

Técnicas de Programación imperativa

Estructura de Datos/Algoritmos I

Depto. de computación-UNRC

Prof. Pablo Castro

Programación Imperativa

Programación Imperativa: Datos + Secuencia de Comandos

Podemos clasificar los lenguajes imperativos según el nivel de abstracción que tienen estos con respecto al lenguaje de máquina

- Alto nivel: Cercano al lenguaje natural (Java, C++, Smalltalk, etc).
- Proveen algunos mecanismos de estructuración de código (C, FORTRAN, etc.)
- Bajo Nivel: Son muy cercanos o iguales al lenguaje máquina (Assembler).

Ejemplo Fortran77

Lee números de un archivo y los suma

```
*Read numbers from input file until it ends, add them up.  
SUM = 0.0  
100 READ(UNIT=IN, FMT=200, END=9999) VALUE  
200 FORMAT(F20.0)  
SUM = SUM + VALUE  
GO TO 100  
9999 WRITE(UNIT=*, FMT=*) 'SUM of values is ', SUM
```

Poco cercano al lenguaje natural

Mecanismos de Estructuración

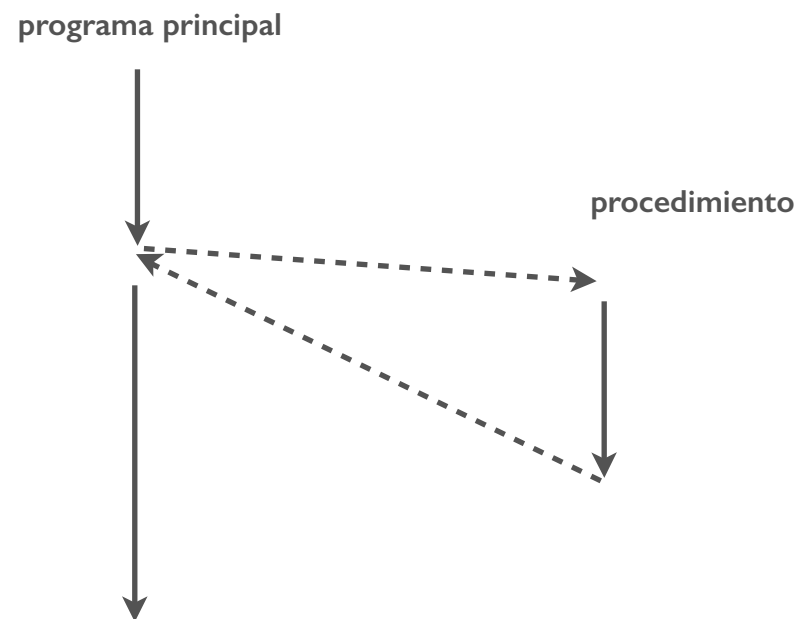
Programas en los lenguajes imperativos iniciales:

Variables
globales + Programa
Principal

- Programas difíciles de entender.
- No existe reuso de código (copy&paste).
- No hay modularización.

Estructuración de programas

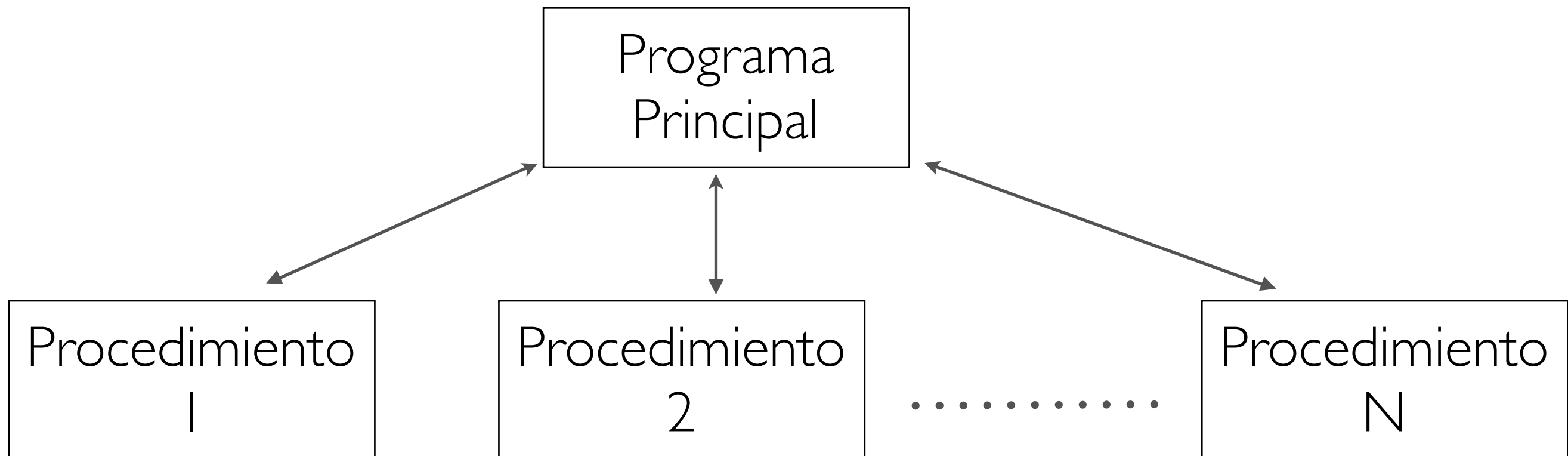
Debido a esto surge la noción de procedimiento:



- Permite reutilizar código.
- Se introduce la noción de variable local (ocultamiento de info).
- El hilo de ejecución no es lineal.

Programación Procedural

- Mecanismo de Reuso: Procedimientos, parámetros, variables locales, etc.
- Beneficios: Programas más fáciles de mantener, reutilización de código, propagación de errores acotada (ocultamiento de información)



EJEMPLO PASCAL

```
program P;
const    maxsize = 80;
type     line = packed array[1..maxsize] of char;
function CountLetters ( words : line) : integer;
var       loop_count : integer; {local variable}
begin
    loop_count := 1;
    while (words[loop_count] <> '.')
    and (loop_count <= maxsize) do
        loop_count := loop_count + 1;
        CountLetter := loop_count - 1
end;
```

Cuenta cantidad de letras de una palabra.

Parámetros

Diversas formas de pasar parámetros:

- Por referencia: Se utiliza el lugar de memoria del parámetro actual.
- Por valor: Se copia el valor del parámetro actual al parámetro formal.

```
Procedure MultOut(var v: Int)
begin
    v := v * v;
    WriteLn(v)
end
```



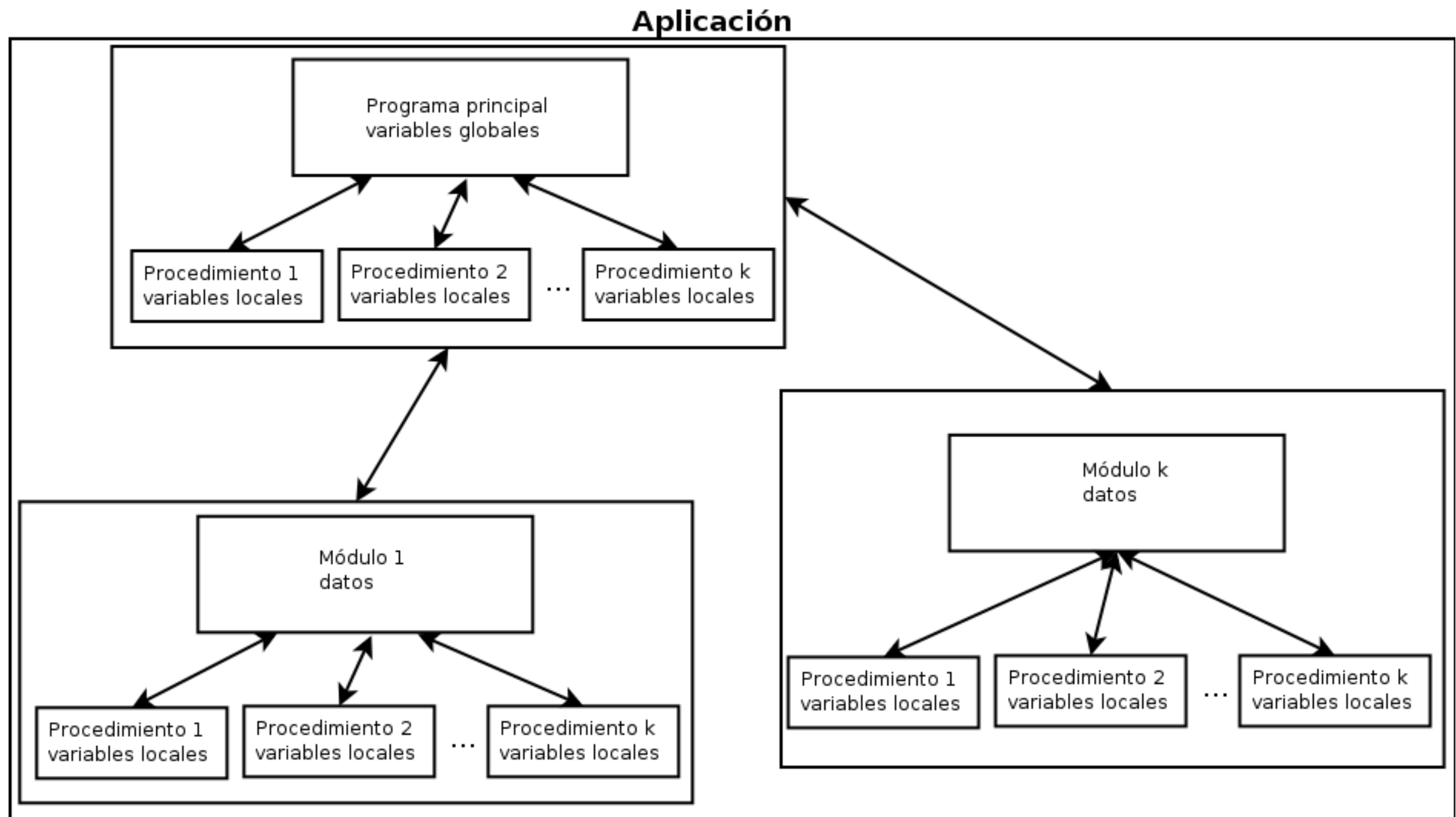
Pasaje por referencia,
cambia el valor de v.

Programación Modular

La programación modular provee una nueva forma de estructurar programas:

- Los procedimientos con funcionalidades comunes y los datos correspondientes se agrupan.
- Un programa se compone de varios módulos que se interconectan entre sí.
- Cada módulo maneja sus propios datos.

Programación modular



Problemas de la programación modular

- No hay ocultamiento de información: los datos de un modulo pueden ser accedidos o cambiados desde otro modulo.
- No hay manejo de múltiples instancias. Por ejemplo, si queremos utilizar dos listas diferentes, en muchos lenguajes necesitaríamos dos módulos diferentes.
- Manejo de memoria explícito, el programador debe ocuparse del manejo de la creación y eliminación de datos.
- Datos separados de las operaciones, por ej., cada operación sobre listas, toma como parámetro la lista a modificar.

Abstracción

La abstracción permite dar soluciones más generales a los problemas. las cuales pueden ser reusadas.

- Abstracción a través de parámetros: Nos abstraemos de los datos por medio de parámetros.
- Abstracción a través de especificación: Se provee una especificación de qué hace el modulo, no cómo se hace. Por ejemplo: perfiles de las operaciones.
- Abstracción a través de datos: podemos introducir nuevos tipos de datos, con sus operaciones. Este tipo de abstracción es la utilizada en POO.