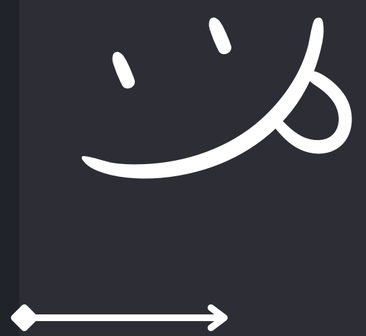
```c
#include <stdio.h>
#include <string.h>

// Programación I - Funciones.

int Funciones()
{
    char prof[20];
    strcpy(prof, "Miguel Silva C.");
    printf("Prof. %s", prof);


    return 0;

}
```

UTN INSPT

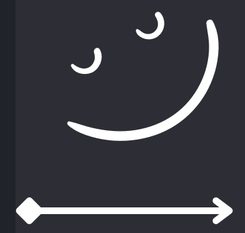


// Programación I - Funciones.

Concepto Funciones() {

Son bloques de código que realizan una tarea específica y están diseñados para ser reutilizados. Una función toma cero o más argumentos como entrada, ejecuta un conjunto de instrucciones y, opcionalmente, devuelve un valor. Las funciones permiten modularizar y organizar el código, haciéndolo más legible, mantenible y reutilizable.

}



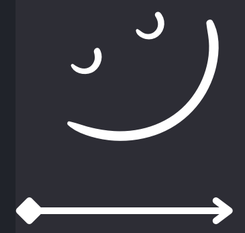
// Programación I - Funciones.

Componentes de una función...

1. DECLARACIÓN DE LA FUNCIÓN (PROTOTIPO):

Especifica el nombre de la función, el tipo de valor que devuelve, y los tipos y números de argumentos que toma. Esto se suele hacer antes de la definición de la función y permite al compilador conocer la existencia de la función antes de su uso.

```c
#include<stdio.h>

#define p printf
#define s scanf

// Prototipos:

int suma(int, int);
```



// Programación I - Funciones.

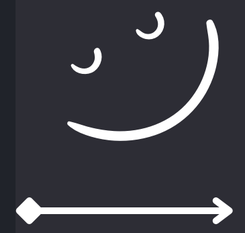
Componentes de una función...

2. DEFINICIÓN DE LA FUNCIÓN:

Contiene el código que especifica qué hace la función. Incluye el tipo de retorno, el nombre de la función, los parámetros y el cuerpo de la función.

```
int suma(int num1, int num2)
{
    int resultado;
    resultado = num1+num2;
    return resultado;
}
```

```
int suma(int num1, int num2)
{
    return num1+num2;
}
```



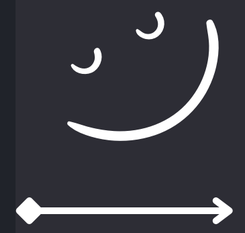
```
// Programación I - Funciones.
```

Componentes de una función...

3. LLAMADA A LA FUNCIÓN:

Es el punto en el código donde se utiliza la función. Se proporciona la lista de argumentos que la función espera.

```
int num1, num2, resultado;

p("Ingrese primer n%cmero: ", 163);
s("%d", &num1);
p("\nIngrese segundo n%cmero: ", 163);
s("%d", &num2);

resultado = suma(num1, num2);
```



// Programación I - Funciones.

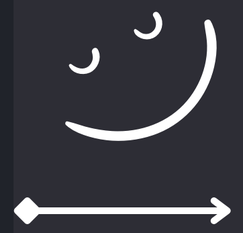
VENTAJAS DE USAR FUNCIONES

1. REUTILIZACIÓN DEL CÓDIGO:
Las funciones permiten reutilizar el mismo bloque de código en diferentes partes del programa sin tener que escribirlo varias veces.

2. MODULARIDAD:
Dividir un programa en funciones facilita la organización y el mantenimiento del código.

3. ABSTRACCIÓN:
Las funciones permiten ocultar los detalles de implementación y exponer solo la interfaz necesaria para su uso.

4. FACILIDAD DE DEPURACIÓN:
Al dividir el código en funciones, es más fácil localizar y corregir errores.



```
// Programación I - Funciones.
```

TIPOS DE FUNCIONES

1. FUNCIONES SIN RETORNO Y SIN PARÁMETROS:

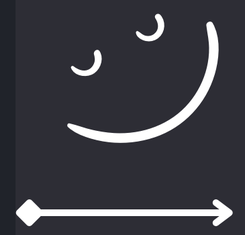
```c
void saludar(void)
{
    printf("**************************\n");
    printf("** SUPER CALCULADORA **\n");
    printf("**************************\n\n");
}
```

```
// Programación I - Funciones.
```

TIPOS DE FUNCIONES

2. FUNCIONES SIN RETORNO PERO CON PARÁMETROS:

```
void imprimirSuma(int num1, int num2, int resultado)
{
    p("\n%d + %d = %d\n", num1, num2, resultado);
}
```

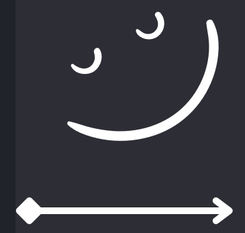


// Programación I - Funciones.

TIPOS DE FUNCIONES

3. FUNCIONES CON RETORNO Y CON PARÁMETROS:

```
int suma(int num1, int num2)
{
    int resultado;
    resultado = num1+num2;
    return resultado;
}
```



// Programación I - Funciones.

TIPOS DE FUNCIONES

4. FUNCIONES CON RETORNO PERO SIN PARÁMETROS:

```
int opcion(void)
{
    int op;
    p(" - Ingrese una opci%cn: ", 162);
    s("%d", &op);
    return op;
}
```

Ejemplo:

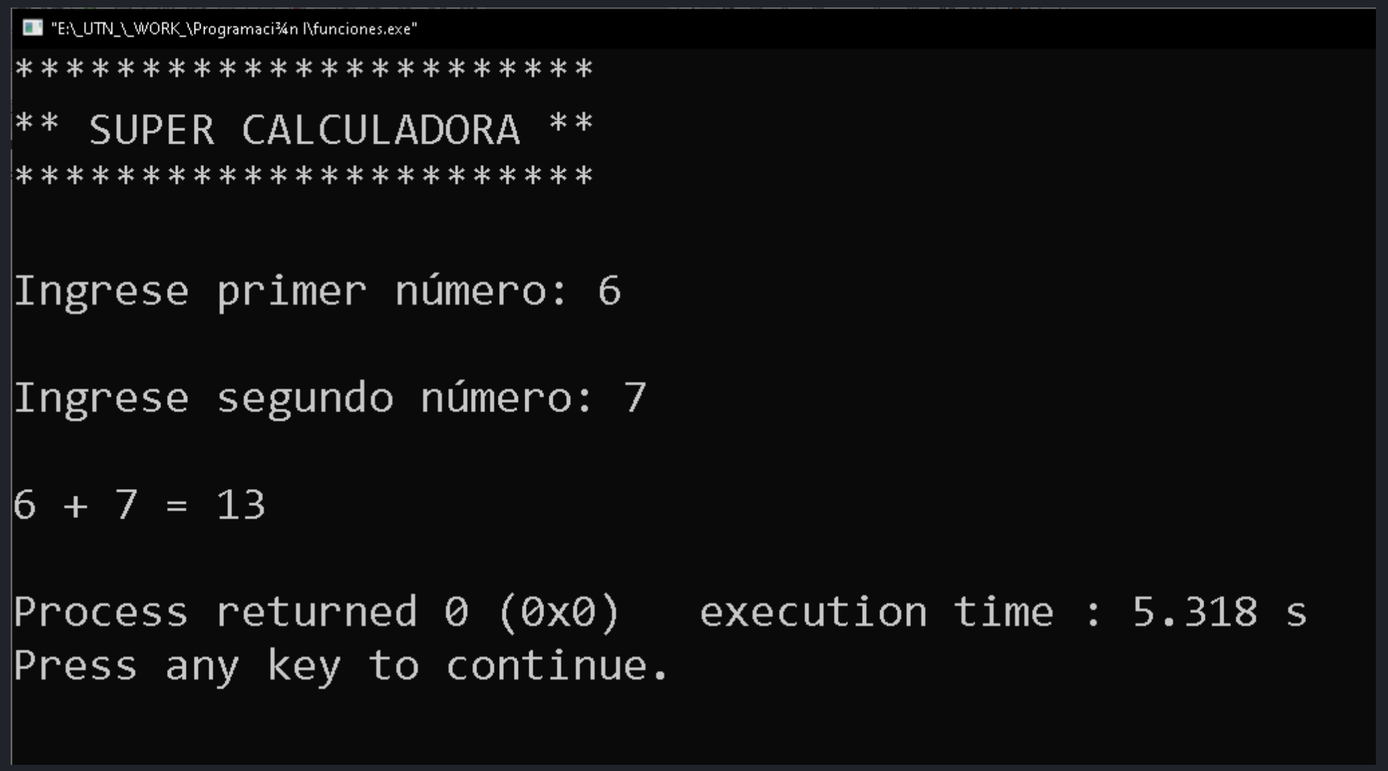
Código

```c
// Programación I - Funciones.

#include<stdio.h>

#define p printf
#define s scanf

// Prototipos:

void saludar(void);
void ingresarDatos(int*, int*);
int suma(int, int);
void imprimirSuma(int, int, int);

int main(void)
{
    int num1, num2, resultado;

    saludar();
    ingresarDatos(&num1, &num2);
    resultado = suma(num1, num2);

    imprimirSuma(num1, num2, resultado);

    return 0;
}

// FUNCIONES:

int suma(int num1, int num2)
{
    int resultado;
    resultado = num1+num2;
    return resultado;
}

void saludar(void)
{
    p("************************\n");
    p("** SUPER CALCULADORA **\n");
    p("************************\n\n");
}

void ingresarDatos(int *num1, int *num2)
{
    p("Ingrese primer n%cmero: ", 163);
    s("%d", num1);
    p("\nIngrese segundo n%cmero: ", 163);
    s("%d", num2);
}

void imprimirSuma(int num1, int num2, int resultado)
{
    p("\n%d + %d = %d\n", num1, num2, resultado);
}
```

```
1   // Programación I - Funciones.
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Ejemplo:

Salida



EJERCITACIÓN (Level 1):

1. Modularizar y poner en funciones todos los ejercicios de las prácticas anteriores. También deben poder ejecutarse tantas veces como se requiera a través de un menú.
2. Crea un programa que pida dos número enteros al usuario y diga si alguno de ellos es múltiplo del otro. Crea una función *EsMultiplo* que reciba los dos números, y devuelve si el primero es múltiplo del segundo.
3. Crear una función que calcule la temperatura media de un día a partir de la temperatura máxima y mínima. Crear un programa principal, que utilizando la función anterior, vaya pidiendo la temperatura máxima y mínima de cada día y vaya mostrando la media. El programa pedirá el número de días que se van a introducir.
4. Crear una subrutina llamada “*login*”, que recibe el DNI del usuario y una contraseña y te devuelve Verdadero si el DNI del usuario es “87654321” y la contraseña es “123456”. Además recibe el número de intentos que se ha intentado al hacer login y si no se ha podido hacer login incremente este valor. Crea un programa principal donde se pida el DNI del usuario y su contraseña y se intente hacer login, solamente tenemos tres oportunidades para intentarlo.
5. Desarrolla un programa que escriba todos los primos hasta el enésimo número indicado por el usuario.

EJERCITACIÓN (Level 2):

1. Escribir dos funciones que permitan calcular:
   - La cantidad de segundos en un tiempo dado en horas, minutos y segundos.
   - La cantidad de horas, minutos y segundos de un tiempo dado en segundos.
   - Escribe un programa principal con un menú donde se pueda elegir la opción de convertir a segundos, convertir a horas, minutos y segundos o salir del programa.
2. El día juliano correspondiente a una fecha es un número entero que indica los días que han transcurrido desde el 1 de enero del año indicado. Queremos crear un programa principal que al introducir una fecha (DD, MM, AAAA) nos diga el día juliano que corresponde. Para ello podemos hacer las siguientes funciones:
   - LeerFecha: Nos permite leer por teclado una fecha (DD, MM, AAAA).
   - ValidarFecha: Valida laa fecha ingresada en la función anterior.
   - DiasDelMes: Recibe un mes y un año y nos dice los días de ese mes en ese año.
   - EsBisiesto: Recibe un año y nos dice si es bisiesto.
   - Calcular_Dia_Juliano: recibe una fecha y nos devuelve el día juliano.

UTN INSPT

REALIZADO POR *MIGUEL SILVA C.*

miguel.silva@inspt.utn.edu.ar