Facultad de Ingeniería
Universidad Nacional de Jujuy

Desarrollo Sistemático de Programas

Unidad 3

Programación Declarativa

Ing. Carlos A. Afranllie

Introducción

Programación Imperativa (clásica)

- Se la conoce también como operacional
- Se basa en dar órdenes a la computadora.
- Cómo resolver el problema

Introducción

Programación Imperativa (clásica)

- A partir de la definición del problema el programador desarrolla un algoritmo o método para resolverlo.

  - Se entiende el universo como una entidad dinámica (que cambia de estado con el tiempo)

  - El programa es un agente modificador que altera dicho estado.

Introducción

Programación Imperativa (clásica)

- Las computadoras actuales responden al enfoque imperativo

  - Poseen una memoria (que almacena el estado del "universo").

  - Microprocesador que procesa secuencias de órdenes.

Introducción

Programación Declarativa

- Intenta centrarse en la descripción formal del problema (qué queremos hacer)

- No interesa la secuencia de pasos para resolverlo.

- Al eliminar el concepto de secuencia temporal de pasos desaparece el concepto de estado de la máquina, el tiempo y por lo tanto el concepto de asignación (que es la operación imperativa por antonomasia).

Introducción

Programación Declarativa

• Esta ausencia de asignación y de estado de la máquina (y por tanto de memoria) se denomina transparencia referencial.

• No existe el concepto de asignación a:=b, sólo el concepto matemático de igualdad.

• Las variables las entenderemos como incógnitas, cuyo valor puede ser o no conocido, pero nunca modificado.

Paradigmas de Programación

Definiciones:

"Es un modelo básico de diseño y desarrollo de programas"

"Es una colección de modelos conceptuales que juntos modelan el proceso de diseño y determinan, al final, la estructura de un programa"

Paradigmas de Programación

Un paradigma de Programación engloba a ciertos lenguajes que comparten:

Elementos estructurales:

¿con qué se confeccionan los programas?

Elementos metodológicos:

¿cómo se confecciona un programa?

Paradigmas de Programación

Clasificación:

- **Procedimentales u operacionales:** Describen etapa a etapa el modo de construir la solución
  - **Programación Imperativa**
  - **Programación Orientada a Objetos**

- **Declarativos:** Señalan las características que debe tener la solución, sin describir cómo procesarla.
  - **Programación Funcional**
  - **Programación Lógica**

Paradigmas de Programación

Procedimentales u operacionales:

• Describen etapa a etapa el modo de construir la solución.

• Cuentan con estructuras de control.

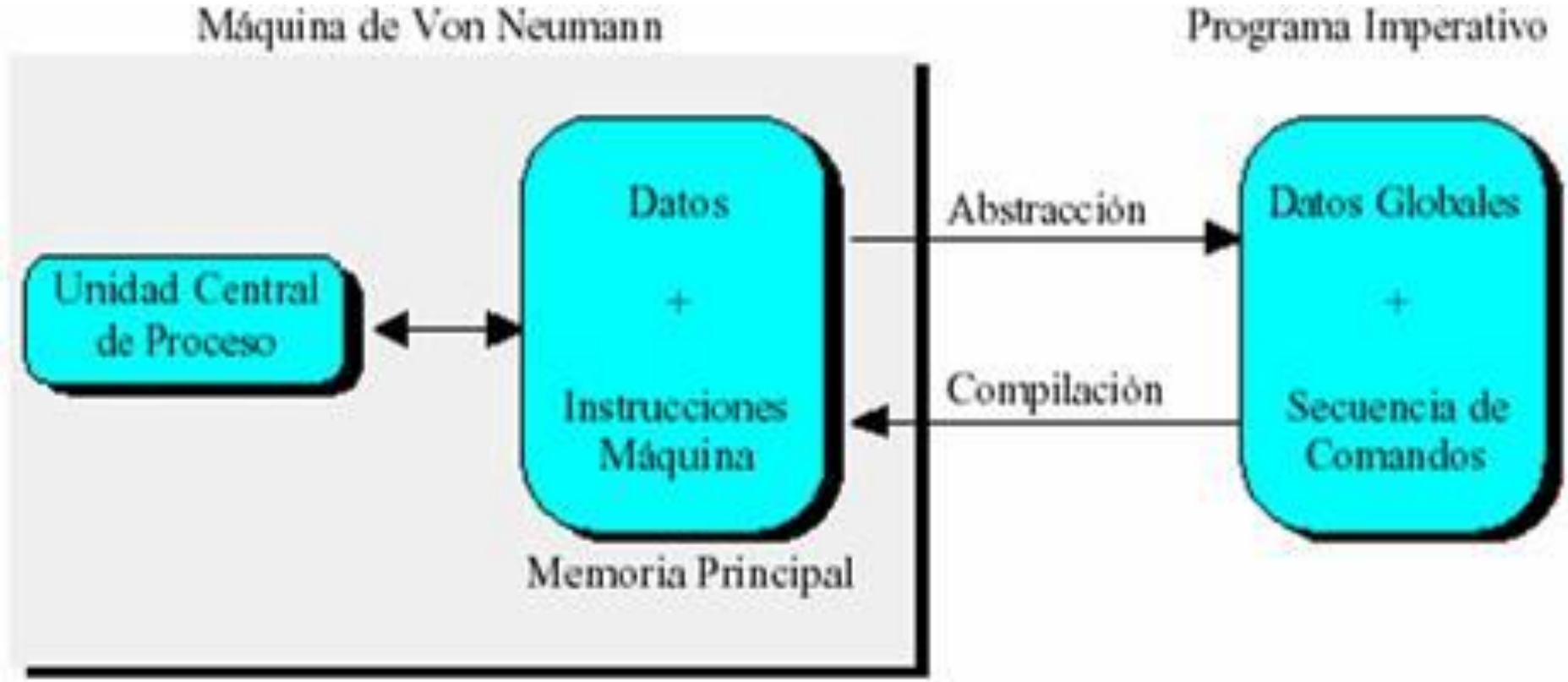
• Cuentan con una semántica que se basa en la transición de estados

• Para obtener el resultado del cómputo cada instrucción que se ejecuta cambia el estado de una máquina hasta que se detiene la ejecución.

Paradigmas de Programación

Programación Imperativa:

Es la más antigua, está basada en la máquina de Von Newman.



Paradigmas de Programación

Programación Imperativa:

- Está orientada a la arquitectura del computador

- No está orientada al modo de pensar humano, sino de la máquina

- El concepto básico es la instrucción o comando

- Poseen asignaciones, saltos condicionales e incondicionales, bucles, etc.

- Afectan o modifican el estado

Paradigmas de Programación

Programación Imperativa:

- Concepto de celda de memoria (variable) para almacenar valores

- Operaciones de asignación

- Repetición

En otras palabras, con los lenguajes imperativos se tiene que expresar el control, es decir **cómo** se va a realizar el proceso de cómputo.

Programación Imperativa:

En los lenguajes imperativos, como su nombre lo dice se dan órdenes como:

Asignación:

- **Contador :=10; /* pone un "10" en la localidad de memoria "Contador" */**
- **x := 0; /* inicia la localidad de memoria "x" con "0" */**

iteración:

```
repeat  /*repite el siguiente bloque de instrucciones hasta que "Contador" tenga el valor "0"*/
{
...
}
until Contador = 0;
```

Programación Imperativa:

Ejemplo Números Primos

```
Programa primos(input, output)
Constante n=50
Variables  i,,j : enteras
           iprimo : lógica
Inicio
           Para i desde 2 hasta n hacer
                      (* ¿ Es primo y ? *)
                      j =2;
                      iprimo = verdaderp
                      Mientras iprimo y (j<=i div 2) hacer
                                 si ( (y mod j) <>0) entonces
                                            j=j+1
                                 sino
                                            iprimo=falso
                                 fin_si
                      fin_mientras
                      (* Si es primo imprime su valor *)
                      si primo entonces
                                 escribir (y)
                      fin_si
           fin_para
Fin.
```

Paradigmas de Programación

Declarativos:

- Señalan las características que debe tener la solución.
- No describen cómo procesarla.
- Poseen una semántica de reducción.
- La entrada se reescribe aplicando las reglas almacenadas en la máquina hasta que ya no se puede aplicar ninguna regla.
- La máquina cuenta con un algoritmo que controla el proceso, por lo que solo es necesario declarar las reglas.

Paradigmas de Programación

Declarativos:

• En los lenguajes declarativos solamente tenemos que expresar **qué** se quiere computar y no **cómo**.

• En los lenguajes declarativos se utilizan variables en el sentido en que lo hace la Matemática.

Paradigmas de Programación

Imperativos vs. Declarativos:

Versión Imperativa:

```
suma(Contador)
X := 0;
repetir
    X := Contador + X
    Contador := Contador - 1;
hasta que Contador = 0;
devolver(X);
```

Versión Declarativa:

```
suma(0) = 0
suma(X) = X + suma(X - 1)
```

Paradigmas de Programación

Imperativos vs. Declarativos:

Instrucción	¿Contador = 0?	Contador	X
Suma(5)	Falso	5	Indefinido
X:=0	Falso	5	0
X := Contador + X	Falso	5	5
Contador := Contador – 1	Falso	4	5
X := Contador + X	Falso	4	9
Contador := Contador – 1	Falso	3	9
X := Contador + X	Falso	3	12
Contador := Contador – 1	Falso	2	12
X := Contador + X	Falso	2	14
Contador := Contador – 1	Falso	1	14
X := Contador + X	Falso	1	15
Contador := Contador – 1	Verdadero	0	15

Paradigmas de Programación

Imperativos vs. Declarativos:

En el caso del ejemplo anterior, suma(5), tenemos el siguiente cómputo:

Suma(5) => 5+Suma(5–1) =>
5+Suma(4) => 5+(4+Suma(4-1)) =>
5+(4+Suma(3)) => 5+(4+(3+Suma(3-1))) =>
5+(4+(3+Suma(2))) => 5+(4+(3+(2+Suma(2-1)))) =>
5+(4+(3+(2+Suma(1)))) => 5+(4+(3+(2+(1+Suma(1-1))))) =>
5+(4+(3+(2+(1+Suma(0))))) => 5+(4+(3+(2+(1+0)))) =>
5+(4+(3+(2+1))) => 5+(4+(3+3)) =>
5+(4+6) => 5+10 =>
15

¿Qué es la “Transparencia Referencial”?

Es reemplazar una subexpresión por otra subexpresión igual que no altera el valor de una expresión.

Sea la expresión x = f(a) entonces podemos sustituir

x+x por f(a) + f(a),

o incluso por 2 * f(a)

"Transparencia Referencial"

- Podemos reemplazar "iguales por iguales".
- Esta propiedad **no** la cumplen los lenguajes imperativos.
- Permite razonar formalmente acerca de los programas
  - Razonamiento Ecuacional
  - Inducción Matemática
- Simplifica el desarrollo y depuración del software
- La programación imperativa carece de la Transparencia referencial, en cambio la declarativa sí la posee.

Paradigmas de Programación

"Transparencia Referencial"

```
Program Prueba;
VAR
  flag:BOOLEAN;

FUNCTION f (n:INTEGER):
INTEGER;
BEGIN
     flag := NOT flag;
     IF flag THEN f := n;
             ELSE f := 2*n
END;
```

```
--- Programa Principal
BEGIN
 flag:=TRUE;
 ...
 WRITE( f(1) );
 WRITE( f(1) );
 ...
 WRITE( f(1) + f(2) );
 WRITE( f(2) + f(1) );
 ...
END.
```

```
<- Escribe 2
<- Escribe 1

<- Escribe 4
<- Escribe 5
```

No se cumplen propiedades matemáticas simples como la propiedad conmutativa, en el ejemplo, f(1) + f(2) no es igual a f(2) + f(1).

Ventajas e Inconvenientes de la Programación Declarativa

• La principal es que son independientes de la máquina y, como se ha comentado, referencialmente transparentes.

• La cantidad de código que debemos escribir es menor, aunque la representación formal de estos problemas puede resultar, en ocasiones, poco evidente.

Ventajas e Inconvenientes de la Programación Declarativa

• Podemos desentendernos del control. Aunque aquí existe una limitación: no podemos hacerlo totalmente.

• Los datos pueden ser tanto de entrada como de salida y se pueden utilizar datos parcialmente construidos, es decir, podemos empezar a ejecutar sin contar con todos los datos.

• Es difícil representar la negación.

Paradigmas de Programación

Imperativos vs. Declarativos:

Imperativos	Declarativos

• Se basan en la máquina Von Newman.
  • Se preocupa por el ¿Cómo?
• No son referencialmente transparentes, es decir, la misma expresión no siempre da los mismos resultados.
• El control depende exclusivamente del programador.
• Se basan en el lenguaje máquina en el que se apoyan, teniendo como instrucción principal la asignación.
• El concepto de variable es un objeto cuyo valor puede cambiar en el tiempo.
	• Se basan en modelos matemáticos.
	  • Se preocupa por el ¿Qué?
	• Intentan ser referencialmente transparentes, es decir, la misma expresión siempre da los mismos resultados.
	• El control lo lleva la máquina
	• Son independientes de la máquina.
	• Las variables son objetos cuyo valor no se conoce, y que, una vez que se le asocia un valor, conserva dicho valor hasta el final.

Aplicaciones de la Programación Declarativa

- Enrutado de aviones para Alitalia
- Configuración de teléfonos móviles - Nokia
- Análisis de ADN y secuencias de proteínas - ICOT / Japón
- Acceso a Bases de Datos en español Software AG España
- Configuración de centrales telefónicas - Ericsson
- Sistema de control de la contaminación del aire - Hungría

Programación Funcional. Lenguajes Funcionales

• En Programación Funcional un programa no es más que una función.

• Se obtiene combinando funciones más simples (que a su vez serán combinaciones de funciones aún más simples).

• Las funciones más simples, que no se pueden descomponer, se denominan primitivas

Programación Funcional. Lenguajes Funcionales

Un programa será pues, algo de la forma:

$$s = f(e)$$

donde *e* es la entrada y *s* la salida en un momento dado.

La ejecución del programa consistirá en evaluar la función *f* para los datos de entrada.

Nótese que una función se evalúa, desde un punto de vista formal, sin necesidad de hacer uso de los conceptos de tiempo ni secuencialidad.

Programación Lógica. Lenguajes Lógicos

• Un programa en Programación Lógica es un conjunto de proposiciones lógicas (que se asumen como ciertas) expresadas mediante alguna notación.

• La ejecución en este caso consistirá en comprobar la veracidad (o falsedad) de un aserto (afirmación), combinando de algún modo las proposiciones dadas inicialmente.

Programación Lógica. Lenguajes Lógicos

De esta manera surgen dos posibilidades:

• ¿La salida *s* está asociada a la entrada *e*? (Nótese que en este caso una entrada puede ir asociada a múltiples salidas).

• ¿Cuáles son los valores de *s* relacionados con un valor de *e*?

Programación Lógica. Lenguajes Lógicos

Para poder modelizar matemáticamente estas relaciones se define la función:

$$R : E\,x\,S \rightarrow \{0\ ,1\}$$

donde *E* es el dominio de las entradas y *S* el dominio de las salidas.

Este tipo de función matemática se llama precisamente relación, puesto que enlaza parejas de elementos de ambos conjuntos.

Diremos que $a \in E$ está relacionado con $b \in S$ si y sólo si *R(a, b) = 1* (también se suele escribir *aRb*).