FUNCIONES

La programación modular es una técnica que consiste en dividir un programa en tareas y dar origen a la creación de pequeños programas llamados módulos, subprogramas o subrutinas con la finalidad de simplificar la elaboración y mantenimiento del mismo.

En Lenguaje C a cada módulo o subprograma se le conoce como función –función secundaria-.

Ventaja de la programación modular:

- Facilita el diseño descendente
- Se simplifica un algoritmo complejo
- Cada módulo se puede elaborar de manera independiente
- La depuración se lleva a cabo en cada módulo
- Creación de bibliotecas con módulos específicos

Función

Es un subprograma que realiza una tarea específica que puede o no recibir valores –parámetros-, en lenguaje C se puede devolver valores de tipo puntero, numérico, carácter, un valor nulo y no se pueden regresar arreglos ni estructuras.

Ejemplo de funciones sin paso de parámetros:

Ejem1

```
#include<stdio.h>
void funcionTexto() /*función secundaria*/
{
  printf("\nSaludos!!!\n");
}
int main() /*función principal*/
{
  funcionTexto();
  return 0;
}
```

Ejem2

```
#include<stdio.h>
void funcionTexto();/*prototipo de funcion*/
int main() /*función principal*/
{
 funcionTexto();
 return 0;
}
void funcionTexto() /*función secundaria*/
{
 printf("\nSaludos!!!\n");
}
```

Ejem3

```
#include<stdio.h>
void suma();
void resta();
void multiplicacion();
void division();
float a=0, b=0, c=0; /*variables globales*/
int main() /*función principal*/
{
  int menu, res;
  do{
  printf("\nSeleccione una opción\n");
  printf("\n1 -suma\n");
  printf("\n2 -resta\n");
  printf("\n3 -multiplicación\n");
  printf("\n4 -división\n\n");
```

```
scanf("%d",&menu);
switch(menu){
case 1:
    suma();
     break;
case 2:
    resta();
     break;
case 3:
    multiplicacion();
     break;
case 4:
    division();
     break;
default:
    printf("\nOpcion no valida\n");
    break;
```

```
    }
 printf("\n\nIngrese 1 si quieres ver el menu nuevamente\\nn");
 scanf("%d",&res);
}while(res==1);
 return 0;
}

void suma() /*función secundaria*/
{
   printf("\nIngrese dos valores\n");
   scanf("%f %f",&a ,&b);
   c=a+b;
   printf("\n%f+%f=%f", a, b, c);
}
```

```
void resta() /*función secundaria*/
{
   printf("\nIngrese dos valores\n");
   scanf("%f %f",&a ,&b);
   c=a-b;
   printf("\n%f-%f=%f", a, b, c);
}
void multiplicacion() /*función secundaria*/
{
   printf("\nIngrese dos valores\n");
   scanf("%f %f",&a ,&b);
   c=a*b;
   printf("\n%f*%f=%f", a, b, c);
}
```

```
void division() /*función secundaria*/
{
   printf("\nIngrese dos valores\n");
   scanf("%f %f",&a ,&b);
   if(b!=0){
     c=a/b;
     printf("\n%f/%f=%f", a, b, c);
     }
   else
    printf("\nError!!!\n");
}
```

Funciones con paso de parámetros

Estas funciones son las más utilizadas en la programación ya que pueden recibir uno o más valores llamados parámetros y regresan un solo valor de tipo entero, real o carácter. Sí se desea regresar un arreglo de carácter es necesario hacerlo desde los parámetros. Los parámetros o valores son enviados del programa principal o de otras funciones. Por lo tanto, dentro de la función se realizan solamente las instrucciones. Es importante revisar que el tipo de dato que regresará la

función sea del mismo tipo que el valor declarado en el encabezado de la misma.

Tipos de parámetros:

- Formales o ficticios

  Ejemplo:

  float div(***int x, int y***)

  Siempre los argumentos formales son variables.

- Actuales o reales

  Z=div(***a,b***)

  Los argumentos reales pueden ser variables, constantes o una expresión aritmética.

Ventajas de usar funciones con parámetros:

- Reducen el tamaño del código
- Pueden reutilizarse en otro programa
- Se pueden crear librerías o bibliotecas personales que incluyan funciones desarrolladas por el propio programador

Ejemplo de funciones con paso de parámetros:

Ejem4

```
#include<stdio.h>
int funcionEntera(int a) /*función secundaria*/
{
  int y;
  y=a*a;
  return y;
}
int main() /*función principal*/
{
  int x;
  x= funcionEntera(5);
 printf("\n%d\n",x);
 return 0;}
```

Ejem5

```
#include<stdio.h>
int funcionEntera(int a); /*prototipo de funcion*/
int main() /*función principal*/
{
  int x,z=10;

  x= funcionEntera(z);
 printf("\n%d\n",x);
  return 0;
}
int funcionEntera(int a) /*función secundaria*/
{
  int y;
  y=a*a;
  return y;}
```

Ejem6

```
#include<stdio.h>
float suma(int a,int  b);
float resta(int a,int b);
float multiplicacion(int a,int b);
float division(int a,int b);
float c=0;
int a, b;
int main(){ /*función principal*/
  int menu, res;
  do{
  c=0;
  printf("\nSeleccione una opción\n");
  printf("\n1 -suma\n");
  printf("\n2 -resta\n");
  printf("\n3 -multiplicación\n");
```

```
printf("\n4 -división\n\n");
scanf("%d",&menu);
switch(menu){
case 1:
   printf("\nIngrese dos valores\n");
   scanf("%d %d",&a ,&b);
   c=suma(a,b);
   printf("\n%d+%d=%f", a, b, c);
   break;
case 2:
   printf("\nIngrese dos valores\n");
   scanf("%d %d",&a ,&b);
   c=resta(a, b);
   printf("\n%d-%d=%f", a, b, c);
   break;
case 3:
   printf("\nIngrese dos valores\n");
```

```
    scanf("%d %d",&a ,&b);
    c=multiplicacion(a,b);
    printf("\n%d*%d=%f", a, b, c);
     break;
 case 4:
    printf("\nIngrese dos valores\n");
    scanf("%d %d",&a ,&b);
    if(b!=0){
    c=division(a,b);
    printf("\n%d/%d=%f", a, b, c);
    }
    else
     printf("\nError\n");
    break;
default:
    printf("\nOpcion no valida\n");
    break;
```

```
    }
 printf("\n\nIngrese 1 si quieres ver el menu nuevamente\n\n");
 scanf("%d",&res);
}while(res==1);
 return 0;
}
float suma(int a,int b) /*función secundaria*/
{
   c=a+b;
   return c;
}
float resta(int a,int b) /*función secundaria*/
{
   c=a-b;
   return c;
}
```

```
float multiplicacion(int a,int  b) /*función secundaria*/
{
   c=a*b;
   return c;
}

float division(int a,int b) /*función secundaria*/
{
   c=a/b;
   return c;
}
```

```
Ejem7
#include<stdio.h>
int a, b;
float baseX(int a,int  b);
int main(){ /*función principal*/
        int res;
         do{
                printf("\n\nIngrese la base de la serie\n\n");
                scanf("%d",&a);
                printf("\n\n¿Cuantos elementos quieres ver de la serie?\n\n");
                scanf("%d",&b);
                baseX(a, b);
                printf("\n\nIngrese 1 si quieres ver el menu nuevamente\n\n");
                scanf("%d",&res);
        }while(res==1);
        return 0;
}
```

```
float baseX(int a,int  b) /*función secundaria*/
{
    int x;
    float c;
    for(x=0; x<=(b-1);x++){
        c=pow(a,x);
        printf("\n\n%f\n\n", c);
    }
 }
```

Ejem8

FUNCIONES Y ARREGLOS
OBTENER Y ALMACENAR LA SUMA DE DOS VECTORES USAR FUNCIONES CON PARAMETROS

```
#include<stdio.h>
#include<stdlib.h>
#define p printf
#define s scanf

int A[10],B[10],C[10],i;
int leerArreglo(int vA[10]);
int imprimeArreglo(int vB[10]);

int main(){
	p("\nLeer Arreglo A\n");
	leerArreglo(A);
	p("\nLeer Arreglo B\n");
	leerArreglo(B);
	for(i=0;i<10;i++)
	{
		C[i]=A[i]+B[i];
	}
	p("\nEl arreglo A es\n");
	imprimeArreglo(A);
	p("\nEl arreglo B es\n");
	imprimeArreglo(B);
	p("\nLa suma de los vectoresC es\n");
```

```
    imprimeArreglo(C);
    system("pause>NULL");
return 0;
    }
   int leerArreglo(int vA[10])
    {
       p("\nDame 10 numeros:\n");
       for(i=0;i<10;i++)
       {
           s("%d",&vA[i]);
       }
     }
    int imprimeArreglo(int vB[10])
    {
        for(i=0;i<10;i++)
        {
            p("%d\t",vB[i]);
        }

    }
```