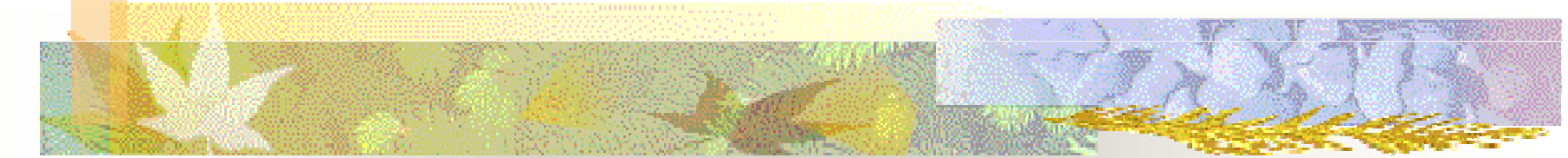
Curso Básico de C
Clase Nº 3



Salida y Entrada por Consola
Estructuras Condicionales

Prof. Patricia Guerrero

Salida y Entrada por Consola

Salida por Consola:

- **Función printf( )**

  Imprime en pantalla valores numéricos, constantes, caracteres o cadenas de texto. Su prototipo es:

  int printf("cadena_de_control", arg1, arg2, ...)

Imprime el texto contenido en **cadena_de_control** junto con el valor de los otros argumentos, de acuerdo con los *formatos* incluidos en **cadena_de_control**.

Salida y Entrada por Consola

Salida por Consola:

- **Función printf( )**

```
int printf("cadena_de_control", arg1, arg2, ...)
```

arg1, arg 2 comienzan con el símbolo de % y terminan con el caracter de conversión

```
int i =  1;
double tiempo = 20;
float masa = 25.67;
printf("Resultado %d: Instante = %lf - Masa = %f\n", i, tiempo,
masa);
```

Salida y Entrada por Consola

Salida por Consola:

- **Función printf( ): Caracteres de Conversión**

%c	Un único carácter
%d, %i	Un entero con signo, en base decimal
%u	Un entero sin signo, en base decimal
%o	Un entero en base octal
%x, %X	Un entero en base hexadecimal
%e, %g	Un número real en coma flotante, con exponente
%f	Un número real en coma flotante, sin exponente
%s	Una cadena de caracteres
%p	Un puntero o dirección de memoria

Salida y Entrada por Consola

Función printf( ):

Entre el signo de % y el carácter de conversión pueden haber uno o más elementos:

% [signo] [longitud] [.precisión] [l/L]conversión

- Un signo negativo (-) indica ***alineación*** por la izquierda.
- Un número entero positivo en longitud, indica la ***anchura mínima*** del campo en caracteres.

Salida y Entrada por Consola

Función printf( ):

% [signo] [longitud] [.precisión] [l/L]conversión

- Un punto (.) separa la ***anchura*** de la ***precisión.***
- Un número entero positivo en ***precisión***, indica número máximo de caracteres a imprimir en una cadena, o el número de decimales de un *float* o *double.*
- Un ***cualificador***: Una 'h' para *short* o una 'l' para *long* y *double.*

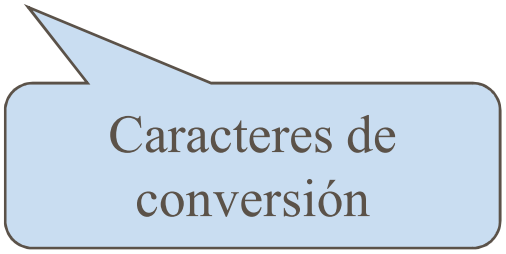
Salida y Entrada por Consola

Entrada por Consola:

- **Función scanf( )**

  Permite leer datos de la entrada estándar (teclado) del equipo. Su prototipo es:

  int scanf(“cadena_de_control", &arg1, &arg2, ...);



Salida y Entrada por Consola

Entrada por Consola:

- **Función scanf( )**

  int scanf(“cadena_de_control", &arg1, &arg2, ...);

  &arg1: representa la posición de memoria en la que se encuentra la variable arg

  int n;
  double distancia;
  char nombre[20];
  scanf("%d%lf%s", &n, &distancia, nombre);

Salida y Entrada por Consola

Entrada por Consola:

- **Función scanf( ): Caracteres de Conversión**

%c	Un único caracter
%d, %i	Un entero con signo, en base decimal
%u	Un entero sin signo, en base decimal
%o	Un entero en base octal
%x, %X	Un entero en base hexadecimal
%e, %E, %f, %g, %G	Un número real en coma flotante, con exponente
%s	Una cadena de caracteres
h, l	Para short, long y double
L	Para long double

Salida y Entrada por Consola

Función scanf( ) - Ejemplo:

```
#include <stdio.h>
main( )
{
      char nombre[15];
      int edad;
      printf(“Escriba su nombre: ");
      scanf("%s", nombre);
      printf("Indique su edad: ");
      scanf("%d", &edad);
}
```

Salida y Entrada por Consola

Entrada por Consola:

- **Función scanf( )**

En la cadena de control de la función **scanf()** pueden incluirse *corchetes* [] para detectar ciertos caracteres:

```
char s[30];

scanf("%[AB \n\t]", s);     /* Lee sólo los caracteres indicados */

scanf(" %[^\n]", s);        /* Lee todos los caracteres distintos a \n */
```

Salida y Entrada por Consola

Funciones putchar( ) y getchar( )

Permiten, respectivamente, imprimir y leer *un sólo caracter* cada vez en la salida o entrada estándar.

Prototipos:

Lo convierte en unsigned int

```
int putchar(int ch);
```

```
int getchar(void);
```

Lee como unsigned int

Salida y Entrada por Consola

Funciones putchar( ) y getchar( )

Permiten respectivamente imprimir y leer *un sólo caracter* cada vez, en la salida o entrada estándar.

```
putchar('a');          //Escribe el caracter a.
                       //Equivale a printf("a");

char c = getchar();    //Lee un caracter y lo asigna a c.
                       //Equivale a scanf("%c", &c);
```

Salida y Entrada por Consola

Funciones putchar( ) y getchar( ) - Ejemplo:

```
main( )
{
      char ch;
      do
      {
            ch = getchar( );
            putchar('-');
      }while(ch != '\n')
}
```

```
main( )
{
      putchar('A');
      putchar('\n');
      putchar('B');
}
```

Salida y Entrada por Consola

Funciones getche( ) y getch( ):

Proporcionan entrada interactiva de caracteres.

Prototipos:

```
int getche(void);

int getch(void);
```

Leen como **unsigned int** y lo transforma en **int**

Utilizan el archivo de cabecera **conio.h**

Salida y Entrada por Consola

Funciones gets( ) y puts( ):

char *gets(char *str);

Lee caracteres de la entrada estándar hasta que se introduce retorno de carro.

int puts(char*str);

Muestra en pantalla la cadena a la que apunta str.

Salida y Entrada por Consola

Funciones getche( ) y getch( ) - Ejemplo:

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

void main()
{
   char ch;
   do
   {
     ch = getch( );
     putchar(toupper(ch));
   }while(ch != 'q');
}
```

Salida y Entrada por Consola

Funciones gets( ) y puts( ) - Ejemplo:

```
#include <stdio.h>

void main( )
{
      char *p, str[80];
      printf("Introduzca una cadena: ");
      p = gets(str);
      if(p)      /* si no es nulo*/
        printf("p: %s cadena: %s", p, str);
}
```

```
#include <stdio.h>

void main( )
{
      puts("uno");
      puts("dos");
      puts("tres");
}
```

Estructuras Condicionales

Permiten variar el flujo del programa de acuerdo a la evaluación de ciertas condiciones.

- **Estructura IF**

```
if (condición)
    sentencia;
```

La **sentencia** se ejecuta sólo si se cumple la **condición**

- **Estructura IF … ELSE**

```
if (condición)
    sentencia1;
else
    sentencia2;
```

Si se cumple la **condición** se ejecuta **sentencia1**, en caso contrario se ejecuta **sentencia2**

Estructuras Condicionales

Estructura IF … ELSE - Ejemplo:

```
#include <stdio.h>
main()
{
   int usuario, clave = 18276;
   printf("Introduce tu clave: ");
   scanf("%d",&usuario);
   if (usuario == clave)
       printf("Acceso permitido");
   else
       printf("Acceso denegado");
}
```

Estructuras Condicionales

- **Estructura IF … ELSE - Múltiples:**

```
if (condición)
        sentencia1;
else if (condición)
        sentencia2;
else if (condición)
        sentencia3;
else
        sentencia4;
```

Realiza una ramificación múltiple, ejecutando ***una*** entre varias partes del programa según se cumpla ***una*** entre ***n*** **condiciones**

Estructuras Condicionales

- **Estructura IF … ELSE - Múltiples:**

```
if (condición)
        sentencia1;
else if (condición)
        sentencia2;
else if (condición)
        sentencia3;
else
        sentencia4;
```

Si una **condición** se cumple, se ejecuta la **sentencia** correspondiente y salta hasta el final de la estructura para continuar con el programa

Es posible utilizar llaves para ejecutar más de una sentencia si se cumple una condición

Estructuras Condicionales

Estructura IF … ELSE - Múltiples - Ejemplo:

```
main( )
{
      float monto;
      printf(“Indique el monto total de la compra: ");
      scanf("%f", &monto);
      if (monto < 10000)
            printf("Lo siento, no tiene descuento.");
      else if (monto < 20000)
            printf(“ Descuento de 5%");
      else if (monto < 30000)
            printf(“Descuento de 10%");
      else
            printf(“Descuento de 20%");
}
```

Estructuras Condicionales

- **Estructura IF Anidadas:**

Una sentencia ***if*** puede incluir otros ***if*** dentro de la parte correspondiente a su **sentencia.** Se les llama ***sentencias anidadas.***

```
if (a >= b)
    if (b != 0.0)
        c = a/b;
```

```
if (a >= b)
{
    if (b != 0.0)
        c = a/b;
}
else
        c = 0.0;
```

Regla: Un ***else*** pertenece al ***if*** más cercano

Estructuras Condicionales

- **Sentencia SWITCH**

```
switch (expresion)
{
        case expresion_cte_1:
                sentencias;
                break;
        case expresion_cte_2:
                sentencias;
                break;
        default:
                sentencias;
}
```

La constante evaluada sólo puede ser de tipo **entero** o **caracter**

default ejecutará sus sentencias en caso de que la opción seleccionada no corresponda a ningún **case**

Cada ***case*** puede incluir **una o más sentencias** sin necesidad de ir entre llaves, se ejecutan todas hasta que se encuentra la sentencia ***break***

Estructuras Condicionales

- **Sentencia SWITCH - Variaciones:**

```
switch (expresion)
{
    case expresion_cte_1:
            sentencia_1;
    case expresion_cte_2:
            sentencia_2;
    ...
    case expresion_cte_n:
            sentencia_n;
    default:
            sentencia;
}
```

Estructuras Condicionales

- **Sentencia SWITCH - Variaciones:**

```
switch (expresion) {
    case expresion_cte_1:
        sentencia_1;
        break;
    case expresion_cte_2: case expresion_cte_3:
        sentencia_2;
        break;
    default:
        sentencia_3;
}
```

Estructuras Condicionales

Sentecia SWITCH - Ejemplo:

```
#include <stdio.h>
main( )
{
    int dia;
    printf("Introduce un número para identificar el día [1-7]: ");
    scanf("%d", &dia);
    switch(dia)
    {
      case 1: printf("Lunes"); break;
      case 2: printf("Martes"); break;
      …
      case 6: printf("Sábado"); break;
      case 7: printf("Domingo"); break;
      default: printf(“Día incorrecto”); break;
    }
}
```

Estructuras Condicionales

Sentencia SWITCH - Ejemplo:

```
char color;
switch (color) {
        case 'a':
        case 'A':   printf("AMARILLO\n");
                    break;
        case 'r':
        case 'R':   printf("ROJO\n");
                    break;
        case 'b':
        case 'B':   printf("BLANCO\n");
                    break;
        default:    printf("OTRO COLOR\n");
}
```