MÉTODOS y FUNCIONES

1. Métodos
2. Funciones
3. Argumentos
4. Alcance Local y Global

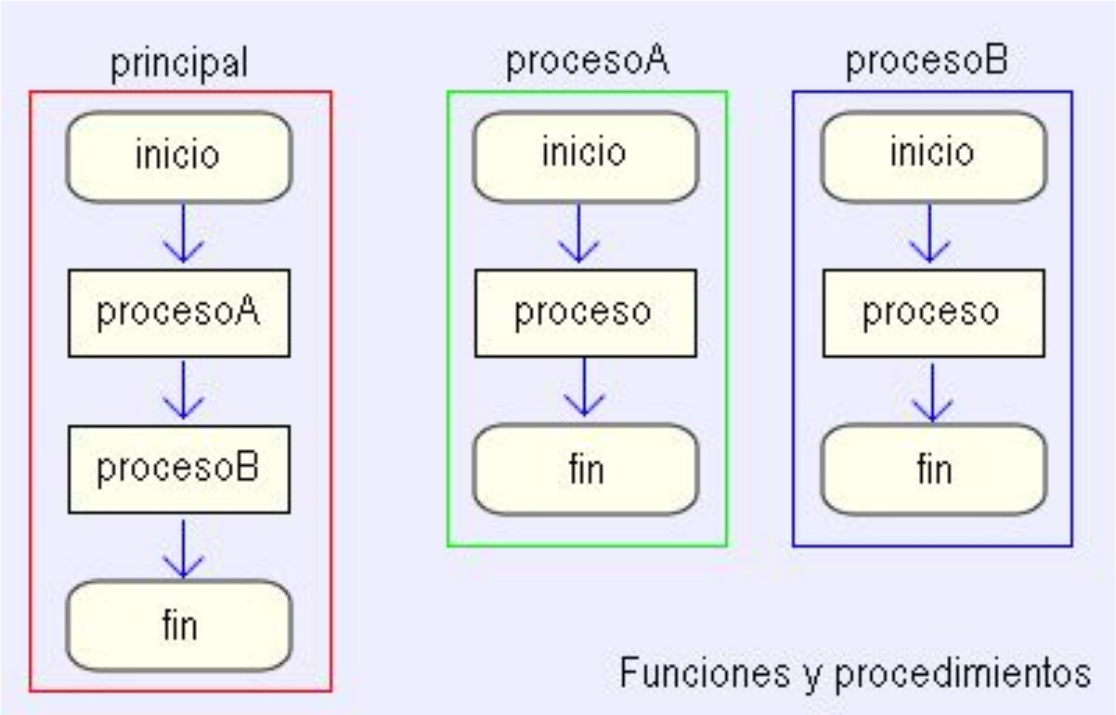
Ejercicios | Ejemplos

MÉTODOS Y FUNCIONES

Definición

- La resolución de problemas complejos se facilita considerablemente si se dividen en problemas más pequeños; y la resolución de estos subproblemas se realiza mediante subalgoritmos.
- Los subalgoritmos son unidades de programa o módulos que están diseñados para ejecutar alguna tarea específica.
- Éstos, son constituidos por funciones o procedimientos (métodos), los cuales se escriben solamente una vez, pero pueden ser referenciados en diferentes puntos del programa, de modo que se puede evitar la duplicación innecesaria del código.
- El módulo principal se ejecuta en una primera instancia, que da la orden de inicio de ejecución de los subprogramas.

MÉTODOS Y FUNCIONES



MÉTODOS Y FUNCIONES - Funciones

Definición

- Una función es un conjunto de líneas de código que realizan una tarea específica y retornan un valor.
- Las funciones pueden tomar parámetros que modifiquen su funcionamiento.
- Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código.
- Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.

Ejemplos

```
<tipo> <nombre> ( [Parámetros] )
{
    cuerpo;
}
```

```
// regresar el cuadrado de un número
double Cuadrado(double n)
{
    return n*n;
}
```

MÉTODOS Y FUNCIONES - Métodos

Definición

- Bajo ciertas circunstancias se necesitará escribir funciones que no regresen valor alguno y para ello se podrán declarar a la función como void.
- La palabra reservada void es utilizada para declarar funciones sin valor de retorno.
- Las funciones que retornan **void** no precisan de la palabra reservada **return**
- La principal diferencia entre una función y un método es que el primero debe retornar un valor y el segundo no.

```
<tipo> <nombre> ( [Parámetros] )
{
    cuerpo;
}
```

```
void ImprimeCuadrado(double n)
{
   printstr( n*n );
}
```

MÉTODOS Y FUNCIONES - Parámetros

Definición

- Las funciones y los métodos operan sobre ciertos valores que son pasados a los mismos ya sea como variables.
- Estas variables se denominan parámetros o argumentos.
- Existen lenguajes que permiten que los valores de estos parámetros sean pasados de dos maneras: **por referencia o por valor**.
- Los parámetros pasados por referencia pueden ser alterados por la función que los reciba, mientras que los parámetros pasados por valor o copia no pueden ser alterados por la función que los recibe, es decir, la función puede manipular a su antojo al parámetro, pero ningún cambio hecho sobre este se reflejará en el parámetro original.

MÉTODOS Y FUNCIONES - Parámetros

Ejemplo con C

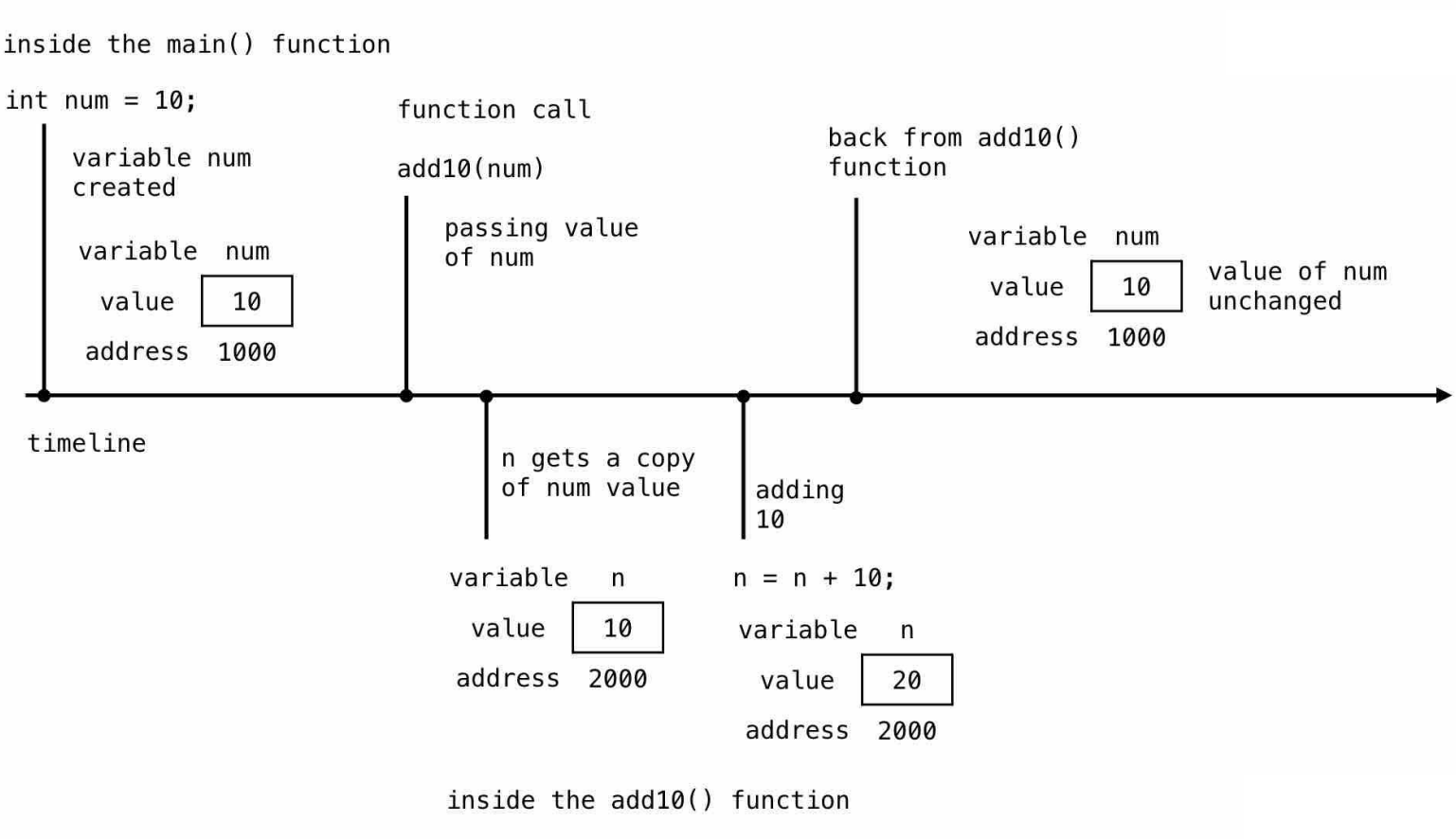
1. Se tiene un método principal Main.
2. Se posee una función.
3. Se llama a la función desde el método Main.
4. Se pasan valores a los parámetros.

MÉTODOS Y FUNCIONES - Parámetros

```c
int main(void) {

  // integer variable
  int num = 10;

  // print value of num
  printf("Value of num before function call: %d\n", num);

  // pass by value
  add10(num);

  // print value of num
  printf("Value of num after function call: %d\n", num);

  return 0;
}
```

```c
// function definition
void add10(int n) {
  n = n + 10;
  printf("Inside add10(): Value %d\n", n);
}
```

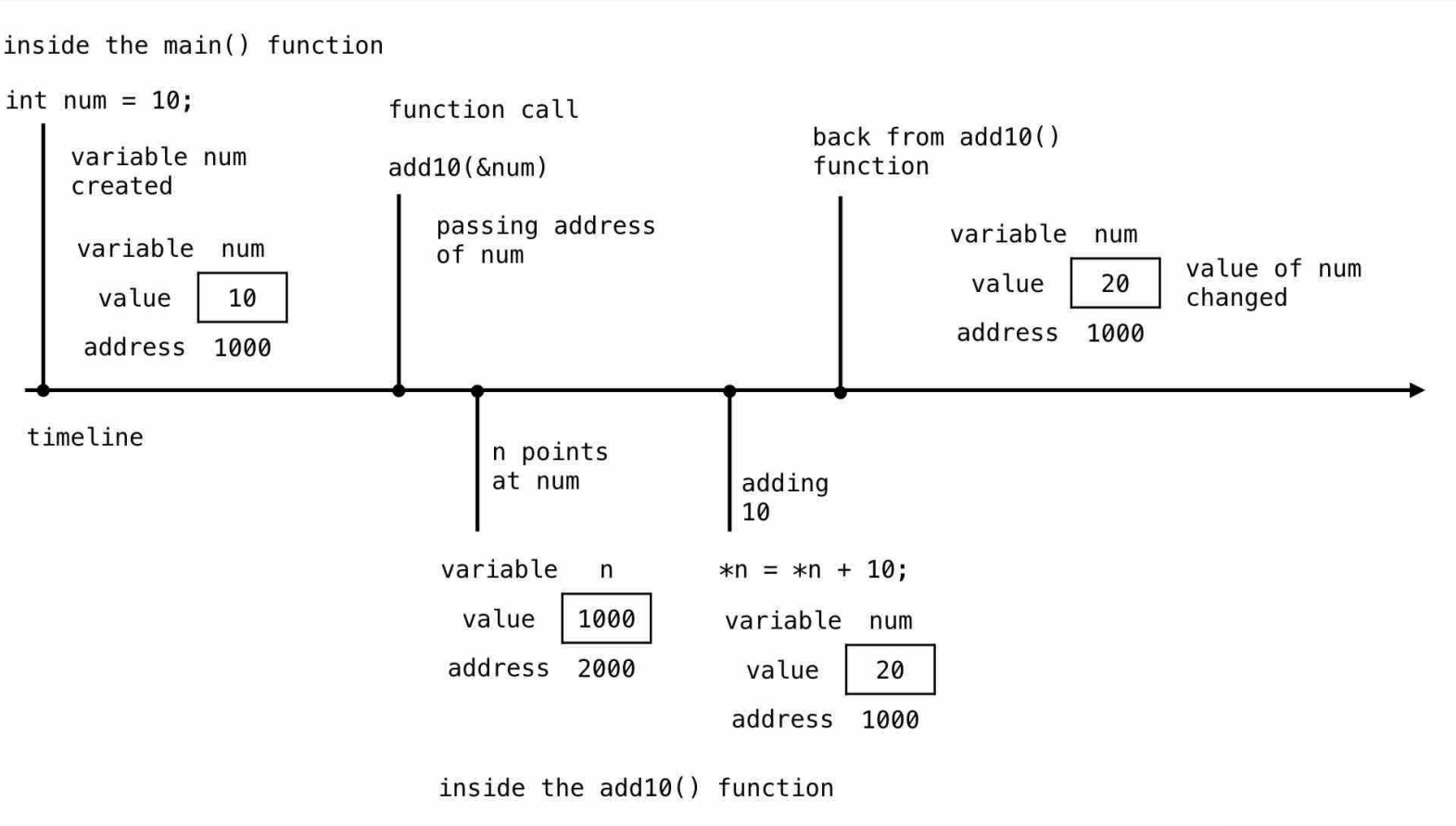
MÉTODOS Y FUNCIONES - Parámetros por Valor



Aclaración de Pasaje por Valor

1. Cuando se pasa **num** a la función el valor de **num,** es decir 10, se guarda en la ubicación **2000**.
2. Y luego agregamos 10 dentro de la función **add10**(), por lo tanto, el nuevo valor en la ubicación 2000 se convierte en 20.
3. Pero el valor en la ubicación 1000 donde se almacena la variable **num** aún permanece en 10, por lo tanto, obtenemos el resultado anterior donde **num** permanece sin cambios.

MÉTODOS Y FUNCIONES - Parámetros por Valor



Aclaración de Pasaje por Referencia

1. Cuando llamamos a la función **add10()** estamos pasando la dirección de memoria de **num** a la función como argumento, por lo tanto, la variable de parámetro de función **n** a la que se asigna la ubicación de memoria 2000, almacena la dirección de la variable **num**, es decir, 1000. Entonces, **n** apunta a **num**.
2. Dentro de la función **add10()** estamos agregando 10 usando el siguiente código: ***n = *n + 10**
3. Donde, ***n** significa el valor en la dirección almacenada en la variable de puntero **n**.
4. Entonces, la dirección almacenada dentro de la variable de puntero **n** es 1000 que apunta a la variable **num**, entonces, ***n** nos da 10 es decir, valor de **num**.
5. Ya que se ha actualizado el valor en la ubicación 1000 desde dentro de la función **add10()**, cuando regresamos de la llamada a la función obtenemos el valor actualizado de la variable **num**.

MÉTODOS Y FUNCIONES - Alcance de variables y parámetros

Alcances

Tanto las variables como los parámetros de las funciones y métodos poseen diferentes alcances (scopes).

Los mismos permiten determinar los valores de las mismas, ya se que éstas se definan con el mismo nombre en distintos scopes del programa.