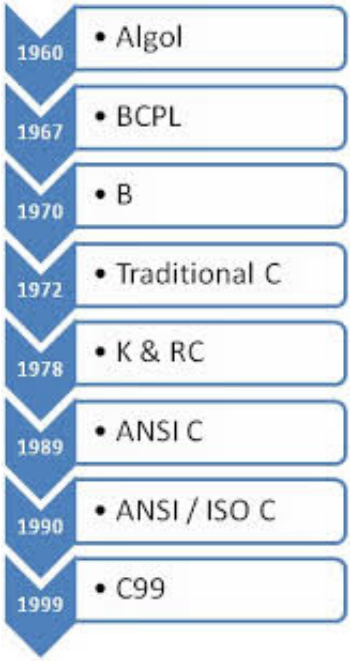
3. Comenzando a programar en C

3.1. Acerca del lenguaje de programación C



¿Por qué programar en C?

- Poderoso y flexible
- Portable
- Soporta la programacin estructurada
- Contiene pocas palabras reservadas (keywords)

3.2. Almacenando información en C

Los programas de computadora usualmente trabajan con diferentes tipos de datos y necesitan una manera de almacenar estos valores que son empleados. Estos valores pueden ser números o caracteres. C tiene dos formas de almacenar valores numéricos: empleando variables o constantes.

Una computadora emplea memoria de acceso aleatorio (RAM) para almacenar información mientras funciona. La RAM es volatil, es decir, "recuerda" mientras la computadora está prendida y pierde su información cuando se apaga la computadora. El byte es la unidad fundamental de almacenamiento de dato de la computadora. La RAM en la computadora está organizada secuencialmente por bytes, un byte enseguida de otro. Cada byte de memoria tiene una única dirección asignada que la distingue de los otros bytes.

- Variables
- Identificadores
- Tipos de datos numéricos de C
- Constantes
  - Literales
  - Simbólicas

Una variable es un lugar en la memoria de la computadora que sirve para poder almacenar información donde

- Esta información puede cambiarse durante la ejecución del programa
- Este lugar está identificado con un nombre (un **identificador**)

Un identificador es una secuencia de dígitos, letras y subguiones donde

- El primer carácter debe ser una letra o un subguión
- Letras mayúsculas y minúsculas son diferentes

Ejemplos correctos e incorrectos de identificadores:

Porcentaje	correcto
nombre_1	correcto
n	correcto
N	correcto
dia1	correcto
_temporal	correcto pero NO recomendable
ingreso#anual	incorrecto
`double`	incorrecto
4dia	incorrecto

Tipos de datos numéricos de C:

Tipo de variable	Palabra reservada	Bytes requeridos	Rango
Caracter	`char`	1	-128 a 127
Entero corto	`short`	2	-32 767 a 32 767
Entero	`int`	4	-2 147 483 647 a 2 147 483 647
Precisión simple en coma flotante	`float`	4	$\pm 1{,}2 \times 10^{-38}$ a $\pm 3{,}4 \times 10^{38}$ (7 dígitos de precisión)

Una constante es lo mismo que una variable salvo que la información que se almacena en ella NO puede cambiarse durante la ejecución del programa. C tiene dos tipos de constantes:

- literales
- simbólicas

Una constante literal es un valor que se tipea directamente en el programa donde este sea necesitado, mientras que una constante simbólica es una constante que es representada por un nombre (identificador.) Cada vez que se necesite el valor de esta constante, se emplea su nombre para acceder a este. C tiene dos métodos para definir estas constantes: la directiva `#define` y la palabra reservada `const`.

3.3. Leyendo y escribiendo información en C

- La función `printf( )`
- La función `scanf( )`

La función `printf( )` en la mayoría de programas que crearás, necesitarás mostrar información en la pantalla o leer información desde el teclado. La función `printf( )` es parte de la biblioteca estándar de C. Para mostrar `Hola mundo` en la pantalla, escribimos:

```
printf("Hola mundo");
```

Si el valor de la variable entera `cont` es `5`, la siguiente directiva

```
printf("\nEl valor de contador es%d", cont);
```

mostrará en la pantalla

```
El valor de contador es 5
```

En este ejemplo, dos argumentos son pasados a `printf( )`. El primero está encerrado en doble comillas y es llamado de **cadena de formato**. El segundo es el nombre de la variable (`cont`) que contiene el valor a imprimir.

La **cadena de formato** provee la descripción de la salida. Esta tiene tres posibles componentes:

- **Texto literal**: es mostrado exactamente como es escrito
- Secuencia de escape: consiste de un backslash (\) seguido de un caracter.
- Especificadores de conversión: consiste de un signo de porcentaje (%) seguido de un caracter; su función es decir a `printf( )` cómo interpretar la variable a ser mostrada.

La función `scanf( )` es también parte de la biblioteca estándar de C. Esta función lee datos desde el teclado según el formato especificado y asigna la información ingresada a una o más variables del programa.

Al igual que `printf( )`, `scanf( )` usa una **cadena de formato** para describir el formato a ingresar. No obstante, esta cadena de formato debe estar conformada por estas dos posibles componentes:

- Especificadores de conversión, los mismos de `printf( )`
- Espacios en blanco

Ejercicio 3.1. Cree un programa que pida dos números enteros m y n. Luego muestre el valor de $m + n$, $m - n$, $m * n$ y m/n

Ejercicio 3.2. Elabore un programa que pida ingresar el radio de un círculo. Luego calcule y muestre su diametro, perímetro y área.

Ejercicio 3.3. Mencione un par de ventajas de las constantes simbólicas sobre las constantes literales.

Ejercicio 3.4. Implemente un programa que pida ingresar tu masa en kilogramos y te muestre tu masa en libras.

Ejercicio 3.5. Escriba un programa que pida ingresar la temperatura en grados Farenheit. Luego, calcule y muestre la temperatura en grados Celcius mediante la fórmula:

$$\text{Temperatura en grados Farenheit} = 1{,}8 \times \text{Temperatura en grados Celcius} + 32.$$

Ejercicio 3.6. Implemente un programa que muestre la cantidad de bytes que la computadora utiliza para los diferentes tipos de datos con los que C trabaja.

Ejercicio 3.7. Cree un programa que pida ingresar las coordenadas de dos puntos A y B que viven en $\mathbb{R}^2$. Luego, calcule y muestre las coordenadas del punto medio del segmento de recta $\overline{AB}$

Ejercicio 3.8. Elabore un programa que pida ingresar las coordenadas de dos puntos A y B que viven en $\mathbb{R}^3$. Luego, calcule y muestre las coordenadas de $A + B$, $A - B$ y $A \times B$ y el valor de $A \cdot B$.