Instrucciones para la toma de decisiones

Los problemas que hasta este punto podemos resolver son de tipo secuencial. Es decir, en el bloque de instrucciones de los programas que los resuelven, las instrucciones se ejecutan una después de otra, desde el inicio hasta el fin del programa.

Por ejemplo, el bloque de programa planteado a continuación es de ejecucuón secuencial:

miProgramaSecuencial.cpp:

```
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
```

Este esquema deja fuera otro tipo de problema en el que, de acuerdo a ciertos eventos, podría ser necesario realizar algunas instrucciones, o podrían no realizarse. Este tipo de programas podría tener este aspecto:

*miPrograma**No**Secuencial.cpp:*

```
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
```

En este esquema, lo que se plantea es que el bloque de instrucciones verde podría, o no, realizarse. Las líneas punteadas sin colorear indican que antes y después podría, o no, haber más instrucciones. Pero lo importante que se quiere resaltar es que hay un bloque que, en algunas ejecuciones del programa sí se realizaría, y en otras ejecuciones del programa no se realizaría. Debido a esto es que le llamamos ejecución no secuencial, porque de un punto del programa, podría saltarse a otro, sin llegar a éste por recorrido secuencial. Esto, obviamente, debería depender del estado previo de ciertas variables o de los valores de entrada.

También podría darse el siguiente caso, que presenta bloques alternos:

*mi**Otro**Programa**No**Secuencial.cpp:*

```
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
. . . . . ;
```

En este esquema, se requeriría realizar las instrucciones del bloque verde, o las instrucciones del bloque azul, pero no ambos. Antes y después de las instrucciones coloreadas, bien podría,

o no, haber instrucciones secuenciales, como lo indican las líneas punteadas. Pero lo que se quiere resaltar es que llega un momento en el que el programa debe ejecutar solo uno de entre dos bloques de instrucciones. Cada uno de estos bloques podría constar de, al menos, una instrucción.

Y así sucesivamente podríamos requerir, de acuerdo a lo complejo que sea el problema del mundo real, una cantidad indeterminada de bloques dentro del programa que plantea la solución a dicho problema. De entre todos estos bloques, el programa realizaría solo uno dependiendo del estado de ciertas variables.

Hay que subrayar que la cantidad de bloques viene determinada por la complejidad del problema, y lenguajes de programación como C/C++ proveen de las instrucciones de control de flujo necesarias para contemplar estas situaciones. Esta y otras características que aún no hemos visto de este lenguaje, nos dan un indicio de la poderosa herramienta que tenemos en nuestras manos.

Para realizar todo lo que se ha explicado arriba, los lenguajes de programación proporcionan lo que se conoce como instrucciones para la toma de decisiones. El lenguaje de programación C/C++ provee dos instrucciones de este tipo:

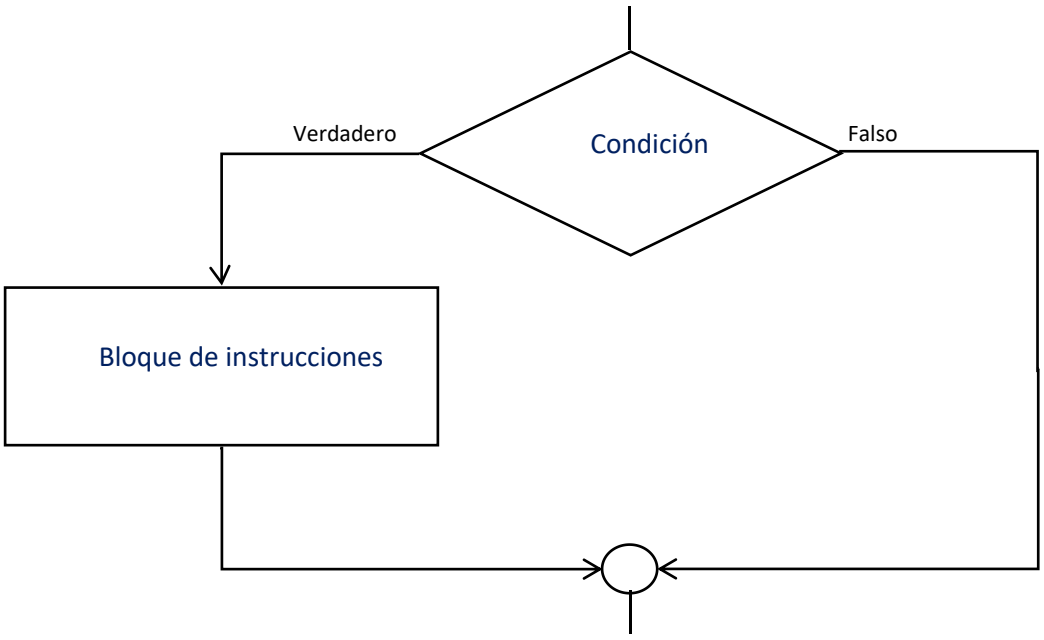
a) La instrucción *if (condición) ... else ...* .

b) La instrucción *switch (expresión) case ... ... otherwise ...* .

Instrucción *if (condición) ... else ...*

Esta es la instrucción, para la toma de decisiones, más simple. Dentro de ella se puede definir un bloque o dos a lo sumo.

La versión de un bloque se ve de la siguiente manera en un diagrama de flujo:



En donde el rombo indica que hay una expresión de comparación justo arriba del bloque, la cual condiciona su ejecución dependiendo de su valor de verdad (valor booleano):

- Si al evaluar la expresión condicional, resulta ser cierta, se ejecuta el bloque, y después se continúa con el resto del programa. Esto es lo que simboliza la rama izquierda del diagrama.

- Si al evaluar la expresión condicional, resulta ser falsa, el control del programa salta hasta el final del bloque y se continúa con el resto del programa. Esto es lo que simboliza la rama derecha de diagrama, la cual no contiene ningún bloque.

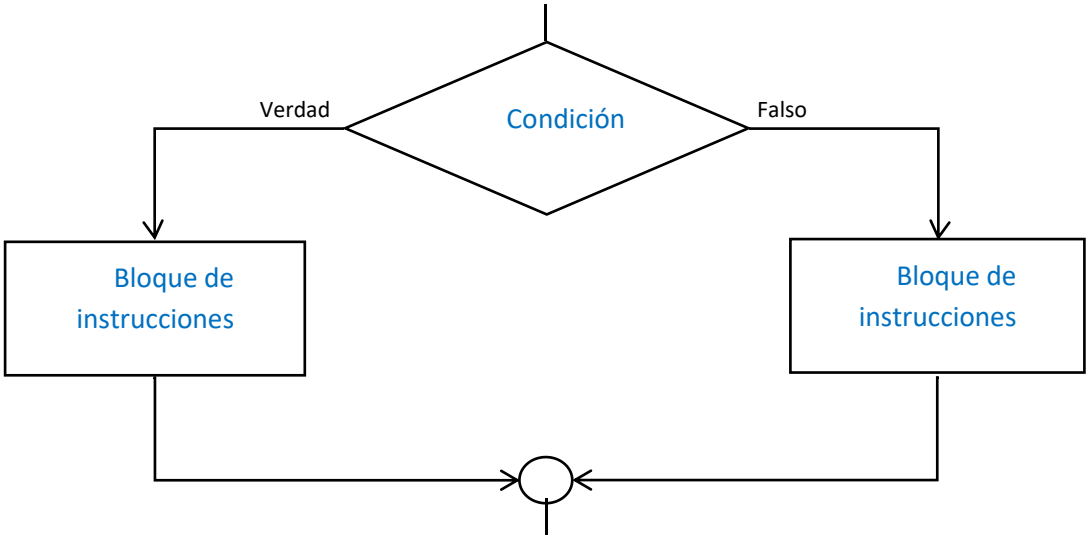
La expresión condicional puede compara cualesquier tipos numéricos, cadenas de carácteres, etc. Solo se requiere que el programador establezca una expresión cuyo valor de verdad, al evaluarla, sea verdadero o falso.

Dentro del programa, la instrucción *if* de un solo bloque se ve así:

```
if (Condición)
{
   . . . . . ;
   . . . . . ;
   . . . . . ;
}
```

C/C++ puede deducir que ese es el bloque que se ejecuta de manera condicionada precisamente porque se encuentra justo a continuación de la palabras reservadas *if* y está encerrado por llaves. La condición que se evalúa para determinar si el bloque se ejecuta o no, se coloca inmediatamente a la derecha de la palabra reservada *if*.

El diagrama de flujo de la versión de *if* de dos bloques se ve de la siguiente manera:



En donde:

- El rombo indica que hay una expresión de comparación justo arriba de ambos bloques, la cual deriva a la ejecución de uno u otro dependiendo de su valor de verdad.

- Si al evaluar la expresión condicional, resulta ser cierta, se ejecuta el bloque que se indica con la rama *Verdadero*, y después se continúa con el resto del programa. Esto es lo que simboliza la rama izquierda del diagrama.

- Si al evaluar la expresión condicional, resulta ser falsa, se ejecuta el bloque que se indica con la rama *Falso*, y después se continúa con el resto del programa. Esto es lo que simboliza la rama derecha de diagrama. Si un segundo bloque no está presente, entonces no se requiere de la palabra reservada *else* y tenemos la versión de *if* de un bloque.

Dentro del programa, la instrucción *if* de dos bloques se ve así:

```
if (Condición)
{
   . . . . . ;
   . . . . . ;
   . . . . . ;
}
else
{
   . . . . . ;
   . . . . . ;
   . . . . . ;
}
```

De nuevo, C/C++ puede identificar los bloques de ejecución alternos debido a que están a continuación de las palabras reservadas *if* y *else* con sus llaves respectivas.

En particular, podría ser necesario colocar una instrucción *if* dentro de alguno de los bloques de un *if*, o dentro de ambos bloques. A esto se le llama anidación, aludiendo a que el *if* más externo abarca y tiene en su interior al *if* más interno, como los nidos de las aves. Los niveles de anidación no tienen límite, ello dependerá del problema que se deba resolver.

Una observación final en cuanto a la construcción de los bloques de instrucciones dentro de las instrucciones de control de flujo es esta: estas instrucciones contienen en su interior bloques de instrucciones encerrados entre llaves, así como las funciones también inician y terminan con llaves; particularmente, si un bloque consta de una sola instrucción, no necesita llaves que lo abarquen.

Ejemplos:

1) Elabore un programa que, dado un valor entero que se ingresa desde teclado, indique si es positivo o negativo.

```
#include "iostream"

using namespace std;

int main(void)
{
	int n;

	cout << endl;
	cout << "DETERMINAR SI UN NÚMERO ES POSITIVO O NEGATIVO" << endl << endl;

	cout << "Digite un número entero: ";
	cin >> n;

	if(n >= 0)
	{
		cout << "El número " << n << " es positivo" << endl;
```

```
    }
    else
    {
        cout << "El número " << n << " es negativo" << endl;
    }

    cout << endl;
    return 0;
}
```

Observe que se ha utilizado una instrucción *if* con sus dos bloques, ya que son dos cosas posibles a hacer dependiendo del signo del número ingresado. Observe también que ambos bloques constan de una sola instrucción, así que podrían omitirse las llaves de apertura y cierre de ambos bloques, así:

```
    if(n >= 0)
        cout << "El número " << n << " es positivo" << endl;
    else
       cout << "El número " << n << " es negativo" << endl;
```

El resto del programa permanece igual.

2) En un almacén, si el total de compra de los clientes es mayor o igual a $60.00, se les aplica un descuento del 20%, si es menor, no hay descuento. Escriba un programa que, dado el valor de compra de un cliente, indique el descuento y el total a pagar.

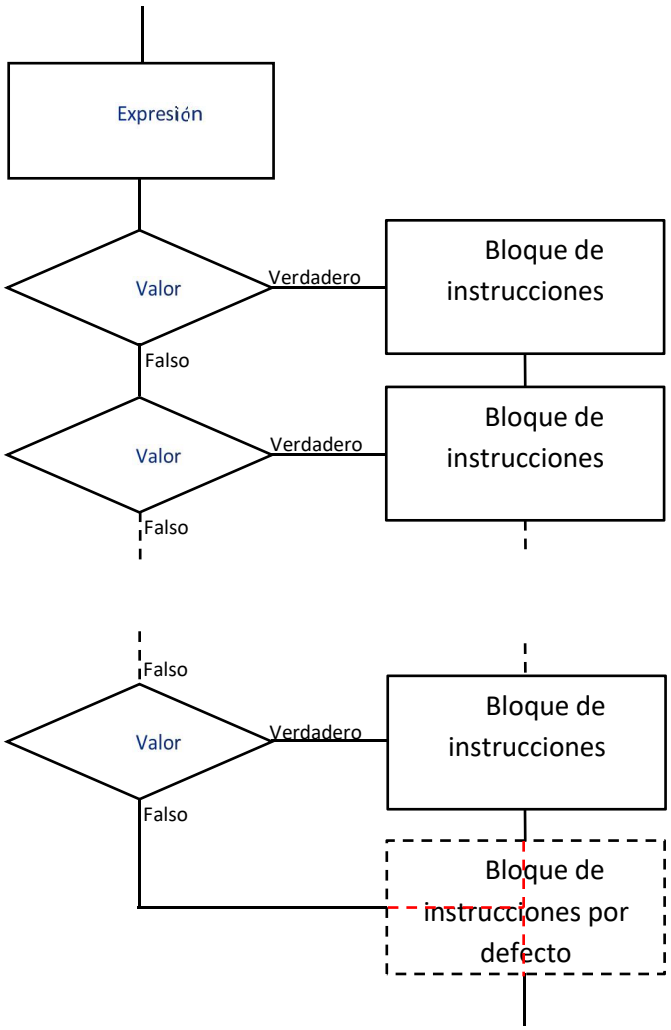
```
#include "iostream"

using namespace std;

int main(void)
{
    float totalCompra, descuento, pago;

    cout << endl;
    cout << "DESCUENTO EN COMPRA EN ALMACÉN" << endl << endl;

    cout << "Digite el monto de la compra: ";
    cin >> totalCompra;

    descuento = 0;
    if (totalCompra >= 60){
        descuento = totalCompra * 0.2;
    }
    pago = totalCompra - descuento;

    cout << "El descuento es de $" << descuento << endl;
    cout << "y el total a pagar es: $" << pago << endl;

    cout << endl;
    return 0;
}
```

Observe que esta instrucción if contempla un solo bloque. Antes del *if* el descuento se asume de $0.0. En la instrucción *if*, si la compra es mayor o igual a $60.00, entonces el descuento se recalcula. Ya afuera de la instrucción *if*, el descuento se resta de la compra y se muestran los resultados.

Instrucción *switch (expresión) case ... ... default ...*

Esta instrucción permite seleccionar entre múltiples opciones de ejecución, por lo que podemos referirnos a ella como una instrucción de ramificación múltiple. De acuerdo al resultado de la evaluación de una expresión, C/C++ optará por ejecutar un bloque de instrucciones de entre varios posibles.

El diagrama de flujo de esta instrucción podemos elaborarlo verticalmente u horizontalmente, en todo caso debe reflejar correctamente su comportamiento. En esta ocasión lo presentamos de la siguiente manera:



En donde:

- El rectángulo en la parte superior contiene en su interior la palabra Expresión, no Condición. Esto significa que la expresión a evaluar obtiene un valor específico, el cuál determina la rama por la que deba ser bifurcado el flujo de ejecución.

- La evaluación de la Expresión nos dará un valor, y si ese valor se encuentra al inicio de una de las ramas, el bloque de esa rama será el que se ejecute.

- Si el flujo de control ingresa a una de las ramas, de allí en adelante todos los bloques serán ejecutados en orden de aparición. Esto sucederá a menos que se coloque la palabra reservada break como última instrucción de cada bloque.

- Hay una última rama, que es opcional, si existe debe ir precedida de la palabra reservada *default:*. Esta rama se ejecutará si el valor retornado por la expresión no fue contemplado en ninguna de las ramas anteriores. La función de esta rama es similar a la rama *else* del *if*.

Dentro del programa, la instrucción *switch* se ve así:

```
switch (Expresión)
{
   case Valor:
        . . . . . .;
        . . . . . .;
   case Valor:
        . . . . . .;
        . . . . . .;
     .
     .
     .
   case Valor:
        . . . . . .;
        . . . . . .;
     .
     .
     .
   default:
        . . . . . .;
        . . . . . .;
}
```

Los bloques definidos en cada rama no están encerrados entre llaves, esto es una excepción entre todas las instrucciones de control de flujo. Es importante hacer notar que la instrucción final de cada bloque debe ser *break;*, o de lo contrario, la siguiente rama también se ejecutará. Si se quiere construir bloques totalmente independientes, estos deben terminar con una instrucción break;. Si un bloque no termina con la instrucción break;, esto implica que el siguiente bloque será ejecutado.

En cuanto a la cantidad de bloques de instrucciones que pueden estar contenidos dentro de esta instrucción, se puede decir que es ilimitada. Solo está determinada por la complejidad del problema a resolver.

Ejemplos:

1) Escriba un programa que, dado un valor entero 1 y 7, se despliegue el día de la semana correspondiente. Su programa debe poder detectar si el valor proporcionado no corresponde y advertir al usuario mediante un mensaje en pantalla.

```
#include "iostream"

using namespace std;

int main(void)
{
      int n;

      cout << endl;
      cout << "MOSTRAR DÍA DE LA SEMANA" << endl << endl;
```

```
    cout << "Digite un número entre 1 y 7: ";
    cin >> n;

    switch (n)
    {
        case 1:
            cout << "Lunes" << endl;
            break;
        case 2:
            cout << "Martes" << endl;
            break;
        case 3:
            cout << "Miércoles" << endl;
            break;
        case 4:
            cout << "Jueves" << endl;
            break;
        case 5:
            cout << "Viernes" << endl;
            break;
        case 6:
            cout << "Sábado" << endl;
            break;
        case 7:
            cout << "Domingo" << endl;
            break;
        default:
            cout << "Número de día no válido" << endl;
    }

    cout << endl;
    return 0;
}
```

2) Elabore un programa que, dada una vocal que se ingresa desde teclado, indique si es abierta o cerrada. Las vocales abiertas son: a, e, o; y las vocales cerradas son: i, u. Su programa debe poder detectar si el dato proporcionado no corresponde a una vocal y advertir al usuario mediante un mensaje en pantalla.

```
#include "iostream"

using namespace std;

int main(void)
{
    char v;

    cout << endl;
    cout << "INDICAR SI UNA VOCAL ES ABIERTA O CERRADA" << endl << endl;

    cout << "Introduzca una vocal: ";
    cin >> v;

    switch(v)
    {
        case 'a':
        case 'e':
        case 'o':
            cout << "La vocal " << v << " es abierta"<< endl;
```

```
            break;
        case 'i':
        case 'u':
            cout << "La vocal " << v << " es cerrada"<< endl;
            break;
        default:
            cout << "El carácter digitado no es una vocal" << endl;
    }

    cout << endl;
    return 0;
}
```

Observe que se han colocado en ramas seguidas las vocales abiertas para aprovechar de que, si se digita cualquiera de estas vocales, se desplegará el mensaje correspondiente, pues solo la última de las ramas de vocal abierta tiene *break*. Lo mismo puede decirse de las vocales cerradas.

3) Escriba un programa que despliegue un menú que permita calcular el perímetro de alguno de estos tres tipos de triángulos: equilátero, isósceles y escaleno.

```
#include "iostream"

using namespace std;

int main(void)
{
    int opcion;
    float perimetro;
    cout << endl;
    cout << "CALCULO DE PERÍMETROS DE TRIÁNGULOS" << endl << endl;

    cout << "Menú de opciones:" << endl;
    cout << "1) Equilatero." << endl;
    cout << "2) Isósceles." << endl;
    cout << "3) Escaleno." << endl;
    cout << "Escoja su opción: ";
    cin >> opcion;

    switch(opcion)
    {
        case 1:
            float lados;
            cout << "Digite la longitud de los tres lados: ";
            cin >> lados;
            perimetro = 3 * lados;
            cout << "El perímetro es: " << perimetro << endl;
            break;
        case 2:
            float dosLados, ladoDistinto;
            cout << "Digite la longitud de los dos lados iguales: ";
            cin >> dosLados;
            cout << "Digite la longitud del lado distinto: ";
            cin >> ladoDistinto;
            perimetro = 2 * dosLados + ladoDistinto;
            cout << "El perímetro es: " << perimetro << endl;
            break;
        case 3:
            float lado1, lado2, lado3;
            cout << "Digite la longitud del primer lado: ";
```

```
                cin >> lado1;
                cout << "Digite la longitud del segundo lado: ";
                cin >> lado2;
                cout << "Digite la longitud del tercer lado: ";
                cin >> lado3;
                perimetro = lado1 + lado2 + lado3;
                cout << "El perímetro es: " << perimetro << endl;
                break;
            default:
                cout << "Opción no válida" << endl;
        }
        cout << endl;
    }
```

Observe en este programa la declaración de variables en lugares distintos al habitual. En C/C++ pueden declararse variables en cualquier lugar de un programa. Cada lugar constituye un ámbito del programa, las variables declaradas en ese ámbito no serán visibles fuera de él. Es decir, El alcance de una variable no va más allá del ámbito en el que ha sido declarada.

Ejercicios:

1) Dados dos enteros que se ingresan desde teclado, indicar cuál de los dos es mayor.

2) Dados dos enteros que se ingresan desde teclado, indicar cuál de los dos es mayor. Su programa también debe contemplar la posibilidad de que sean iguales e indicarlo con un mensaje en pantalla.

3) Dados dos enteros, a y b, indicar si el primero es o no divisible entre el segundo.

4) Utilice la fórmula cuadrática para encontrar las raíces de un polinomio de la forma ax^2+bx+c. Recuerde que la fórmula es:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Su programa debe contemplar que si $b^2-4ac < 0$ entonces no hay solución en los reales y desplegar un mensaje indicándolo.

5) Dado un valor entero 1 y 12 despliegue el mes correspondiente. Su programa debe poder detectar si el valor proporcionado no corresponde y advertir al usuario mediante un mensaje en pantalla.