Caracteres y Cadenas
Conversión de Datos
Funciones y procedimientos
Archivos cabecera

Fundamentos de programación

Agenda

- Caracteres y Cadenas
- Conversión de Datos
- Introducción a las funciones y procedimientos
- Archivos cabecera
- Librería matemática de C

Tipo de datos: Caracter y Cadenas de carácter (String)

- El tipo de dato carácter (char) es utilizado para el almacenamiento de una letra. Ej.: ‘A’
- Las cadenas son datos que contienen letras, y/o números, y/o caracteres especiales, como espacios en blanco, asteriscos, entre otros. Ej.: “HOLA”
- El caracter ocupa 1 byte de memoria.
- Las cadenas de caracteres seran tratadas más adelante.

Tipo de datos carácter y cadenas de carácter (String)

- El código ASCII es el mas usado para representar caracteres
- ASCII define 127 caracteres. Ej.:
  - Caracter A = código ASCII 65

Operaciones con caracteres

- Con los caracteres se pueden aplicar las mismas funciones que sobre los números enteros, debido a que en la computadora cada carácter tiene su código numérico correspondiente. Ejemplo:
  - ‘A’ + 2 == ‘C’
  - ‘E’ – 1 == ‘D’
- En lenguaje C, sobre las cadenas, se pueden efectuar operaciones, a través de “funciones”, más no de operadores.

Conversión de tipo de datos

- Convertir de un tipo de datos a otro. Ej.: de un entero (3) a un real (3.0)
- Puede haber pérdida de datos en el proceso. Al convertir de un real (3.2) a un entero (3), por ejemplo.
- Existen dos formas de conversión de tipo de datos: implícita y explícita.

Conversión de tipo de datos Implícita

- Se da en forma automática.
- Esta se efectúa cuando realizamos una asignación de un valor de un tipo de datos a uno de diferente tipo. Por ejemplo:
  - Si **n1** es una variable entera. Al realizar la siguiente asignación:
  - n1 = 4.55
  - n1 se trunca, solo se le asigna 4.

Conversión de tipo de datos Explícita

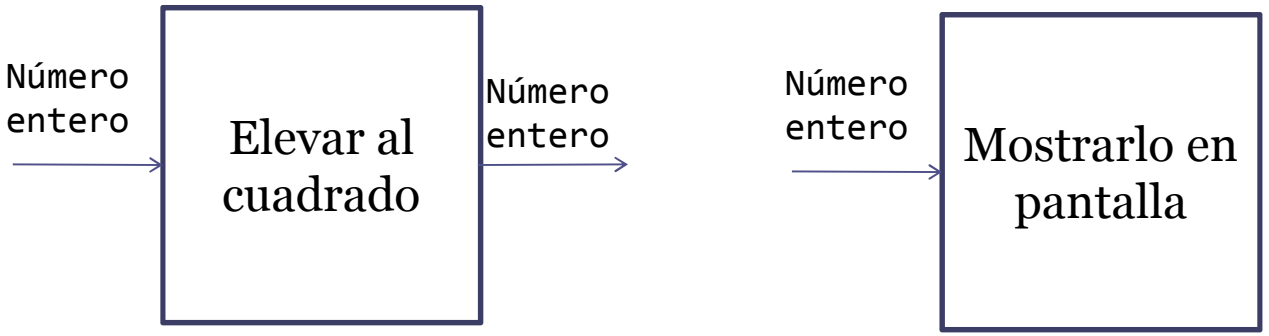
- **Llamada también "cast"**
- **En el lenguaje C se utiliza el operador (cast)**
- Este operador se lo utiliza, anteponiendo al valor, el tipo de dato entre paréntesis. Por ejemplo:
- Si **n1** es una variable entera. Al realizar la siguiente asignación:
- n1 = **(int)** 4.55;
- n1 se trunca, solo se le asigna 4.
- **De forma general:**
- variable = *(tipo de dato)* valor;

Módulos de programa en C

- Un programa en C contiene:
  - Funciones definidas por el programador: son aquellas que escribe el programador.
  - Funciones “pre-empacadas”: disponibles al programador en librerías (stdio.h):

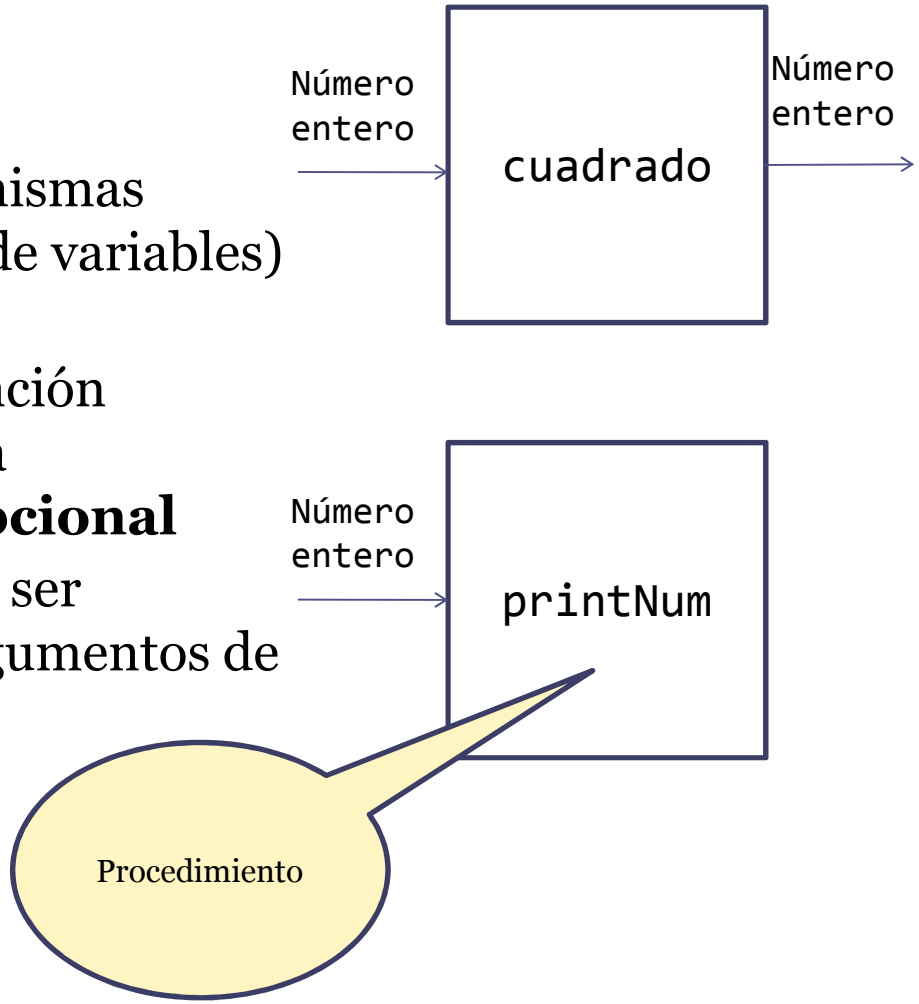
Funciones en C

- Realizan una tarea específica.
- Pueden o no recibir valores.
- Pueden o no retornar valores.
  - Las funciones que no retornan valores son llamadas procedimientos.



Funciones en C

- Las funciones en C tienen:
  - Un **identificador único** (mismas reglas para definir nombres de variables) .
  - Valores de entrada que la función necesita para realizar la tarea (argumentos de entrada). **Opcional**
  - Un valor de salida que puede ser entregado por la función (argumentos de salida). **Opcional**

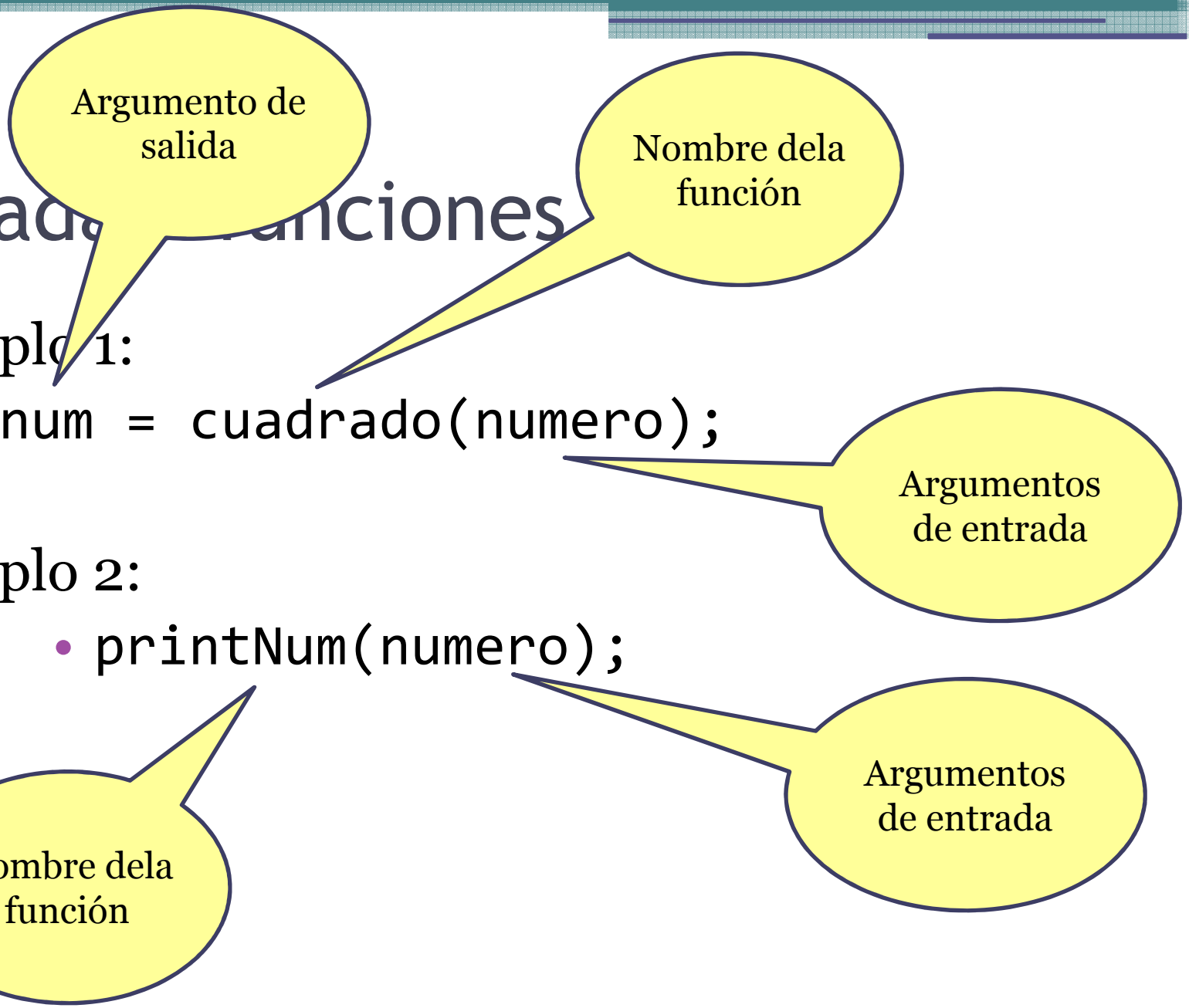


Llamada a funciones

- Las funciones se invocan mediante una llamada a una función.
- La llamada a una función se la realiza:
  - Utilizando su identificador único.
  - Enviando los argumentos necesarios para que la función pueda realizar su labor.
- La función realiza su tarea y retorna un valor (si es el caso)
- Algunas veces no sabemos cómo realiza su tarea, solo nos importa los resultados.

Llamada a funciones

- Ejemplo 1:
  - `num = cuadrado(numero);`
- Ejemplo 2:
  - `printNum(numero);`



Llamada a funciones

- `#include <stdio.h>`

Librería que contiene la función.
Puede ser nuestra propia librería

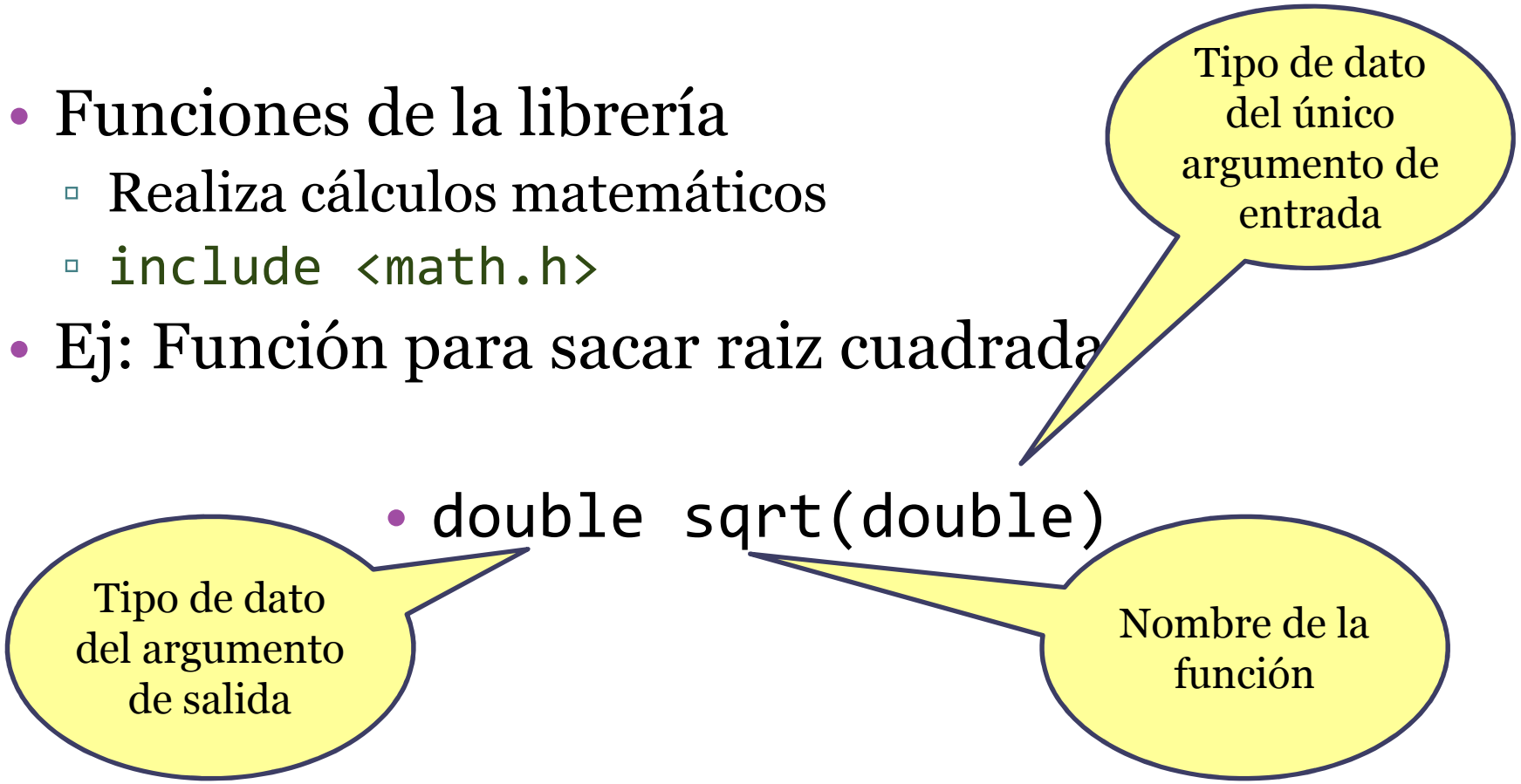
- `printf("El valor es %i", numero);`

Nombre dela función

Dos argumentos de entrada.
Varios argumentos de entrada son separados con comas

Funciones de la librería Math

- Funciones de la librería
  - Realiza cálculos matemáticos
  - `include <math.h>`
- Ej: Función para sacar raiz cuadrada
  - `double sqrt(double)`

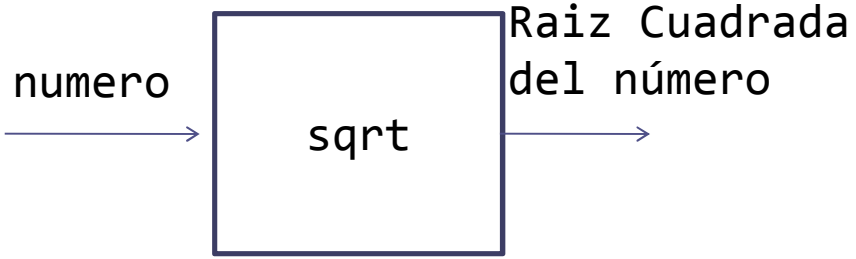


Funciones de la librería Math

```
#include <stdio.h>
#include <math.h>

main()
{
    double cuad, entrada;
    printf("Ingrese un numero: ");
    scanf("%i", &entrada);
    cuad = sqrt(entrada);
    printf("La raiz cuadrada de %d es: %d", entrada, cuad);
}
```

Llamada a la función



Funciones de la librería Math

Function	Description	Example
`sqrt( x )`	square root of x	`sqrt( 900.0 ) es 30.0` `sqrt( 9.0 ) es 3.0`
`exp( x )`	exponential function e^x	`exp( 1.0 ) es 2.718282` `exp( 2.0 ) es 7.389056`
`log( x )`	natural logarithm of x (base e)	`log( 2.718282 ) es 1.0` `log( 7.389056 ) es 2.0`

`log10( x )`	logarithm of x (base 10)	`log10( 1.0 ) es 0.0`
		`log10( 10.0 ) es 1.0`
		`log10( 100.0 ) es 2.0`

`fabs( x )`	absolute value of x	`fabs( 5.0 ) es 5.0`
		`fabs( 0.0 ) es 0.0`
		`fabs( -5.0 ) es 5.0`

`ceil( x )`	rounds x to the smallest integer not less than x	`ceil( 9.2 ) es 10.0` `ceil( -9.8 ) es -9.0`
`floor( x )`	rounds x to the largest integer not greater than x	`floor( 9.2 ) es 9.0` `floor( -9.8 ) es -10.0`
`pow( x, y )`	x raised to power y (x^y)	`pow( 2, 7 ) es 128.0` `pow( 9, .5 ) es 3.0`
`fmod( x, y )`	remainder of x/y as a floating point number	`fmod( 13.657, 2.333 ) es 1.992`
`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 ) es 0.0`
`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 ) es 1.0`
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 ) es 0.0`

Funciones de la librería Math más comunes

Función main

- Cada programa que hemos hecho se encuentra dentro de lo que se denomina la función principal (main).
- Main también es una función que puede recibir parámetros y retornar valores.

Función main

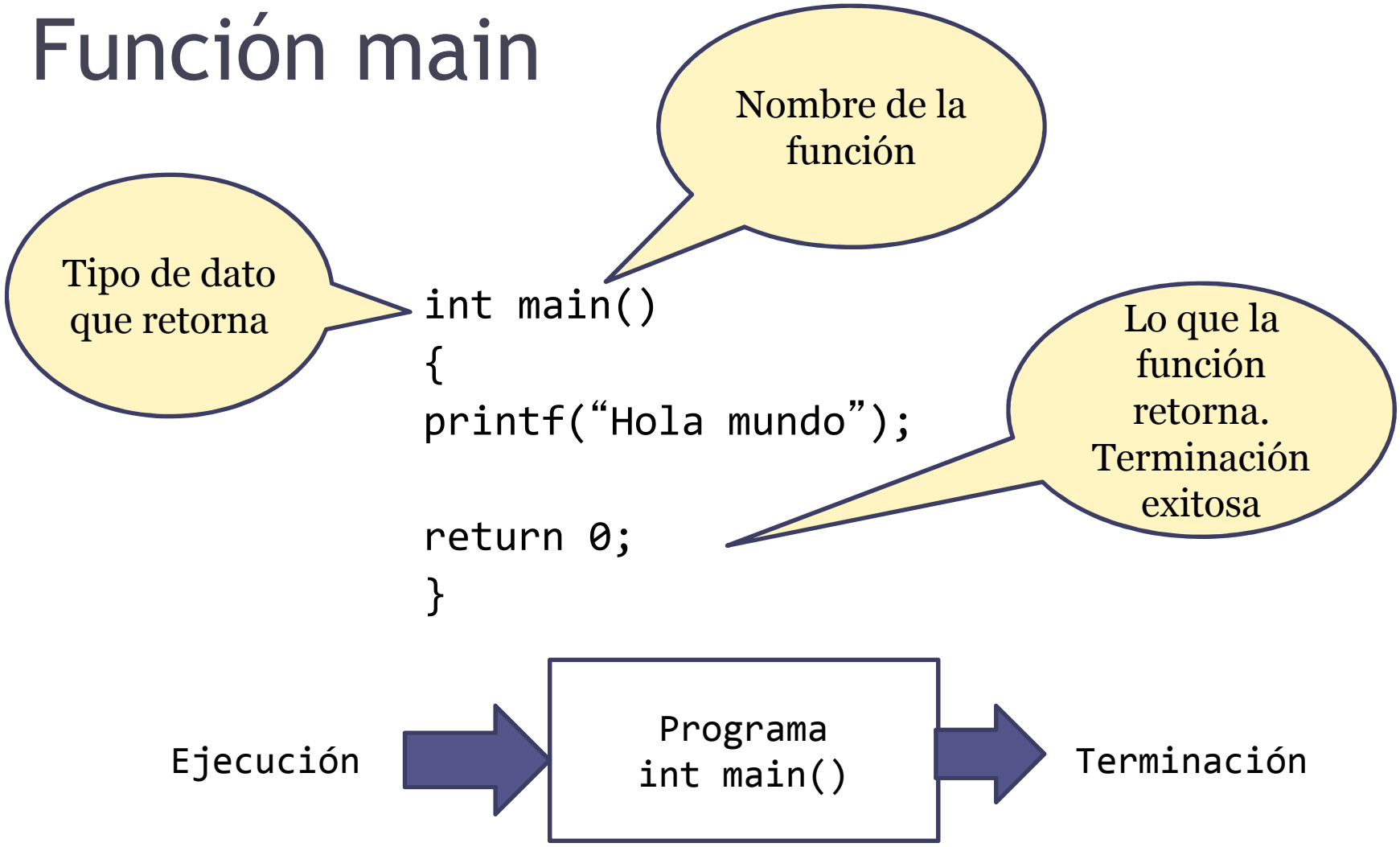
- main()
- {
  - printf(“Hola mundo”);
- }

```
void main()
{
printf(“Hola mundo”);
}
```

```
int main()
{
printf(“Hola mundo”);
return 0;
}
```

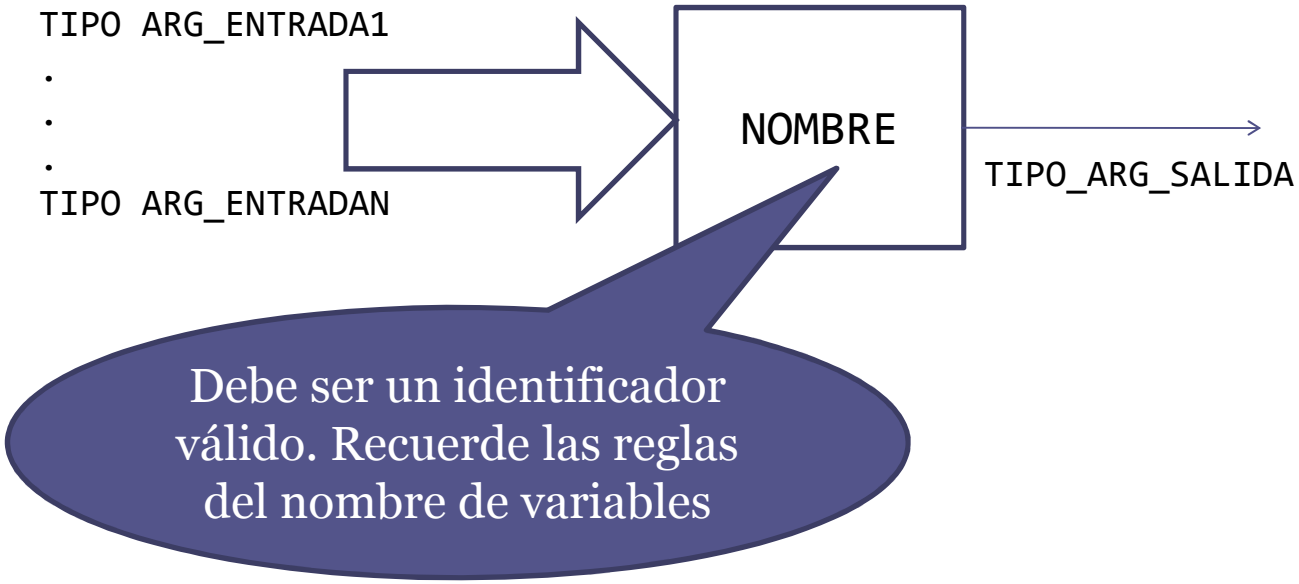
```
int main(int n, char**
  args)
{
printf(“Hola mundo”);
return 0;
}
```

Función main



Funciones propias:
Prototipo y definición

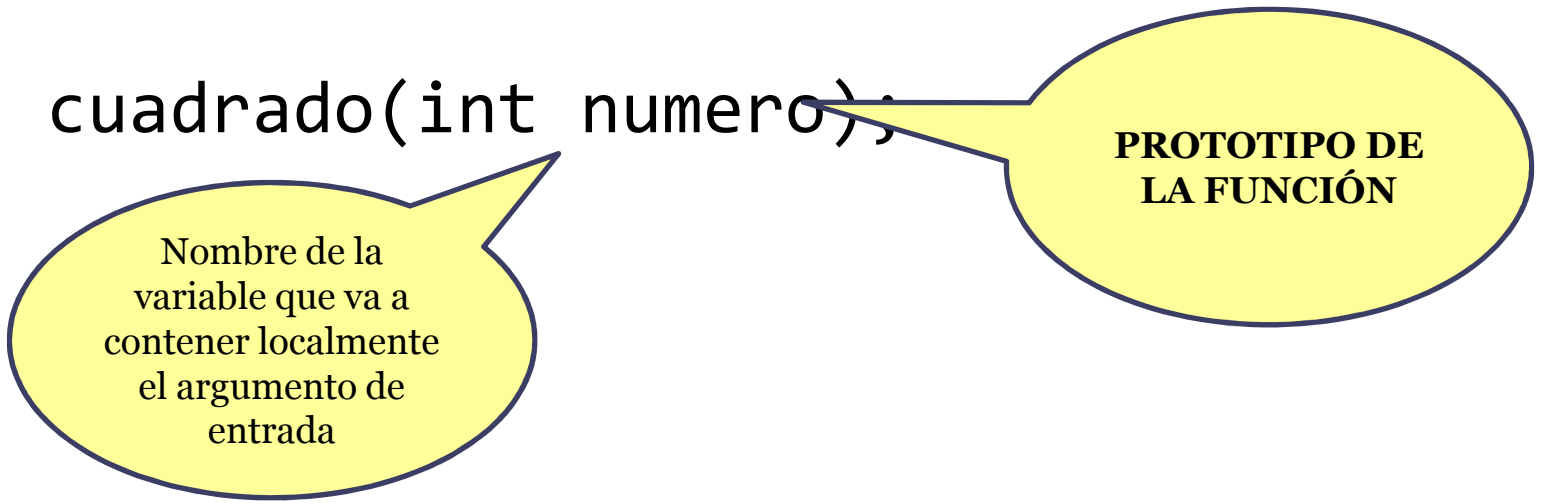
- Formato:
- `TIPO_ARG_SALIDA NOMBRE(TIPO_1 ARG_ENTRADA1 ,.., TIPO_N ARG_ENTRADAN)`



Funciones propias:
Prototipo y definición

- Suponga que queremos hacer una función llamada cuadrado la cual reciba como parámetro un número y retorne el cuadrado de dicho número.

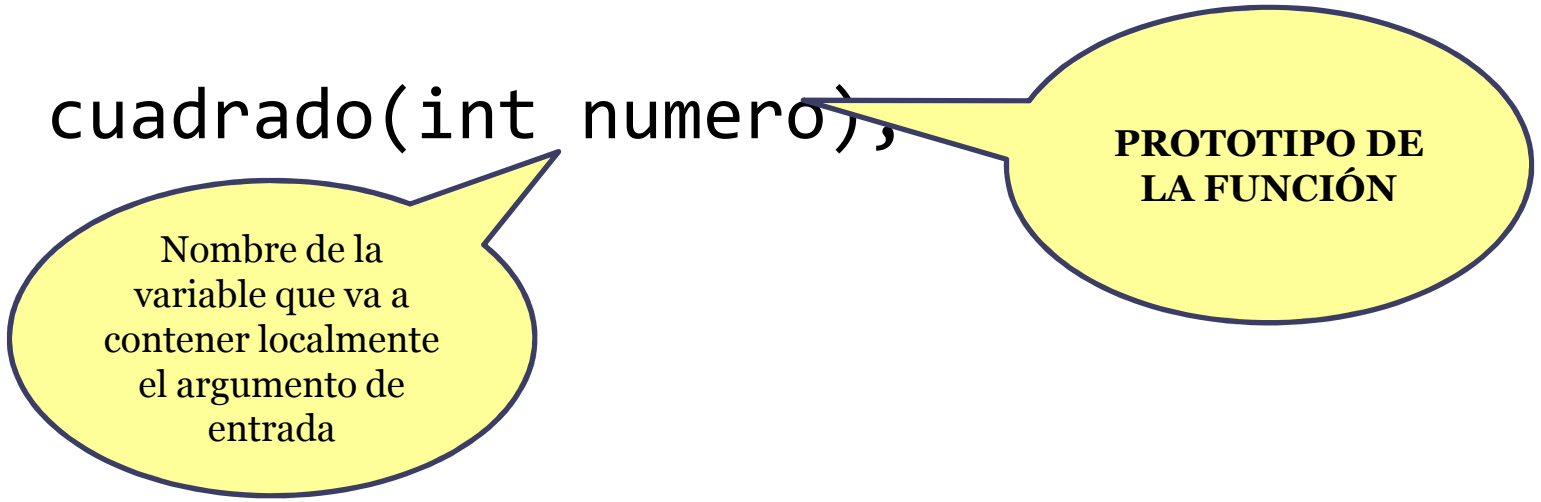
- `int cuadrado(int numero);`



Funciones propias:
Prototipo y definición

- El **prototipo** nos permite conocer el número de parámetros y su tipo de dato, así como el tipo de dato del parámetro de salida.

- `int cuadrado(int numero);`



Funciones propias:
Prototipo y definición

```
int cuadrado(int numero)
{
        int cuad;
        cuad = numero*numero;
        return cuad;
}
```

DEFINICIÓN DE LA FUNCIÓN. IMPLEMENTACIÓN DEL ALGORITMO

Retorna el mismo tipo de dato definido como salida de la función. **El retorno es obligatorio**

Funciones propias:
Prototipo y definición

- La definición implementa el algoritmo de la función. Es como se resuelve el problema. En este caso de elevar al cuadrado

```
int cuadrado(int numero)
{
        int cuad;
        cuad = numero*numero;
        return cuad;
}
```

Funciones propias

```
#include <stdio.h>

//Prototipos
int cuadrado(int numero);

int main()
{
    //Variables
    int numero, n;
    printf("Ingrese un numero: ");
    scanf("%i", &numero);
    n = cuadrado(numero);
    printf("El cuadrado de %i es %i", numero,n);
}

//Definicion
int cuadrado(int n)
{
    int cuad;
    cuad = n*n;
    return cuad;
}
```

El prototipo se ubica antes de la función main.

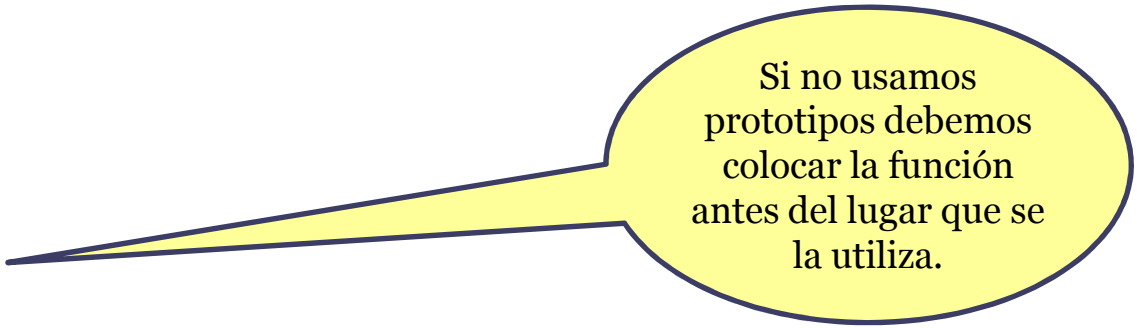
La definición después de la función main.

Funciones propias

```
#include <stdio.h>

//Definicion
int cuadrado(int n)
{
    int cuad;
    cuad = n*n;
    return cuad;
}

int main()
{
    //Variables
    int numero, n;
    printf("Ingrese un numero: ");
    scanf("%i", &numero);
    n = cuadrado(numero);
    printf("El cuadrado de %i es %i", numero,n);
}
```



Funciones propias
Sin valor de retorno

- Llamadas procedimientos.
- El valor de retorno se lo define como void
- No necesita el return;

    - `void muestraLetra(char letra)`

Ejemplos

- Implemente una función que reciba tres números y muestre el máximo entre los tres.
- Implemente una función que permita calcular la hipotenusa de un triángulo dados sus catetos.
- Implemente una función que permita calcular el factorial de un número.

Funciones: Definición y Prototipos

- Archivos de Encabezamiento
  - Contiene los prototipos para la librería de funciones
  - <stlib.h>, <math.h>
  - #include <nombre-del-archivo>
    - #include <math.h>

Funciones: Definición y Prototipos

- Archivos de Encabezamiento personalizados
  - Crear un archivo con funciones
  - Guardar con extensión .h
  - Usar en otros archivo con #include “nombre.h”
  - Usar guardias de definición:
    - #ifndef - #define - #endif
  - Reusar funciones

Funciones: Definición y Prototipos

Standard library header	Explanation
`<assert.h>`	Contains macros and information for adding diagnostics that aid program debugging.
`<ctype.h>`	Contains function prototypes for functions that test characters for certain properties, and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa.
`<errno.h>`	Defines macros that are useful for reporting error conditions.
`<float.h>`	Contains the floating point size limits of the system.
`<limits.h>`	Contains the integral size limits of the system.
`<locale.h>`	Contains function prototypes and other information that enables a program to be modified for the current locale on which it is running. The notion of locale enables the computer system to handle different conventions for expressing data like dates, times, dollar amounts and large numbers throughout the world.
`<math.h>`	Contains function prototypes for math library functions.
`<setjmp.h>`	Contains function prototypes for functions that allow bypassing of the usual function call and return sequence.
`<signal.h>`	Contains function prototypes and macros to handle various conditions that may arise during program execution.
`<stdarg.h>`	Defines macros for dealing with a list of arguments to a function whose number and types are unknown.
`<stddef.h>`	Contains common definitions of types used by C for performing certain calculations.
`<stdio.h>`	Contains function prototypes for the standard input/output library functions, and information used by them.
`<stdlib.h>`	Contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers, and other utility functions.
`<string.h>`	Contains function prototypes for string processing functions.
`<time.h>`	Contains function prototypes and types for manipulating the time and date.

Algunas cabeceras estándar de C

Ejemplo

- Coloque la función cuadrado en un fichero cabecera separado llamado operaciones.h