

# HERRAMIENTAS DE INFORMÁTICA

## Lenguaje C



**El AMOR es sufrido, es benigno; el AMOR no tiene envidia,  
el AMOR no es jactancioso, no se envanece;  
no hace nada indebido, no busca lo suyo,  
no se irrita, no guarda rencor;  
no se goza de la injusticia, más se goza de la verdad.  
Todo lo sufre todo lo cree, todo lo espera y todo lo soporta.  
El AMOR nunca deja de SER.  
1 Corintios 13:4**

**@Maybel Gil**

# SUBPROGRAMAS

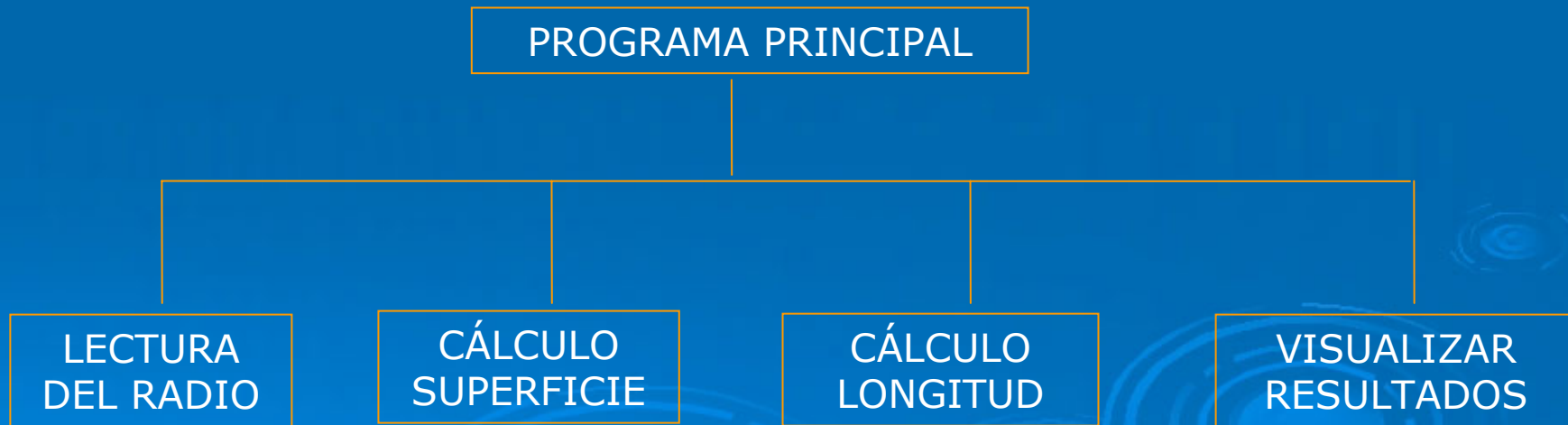
Φ La programación modular consiste en dividir un programa en subprogramas llamados subrutinas o módulos, evitando la creación de enormes programas que resulten imposible de manejar.

Φ Cuando un grupo de pasos se repite en varios lugares del programa, es útil convertir ese conjunto de instrucciones en un subprograma, para evitar una inútil repetición de pasos.

Pueden de ser de dos tipos:

- Φ Procedimientos y
- Φ Funciones.

# DESCOMPOSICIÓN MODULAR



# SUBPROGRAMAS

- La división de un programa extenso en varios subprogramas agiliza el trabajo de programación y facilita la búsqueda de errores al momento de depurar el programa. Los subprogramas declarados en un programa pueden ser llamados desde cualquier parte de un programa las veces que sea necesario.
  
- Cuando se realiza la llamada a un subprograma, el control del programa se traslada hasta el subprograma, lo ejecuta y regresa a la siguiente instrucción desde el lugar donde fue llamado.

# FLUJO DE CONTROL EN LA LLAMADA DE UN SUBPROGRAMA

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

```
...
```

```
int main(){
```

```
    sentencia 1;
```

```
    sentencia 2;
```

```
    Llamada Sub Programa
```

```
    sentencia 4;
```

```
    return 0;
```

```
}
```

```
Subprograma<nombre>{
```

```
...
```

```
    sentencia 1;
```

```
        sentencia 2;
```

```
        sentencia 3;
```

```
        return promedio;
```

```
}
```

# SUBPROGRAMA

Calcular el área de un rectángulo:

Sub-problemas:

- ❖ Entrada de datos de altura y base
- ❖ Cálculo del área
- ❖ Salida de Resultados

Módulos

- ❖ LeerDatos(altura, base)
- ❖ Area = CalcularArea(altura,base)
- ❖ Escribir(area)

# FUNCIÓN

Matemáticamente una función es una operación que toma uno o más valores llamados argumentos y produce un valor llamado argumento

Ejemplo :

$F(x) = 4 / (x^2 + 1)$  Función de un sólo argumento

$$F(1) = 4 / (1 + 1) = 4/2 = 2$$

$F(a,b) = a^2 + b^2$  Función de dos argumentos

# FUNCIÓN

En lenguaje C y C++ **no** existe diferencia entre funciones y procedimientos: a todas las subrutinas se les llama **funciones**.

Una función se declara una vez pero puede usarse (mediante llamadas) tantas veces como sea necesario.

## TIPOS DE FUNCIONES EN C, C++

⊕ **Funciones de Biblioteca:** Los lenguajes C y C++ tiene sus propias funciones incorporadas que permiten realizar ciertas operaciones o cálculos de uso común.

⊕ **Funciones definidas** (diseñadas o codificadas) por el programador para realizar determinadas tareas.

# BIBLIOTECA ESTÁNDAR DE C,C++

- ⊕ Contiene una amplia colección de funciones para llevar a cabo cálculos matemáticos comunes:
  - ❖ Manipulación de cadenas de caracteres.
  - ❖ Manipulación con caracteres
  - ❖ Operaciones estándares de entrada y salida
  - ❖ Operaciones matemáticas

⊕ Una biblioteca de funciones comunes es construida una vez en un archivo de biblioteca que se proporciona como parte del compilador de C/C++.

**stdlib.h, math.h, time.h, ctype.h, string.h,  
stdio.h, malloc.h, conio.h, iostream.h**

**@Maybel Gil**

# FUNCIONES DEFINIDAS POR EL USUARIO

## Φ Sintaxis

```
tipo_resultado nombre (arg1, arg2,...,argn)
{
...conjunto de sentencias...
  return expresión;
}
```

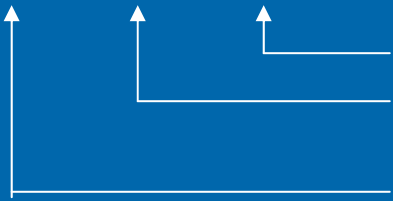
```
int suma( int a, int b){
    return a+b
}
```

# FUNCIONES DEFINIDAS POR EL USUARIO

## ❖ Encabezado - Prototipo

tipo\_resultado nombre (lista\_parámetros\_formales);

void main()



No hay parámetros formales

Nombre de la función

tipo\_resultado ( no tiene)

## ❖ Cuerpo de la función

- ❖ Declaraciones de variables locales
- ❖ Código ejecutable o conjunto de sentencias.

## ❖ Sentencia RETURN

- ❖ Sentencia que devuelve el valor obtenido como resultado de la ejecución de una función

# LLAMADA DE UNA FUNCIÓN

⊕ Si la función es de un tipo determinado  
<var> = <nombre\_función>(<lista de parámetros formales>)

**dias = maxDias(mes, agno)**

⊕ Si la función no tiene tipo asociado  
<nombre\_función>(<lista de parámetros formales>)

**Estrellas(5);**

⊕ Dentro de una expresión  
<var> = E(x + <nombre\_función>(<lista de parámetros formales>))

**Y = mayor(a + factorial(b), b, c);1**

# PROCEDIMIENTOS

❶ En lenguaje C/C++ no se distingue entre procedimiento y funciones. Un procedimiento sería una función que no devuelve ningún valor (void). A esta función se le llama **funciones void**.

❖ En estas funciones no es necesario el return. No obstante si se desea salir prematuramente se puede usar el return.

**Si la función devuelve más de un valor los parámetros se denominan parámetros por referencia (precedidos por **&**).**

# FUNCIÓN VOID Estrellas SIN ARGUMENTOS O PARÁMETROS

```
#include <stdio.h>
#include <stdlib.h>
//prototipos
void Estrellas();
int main(){
    Estrellas();           //llamada a la Función.
    puts("Mensaje");
    Estrellas();           // llamada a la Función
    system("PAUSE");
    return 0;
}
void Estrellas() //declaración de la función
//Visualizar 9 asteriscos
{
    puts("*****");
}
```

# VARIABLES

## ❖ Locales

Son variables que están declaradas dentro de un subprograma y se dice que es local al subprograma. Una variable local sólo está disponible durante el procedimiento del subprograma. Su valor se pierde una vez que el subprograma se ejecuta.

## ❖ Globales

Son variables que están declaradas en el programa principal.

# PARÁMETROS

Los parámetros son identificadores que sirven de **enlace** entre el cuerpo principal y los módulos que integran el programa, permitiendo el traspaso de información entre ambos.

Tipos:

## ❖ **Parámetros Actuales**

intercambio (a,b);

## ❖ **Parámetros Formales**

Valor : void Estrellas (unsigned int n);

Referencia : void Intercambio (int &x,int &y);

# PARÁMETROS ACTUALES

Son los que se encuentran en los **llamados** de los subprogramas en el cuerpo principal del programa.

Estos parámetros son “variables globales” al programa y el uso de ellos es de entregar y/o recibir información.

# PARÁMETROS FORMALES O FICTICIOS

Se encuentran en la cabecera de la declaración de los subprogramas. Ellos sirven para contener los valores de los parámetros actuales cuando se invoca al subprograma.

# PARÁMETROS FORMALES TIPO VALOR

Cuando se pasan parámetros por valor en memoria sucede lo siguiente: se obtiene una **copia** temporal de la variable usada y dentro de la función o procedimiento se trabaja con la copia obtenida. De esta forma la variable introducida como parámetro, *no será afectada* en su valor inicial al terminar el proceso, sin importar las diferentes operaciones que se realicen con dicha copia.

```
void geometria( float longitud, float anchura,  
               float &area, float &perimetro);
```



# PARÁMETROS FORMALES TIPO REFERENCIA

Cuando se pasan parámetros por Referencia del programa principal a un procedimiento, los cambios que se realicen a una variable introducida como parámetro, *modifican el contenido* de dicha variable. En este caso específico los parámetros deben estar precedidos por el carácter &, lo cual indica al compilador que los valores enviados como parámetro pueden **"CAMBIAR"**.

```
Procedure Geometria(Longitud, Anchura: real;  
    Var Area, Perimetro: real);
```

# CORRESPONDENCIA DE PARÁMETROS

Los parámetros actuales en la invocación del procedimiento deben coincidir en:

- ❖ Número de parámetros
- ❖ Orden de los parámetros
- ❖ Tipo de datos con los parámetros ficticios de la declaración de la función -

Nota: Los nombres de los parámetros actuales y formales **pueden** ser los mismos, aunque se recomienda **sean diferentes** a efectos de legibilidad del programa

# EJERCICIO 1

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void geometria(float Longitud,float Anchura,
               float &Area, float &Perimetro);

int main(){
float    x,y,a,p;

    puts("Ingresar Longitud: ");
    scanf("%f",&x);
    puts("Ingresar Anchura: ");
    scanf("%f",&y);
    geometria(x,y,a,p);
    printf("El Area es      : %.2f\n",a);
    printf("El Perimetro es : %.2f\n",p);
    system("PAUSE");
return 0;
}

void geometria(float Longitud,float Anchura,
               float &Area, float &Perimetro){
    Area    = Longitud * Anchura;
    Perimetro = 2 * (Longitud + Anchura);
}
```

# CORRESPONDENCIA DE PARÁMETROS

```
float x,y,a,p;
```

```
void geometria(float Longitud,float Anchura,  
              float &Area, float &Perimetro);
```

```
int main {...  
    geometria(7,4,a,p);
```

```
...  
}
```

28

22

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int signo(int n); //prototipos
int main(){
    int Num;
```

```
    puts("Ingrese un número:");
    scanf("%d",&Num);
    printf("El signo es %d \n",signo(Num));
    switch (signo(Num)){
        case 1 : puts("El número es positivo");
                break;
        case -1 : puts("El número es negativo");
                break;
        case 0 : puts("El número es cero");
                break;
    }
    system("PAUSE");
    return 0;
}
```

## EJERCICIO 2

```
int signo(int x){ //declaración de la función
    //variables locales
    int aux;
    if (x>0) aux =1;
    else if (x<0) aux=-1;
    else aux=0;
    return aux;
}
```

@Maybel Gil

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int maxDias(unsigned int n, int a); //prototipos
int main(){
    int mes,agno;
    puts("Ingrese un mes [1-12]:");
    scanf("%d",&mes);
    puts("Ingrese el ano:");
    scanf("%d",&agno);
    printf("La cantidad de días es
%d\n",maxDias(mes,agno));
    system("PAUSE");
    return 0;
}
int maxDias(unsigned int m, int a){
    int cant;
    switch (m) {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12: cant= 31;
                break;

```

## EJERCICIO 3

```

        case 4:
        case 6:
        case 9:
        case 11: cant= 30;
                break;
        case 2 : if ((a%4 ==0) && ((a%100 !=
0) || (a %400==0)))          cant=29;
                else cant=28;
                break;
    }    return cant;
}

```