



Estilos de programación

- Programación imperativa
- Programación funcional
- Programación lógica

Programación Imperativa

Programación Imperativa (I)

✎ Nociones básicas:

- Variable con estado (valor modificable)
- Secuencia de cambios de estado
- Procedimientos

✎ Procedimiento:

- Secuencia de actuaciones sobre el estado de ciertas variables para alcanzar unos valores que cumplan unas determinadas condiciones. Tienen nombre y parámetros (de entrada y de salida).
- Los procedimientos colaboran pasando valores de los parámetros de salida de unos a los parámetros de entrada de otros

✎ Estructuras de datos:

- Datos predefinidos (v. lógicos, números, caracteres)
- Esquemas de array, de registro y var. dinámicas

Programación Imperativa (II)

✎ Programa:

Declaraciones de procedimientos
+
rutina principal (secuencia de llamadas a procedimientos)

✎ Ejecución de un programa:

Máquina virtual que sigue las pautas de comportamiento expresadas en las rutinas.

✎ Modelo de cálculo:

Máquina de Turing, autómatas de estados finitos cuyo comportamiento se basa en la transición de estados.

Programación Imperativa (III)

Ejemplo 1:

Procedimiento para calcular el factorial de un número natural en Modula-2

```
PROCEDURE factorial(n: CARDINAL; VAR F: CARDINAL);
BEGIN
  F := 1;
  WHILE n > 0 DO
    F := n * F;
    n := n - 1;
  END
END factorial;
```

Programación Imperativa (IV)

Ejemplo 1:

Diagrama de flujo del procedimiento **factorial(n, F)**

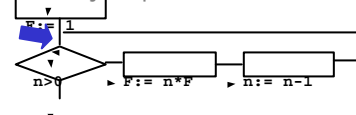


Tabla de valores de las variables en el punto de observación para la llamada **factorial(3, F)**

n	3	2	1	0
F	1	3	6	6

Programación Imperativa (V)

Características destacables:

✍ Se fija completamente el orden en el que se deben realizar las operaciones con ayuda de unos patrones de control del flujo de ejecución (secuencia, alternativa y ciclo) que sirven para construir el esqueleto de las rutinas.

✍ Se pueden fijar puntos de observación en el texto de una rutina y considerar los valores de las variables (estado) cuando el flujo de ejecución pasa por dichos puntos. Estos valores pueden cambiar de un punto a otro y en el mismo punto en momentos distintos de la ejecución.

Programación Declarativa

7

Programación Funcional

Programación Declarativa

8

Programación Funcional (I)

✍ Nociones básicas:

- Función (constructora y definida)
- Aplicación de una función

✍ Función definida:

- Anidamiento de aplicaciones de funciones que establecen los cálculos necesarios para obtener el resultado a partir de los argumentos, con independencia del orden en que se realicen.
- Cada aplicación de una función representa el valor que produce (sólo hay parámetros de entrada).

✍ Estructuras de datos:

- Suma directa y producto cartesiano
- Funciones como argumentos

Programación Declarativa

9

Programación Funcional (II)

✍ Programa:

- Declaraciones de funciones
- +
- expresión principal
- (anidamiento de aplicaciones de funciones evaluables)

✍ Ejecución de un programa:

- Evaluador de expresiones que fija el orden de los cálculos y los almacenamientos temporales necesarios para guardar cálculos parciales.

✍ Modelo de cálculo:

- Reescritura controlada por patrones.

Programación Declarativa

10

Programación Funcional (III)

Ejemplo 2:

Función para calcular el factorial de un número natural en Haskell

```
factorial 0 = 1
factorial (n+1) = (n+1)*(factorial n)
```

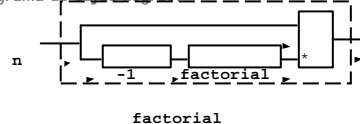
Programación Declarativa

11

Programación Funcional (IV)

Ejemplo 2:

Diagrama de cajas negras:



Pasos de reducción:

```
factorial 3 -> 3 * factorial 2
            -> 3 * (2 * factorial 1)
            -> 3 * (2 * (1 * factorial 0))
            -> 3 * (2 * (1 * 1))
            -> 3 * (2 * 1)
            -> 3 * 2
            -> 6
```

Programación Declarativa

12

Programación Funcional (V)

Características destacables:

- ✍ Se anidan aplicaciones de funciones. No se secuencian.
- ✍ Se utilizan parámetros que toman sus valores en el momento en que se producen las llamadas a las funciones y los mantienen en toda la expresión. No se utilizan variables que cambian de valor.
- ✍ Los cálculos se realizan por paso de valores, resultados de llamadas a funciones, a otras funciones siguiendo una determinada pauta.

Programación Declarativa

13

Programación Lógica

Programación Declarativa

14

Programación Lógica (I)

✍ Nociones básicas:

- Relación
- Satisfacción de una relación
- Variable lógica

✍ Relación:

- Conjunción de relaciones secundarias que establecen la relación principal entre los argumentos, con independencia del orden en que se comprueben.
- Las relaciones colaboran utilizando parámetros comunes que actúan como canales de comunicación.
- La comunicación puede ser bidireccional.

✍ Estructuras de datos:

- Expresiones funcionales construidas con funciones constructoras.

Programación Declarativa

15

Programación Lógica (II)

✍ Programa:

- Declaraciones de relaciones
- + conjunción de relaciones (cuestión)

✍ Ejecución de un programa:

- Evaluador de relaciones que fija el orden de verificación y los almacenamientos temporales necesarios para guardar los cálculos parciales.
- Durante el proceso de verificación de una relación se puede llegar a determinar valores para parámetros desconocidos de la relación.

✍ Modelo de cálculo:

- Resolución SLD con unificación.

Programación Declarativa

16

Programación Lógica (III)

Ejemplo 3:

Función para calcular el factorial de un número natural en Prolog:

```
factorial(0,1).
factorial(N,F):- diferencia(N,1,N1),
                  factorial(N1,F1),
                  producto(N,F1,F).
```

Programación Declarativa

17

Programación Lógica (IV)

Ejemplo 3:

Árbol de resolución para `factorial(3,F)`

```
f(3,F).
d(3,1,N1),f(N1,F1),p(3,F1,F).
... {N1/2}
f(2,F1),p(3,F1,F).
d(2,1,N2),f(N2,F2),p(2,F2,F1),p(3,F1,F).
... {N2/1}
f(1,F2),p(2,F2,F1),p(3,F1,F).
d(1,1,N3),f(N3,F3),p(1,F3,F2),p(2,F2,F1),p(3,F1,F).
... {N3/0}
f(0,F3),p(1,F3,F2),p(2,F2,F1),p(3,F1,F). {F3/1}
p(1,1,F2),p(2,F2,F1),p(3,F1,F). ... {F2/1}
p(2,1,F1),p(3,F1,F). ... {F1/2}
p(3,2,F). ... {F/6}
```

Programación Declarativa

18

Programación Lógica (V)

Características destacables:

✗ Se indican secuencias de llamadas a relaciones cuya validación puede producir la asignación de valores a argumentos desconocidos.

✗ Las llamadas a relaciones no se anidan.

✗ No se utilizan variables que cambian de estado.

✗ Los cálculos se realizan por paso de valores a través de argumentos comunes entre relaciones y por asignación de valores a incógnitas durante el proceso de satisfacción de relaciones.