Sesión 3: Estructura de Control Selectiva

Contenidos

a) Tipo Boolean. Variables y expresiones booleanas
b) Proposición if
c) Proposición if else
d) Ejercicios

Aprendizajes esperados

- Aprende a usar en programación las instrucciones que permiten evaluar las condiciones lógicas.

¿Qué son las estructuras control?

- Los programas vistos hasta ahora consisten de sólo una lista de comandos que se ejecutan de forma ordenada.
  - El programa no puede elegir si realiza o no algún comando.
  - El programa no puede realizar el mismo comando más de una vez.
  - Tales programas son extremadamente limitados!
- Las Estructuras de Control permiten a un programa basar su comportamiento en función al valor que tomen sus variables.

Solo para programadores C++

- Los tipos de sentencias son prácticamente idénticos tanto en C como en C++.
- Principal diferencia: las condiciones true/false deben ser booleanas, no numéricas!

El tipo boolean

- boolean es uno de los ocho tipos primitivos.
  - Los booleans se usan para tomar decisiones si/no.
  - Todas las estructuras de control usan booleans.
- Hay exactamente dos valores booleanos : true (“yes”) y false (“no”)
  - boolean, true, y false se escriben en minúsculas.
- El término booleans va en honor a George Boole, el fundador de la lógica Booleana.

Declarando variables booleanas

- Las variables boolean se declaran como cualquier otro tipo de variable:

```
boolean casado;
boolean aprobado;
boolean tareaCompleta = false;
```

- Los valores boolean pueden ser asignados a variables booleanas:

```
tareaCompleta= true;
```

Expresiones booleanas

- Una expresión booleana es aquella que después de evaluarse puede tomar uno de dos posibles resultados: verdadero o falso.

- En lenguaje C++, los valores verdadero y falso se representan, respectivamente, con los valores true y false.

Operadores de comparación

- Una manera de formar expresiones booleanas es utilizando los operadores de comparación:

a == b	- igualdad
a < b	- menor a
a > b	- mayor a
a != b	- distinto de
a <= b	- menor o igual a
a >= b	- mayor o igual a

Comparaciones Numéricas

- Las siguientes comparaciones numéricas dan un resultado boolean:

```
x < y     // es x menor que y?
x <= y    // es x menor o igual a y?
x == y    // es x igual a y? (no usar  = )
x != y    // es x diferente a y?
x >= y    // es x  mayor  o  igual  a  y?
x > y     // es x  mayor que y?
```

- Recuerde: No usar == o != para números punto flotante.

La proposición if

- La proposición if tiene la forma:

  if (*expresion-booleana*) *sentencia*

- Ejemplos:

```
if (pasedeCurso)  cout<<“Lo Máximo!";
if (x > maximo) maximo = x;
if (precioLibro < 40.00) descu=10;
```

- La sentencia if controla *una* sentencia
  - Sin embargo muchas veces queremos controlar un *grupo de* sentencias.

Sentencias Compuestas

- Podemos usar llaves para agrupar varias sentencias en una sentencia “compuesta” :

  { *sentencia*; *sentencia*; …; *sentencia*; }

- Las llaves pueden agrupar cualquier número de sentencias:

```
{ }          // OK--sentencia “vacia”
{ x = 0; }  // OK—las llaves no son necesarias
{ temp = x;
  x = y;
  y = temp; }   //uso tipico
```

- La sentencia compuesta es el único tipo de sentencia que no finaliza en punto y coma.

Estructuras de decisión

- Una de las principales aplicaciones de las expresiones booleanas es la toma de decisiones:

```
if (expresion) {
  // código a ejecutar si la
  // expresión es verdadera
} else {
  // código a ejecutar si la
  // expresión es falsa
}
```

La proposición if

- La proposición if controla *una sentencia*, pero puede ser una sentencia compuesta.
- Ejemplo:

```
if (costo < dineroEnBolsillo) {
    cout<<“Gasto $" + costo;
    dineroEnBolsillo = dineroEn Bolsillo - costo;
}
```

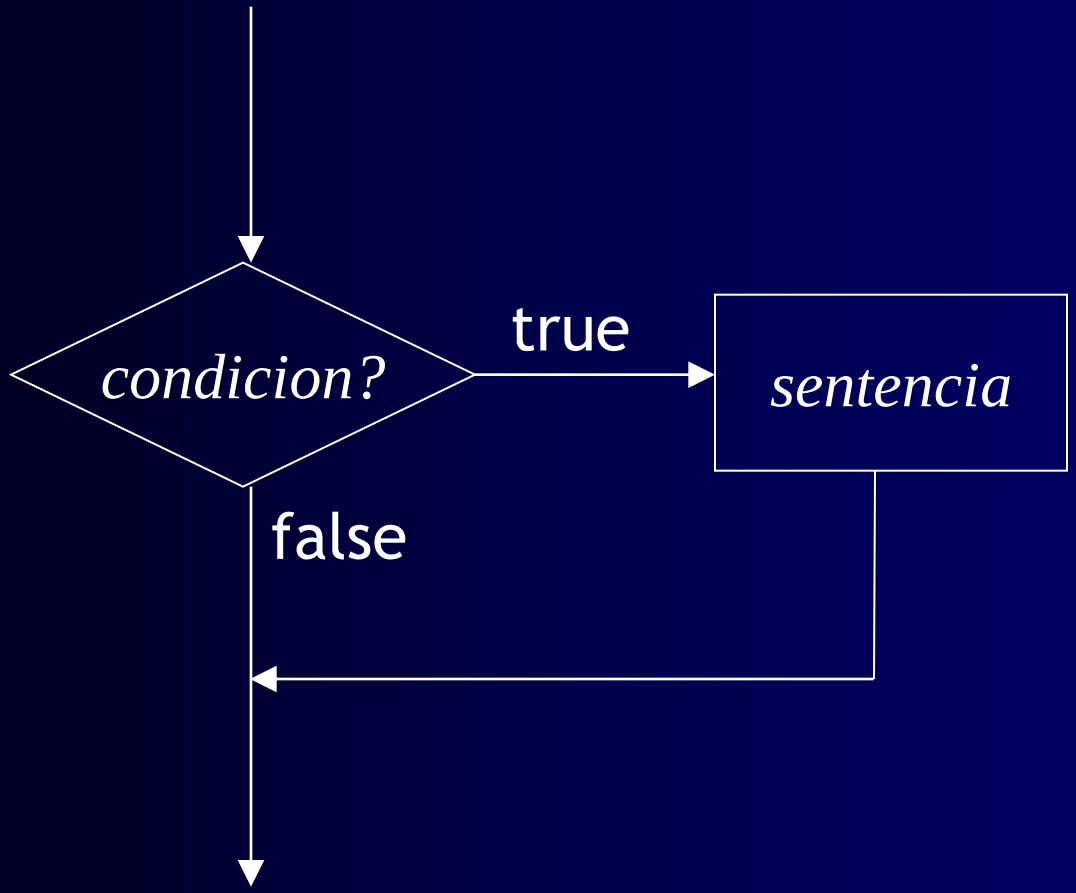
- Es un buen estilo usar las llaves aún si la proposició if controla sólo una sentencia simple:

```
if (costo > dineroEnBolsillo) {
    cout<<“No puedes comprarlo aún!";
}
```

- Personalmente hago una excepción a este estilo de regla cuando la sentencia controlada se ajusta fácilmente en la misma linea con el if:

```
if (x < 0) x = -x;   // valor  absoluto   de x
```

Diagrama de Flujo para la sentencia if



La proposición if-else

- La proposición if-else elige cual de las dos sentencias ejecutar
- La proposición if-else tiene la forma:

  if (*condicion*) *sentencia1* ;
  else *sentencia2* ;

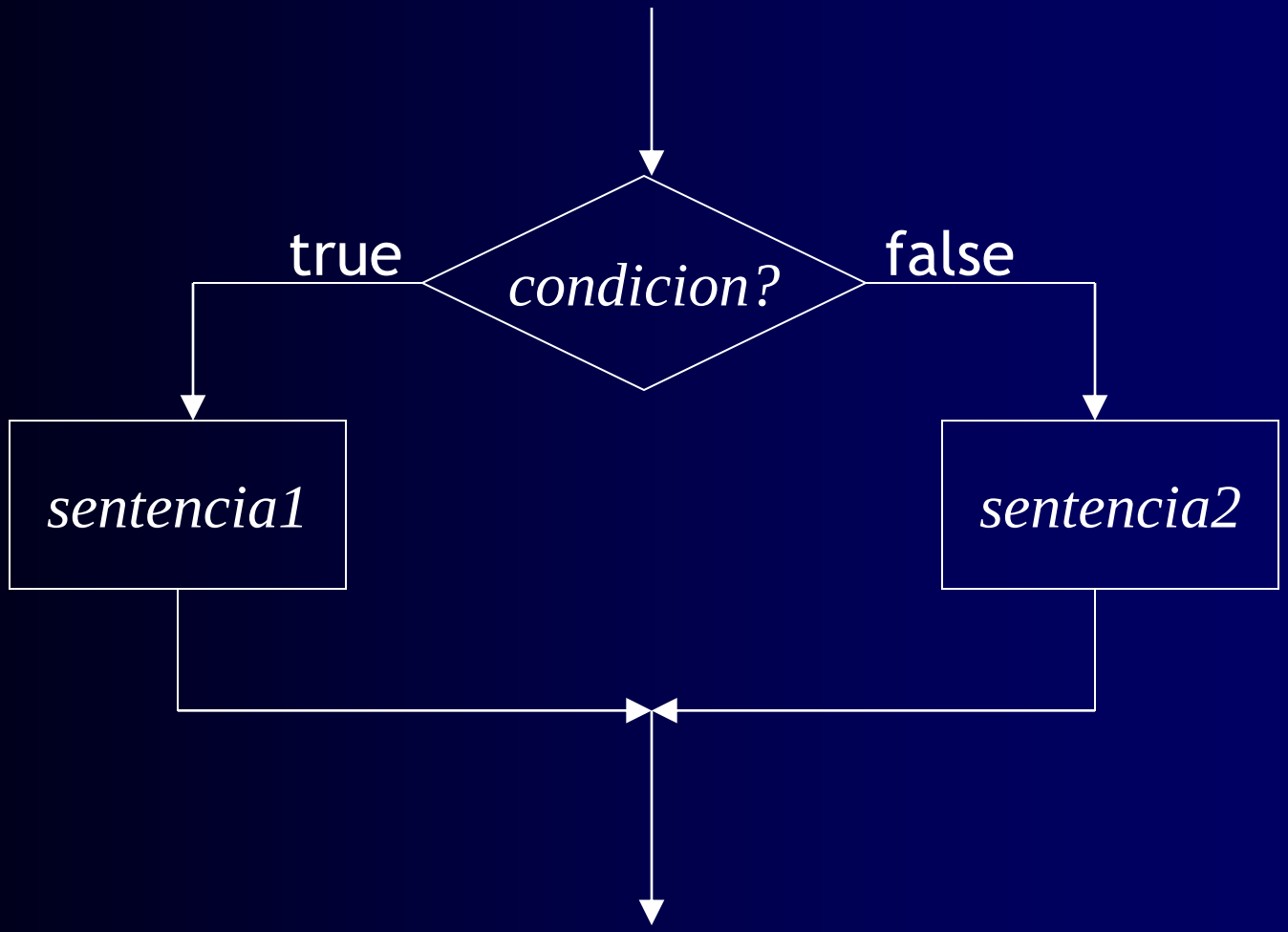
- Una(o ambas) sentencias pueden ser compuestas.
- Observe el punto y coma luego de cada sentencia.

Ejemplo de proposición if-else

- ```
  if (x >= 0) absX = x;
  else absX = -x;
  ```

- ```
  if (CostoItem <= SaldoBanco) {
      cout<<CostoItem;
      SaldoBanco = SaldoBanco - CostoItem;
  }
  else
      cout<<“Saldo Insuficiente”;
  ```

Diagrama de Flujo para la sentencia if - else



Observación: el operador “mod”

- El operador módulo, o “mod,” retorna el resto de una división entera.
- El simbolo para esta operación es %
- Ejemplos:

  57 % 10 retorna 7

  20 % 6 retorna 2

Regla útil:

- x es divisible por y si x % y == 0

Proposiciones anidadas if

- Un año es bisiesto si es divisible por 4 pero no por 100, a menos que también sea divisible por 400

```
if ( aa % 4 == 0) {
    if ( aa % 100 == 0) {
        if ( aa % 400 == 0) bisiesto = true;
        else bisiesto = false;
    }
    else bisiesto = true;
}
else bisiesto = false;
```

Operaciones sobre booleanos

- Supongamos que `p` y `q` son booleanos
- Hay cuatro operaciones básicas sobre booleanos:
  - Negación (“not”):
    `!p` es `true` si `p` es `false` (y `false` en caso contrario)
  - Conjunción (“and”):
    `p && q` es `true` si `p` y `q` son `true`
  - Disyunción (“or”):
    `p || q` es `true` if al menos uno `p` o `q` es `true`
  - or Exclusivo (“xor”):
    `p ^ q` es `true` si solo uno `p` y `q` es `true`

Operadores booleanos

- Las expresiones booleanas pueden combinarse mediante los operadores booleanos clásicos:

&&	- conjunción (and)
||	- disyunción (or)
!	- negación (not)

- Notar que estos operadores difieren de los operadores lógicos binarios: &, |, ~

- Notar que, en muchos casos, los operadores && y || pueden reemplazarse, respectivamente, por el producto y la suma.

Ejercicios

1. Elabore un programa que pida al usuario cinco números enteros e imprima el menor y el mayor de ellos.

2. Escriba un programa que tome como entrada las coordenadas de un punto en el plano cartesiano e indique al usuario en qué cuadrante se encuentra el punto.

Ejercicios

3. Elabore un programa que pida al usuario cuatro calificaciones parciales entre 0 y 10. El programa debe verificar que los datos estén en el rango adecuado y calcular e imprimir el promedio. De acuerdo al promedio, deberá también imprimir alguno de los siguientes mensajes: de 7a 10: “Aprobado”, de 5 a 7: “Derecho a sustitutorio”, de 2 a 5: “Desaprobado”, y de 0 a 2: “Malo”.

4. Elabore un programa que pida al usuario su fecha de nacimiento (en el formato día, mes, año), así como la fecha actual, y calcule la edad del usuario en días. El programa debe verificar que las fechas son correctas (e.g., 30/02/1998 y 10/13/1980 no lo son) y tomar en cuenta los años bisiestos.

Problema Propuesto

- Escriba un programa que pida al usuario tres números enteros entre 0 y 10 (inclusive).
- El programa debe verificar que los números están en el rango correcto, y en ese caso imprimir el promedio de los números. En caso contrario debe imprimir un mensaje de error indicando cuál de los números está fuera de rango.

Problema Propuesto

- Elabore un programa que pida al usuario los coeficientes de una cuadrática y calcule sus raíces, las cuales posiblemente son complejas. El programa deberá indicar de qué tipo son las raíces (reales, imaginarias o complejas) e imprimir los resultados en el formato adecuado.

Pruebas Simples

- Una prueba simple de año bisiesto:

```
if (aa % 4 == 0 &&
    (aa % 100 != 0 || aa % 400 == 0))
        bisiesto = true;
else bisiesto = false;
```

- Una prueba aún más simple de año bisiesto :

```
bisiesto = aa % 4 == 0 &&
    (aa % 100 != 0 || aa % 400 == 0);
```