# Computación y Estructuras Discretas I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Tecnologías de Información y Comunicaciones

UNIVERSIDAD ICESI

2024-2

# Agenda del día

1. **Métodos de Demostración**
   - Presentación del tema
   - Ejercicios

2. **Presentación del siguiente tema**
   - Introducción a Coq

# Agenda del día

1. **Métodos de Demostración**
   - Presentación del tema
   - Ejercicios
2. **Presentación del siguiente tema**
   - Introducción a Coq

## Preguntas de interés

1. ¿Qué es una demostración?

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido?

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido? *Cuando las premisas o hipótesis verdaderas dan como resultado la conclusión verdadera.*

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido? *Cuando las premisas o hipótesis verdaderas dan como resultado la conclusión verdadera.*
3. ¿Qué forma tiene un teorema?

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido? *Cuando las premisas o hipótesis verdaderas dan como resultado la conclusión verdadera.*
3. ¿Qué forma tiene un teorema? $\mathbf{p} \rightarrow \mathbf{q}$

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido? *Cuando las premisas o hipótesis verdaderas dan como resultado la conclusión verdadera.*
3. ¿Qué forma tiene un teorema? $\mathbf{p} \rightarrow \mathbf{q}$
4. ¿Qué tipo de demostraciones existen?

## Preguntas de interés

1. ¿Qué es una demostración? *Es un argumento en que se ha empleado una razonamiento lógico para demostrar la veracidad de algo.*
2. ¿Cuándo un argumento es válido? *Cuando las premisas o hipótesis verdaderas dan como resultado la conclusión verdadera.*
3. ¿Qué forma tiene un teorema? $\mathbf{p} \rightarrow \mathbf{q}$
4. ¿Qué tipo de demostraciones existen? *Directas e Indirectas ...*

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

**Ejemplo**

*La suma de enteros pares es par.*

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

## Ejemplo

*La suma de enteros pares es par. Escribimos este enunciado en la forma* $\boldsymbol{p} \rightarrow \boldsymbol{q}$

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

## Ejemplo

*La suma de enteros pares es par. Escribimos este enunciado en la forma* $\boldsymbol{p} \rightarrow \boldsymbol{q}$

*Si n es par y m es par, entonces* $n + m$ *es par*

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

## Ejemplo

*La suma de enteros pares es par. Escribimos este enunciado en la forma* $\boldsymbol{p} \rightarrow \boldsymbol{q}$

*Si n es par y m es par, entonces* $n + m$ *es par*

*Suponemos que la hipótesis de esta implicación es verdadera, es decir* $n = 2k$ *y* $m = 2l$ *donde k y l son enteros.*

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

## Ejemplo

*La suma de enteros pares es par. Escribimos este enunciado en la forma* $\boldsymbol{p} \rightarrow \boldsymbol{q}$
*Si n es par y m es par, entonces* $n + m$ *es par*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir* $n = 2k$ *y* $m = 2l$ *donde k y l son enteros. Se sigue que* $n + m = 2k + 2l = 2(k + l)$ *y sabemos que* $k + l$ *es un entero.*

# Método Directo

Se supone la(s) hipótesis y se emplea un proceso lógico deductivo para demostrar de manera directa la conclusión.

## Ejemplo

*La suma de enteros pares es par. Escribimos este enunciado en la forma* $\boldsymbol{p} \rightarrow \boldsymbol{q}$
*Si n es par y m es par, entonces* $n + m$ *es par*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir* $n = 2k$ *y* $m = 2l$ *donde k y l son enteros. Se sigue que* $n + m = 2k + 2l = 2(k + l)$ *y sabemos que* $k + l$ *es un entero. Por tanto* $n + m$ *es par.*

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

- ¿Por qué es válido el método de la contrarecíproca?

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

- ¿Por qué es válido el método de la contrarecíproca?
  $p \rightarrow q$ es lógicamente equivalente a $(\neg q \rightarrow \neg p)$.

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

- ¿Por qué es válido el método de la contrarecíproca?
  $p \rightarrow q$ es lógicamente equivalente a $(\neg q \rightarrow \neg p)$.
- ¿Qué significa que dos fórmulas sean lógicamente equivalentes?

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

- ¿Por qué es válido el método de la contrarecíproca?
  $p \rightarrow q$ es lógicamente equivalente a $(\neg q \rightarrow \neg p)$.
- ¿Qué significa que dos fórmulas sean lógicamente equivalentes?
  Los valores de verdad de ambas fórmulas siempre son iguales .

# Método Indirecto - Contrarecíproca

Para mostrar $p \rightarrow q$ se muestra $\neg q \rightarrow \neg p$.

- ¿Por qué es válido el método de la contrarecíproca?
  $p \rightarrow q$ es lógicamente equivalente a $(\neg q \rightarrow \neg p)$.
- ¿Qué significa que dos fórmulas sean lógicamente equivalentes?
  Los valores de verdad de ambas fórmulas siempre son iguales .
- En ocasiones es más fácil demostrar su contrarecíproca que la implicación inicial

# Método Indirecto - Contrarrecíproca

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*

# Método Indirecto - Contrarrecíproca

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde $k$ es un entero.*

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde $k$ es un entero. Se sigue que $n^2 = 4k^2 = 2(2k^2)$.*

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde $k$ es un entero. Se sigue que $n^2 = 4k^2 = 2(2k^2)$.*
*Por tanto $n^2$ se puede representar como $2m$ donde $m = 2k$ y concluimos que $n^2$ es par.*

**Ejemplo**

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde $k$ es un entero. Se sigue que $n^2 = 4k^2 = 2(2k^2)$.*
*Por tanto $n^2$ se puede representar como $2m$ donde $m = 2k$ y concluimos que $n^2$ es par.*

¿Porqué es necesario la contrarrecíproca para probar esta propiedad?

**Ejemplo**

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde $k$ es un entero. Se sigue que $n^2 = 4k^2 = 2(2k^2)$.*
*Por tanto $n^2$ se puede representar como $2m$ donde $m = 2k$ y concluimos que $n^2$ es par.*

¿Porqué es necesario la contrarrecíproca para probar esta propiedad? En este caso la demostración directa no nos muestra un camino muy obvio.

# Método Indirecto - Contrarrecíproca

## Ejemplo

*Para todo entero n, si $n^2$ es impar entonces n es impar.*
*Como $p \rightarrow q \equiv \neg q \rightarrow \neg p$ lo anterior es equivalente a decir que para todo entero n, si n es par, $n^2$ es par.*
*Suponemos que la hipótesis de esta implicación es verdadera, es decir $n = 2k$ donde k es un entero. Se sigue que $n^2 = 4k^2 = 2(2k^2)$.*
*Por tanto $n^2$ se puede representar como $2m$ donde $m = 2k$ y concluimos que $n^2$ es par.*

¿Porqué es necesario la contrarrecíproca para probar esta propiedad?
En este caso la demostración directa no nos muestra un camino muy obvio.
Si suponemos que $n^2 = 2k + 1$ tenemos que $n = \pm\sqrt{2k+1}$ lo cual no es muy útil.

# Método de Contradicción - Reducción al absurdo

Para probar que $p$ es verdadero,

# Método de Contradicción - Reducción al absurdo

Para probar que $p$ es verdadero,

- Suponemos que podemos encontrar una contradicción $q$ tal que $\neg p \rightarrow q$ es verdadera.

# Método de Contradicción - Reducción al absurdo

Para probar que $p$ es verdadero,

- Suponemos que podemos encontrar una contradicción $q$ tal que $\neg p \rightarrow q$ es verdadera.
- Como $\neg p \rightarrow q \equiv \neg p \rightarrow F\ \neg p$ debe ser falsa y por tal $p$ es verdadera.

# Método de Contradicción - Reducción al absurdo

Para probar que $p$ es verdadero,

- Suponemos que podemos encontrar una contradicción $q$ tal que $\neg p \rightarrow q$ es verdadera.
- Como $\neg p \rightarrow q \equiv \neg p \rightarrow F\ \neg p$ debe ser falsa y por tal $p$ es verdadera.
- Esta técnica se utiliza cuando podemos encontrar una contradicción.

# Método de Contradicción - Reducción al absurdo

Para probar que $p$ es verdadero,

- Suponemos que podemos encontrar una contradicción $q$ tal que $\neg p \rightarrow q$ es verdadera.
- Como $\neg p \rightarrow q \equiv \neg p \rightarrow F\ \neg p$ debe ser falsa y por tal $p$ es verdadera.
- Esta técnica se utiliza cuando podemos encontrar una contradicción.
- Consecuentemente para probar $p \rightarrow q$ debemos probar que $(p \wedge \neg q) \rightarrow C$, tal que C es una contradicción.

# Método de Contradicción - Reducción al absurdo

## Ejemplo

*Si* $n^2$ *es par entonces n es par* $(p \rightarrow q)$

# Método de Contradicción - Reducción al absurdo

## Ejemplo

*Si $n^2$ es par entonces n es par* $(p \rightarrow q)$
*Suponemos que $n^2$ es par y n es impar* $(p \wedge \neg q)$ .

# Método de Contradicción - Reducción al absurdo

## Ejemplo

*Si $n^2$ es par entonces n es par $(p \rightarrow q)$*
*Suponemos que $n^2$ es par y n es impar $(p \wedge \neg q)$ . Por lo tanto existe un entero k tal que $n = 2k + 1$*

## Ejemplo

*Si* $n^2$ *es par entonces n es par* $(p \rightarrow q)$
*Suponemos que* $n^2$ *es par y n es impar* $(p \wedge \neg q)$ *. Por lo tanto existe un entero k tal que* $n = 2k + 1$ *ahora,*
$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$

## Ejemplo

*Si* $n^2$ *es par entonces n es par* $(p \rightarrow q)$
*Suponemos que* $n^2$ *es par y n es impar* $(p \wedge \neg q)$ *. Por lo tanto existe un entero k tal que* $n = 2k + 1$ *ahora,*
$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ *por tal* $n^2 = 2l + 1$ *donde* $l = 2k^2 + 2k$

## Ejemplo

*Si $n^2$ es par entonces n es par $(p \rightarrow q)$*
*Suponemos que $n^2$ es par y n es impar $(p \wedge \neg q)$ . Por lo tanto existe un entero k tal que $n = 2k + 1$ ahora,*
$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ *por tal* $n^2 = 2l + 1$
*donde* $l = 2k^2 + 2k$ *entonces* $n^2$ *es impar.*

**Ejemplo**

*Si $n^2$ es par entonces n es par $(p \rightarrow q)$*
*Suponemos que $n^2$ es par y n es impar $(p \wedge \neg q)$ . Por lo tanto existe un entero $k$ tal que $n = 2k + 1$ ahora,*
*$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ por tal $n^2 = 2l + 1$ donde $l = 2k^2 + 2k$ entonces $n^2$ es impar. Hemos llegado a una contradicción pues $n^2$ es par y $n^2$ es impar*

# Disyunción de Casos

Para demostrar una implicación de la forma,
$p = p_1 \vee p_2 \vee \ldots \vee p_n \rightarrow q$.

# Disyunción de Casos

Para demostrar una implicación de la forma,
$p = p_1 \vee p_2 \vee \ldots \vee p_n \rightarrow q$.

- Se demuestra que $p_1 \rightarrow q \wedge p_2 \rightarrow q \wedge p_3 \rightarrow q \wedge \ldots \wedge p_n \rightarrow q$

# Disyunción de Casos

## Ejemplo

$\forall x, y \in \mathbb{R}$, $|xy| = |x||y|$, *donde* $x$ *e* $y$ *son números reales*

# Disyunción de Casos

## Ejemplo

$\forall x, y \in \mathbb{R}$, $|xy| = |x||y|$, *donde* $x$ *e* $y$ *son números reales*
*Tenemos* $p =$ $x$ *e* $y$ *son números reales,* $q = |xy| = |x||y|$

## Ejemplo

$\forall x, y \in \mathbb{R}$, $|xy| = |x||y|$, *donde* $x$ *e* $y$ *son números reales*
*Tenemos* $p = x$ *e* $y$ *son números reales,* $q = |xy| = |x||y|$
*Sabemos que* $p \equiv p_1 \vee p_2 \vee p_3 \vee p_4$ *donde* $p_1 = x \geq 0 \wedge y \geq 0$, $p_2 = x \geq 0 \wedge y < 0$, $p_3 = x < 0 \wedge y \geq 0$, $p_4 = x < 0 \wedge y < 0$

**Ejemplo**

$\forall x, y \in \mathbb{R}$, $|xy| = |x||y|$, *donde* $x$ *e* $y$ *son números reales*
*Tenemos* $p = x$ *e* $y$ *son números reales,* $q = |xy| = |x||y|$
*Sabemos que* $p \equiv p_1 \vee p_2 \vee p_3 \vee p_4$ *donde* $p_1 = x \geq 0 \wedge y \geq 0$,
$p_2 = x \geq 0 \wedge y < 0$, $p_3 = x < 0 \wedge y \geq 0$, $p_4 = x < 0 \wedge y < 0$
*Por tanto para demostrar* $p_1 \vee p_2 \vee p_3 \vee p_4 \rightarrow q$ *probamos*
$p_1 \rightarrow q \wedge p_2 \rightarrow q \wedge p_3 \rightarrow q \wedge p_4 \rightarrow q$

# Disyunción de Casos

## Ejemplo

$p_1 \rightarrow q$ : *suponemos* $p_1$, *lo cual nos dice que* $x \geq 0 \wedge y \geq 0$, *entonces* $xy \geq 0$ *por lo que* $|xy| = xy = |x||y|$

## Ejemplo

$p_1 \rightarrow q$ : *suponemos* $p_1$, *lo cual nos dice que* $x \geq 0 \wedge y \geq 0$, *entonces* $xy \geq 0$ *por lo que* $|xy| = xy = |x||y|$

$p_2 \rightarrow q$ : *suponemos* $p_2$, *lo cual nos indica que* $x \geq 0 \wedge y < 0$, *entonces* $xy \leq 0$ *por lo que* $|xy| = -xy = x(-y) = |x||y|$

## Ejemplo

$p_1 \rightarrow q$ : *suponemos* $p_1$, *lo cual nos dice que* $x \geq 0 \wedge y \geq 0$, *entonces* $xy \geq 0$ *por lo que* $|xy| = xy = |x||y|$
$p_2 \rightarrow q$ : *suponemos* $p_2$, *lo cual nos indica que* $x \geq 0 \wedge y < 0$, *entonces* $xy \leq 0$ *por lo que* $|xy| = -xy = x(-y) = |x||y|$
$p_3 \rightarrow q$ : *suponemos* $p_3$, *lo cual nos lleva a* $x < 0 \wedge y \geq 0$, *entonces* $xy \leq 0$ *por lo que* $|xy| = -xy = (-x)y = |x||y|$

## Ejemplo

$p_1 \rightarrow q$ : *suponemos* $p_1$, *lo cual nos dice que* $x \geq 0 \wedge y \geq 0$, *entonces* $xy \geq 0$ *por lo que* $|xy| = xy = |x||y|$
$p_2 \rightarrow q$ : *suponemos* $p_2$, *lo cual nos indica que* $x \geq 0 \wedge y < 0$, *entonces* $xy \leq 0$ *por lo que* $|xy| = -xy = x(-y) = |x||y|$
$p_3 \rightarrow q$ : *suponemos* $p_3$, *lo cual nos lleva a* $x < 0 \wedge y \geq 0$, *entonces* $xy \leq 0$ *por lo que* $|xy| = -xy = (-x)y = |x||y|$
$p_4 \rightarrow q$ : *suponemos* $p_4$, *lo cual nos dice que* $x < 0 \wedge y < 0$, *entonces* $xy > 0$ *por lo que* $|xy| = (-x)(-y) = |x||y|$
*Hemos completado la demostración.*

# Contraejemplo

Un teorema que tenga la forma $\forall x P(x)$ y del cual tengamos duda que sea verdadero o que no se pueda encontrar una demostración, se halla un elemento $x' \in D$, en donde $P(x')$ sea falso.

## Ejemplo

*"Todos los números primos son impares"*

# Contraejemplo

Un teorema que tenga la forma $\forall x P(x)$ y del cual tengamos duda que sea verdadero o que no se pueda encontrar una demostración, se halla un elemento $x' \in D$, en donde $P(x')$ sea falso.

## Ejemplo

*"Todos los números primos son impares"*
*Podemos demostrar que esta sentencia es falsa si encontramos un contraejemplo. Es decir, si encontramos un número primo que sea par.*

# Contraejemplo

Un teorema que tenga la forma $\forall x P(x)$ y del cual tengamos duda que sea verdadero o que no se pueda encontrar una demostración, se halla un elemento $x' \in D$, en donde $P(x')$ sea falso.

## Ejemplo

*"Todos los números primos son impares"*
*Podemos demostrar que esta sentencia es falsa si encontramos un contraejemplo. Es decir, si encontramos un número primo que sea par. 2 es un número par y es un número primo. Hemos probado que la sentencia "Todos los números primos son impares" es falsa.*

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

**Ejemplo**

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

## Ejemplo

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$
*Debemos probar* $p_1 \leftrightarrow p_2$

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

## Ejemplo

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$
*Debemos probar* $p_1 \leftrightarrow p_2$
$p_1 \rightarrow p_2$ : *Suponemos* $p_1$ *lo cual nos dice que* $n = 2k$ *siendo* $k$ *un entero.*

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

## Ejemplo

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$
*Debemos probar* $p_1 \leftrightarrow p_2$
$p_1 \rightarrow p_2$ : *Suponemos* $p_1$ *lo cual nos dice que* $n = 2k$ *siendo* $k$ *un entero.*
*Se sigue que* $n + 1 = 2k + 1$ *luego* $n + 1$ *es un entero impar.*

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

## Ejemplo

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$
*Debemos probar* $p_1 \leftrightarrow p_2$
$p_1 \rightarrow p_2$ : *Suponemos* $p_1$ *lo cual nos dice que* $n = 2k$ *siendo* $k$ *un entero.*
*Se sigue que* $n + 1 = 2k + 1$ *luego* $n + 1$ *es un entero impar.*
$p_2 \rightarrow p_1$ : *Suponemos* $p_2$. *Esto nos indica que* $n + 1 = 2k + 1$ *siendo* $k$ *un entero.*

# Demostración por equivalencia

Un teorema que tenga la forma $p \leftrightarrow q$, siendo $p, q$ proposiciones, se demuestra empleando la tautología $(p \leftrightarrow q) \Leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$. Cada implicación se demuestra de forma independiente, empleando métodos no necesariamente iguales. En ocasiones la doble implicación se demuestra de forma similar de ida y vuelta.

**Ejemplo**

*Si* $p_1 = n$ *es un entero par y* $p_2 = n + 1$ *es un entero impar,* $p_1 \equiv p_2$
*Debemos probar* $p_1 \leftrightarrow p_2$
$p_1 \rightarrow p_2$ : *Suponemos* $p_1$ *lo cual nos dice que* $n = 2k$ *siendo* $k$ *un entero.*
*Se sigue que* $n + 1 = 2k + 1$ *luego* $n + 1$ *es un entero impar.*
$p_2 \rightarrow p_1$ : *Suponemos* $p_2$. *Esto nos indica que* $n + 1 = 2k + 1$ *siendo* $k$ *un entero.*
*Se sigue que* $n = 2k$ *luego* $n$ *es un entero par.*

# Agenda del día

1. **Métodos de Demostración**
   - Presentación del tema
   - Ejercicios

2. **Presentación del siguiente tema**
   - Introducción a Coq

# Ejercicios

**Ejercicio**

*Demuestre que el cuadrado de un número par es un número par utilizando una demostración directa.*

# Ejercicios

**Ejercicio**

*Demuestre que el cuadrado de un número par es un número par utilizando una demostración por reducción al absurdo.*

# Ejercicios

**Ejercicio**

*Demuestre que todo entero impar es una diferencia de cuadrados utilizando una demostración directa.*

# Ejercicios

**Ejercicio**

*Demuestre que la suma de dos impares es par.*

# Ejercicios

## Ejercicio

*Demuestre que si $n$ es un entero y $3n + 2$ es par, entonces $n$ es par usando:*

a) *Una demostración indirecta.*

b) *Una demostración por reducción al absurdo.*

# Ejercicios

**Ejercicio**

*Demuestre que se cumple, o que no, que el producto de dos números irracionales es irracional.*

# Ejercicios

**Ejercicio**

*Demuestre que se cumple, o que no, que el producto de dos números racionales es racional.*

# Ejercicios

**Ejercicio**

*Demuestre que se cumple, o que no, que el producto de un número racional distinto de cero y un número irracional es un número irracional.*

# Ejercicios

**Ejercicio**

*Demuestre que si $x$ es un número racional distinto de cero, entonces $1/x$ es racional.*

# Ejercicios

**Ejercicio**

*Sea $n$ un entero positivo, demuestre que $n$ es par si y sólo sí $7n + 4$ es par.*

# Agenda del día

1. Métodos de Demostración
   - Presentación del tema
   - Ejercicios

2. **Presentación del siguiente tema**
   - Introducción a Coq

## ¿Qué es Coq?

## ¿Qué es Coq?

- Se trata de un asistente de demostraciones.
- Un programa que ayuda a llevar a cabo pruebas matemáticas e indica con seguridad que lo que se ha verificado sea cierto.
- Coq tiene diversas aplicaciones.
  - Formalizar pruebas ya existentes para comprobar con seguridad que son ciertas.
  - Ayudar a demostrar nuevos teoremas.
  - Extraer programas correctos por construcción.
  - Probar que determinado programa, lenguaje, función, o sistema de tipos preserva una invariante.
  - (semi)automatizar todas estas tareas.

## ¿Algo adicional?

## ¿Algo adicional?

- A partir de especificaciones en Coq se pueden extraer programas ejecutables en Objective Caml o Haskell.
- En Coq todo se formaliza en una lenguaje que se denomina CIC (Calculus of Inductive Constructions).
  - CoC (Calculus of Constructions) es una teoría de tipos que puede servir como un lenguaje funcional tipado o una base de construcción matemática.
  - CIC es una variante de CoC que añade tipos inductivos.
  - La importancia de los tipos inductivos es que permite la creación de nuevo tipos a partir de constantes y funciones.
  - Características que permiten bridar un rol similar a las estructuras de datos.
  - Permite una teoría de tipos que adiciona conceptos como números, relaciones y árboles.

## Ejemplo de definición inductiva de los números naturales en coq

```
Inductive nat : Set :=
   | 0 : nat
   | S : nat -> nat.
```

- Este es el ejemplo estándar de como codificar números naturales usado la codificación de Peano.
- Un número natural se crea a partir de la constante “0”
- La función “S” a otro número natural.
- “S” es la función sucesor que representa sumar 1 a un número
- “0” es cero, “S 0” es uno, ”S (S 0)” es dos, ”S (S (S 0))” es tres, etc.

## Además...

**Además...**

- Todos los juicios lógicos en Coq son juicios tipados.
- El eje de Coq es de hecho un algoritmo de verificación de tipos.
- Algoritmos ejecutables

## (Paréntesis - ¿Qué es la programación funcional?)

## (Paréntesis - ¿Qué es la programación funcional?)

- Tanto Objective Caml como Haskell son ejemplos de lenguajes funcionales.
- Otros lenguajes como C++, Python, Scala o el mismo Java soportan elementos de este paradigma.
- La programación funcional es un paradigma de programación en donde los programas se construyen aplicando y componiendo funciones.
- Es un paradigma de programación declarativo (declara lo que el programa debe cumplir más no como lo ha de hacer, a diferencia del imperativo que se basa en una secuencia de pasos detallados que modifican el estado del programa).
- Se basa en funciones que no modifican variables pero si generan nuevas como salida.
- La salida de una función pura solamente depende de sus parámetros de entrada (no hay impactos externos y se evitan efectos secundarios)

## (Paréntesis - ¿Por qué es importante la programación funcional?)

**(Paréntesis - ¿Por qué es importante la programación funcional?)**

- Funciones puras garantizan que el estado fuera del programa no se altere.
- Más fáciles de probar.
- Conducen a menos bugs.
- Código funcional tiende a ser más fácil de comprender.
- La recursión es más simple.
- Entre otros.

## Función imperativa de potencia

```
function pow(x, n) {
    let res = 1;
    for (let i = 0; i < n; i++) {
        res *= x;
    }
    return res;
}
```

## Función funcional de potencia

```
function pow(x, n) {
    if (n == 1) {
        return x;
    }
    return x * pow(x, n - 1);
}
```

## ¿Por qué es relevante Coq?

## ¿Por qué es relevante Coq?

- En la práctica, hay muchos programas con errores que tienen un coste tremendo, desde fallos de seguridad, hasta errores de cálculo.
- En general, las metodologías de desarrollo intentan evitar estos problemas siguiendo una serie de técnicas como el uso de tests.
- Desafortunadamente estas técnicas no pueden asegurar que el programa sea correcto, sólo que no tenga fallos conocidos.
- Mediante los métodos formales podemos asegurarnos de que un programa cumpla con determinadas propiedades matemáticamente demostrables:
  - Que una función acabe en tiempo finito
  - Que converja correctamente a un resultado.
  - Que no traspase información que ha de permanecer secreta.

## Vistazo general

- Coq actúa como todo REPL (read-print-evaluate loop).
- Lee las declaraciones que introducimos, las evalúa e imprime el resultado.

## Ejemplos sobre verificación de tipos

```
Coq < Check 0.
0
     : nat

Coq < Check Nat.add.
Nat.add
     : nat -> nat -> nat
```

## Verificación de tipos

- Check es un comando que nos da el tipo de una expresión.
- En el primer caso, el 0 es un número natural.
- El tipo de Nat.add, viene dado como nat -> nat -> nat.
- Esto indica que add toma un número natural (nat), devuelve algo que toma otro natural y devuelve un natural (nat -> nat).

## Funciones y parámetros

- En Coq se utiliza la noción de función de modo mś formal que en otros lenguajes.
- Estamos acostumbrados a tener funciones que toman varios parámetros, como `int suma(int a, b).`
- En Coq las funciones toman un parámetro y devuelven un parámetro.
- Estas funciones se trabajan mediante el uso de funciones parciales y la técnica denominada “currying”.
- Una función como suma, en Coq, toma un número natural, devuelve una función que toma un sólo número natural, le suma el anterior, y devuelve el resultado, que es otro natural.
- A las funciones que toman o devuelven otras funciones como parámetros se las denomina funciones de orden superior.
- Simplemente, `add` es una función que, a efectos prácticos, toma dos naturales y nos devuelve otro, mediante una función intermedia de orden superior.

## Ejemplo sobre el cómputo de funciones

```
Coq < Compute Nat.add 2 3.
      = 5
      : nat
```

## ¿Qué nos presenta el anterior ejemplo?

**¿Qué nos presenta el anterior ejemplo?**

- `add` se aplica sobre el parámetro 2 y retorna otra función que se aplica sobre 3 para retornarnos el valor 5.
- Adicionalmente nos indica el tipo del valor de retorno, un natural.

## ¿Qué nos presenta el anterior ejemplo?

- `add` se aplica sobre el parámetro 2 y retorna otra función que se aplica sobre 3 para retornarnos el valor 5.
- Adicionalmente nos indica el tipo del valor de retorno, un natural.

## ¿Pero qué es un tipo en Coq?

## ¿Qué nos presenta el anterior ejemplo?

- `add` se aplica sobre el parámetro 2 y retorna otra función que se aplica sobre 3 para retornarnos el valor 5.
- Adicionalmente nos indica el tipo del valor de retorno, un natural.

## ¿Pero qué es un tipo en Coq?

- Función de primera clase (se puede tratar como otro valor del lenguaje).
- Se construye de manera inductiva.

## Ejemplo sobre definición inductiva en Coq

```
Coq < Print nat.
Inductive nat : Set :=  O : nat | S : nat -> nat
```

## ¿Qué nos presenta el anterior ejemplo?

## ¿Qué nos presenta el anterior ejemplo?

- nat es un tipo inductivo (perteneciente a Set (conjunto)) que se puede formar mediante dos constructores:
  1. O que representa el cero.
  2. Una función S (función sucesor) que toma un nat y devuelve otro nat.

## Ejemplo de verificación de tipo de sucesor

```
Coq < Check S (S (S O)).
3
     : nat
```

## ¿Qué aprendemos del ejemplo anterior?

**¿Qué aprendemos del ejemplo anterior?**

- 3 es simplemente azúcar sintáctico para S (S (S O))
- Es la codificación de Peano que vimos previamente.
- Utiliza un sólo dígito y aunque es poco eficiente permite realizar pruebas inductivas con mayor facilidad.

## Ejemplo de verificación de tipo de add

```
Coq < Print Nat.add.
Nat.add =
fix add (n m : nat) {struct n} : nat :=
  match n with
  | 0 => m
  | S p => S (add p m)
  end
     : nat -> nat -> nat
```

- fix es un operador para definir funciones recursivas (se llama a sí misma).
- (n m : nat) {struct n} : nat. indica que toma 2 parámetros, n y m, de tipo nat
- struct n indica que la recursión es estructural en n, es decir, que cuando add se llama a sí misma, n va a ser siempre inferior.
- Esto garantiza que add siempre devuelva un resultado en tiempo finito.
- El último nat es el tipo del valor de retorno de la función.
- Decimos que toma dos parámetros porque el currying es automático.

- Luego del := viene el cuerpo en de la función.
- match es un operador para encajar patrones.
- Se indica la variable para hacer el match junto a with y los patrones respectivos.
- En este caso se encaja con n
- En el primer caso si n encaja con 0, la función devuelve m pues 0 + m = m.
- Si n encaja con S p, es decir, cuando n es el sucesor de otro nat p, la función devuelve S (add p m), el sucesor de añadir p a m.
- Aquí se percibe la recursión estructural en n, porque al llamarse a sí misma, el valor de n siempre tendrá que disminuir para llegar al caso base.
- Finalmente, end cierra la función, y se presenta el tipo, nat -> nat -> nat.

**Ejercicio**

*Defina la función recursiva del factorial en Coq*

## Solución

```
Fixpoint factorial (n:nat) : nat :=
match n with
| O => (S O)
| S p => (S p) * (factorial p)
end.
```

**Ejercicio**

*Defina la función recursiva de la multiplicación de dos números n y m en Coq*

## Solución

```
Fixpoint multiplicar (n m:nat) : nat :=
match n with
| O => O
| S p => m + (multiplicar p m)
end.
```

## Primer ejemplo de demostración

```
Coq < Theorem add_0_m: forall m:nat, 0 + m = m.
```

## Primer ejemplo de demostración

```
1 subgoal
  ============================
  forall m : nat, 0 + m = m
add_0_m <
```

## Primer ejemplo de demostración

```
add_0_m < intros.
  m : nat
  ============================
  0 + m = m
```

## Primer ejemplo de demostración

```
add_0_m < simpl.
  m : nat
  ============================
  m = m
```

## Primer ejemplo de demostración

```
add_0_m < reflexivity.
No more subgoals.
```

## Primer ejemplo de demostración

```
add_0_m < Qed.
intros.
simpl.
reflexivity.
Qed.
add_0_m is defined
Coq <
```