ITEDES

Educación Digital

Módulo:

Fundamentos de Ingeniería de Software

Segmento:

Algoritmos y Estructuras de Datos

Tema:

Funciones

Prof. Germán C. Basisty
german.basisty@itedes.com

Índice de Contenidos

Funciones

Una **subrutina** o **subprograma** (también llamada **procedimiento**, **función** o **rutina**), como idea general, se presenta como un subalgoritmo que forma parte del algoritmo principal, el cual permite resolver una tarea específica.

Desde un punto de vista práctico, podemos decir que una función es una parte de un programa (subrutina) con un nombre, que puede ser invocada (llamada a ejecución) desde otras partes tantas veces como se desee; un bloque de código que puede ser ejecutado como una unidad funcional. Opcionalmente puede recibir parámetros, se ejecuta y puede devolver un valor. Desde el punto de vista de la organización, podemos decir que una función es algo que permite un cierto orden en una maraña de algoritmos.

El uso de funciones favorece la reutilización de código, dado que una función particular que resuelva un problema específico puede ser utilizada desde múltiples programas.

También hace más sencillo el mantenimiento del código, ya que evita “copiar y pegar” bloques de instrucciones, y de esta manera si hay que hacer alguna corrección, basta con modificar la biblioteca de funciones y el cambio impactará todas las aplicaciones afectadas (en lenguajes compilados eventualmente puede ser necesario recompilar).

Punto de entrada

Cada programa debe tener una sola función principal (generalmente llamada main), que desde la óptica del programador define el punto de entrada al programa.
Esta función principal es la que se ejecuta cuando se lanza el programa, y es la encargada de gestionar el flujo del mismo, transfiriendo el control a otras funciones en la medida que sea necesario.

Espectro de visibilidad de las variables

Las variables pueden ser *globales* o *locales* dependiendo de su ámbito de acción.
Una **variable local** es aquella cuyo ámbito se restringe a la función que la ha declarado y se dice entonces que la variable es local a esa función. Esto implica que esa variable sólo va a poder ser manipulada en dicha sección, y no se podrá hacer referencia fuera de dicha sección.
Una **variable global** es aquella que se define fuera del cuerpo de cualquier función.El ámbito de una variable global son todas las funciones que componen el programa, cualquier función puede acceder a dichas variables para leer y escribir en ellas. Es decir, se puede hacer referencia a ella en cualquier parte del programa.

Consideraciones sobre BASH

*En **BASH**, todas las variables son globales salvo que se indique lo contrario.* Las variables locales sólo pueden ser utilizadas dentro de funciones, no en el cuerpo principal del programa.
Dentro de una función, para declarar una variable como local:

```
function miFuncion() {
        declare -i miVariable=0
}
```

Para pasar parámetros a una función en **BASH** los escribimos justo después de la llamada. Los parámetros son recibidos como $1, $2, $3, etc donde el número representa el orden en que fueron pasados. Es una buena práctica declarar variables locales con nombres adecuados según la lógica de la función, y asignarles los parámetros como corresponda, por ejemplo:

```
function saludar() {
        declare nombre1=$1
        declare nombre2=$2

        echo “Hola $nombre1 y $nombre2!!!!”
}

declare miNombre=”Juan”
declare tuNombre=”Pedro”

saludar $miNombre $tuNombre
```

Para retornar un valor desde una función en bash, se debe utilizar **echo**, ya sea de una variable local o de una operación. El resultado puede ser almacenado dentro de una variable del cuerpo principal del programa.

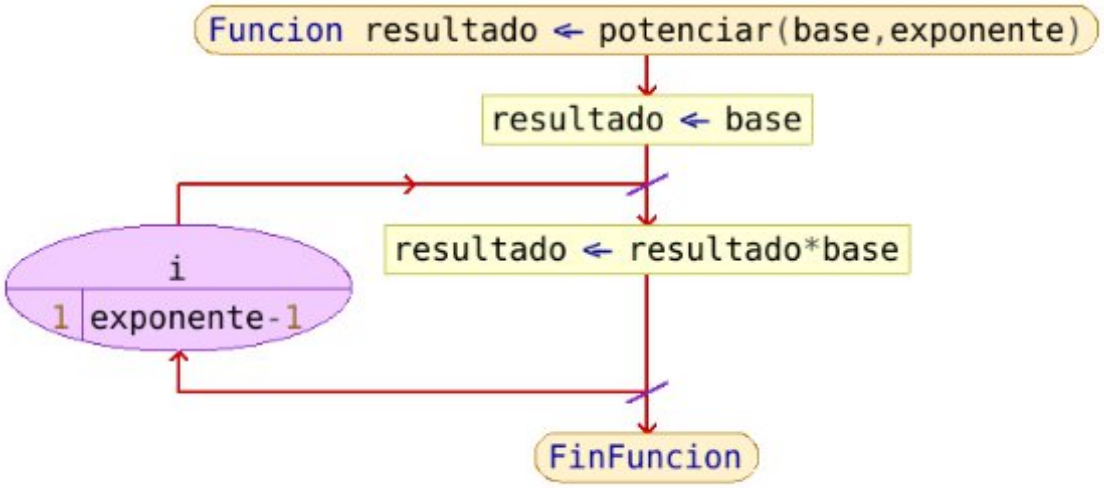
Ejemplo 1

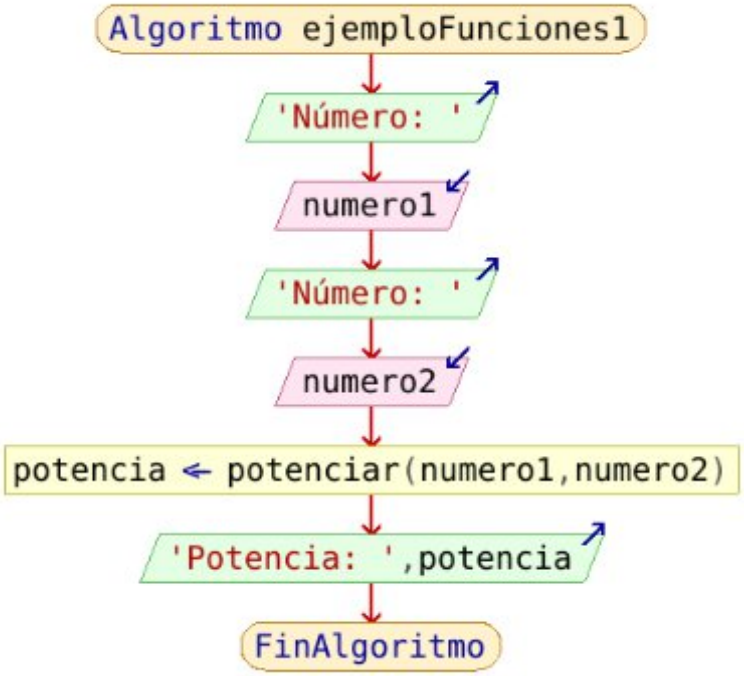
Si entendemos la potenciación como una sucesión de multiplicaciones, podríamos generar una función que reciba como parámetros la base y el exponente, para luego ser utilizada allí donde sea necesario:

Pseudocódigo

```
SubProceso resultado <- potenciar(base, exponente)
        resultado = base

        Para i = 1 Hasta exponente - 1 Con Paso 1
                resultado = resultado * base
        FinPara
FinSubProceso

Algoritmo ejemploFunciones1
        Escribir "Número: "
        Leer numero1
        Escribir "Número: "
        Leer numero2

        potencia = potenciar(numero1, numero2)

        Escribir "Potencia: ", potencia
FinAlgoritmo
```

Diagrama de flujo





BASH

```
#!/bin/bash

function pow() {
        declare -i powBase=$1
        declare -i powExponent=$2
        declare -i result=powBase

for ((i = 1; i < powExponent; i++)) {
                result=$((result * powBase))
        }

        echo $result
}

declare -i base=0
read -p “Base: “ base

declare -i exponent=0
read -p "Exponente: " exponent

declare -i powResult=$( pow $base $exponent )

echo "Potencia: $powResult"

exit 0
```

Python

```
def pow(base, exponent):
        result = base

        for i in range(1, exponent):
                result *= base

        return result


base = int(input("Ingrese base: "))
exponent = int(input("Ingrese exponente: "))

print("Potencia: " + str(pow(base, exponent)))
```

Java

```
import java.util.Scanner;

public class Ejemplo1 {
    public static Integer pow(Integer base, Integer exponent) {
        Integer result = base;

        for(Integer i = 1; i < exponent; i++)
            result *= base;

        return result;
    }
    public static void main(String args[]) {
        Scanner teclado = new Scanner(System.in);

        System.out.print("Ingrese base:");
        Integer base = Integer.parseInt(teclado.nextLine());

        System.out.print("Ingrese exponente:");
        Integer exponent = Integer.parseInt(teclado.nextLine());

        Integer result = pow(base, exponent);

        System.out.println("Resultado: " + result.toString());
    }
}
```

C#

```
using System;

namespace cs
{
   class Program
   {
      static int pow(int basex, int exponent)
      {
         int result = base;

         for(Integer i = 1; i < exponent; i++)
            result *= base;

         return result;
      }

      static void Main(string[] args)
      {
         Console.Write("Base: ");
         int basex = Int32.Parse(Console.ReadLine());

         Console.Write("Exponente: ");
         int exponent = Int32.Parse(Console.ReadLine());

         int result = pow(basex, exponent);

         Console.WriteLine("Resultado: " + result.ToString());
      }
   }
}
```

JavaScript

```
function pow(base, exponent) {
    let result = base;

    for(let i = 1; i < exponent; i++)
        result *= base;

    return result;
}

function main() {
    const base = parseInt(prompt("Base"));
    const exponent = parseInt(prompt("Exponente"));
    const result = pow(base, exponent);

    alert(`Resultado: ${result}`);
}
```

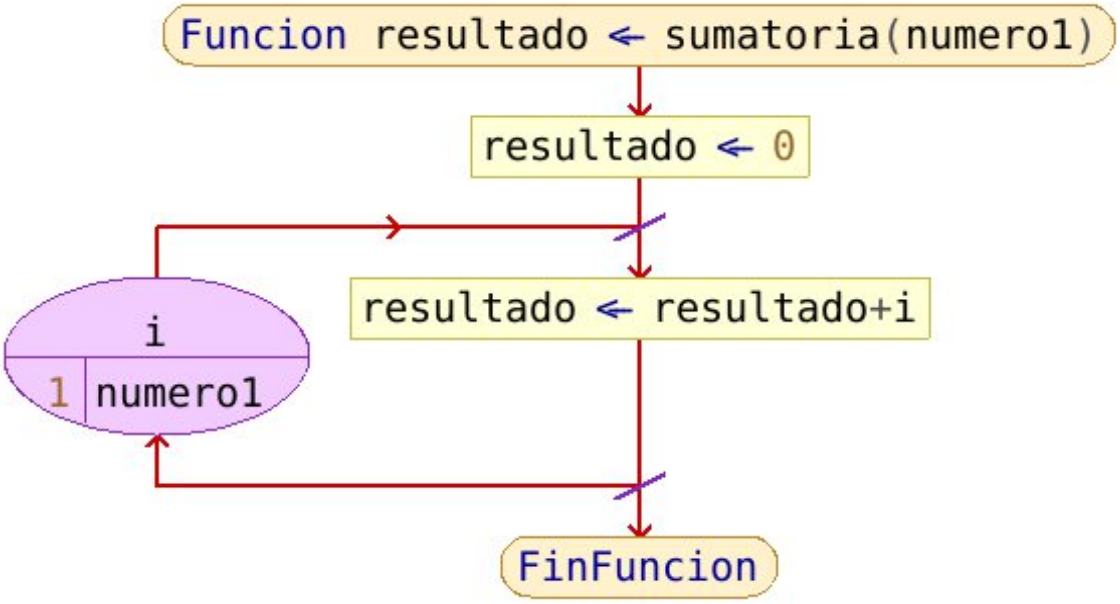
Ejemplo 2

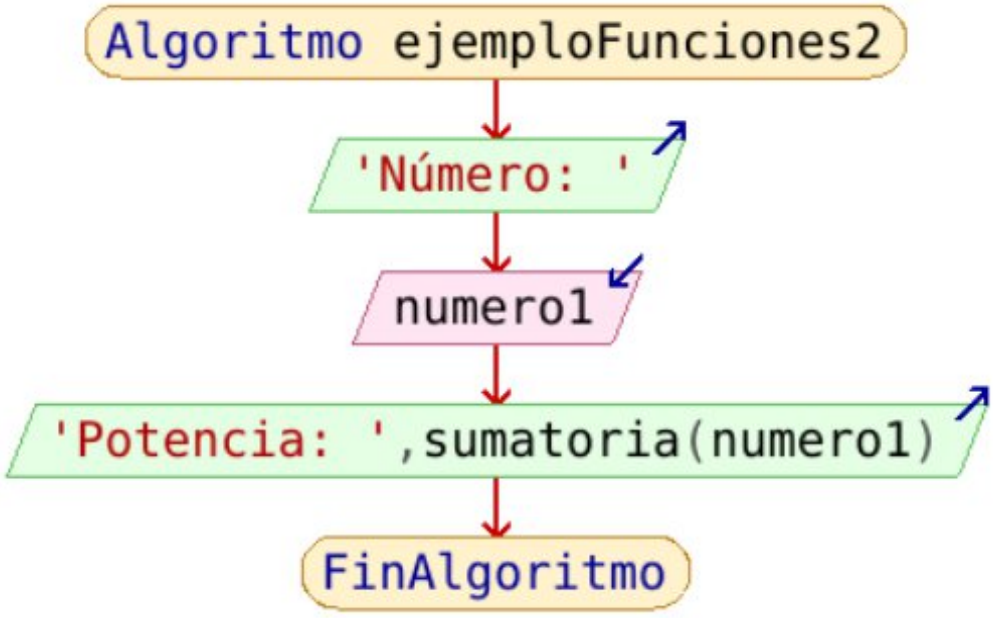
Realizar un algoritmo que reciba un número entero positivo y muestre la sumatoria.

Pseudocódigo

```
Funcion resultado <- sumatoria(numero1)
        resultado = 0
        Para i <- 1 Hasta numero1 Hacer
                resultado <- resultado + i
        FinPara
FinFuncion

Algoritmo ejemploFunciones2
        Escribir 'Número: '
        Leer numero1

        Escribir 'Potencia: ', sumatoria(numero1)
FinAlgoritmo
```

Diagrama de flujo





BASH

```
#!/bin/bash

function summatory() {
        declare -i number=$1
declare -i result=0

for((i = 1; i <= number; i++)) {
result=$((result + i))
}

echo $result
}

declare -i number1=0
read -p "Número: " number1

declare -i summatoryResult=$(summatory $number1)

echo "Sumatoria: $summatoryResult"

exit 0
```

Python

```
def summatory(number):
    result = 0

    for i in range(1, number + 1):
        result += i

    return result

number = int(input("Ingrese un número: "))

print("La sumatoria es: " + str(summatory(number)))
```

Java

```
import java.util.Scanner;

public class Ejemplo2 {
    public static Integer sumatory(Integer number) {
        Integer result = 0;

        for(Integer i = 1; i <= number; i++)
            result += i;

        return result;
    }

    public static void main(String args[]) {
        Scanner teclado = new Scanner(System.in);

        System.out.print("Ingrese un numero:");
        Integer number = Integer.parseInt(teclado.nextLine());

        System.out.println("a sumatoria es: " + sumatory(number).toString());
    }
}
```

C#

```
using System;

namespace cs
{
    class Program
    {
        static int sumatory(int number)
        {
            int result = 0;

            for(Integer i = 1; i <= number; i++)
                result += i;

            return result;
        }
        static void Main(string[] args)
        {
            Console.Write("Ingrese un numero: ");
            int number = Int32.Parse(Console.ReadLine());

            Console.WriteLine("Resultado: " + sumatory(number).ToString());
        }
    }
}
```

JavaScript

```
function sumatory(number) {
    let result = 0;

    for(let i = 1; i <= number; i++)
        result += i;

    return result;
}

function main() {
    const number = parseInt(prompt("Ingrese un numero"));

    alert(`Resultado: ${sumatory(number)}`);
}
```

Ejercitación

1) Crear una función que reciba un número entero y devuelva la cantidad de divisores, por ejemplo, para el número 16, sus divisores son 1, 2, 4, 8, 16, por lo que la respuesta debería ser 5 (utilizar módulo). Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.

2) Crear un programa que pida dos número enteros al usuario y diga si alguno de ellos es múltiplo del otro. Desarrollar una función que ayude a que el proceso principal sea legible. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.

3) Crear una función que reciba tres números enteros y devuelva el valor del mayor de ellos. Por ejemplo, para los números 5, 7 y 5, devolvería el valor 7. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.

4) Crear una función que devuelva el valor lógico "verdadero" o "falso" según si el número que se indique como parámetro es par o no lo es. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.

5) Crear una función que reciba un número y lo devuelva elevado al cubo. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.

6) Desarrollar una calculadora que realice suma, resta, multiplicación, división y potencia. Repetir hasta que el usuario decida salir. Utilizar funciones. Presentar diagrama de flujo, pseudocódigo y código fuente funcionando en BASH, Python, Java, C# y JavaScript + HTML.