



PROGRAMACIÓN

Notas de Práctica - TP N°5

P.U. PABLO AGÜERO - L.I. GABRIELA BARRAZA - P.U. JOSEFINA LOBO



REPASO TEÓRICO

REPASO TEORICO

i.

Funciones

ii.

Declaración
Definición
Invocación

iii.

Funciones
void

iv.

Ámbito de las
variables

i. Funciones

Encapsulan cálculos, constituyen una parte aislada y autónoma del programa.

Si su diseño es correcto, al invocarlas, importa **QUE HACE** y no **COMO**.



Tipos de Funciones



Funciones de biblioteca

```
printf() }  
scanf() } stdio.h
```

```
sqrt(x) }  
pow(x,y) } math.h
```

```
isalpha(x) }  
islower() } ctype.h
```

Funciones definidas por el Usuario

```
int sumar(int a, int b)
```

```
int descuento(int total, int promo)
```

ii. Declaración

La declaración debe coincidir con la definición

Las funciones se declaran antes de main, indicando:

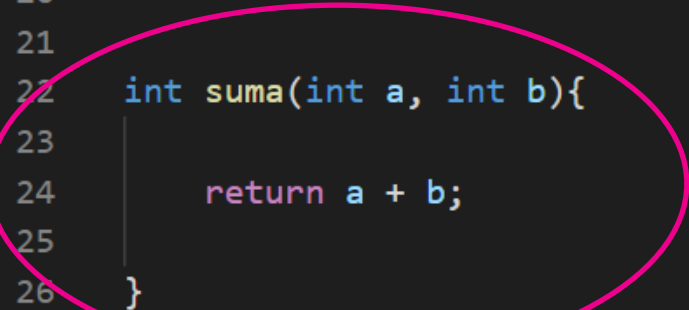
tipo Nombre (parámetros)

```
1  #include <stdio.h>
2
3  int suma(int a, int b);
4
5  int main(void){
6      int num1, num2, total;
7
8      printf("Ingrese un número entero: ");
9      scanf("%d",&num1);
10
11     printf("Ingrese otro número entero: ");
12     scanf("%d",&num2);
13
14     total = suma(num1, num2);
15
16     printf("La suma de los números es: %d", total);
17     return 0;
18 }
```

ii. Definición

Se pueden definir en cualquier parte del programa, incluso en otros archivos, por ahora lo haremos a continuación de main. En la definición se especifica lo que hace la función.

```
15  
16     printf("La suma entre %d y %d es i  
17  
18     return 0;  
19 }  
20  
21  
22 int suma(int a, int b){  
23     |  
24     return a + b;  
25     |  
26 }
```



Pasaje de
parámetros
por valor

ii. Invocación

Se puede invocar a una función en cualquier parte del programa principal y dentro de otra función.

Se debe pasar por parámetros los valores que recibirá la función en el orden que están definidos.

```
1  #include <stdio.h>
2  int suma(int num1, int num2);
3
4  int aux;
5
6  int main(){
7      int a=1, b=3, s;
8      s = suma(a,b);
9      printf("%d",s);
10 }
11
12 int suma(int num1, int num2){
13     return(num1+num2);
14 }
```



Del papel al código

Algoritmos

```
FUNCION suma(a,b): entero, entero -> entero  
    Retornar(a + b)  
FIN
```

Lenguaje C

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



iii. Funciones void

Son un tipo especial de funciones que **no devuelven ningún valor**, sino que producen algún efecto.

```
28
29
30 void imprimirDatos(int dni, int prom){
31     |
32     printf("DNI: %d \n",dni);
33     printf("Promedio: %c \n",prom);
34     |
35 }
```

Algoritmos

```
PROCEDIMIENTO imprimirDatos(dni,prom): entero, entero
    ESCRIBIR(dni)
    ESCRIBIR(prom)
FIN
```



iv. Ámbito de las variables

Variables locales son aquellas que se declaran dentro de una función. Solo **existen mientras dure la ejecución** de la función cuando son invocadas.

Solo pueden ser usadas por la función.

Cuando se termina de ejecutar la función, se pierde su valor.

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

iv. Ámbito de las variables

Variables
públicas

Variables globales se declaran fuera de main. Existen mientras dure la ejecución del programa. Pueden ser accedidas desde cualquier parte del programa. Se mantiene su valor durante toda la ejecución.

```
#include <stdio.h>
void incrementa();
```

```
int aux;
```

```
int main(){
    aux=1;
    incrementa();
    printf("%d",aux);
    return 0;
}
```

// Muestra valor 2

```
void incrementa(){
    aux++;
}
```



05

TRABAJO PRÁCTICO