



GUIA TEORICA N° 2 HERRAMIENTAS DE INFORMÁTICA I APUNTES: SUBPROGRAMAS.

De cierto te bendeciré con ABUNDANCIA
y te multiplicaré grandemente
y habiendo esperado con PACIENCIA,
alcanzó la promesa. Hebreos 6:13-15

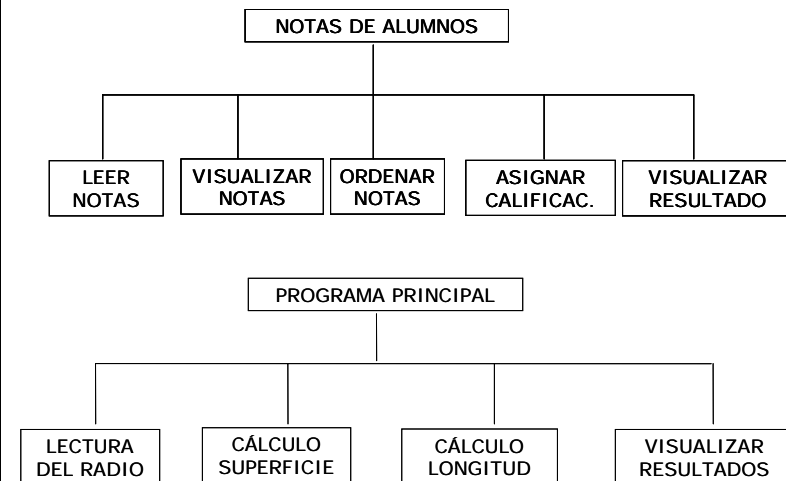
Profesor : DELY M. GIL A.

VALENCIA, 2008

SUBPROGRAMAS

- La programación modular consiste en dividir un programa en subprogramas llamados subrutinas o módulos, evitando la creación de enormes programas que resulten imposibles de manejar.
- 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 ser de dos tipos:
 - Procedimientos y
 - Funciones.

DESCOMPOSICIÓN MODULAR



SUBPROGRAMAS

- Calcular el área de un rectángulo:

Sub-problemas

Entrada de datos de la altura y base
Calculo del área
Salida de resultados

Módulos

LeerDatos(altura,base)
area = CalcularArea(altura,base)
EscribirArea(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

VENTAJAS DE LA PROGRAMACIÓN MODULAR

- Permite la fácil comprensión del programa completo
- Permite la localización rápida de errores.
- Se puede modificar un módulo sin afectar a los demás.
- Permite la reutilización de módulos, evitando la duplicación innecesaria de módulos.
- Facilita la escritura y depuración de un programa ya que cada módulo representa una parte bien definida del problema.
- Favorece la portabilidad ya que se pueden escribir programas sin prestar atención a las características de un sistema en particular.

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++

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

- Φ Funciones definidas (diseñadas o codificadas) por el programador para realizar determinadas tareas.

BIBLIOTECA ESTÁNDAR DE C, C++

- Φ 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

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

BIBLIOTECA ESTÁNDAR DE C, C++**Investigue algunas funciones de las siguientes bibliotecas:**

- Φ stdlib.h:

- Φ math.h:

- Φ time.h:

- Φ ctype.h:

- Φ string.h:

- Φ stdio.h:

- Φ malloc.h:

- Φ conio.h

- Φ iostream.h

FUNCIONES DEFINIDAS POR EL USUARIO * 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 * CUERPO DE UNA FUNCIÓN

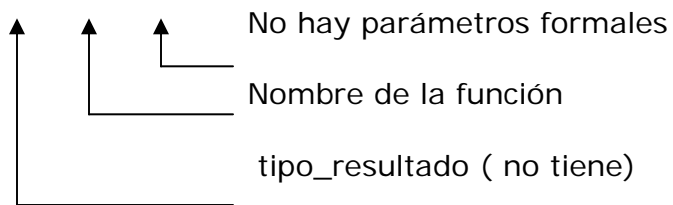
- ⊕ Declaraciones de variables locales
- ⊕ Código ejecutable o conjunto de sentencias.

*** SENTENCIA RETURN**

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

FUNCIONES DEFINIDAS POR EL USUARIO *ENCABEZADO - PROTOTIPOS

```
tipo_resultado nombre (lista_parámetros_formales);
void main()
```



LLAMADA DE UNA FUNCIÓN

- ⊕ Si la función es de un tipo determinado
<var> = <nombre_función>(<lista de parámetros formales>)
- ⊕ Si la función no tiene tipo asociado
<nombre_función>(<lista de parámetros formales>)
- ⊕ Dentro de una expresión
<var> = E(x + <nombre_función>(<lista de parámetros formales>))
Y = mayor(a + factorial(b), b,c);

PROCEDIMIENTOS

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("*****");
}
```

FUNCIÓN VOID Estrellas CON ARGUMENTO O PARÁMETRO FORMAL

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

//prototipos
void Estrellas(unsigned int n);

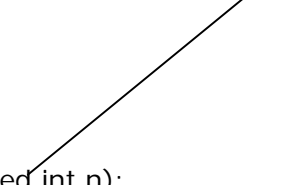
int main(){

    Estrellas(4);          //llamada al Proced.
    puts("Mensaje");
    Estrellas(7);         //llamada al Proced.
    system("PAUSE");
    return 0;
}

void Estrellas(unsigned int n) //declaración de la función

//Visualizar 9 asteriscos
{
    //variables locales
    unsigned int i;

    for (i=0; i<n; i++)
        putchar('*');
}
```



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);

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

//variables globales
int x,y;

//prototipos
void intercambio(int &a,int &b);

int main(){
    //locales del main
    system("CLS");
    puts("Ingrese el valor de X:\n");
    scanf("%d",&x);
    puts("Ingrese el valor de Y:\n");
    scanf("%d",&y);
    intercambio(x,y);
    printf("El valor actual de X es %d\n",x);
    printf("El valor actual de Y es %d\n",y);
    system("PAUSE");
    return 0;
}

//implementación de la función
void intercambio(int &a,int &b){

    //variables locales
    int temp;

    temp =a;
    a   =b;
    b   =temp;
}
```

Parámetros formales

Parámetros actuales

PARÁMETROS ACTUALES

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

Estos parámetros son "variables locales" al programa y el uso de ellos es entregar y/o recibir información.

PARÁMETROS FORMALES O FICTICIOS

Se encuentran en la **cabecera de la declaración** los subprogramas. Ellos sirven para contener los valores de los parámetros actuales cuando se invoca el 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 se trabaja con la copia obtenida. De esta forma la variable introducida como parámetro, no será afectada en su valor inicial la 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, 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 parámetros pueden CAMBIAR.

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

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

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
//variables globalesç
float x,y,a,p;
void geometria(float Longitud,float Anchura,
               float &Area, float &Perimetro);

int main(){

    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){

// Procedimiento que calcula el área y perímetro

    Area      = Longitud * Anchura;
    Perimetro = 2 * (Longitud + Anchura);
}

```

Parámetros formales

Parámetros actuales

```

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

//prototipos
int suma(int n);

int main(){

    int Num;

    puts("Cantidad de números a sumar:");
    scanf("%d",&Num);
    printf("La suma de la serie es %d
\n",suma(Num));
    system("PAUSE");
    return 0;
}

int suma(int x){ //declaracion de la función

//variables locales
int total=0,i;

    for (i=1;i<=x;i++)
        total += i;
    return total;
}

```

```

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

//prototipos
int signo(int n);

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;
}

int signo(int x){ //declaracion de la función

    //variables locales
    int aux;
    if (x>0) aux = 1;
    else if (x<0) aux=-1;
    else aux=0;
    return aux;
}

```

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
//prototipos
int maxDias(unsigned int n, int a);
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){ //declaracion de la
función
int cant;
    switch (m) {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12: cant= 31;
                break;

        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;
}

```

EJERCICIOS

1. Determinar y visualizar el número más grande de dos números dados, mediante un subprograma.
2. Escribir un programa que utilice un procedimiento para convertir coordenadas polares (ángulo: θ y módulo: r) a rectangulares (x e y) : $x=r\cos\theta$, $y=r\sin\theta$ (Los parámetros de entrada son: r = módulo; θ = theta = ángulo)
3. Escribir una función lógica DIGITO que determine si un carácter es uno de los dígitos del '0' al '9'
4. Escribir una función lógica VOCAL que determine si un carácter es una vocal.
5. Emular una calculadora usando procedimientos y/o funciones para cada operación (Suma, Resta, Multiplicación y División). Validar las entradas (Num1 y Num2) y el operador (+, -, *, /). Se debe desplegar un Menú con las opciones de las operaciones.
6. Escriba un programa para calcular la suma: $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{\text{Terms}}$, donde Terms es el número de términos y está especificado por el usuario. Esta suma se conoce como serie armónica.
7. Calcular la suma de los términos de la siguiente serie: $\frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \dots + \frac{n}{2^n}$
8. Implementar una función EXPONENTE que permita hallar el valor de X^Y , siendo X un número real e Y un entero.
9. Escribir una función que lea x y n y calcule la suma de la progresión geométrica : $1 + x + x^2 + x^3 + \dots + x^n$
Realizar dicho subprograma usando la función anterior (debe modificar el tipo de Y) .
10. Escribir una función que dado tres valores A,B y C que representan longitudes en centímetros (cm.) , devuelva los siguientes valores:
 - a) 0 si esas tres longitudes forman un triángulo equilátero (Tres lados iguales).
 - b) 1 si esas tres longitudes forman un triángulo isósceles (Dos lados iguales).
 - c) -1 si esas tres longitudes forman un triángulo escaleno (Tres lados diferentes).
11. Realizar un procedimiento para hallar las raíces de una ecuación de segundo grado del tipo: $Ax^2 + Bx + C = 0$
12. Realizar una función lógica que permita saber si una fecha es válida.
13. Diseñe una función FACTO que permita obtener el factorial de un número entero positivo. (Ejm. $5! = 5 \times 4 \times 3 \times 2 \times 1$)
14. Calcular el coeficiente del binomio, con la función factorial anteriormente codificada.

$$\binom{m}{n} = \frac{m!}{n!(m-n)!}$$
 Donde $m! = 1$ si $m = 0$ (Ejm. : $0! = 1$)
 $1 \times 2 \times 3 \times \dots \times m$ si $m < > 0$ (Ejm. : $4! = 4 \times 3 \times 2 \times 1$; $9! = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$)

$$\binom{7}{3} = \frac{7!}{3! \times (7-3)!} = \frac{7!}{3! \times 4!}$$
15. El valor de e^x se puede aproximar por la suma de $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$
Escribir un programa que tome un valor de x y n como entrada y visualice la suma del valor de e^x . Realizar dicha suma utilizando los subprogramas x^n (EXPONENTE) y $n!$ (FACTO)