LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

INGENIERÍA EN INFORMÁTICA

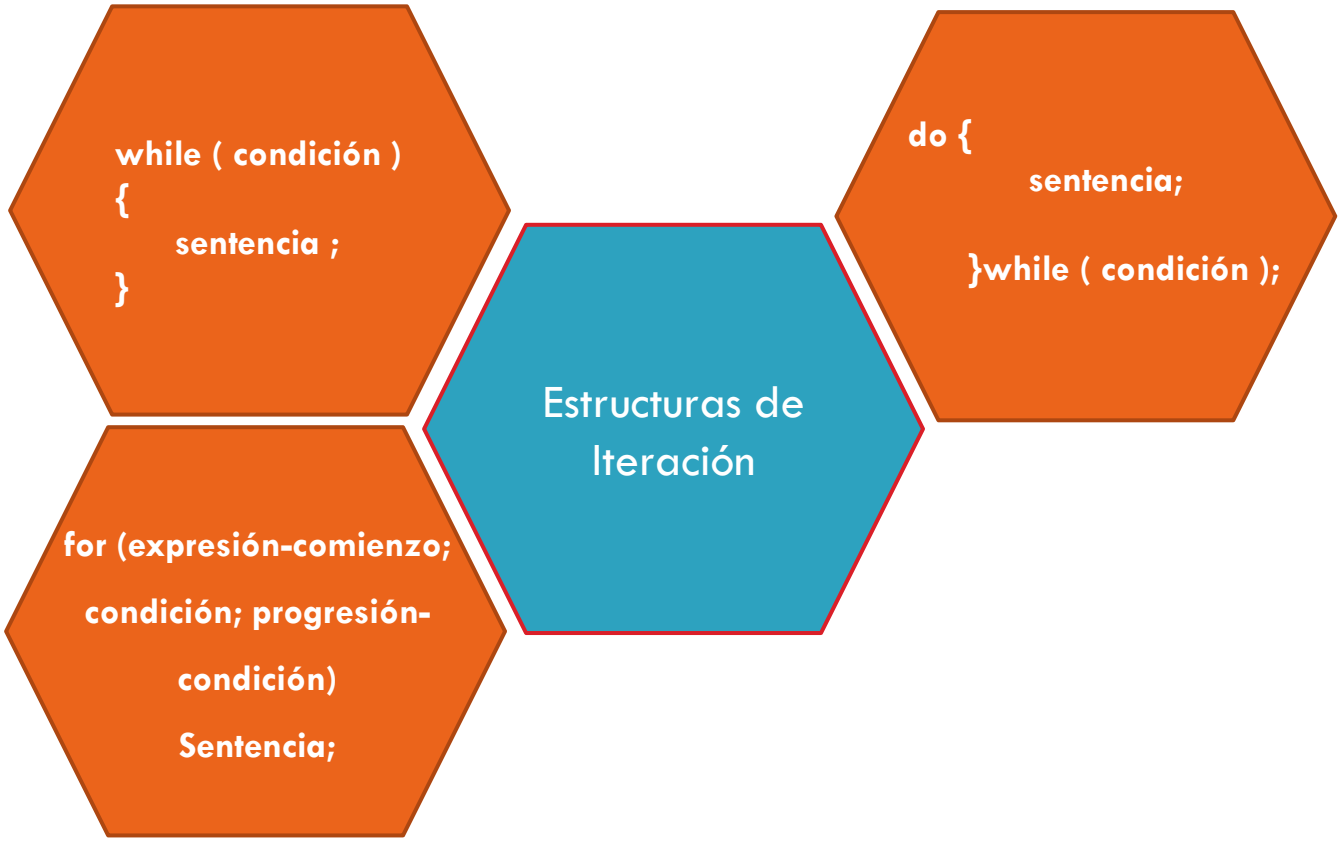
PROGRAMACIÓN

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

2024

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.

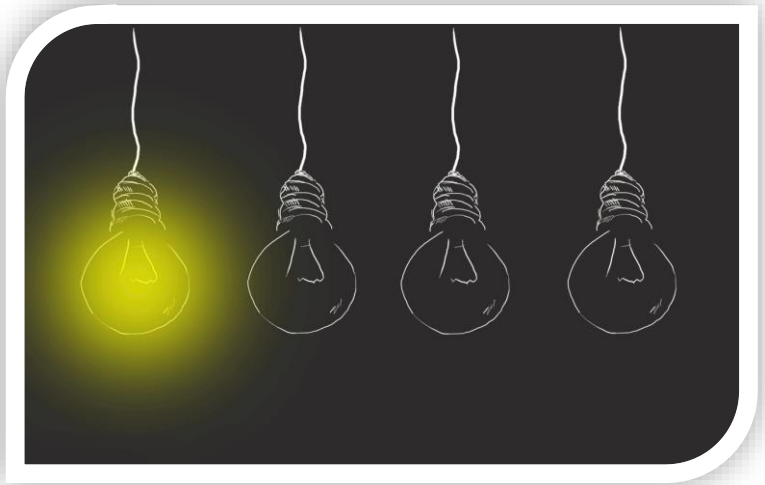
proposiciones: Las declaraciones dentro de la función y las proposiciones constituyen el cuerpo de la función que estará encerrado entre llaves : **{ }**

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.



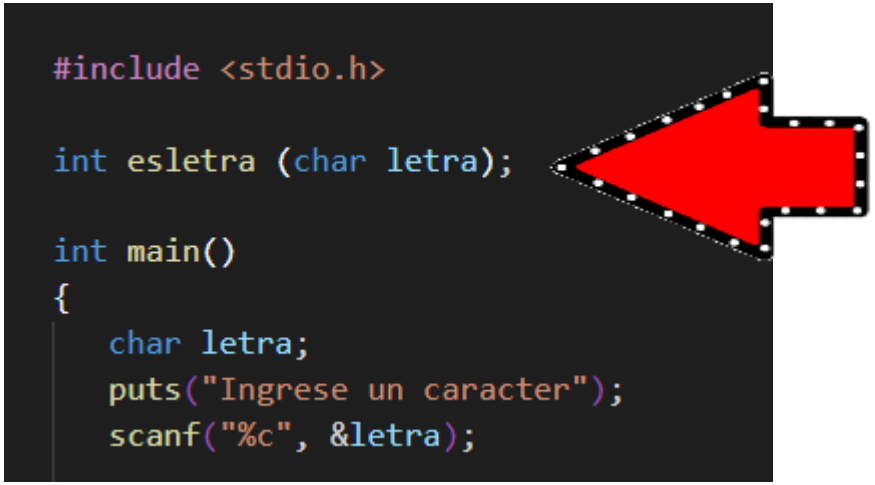
```
void nada  () {};     // no regresa ni hace nada
```

En resumen, las funciones:

- Se declaran
- Se definen
- Se asocian a un tipo de datos
- Se invocan
- Se ejecutan
- Pueden devolver valor
- 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



```c
#include <stdio.h>

int esletra (char letra);

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

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

Invocación

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);
}
```

Funciones de Biblioteca

- C proporciona funciones predefinidas que no forman parte del lenguaje (denominadas funciones de biblioteca). Se invocan por su nombre y los parámetros opcionales que incluye. Después de que la función sea llamada, el código asociado con la función se ejecuta y, a continuación, se retorna el valor obtenido.
- En C, a diferencia de las funciones definidas por el usuario que requieren una declaración o prototipo en el programa, que indica al compilador el nombre por el cual ésta será invocada, el tipo y el número y tipo de sus argumentos, las funciones de biblioteca requieren que se incluya el archivo donde está su declaración.

Ejemplo

Biblioteca donde se encuentra la declaración de las funciones de entrada y salida

```c
#include<stdio.h>

int main(void)
{
    int edad;

    printf("Ingrese su edad: ");

    scanf("%d", &edad);

    printf("La edad ingresada es: %d", edad);
    return 0;
}
```

Función de Salida

Función de Entrada

Ejemplo

En el siguiente ejemplo, de la función seno, de la biblioteca math.h

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

int main(void)
{
    double result, x = 0.5;
    int  i;

    for(i=1;i<=20;i++)
    {
        result=sin(x);
        x = x+0.5;
        printf("El seno() de %lf es %lf\n", x, result);
     }
     return 0;
}
```

Biblioteca donde se encuentra la declaración de las funciones matemáticas

Función seno

Funciones de Biblioteca

Algunas funciones de biblioteca que podemos mencionar son:

<stdio.h>

printf()
scanf()
getchar() car = getchar();
putchar() putchar(car);

<math.h>

sqrt(x): raiz cuadrada de x
pow(x,y): x elevado a la potencia y
sin(x): seno de x expresado en radianes
cos(x): coseno de x expresado en radianes

<ctype.h>

isalpha(c): retorna verdadero si c es una letra mayúscula o minúscula. Retorna falso si c no es letra.
islower(c): retorna verdadero si c es una letra minúscula. Retorna falso si c no es una letra minúscula.
isupper(c): retorna verdadero si c es una letra mayúscula. Retorna falso si c no es una letra mayúscula.

Funciones de Biblioteca

En la tabla, x e y son de tipo double, las funciones regresan double. Los ángulos para las funciones trigonométricas están expresados en radianes.

Función	Descripción	Ejemplo de uso
sqrt	raíz cuadrada de x	sqrt (900.0) es 30.0
exp(x)	función exponencial e^x	exp(1.0) es 2.718282 exp(2.0) es 7.389056
log(x)	logaritmo natural de x (base e)	log(2.718282) es 1.0 log(7.389056) es 2.0
log10(x)	logaritmo de x (base 10)	log10(1.0) es 0.0 log10(100) es 2.0
ceil(x)	redondea a x al entero más pequeño que no sea menor que x	ceil(9.2) es 9.0 ceil(-9.8) es -10.0
floor(x)	redondea a x al entero más grande no mayor que x	floor(9.2) es 9.0 floor(-9.8) es -9.0
pow(x,y)	x elevado a la potencia y	pow(2.7) es 128.0
sin(x)	seno de x expresado en radianes	sin(0.0) es 0.0
cos(x)	coseno de x expresado en radianes	cos(0.0) es 1.0
tan(x)	tangente de x expresado en radianes	tan(0.0) es 0.0

Funciones de Biblioteca

Funciones de uso común de la biblioteca de manejo de caracteres: <ctype.h>

Función	Descripción
int isdigit (int c)	SI (c es un dígito) ENTONCES regresa un valor verdadero SINO regresa un valor falso
int isalpha (int c)	SI (c es una letra) ENTONCES regresa un valor verdadero SINO regresa un valor falso
int isalnum (int c)	SI (c es un dígito o una letra) ENTONCES regresa un valor verdadero SINO regresa un valor falso
int islower (int c)	SI (c es una letra minúscula) ENTONCES regresa un valor verdadero SINO regresa un valor falso
int isupper (int c)	SI (c es una letra mayúscula) ENTONCES regresa un valor verdadero SINO regresa un valor falso
int tolower (int c)	SI (c es una letra mayúscula) ENTONCES regresa c como una letra minúscula SINO regresa c sin cambios
int toupper (int c)	SI (c es una letra minúscula) ENTONCES regresa c como una letra mayúscula SINO regresa c sin cambios

Funciones de Biblioteca

¿Podemos nosotros crear nuestra propia biblioteca con funciones personalizadas?

La respuesta a esa pregunta es **SI**, podemos crear una biblioteca con nuestras funciones (Funciones definidas por el usuario). Explicaremos mediante los pasos a seguir mediante el siguiente ejemplo:

Realice una función que dado dos números enteros devuelva el resultado de su suma.

Funciones de Biblioteca

Opción Nº 1

1. Escribir el prototipo de la función y la definición de la misma.

```c
int suma(int x, int y); // prototipo de la función


int suma(x,y)
{
    return(x+y); //retorno de la función
}
```

2. Guarde el archivo con el nombre: misFunciones.h, en el lugar donde se instalo el compilador. Por ejemplo: **C:\MinGW\include**

3. Ahora usemos nuestra librería…

```c
#include <stdio.h>
#include <misFunciones.h>

int main(void)
{
    int num1 = 5, num2 = 10, resultado;

    resultado = suma(num1, num2);
    printf("El resultado es: %d", resultado);

    return 0;
}
```

Funciones de Biblioteca

Opción Nº 2

1. Escribir el prototipo de la función y la definición de la misma.

```c
int suma(int x, int y); // prototipo de la función


int suma(x,y)
{
    return(x+y); //retorno de la función
}
```

2. Guarde el archivo con el nombre: misFunciones.h, en el mismo lugar donde se encuentra su codigo .c principal

3. En este caso para hacer uso de esa librería debe incluirla usando comillas.

```c
#include <stdio.h>
#include "misFunciones.h"

int main(void)
{
    int num1 = 5, num2 = 10, resultado;

    resultado = suma(num1, num2);
    printf("El resultado es: %d", resultado);

    return 0;
}
```

```c
#include <stdio.h>

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

    return 0;
}
```