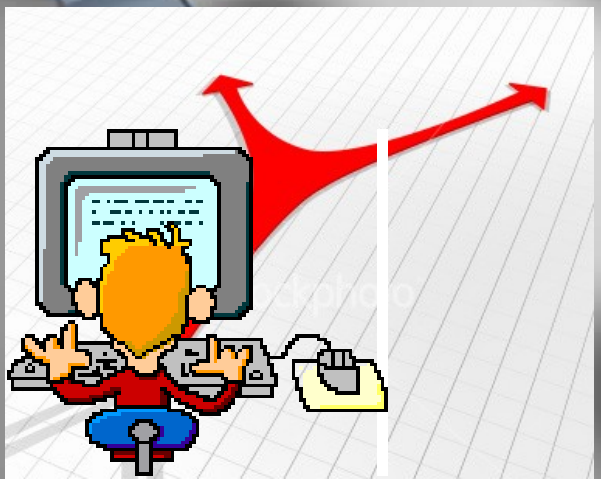
# Programación I

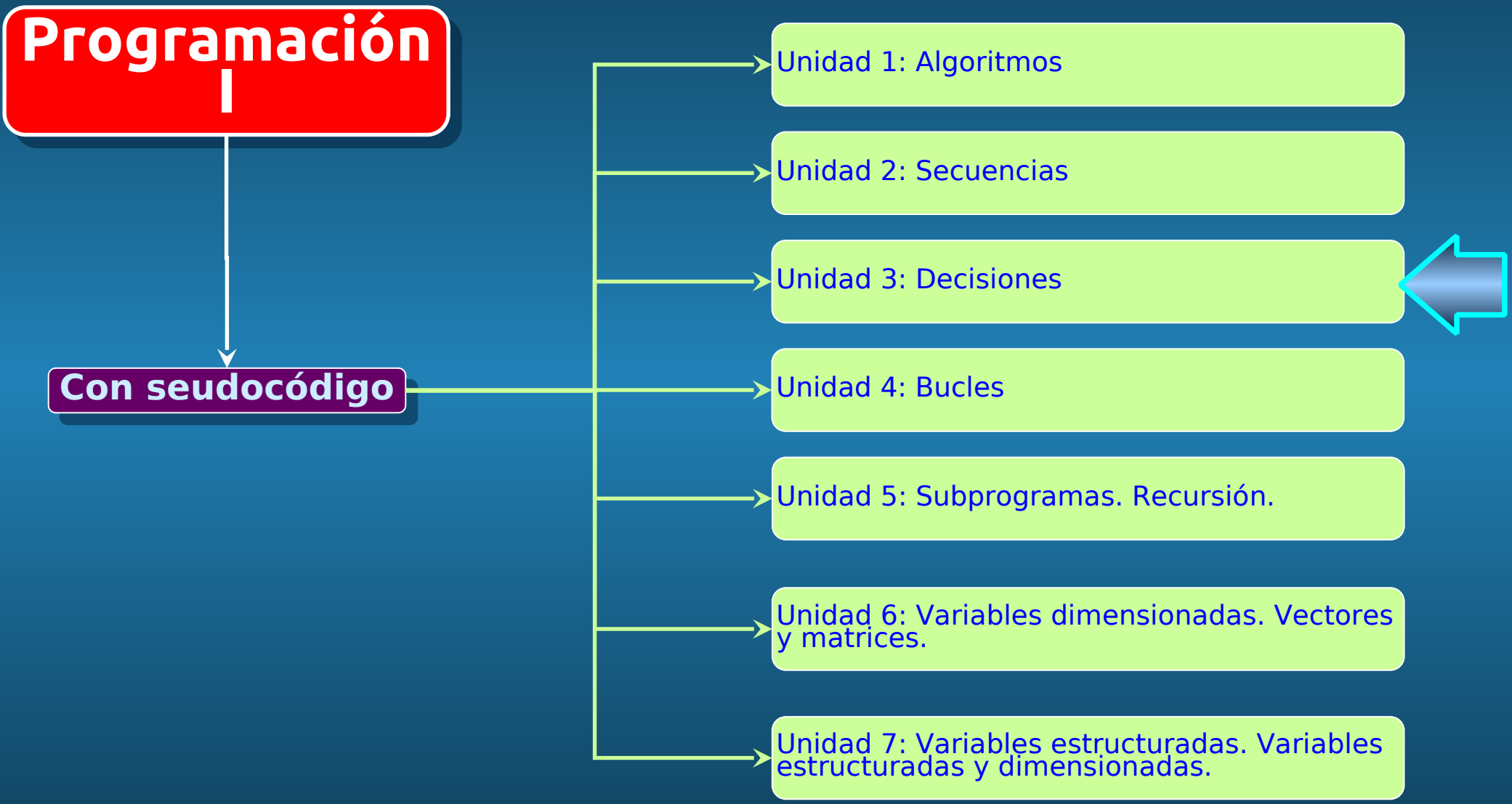
## Ing. Carlos R. Rodríguez

### Tecnicatura Universitaria en Programación



UNIVERSIDAD TECNOLÒGICA NACIONAL

*Facultad Regional Mendoza*


Programación
I
Con seudocódigo
Unidad 1: Algoritmos
Unidad 2: Secuencias
Unidad 3: Decisiones
Unidad 4: Bucles
Unidad 5: Subprogramas. Recursión.
Unidad 6: Variables dimensionadas. Vectores y matrices.
Unidad 7: Variables estructuradas. Variables estructuradas y dimensionadas.

# Verificación de Programas

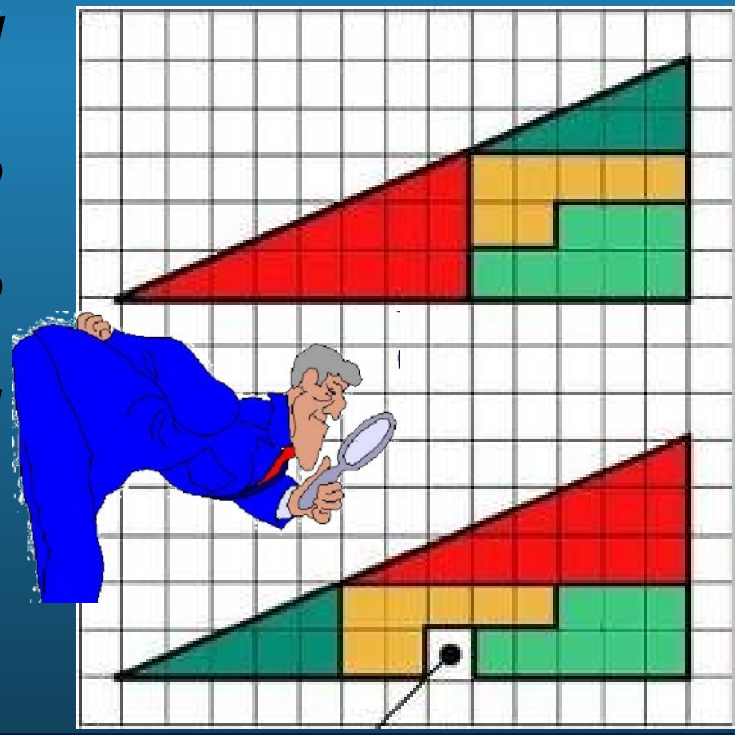
"*La prueba experimental de programas puede ser empleada para demostrar la presencia de errores, pero nunca su ausencia*"

Niklaus Wirth

# Verificación de Programas...

"*Es necesario hacer abstracción de los procesos individuales y deducir, a partir del modelo de conducta del programa, postulados de validez general. Este método de prueba se llama verificación de programas*"

Niklaus Wirth



# Programar implica:

Las habilidades de resolver problemas del tipo:

- Dado un enunciado, elaborar el algoritmo (*y – habitualmente – codificarlo*).
  - **Es la situación ideal**: se trata de elaborar software nuevo, sin más compromisos que las **especificaciones**.
  - Hay **plena libertad** de decidir el mejor **diseño** de nuestro programa.

# Programar implica...

Las habilidades de resolver problemas del tipo:

- Dado un algoritmo (*en código fuente*) deducir qué hace (*reconstruir el **enunciado***).
  - Implica la lectura comprensiva del código, ***entendiendo** qué sucede en cada paso* e intentando *reconstruir por qué se escribió de esa forma*.

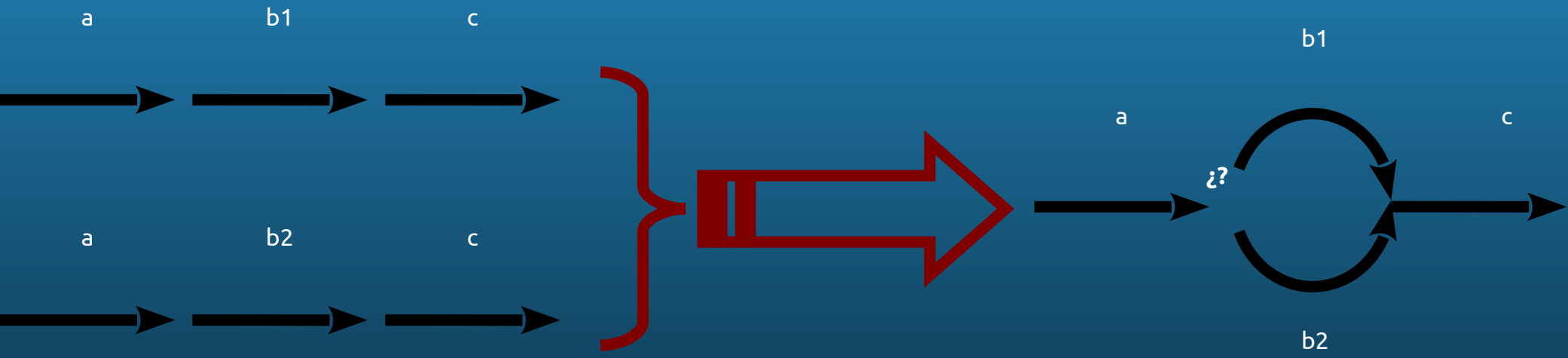
# Programar implica...

Las habilidades de resolver problemas del tipo:

- Dado un algoritmo y una especificación encontrar y corregir los errores.
  - Es la **tarea más complicada**, pues no sólo implica leer código elaborado por ***otros*** (*o por* ***nosotros***) sino también entenderlo con una profundidad suficiente como para afrontar el riesgo de modificarlo.

# Decisiones

Permiten que en lugar de **elaborar** (*por ejemplo*) **dos programas similares** para ***dos casos apenas distintos***, elaboremos uno solo con dos acciones alternativas, según el caso.



# Decisiones...

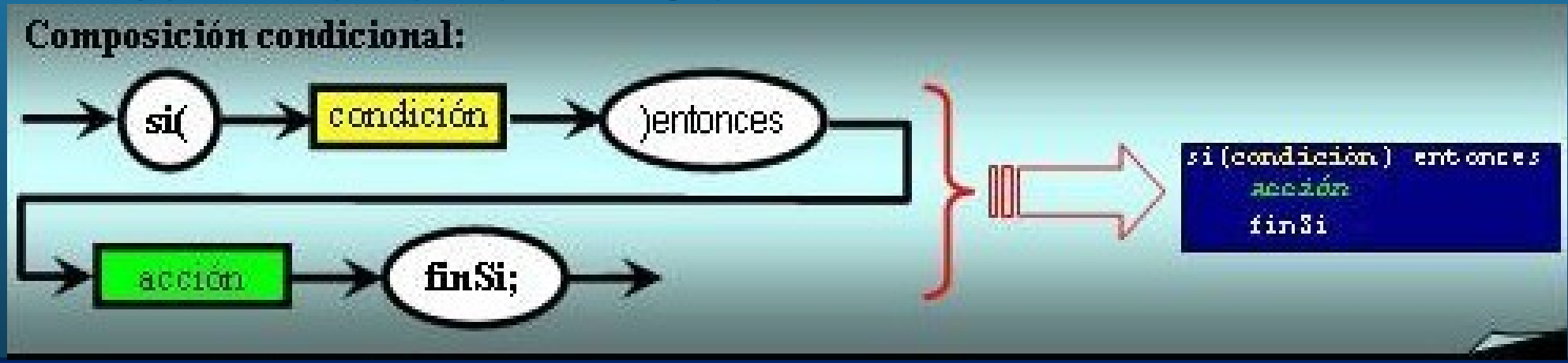
## Composición **Condicional**

- En este caso, **se ejecuta una** acción **si** (*y solo si*) una condición (*expresión relacional o lógica*) es verdadera..

# Decisiones...

## Composición **Condicional**

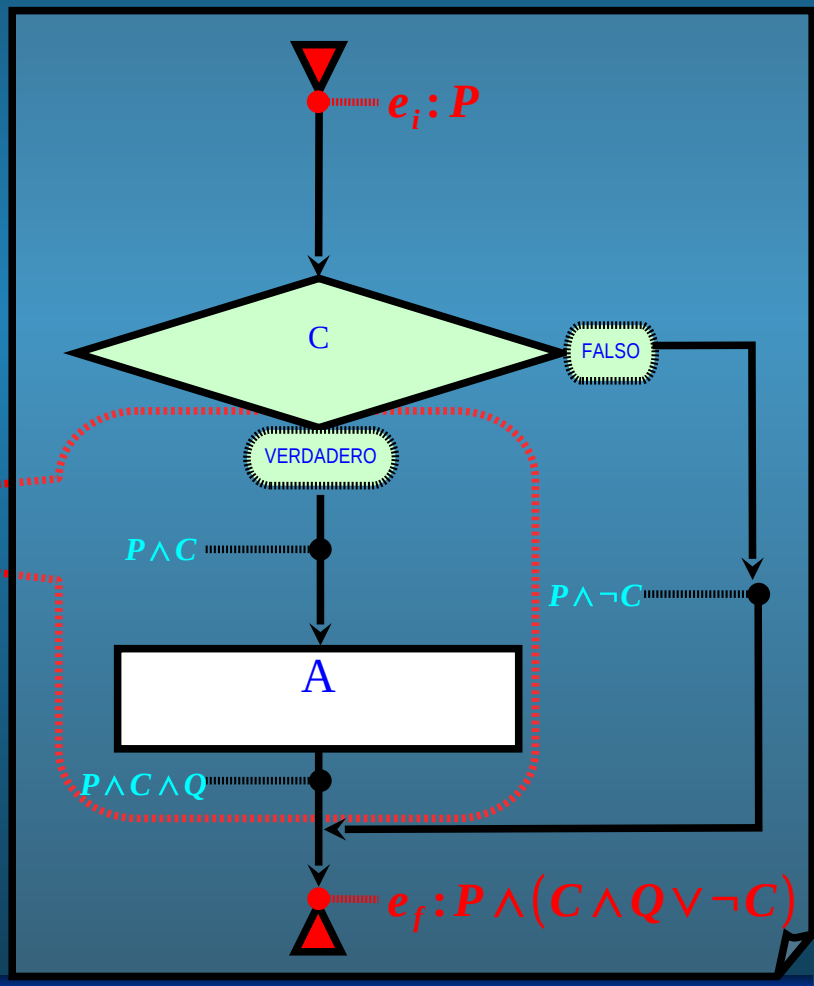
- Sintaxis:
  - La condición va ubicada *dentro de paréntesis* luego de la palabra clave si y el código correspondiente a la acción (*simple* o *compuesta*) va en *renglones sucesivos*, *hasta terminar en un* finSi.



# Decisiones...

## Composición **Condicional**

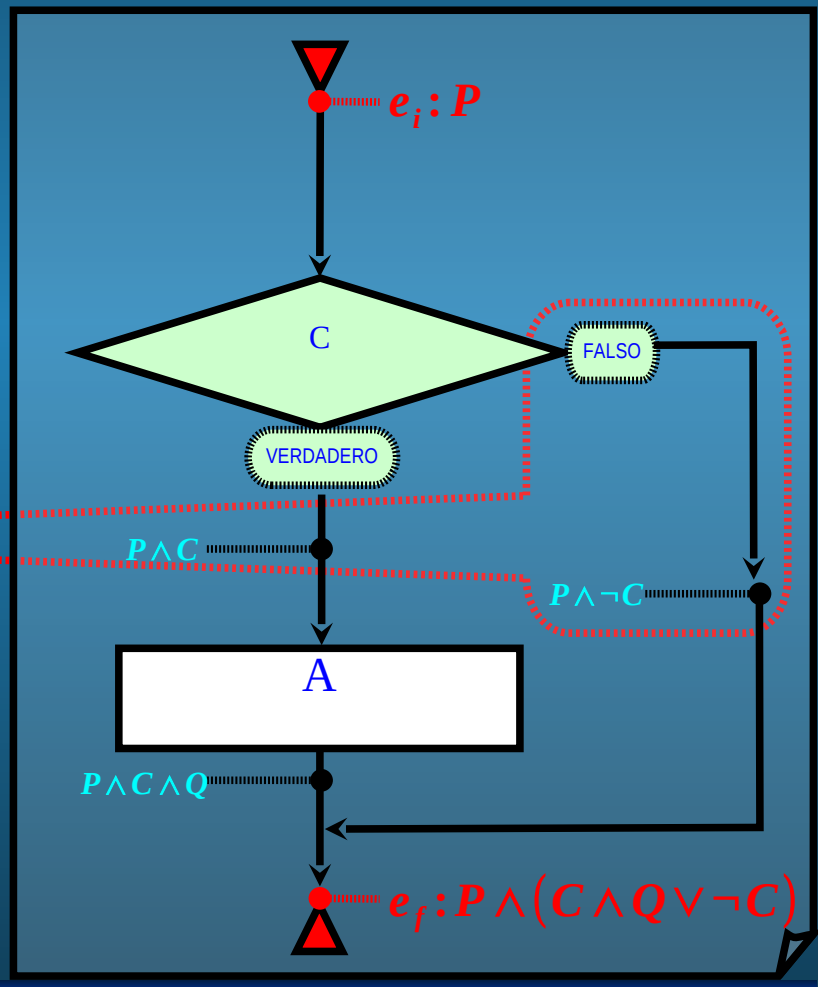
- Semántica:
  - Dadas las **precondiciones** P (estado *inicial*).
  - Al evaluar la condición *sucede una de dos cosas*:
    - Es verdadera
      - ✔ Se cumplen P y C y se ejecuta la **acción** A, con **consecuencias** Q.



# Decisiones...

## Composición **Condicional**

- Semántica:
  - Dadas las **precondiciones** P (estado *inicial*).
  - Al evaluar la condición *sucede una de dos cosas*:
    - Es falsa
      - ✔ Se cumplen P y la negación de C y se pasa directamente al final de la estructura.



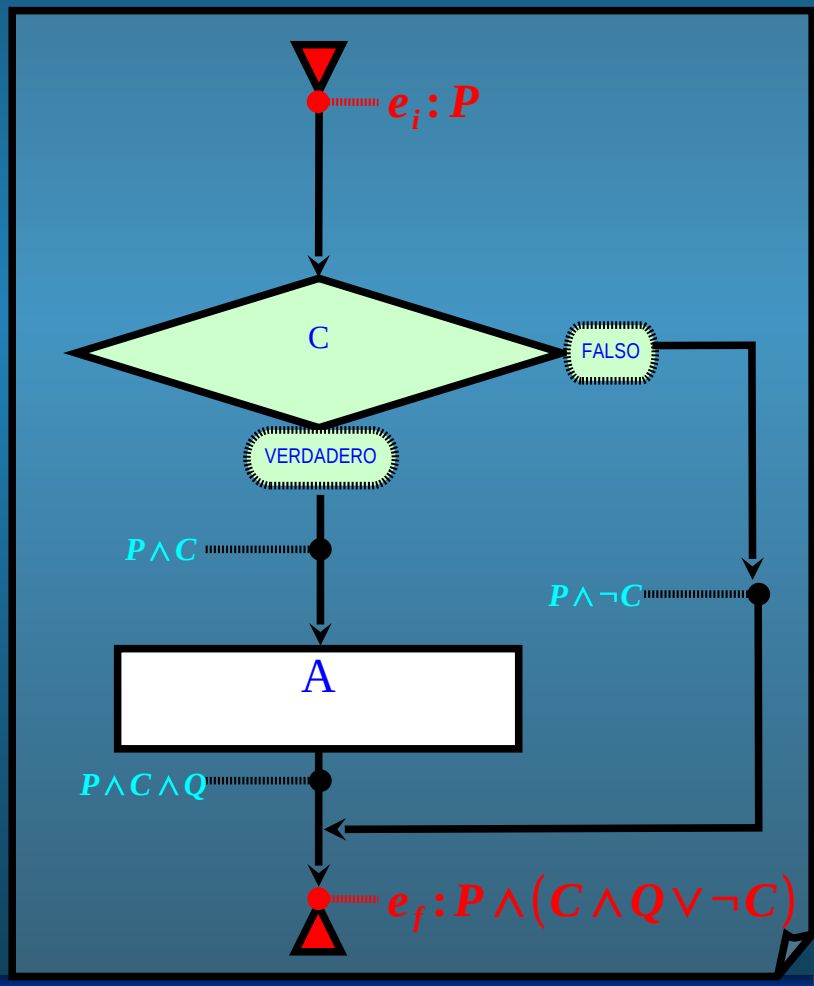
# Decisiones...

## Composición **Condicional**

- Semántica:
  - El estado final es:

$$e_f : P \wedge (C \wedge Q \vee \neg C)$$

  - ...que se lee: "*dado P, si la condición C es cierta se ejecuta la acción A produciendo las consecuencias Q, sino la condición C es falsa*".



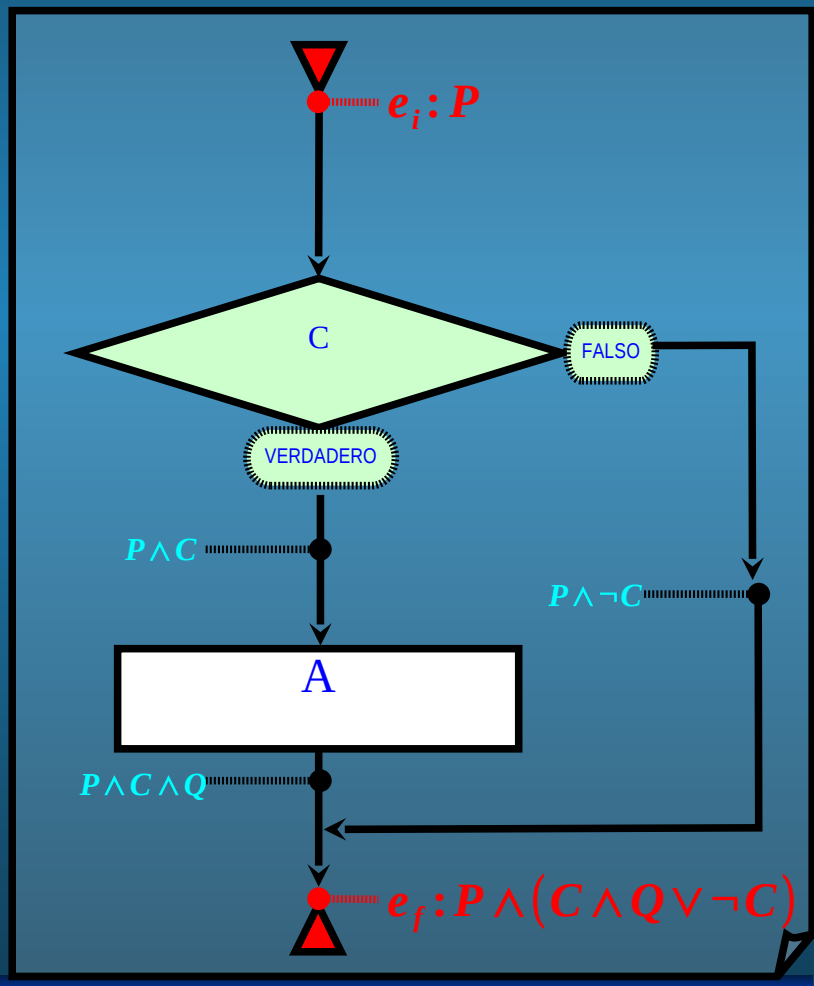
# Decisiones...

## Composición **Condicional**

- Semántica:
  - El estado final es:

$$e_f : P \wedge (C \wedge Q \vee \neg C)$$

  - Sólo existirán consecuencias de la **acción** *si la condición es* ***verdadera***.



# Decisiones...

## Composición Condicional

- ¿Qué hace?

```
#include <program1.h>
/**
*    Enunciado:
*/
principal
real uno,dos;
limpiar;
leerM(uno,"Valor 1:");
leerM(dos,"Valor 2:");
si(uno > dos) entonces
    dos = uno;
    finSi
mostrar << uno  << " - " << dos << salto;
pausa;
finPrincipal
```

# Decisiones...

## Composición **Condicional**

- ¿Qué hace?
  - **Dados 2 valores**, muestra el **mayor**.

```
#include <program1.h>
/**
*    Enunciado: Dados dos valores, mostrar el mayor.
*/
principal                                          // Unidad de programa principal
real uno,dos;
limpiar;                                           // Limpia la pantalla.
leerM(uno,"Valor 1:");
leerM(dos,"Valor 2:");
si(uno > dos) entonces                             // Si el primer valor es mayor al segundo,
    dos = uno;                                     // copia el primer valor sobre el segundo.
    finSi
mostrar << "El mayor es: " << dos << salto;           // Muestra el mayor.
pausa;                                             // Pausa antes de finalizar.
finPrincipal                                       // Fin de unidad de programa principal
```

```
Zinjal - Consola de Ejecucion
Valor 1:15.3
Valor 2:17.2
El mayor es: 17.2
En pausa. <Escape> para continuar...
```

# Decisiones...

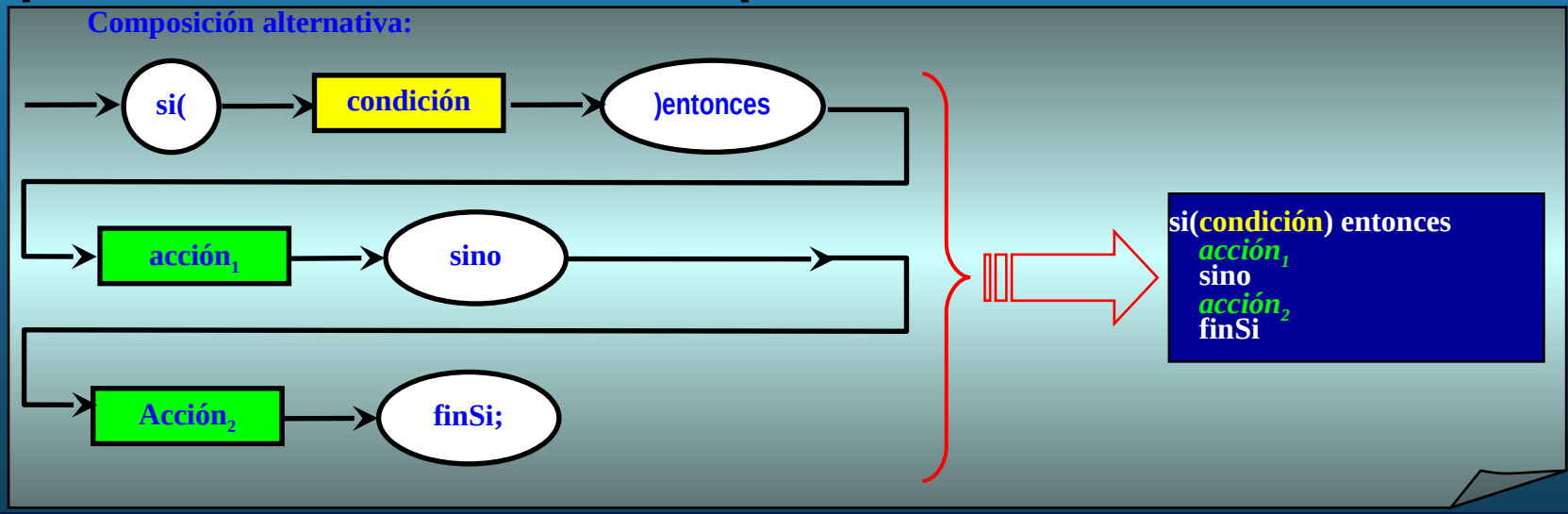
## Composición **Alternativa**

- Se ejecuta una acción siempre.
  - La **primera** si la **condición** es verdadera.
  - La **segunda** si es falsa.

# Decisiones...

## Composición **Alternativa**

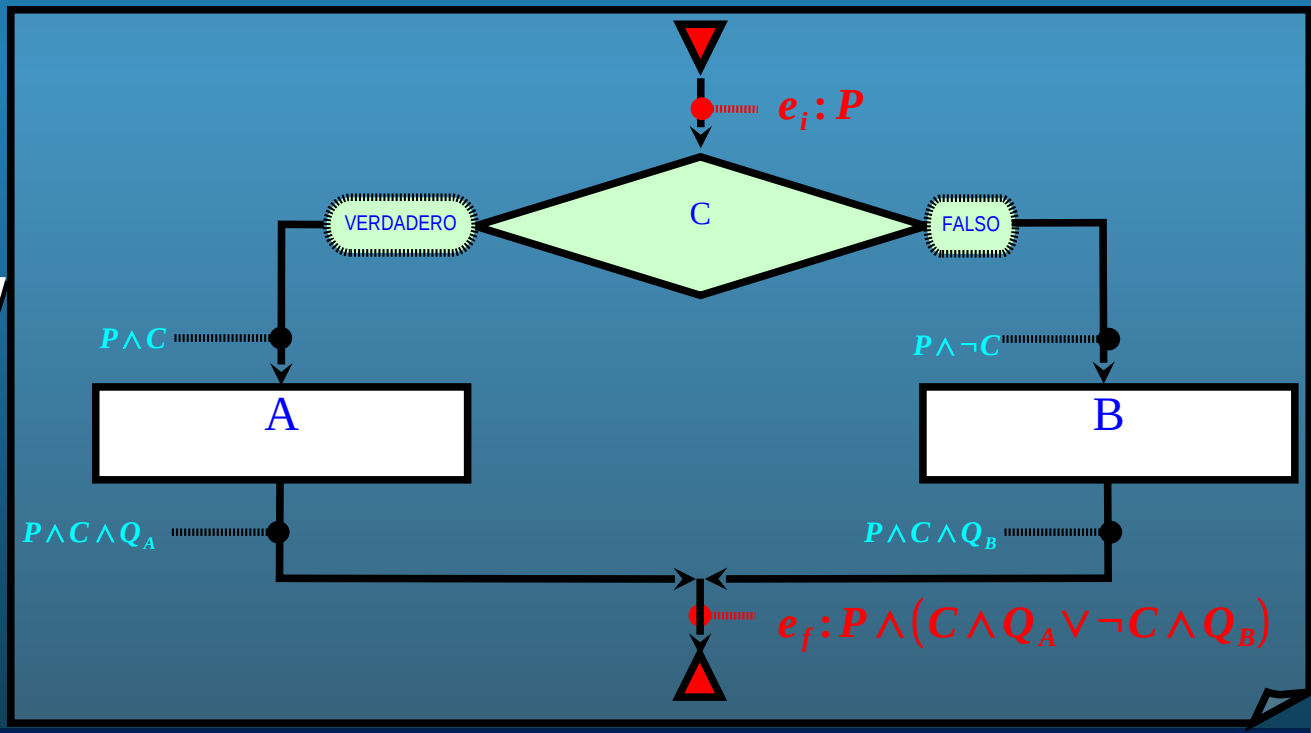
- Sintaxis:
  - La **condición** va dentro de los *paréntesis* *luego del* si, y el código de la 1ra. acción (*simple o compuesta*) va en líneas sucesivas hasta el sino, y empieza la 2da. acción, que termina en finSi.



# Decisiones...

## Composición **Alternativa**

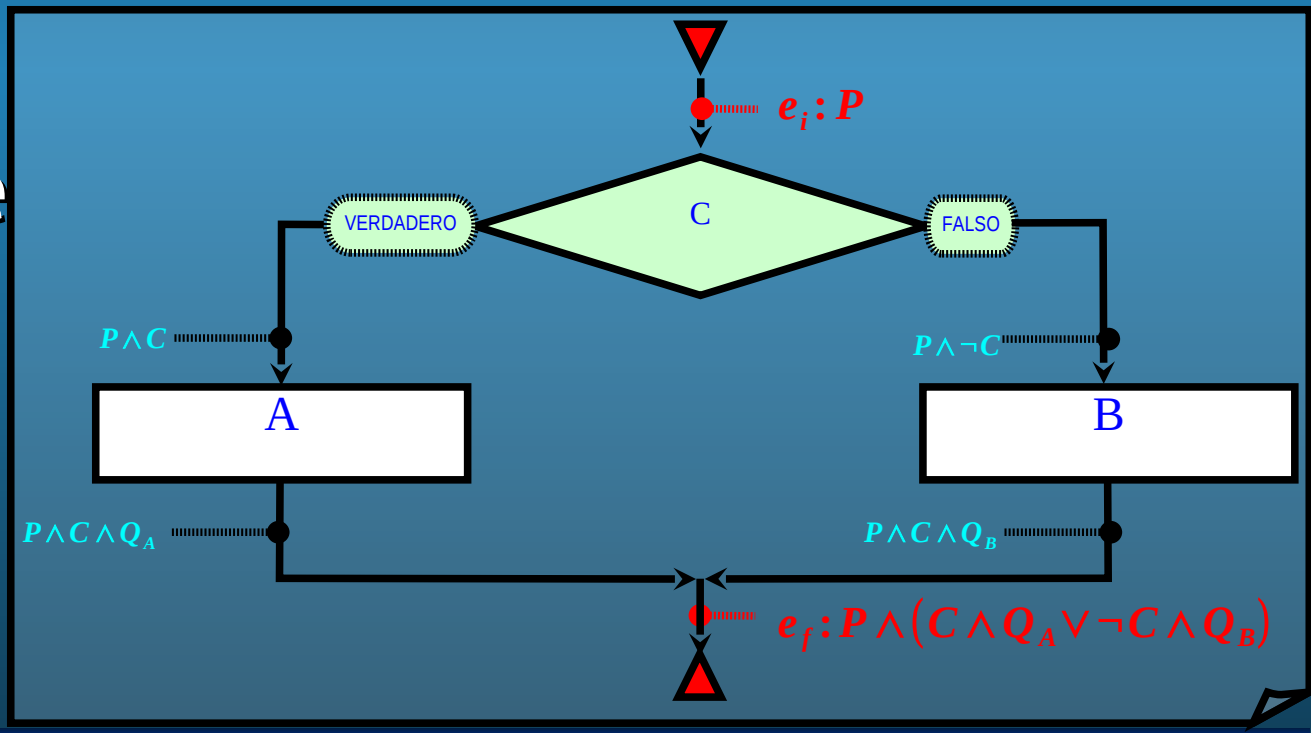
- Semántica:
  - Dadas las **precondiciones** P (*estado inicial*).
  - Al evaluar la **condición** *sucede una de dos cosas*:
    - Es verdadera y se cumplen P y C y se ejecuta la acción A.



# Decisiones...

## Composición **Alternativa**

- Semántica:
  - Dadas las **precondiciones** P (*estado inicial*).
  - Al evaluar la **condición** *sucede una de dos cosas*:
    - Es falsa y se cumplen P y ¬C y se ejecuta la acción B.



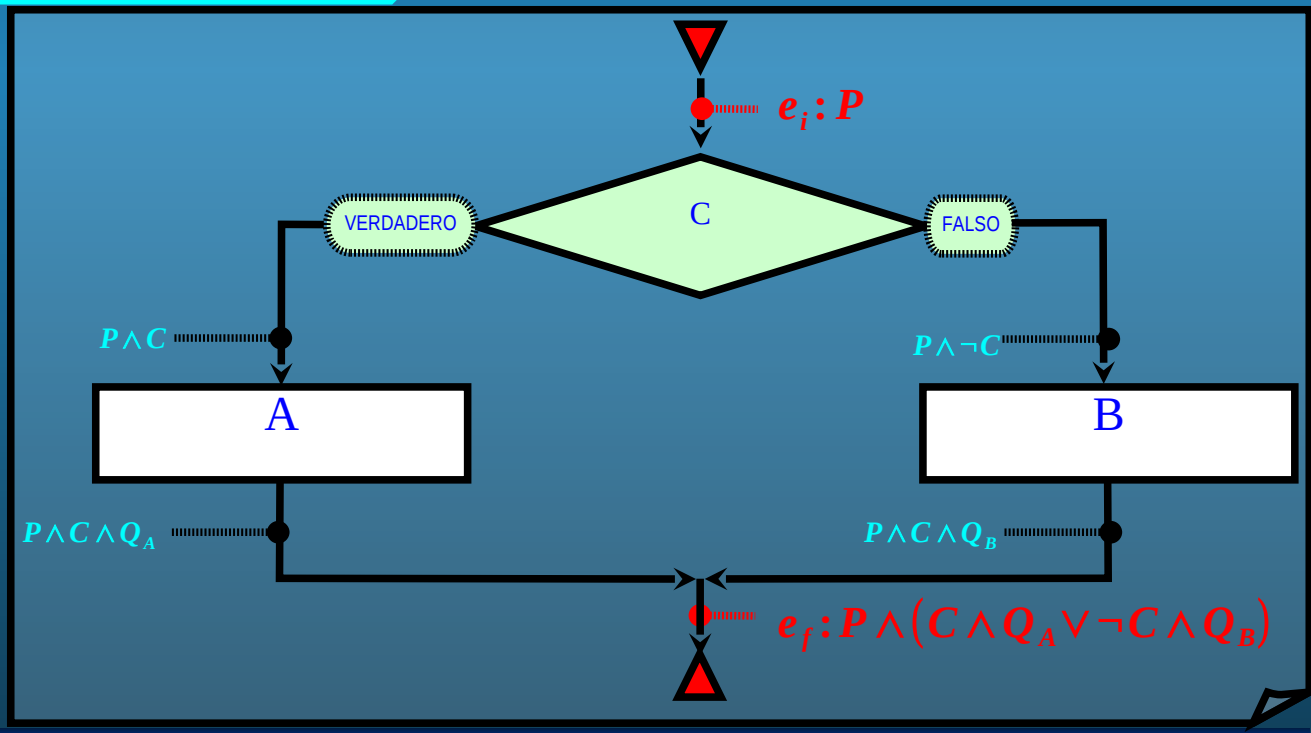
# Decisiones...

## Composición **Alternativa**

- Semántica:
  - A la salida siempre se verifica un estado final:

$$e_f : P \wedge (C \wedge Q_A \vee \neg C \wedge Q_B)$$

  - ...que se lee: "dado P, si la condición C es cierta se ejecuta la acción A, sino se ejecuta B".



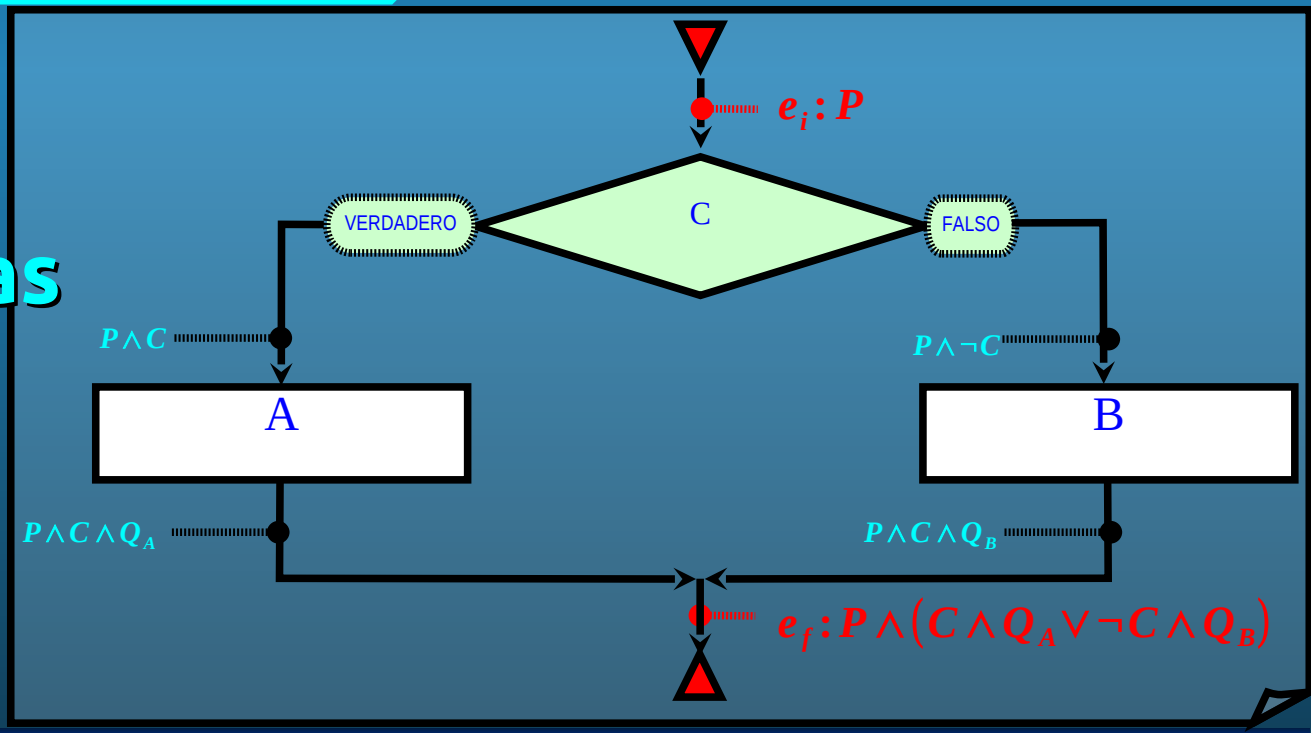
# Decisiones...

## Composición **Alternativa**

- Semántica:
  - A la salida siempre se verifica un estado final:

$$e_f : P \wedge \left( C \wedge Q_A \vee \neg C \wedge Q_B \right)$$

  - Siempre existirán **consecuencias** de una u otra acción.



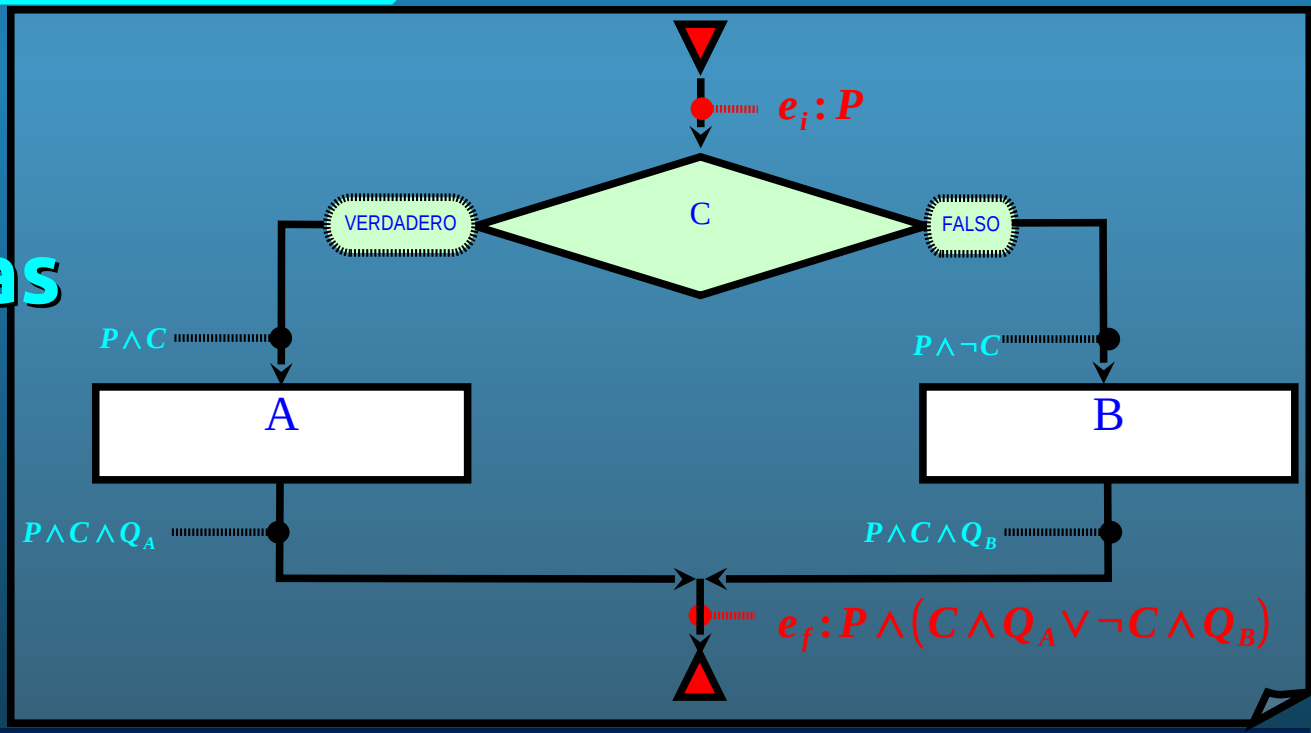
# Decisiones...

## Composición **Alternativa**

- Semántica:
  - A la salida siempre se verifica un estado final:

$$e_f : P \wedge \left( C \wedge Q_A \vee \neg C \wedge Q_B \right)$$

  - Siempre existirán **consecuencias** de una u otra acción.



# Decisiones...

## Composición Alternativa

- ¿Qué hace?

```
principal
real uno,dos;
limpiar;
leerM(uno,"Valor 1:");
leerM(dos,"Valor 2:");
si(uno > dos) entonces
    mostrar << "El mayor es: " << uno << salto;
    sino
    mostrar << "El mayor es: " << dos << salto;
    finSi
pausa;
finPrincipal
```

# Decisiones...

## Composición **Alternativa**

- **¿Qué hace?**
  - Es otro código equivalente para mostrar el mayor entre dos valores (*por la ejecución son **indistinguibles***):

```
#include <program1.h>
/**
*    Enunciado: Dados dos valores, mostrar el mayor.
*/
principal                                          // Unidad de programa principal
real uno,dos;
limpiar;                                           // Limpia la pantalla.
leerM(uno,"Valor 1:");
leerM(dos,"Valor 2:");
si(uno > dos) entonces                             // Si el primer valor es mayor al segundo,
    mostrar << "El mayor es: " << uno << salto;    // muestra el primero
    sino                                           // sino
    mostrar << "El mayor es: " << dos << salto;    // muestra el segundo.
    finSi
pausa;                                             // Pausa antes de finalizar.
finPrincipal                                       // Fin de unidad de programa principal
```

```
Zinjal - Consola de Ejecucion
Valor 1:15.3
Valor 2:17.2
El mayor es: 17.2
En pausa. <Escape> para continuar...
```