Tema 2: Notación para la definición y el diseño de algoritmos.

- Duración: 16 horas
- Índice:

Etapas en la resolución de un problema

1. Notación para la definición de problemas
   - Variables y tipos de datos
   - Notación para la especificación de algoritmos
2. Notación para el diseño de algoritmos
   - Asignación
   - Composición secuencial
   - Análisis por casos

Objetivos

1. Notación para la definición de problemas

* Aprender a **definir** correctamente problemas simples, entendiendo la necesidad de **especificar** adecuadamente el comportamiento del algoritmo
* Conocer los **tipos de datos** elementales, las tuplas y los vectores
* Aprender la **notación** para la definición de problemas

Objetivos

2. Notación para el diseño de algoritmos

* Definir las instrucciones básicas **asignación**, **composición secuencial** y **condicional** de la notación algorítmica elegida para el diseño de algoritmos. La semántica de dicha notación será la de transformación de estados.
* Explicitar el uso de **estrategias** de serialización y de análisis por casos en el diseño de algoritmos

3

Bibliografía

♦**"Curso de Programación"**. Castro, J., Cucker, F., Messeguer, X., et al. McGraw-Hill, 2ª ed., 1995.

♦"Program Derivation". Dromey R.G. Addison-Wesley, 1989.

4

Hemos visto que ...

Para establecer **QUÉ** problema se quiere resolver -**definirlo**- resulta necesario determinar:

* sus **DATOS** -objetos sobre los que se aplica la acción-
* sus **RESULTADOS**
* el **EFECTO** de la ejecución de la acción ó **especificación**

5

1. Notación para la definición de problemas

Para establecer **QUÉ** problema se quiere resolver -**definirlo**- resulta necesario determinar:

* sus **DATOS** -objetos sobre los que se aplica la acción-
* sus **RESULTADOS**

1.-Definir sus informaciones relevantes Notación para la definición de sus informaciones relevantes

* el **EFECTO** de la ejecución de la acción ó **especificación**

2.-Especificarlo como una transformación de estados Notación para su especificación

Ejemplo del problema de multiplicar dos números enteros positivos

- **DATOS:** multiplicando=A y multiplicador=B
- **RESULTADOS:** producto

algoritmo multiplicación_á_la_russe
(DATOS multiplicando, multiplicador **:** entero**;**
RESULTADOS producto **:** entero**)**

1.-Notación para la definición de sus informaciones relevantes

A: Designar cada una de ellas con un nombre que las identifique **unívocamente**; i.e.

- DEFINIR **VARIABLES DATO y RESULTADO**

B: Clasificar cada una de ellas según el conjunto de valores que puede tomar y las operaciones que se pueden realizar sobre ellas ; i.e.

- DEFINIR EL **TIPO** DE LAS VARIABLES

C: Añadir **cabecera**

7

2.-Notación para su especificación: asertos ó predicados

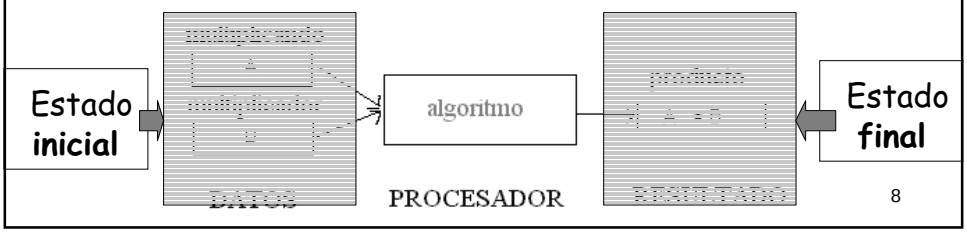
- **ESPECIFICACIÓN, como transformación de estados:**

✓Descripción del Estado inicial, antes de ejecutarla:

{multiplicando=A y multiplicador=B y multiplicando≥0 y multiplicador≥ 0}

✓ Descripción del Estado final, tras ejecutarla:

$\{A \times B = \sum_{i=1..A} B = producto\}$



8

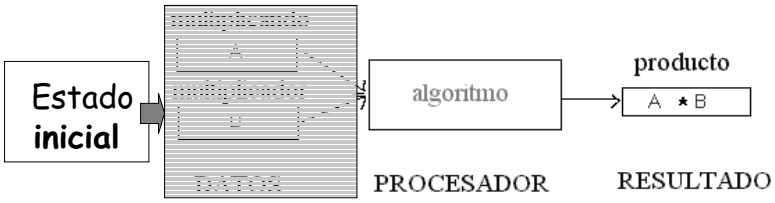
2.-Notación para su especificación: asertos ó predicados

✓Descripción del Estado inicial, antes de ejecutarla:

{multiplicando=A y multiplicador=B y multiplicando≥0 y multiplicador≥ 0}

➧ **PRECONDICIÓN** DEL PROBLEMA, **P**

Expresa las condiciones y relaciones que cumplen los **DATOS** y que definen el conjunto de **ESTADOS INICIALES** del problema; i.e.



9

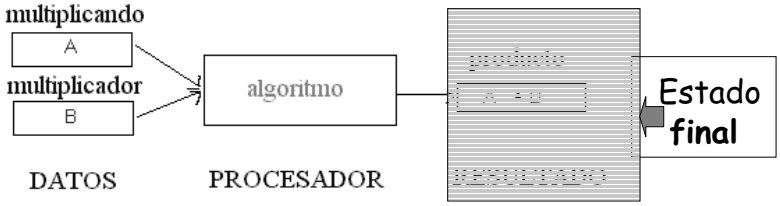
2.-Notación para su especificación: asertos ó predicados

✓Descripción del Estado final, tras ejecutarla:

{$A \times B = \sum_{i=1..A} B$=producto}

➧ **POSTCONDICIÓN** DEL PROBLEMA, **Q**

Expresa las relaciones entre los **DATOS** proporcionados y los **RESULTADOS** esperados y que definen el conjunto de **ESTADOS FINALES**; i.e.



10

Resumen

algoritmo multiplicación_á_la_russe
(DATOS multiplicando, multiplicador **:** entero**;**
RESULTADOS producto **:** entero**)**

P={multiplicando=A ∧ multiplicador=B ∧ multiplicando≥0 ∧
multiplicador≥0**}**

Q= {$A*B=\sum_{i=1..A} B$=producto**}**

11

Objetivo de la etapa de definición y su notación formal

Los requerimientos de la Programación ☞

- Ser capaces de formular el algoritmo que resuelve un problema
- Ser capaces de razonar acerca de la **corrección** y conveniencia del algoritmo propuesto

Etapa de Definición de un algoritmo/problema:

establecer **QUÉ** hace el algoritmo, ...
antes de establecer **CÓMO** lo hace

Especificación formal: expresar la ejecución del algoritmo como una transformación de estados, **P** y **Q**

12

Objetivo de la etapa de definición y su notación formal

Los requerimientos de la Programación ☞

Etapa de Definición de un algoritmo/problema:

☞

Interpretación de la corrección de un algoritmo

un algoritmo S es correcto SII

*** comienza su ejecución en un estado que describe P**

***Y termina su ejecución en un estado que describe Q**

P S Q

13

Ejemplo: definir el problema de la división de dos enteros

algoritmo divisiónZ
(**DATOS** num1, num2: entero;
RESULTADOS q,r: entero)

P= {num1≥0 ∧ num2>0}

Q= {num1 = num2 * q + r ∧ 0≤r<num2 }

☞

P divisiónZ **Q**

14

1.1. Variables y tipos de datos

¿Qué es UN TIPO de datos?

- Conjunto de **valores**

+

- Conjunto de **operaciones** permitidas sobre ellos

15

1.1. Variables y tipos de datos

¿Qué es UNA VARIABLE de un tipo de datos?

- Una variable de un tipo de datos es una **instancia de éste**, tal que únicamente podrá ser manipulada a través de las operaciones definidas sobre el tipo
- Mecanismo de **declaración de variables**

16

1.1. Variables y tipos de datos

TIPOS BÁSICOS o elementales

- **Entero**:
  - Valores: ..-2, -1, 0, 1, 2...
  - Operaciones: aritméticas y relacionales
- **Real**:
  - Valores: representación exponencial o decimal
  - Operaciones: aritméticas y relacionales

17

1.1. Variables y tipos de datos

TIPOS BÁSICOS o elementales

- **Carácter**:
  - Valores: letras minúsculas, mayúsculas, cifras y signos especiales ('.','¿','?',...)
  - Operaciones: relacionales
- **Lógico**:
  - Valores: cierto y falso
  - Operaciones: and, or, implicación,...

18

Ejemplo: dados dos números enteros estrictamente positivos, siendo el primero el doble del segundo, calcular su suma

algoritmo suma

(**DATOS** a, b: entero;

RESULTADOS resul: entero**)**

P={....} suma **Q={.....}**

19

Ejemplo: intercambiar los valores de dos variables enteras

Atención a la especificación cuando los DATOS son ,también, RESULTADOS

algoritmo intercambio

(**DATOS** x, y: entero;

RESULTADOS x, y: entero**)**

P={....} intercambio **Q={.....}**

20

Ejemplos de definición propuestos

- Copiar el valor de una variable entera en otra
- Indicar si un número entero es cuadrado perfecto
- Dado un número real cualquiera *x*, calcula el mayor número entero *n* tal que *n* es menor o igual que *x*. Por ejemplo, si *x*=3.3 entonces *n*=3 y si *x*=-3.3 entonces *n*=-4
- Dado un número de segundos inferior a un millón, *n*, calcular su equivalente en días, horas, minutos y segundos

21

1.1. Variables y tipos de datos

Ejemplo a: considérese una medida de longitud representada por tres números enteros: metros, centímetros y milímetros. Definir una algoritmo que dada una medida de longitud la incremente en un milímetro

¿?

Ejemplo b: sumar dos números complejos

¿?

22

1.1. Variables y tipos de datos

☞ **En los ejemplos a y b se dispone del mismo "tipo" de información, medida o complejo**

- Se define en función de otras informaciones
- Puede ser tratada como un todo (complejo) o por partes (parte real e imaginaria)

23

1.1. Variables y tipos de datos

TIPOS COMPUESTOS

- **Tupla**:

  Valores: tipo **HETEROGÉNEO**

  Notación:

```
tipo
   nombre_tipo = tupla
         nombre_campo1:tipo_campo1;
         nombre_campo2:tipo_campo2;
         ...
   ftupla
ftipo
```

24

1.1. Variables y tipos de datos

Ejemplo a: **tipo**

medida = **tupla**

metro: entero;
centros: entero;
miltros: entero;

ftupla

ftipo

algoritmo suma1mil (**DATOS** md: medida;
RESULTADOS md:medida**)**

25

1.1. Variables y tipos de datos

Ejemplo b: **tipo**

complejo = **tupla**

preal: real;
pimag: real;

ftupla

ftipo

algoritmo sumaC (**DATOS** num1,num2:complejo;
RESULTADOS sum: complejo**)**

26

1.1. Variables y tipos de datos

Definición del ejemplo a:

```
algoritmo suma1mil (DATOS md: medida;
                    RESULTADOS md:medida)
  P={md=MD}
tipo
   medida  =  tupla
         metro: entero;
         centros: entero;
         miltros: entero;
   ftupla
ftipo
```


1.1. Variables y tipos de datos

TIPOS COMPUESTOS

- **Tupla:**
  - Valores: tipo **HETEROGÉNEO**
  - Operaciones: **ACCESO** a un **campo** de una tupla

nombre_variable. nombre_campo

1.1. Variable... datos

¿Cómo **expresar** el valor de los metros, centímetros y milímetros de md ó MD?

Definición del ejemplo a:

algoritmo suma1mil (**DATOS** md: medida;
RESULTADOS md:medida)

P={md=MD ∧ MD.metro≥**0** ∧ **0**≤MD.centros≤**99** ∧ **0**≤MD.miltros≤**9}**

tipo
  medida **= tupla**
    metro**:** entero**;** — **de 0 en adelante**
    centros**:** entero**;** — **de 0 a 99**
    miltros**:** entero**;** — **de 0 a 9**
  ftupla
ftipo

29

1.1. Variables y tipos de datos

Definición del ejemplo a:

algoritmo suma1mil (**DATOS** md: medida;
RESULTADOS md:medida)

P={md=MD∧

MD.metro≥**0** ∧ **0**≤MD.centros≤**99** ∧**0**≤MD.miltros≤**9}**

¿ Q= { } ?

Si MD.miltros ≤ 9
entonces sumar uno a MD.miltros
y dejar el resto tal cual
sino (≥9) .....

30

1.1. Variables y tipos de datos

Acceso para modificación

$$Q=\{\ (MD.miltros+1 \le 9 \wedge md.miltros = MD.miltros+1 \wedge md.centros = MD.centros \wedge md.metro = MD.metro)$$

Acceso para consulta

$$\vee\ (MD.miltros+1=10 \wedge md.miltros = 0 \wedge ((MD.centros+1 \le 99 \wedge md.centros = MD.centros+1 \wedge md.metro = MD.metro) \vee (MD.centros+1 = 100 \wedge md.centros = 0 \wedge md.metro = MD.metro+1)))\}$$

1.1. Variables y tipos de datos

TIPOS COMPUESTOS

I Tupla: en PASCAL "record":

Valores: tipo **HETEROGÉNEO**

Notación:

```
type
   nombre_tipo = record
        nombre_campo1:tipo_campo1;
        nombre_campo2:tipo_campo2;
   end;
```

1.1. Variables y tipos de datos

Ejemplo: dada la secuencia de números (4,7,10,13,16,19)

a) Calcular la suma de sus valores

b) Comprobar si el tercer término es par

c) Enumerarla inversamente

d) Intercambiar su elemento 6º con el 7º

¿?

33

1.1. Variables y tipos de datos

☞ En todos los apartados del ejemplo anterior se dispone del mismo "tipo" de información, **secuencia o lista** de elementos **del mismo tipo (homogéneos)**

- Acceso secuencial
- Acceso directo, por posición

34

1.1. Variables y tipos de datos

TIPOS COMPUESTOS

I Vector:

Valores: tipo **Homogéneo**

Notación:

```
tipo
   nombre_vector=vector[b1..bn]de tipo_base;
ftipo
```

35

1.1. Variables y tipos de datos

```
tipo
   lista  =  vector[1..6]de entero;
ftipo
```

a):

algoritmo sumaV (**DATOS** l6:lista;
RESULTADOS suma: entero)

b):

algoritmo consulta (**DATOS** l6:lista;tercero:entero;
RESULTADOS espar: lógico)

36

1.1. Variables y tipos de datos

Definición del apartado a):

algoritmo sumaV (**DATOS** l6:lista;
RESULTADOS suma: entero)

P= {}

¿Cómo consultar el valor de los elementos de l6?

¿ Q= {suma = $\sum_{i=1..6}$ ....**} ?**

37

1.1. Variables y tipos de datos

Definición del apartado b):

algoritmo consulta (**DATOS** l6:lista; tercero:entero;
RESULTADOS espar: lógico)

P= {1≤tercero≤6**}**

¿Cómo consultar el valor del tercer elemento de l6?

¿ Q= {espar= (.... **mod** 2=0)**} ?**

38

1.1. Variables y tipos de datos

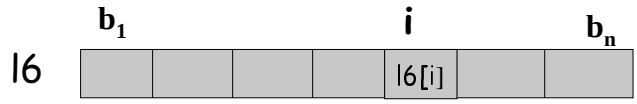
TIPOS COMPUESTOS

I Vector:

Valores: tipo **HOMOGÉNEO**

Operaciones: **ACCESO** a un elemento en posición ***i***, $\mathbf{b_1} \leq \boldsymbol{i} \leq \mathbf{b_n}$

nombre_variable[**i**]



39

1.1. Variables y tipos de datos

Definición del apartado a):

```
tipo
   lista = vector[1..6]de entero;
ftipo
algoritmo sumaV (DATOS l6:lista;
                 RESULTADOS suma: entero)
```

P= {}

Q= {suma = $\sum_{i=1..6}$ l6[**i**]**}**

40

1.1. Variables y tipos de datos

Definición del apartado b):

tipo

lista = **vector[**1..6**]de** entero;

ftipo

algoritmo consulta(**DATOS** l6:lista; tercero:entero;

RESULTADOS espar: lógico**)**

P= {1≤tercero≤6**}**

Q= {espar= (l6[**tercero**] **mod** 2=0)**}**

¿Cómo generalizarías la definición de este problema?

41

1.1. Variables y tipos de datos

TIPOS COMPUESTOS

I Vector: en PASCAL "array":

```
const Nmax=N;
```

```
type
nombre_vector= array [1..Nmax] of tbase;
```

```
var n:integer;
```

42

1.1. Variables y tipos de datos

TIPOS COMPUESTOS- Comparación

- Para **"crear"** una tupla o un vector se declara la variable correspondiente del tipo
- Los elementos de una tupla son heterogéneos y los de un vector homogéneos
- Un elemento de una tupla se denota por su nombre y un elemento de un vector por su posición

43

Hemos visto que ....

- **ESPECIFICACIÓN, como transformación de estados:**

✓Descripción del Estado inicial, antes de ejecutarla

➘ **PRECONDICIÓN** DEL PROBLEMA, **P**

✓ Descripción del Estado final, tras ejecutarla

➘ **POSTCONDICIÓN** DEL PROBLEMA, **Q**

2.-Notación: asertos ó predicados

44

2.-Notación: asertos ó predicados

- ESPECIFICACIÓN, como transformación de estados:

Interpretación de la corrección de un algoritmo

un algoritmo **S** es correcto SII

* comienza su ejecución en un estado que describe P

*Y termina su ejecución en un estado que describe Q

P S Q

45

Interpretación de la corrección de un algoritmo

Ejemplo: sea la definición

algoritmo S (**DATOS** x: entero;**RESULTADOS** x: entero)

P= {x=X}

Q=

donde **Q** puede substituirse por cualquiera de estos 3 predicados: {x=|X|}, {|x|=X }, {|x|=|X|}

¿En qué caso(s) se obtiene una definición correcta para un algoritmo **S** que calcula el valor absoluto de x ?

46

Interpretación de la corrección de un algoritmo

Ejemplo: dados dos números enteros a y b, se quiere calcular su suma atendiendo a ciertas condiciones; esto es,

- si ambos números son positivos, entonces el resultado esperado es su suma aritmética
- si alguno es negativo o cero, el resultado esperado es cero

¿**Es <u>correcta</u>** la siguiente definición del problema

algoritmo suma (**DATOS** a,b: entero;

RESULTADOS result: entero**)**

P= {}

Q= {(result=a+b) $\vee$ (result=0)**} ?**

47

Interpretación de la corrección de un algoritmo

Ejemplo: dada una secuencia de caracteres ('pepito') se desea sustituir en ella todas las apariciones del carácter x ('e') por y ('a'). ¿Define correctamente la solución del problema enunciado **Q={**ningún elemento de la secuencia es igual a x **}**

NO, porque existen otros enunciados que satisfacen la misma **Q sin necesidad de haber <u>substituido</u> x por y** ; i.e.

- substituir **todos** los elementos de una secuencia de caracteres por el carácter y (siendo **y#x)**
- eliminar todos los elementos de una secuencia de caracteres que sean **iguales a x**

48

1.2. Notación para la especificación

¿ASERTOS O PREDICADOS?

porque **un predicado define el conjunto de estados que lo satisfacen**

i.e. **P= {x<y}** define C_E $E_1=\{x=3, y=7\}$ $E_2=\{x=0, y=22\}$ .....

- **I** E_i asigna valores a todas las variables de **P**
- **I** Al substituir los valores de E_i en **P** ésta es **cierta**

49

1.2. Notación para la especificación

❑ **un predicado define el conjunto de estados que lo satisfacen:**

Tautología: fórmula satisfecha por cualquier estado

➨ **V** define el conjunto de todos los estados

Contradicción: fórmula que no satisface ningún estado

➨ **F** define el conjunto vacío de estados

50

❑ un predicado define el conjunto de estados que lo satisfacen:

¬F1 define el conjunto de estados formado por aquellos estados que **no** satisfacen F1
F1∧F2 define el conjunto de estados formado por la **intersección** del conjunto de estados que definen F1 y F2
F1∨F2 define el conjunto de estados formado por la **unión** del conjunto de estados que definen la F1 y F2
F1↔F2 es cierta cuando **F1 y F2** definen el **mismo** conjunto de estados (**equivalencias**)

51

❑ un predicado define el conjunto de estados que lo satisfacen:

❑ **F1→F2** es cierta cuando el conjunto de estados que define **F2 contiene o es igual** al conjunto de estados que define F1

F1	F2	**F1→F2**
F	F	V
F	V	V
V	**F**	**F**
V	V	V

52

❑ **un predicado define el conjunto de estados que lo satisfacen:**

Ejemplo: interpretar en términos de conjuntos de estados las fórmulas:

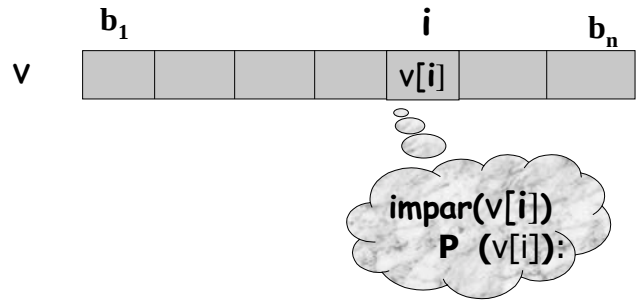
- $x>0 \rightarrow x\geq 0$
- $x\geq 0 \rightarrow x>0$
- $P \wedge Q \rightarrow P$
- $P \wedge Q \rightarrow Q$
- $P \rightarrow P \vee Q$
- $Q \rightarrow P \vee Q$

53

1.2. Notación para la especificación

❑ **un predicado define el conjunto de estados que lo satisfacen:**

v:**vector[b_1..b_n]de** tipo_base

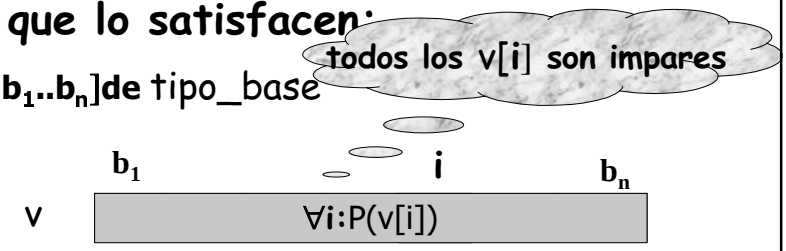


54

1.2. Notación para la especificación

- **un predicado define el conjunto de estados que lo satisfacen:**

v:**vector[b_1..b_n]de** tipo_base



$\forall i: b_1 \le i \le b_n : P(v[i])$: **todos** los elementos de v cumplen la propiedad P

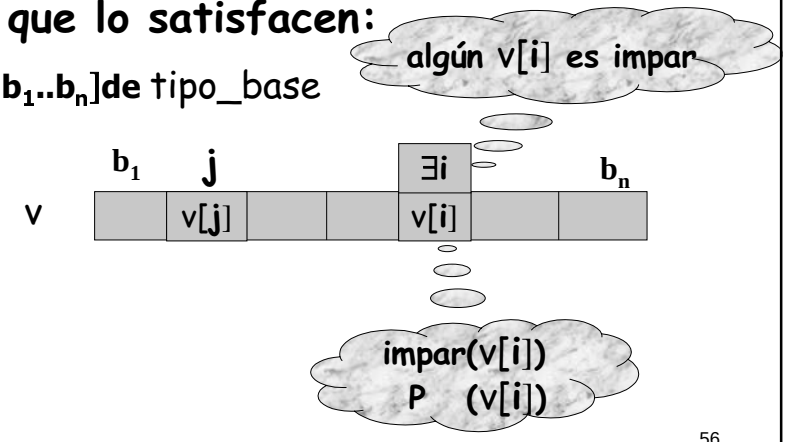
$\forall i: Rango(i) : P(v[i]) \Leftrightarrow \forall i (Rango(i) \rightarrow P(v[i]))$

55

1.2. Notación para la especificación

- **un predicado define el conjunto de estados que lo satisfacen:**

v:**vector[b_1..b_n]de** tipo_base

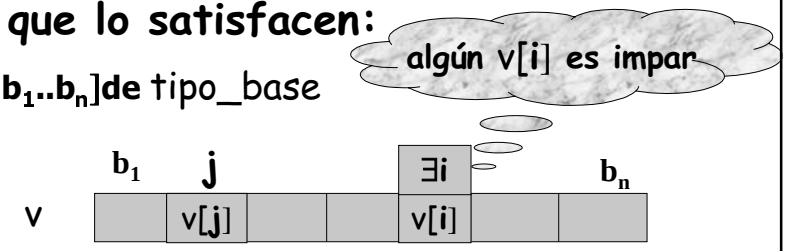


56

1.2. Notación para la especificación

- **un predicado define el conjunto de estados que lo satisfacen:**

v:**vector[b_1..b_n]de** tipo_base



$\exists i: b_1 \leq i \leq b_n : P(v[i])$: **al menos UN** (algún) elemento de v cumple la propiedad P

$\exists i: Rango(i): P(v[i]) \Leftrightarrow \exists i (Rango(i) \wedge P(v[i]))$

57

RESUMEN PARA EL ALUMNO

LENGUAJE DE PREDICADOS

Predicado (fórmula atómica):

relación entre objetos a la que siempre es posible asignarle un valor cierto o falso, en función de los valores involucrados

Ejemplo: "el elemento *x* es menor que el elemento *y*" se expresa con el predicado {x < y}

58

RESUMEN PARA EL ALUMNO

LENGUAJE DE PREDICADOS

- **Sintaxis**
  - **Un predicado es una fórmula bien formada (fbf)**
  - **Si F1 y F2 son fbf, también son fbf F1∧ F2, F1∨ F2, ¬F1, F1→ F2, F1↔ F2**

Ejemplo: "el real *x* es la raíz cuadrada positiva del real *y*" se expresa con el predicado $\{x^2=y \wedge x\geq 0\}$

59

RESUMEN PARA EL ALUMNO

LENGUAJE DE PREDICADOS

- **Semántica-Tablas de verdad**

F1	F2	F1∧F2	F1∨F2	¬F1	F1↔F2	**F1→F2**
F	F	F	F	V	V	V
F	V	F	V	V	F	V
V	**F**	F	V	F	F	**F**
V	V	V	V	F	V	V

60

RESUMEN PARA EL ALUMNO

❑ Algunas leyes de equivalencia

1. $\neg(\neg P)=P$
2. $P\wedge Q=Q\wedge P$; $P\vee Q=Q\vee P$ Conmutativa
3. $(P\wedge Q)\wedge R=P\wedge(Q\wedge R)$; $(P\vee Q)\vee R=(P\vee(Q\vee R)$ Asociativa
4. $P\vee(Q\wedge R)=(P\vee Q)\wedge(P\vee R)$; $P\wedge(Q\vee R)=(P\wedge Q)\vee(P\wedge R)$ Distributiva
5. $\neg(P\wedge Q)=\neg P\vee\neg Q$; $\neg(P\vee Q)=\neg P\wedge\neg Q$ L. de de Morgan
6. $P\wedge\neg P=F$; $P\wedge V=P$; $P\wedge F=F$
7. $P\vee\neg P=V$; $P\vee V=V$; $P\vee F=P$
8. $P\wedge P=P\vee P=P$

61

Ejemplos de definición propuestos

- Calcular el máximo de dos enteros
- Indicar si un número entero *x* es múltiplo de *y*
- Sea el vector *b*:vector[1..Nmax]de entero. Sea *n* el número de elementos del vector y sean *j*, y *k* dos enteros tales que $1\leq j\leq k\leq n\leq Nmax$. Traducir los siguientes predicados:
  - Todos los elementos de *b* entre el *j* y el *k* son cero
  - Ningún elemento de *b* entre el *j* y el *k* es cero
  - Alguno de los elementos de *b* entre el *j* y el *k* es cero
  - Todos los ceros de *b* están entre *j* y *k*
  - Los elementos de *b* están ordenados ascendentemente
  - La suma de los elementos de *b* está almacenada en *s*
  - Determinar si el elemento *x* está en *b*

62

Ejemplo : calcular el máximo de dos enteros

algoritmo máximo (**DATOS**
RESULTADOS **)**

63

Ejemplo 12: indicar si un número entero *x* es múltiplo de *y*

algoritmo múltiplo (**DATOS**
RESULTADOS **)**

64

Ejemplo: sea el vector *b*:**vector[1..Nmax]de entero**. Sea *n* el número de elementos del vector y sean *j*, y *k* dos enteros tales que $1 \leq j \leq k \leq n \leq$ **Nmax**. Traducir los siguientes predicados:

- Todos los elementos de *b* entre el *j* y el *k* son cero

- Ningún elemento de *b* entre el *j* y el *k* es cero

65

Ejemplo 13 ...

- Alguno de los elementos de *b* entre el *j* y el *k* es cero

- Todos los ceros de *b* están entre *j* y *k*

66

Ejemplo 13 .....

- Los elementos de *b* están ordenados ascendentemente
- La suma de los elementos de *b* está almacenada en *s*
- Determinar si el elemento *x* está en *b*

67

1.2. Notación para la especificación

Ejemplos propuestos:

- Dado un vector *v*:vector[1..Nmax]de entero duplicar el valor de todos sus elementos desde el central hasta el último
- Dado un vector *v*:vector[1..Nmax]de entero indicar si todos sus elementos son pares

68

Uso de funciones en la especificación

Dado un vector v:**vector[1..Nmax]de carácter**, contar el número de apariciones del carácter 'a' en v

algoritmo contar (**DATOS** v:vector[1..**Nmax**]de carácter;
n:entero; la_a: carácter;
RESULTADOS cuántos: natural)

P= {1≤n≤Nmax ∧ la_a='a'**}**

¿ Q={cuántos =(Σi:1≤i≤n:.....**}?**

¿Cómo contar "uno más" cada vez que v[i]='a'?

igual: Carácter x Carácter→ {0,1} /si x,y∈ Carácter
igual(x,y)=1 si x=y
igual(x,y)=0 si x≠y

69

Interpretación de la corrección de un algoritmo

Ejemplo: dado un vector v de enteros obtener en la variable max su máximo

¿**Es correcta** la siguiente definición del problema

algoritmo máximo (**DATOS** v: vector[1..**Nmax**]de entero;
n:entero;
RESULTADOS max: entero)

P= {1≤n≤Nmax}

Q= { ∀i:1≤i≤n:v[i]≤max **} ?**

70

Ejemplos propuestos:

- Dado un número entero n, se quiere determinar si es o no impar, ¿**es <u>correcta</u>** la siguiente definición del problema

algoritmo esimpar (**DATOS** n**:** entero**; RESULTADOS** impar**:** lógico**)**

P= {$n \neq 0$}

Q= {$(n \bmod 2 \neq 0 \wedge \text{impar}) \vee \neg\text{impar}$}**?**

- Dado un vector v de enteros obtener en la variable pos <u>la posición</u> de su máximo

- Dado un vector v de enteros obtener en la variable pos <u>la posición</u> de su <u>primer</u> máximo

71