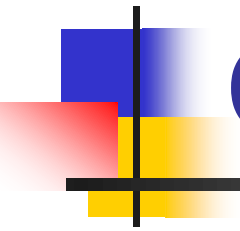
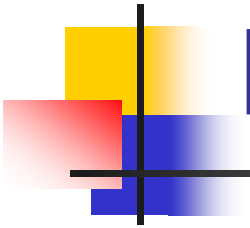


Curso Básico de C
Clase Nº 1

Características del Lenguaje
Comentarios, Palabras Claves e Identificadores
Constantes, Tipos de Datos y Variables
Inclusión de Archivos

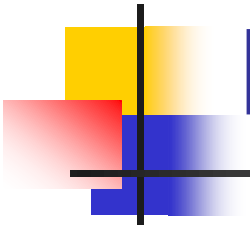
Prof. Patricia Guerrero



Lenguaje C

Características:

- Lenguaje de Alto Nivel
- Compilado
- Estructurado
- Es case sensitive
- Hace uso del concepto de función
- Portabilidad de los archivos fuente

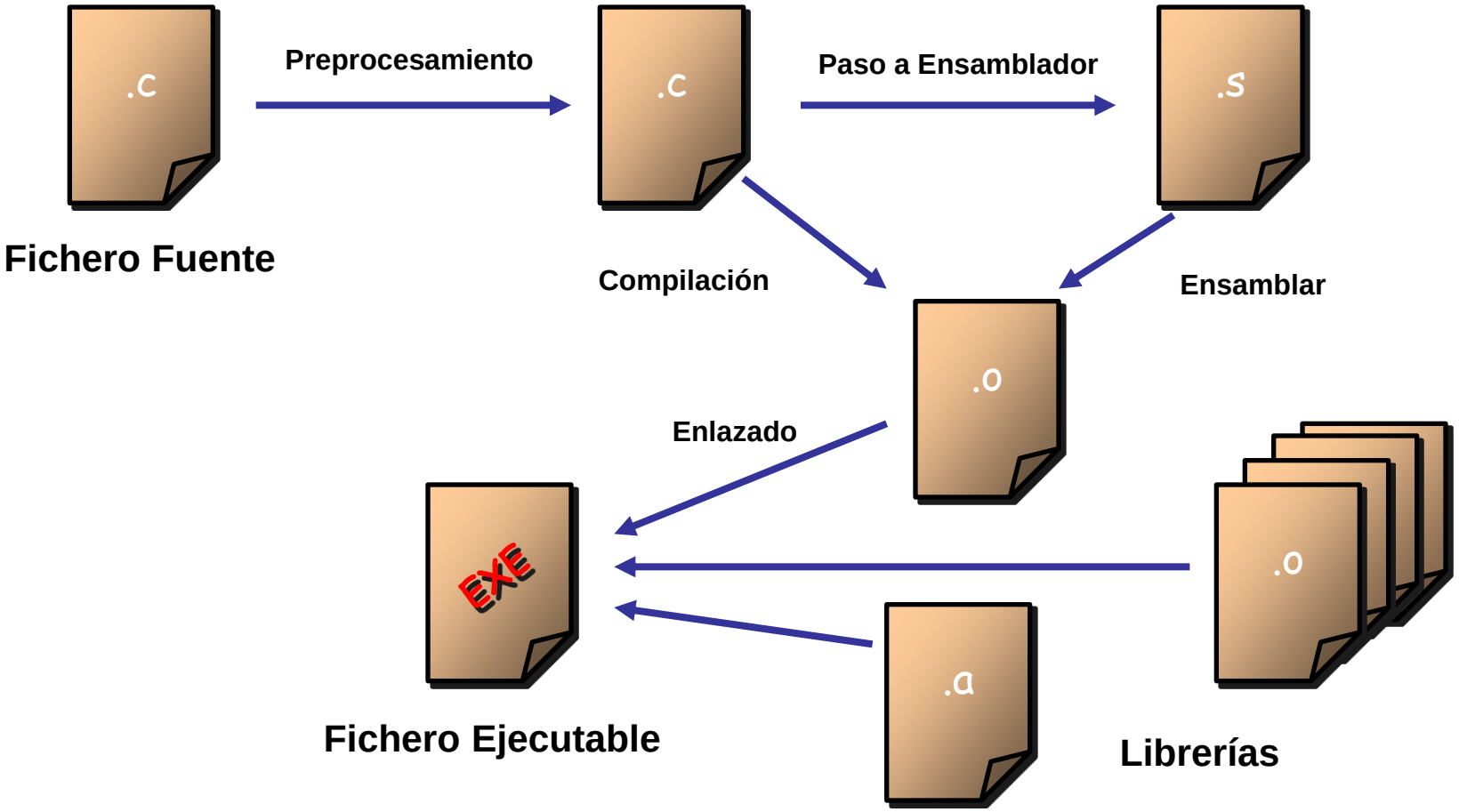


Lenguaje C

Características:

- C está constituido por tres elementos:
  - **Compilador:** Traduce código fuente a código máquina
  - **Preprocesador:** Facilita la tarea del compilador, permite sustituir constantes simbólicas e incluir archivos
  - **Librería Estándar:** Funciones preprogramadas agrupadas en un conjunto de librerías de código objeto

Lenguaje C


.c
Preprocesamiento
.c
Paso a Ensamblador
.s
Fichero Fuente
Compilación
Ensamblar
.o
Enlazado
EXE
.o
.a
Fichero Ejecutable
Librerías

Lenguaje C

Estructura básica de un programa:

```
Inclusión de librerías
Declaraciones globales

main ( ) {
        variables locales
        bloque de instrucciones
}

funcion1 ( ) {
        variables locales
        bloque de instrucciones
}
```

Un programa es un conjunto de instrucciones que generalmente se ejecutan de manera secuencial

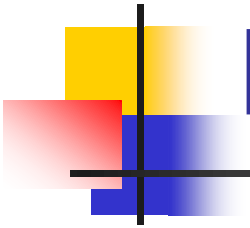
Una función es un subprograma, que es llamado por el programa principal

Lenguaje C

Comentarios:

Para poner comentarios en un programa escrito en C se usan los símbolos: /* y */ o //

```
/* Este es un ejemplo de comentario */

/* Un comentario también puede
estar escrito en varias líneas */

variable_1 = variable_2;        // En esta línea se asigna a
                                // variable_1 el valor
                                // contenido en variable_2
```

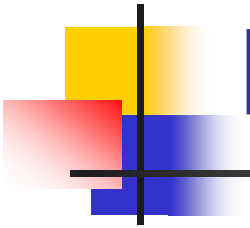


Lenguaje C

Palabras Claves:

Representan un conjunto de identificadores reservados

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



Lenguaje C

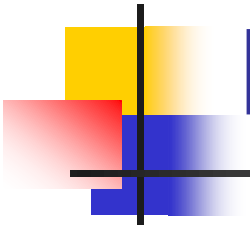
Identificadores:

Nombre dado a las variables o funciones.

1. Un identificador se forma con una secuencia de letras (minúsculas de la a a la z; mayúsculas de la A a la Z; y dígitos del 0 al 9)

2. El carácter subrayado o underscore (_) se considera como una letra más

3. Un identificador no puede contener espacios en blanco, ni otros caracteres distintos de los citados (*, ; . : - +)



Lenguaje C

Identificadores:

4. El primer carácter de un identificador debe ser siempre una letra o un (_)

5. Se hace distinción entre letras mayúsculas y minúsculas

6. ANSI C permite definir identificadores de hasta 31 caracteres de longitud

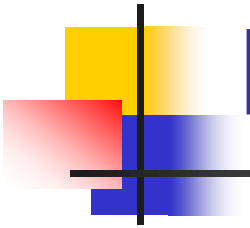
Lenguaje C

Identificadores

Ejemplo:

año_nac	velocidad_de_la_luz	Distancia1
tiempo-total	%final	1_valor
caso_A	PI	dolares$
_num	Tiempo	número2

Es aconsejable ***elegir los nombres*** de las funciones y variables de acuerdo a lo que representan



Lenguaje C

Constantes:

Mantienen su valor a lo largo de todo el programa

1. ***Constantes numéricas***. Son valores numéricos, enteros o de punto flotante. Se permiten también constantes *octales* y *hexadecimales*

2. ***Constantes carácter***. Cualquier carácter individual encerrado entre apóstrofos ('a' 'Y' ')' '+')

3. ***Cadenas de caracteres***. Un conjunto de caracteres alfanuméricos encerrados entre comillas

Lenguaje C

Constantes:

4. ***Constantes simbólicas***. Poseen un identificador y se definen mediante la palabra clave ***const***

23484	Constante tipo int
011	Constante octal (9 en base 10)
1.23	Constante tipo double
23.963f	Constante tipo float
'7'	Constante carácter. Valor ASCII 55
"Informática I"	Constante de carácteres
const int i = 10;	Constante simbólica

Lenguaje C

Constantes:

Para indicar al compilador que se trata de una constante, se usa la directiva **#define**:

#define <identificador> <valor>

```
#define pi 3.1416
main()
{
   int r = 10;
   float perimetro =  2*pi*r;
}
```

Lenguaje C

Constantes de Tipo Enumeración

Define los posibles valores de un identificador o variable.

Ejemplo:

```
enum  dia {lunes, martes, miercoles, jueves , viernes,
                    sabado, domingo};

enum dia dia1, dia2;
dia1 = martes;
```

Estos valores representan constantes de tipo int: **lunes** es **0**, **martes** es **1**, **miercoles** e **2**, y así sucesivamente

```
enum cartas {oros, copas, espadas, bastos} carta1, carta2;
```

Lenguaje C

Tipos de Datos:

Tipo	Tamaño	Rango de valores
char	1 byte	-128 a 127
int	4 bytes	-2.147.483.648 a 2.147.483.647
float	4 bytes	3,4 E-38 a 3,4 E+38
double	8 bytes	1,7 E-308 a 1,7 E+308

Lenguaje C

Calificadores de Tipo:

Modifican el rango de valores de un determinado tipo

- **Signed:** La variable lleva signo

	Tamaño	**Rango de valores**
signed char	1 byte	-128 a 127
signed int	4 bytes	-2.147.483.648 a 2.147.483.647

Lenguaje C

Calificadores de Tipo:

- **Unsigned:** La variable no lleva signo

	Tamaño	Rango de valores
unsigned char	1 byte	0 a 255
unsigned int	4 bytes	0 a 4.294.967.295

- **Short:** Rango de valores en formato corto

	Tamaño	Rango de valores
short char	1 byte	-128 a 127
short int	2 bytes	-32.768 a 32.767

Lenguaje C

Calificadores de Tipo:

- **Long:** Rango de valores en formato largo

	Tamaño	Rango de valores
long char	4 bytes	-2.147.483.648 a 2.147.483.647
long int	4 bytes	-2.147.483.648 a 2.147.483.647

También es posible combinar calificadores entre sí:

- signed long int = long int = long
- unsigned long int = unsigned long

  4 bytes 0 a 4.294.967.295 (El mayor entero de C)

Lenguaje C

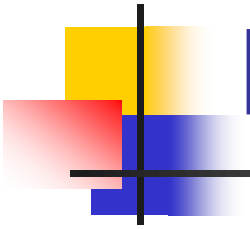
Variables:

Una variable es una posición de memoria referenciada por un identificador. Su valor puede cambiar durante la ejecución del programa.

Declaración e Inicialización:

Toda variable debe ser declarada antes de utilizarse en el programa

<tipo> <identificador> = valor;
<tipo> <identificador1>, <identificador2>, <identificador3>;



Lenguaje C

Variables - Ejemplo:

```
/* Uso de las variables para la suma de dos valores */

    main()
    {
        int num1 = 4, num2 = 6, num3;
        num3 = num1 + num2;
    }
```

Lenguaje C

Variables - ¿Dónde se declaran?

- Pueden ser declaradas antes de la función principal: ***Variables Globales***

Pueden ser utilizadas en cualquier parte del programa y viven durante toda la ejecución del mismo.

- Pueden ser declaradas dentro de una función: ***Variables Locales***

Sólo pueden ser utilizadas por la función que la declara. Se crean cuando se llama a la función y se destruye cuando ésta finaliza.

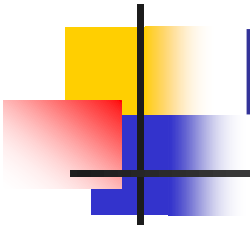
Lenguaje C

Variables - ¿Dónde se declaran?

- ***Variables Globales - Variables Locales***

Ejemplo:

```
int a;          /* variable global */
main()
{
      int b = 4;        /* variable local */
}
```



Lenguaje C

Variables - Duración y Visibilidad:

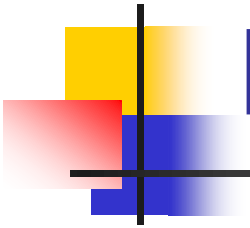
Se refiere al modo de almacenamiento, el cual determina **cuándo se crea una variable**, **cuándo deja de existir** y **desde dónde se puede acceder a ella**

- ***auto:*** Son variables locales. Sólo son visibles en el bloque donde están declaradas, se crean cuando comienza a ejecutarse el bloque y se destruyen cuando éste finaliza su ejecución. No son inicializadas por defecto.

Lenguaje C

Variables Auto - Ejemplo:

```
{
    int i = 1, j = 2;          //se declaran e inicializan i y j
    ...
    {
            float a = 7, j = 3;          // se declara una nueva variable j
            ...
            j = j + a;                   // aquí j es float
            ... // la variable int j es invisible
            ... // la variable i es visible
    }
    ... // fuera del bloque, a ya no existe
    ... // la variable j = 2 existe y es entera
}
```



Lenguaje C

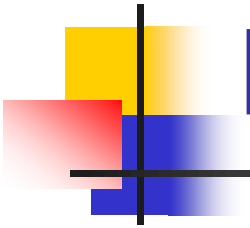
Variables - Duración y Visibilidad:

- ***extern:*** Son variables globales. Son visibles por todas las funciones que están entre la definición y el fin del fichero. Existen durante toda la ejecución del programa. Son inicializadas en cero.

Lenguaje C

Variables Extern - Ejemplo:

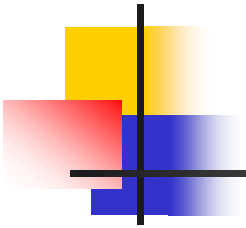
```
int i = 1, j, k;              // se declaran antes de main()
int func1(int, int);

main()
{
      int i = 3;              // i = 1 se hace invisible
      ...                     // j, k son visibles
}

int func1(int i, int m)
{
      int k = 3;              // k = 0 se hace invisible
      ...                     // i = 1 es invisible
}
```



Lenguaje C

Variables - Duración y Visibilidad:

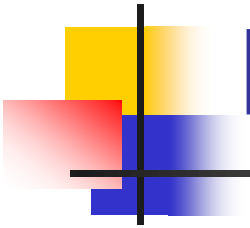
- ***static:*** Las variables static declaradas dentro de un bloque conservan su valor en las llamadas sucesivas a dicho bloque. Su inicialización sólo se realiza la primera vez.



Lenguaje C

Variables Static - Ejemplo:

```
int Series()
{
    static int contador;
    contador = contador +10;
    return contador;
}
```



Lenguaje C

Variables - Conversiones

Casting (Conversión Explicita): Mecanismo usado para cambiar de tipo expresiones y variables.

```
int a;
float b;
char c;

b = 65.0;
a = (int)b;       /*a vale 65 */
c = (char)a;  /* c vale 65 (Código ASCII de 'A') */
```

Lenguaje C

Variables - Conversiones

Conversión Implícita: Ocurre cuando en una expresión se combinan variables de diferentes tipos.

```
int a;
float b;
```

a + b

Ocurre una **promoción**:
a es promovida al tipo de b

long double > double > float > unsigned long > long > unsigned int > int > char

```
int x;
double i = 2.5, j = 5.3;
x = ((i * j) – j) + 1;
```

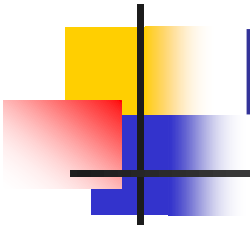
Conversión por **asignación**

Lenguaje C

Sentencias de Escape:

Para definir las constantes de tipo carácter asociadas a caracteres especiales se usan secuencias de escape:

Nombre completo	Constante	en C	ASCII
sonido de alerta	BEL	\a	7
nueva línea	NL	\n	10
tabulador horizontal	HT	\t	9
retroceso	BS	\b	8
retorno de carro	CR	\r	13
salto de página	FF	\f	12
barra invertida	\	\\	92
apóstrofo	'	\'	39
comillas	"	\"	34
carácter nulo	NULL	\0	0



Lenguaje C

Inclusión de Archivos:

- La directiva **#include** permite añadir librerías o funciones que se encuentran en otros archivos.
  - Indicando al compilador la ruta donde se encuentra el archivo
    #include "C:\includes\funcion.h"
  - Indicando que se encuentran en el directorio por defecto del compilador
    #include <funcion.h>