Sentencias

- Una sentencia es una acción a realizar, como por ejemplo calcular un valor.
- C++ define sentencias simples, también llamadas sentencias expresión y sentencias compuestas, también llamadas bloques.
- Una sentencia simple es una expresión seguida de un `;` (que la "convierte" en sentencia)

```
x = 0;
i++;
```

- Un bloque es un conjunto de sentencias dentro de un par de llaves `{}`

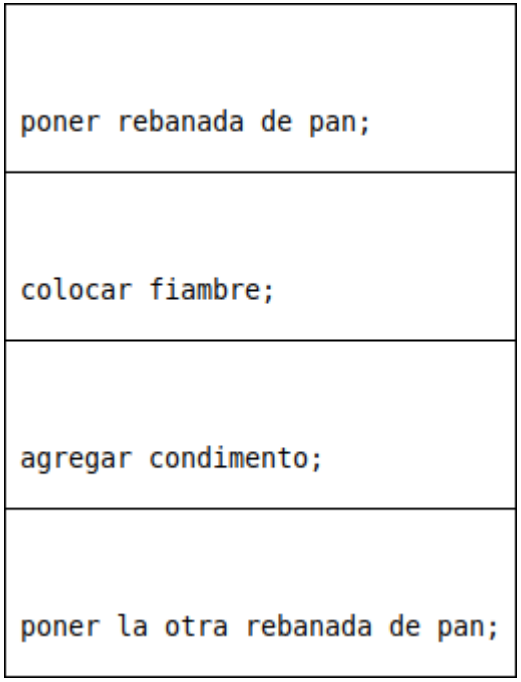
```
{
        x = 0;
        y = 2 * x;
}
```

Estructuras de control

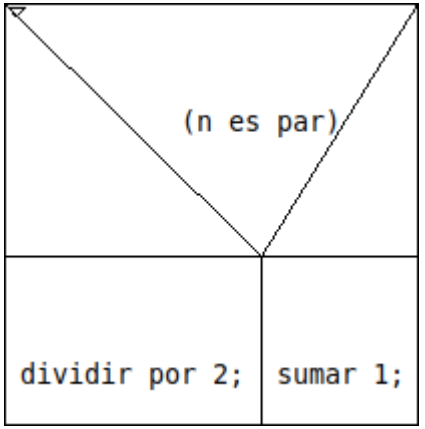
- Existen 3 estructuras de control
  - Secuencial: una sentencia tras otra
  - Selección: el grupo de sentencias a ejecutar depende de una condición
  - Iteración: se repite una o más sentencias una cantidad de veces
- C++ posee sentencias que implementan las estructuras de selección e iteración

Diagramas Nassi Shneiderman

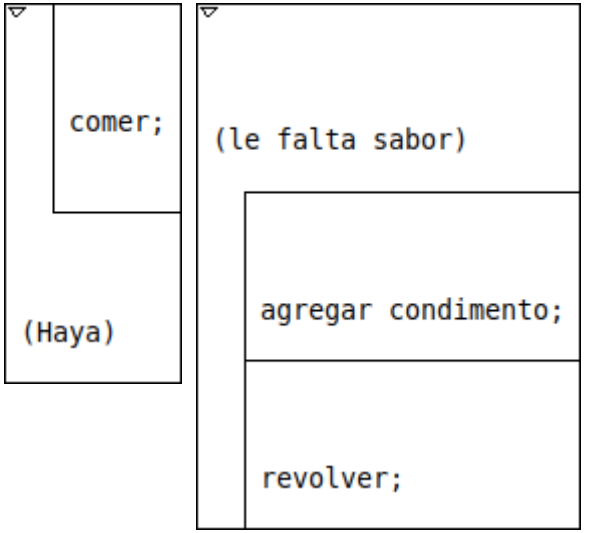
Secuencial



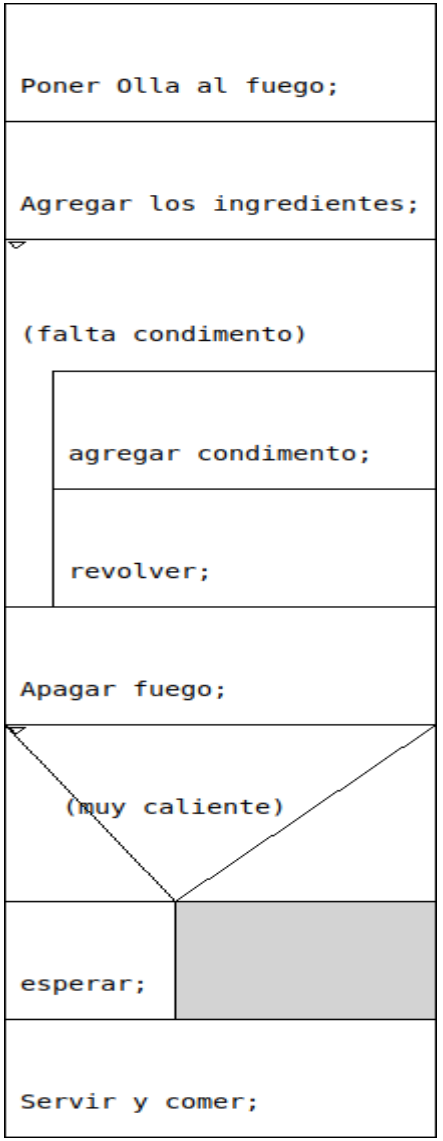
Selección



Iteración



Ejemplo “Guiso”



Selección

```
if (condición)
        sentencia por verdadero
else
        sentencia por falso
```

- El else es optativo
- Cada else se “acopla” al if más cercano

El if (a>b) es el más cercano

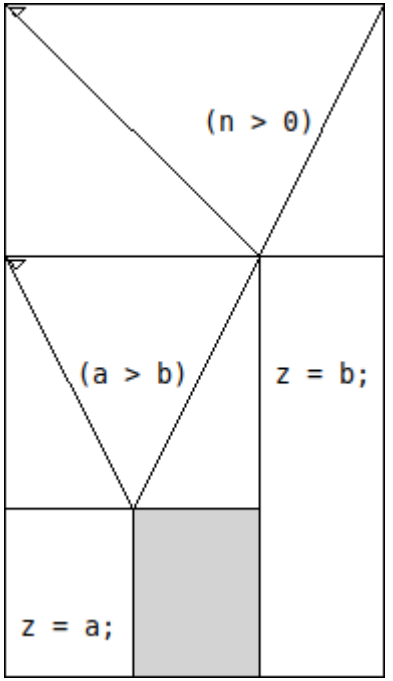
```
if (n > 0)
        if (a > b)
                z = a;
        else
                z = b;
```

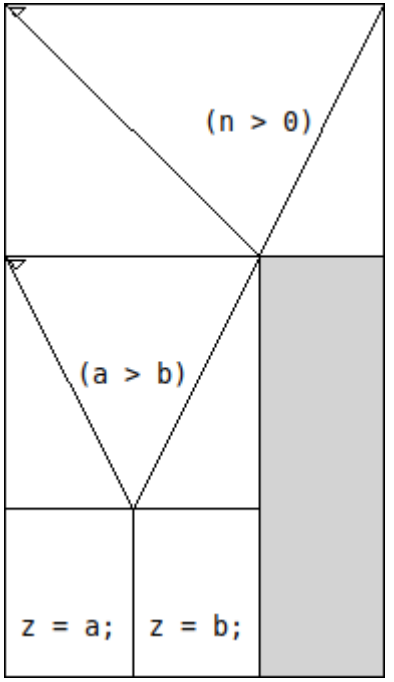
Para acoplar al primer if

```
if (n > 0) {
        if (a > b)
                z = a;
} else {
          z = b;
}
```

Diagrama para selección

```
if (n > 0)
        if (a > b)
                z = a;
        else
                z = b;
```






```
if (n > 0) {
        if (a > b)
                z = a;
} else {
        z = b;
}
```

Condiciones anidadas

Se puede encadenar if – else if de modo tal de lograr una selección por varios casos. Donde entra en la primera que cumple la condición (y solo en esa)

Genéricamente

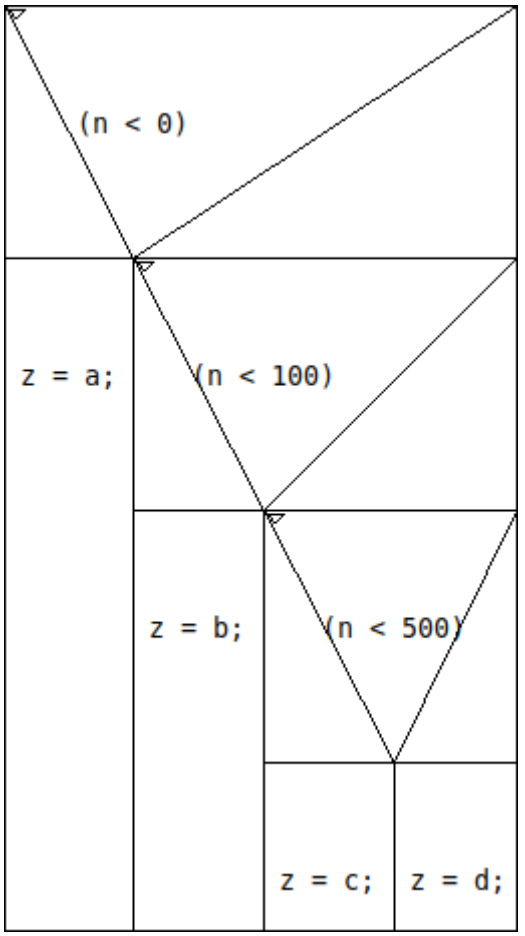
```
if (condición1)
        sentencia1
else if (condición2)
        sentencia2
else if (condición3)
        sentencia3
else
        sentencia por casos restantes
```

Ejemplo

```
if (n < 0)
        z = a;
else if (n < 100)
        z = b;
else if (n < 500)
        z = c;
else
        z = d;
```

Diagrama de condiciones anidadas

```
if (n < 0)
        z = a;
else if (n < 100)
        z = b;
else if (n < 500)
        z = c;
else
        z = d;
```



Selección múltiple

Forma genérica

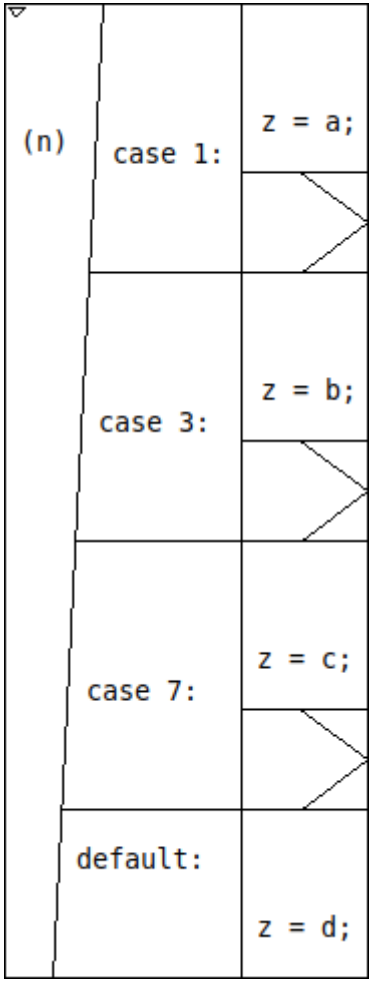
```
switch (expresión) {
case etiqueta1:
       sentencias1
       break;
case etiqueta2:
       sentencias2
// Sin break continúa la ejecución
case etiqueta3:
       Sentencias3
       break;
default:
        sentencias caso contrario
}
```

Ejemplo

```
switch (n) {
case 1:
       z = a;
       break;
case 3:
       z = b;
       break;
case 7:
       z = c;
       break;
default:
       z = d;
}
```

Diagrama de selección múltiple

```
switch (n) {
case 1:
        z = a;
        break;
case 3:
        z = b;
        break;
case 7:
        z = c;
        break;
default:
        z = d;
}
```



Operador ? :

Evalúa la condición, según el resultado evalúa solo la expresión que corresponde y ese es el resultado devuelto:

```
condición ? expresión-si-verdadero : expresión-si-falso
```

Ejemplo:

```
x = n > 100 ? y+2 : z*3;
```

Equivale a:

```
if (n > 100)
        x = y+2;
else
        x = z*3;
```

```
max = a > b ? a : b;
```

```
if (a > b)
        max = a;
else
        max = b;
```

Ciclo Mientras

Itera cero o más veces (eventualmente infinitas)

```
while (condicion sea verdadera)
    sentencia
```

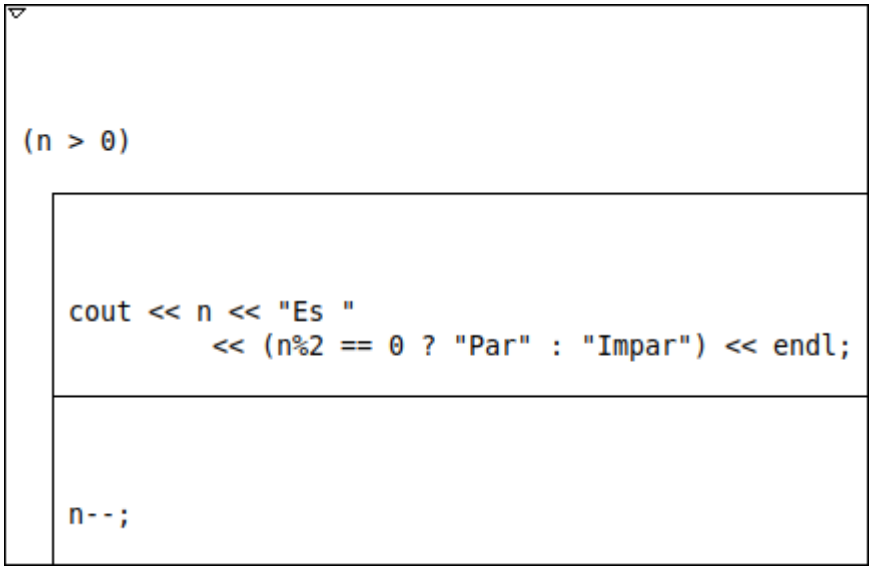
Ejemplos

```
while (n > 0) {
    cout << n << "Es "
         << (n%2 == 0 ? "Par" : "Impar") << endl;
    n--;
}

while (a > b)
    b *= 2;
```

Diagrama mientras

```
while (n > 0) {
    cout << n << "Es "
         << (n%2 == 0 ? "Par" : "Impar") << endl;
    n--;
}
```



Ciclo hacer mientras

Itera una o más veces (eventualmente infinitas)

```
do
    sentencia
while (condicion sea verdadera);
```

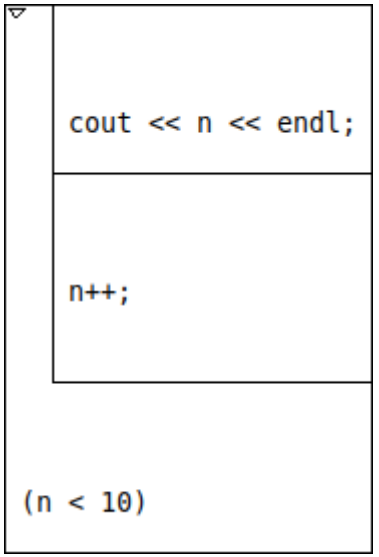
Ejemplos

```
do {
    cout << n << endl;
    n++;
} while (n < 10);

do
    cout << n << endl;
while (++n < 10);
```

Diagrama hacer mientras

```
do {
     cout << n << endl;
     n++;
} while (n < 10);
```



Ciclo Para

Forma general

```
for (expresión-antes; condición; expresión-final)
    sentencia
```

Es equivalente a

```
expresión-antes;
while (condición) {
    sentencia
    expresión-final;
}
```

Ejemplo ciclo Para

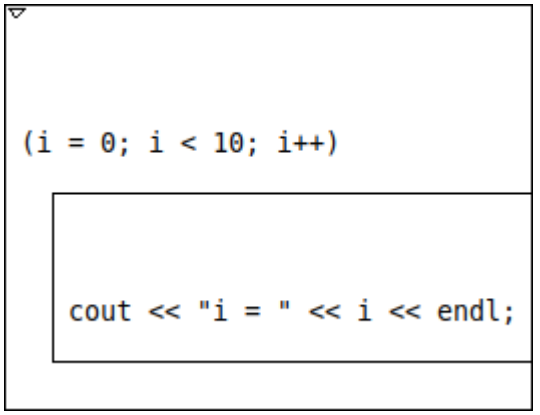
```
for(i = 0; i < 10; i++)
        cout << "i = " << i << endl;
```

Es equivalente a

```
i = 0;
while (i < 10) {
        cout << "i = " << i << endl;
        i++;
}
```

Diagrama ciclo para

```
for(i = 0; i < 10; i++)
        cout << "i = " << i << endl;
```



Ejemplo ciclo Para

```
for(n = 100,j = 2; j < n; n -= j, j += 2) {
        cout << "n = " << n << " j = " << j << endl;
}
```

Es equivalente a

```
n = 100;
j = 2;
while (j < n) {
        cout << "n = " << n << " j = " << j << endl;
        n -= j;
        j += 2;
}
```

break y continue

- Ambos sirven para cortar la secuencia normal de ejecución de un ciclo iterativo, es decir: **while**, **do while** y **for**.
- El **break**, como ya vimos también se puede utilizar en un **switch**
- El **break** sale del ciclo o el switch en el que se encuentra, finalizándolo y pasando a la sentencia siguiente
- El **continue** sale solamente de la iteración actual y va al punto de evaluación para comprobar si sigue o no iterando.
  - En el caso del **for** también evalúa la expresión final antes de evaluar la condición

Licencia