LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

INGENIERÍA EN INFORMÁTICA

PROGRAMACIÓN

Unidad 4: Descomposición de problemas , diseño modular y funciones. Funciones definidas por el usuario. Variables locales y globales: alcances.

2023

Lic. Mariela A. Velázquez

Repasemos lo visto

¿Cuál es la metodología de trabajo propuesta?

Recordemos…

- Un programa escrito en código C es un conjunto de definiciones de variables y funciones, lideradas por **main**

- *Un programa puede residir en varios archivos fuentes, compilados independientemente y cargarse junto con* ***funciones de biblioteca.***

Funciones

- Una función encapsula cálculos y resguarda la información.
- Si las funciones se diseñaron en forma adecuada, puedo ignorar el COMO lo hace , es suficiente saber QUE HACE….

Tipos de Funciones

Existen dos tipos de funciones:

Funciones de biblioteca
printf
scanf
sqrt
Isalpha
Etc.

Funciones definidas por el usuario
int dividir(int a,int b);
int esletra(char car);

De Módulo a Función

- Los módulos tienen propósitos específicos.
- Constituyen una parte aislada y autónoma del programa.
- Se comunican entre si a través de argumentos y valores regresados por las funciones.
- El argumento es el canal de comunicación entre funciones.

De Módulo a Función

FUNCIÓN esletra (car): carácter → entero

esletra ←0

SI (car >='a' ∧ car<= 'z') ENTONCES

esletra ←1

FIN_SI

Retornar(esletra)

```
int esletra (char letra)
{
    int esletra =0;
    if (97 <= letra &&  letra  <= 122)
        esletra= 1;
    return (esletra );
}
```

La proposición return es el mecanismo para que la función regrese un valor a su invocador.

Sintaxis para definir una función

```
Tipo nombre_func(lista de parámetros) //encabezado
{
        declaraciones; //sector de declarativas de vbles locales
        proposiciones; // cuerpo de la función.
}
```

Tipo asociado a la función. Indica el tipo de retorno de la función (int, float, long,char, etc).

nombre_func debe ser un identificador válido.

(lista de parámetros) listado de los parámetros formales de la función: tipo1 p1, …tipok pk

declaraciones: tipo1 var1; tipo2 var 2;…tipoi var i; // Estas declarativas definen la lista de variables **LOCALES** a la función.

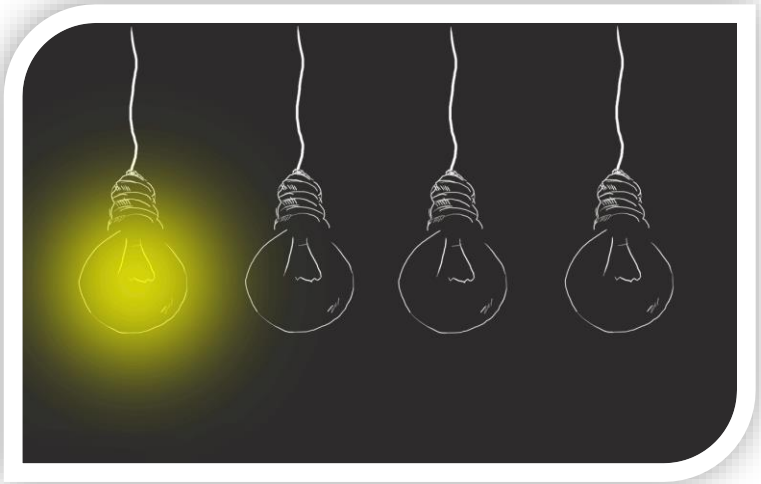
proposiciones: Las declaraciones dentro de la función y las proposiciones constituyen el cuerpo de la función que estará encerrado entre llaves : **{ }**

Del tipo de retorno y del tipo de la función

- La proposición **return** permite que la función "regrese" un valor al medio que la invocó.
- **Forma de uso**: return *expresión;*
- *Los () son optativos.*

A tener en cuenta

- Usar nombres representativos para las funciones.
- El tipo de retorno de la función indica si habrá de devolver valores o no.
- Tipo void , no retorna valores. Una función de este tipo **"produce efectos".**
- Si se omite el tipo de los parámetros, se asume entero.
- La lista de parámetros puede estar vacía, en cuyo caso los paréntesis estarán vacíos.



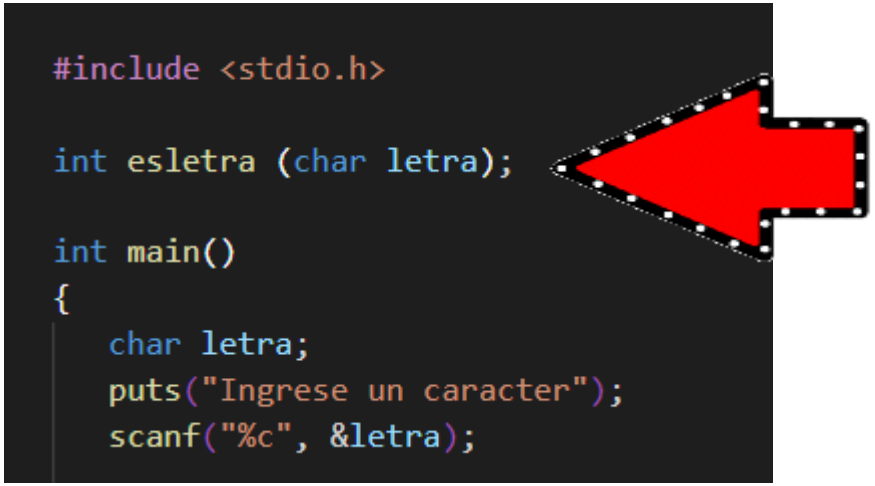
```
void nada  () {};     // no regresa ni hace nada
```

En resumen, las funciones:

- Se declaran
- Se definen
- Se asocian a un tipo de datos
- Se invocan
- Se ejecutan
- Pueden devolver valor /valores
- Pueden ejecutar tareas sin retornar valores
- Pueden llamarse o invocarse desde distintas partes de un programa

Donde deben estar las declaraciones de las funciones definidas por el usuario?

- Al inicio del programa, antes de main
- ¿Donde deben estar las definiciones de las funciones ?



Un tipo particular de funciones

- Son las del tipo void.
- La función no devuelve ningún valor, a cambio, decimos que **"produce un efecto"**.

```
void dibujar_figura(int n);
```

Lo que conocemos en algoritmo como PROCEDIMIENTOS

Prototipo de una función

```c
#include<stdio.h>

int potencia(int base, int exponente);

int main(void)
{
    int base, exponente, resultadoPotencia;

    printf("Ingresar la base: ");
    scanf("%d", &base);

    printf("Ingresar el exponente: ");
    scanf("%d", &exponente);

    resultadoPotencia = potencia(base, exponente);

    printf("El resultado de la potencia es: %d", resultadoPotencia);
    return 0;

}
```

Esta declaración debe coincidir con la definición y uso de la función convertir

Prototipo de una función

- Es un error que la declaración de una función no coincida con su definición.
- Se recomienda hacer una buena selección para los nombres de los parámetros.
- Parámetros Formales y Parámetros actuales : coincidencia en orden, tipo y cantidad.

```c
#include <stdio.h>

int esletra (char letra);

int main()
{
   char letra;
   puts("Ingrese un caracter");
   scanf("%c", &letra);

   if(esletra(letra)==1)
   {
      printf("El caracter ingresado es una letra");
   }
    else{
        printf("El caracter ingresado NO es una letra");
    }
    return 0;
}

int esletra (char letra)
{
     prueba();
      int esletra =0;
      if (97 <= letra &&  letra  <= 122)
          esletra= 1;
      return (esletra );
}
```

declaración

Parámetros Actuales

Parámetros Formales

definición

Variables Locales

Declaradas en el contexto de una función, comienzan a “existir” cuando se invoca a la función y desaparecen cuando la función termina de ejecutarse.

Solo pueden ser usadas por la función que “LAS CONTIENE”.

No retienen sus valores entre dos invocaciones sucesivas a la función.

Variables globales

- Son externas a todas las funciones.
- Pueden ser accedidas por cualquier función y usadas como parámetros actuales para comunicar datos entre funciones.
- Existen en forma “permanente”
- Conservan sus valores entre distintas invocaciones.

Alcance de un identificador

- El alcance de un nombre es la parte del programa dentro de la cual se puede usar ese nombre.
- Variables Locales es lo mismo que variables privadas.
- Variables globales es lo mismo que variables públicas.

La proposición return

- La proposición return tiene 2 usos importantes:
- Devuelve un valor: return(expresión);
- Provoca la salida inmediata de una función: return;

Considere la siguiente definición de una función que calcula el cubo de un entero.

Versión 1

```
int cubo(int num)
{
    int aux;
    aux = num*num*num;
    return(aux);
}
```

Versión 2

```
int cubo(int num)
{
        return(num*num*num);
}
```

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```