UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

Conceptos básicos de C

Laboratorio de Sistemas Operativos
Universidad de Antioquia

Conceptos básicos de C

- Ejemplo programación C
- Uso del gcc
- Apuntadores
- Ubicación dinámica de memoria
- Llamados al sistema

UNIVERSIDAD DE ANTIOQUIA
1803

¿Qué es C ?

- Es un lenguaje de programación de propósito general.
- Desarrollado por Dennis Ritchie entre 1969 y 1973 en Bell Labs.

- Es uno de los lenguajes más utilizados actualmente en:
  - Sistemas operativos como Linux.
  - Programación de sistemas embebidos: electrodomésticos, automóviles, dispositivos médicos, etc.

Evolución del lenguaje

- **1972**: Aparece C.
- **1978**: Brian Kernighan and Dennis Ritchie escriben la primera especificación informal (libro: The C Programming Language).
- **1989**: C89 ANSI C estándar.
- **1990**: C90 ANSI C adoptado por ISO (International Organization for Standardization).
- **1999**: C99.
- **2011**: C11.

Estructura de un programa en C

```
1 /*
2   Breve descripción del programa
3 */
4 #include <stdio.h>
5 #include <unistd.h>
6
7 //Comentario de una línea
8 char nombre[50];
9 /*Comentario de Varias Líneas*/
10 ----------------------------------------------------------------------
11 int main () {
12   int pid = getpid();
13   printf("Ingrese su nombre por favor: ");
14   scanf("%s", nombre);
15   printf("Hola %s, su id de proceso es %d\n", nombre, pid);
16   return(0);
17 }
18 ----------------------------------------------------------------------
```

Librerías

Variables Globales

Punto de Entrada

Variable Local

Entrada y Salida Estándar

Terminación del programa Retorna 0 (sin errores)

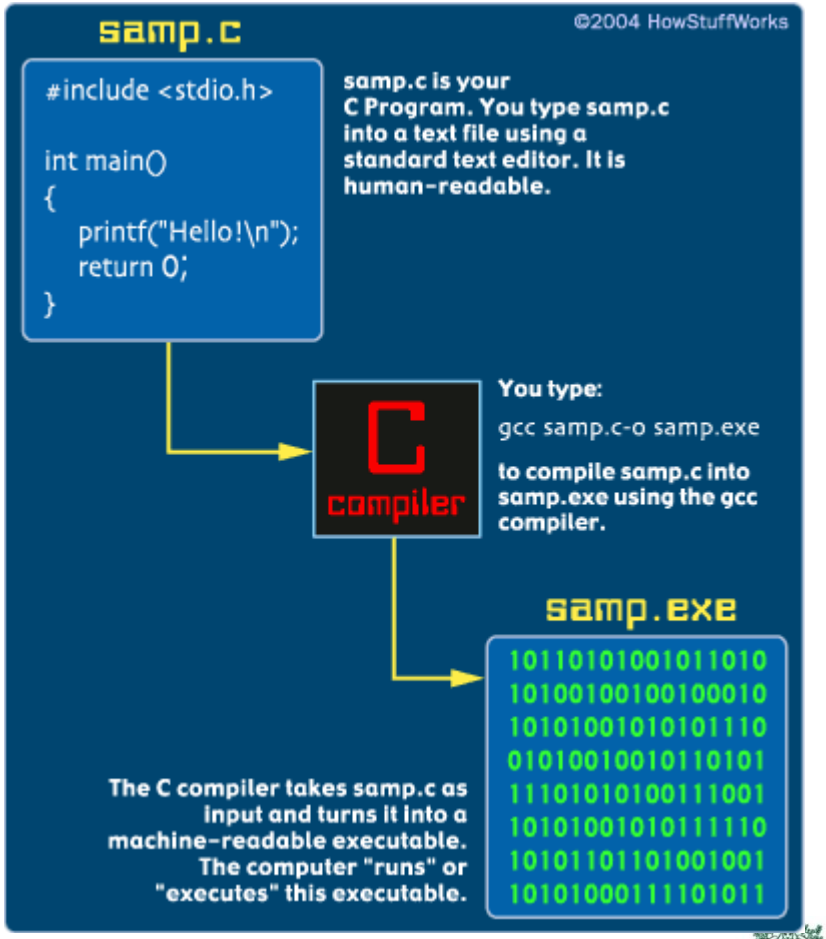
UNIVERSIDAD DE ANTIOQUIA
1803

Compilación de un programa en C

- El programa anterior se puede compilar así:

  $ gcc -Wall Lab1-1.c -o Lab1-1

- Donde:
  - **gcc** es el compilador.
  - **-Wall**: habilita los mensajes de advertencia (**warnings**) del compilador.
  - **Lab1-1.c** es el archivo de entrada a compilar.
  - **-o Lab1-1** indica el nombre del archivo de salida.



Tomado de:
http://computer.howstuffworks.com/c1.htm

UNIVERSIDAD DE ANTIOQUIA
1803

Generalidades del lenguaje

- Convenciones de nombres (palabras claves).
- Inclusión de librerías (Propias y del sistema).
- Constantes (simbólicas, implícitas).
- Variables, expresiones y operadores.
- Type casting.
- Entrada y salida estándar.
- Toma de decisiones, bucles y condicionales.
- Arreglos, Strings.
- Funciones.
- Estructuras, uniones

Palabras reservadas ANSI C

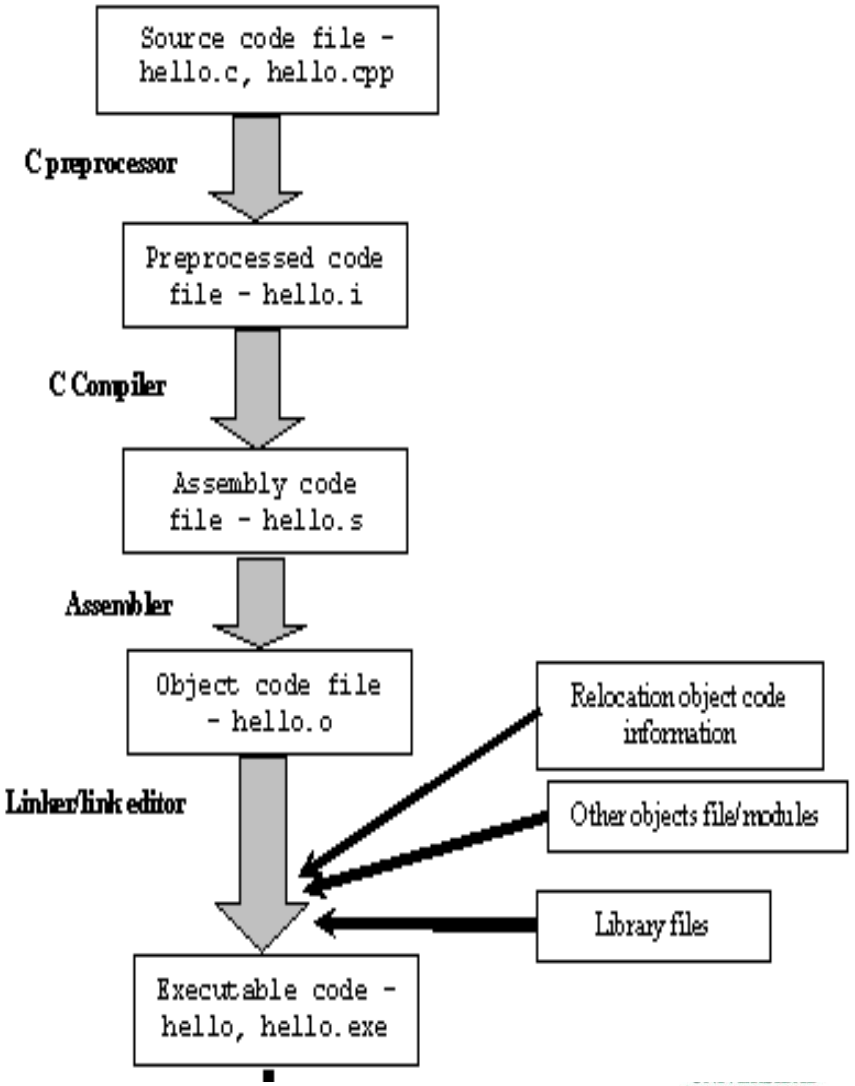
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

UNIVERSIDAD DE ANTIOQUIA
1803

Directivas del preprocesador

- Parte del compilador que realiza la primera etapa de traducción o compilación de un programa en instrucciones de máquina.

- Procesa el archivo fuente y actúa sobre ordenes denominadas **directivas del preprocesador**.



Directivas del preprocesador

- Comienzan con #.
- Permiten buenas prácticas de programación.
- Usos:
  1. Inserción de contenido de un archivo en su programa: #include
  2. Definición de macros para reemplazar una cadena por otra: #define
  3. Permite la compilación de partes de un programa (Escritura de código fuente para dos o mas sistemas) : #if, #ifdef, #else, #endif,

```
#define TEST "Hello, World!"
const char str[] = TEST;
```

gcc -E test.c

```
const char str[] = "Hello, World!" ;
```

Archivos cabecera

- #include <stdio.h>
- #include “demo.h”

```
#include <stdio.h>
int main (void) {
 printf ("Hello, world!\n");
 return 0;
}
```

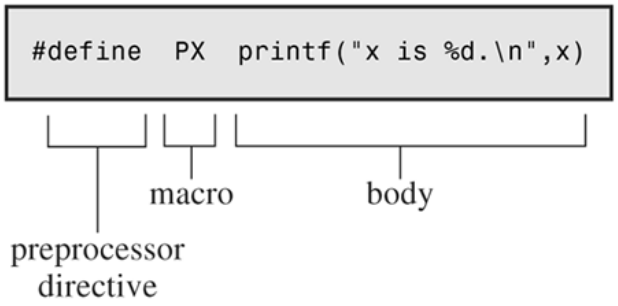
gcc -E hello.c

```
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3

extern FILE *stdin;
extern FILE *stdout;
extern FILE *stderr;
extern int fprintf (FILE * __stream,
                    const char * __format, ...) ;
extern int printf (const char * __format, ...) ;

[ ... additional declarations ... ]

# 1 "hello.c" 2
int main (void) {
  printf ("Hello, world!\n");
  return 0;
}
```

UNIVERSIDAD DE ANTIOQUIA
1803

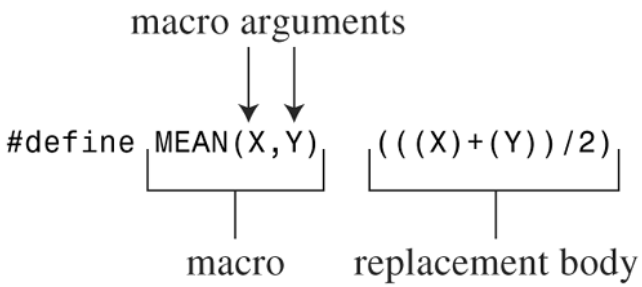
Definición de macros

- Una macro define un símbolo equivalente a una parte de código en C.
- Para definir macros se utiliza la directiva **#define**.




```
#define PI 3.14159
#define IVA 16
#define BUFFER 1024
```

- Es posible definir macros que acepten argumentos




```
#define cuadrado (x) ((x)*(x))
```

UNIVERSIDAD DE ANTIOQUIA
1803

Tipos de datos básicos

- Los tipos de datos básicos incorporados en C son **enteros**, **reales** y **carácter**.

Tipos de datos enteros

Tipo de dato	Tamaño en bytes	Tamaño en bits	Valor mínimo	Valor máximo
signed char	1	8	-128	127
unsigned char	1	8	0	255
signed short	2	16	-32.768	32.767
unsignet short	2	16	0	65.535
signed int	2	16	-32.768	32.767
unsigned int	2	16	0	65.535
signed long	4	32	-2.147.483.648	2.147.483.647
unsigned long	4	32	0	4.294.967.295

Tipos de datos reales

Tipo de dato	Tamaño en bytes	Tamaño en bits	Valor mínimo	Valor máximo
float	4	32	3.4E-38	3.4E+38
double	8	64	1.7E-308	1.7E+308
long double	10	80	3.4E-4932	1.1E+4932

Tipos de datos básicos

- De los tipos de datos enteros, el tipo char se utiliza para representar caracteres.
- Constantes tipo char:
  - Pueden ser caracteres encerrados entre comillas (‘A’, ‘b’, ‘p’)
  - Caracteres no imprimibles (tabulación, avance de pagina) los cuales se representan con secuencias de escape (‘\n’, ‘\t’).

Secuencias de escape

Carácter	Significado	Código ASCII
\a	Carácter de alerta (timbre)	7
\b	Retroceso de espacio	8
\f	Avance de página	12
\h	Nueva línea	10
\r	Retorno de carro	13
\t	Tabulación (horizontal)	9
\v	Tabulación (vertical	11
\\	Barra inclinada	92
\?	Signo de interrogación	63
\´	Comilla	39
\´´	Doble comilla	34
\nnn	Número octal	-
\xnn	Número hexadecimal	-
´\0´	Carácter nulo (terminación de cadena)	-

UNIVERSIDAD DE ANTIOQUIA
1803

Variables

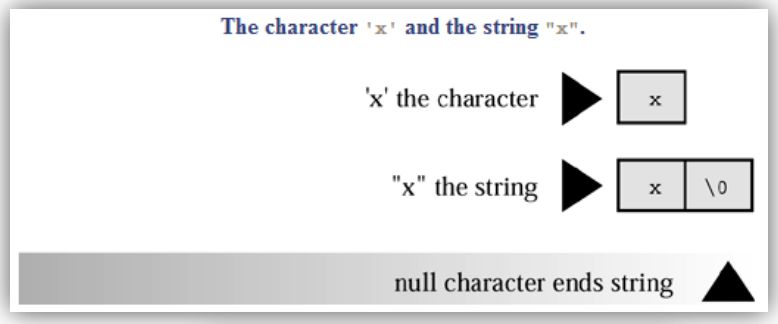
- **Todas las variables en C se declaran o definen antes de ser utilizadas.**
- Una declaración indica el tipo de variable.
- Si la declaración produce también un almacenamiento (se inicia), entonces es una definición.

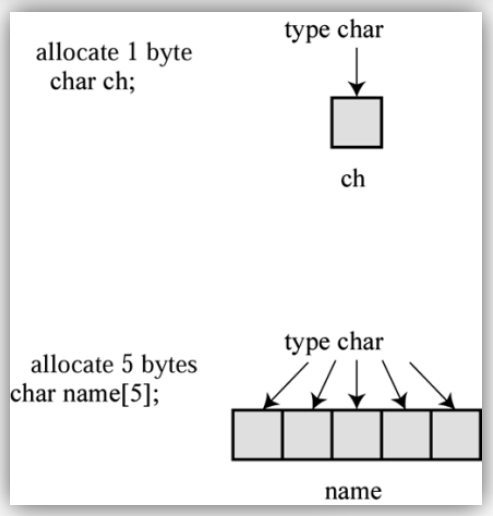
```
int x    , no = 0;                  //definen las variables x y no
float total = 42.125                //define la variable total

int v1, int v2, int v3, int v4;

const int z = 4350

char car = ´M´;
```

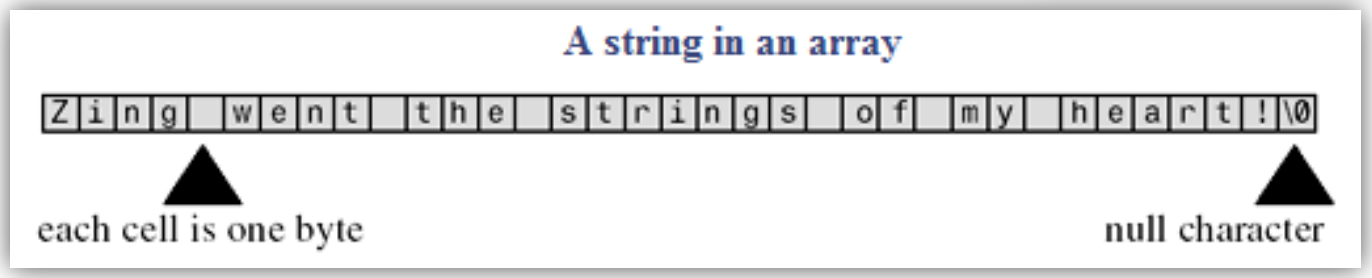
La palabra reservada **const** permite definir determinadas variables con valores constantes que no se pueden modificar

Cadenas de caracteres

- Constan de 0 o mas caracteres colocados entre comillas dobles.
- La cadena se almacena en memoria como una serie de valores ASCII de tipo **char** de un solo byte y se termina con un byte 0 (Carácter nulo)
- Si la declaración produce también un almacenamiento (se inicia), entonces es una definición.







UNIVERSIDAD DE ANTIOQUIA
1803

Tipos enumerados

- El tipo **enum** es una lista ordenada de elementos como constantes enteras.
- A menos que se indique lo contrario el primer miembro del conjunto enumerado toma el valor de 0, sin embargo, se pueden especificar los valores.

```
enum diasSemana {Lunes, Martes, Miercoles, Jueves, Viernes,
Sabado, Domingo};
```

- Un tipo enumerado se puede usar para declarar una variable

```
enum diasSemana Laborable;
```

- Algunos ejemplos de uso

```
Laborable = Sabado;
if(Laborable >= Viernes) {
  printf("Hoy no hay que trabajar");
}
```

UNIVERSIDAD DE ANTIOQUIA
1803

Expresiones y operadores

Operadores aritméticos

Operador	Descripción	Ejemplo
*	Multiplicación	(a*b)
/	División	(a/b)
+	Suma	(a+b)
-	Resta	(a-b)
%	Módulo	(a%b)

Operadores relacionales

Operador	Descripción	Ejemplo
<	Menor que	(a<b)
<=	Menor que o igual	(a <= b)
>	Mayor que	(a > b)
>=	Mayor o igual que	(a >= b)
= =	Igual	(a = = b)
!=	No igual	(a != b)

UNIVERSIDAD DE ANTIOQUIA
1803

Expresiones y operadores

Operadores de incremento y decremento

Operador	Descripción	Ejemplo
++	Incremento en i	++i, i++
--	Decremento en i	--j, j--

Operadores de manipulación de bits

Operador	Descripción	Ejemplo
&	AND bit a bit	C = A&B;
|	OR inclusiva bit a bit	C = A|B;
^	OR exclusiva bit a bit	C = A^B;
<<	Desplazar bits a izquierda	C = A<<B;
>>	Desplazar bits a derecha	C = A>>B;
-	Complemento a uno	C = -B

Operadores de asignación

O = a *op* b ⟷ a *op* = b;

UNIVERSIDAD DE ANTIOQUIA
1803

Prioridad y precedencia de operadores

- Las expresiones en C constan de diversos operandos y operadores.

Orden de evaluación y prioridad de operadores (asociatividad)

Nivel	Operadores	Orden de evaluación
1	( ) . [ ] >	izquierda-derecha
2	* & ! ~ ++ - - + - *(conversión de tipo sizeof)*	derecha-izquierda
3	* / %	izquierda-derecha
4	+ -	izquierda-derecha
5	<< >>	izquierda-derecha
6	< <= > >=	izquierda-derecha
7	== !=	izquierda-derecha
8	&	izquierda-derecha
9	^	izquierda-derecha
10	|	izquierda-derecha
11	&&	izquierda-derecha
12	||	izquierda-derecha
13	?:	derecha-izquierda
14	= *= /= += -= %= <<= >>= &= ^= |=	derecha-izquierda
15	,	izquierda-derecha

UNIVERSIDAD DE ANTIOQUIA
1803

Sentencias de control

- Las Instrucciones de control en c son similares a las de Java: **if, switch, for y while.**

```
condition ? expression1 : expression2;
```

```
if (expression)
   statement1;
else
   statement2;
```

```
for (initialization expression; test expr; increment expr)
program statement;
```

```
while(control expression)
   program statement;
```

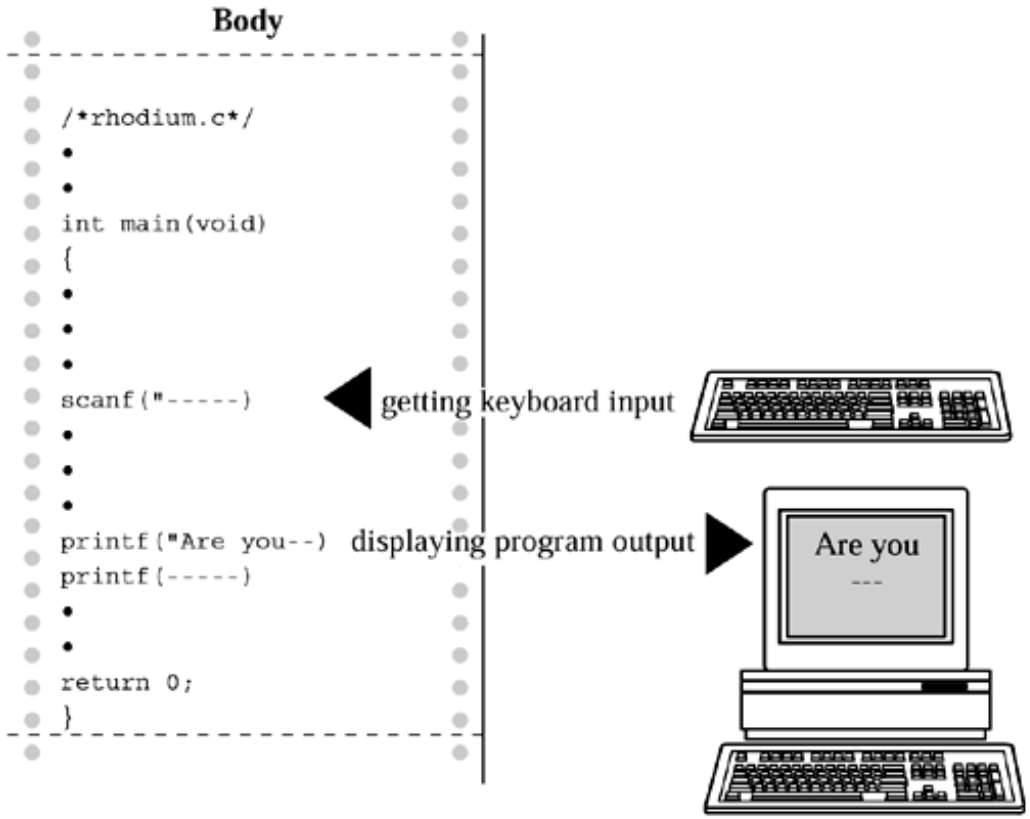
```
switch (integer expression) {
   case constant1:
      statement1;
      break;
   case constant2:
      statement2;
      break;
   ...
   default:
      statement;
}
```

```
do
   program statement;
while (control expression);
```

UNIVERSIDAD DE ANTIOQUIA
1803

Entrada y salida

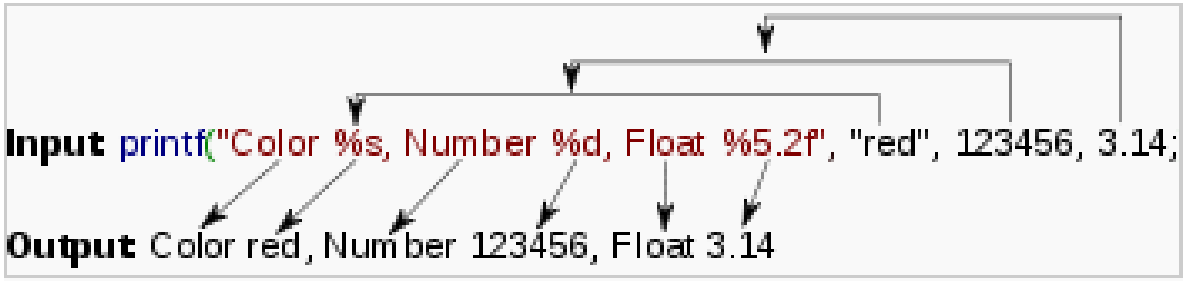
- Las funciones **scanf** y **printf** permiten comunicarse con un programa.
- Ambas funciones utilizan una cadena de control y una lista de argumer



Salida

- Las funcione empleada para esto es **printf** or **fprintf**
- Además de la función **printf** existen otras funciones de salida como **putc**, **puts** entre otras.
- Para poderla usar se tiene que incluir el archivo **stdio.h**
- **Sintaxis:**
  - **Cadena de control**: Determina el formato de escritura de los datos.
  - **Datos (dato1, dato2,…,datoN)**: Datos o variables de datos a escribir.

```
printf("cadena de control"<,dato1,dato2,...,datoN>);
```

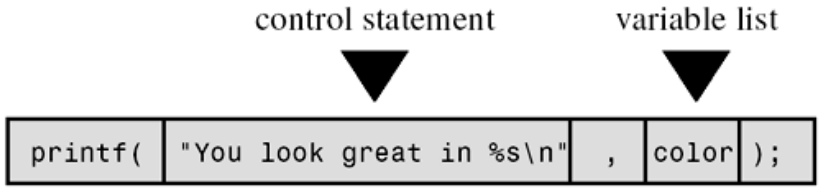


UNIVERSIDAD DE ANTIOQUIA
1803

Salida

```
printf("cadena de control"<,dato1,dato2,...,datoN>);
```

- **Cadena de control**: Tiene 2 componentes:
  1. **Texto**.
  2. **Identificadores**. (**%código**; donde el código indica el formato de salida de la variable).



Códigos de identificadores

Identificador	Formato
% d	Entero decimal
% c	Carácter simple
% s	Cadena de caracteres
% f	Coma flotante (decimal)
% e	Coma flotante (notación exponencial)
% g	Usa el %f o el %e más corto
% u	Entero decimal sin signo
% o	Entero octal sin signo
% x	Entero hexadecimal sin signo

Salida – algunos ejemplos

`printf("ABC");`	ABC (cursor after the C)
`printf("%d\n",5);`	5 (cursor at start of next line)
`printf("%c %c %c",'A','B','C');`	A B C

`printf("From sea ");`
`printf("to shining ");`
`printf ("C");`
	From sea to shining C

`printf("From sea \n");`
`printf("to shining \n");`
`printf ("C");`
	From sea
	to shining
	C

`leg1=200.3; leg2=357.4;` `printf("It was %f miles",leg1+leg2);`	It was 557.700012 miles
`num1=10; num2=33;` `printf("%d\t%d\n",num1,num2);`	10 33
`big=11e+23;` `printf("%e \n",big);`	1.100000e+24
`printf("%c \n",'?');`	?
`printf("%d \n",'?');`	63
`printf("\007 That was a beep\n");`	try it yourself

UNIVERSIDAD DE ANTIOQUIA
1803

Salida formateada

- Permite un mayor control de la apariencia de la salida en pantalla.
- Es posible controlar que tantas columnas serán usadas a la salida del contenido de una variable particular al especificar el **campo ancho (#)**.

%#especificador

```
%5d   Uso de 5 columnas para el despliegue de un entero d

%3c   Despliega el caracter c en 3 columnas

%13x  Despliega el entero hexadecimal en 13 columnas
```

- En los casos anteriores, el valor del argumento es justificado a la derecha y llenado con espacios (hasta completar el valor de campo ancho).
- Si se desea justificar a la izquierda se coloca un **menos (–)** antes del campo **ancho (#)**. Por ejemplo: **%-3c**.
- Si se desea llenar con ceros se debe colocar un **cero (0)** antes del campo ancho **ancho (#)**. Por ejemplo: **%04d**.

Salida formateada – caracteres y enteros

```
#include <stdio.h>
main() {
    char lett='w';
    int i=1,j=29;
    printf ("%c\n",lett);
    printf ("%4c\n",lett);
    printf ("%-3c\n\n",lett);
    printf ("%d\n",i);
    printf ("%d\n",j);
    printf ("%10d\n",j);
    printf ("%010d\n",j);
    printf ("%-010d\n",j);
    printf ("%2o\n",j);
    printf ("%2x\n",j);
}
```

```
w
   w
w

1
29
        29
0000000029
29
35
1d
```

UNIVERSIDAD DE ANTIOQUIA
1803

Salida formateada – Números reales

```
#include <stdio.h>
main() {
   float x=333.123456;
   double y=333.12345678901234 56;
   printf ("%f\n",x);
   printf ("%.1f\n",x);
   printf ("%20.3f\n",x);
   printf ("%-20.3f\n",x);
   printf ("%020.3f\n",x);
   printf ("%f\n",y);
   printf ("%.9f\n",y);
   printf ("%.20f\n",y);
   printf ("%20.4e\n",y);
}
```

```
333.123444
333.1
              333.123
333.123
0000000000000333.123
333.123457
333.123456789
333.12345678901232304270
           3.331e+02
```

%10.4f

field width

number of decimal places

printf("%10.4f",4.0/3.0); ⟶ ----1.3333

printf("%10.4e",4.0/3.0); ⟶ _1.333e+10

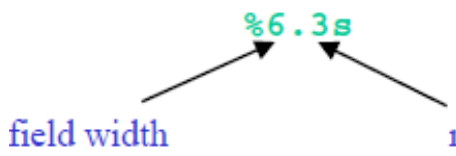
number of significant figures

UNIVERSIDAD DE ANTIOQUIA
1803

Salida formateada – Cadenas de caracteres

```c
#include <stdio.h>
main() {
    static char s[]="an evil presence";
    printf ("%s\n",s);
    printf ("%7s\n",s);
    printf ("%20s\n",s);
    printf ("%-20s\n",s);
    printf ("%.5s\n",s);
    printf ("%.12s\n",s);
    printf ("%15.12s\n",s);
    printf ("%-15.12s\n",s);
    printf ("%3.12s\n",s);
}
```

```
an evil presence
an evil presence
     an evil presence
an evil presence
an ev
an evil pres
    an evil pres
an evil pres
an evil pres
```

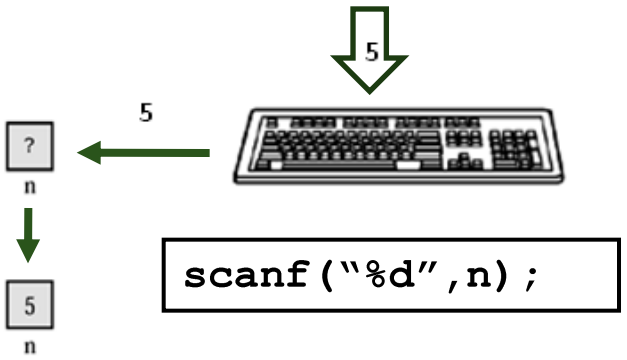


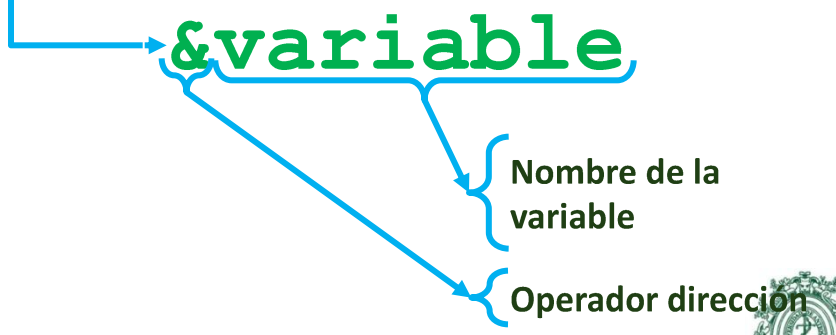

```c
printf("3.4s\n","Sheridan");
```
→ Sher

Entrada

- Las funcione empleada para esto es **scanf** o **fscanf**
- Además de la función **scanf** existen otras funciones de salida como **getc** y **gets** entre otras.
- Para poderla usar se tiene que incluir el archivo **stdio.h**
- **Sintaxis:**
  - **Cadena de control**: Determina el formato de escritura de los datos.
  - **Datos (dato1, dato2,…,datoN)**: Datos o variables de datos a escribir.

```
printf("cadena de control"<,<&>dato1,<&>dato2,...,<&>datoN>);
```





UNIVERSIDAD DE ANTIOQUIA
1803

Entrada

```
printf("cadena de control"<,<&>dato1,<&>dato2,...,<&>datoN>);
```

Identificadores de formato de scanf

Identificador	Formato
% d	Entero decimal
% c	Carácter simple
% s	Cadena de caracteres
% f	Coma flotante
% e	Coma flotante
% u	Entero decimal sin signo
% o	Entero octal sin signo
% x	Entero hexadecimal sin signo
& h	Entero corto

```
#include <stdio.h>
main()   {
   int pin;
   printf("Please type in your PIN\n");
   scanf("%d",&pin);
   printf("Your access code is %d\n",pin);}
```

```
Please type your PIN
4589
Your access code is 4589
```

UNIVERSIDAD DE ANTIOQUIA
1803

Entrada

Identificadores de formato de scanf

Identificador	Formato
% d	Entero decimal
% c	Carácter simple
% s	Cadena de caracteres
% f	Coma flotante
% e	Coma flotante
% u	Entero decimal sin signo
% o	Entero octal sin signo
% x	Entero hexadecimal sin signo
& h	Entero corto

```
printf ("Introduzca ciudad y provincia : ");
scanf ("%s %s, ciudad, provincia);
```

```
scanf ("%d", &cuenta);
scanf ("%205", direccion)
scanf ("%d%d", &r, &c);
scanf ("%d*c%d", &x, &y);
```

UNIVERSIDAD DE ANTIOQUIA
1803

Ejercicio

- Realizar un programa que cumpla las siguientes características
  1. Solicite el primer nombre del estudiante y almacenarlo como una cadena de caracteres.
  2. Nota obtenida y almacenarlo como un flotante.
  3. Imprima:
     - nombre: máx. 5 caracteres, alineado a la izq.
     - nota: ancho del campo 5, máx. 2 decimales
  4. La tecla q para terminar

Referencias

- Introduction to the C Programming Language, Science & Technology Support, High Performance Computing, Ohio Supercomputer Center
  - http://www.osc.edu/supercomputing/training/C/cprog_0506_pdf.pdf
- Linux Syscall quick reference
  - http://www.digilife.be/quickreferences/qrc/linux%20system%20call%20quick%20reference.pdf
- The basics of C programming
  - http://www.howstuffworks.com/c.htm
- Compiler, assembler, linker and loader: A brief history
  - http://www.tenouk.com/ModuleW.html
- The C storage classes, scope and memory allocation
  - http://www.tenouk.com/ModuleW.html

UNIVERSIDAD DE ANTIOQUIA
1803

Referencias

- Pointers in C++
  - http://www.cprogramming.com/tutorial/lesson6.html
- Programación en C
  - http://www.mhe.es/ceo_link.php?tipo=2_02_APN&isbn=8448198441&sub_materia=98&materia=21&nivel=U&comunidad=Castellano&ciclo=0&portal=&letrero=&cabecera=