Fundamentos de Programación

UNIDAD 4: MODULARIDAD

Facultad de Ingeniería
Universidad Nacional de Jujuy

PROBLEMÁTICA (1)

PROBLEMAS GRANDES Y COMPLEJOS

PROBLEMÁTICA (2)

REPETIR CÓDIGO

PROBLEMÁTICA (3)

CÓDIGO POCO LEGIBLE

PROBLEMÁTICA (4)

MANTENIMIENTO

1. Análisis del Problema
2. Diseño del Algoritmo
3. Codificación
4. Compilación y Ejecución
5. Verificación y Depuración
6. Documentación y Mantenimiento

MODULARIDAD (1)

- Surge de la técnica Divida y Vencerás.
- La resolución de un problema se descompone en Módulos.
- Cada módulo se divide en nuevos módulos hasta que el problema se reduce a actividades básicas.
- Se van fusionando y conectando los módulos para resolver el problema original.

MODULARIDAD (2)

LA PROGRAMACIÓN MODULAR

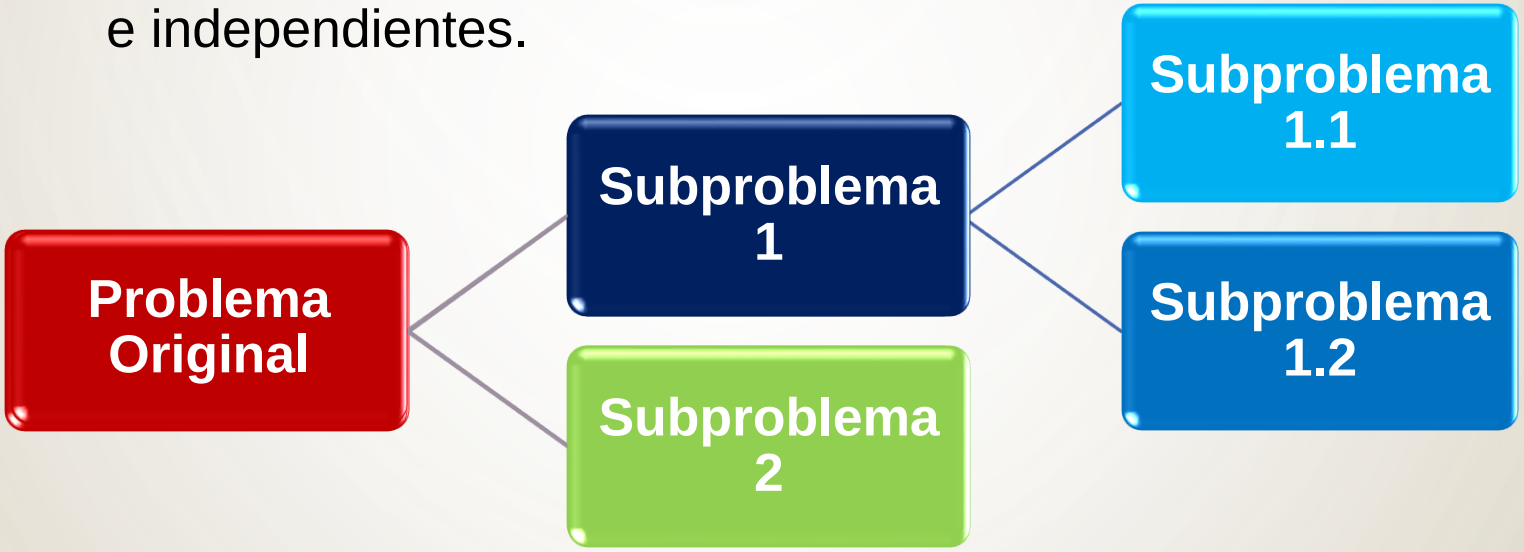
- es un método de diseño flexible
- permite dividir un programa en unidades de trabajo (subprogramas)
- se realiza mediante descomposición de problemas, abstracción y módulos

MODULARIDAD(3)

- Abstracción
  - Permite representar los objetos relevantes del problema.

MODULARIDAD (4)

- Descomposición de problemas
  - Un problema complejo puede dividirse en problemas sencillos e independientes.



MODULARIDAD (5)

- Módulos
  - un programa puede estar formado por partes independientes que resuelven subproblemas específicos.
  - pueden analizarse, codificarse y probarse por separado
  - el módulo principal controla el flujo de acciones
  - se clasifican en Procedimientos y Funciones

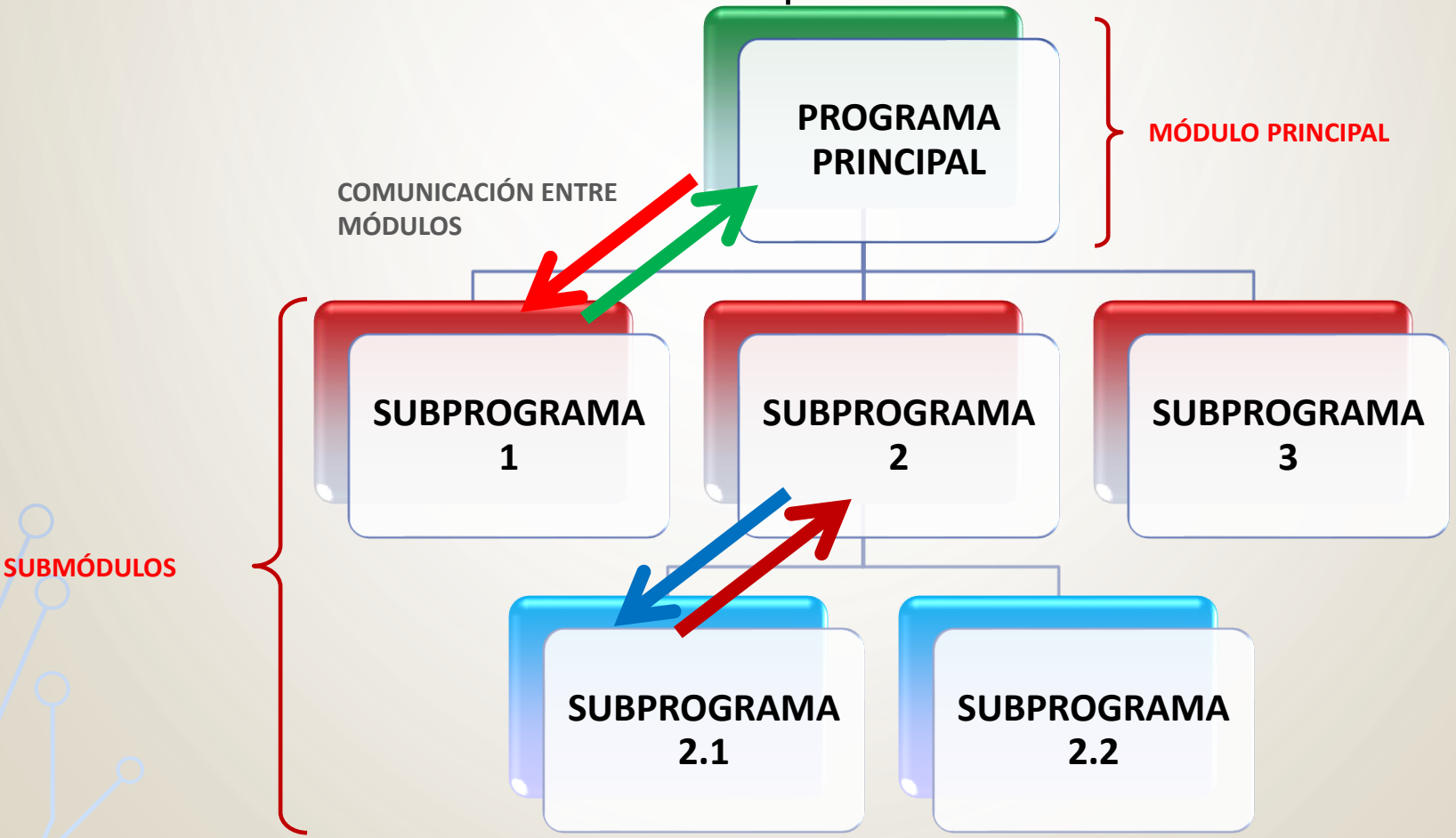
MODULARIDAD (6)

- Entradas: se conoce el conjunto de datos con los que trabajará el módulo.
- Propósito: se conoce el objetivo del módulo (qué hace).
- Salidas: se conoce el resultado que generará el módulo.

Cada módulo tiene una tarea bien definida.

MODULARIDAD (7)

- Los módulos pueden ser usados por otros módulos para resolver un problema
- Un módulo se puede comunicar con otro a través de una interfaz de comunicación que debe estar bien definida



FUNCIONES (1)

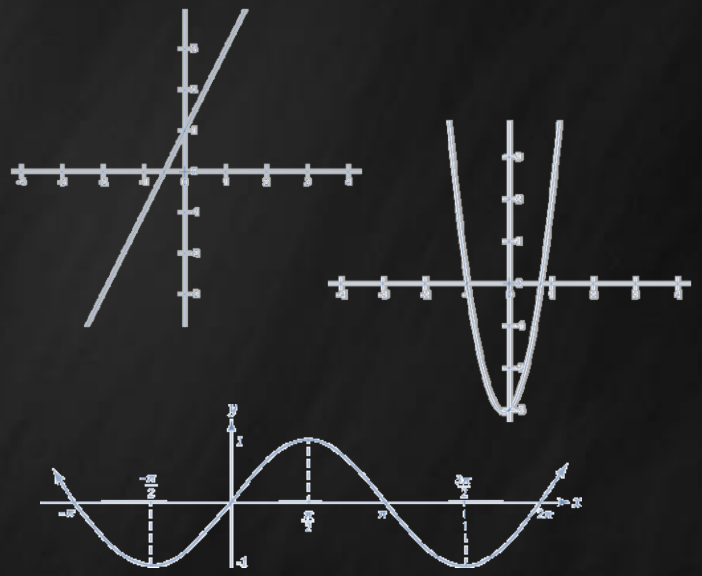
Las funciones matemáticas se definen como la expresión matemática de la relación existente entre dos variables o magnitudes.

Por ejemplo:

$F(x) = 2x + 1$

$F(x) = 4x^2 + x - 3$

$F(x) = \text{sen } x$



FUNCIONES (2)

Parámetros Formales: son valores genéricos (variables) con los que se define la función.

Por ejemplo: $F(x) = 2x + 1$

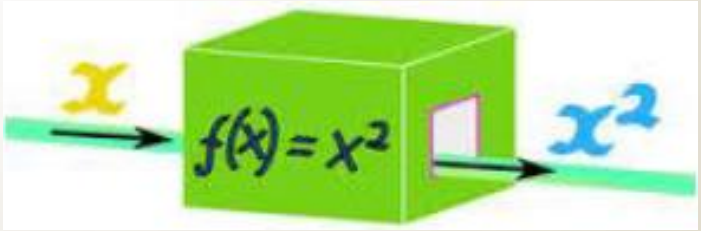
Parámetros Actuales: son valores específicos que utilizará la función para calcular un resultado.

Por ejemplo: $F(3) = 2 \cdot 3 + 1$

$F(3) = 7$

FUNCIONES (3)

- Una función es un módulo o subprograma que toma una lista de valores llamados argumentos o parámetros y devuelve un único valor.
- Las funciones se definen de un tipo de dato simple (entero, real, carácter, lógico).
- Las funciones pueden ser internas o definidas por el usuario.



FUNCIONES (4)

```
tipo_función nombre_función (parámetros formales)
{tipo_dato nombre_variables; //variables de la función
   ACCIONES;
   return resultado_de_la_función;
}
```

- **tipo_función**: indica el tipo de resultado que devolverá la función.
- nombre_función: especifica el nombre de la función
- **parámetros formales**: indica los valores (y sus tipos) que utilizará la función.
- **variables de la función**: son las variables creadas sólo para la función (locales).
- **ACCIONES**: sentencias secuenciales, selectivas o repetitivas que implementan la operación.
- **return resultado**: asigna el resultado final a la función.

INVOCACIÓN DE FUNCIONES

- Un función puede invocarse:

```
variable←nombre_función(parámetros_actuales)
```

```
variable=nombre_funcion(parámetros_actuales);
```

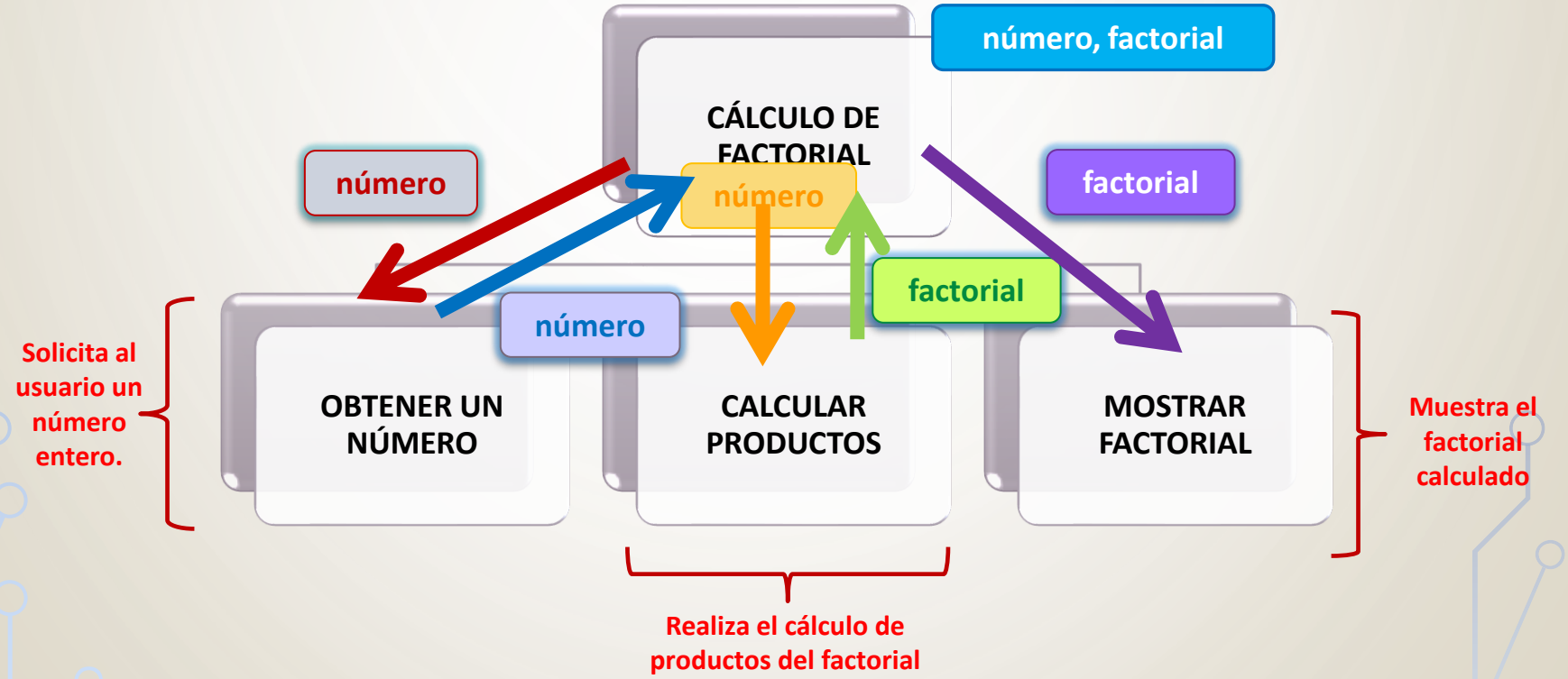
```
ESCRIBIR "Resultado:", nombre_función(parámetros_actuales)
```

```
cout << "Resultado:" << nombre_función(parámetros_actuales);
```

- Al invocar una función:
  1. A cada parámetro formal se le asigna el valor de su correspondiente parámetro actual.
  2. Se ejecuta el cuerpo de acciones de la función.
  3. Se asigna el resultado a la función y se retorna al punto de llamada.

EJEMPLO DE FUNCIONES (1)

- Ejemplo: Diseñe un programa modular que calcule el factorial de un número ingresado por el usuario.



EJEMPLO DE FUNCIONES (2)

- Ejemplo: Diseñe un programa modular que calcule el factorial de un número ingresado por el usuario.

```
#include <iostream>
using namespace std;
```

```
int factorial(int n);
```

```
main()
{ int num, fact;
     cout << "Ingrese número: ";
     cin >> num;
     fact=factorial(num);
     cout << "Factorial: " << fact << endl;
     system("pause");
}
```

```
int factorial (int n)
{ int i,f;
  f=1;
  for(i=1;i<=n;i=i+1)
       f=f*i;
  return f;
}
```

EJEMPLO DE FUNCIONES (3)

- Ejemplo: Diseñe un programa modular que calcule el producto (mediante sumas) de 2 números ingresados por el usuario.

```
#include <iostream>
using namespace std;
```

```
int producto(int a, int b);
```

```
main()
{ int num1, num2, prod;
      cout << "Ingrese número: ";
      cin >> num1;
      cout << "Ingrese número: ";
      cin >> num2;
      prod=producto(num1,num2);
      cout << "Producto: " << prod << endl;
      system("pause");
}
```

```
int producto(int a, int b)
{ int i,p=0;
  for(i=1;i<=b;i=i+1)
        p=p+a;
  return p;
}
```

EJEMPLO DE FUNCIONES (4)

```
#include <iostream>
#include <stdlib.h>

using namespace std;

int division(int n1,int n2);

main()
{
   int num1,num2,resultado;
   cout << "Ingrese divisor: ";
   cin >> num1;
   cout << "Ingrese dividendo: ";
   cin >> num2;
   resultado=division(num1,num2);
   cout << "Resultado: " << resultado << endl;
   system("pause");
}

int division(int n1,int n2)
{
   int cociente=0;
   while(n1 >= n2)
    { n1=n1-n2;
      cociente++;
    }
   return cociente;
}
```

Prototipo de la función

Argumentos de la función

Tipo de dato de la función

Módulo Principal

Llamada o Invocación

Parámetros Actuales

Definición de la función

Parámetros Formales

Valor de retorno

BIBLIOGRAFÍA

- Sznajdleder, Pablo Augusto. Algoritmos a fondo. Alfaomega. 2012.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Hernández, Roberto *et al.* Estructuras de datos y algoritmos. Prentice Hall. 2001.
- Fundamentos Básicos de Programación en C++. Martínez del Rio. Jaén. 2015
- Curso de C++. https://www.programarya.com/Cursos/C++