Estructuras condicionales o de selección

Jonatan Goméz Perdomo, Ph.D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D.(c)
eccubidesg@unal.edu.co

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

Agenda

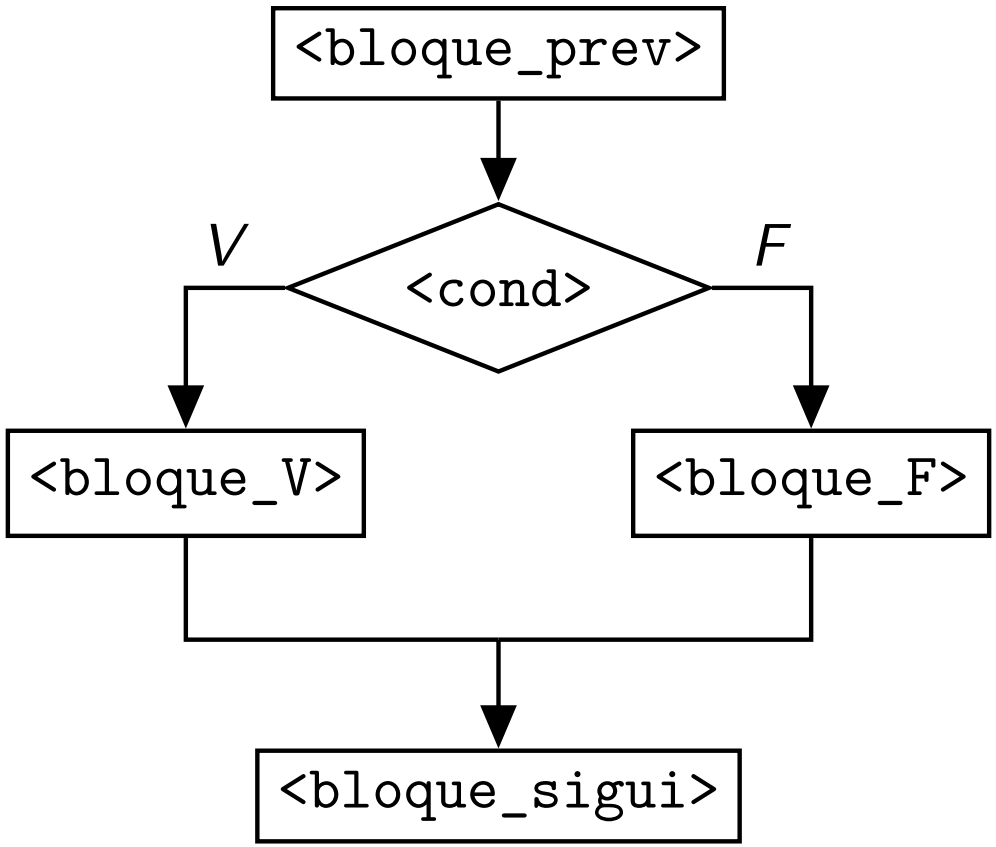
El condicional if I

Es posible tener programas en los que se deban cubrir diferentes casos, para los cuales se deberán retornar diferentes valores dadas unas condiciones.

La estructura de control condicional o de selección permite ejecutar, o un grupo de instrucciones u otro grupo si una condición se cumple o no.

Representación de condicionales con diagramas de flujos

Un condicional se puede representar gráficamente mediante un diagrama de flujo de la siguiente manera.



Esquema en de un condicional y su ejecución I

Un esquema textual que en Java representa una estructura condicional es la que se da en el siguiente fragmento de código.

```
<bloque_prev>
if(<cond>){
    <bloque_V>
}else{
    <bloque_F>
}
<bloque_sigui>
```

Donde, después de ejecutar el bloque de instrucciones previas `<bloque_prev>`, se ejecutará el bloque `<bloque_V>` si `<cond>` se evalúa verdadero, en caso de que `<cond>` se evalúe falso de ejecutará `<bloque_F>`.

Esquema en de un condicional y su ejecución II

```
<bloque_prev>
if(<cond>){
   <bloque_V>
}else{
   <bloque_F>
}
<bloque_sigui>
```

Después de ejecutar el bloque <bloque_V> o el bloque <bloque_F> se continua con la ejecución del resto del programa, es decir, el resto de las instrucciones del bloque <bloque_sigui> que aparece después de la estructura if.

Agenda

Valor absoluto de un número I

Ejemplo

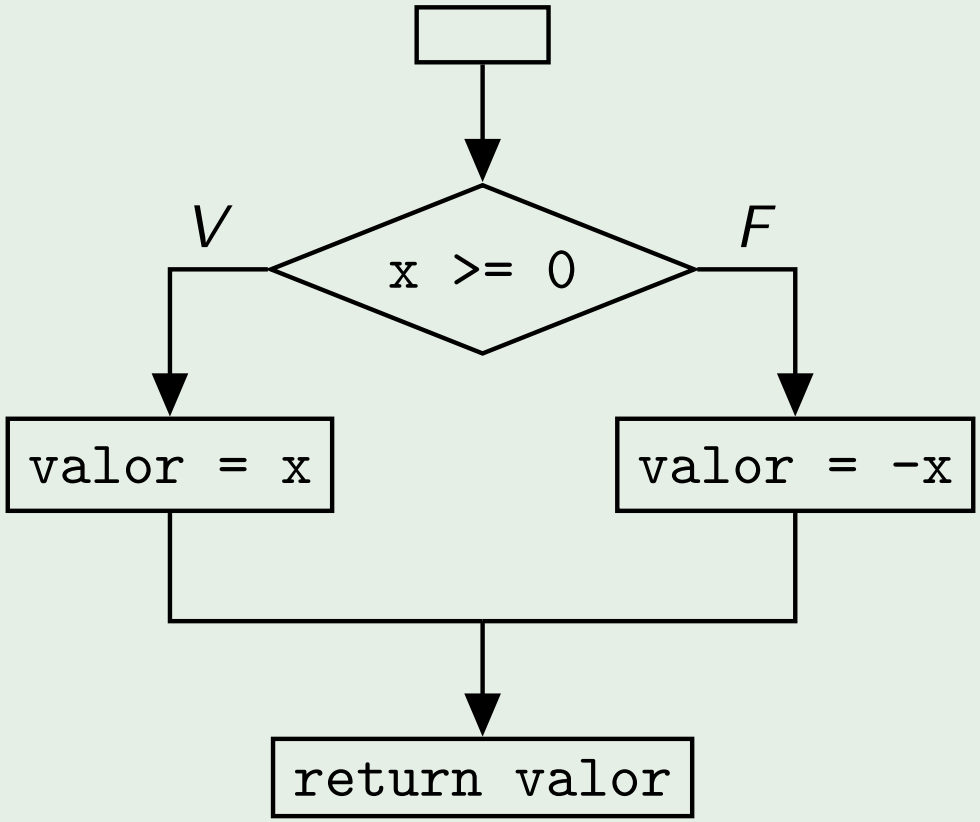
La función que permite calcular el valor absoluto de un número real es una función que recibe como parámetro de entrada un número real y retorna la distancia de ese valor al origen. La función valor absoluto en notación matemática se define como

$$\begin{aligned} \mathit{valor\_absoluto} : \mathbb{R} &\to \mathbb{R} \\ (x) &\mapsto \begin{cases} x, & \text{si } x \geq 0; \\ -x, & \text{en otro caso.} \end{cases} \end{aligned}$$

Valor absoluto de un número II

Ejemplo (continuación)

Un diagrama de flujo que representa el condicional que está implícito en la función valor absoluto es el siguiente



Valor absoluto de un número III

Ejemplo (continuación)

La codificación en Java de esta función junto con su programa principal es:

```
import java.util.Scanner;

public class Real {
```

```
    public static double valorAbsoluto(double x) {
        double valor;
        if (x >= 0) {
            valor = x;
        } else {
            valor = -x;
        }
        return valor;
    }
```

Valor absoluto de un número IV

Ejemplo (continuación)

La codificación en Java de esta función junto con su programa principal es:

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double n;
        System.out.println("n?");
        n = sc.nextDouble();
        System.out.println("El valor absoluto de "
                + n + " es " + valorAbsoluto(n));
    }
}
```

Agenda

El máximo entre dos números I

Ejemplo

Una función que permite determinar el máximo de dos números reales, se puede definir como

$$maximo\_dos\_numeros : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$$

aquí se tienen dos casos, si el número a es mayor que b el valor máximo es a; en otro caso se debe retornar b. En notación matemática esto puede ser escrito de la siguiente forma

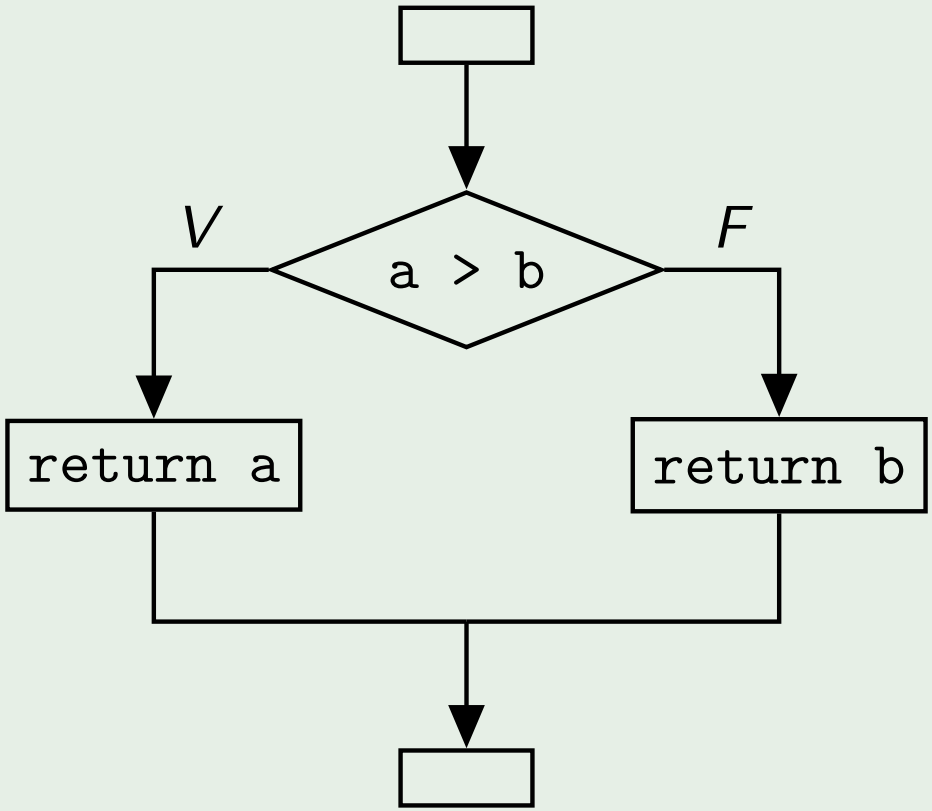
$$maximo\_dos\_numeros : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

$$(a, b) \mapsto \begin{cases} a, & \text{si } a > b; \\ b, & \text{en otro caso.} \end{cases}$$

El máximo entre dos números II

Ejemplo (continuación)

Un diagrama de flujo que representa el condicional que está implícito en la función máximo entre dos números es el siguiente



El máximo entre dos números III

Ejemplo (continuación)

La regla de traducción de la función es similar a la anterior, sólo hay que tener en cuenta la instrucción condicional, y que si no se cumple la condición especificada en el `if`, se ejecutará el flujo de instrucciones especificado bajo el alcance de la instrucción `else`

```
public static double max(double a, double b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

Agenda

El operador condicional ternario I

En Java y particularmente en las últimas décadas en muchos lenguajes de programación se dispone de un operador ternario que actúa como un condicional `if` compacto, este operador condicional `?:` es utilizado en los casos en los cuales se tienen que evaluar expresiones muy simples, de forma rápida y que usualmente retornan algún resultado, el cual depende se si la condición se evalúa *falso* o *verdadero*.

El operador condicional ternario II

A partir de la sintaxis general de un condicional `if`

```
if(<cond>){
    <bloque_1>
}else{
    <bloque_2>
}
```

se tiene que la sintaxis general de un operador condicional ternario que es equivalente al condicional `if` es la siguiente

```
<cond> ? <bloque_1> : <bloque_2>;
```

El operador condicional ternario III

es importante tener en cuenta que el operador condicional siempre retorna el resultado de la evaluación de la expresión seleccionada.

Ejemplo

La siguiente función permite calcular el valor absoluto de un número real, pero utilizando el operador condicional ternario en vez del condicional `if` presentado en un ejemplo previo

```
public static double valorAbsoluto2(double x) {
    double valor;
    valor = (x >= 0) ? x : -x;
    return valor;
}
```

El operador condicional ternario IV

Ejemplo

Una función equivalente a la anterior, que es más compacta y que también permite calcular el valor absoluto, es la siguiente

```
public static double valorAbsoluto2(double x) {
    return (x >= 0) ? x : -x;
}
```

Agenda

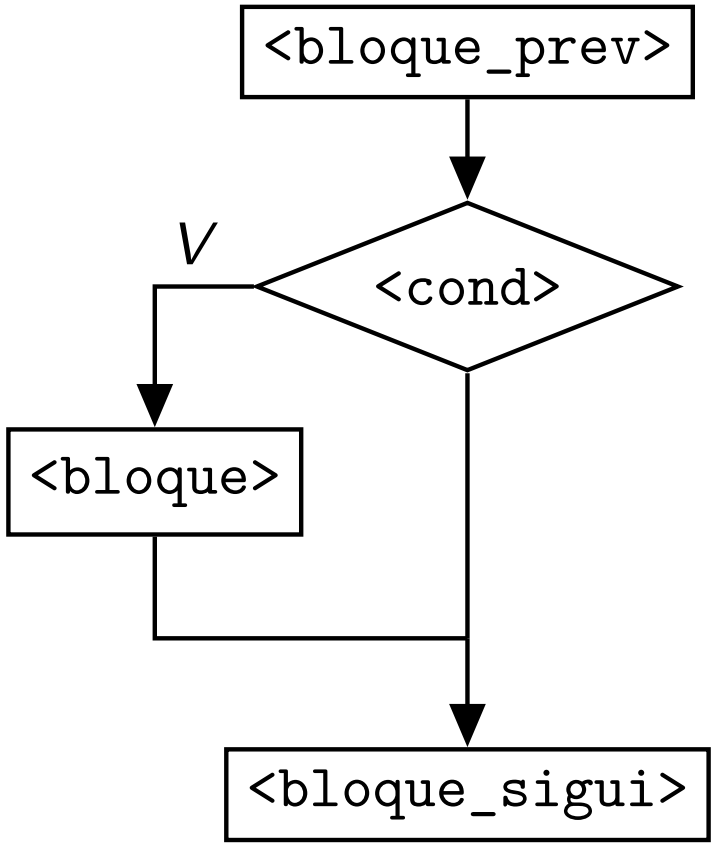
La estructura condicional sin opción alternativa I

Cuando en el flujo de un programa se desea que se ejecute un grupo de instrucciones cuando un condicional se evalué verdadero, y que se continue con la ejecución del resto del programa se haya o no ejecutado la instrucciones del condicional, se tiene un caso de condicional sin opción alternativa.

Este tipo de estructura se suele utilizar cuando se desea agregar una evaluación intermedia de una expresión cuando la condición se evalúa verdadero, pero que no tiene impacto sobre la ejecución del resto del programa que le sigue al condicional.

La estructura condicional sin opción alternativa II

Una representación mediante diagramas de flujos de un condicional sin opción alternativa es la siguiente



La estructura condicional sin opción alternativa en Java I

En un `if` la parte alternativa `else` es opcional, es decir, en el siguiente fragmento de código

```
<bloque_prev>
if(<cond>){
    <bloque>
}
<bloque_sigui>
```

se ejecutará un grupo de instrucciones `<bloque>` si `<cond>` se evalúa verdadero, en cualquier caso, se haya o no ejecutado el bloque de instrucciones `<bloque>`, se salta a la siguiente estructura `<bloque_sigui>` después del condicional `if`.

Agenda

Impresión de números con signo I

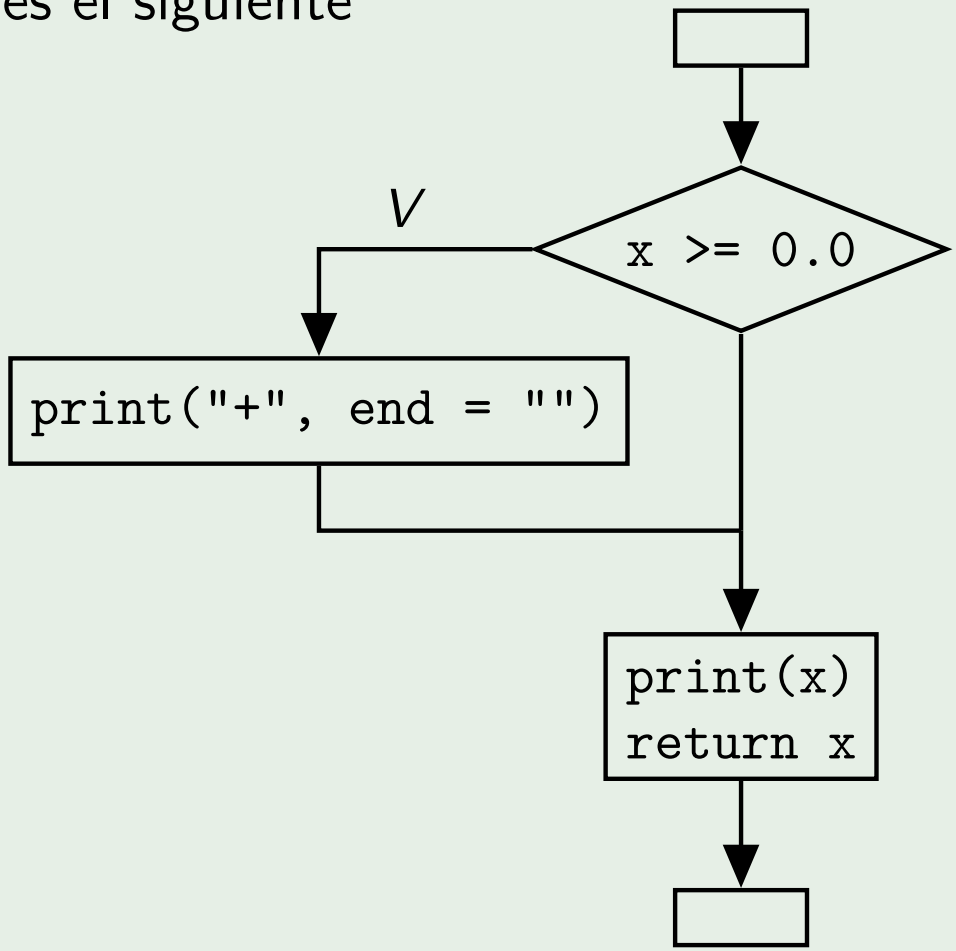
Ejemplo

Cuando se realiza la impresión de un número en la consola o cualquier otro dispositivo de salida, a veces se desea que si un número es no negativo, este sea impreso con el signo +, es decir, que para el caso del número 3.14159265 su impresión debería ser +3.14159265 en vez de su tradicional representación 3.14159265. Para el caso de los números negativos la impresión es la usual.

Impresión de números con signo II

Ejemplo (continuación)

Un diagrama de flujo que describe las instrucciones para imprimir un número con signo es el siguiente



Impresión de números con signo III

Ejemplo (continuación)

Una función en Java que permite imprimir los números con signos es la siguiente

```
public static double imprimirNumero(double x){
    if(x > 0.0){
        System.out.println("+");
    }
    System.out.println(x);
    return x;
}
```

obsérvese como la ejecución del cuerpo del condicional no afecta las instrucciones que siguen después del condicional.

Agenda

El operador lógico "condicional" I

Ejemplo

En los lenguajes de programación típicamente están definidos los operadores lógicos de la negación ($\neg$), la conjunción ($\wedge$) y la disyunción ($\vee$), pero el condicional y el bicondicional no lo están, por lo tanto si se quiere utilizar estos operadores es necesario construir las funciones que permitan utilizar estos operadores. Para el caso del condicional y a partir de la tabla de verdad para el operador condicional definido en el capítulo de lógica

$\xi(p)$	$\xi(q)$	$\xi(p \rightarrow q)$
V	V	V
V	F	F
F	V	V
F	F	V

El operador lógico "condicional" II

Ejemplo (continuación)

se puede definir una función que permite calcular la operación condicional de un par de variables booleanas y que retorna el resultado de operar los valores mediante un condicional, de la siguiente manera

$$\mathit{condicional} : \mathbb{B} \times \mathbb{B} \longrightarrow \mathbb{B}$$

aquí se tienen dos casos, primero, si el antecedente es verdadero y el consecuente es falso, entonces el resultado de aplicar el condicional el falso, para cualquier otro caso el condicional es verdadero. En notación matemática esto puede ser escrito de la siguiente manera

$$\mathit{condicional} : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$$

$$(p, q) \mapsto \begin{cases} F, & \text{si } \big(\xi(p) = V\big) \wedge \big(\xi(q) = F\big); \\ V, & \text{en cualquier otro caso.} \end{cases}$$

El operador lógico "condicional" III

Ejemplo (continuación)

Una posible codificación en de esta función seria

```
public static boolean condicional(boolean p, boolean q) {
    if (p == true && q == false) {
        return false;
    } else {
        return true;
    }
}
```

El operador lógico "condicional" IV

Ejemplo (continuación)

obsérvese que es posible construir una función que utilice sólo una estructura `if` sin la sentencia `else` que es mucho más sencilla que la función presentada anteriormente,

```
public static boolean condicional(boolean p, boolean q) {
    if (p) {
        return q;
    } else {
        return true;
    }
}
```

en esta función se tiene en cuenta que si la premisa `p` tiene valor `True`, entonces el resultado está dado por el valor de la conclusión `q`, y si el valor de `p` es `False`, entonces el condicional tendrá como valor `True`.

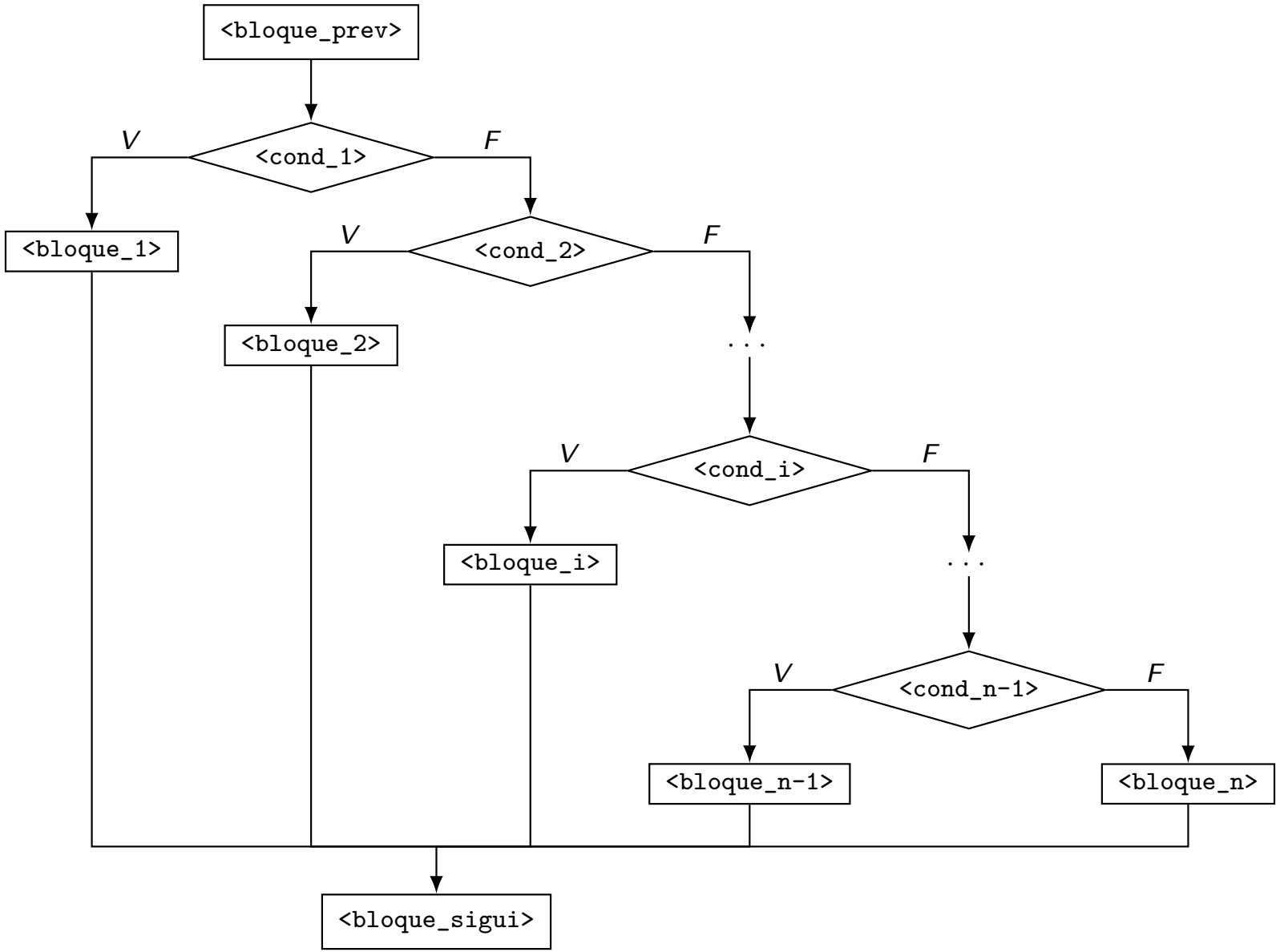
Agenda

Estructuras condicionales enlazadas I

Otra de las opciones para utilizar una estructura condicional es la de enlazar varias estructuras condicionales, de tal manera que solamente se pueda ejecutar un grupo de instrucciones dependiendo de cual de las opciones se evalúa verdadero. De la misma manera que en el caso del condicional tradicional la parte alternativa del final es opcional.

Una representación mediante diagramas de flujos de una secuencia de condicionales enlazados es la que se presenta a continuación.

Estructuras condicionales enlazadas II



Estructuras condicionales enlazadas III

La codificación en Java de las estructuras condicionales enlazadas es la siguiente

```
<bloque_prev>
if(<cond_1>){
   <bloque_1>
}else if(<cond_2>){
   <bloque_2>
}
...
}else if(<cond_i>){
   <bloque_i>
}
...
}else if(<cond_n-1>){
   <bloque_n-1>
}else{
   <bloque_n>
}
<bloque_sigui>
```

Estructuras condicionales enlazadas IV

donde después de ejecutar las instrucciones previas <bloque_prev> se ejecutará <bloque_1> si <cond_1> se evalúa verdadero, en caso de que <cond_1> se evalúe falso de ejecutará <bloque_2> si <cond_2> se evalúa verdadero, y así se continuará revisando cada una de las condiciones si la anterior se evalúa falso. Si algún <cond_i> se evalúa verdadero se ejecuta su respectivo <bloque_i> y después de ejecutar todas las instrucciones del <bloque_i> se continua ejecutando las instrucciones siguientes al condicional enlazado <bloque_sigui>.

Si ninguna <cond_i> se evalúa verdadero y la parte `else` existe al final de las estructuras `if` enlazadas entonces se ejecutarán las instrucciones del <bloque_n>

Agenda

El descuento del día I

Ejemplo

Una tienda tiene las siguientes promociones

Si un cliente lleva más de 5 productos del mismo tipo le realizan un descuento del 5%. Si lleva más de 10 productos del mismo tipo le realizan un descuento del 10%. Si lleva más de 20 productos del mismo tipo le realizan un descuento del 20%. Construya un programa que dado el número de productos y el precio de cada producto determine el valor a pagar por el cliente.

El descuento del día II

Ejemplo (continuación)

La siguiente función permitirá calcular el valor deseado

$$pago\_final(n, precio) = valor$$

Si se establecen las variables:

$n :=$ Número de cada uno de los productos

$precio :=$ Valor de cada uno de los productos

$valor :=$ Valor total a pagar despues de aplicar el descuento

entonces

El descuento del día III

Ejemplo (continuación)

$$pago\_final : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$$

$$(n, precio) \mapsto \begin{cases} n * precio, & n \leq 5; \\ n * precio * 0.95, & 5 < n \leq 10; \\ n * precio * 0.90, & 10 < n \leq 20; \\ n * precio * 0.80, & \text{en otro caso.} \end{cases}$$

El descuento del día IV

Ejemplo (continuación)

La codificación en Java de esta función es

```
public static double pagoFinal(int n, double precio) {
     double valor;
     if (n <= 5) {
     valor = n * precio;
     } else if (5 < n && n <= 10) {
     valor = n * precio * 0.95;
     } else if (10 < n && n <= 20) {
     valor = n * precio * 0.90;
     } else {
     valor = n * precio * 0.80;
     }
     return valor;
}
```

El descuento del día V

Ejemplo (continuación)

Otra posible escritura de la función puede ser

```
public static double pagoFinal(int n, double precio) {
    if (n <= 5) {
        return n * precio;
    } else if (5 < n && n <= 10) {
        return n * precio * 0.95;
    } else if (10 < n && n <= 20) {
        return n * precio * 0.90;
    } else {
        return n * precio * 0.80;
    }
}
```

El descuento del día VI

Ejemplo (continuación)

Si se invoca la función anterior con los siguientes argumentos

n	precio
4	10000
8	10000
15	10000
25	10000

y si se imprimen los resultados mediante las instrucciones de la izquierda, entonces los resultados que se obtiene son los presentados a la derecha

```
pago_final(4,10000)
pago_final(8,10000)
pago_final(15,10000)
pago_final(25,10000)
```

```
40000
76000.0
135000.0
200000.0
```

Agenda

La estructura de conmutación (switch) I

Ejemplo (continuación)

Cuando se tiene una instrucción `if` enlazada en la cual las condiciones consisten en comparar una misma variable con un grupos de valores enteros como la que se presenta a continuación:

```
if(<var_entera> == <num_1>){
   <bloque_1>
}else if(<var_entera> == <num_2>){
   <bloque_2>
}
   ...
   ...
}else if(<var_entera> == <num_n-1>){
   <bloque_n-1>
}else{
   <bloque_n>
}
```

La estructura de conmutación (switch) II

Ejemplo (continuación)

En este caso se utiliza una instrucción `switch`, la cual sirve para abreviar una instrucción `if` enlazada como la anterior:

```
switch(<var_entera>){
   case <num_1>:
      <bloque_1>
   break;
   case <num_2>:
      <bloque_2>
   break;
     ...
     ...
   case <num_n-1>:
      <bloque_n-1>
   break;
   default:
      <bloque_n>
   break;
}
```

Detección de vocales minúsculas I

Ejemplo (continuación)

La siguiente función permite determinar si dada una letra, ésta es una vocal minúscula, en caso de no ser así, entonces por defecto se retorna *falso*.

$$es\_vocal\_minuscula : \mathbb{ASCCII} \to \mathbb{B}$$

$$(ch) \mapsto \begin{cases} V, & \text{si } ch = \texttt{a}; \\ V, & \text{si } ch = \texttt{e}; \\ V, & \text{si } ch = \texttt{i}; \\ V, & \text{si } ch = \texttt{o}; \\ V, & \text{si } ch = \texttt{u}; \\ F, & \text{en otro caso.} \end{cases}$$

Detección de vocales minúsculas II

Ejemplo (continuación)

```
public static boolean esVocalMinuscula(char ch) {
    boolean value;
    switch (ch) {
        case 'a':
            value = true;
            break;
        case 'e':
            value = true;
            break;
        case 'i':
            value = true;
            break;
```

Detección de vocales minúsculas III

Ejemplo (continuación)

```
            case 'o':
                value = true;
                break;
            case 'u':
                value = true;
                break;
            default:
                value = false;
                break;
        }
        return value;
    }
```

Detección de vocales minúsculas III

Cuando en una instrucción `switch` varios casos se tratan de la misma forma, entonces estos se pueden agrupar en uno o varios casos:

Ejemplo (continuación)

```
public static boolean esVocalMinuscula(char ch) {
    boolean value;
    switch (ch) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            value = true;
            break;
        default:
            value = false;
            break;
    }
    return value;
}
```

Detección de vocales minúsculas IV

En este ejemplo se agruparon todos lo casos de las vocales en uno solo y se retornó directamente el resultado de la evaluación de cada caso:

Ejemplo (continuación)

```
public static boolean esVocalMinuscula(char ch) {
    switch (ch) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            return true;
        default:
            return false;
    }
}
```

Problemas varios I

Problemas

1. Dado un número entero, determinar si ese número corresponde al código ASCII de una vocal minúscula.
2. Dada una cadena de longitud 1, determine si el código ASCII de primera letra de la cadena es par o no.
3. Dado un carácter, construya un programa en para determinar si el carácter es un dígito o no.

Problemas varios II

Problemas

4. Dado un número real x, construya una función que permita determinar si el número es positivo, negativo o cero. Para cada caso de debe imprimir el texto que se especifica a continuación:

   Positivo: "`El número` x `es positivo`"
   Negativo: "`El número` x `es negativo`"
   Cero (0)**:** "`El número` x `es el neutro para la suma`"

5. Dado el centro y el radio de un círculo, determinar si un punto de $\mathbb{R}^2$ pertenece o no al interior del círculo.

6. Dadas tres longitudes positivas, determinar si con esas longitudes se puede construir un triángulo.