LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

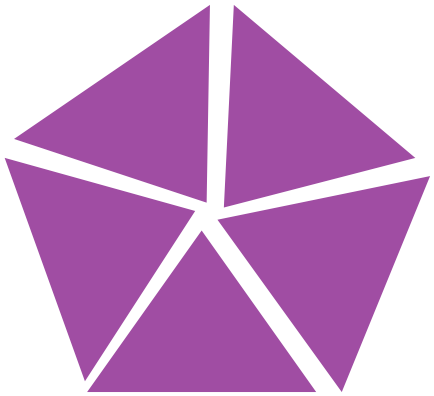
Unidad 4: Funciones. Descomposición de problemas, diseño modular y funciones. Funciones definidas por el usuario. Funciones de biblioteca.

Lic. Mariela A. Velázquez

¿Cuál es la metodología de trabajo propuesta?

Metodología de Trabajo

- Método de diseño top down



- Divide & Conquer

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);

Funciones de biblioteca

- *No forman parte del lenguaje.*
- *Disponible en todas las implementaciones.*

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

ALGORITMO esletra

ENTRADA: car: carácter

SALIDA: es: entero

esletra ←0

SI (a <= car <= z) ENTONCES

esletra ←1

FIN_SI

Retornar esletra

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

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 nombre_func (lista de parámetros)

- **Tipo asociado a la función**. Indica el tipo de retorno de la función.
- **nombre_func** debe ser un identificador válido.
- **(lista de parámetros)** listado de los parámetros formales de la función: tipo1 p1, tipo2 p2,…tipok pk

declaraciones; //sector de declarativas de vbles

- De la forma:

  tipo1 var1; tipo2 var 2;…tipoi var i;

- Estas declarativas definen la lista de variables

  LOCALES a la función.

proposiciones; // cuerpo de la función.

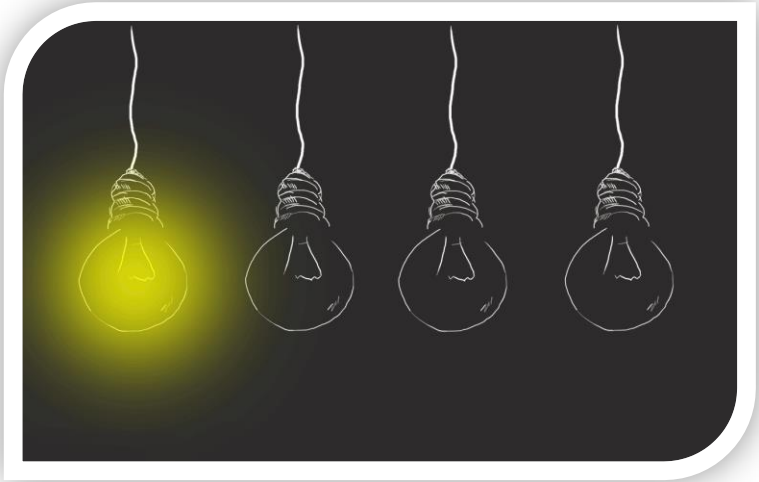
- Conjunto de proposiciones o instrucciones válidas.
- Las declaraciones dentro de la función y las proposiciones constituyen el cuerpo de la función que estará encerrado entre llaves : { }

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 activos 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.



A tener en cuenta

- La lista de parámetros puede estar vacía, en cuyo caso los paréntesis estarán vacíos.
- Se pueden omitir algunas partes en la función, por ejemplo las declarativas y las proposiciones del cuerpo:

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

- C no permite definir una función dentro de otra.

En resumen, las funciones:

- Se declaran
- Se definen
- Se asocian a un tipo de datos
- Se invocan
- Se ejecutan
- Se pueden usar como factores de una expresión
- 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 ?
- En cualquier orden, en esta etapa inicial, se recomienda que estén en el archivo fuente del programa que las invoca.

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);

Ejercicio:

Diseñar e implementar una función que calcule el potencia de un numero entero. Realice los supuestos necesarios.

Prototipo de una función

```
#include<stdio.h>

int potencia(int base, int exponente);

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

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

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

   potencia = potencia(base, exponente);

   return 0;

}
```

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

Prototipo de una función

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

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);
}
```

Para analizar

```
...
printf("Ingrese un entero");
scanf("%d",&n);
printf("El datos ingresado es %d\n", n);
cubo(5);
...
```

¿Dónde se almacena el valor del cubo de 5?

Cubo(5), ¿está invocado de forma correcta ?

Ejemplo:

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

int main(void)

{   double result, x = 0.5; inti;
    for(i=1;i<=20;i++)
    {
       result= sin(x);

       x = x+0.5;
       printf("El seno() de %lfes %lf\n", x, result);
     }

     return 0;

}
```

Programa que escribe la tabla de valores de sen(x)

Esta función en particular no tiene datos de entrada

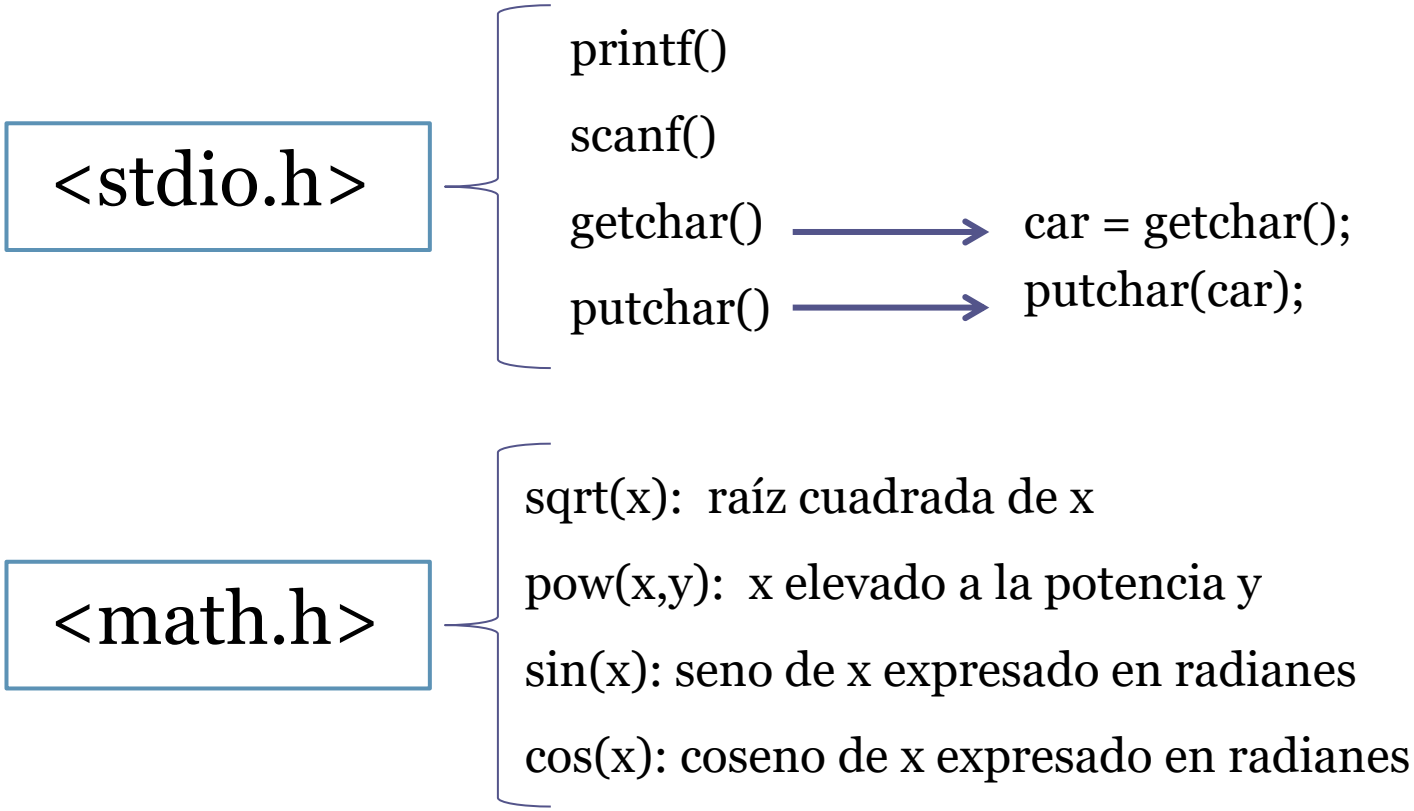
```
#include<stdlib.h>
#include<math.h>

void tabla(void);

int main(void)
{
   tabla();
   return0;
}

void tabla(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;
}
```

Ejemplo:

- Diseñe y escriba un algoritmo que lea una frase y cuente la cantidad de consonantes minúsculas que hay en la misma. Haga todos los supuestos que considere necesarios y escríbalos.

Funciones de biblioteca

- *No forman parte del lenguaje.*
- *Algunas funciones de biblioteca que podemos mencionar son:*



Funciones de biblioteca

<ctype.h>

int isalpha (int c)

SI (c es 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

¿Podemos nosotros crear nuestra propia biblioteca con funciones personalizadas?

Creando nuestra propia biblioteca de Funciones…

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

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

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

Guarde el archivo con el nombre: misFunciones.h

En la siguiente en la carpeta: C:\MinGW\include

Funciones de Biblioteca

Pudo observar que se almacena en la carpeta:

C:\MinGW\include??

Ahora usemos nuestra librería…

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

int main(void) {

        setvbuf(stdout,NULL,_IONBF,0);
        int num1,num2,resultado;
            printf("ingresa un numero: ");
            scanf("%d",&num1);

            printf("\nIngresa otro numero: ");
            scanf("%d",&num2);

            resultado=suma(num1,num2);

            printf("la suma de los dos numeros es: %d \n", resultado);
        return 0;
}
```

```c
#include <stdio.h>

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

    return 0;
}
```