1.2. Predicados para la especificación de algoritmos

1.2.1. ¿Qué es un predicado?

Un **predicado**, **aserto** o **aserción** es una afirmación que puede ser *cierta* o *falsa*. En el diseño de algoritmos se escriben predicados para especificar su comportamiento mediante la descripción de las relaciones que debieran satisfacer los valores de sus datos. Los predicados se escriben a modo de comentarios.

```
/* Pre: n ≥ 0 */
/* Post: factorial(n) = (Πα ∈ [1, n].α) */
void factorial (const int n) {
    int i = 0;
    r = 1;
    /* i = 0 ∧ r = (Πα ∈ [1, i].α) */
    while ( i! = n ) {
        /* i ≥ 0 ∧ i < n ∧ r = (Πα ∈ [1, i].α) */
        i = i + 1;
        r = i*r;
        /* i ≥ 0 ∧ i ≤ n ∧ r = (Πα ∈ [1, i].α) */
    }
    /* r = (Πα ∈ [1, n].α) */
    return r;
}
```

Algoritmo 1.3: Función `factorial(n)`

La función anterior ha sido especificada mediante un par de predicados, sus pre y postcondiciones. Además se han escrito en diferentes puntos de su código cuatro predicados o aserciones adicionales que lo documentan. Más adelante, en esta asignatura, se comprobará la importancia de estos predicados o aserciones para razonar sobre la corrección del código diseñado.

1.2.2. Escritura matemática de predicados para la especificación de algoritmos

Para formular predicados (precondiciones, postcondiciones y aserciones insertas en el código) admitiremos un lenguaje más rico que el disponible habitualmente en los lenguajes de programación para construir expresiones booleanas y utilizaremos unos símbolos, que presentan alguna diferencia, para denotar los operadores lógicos y de relación.

Esta primera tabla muestra cómo escribir **predicados atómicos** como expresiones en la que intervienen elementos del programa y que, al ser evaluadas, proporcionan un resultado de tipo *bool*:

SON PREDICADOS ATÓMICOS	EJEMPLOS DE PREDICADOS ATÓMICOS
Un dato **constante** de tipo *bool*	$cierto$
	$falso$
Un dato **variable** de tipo *bool*	$existe$
	$ocupado[i]$
Una **expresión de relación** (trabajaremos con estos operadores de relación: $<, \leq, >, \geq, =$ y $\neq$)	$x^2 = y$
	$x \geq y$
Una **invocación a un algoritmo** que devuelva un valor de tipo *bool*	$estaOrdenada(laTabla)$
	$estaAlmacenado(dato, laTabla)$

Un **predicado** podrá limitarse a un predicado atómico, podrá construirse aplicando operaciones lógicas entre predicados ($\neg$, $\wedge$, $\vee$, $\rightarrow$ y $=$) o podrá escribirse utilizando los cuantificadores lógicos $\forall$ y $\exists$.

SON PREDICADOS	**EJEMPLOS DE PREDICADOS**
Cualquier **predicado atómico**	$cierto$
	$ocupado[i]$
	$x \geq 0$
	$estaOrdenada(LaTabla)$
Una **negación lógica** ($\neg P$)	$\neg(x^2 = y)$
de un predicado P	$\neg ocupado[i]$
Una **conjunción lógica** ($P \wedge Q$)	$\neg(x^2 = y) \wedge ocupado[i]$
de dos predicados P y Q	$(x \geq 0) \wedge \neg(x^2 = y) \wedge ocupado[i]$
Una **disyunción lógica** ($P \vee Q$)	$\neg(x^2 = y) \vee ocupado[i]$
de dos predicados P y Q	$(x \geq 0) \vee (\neg(x^2 = y) \wedge ocupado[i])$
Una **condicional** ($P \rightarrow Q$)	$\neg(x^2 = y) \rightarrow ocupado[i]$
de dos predicados P y Q	$(x \geq 0) \rightarrow (x^2 = y)$
Una **bicondicional** ($P = Q$)	$\neg(x^2 = y) = ocupado[i]$
de dos predicados P y Q	$(x \geq 0) = (x^2 = y)$
Una **cuantificación universal** ($\forall \alpha \in D.P(\alpha)$),	$(\forall \alpha \in [1, N].v3[\alpha] = v1[\alpha] + v2[\alpha])$
siendo D un dominio de valores y P un predicado	$(\forall \alpha \in [1, N].(\forall \beta \in [1, N].T[\alpha, \beta) = T[\beta, \alpha]))$
Una **cuantificación existencial** ($\exists \alpha \in D.P(\alpha)$),	$(\exists \alpha \in [1, N].laTabla[\alpha] = dato)$
siendo D un dominio de valores y P un predicado	$(\exists \alpha \in [2, n-1].(n \% \alpha) = 0)$

Cuantificadores lógicos

Un **cuantificador** es una abreviatura para denotar operaciones análogas sobre todos los valores de un dominio. Para construir predicados se han presentado dos cuantificadores lógicos, el **cuantificador universal** ($\forall \alpha \in D.P(\alpha)$) y el **cuantificador existencial** ($\exists \alpha \in D.P(\alpha)$). El primero de ellos expresa la conjunción lógica de todos los predicados $P(\alpha)$ que se pueden obtener sustituyendo el parámetro α por todos y cada uno de los elementos del dominio D, mientras que el segundo expresa la disyunción lógica de esos mismos predicados. Sea $D = \{d_1, d_2, ..., d_{K-1}, d_K\}$ un conjunto, entonces:

$$(\forall \alpha \in D.P(\alpha)) = P(d_1) \wedge P(d_2) \wedge \cdots \wedge P(d_{K-1}) \wedge P(d_K)$$

$$(\exists \alpha \in D.P(\alpha)) = P(d_1) \vee P(d_2) \vee \cdots \vee P(d_{K-1}) \vee P(d_K)$$

Sea $\mathbb{N} = \{0, 1, 2, ...\}$ el conjunto de los números naturales, entonces los predicados ($\forall \alpha \in \mathcal{N}.P(\alpha)$) y ($\exists \alpha \in \mathbb{N}.P(\alpha)$) equivalen a:

$$(\forall \alpha \in \mathbb{N}.P(\alpha)) = P(0) \wedge P(1) \wedge P(2) \wedge P(3) \wedge P(4) \wedge \ldots \qquad \text{y}$$

$$(\exists \alpha \in \mathbb{N}.P(\alpha)) = P(0) \vee P(1) \vee P(2) \vee P(3) \vee P(4) \vee \ldots$$

Los dos cuantificadores lógicos se resumen en la siguiente tabla.

CUANTIFICADOR LÓGICO	**FORMULACIÓN**	**DEFINICIÓN: VALOR DEL PREDICADO**
Universal	$(\forall \alpha \in D.P(\alpha))$	$P(d_1) \wedge P(d_2) \wedge ... \wedge P(d_K)$
id.	$(\forall \alpha \in \phi.P(\alpha))$	$cierto \quad [true]$
Existencial	$(\exists \alpha \in D.P(\alpha))$	$P(d_1) \vee P(d_2) \vee ... \vee P(d_K)$
id.	$(\exists \alpha \in \phi.P(\alpha))$	$falso \quad [false]$

Cuantificadores aritméticos

Puede resultar también muy útil definir cuantificadores para construir ciertas expresiones aritméticas en las que intervienen la totalidad de los elementos de un conjunto o dominio de valores. En la tabla que sigue se presentan los cuantificadores aritméticos **sumatorio**, **producto**, **conteo**, **máximo** y **mínimo**.

CUANTIFICADOR ARITMÉTICO	FORMULACIÓN	DEFINICIÓN: VALOR DE LA EXPRESIÓN
Sumatorio	$(\Sigma\alpha \in D.E(\alpha))$	$E(d_1) + E(d_2) + ... + E(d_K)$
id.	$(\Sigma\alpha \in \phi.E(\alpha))$	0
Producto	$(\Pi\alpha \in D.E(\alpha))$	$E(d_1) \times E(d_2) \times ... \times E(d_K)$
id.	$(\Pi\alpha \in \phi.E(\alpha))$	1
Conteo	(**Núm** $\alpha \in D.P(\alpha)$)	Núm$(P(d_1), P(d_2), ..., P(d_K))$
id.	(**Núm** $\alpha \in \phi.P(\alpha)$)	0
Máximo	(**Máx** $\alpha \in D.E(\alpha)$)	*Máx*$(E(d_1), E(d_2), ..., E(d_K))$
id.	(**Máx** $\alpha \in \phi.E(\alpha)$)	$indefinido$
Mínimo	(**Mín** $\alpha \in D.E(\alpha)$)	*Mín*$(E(d_1), E(d_2), ..., E(d_K))$
id.	(**Mín** $\alpha \in \phi.E(\alpha)$)	$indefinido$

Donde:

- $D = \{d_1, d_2, ..., d_{K-1}, d_K\}$ define un dominio de valores.
- $E(\alpha)$ denota una expresión aritmética.
- $P(\alpha)$ denota un predicado.
- $Núm(P(d_1), P(d_2), \ldots, P(d_i), \ldots, P(d_{K-1}), P(d_K))$ denota el número de predicados $P(d_i)$ cuyo valor es *cierto*, con $i \in [0, K]$.
- $Máx(E(d_1), E(d_2), \ldots, E(d_i), \ldots, E(d_{K-1}), E(d_K))$ denota el máximo de los valores de las expresiones $E(d_i)$, con $i \in [0, K]$.
- $Mín(E(d_1), E(d_2), \ldots, E(d_i), \ldots, E(d_{K-1}), E(d_K))$ denota el mínimo de los valores de las expresiones $E(d_i)$, con $i \in [0, K]$.