



Instituto Politécnico Nacional
La Técnica al Servicio de la Patria

**Unidad Profesional Interdisciplinaria de
Ingeniería Campus Tlaxcala UPIIT**

Fundamentos de Programación

Esaú Eliezer Escobar Juárez

Ingeniería en Inteligencia Artificial (IIA)

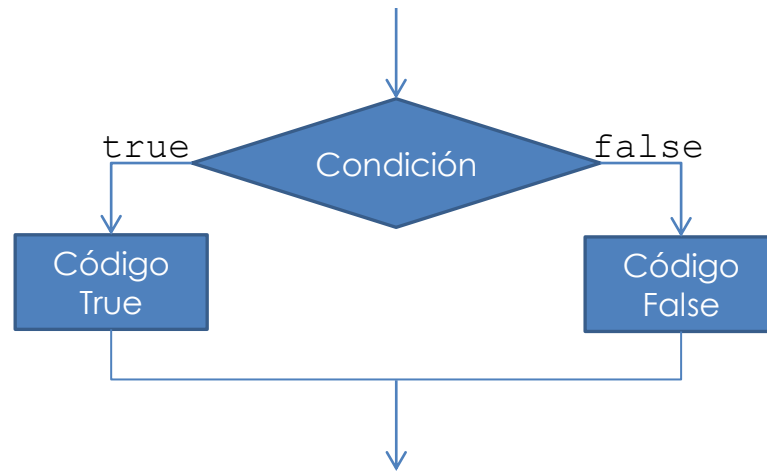
Operadores relacionales

- Comparaciones (condiciones)
 - Expresión Op_relacional Expresión
- Concordancia de tipo entre expresiones
- Resultado: true o false

Algebraico	Leng. C	Ejemplo	Significado
Operadores de igualdad			
=	==	x == y	x es igual que y
≠	!=	x != y	x no es igual que y
Operadores de relación			
>	>	x > y	x es mayor que y
<	<	x < y	x es menor que y
≥	>=	x >= y	x es mayor o igual que y
≤	<=	x <= y	x es menor o igual que y

Estructuras de control

Bifurcación Condicional



```
if(condición) {  
    → códigoTrue  
}
```

```
else{  
    → códigoFalse  
}
```

Opcional: puede no haber else

```
int num;  
printf("Número: ");  
scanf("%d",&num);  
if(num % 2 == 0){  
    printf("%d es par",num);  
}  
else{  
    printf("%d es impar",num);  
}
```

Bifurcación Condicional

Operador condicional ?

Condición ? códigoTrue : códigoFalse;

```
if(num < 0){  
    printf("negativo");  
}  
else{  
    printf("positivo");  
}
```

```
num<0 ? printf("negativo") : printf("positivo");
```

Bloques de código

Entre llaves

```
{  
  instrucción1  
  instrucción1  
  ...  
  instrucción1  
}
```

Tab ó
3 esp. →

Posición

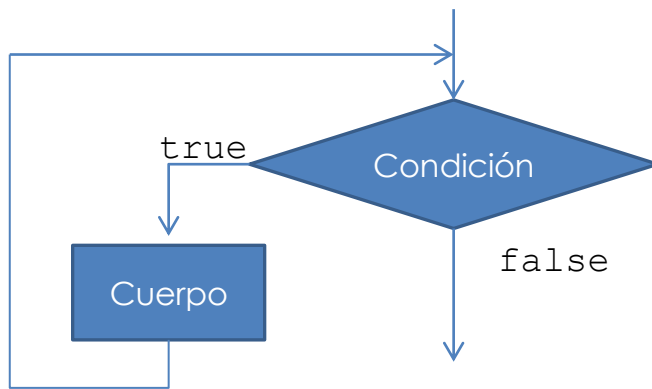
```
if(num > 0)          if(num > 0){  
{                    printf("Positivo");  
  printf("Positivo"); }  
}
```

- No necesitamos llaves si sólo hay una instrucción
- No poner el if y la instrucción objetivo en la misma línea:

```
if(num > 0) printf("Positivo");
```

Estructuras de control

Bucle While



```
while (condición) {  
    → cuerpo  
}
```

Si la condición es false al empezar, no se ejecuta el cuerpo ninguna vez

While

- ¿Cómo le hacemos si no sabemos cuantas repeticiones hacer?
- Controlar las repeticiones con un centinela.
- El centinela debe ser un valor claramente diferente de los valores posibles de operación.

Tipos float

- Especificar precisión en printf:

```
printf("2 digitos: %.2f, 4 digitos:  
%.4f\n", x);
```

Por default es precisión de 6 dígitos: %.6f

Estructuras de control

Bucle for (while controlado por repetición)

```
for (inicialización; condición; paso ) {  
    cuerpo  
}
```

Equivalencia

```
Inicialización;  
while ( condición ) {  
    cuerpo  
    paso;  
}
```

Diagrama de anotación para el bucle `for`:

`for(contador = 1; contador <= 10; ++contador)`

- Palabra reservada for**: apunta a `for`
- Nombre de la variable de control**: apunta a `contador`
- Valor inicial de la variable de control**: apunta a `= 1`
- Condición de continuación de ciclo**: apunta a `contador <= 10`
- Valor final de la variable de control con el que la condición es verdadera**: apunta a `10`
- Incremento de la variable de control**: apunta a `++contador`

for (consideraciones)

- Dentro de los componentes podemos tener expresiones

```
int x = 2, y = 10;  
for(j=x; j<=4*x*y; j += y/x)
```

```
for(j=2; j<=80; j+=5)
```

- El incremento puede ser negativo
- Si al inicio la condición es falsa, el ciclo no se ejecuta.
- La variable de control puede no utilizarse en el cuerpo del ciclo.
- *Bucles infinitos*

```
for (int i = 1; i <= 100; i--) ...
```

1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 ...

Cada vez más lejos del valor final (100)

Es un error de diseño/programación

Ámbito de la variable contadora

- *Declarada en el propio bucle*

```
for (int i = 1; ...)
```

Sólo se conoce en el cuerpo del bucle (su ámbito)

No se puede usar en instrucciones que sigan al bucle

- *Declarada antes del bucle*

```
int i;
```

```
for (i = 1; ...)
```

Se conoce en el cuerpo del bucle y después del mismo

Ámbito externo al bucle

for vs while

- Los bucles for se pueden reescribir como bucles condicionados

```
for (int i = 1; i <= 100; i++) cuero
```

Es equivalente a:

```
int i = 1;  
while (i <= 100) {  
    cuero  
    i++;  
}
```

- La inversa no es siempre posible:

```
int i;  
scanf("%d", &i);  
while (i != 0) {  
    cuero  
    scanf("%d", &i);  
}
```

¿Bucle for equivalente?

No sabemos cuantos números
Ingresará el usuario

Switch

• La escala if – else

```
if (num == 3) {  
    printf("alto\n");  
}  
else{  
    if (num == 2) {  
        printf("medio\n");  
    }  
    else{  
        if (num == 1) {  
            printf("bajo\n");  
        }  
        else{  
            printf("Valor invalido\n");  
        }  
    }  
}
```



```
if (num == 3) {  
    printf("alto\n");  
}  
else if (num == 2) {  
    printf("medio\n");  
}  
else if (num == 1) {  
    printf("bajo\n");  
}  
else {  
    printf("Valor invalido\n");  
}
```

Switch

- Selección entre posibles valores de una expresión

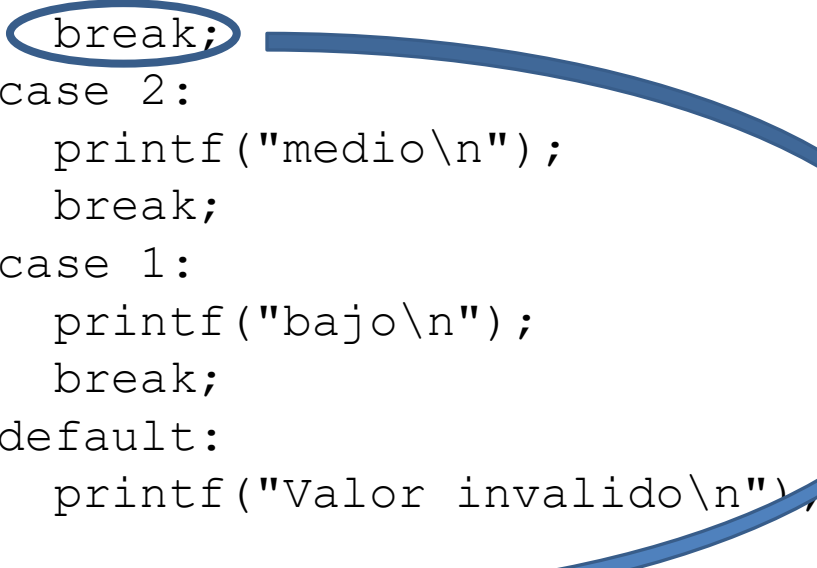
```
switch (expresión) {  
    case constante1:  
        código1  
        [break;]  
    case constante2:  
        código2  
        [break;]  
    ...  
    case constanteN:  
        códigoN  
        [break;]  
    [default:  
        códigoDefault  
    ]  
}
```

```
switch(num) {  
    case 3:  
        printf("alto\n");  
        break;  
    case 2:  
        printf("medio\n");  
        break;  
    case 1:  
        printf("bajo\n");  
        break;  
    default:  
        printf("Valor invalido\n");  
}
```

Switch


- La instrucción break interrumpe el switch

```
switch(num) {  
    case 3:  
        printf("alto\n");  
        break;  
    case 2:  
        printf("medio\n");  
        break;  
    case 1:  
        printf("bajo\n");  
        break;  
    default:  
        printf("Valor invalido\n");  
}
```



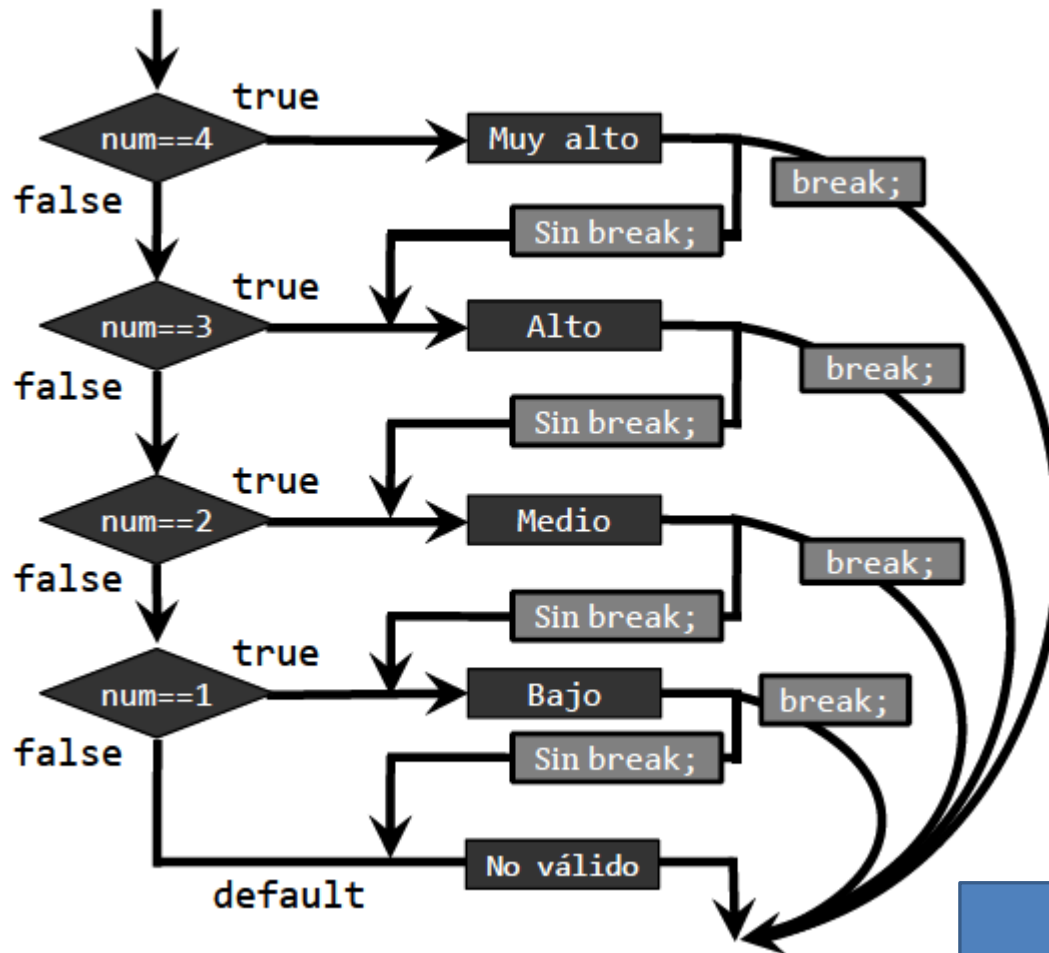
Switch

```
switch(num) {  
    case 3:  
        printf("alto\n");  
    case 2:  
        printf("medio\n");  
    case 1:  
        printf("bajo\n");  
    default:  
        printf("Valor invalido\n");  
}
```



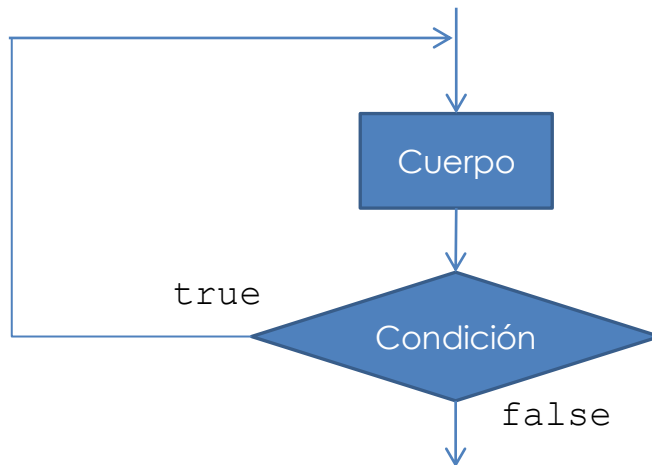
Switch

- Con o sin break



Estructuras de control

Bucle Do-While



```
do {  
    → cuerpo  
} while (condición);
```

Si la condición es false, al menos se ejecuta el cuerpo una vez

Estructuras de control

break

- Permite salir del bloque de código de ciclos y del switch.

continue

- Obliga a continuar con la siguiente iteración en los ciclos, dejando sin ejecución lo que aparezca debajo de la instrucción en esa iteración.

Operadores lógicos

- Usados para crear condiciones compuestas

Si es mujer **y** tiene mas de 65 años entonces incrementar la variable mujerTerceraEdad

```
if (genero ==1 && edad>=65)
    mujerTerceraEdad++;
```

Si el alumno tiene un promedio mayor a 9 en el semestre **o** tiene una calificación mayor a 9 en el examen entonces la calificación es 9.

```
if (promedioSemestral >= 9 || examenFinal >= 9)
    printf("La calificación es 9\n");
```

Operadores lógicos

	!
true	false
false	true

NO (not)
!

	!
1	0
0	1

&&	true	false
true	true	false
false	false	false

Y (and)
&&

&&	1	0
1	1	0
0	0	0

	true	false
true	true	true
false	true	false

O (or)
||

	1	0
1	1	1
0	1	0