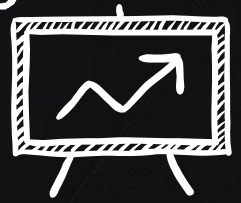
Universidad Autónoma del Estado de México
Facultad de Ingeniería
Ingeniería en Computación

Programación Estructurada

Programación Modular

Elaboró: Silvia Edith Albarrán Trujillo

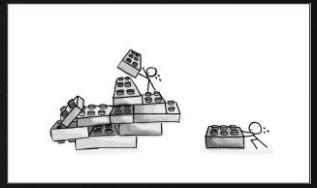
Septiembre 2019



Propósito de la Unidad de Aprendizaje

- Aplicar el paradigma de la programación estructurada para representar en términos algorítmicos la solución de un problema real automatizable.
- Elaborar programas completos utilizando el paradigma de la programación estructurada y mostrando en ellos el dominio pleno de variables simples, vectores, matrices, registros y modularidad.

Estructura de la Unidad de Aprendizaje



1. IDENTIFICAR LAS FASES DE LA METODOLOGÍA DE PROGRAMACIÓN ESTRUCTURADA PARA LA SOLUCIÓN DE PROBLEMAS
2. APLICAR LA PROGRAMACIÓN ESTRUCTURADA EN LA SOLUCIÓN DE PROBLEMAS UTILIZANDO LENGUAJE INFORMAL Y DIAGRAMAS DE FLUJO
3. UTILIZAR ARREGLOS UNIDIMENSIONALES Y BIDIMENSIONALES PARA EL ALMACENAMIENTO DE DATOS EN LA SOLUCIÓN DE PROBLEMAS
4. USAR LAS TÉCNICAS DE PROGRAMACIÓN MODULAR EN EL DESARROLLO DE PROGRAMAS INFORMÁTICOS
5. UTILIZAR LOS REGISTROS PARA ALMACENAR Y MANIPULAR INFORMACIÓN EN EL DESARROLLO DE PROGRAMAS INFORMÁTICOS

Guión para el uso de este material

La información de esta presentación contiene ideas generales que serán explicadas en la clase.

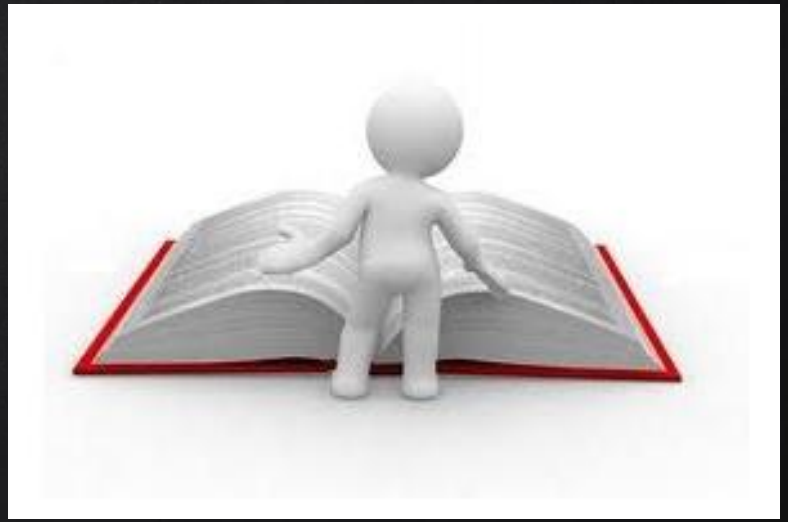
Para ampliar la información que se presenta en esta presentación se incluye al final un apartado de bibliografía.

La presente contiene sólo información de la unidad 4. USAR LAS TÉCNICAS DE PROGRAMACIÓN MODULAR EN EL DESARROLLO DE PROGRAMAS INFORMÁTICOS

Una vez concluida esta unidad el alumno conocerá las técnicas de programación modular y podrá utilizarlas para diseñar programas que permitan automatizar problemas.

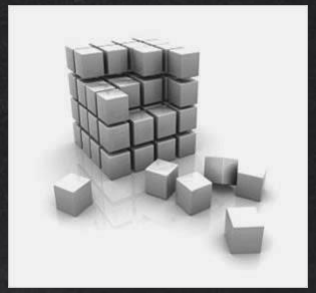
Contenido

- ✘ Portada
- ✘ Propósito de la Unidad de Aprendizaje
- ✘ Estructura de la Unidad de Aprendizaje
- ✘ Guión para uso de este material
- ✘ Contenido del material
- ✘ Objetivo de la Unidad IV
- ✘ Programación Modular
  1. Estructura de un programa modular
  2. Módulos
  3. Variables Globales y Locales
  4. Parámetros
  5. Llamada Subprogramas
  6. Verificación de un programa modular
- ✘ Conclusiones
- ✘ Bibliografía



Objetivo de la Unidad de Aprendizaje

- DISEÑAR PROGRAMAS COMPLETOS UTILIZANDO EL PARADIGMA DE LA PROGRAMACIÓN ESTRUCTURADA Y MOSTRANDO EN ELLOS EL DOMINIO PLENO DE MODULARIDAD.



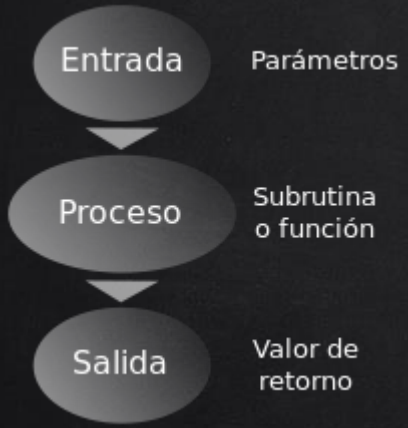
Programación Modular

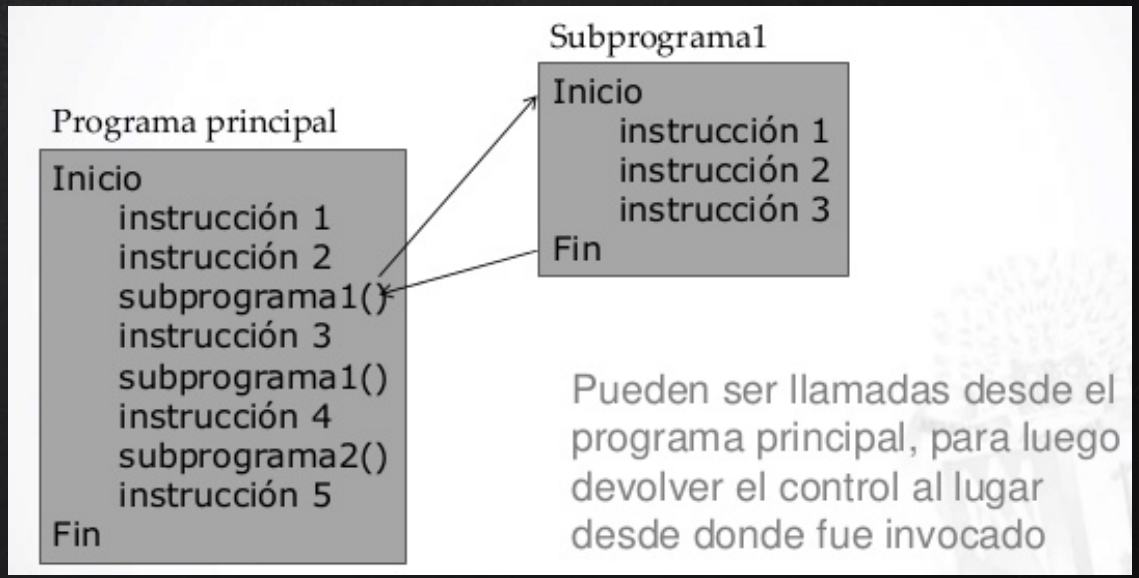
Definición

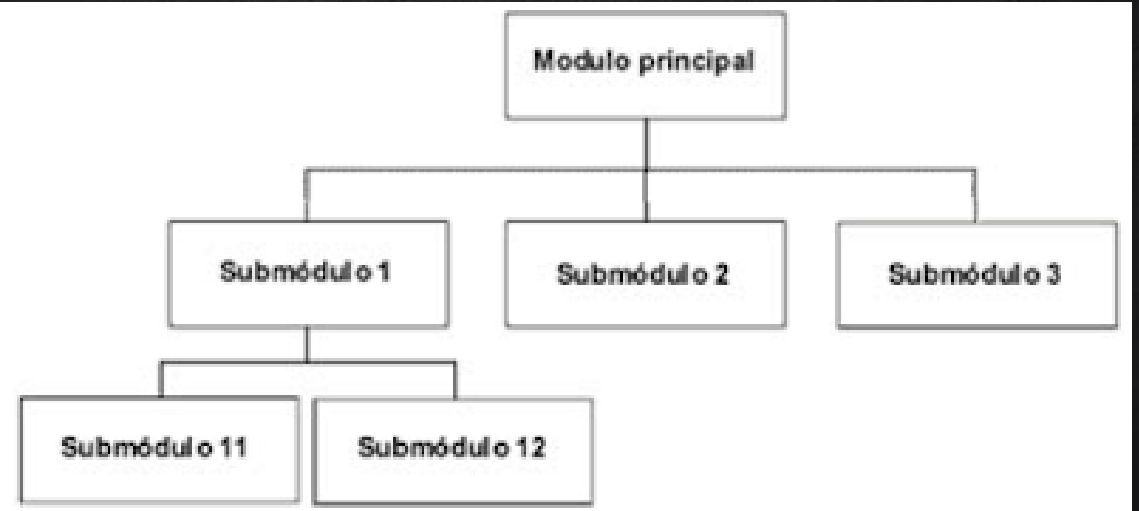
Técnica de programación que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes

Estructura de un programa modular

Estructura de un programa Modular



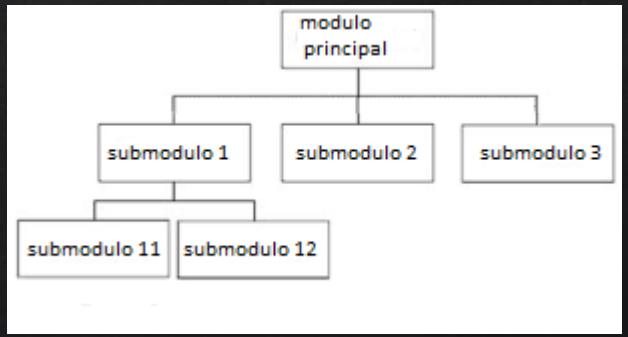




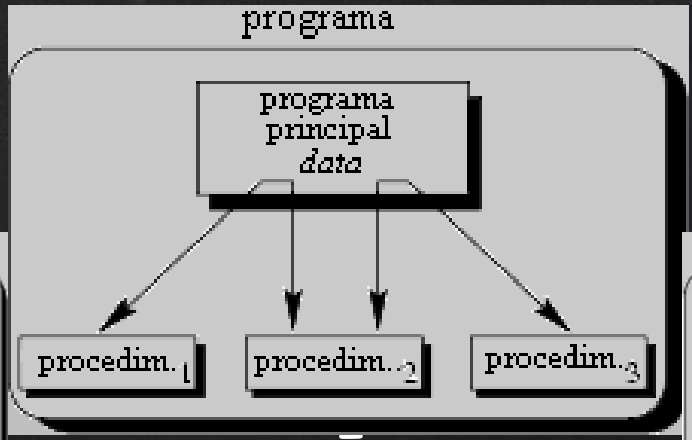
Descomposición modular de un programa

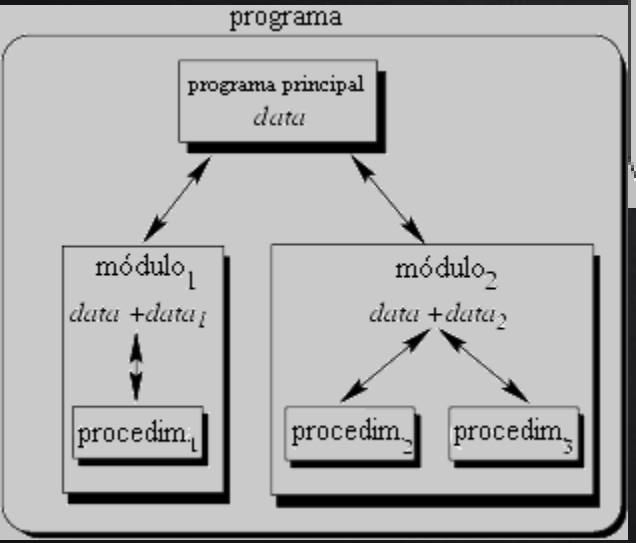
2.

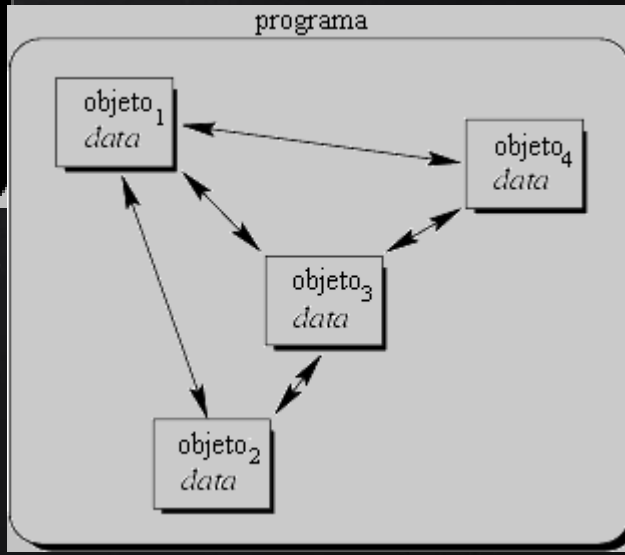
Módulos


modulo principal
submodulo 1
submodulo 2
submodulo 3
submodulo 11
submodulo 12

Tipos de Módulos

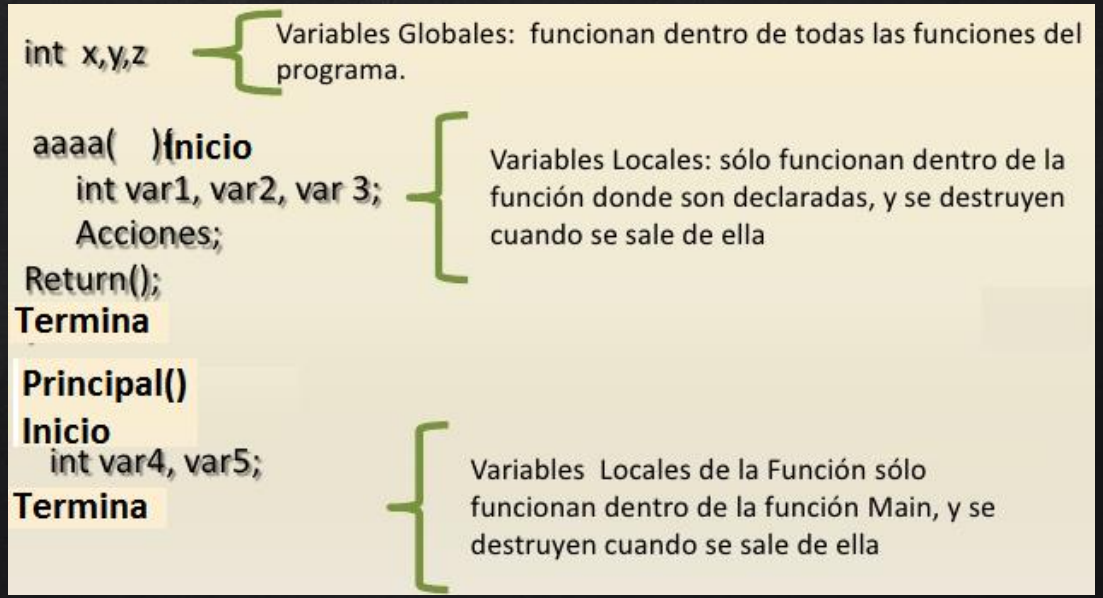






Variables Globales y Locales

Declaración de variables Globales y Locales



Parámetros

- Por valor
- Por referencia (no se utilizan en este curso)

Parámetros por valor

```
1
2   Funcion r<-suma(n1,n2)
3
4       r<-n1+n2;
5
6   FinFuncion
7
8   Proceso sin_titulo
9
10      Escribir "Resultado: ",suma(2,3);
11
12  FinProceso
13
```

```
1   Funcion miFuncion(a)
2
3       a<-a+5;
4
5       Escribir a;
6
7   FinFuncion
8
9
10  Proceso sin_titulo
11
12      i<-20;
13
14      miFuncion(i);
15
16      Escribir "El valor de i es: ",i;
17
18  FinProceso
19
```

Cuando ejecutamos el programa, la salida que produce es:

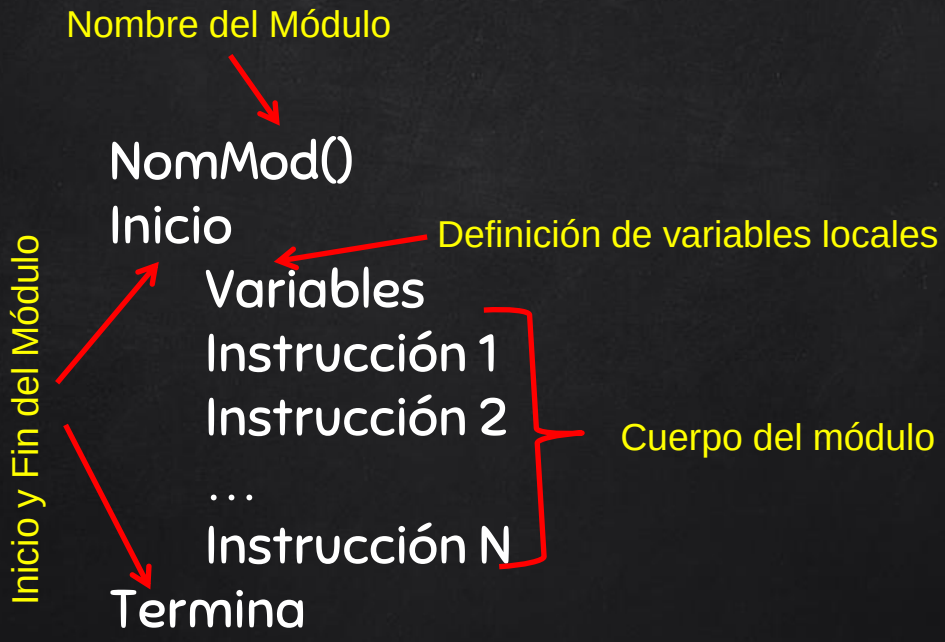
25
El valor de i es: 20

Parámetros por referencia

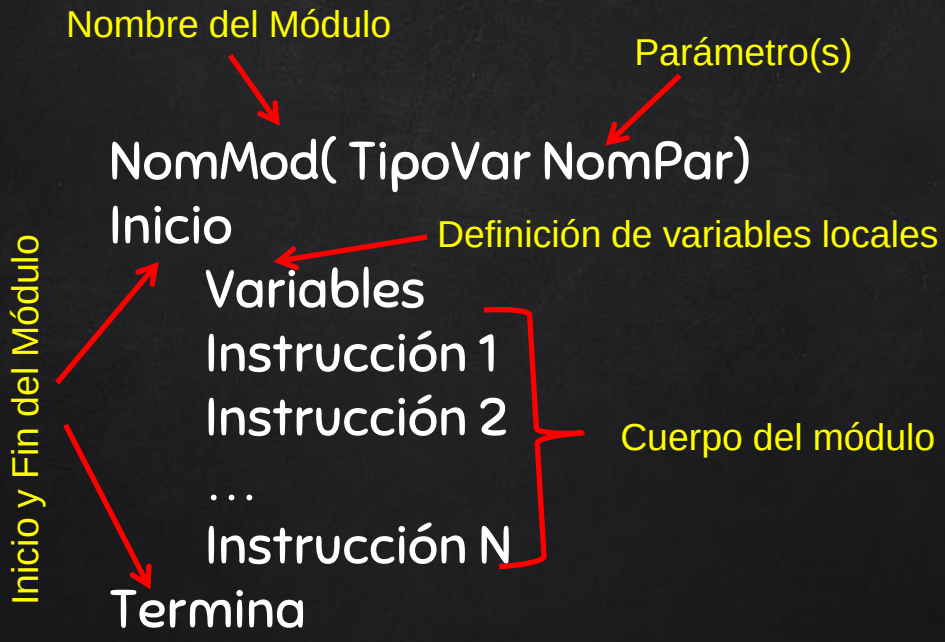
```
1  Funcion miFuncion(a Por Referencia)
2
3      a<-a+5;
4
5      Escribir a;
6
7  FinFuncion
8
9
10 Proceso sin_titulo
11
12     i<-20;
13
14     miFuncion(i);
15
16     Escribir "El valor de i es: ",i;
17
18 FinProceso
19
```

```
1
2  Funcion acum<-suma(arreglo)
3
4      acum<-0;
5      Para j<-0 Hasta 2-1 Con Paso 1 Hacer
6
7          acum<-acum+arreglo[j];
8
9      FinPara
10
11 FinFuncion
12
13 Proceso Principal
14
15     Dimension arr_num[2];
16
17     Para i<-0 Hasta 2-1 Con Paso 1 Hacer
18
19         Escribir "Digite numero ",(i+1);
20         Leer num;
21
22         arr_num[i]<-num;
23
24     FinPara
25
26     Escribir "La suma es: ",suma(arr_num);
27
28 FinProceso
29
```

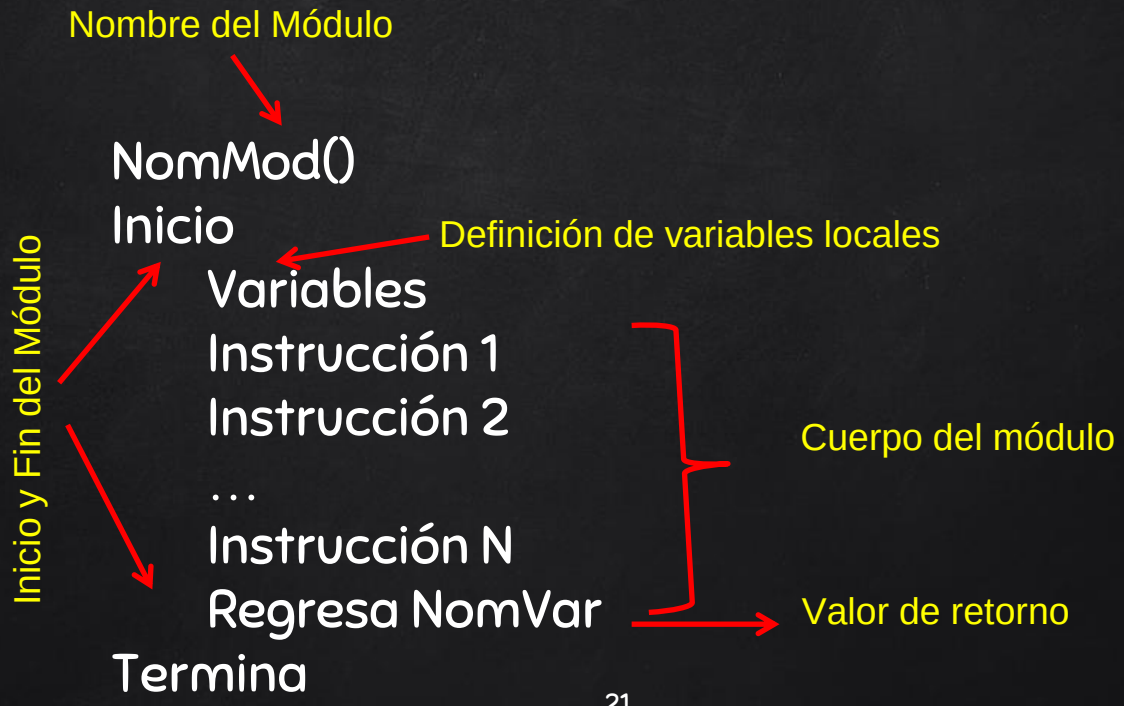
No reciben parámetros ni regresan valor



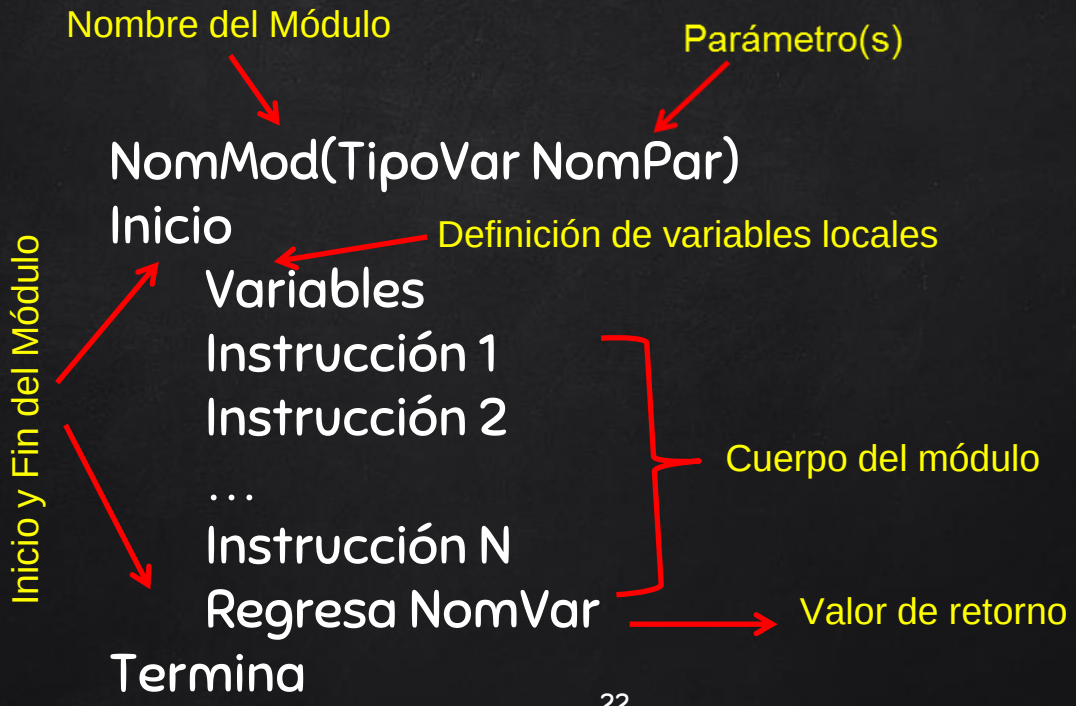
Sí reciben parámetros no regresan valor



No reciben parámetros
Sí regresan valor

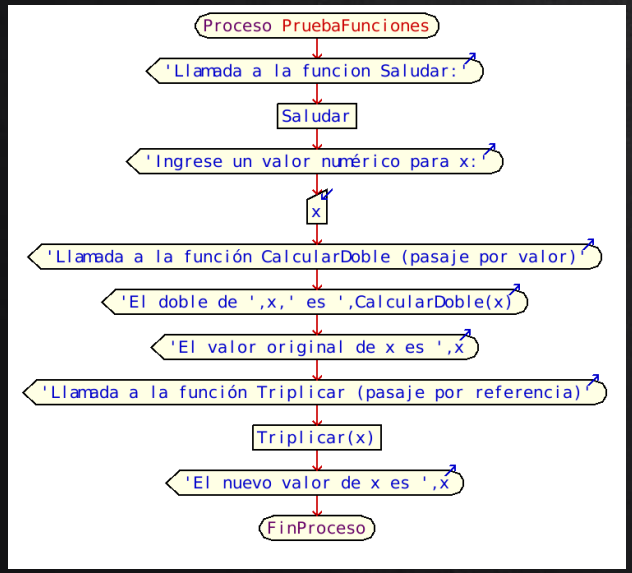


RECIBEN PARÁMETROS Y REGRESAN VALOR



5.

Llamada a Subprogramas




```
1
2   Funcion r<-suma(n1,n2)
3
4       r<-n1+n2;
5
6   FinFuncion
7
8   Proceso sin_titulo
9
10      Escribir "Resultado: ",suma(2,3);
11
12  FinProceso
13
```

Verificación un programa modular

(Pruebas de Escritorio)

Ejemplo

Principal()	Mod1(x: E, y: E)
Inicio h,u,e: E h=2 u=28 e=Mod1(u,h) Termina	Inicio l E l=y div x Regresa l Termina
h: 2 u: 28 Mod1(28,2) e :14	x:28 y:2 l:14

- Se definen las varibles globales y locales
- Se definen los módulos y se realiza el seguimiento de cada una de las instrucciones

Ejemplo

Definición de variables globales A: E	Principal() Inicio x,y,z,w,r: E leer x leer y leer z leer w Si (y<>0 y w<>0) entonces r=Mod1(x,y,z,w) escribe ("Resultado:", r , "/", A) Fin si Termina	Mod1(x: E, y: E, z:E, w:E) Inicio l E l=(x*w)+(z*y) A=y*w Regresa l Termina
A: 28	Si x: 3 A:28 y: 4 z:6 w:7 r=Mod1(3,4,6,7) r:45	x:3 y:4 z:6 w:7 l:(3*7)+(6*4)=45 A:28

- ✘ Se definen las varibles globales y locales
- ✘ Se definen los módulos y se realiza el seguimiento de cada una de las instrucciones

Ejemplo

Definición de variables globales A: E	Principal() Inicio s,t: E A=50 s=2 t=1 A=A+ Modulo1(s+t) Escrib e A Termina	E Modulo1(k: E) Inicio M: E Leer M k=k*A +M Regre sa k Termina
	s: t: A:	Si M=5 A:	k:
	s: 2 t: 1 A: 202	Si M=2 k: A:

- Ahora se hará la prueba del siguiente ejercicio.
- Se sugieren los valores de entrada que deberán utilizarse en los módulos.

Conclusiones

La técnica de programación es muy útil para realizar programas complejos.

El uso de la programación modular permite simplificar y automatizar problemas que en su conjunto son complejos.

El uso de módulos permite el reuso de código.

La verificación de código modular es más sencillo y rápido que la verificación de programas secuenciales complejos.

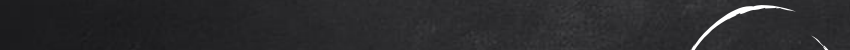
Conclusiones

- La compresión de la programación modular permite realizar la programación de manera más sencilla.

- La programación es un antyecedente importate en la comprensión del paradigma de programación orientada a objetos.

Bibliografía

1. Alcalde Lancharro, E., & García López, M. I. G. U. E. L. (1992). Metodología de la programación. McGraw-Hill, Madrid.
2. Cairó Battistutti, O. , (2006). Metodología de Programación. Ed. Alfaomega.
3. Cairó Battistutti, O. , (2006). Fundamentos de programación. Piensa en C. Ed. Alfaomega



Bibliografía

1. Joyanes Aguilar, L. (2003). Fundamentos de programación: algoritmos y estructura de datos y objetos.
2. Levine, G. (2001). Computación y programación moderna. Perspectiva integral de la información.
3. López Román, L. (2011).Programación estructurada y Orientada a Objetos. Ed. Alfaomega.