


Programación

UD2: Estructuras de Control

Jaime Matas Bustos
CFGS Desarrollo de Aplicaciones Multiplataforma
C.E.S. Cristo Rey

Contenido del Tema

Controlar el flujo

1. Motivación
   - Las estructuras de control
2. La estructura condicional
   - Estructura condicional simple
   - Estructura condicional doble
   - Estructura condicional múltiple
3. Estructuras repetitivas
   - Bucles do while
   - Bucles while
   - Bucles for
4. Nota final

A veces tendremos que escoger entre diferentes decisiones en función de los datos o querremos realizar varias veces una determinada tarea.

Ejemplo

Encontrar las raíces de un polinomio de segundo grado

$$p(x) = ax^2 + bx + c$$

$$p(x1) = 0, p(x2) = 0$$

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

-nan —> 2*a =0
o raíz negativa

```
1  /* Programa que calcula las raíces de una ecuación
2  de segundo grado (PRIMERA APROXIMACIÓN)
3  Entradas: Los parámetros de la ecuación
4  Salidas: Las raíces de la parábola */
5  #include <iostream>
6  #include <cmath>
7  using namespace std;
8  int main(){
9    double a,b,c;  // Parámetros de la ecuación
10   double x1,x2;  // Raíces obtenidas
11
12   cout << "\nIntroduce coef.  de 2o grado:  ";
13   cin >> a;
14   cout << "\nIntroduce coef.  de 1er grado:  ";
15   cin >> b;
16   cout << "\nIntroduce coef.  independiente:  ";
17   cin >> c;
18   x1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
19   x2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
20   cout << "Las raíces son:  "<< x1 <<
21     " y "<< x2 << endl;
22 }
```

El flujo de control: 8,9,10,11,12,13,14,15,16,17,18,19,20,21,FIN

Estructuras de control

- Estructura secuencial
- Estructura condicional
- Estructura repetitiva

El diseño de condiciones

Condición

Es una expresión cuyo resultado es un dato booleano o lógico. Se construyen fundamentalmente a partir de los operadores relacionales

Operador relacional

Es una operación que se aplica a dos valores del mismo tipo y que devuelve un valor lógico, `true` o `false`

Los operadores relacionales representan las operaciones de comparación a las que estamos acostumbrados

x es igual a y	`x == y`
x es distinto de y	`x != y`
x es menor (mayor) estricto que y	`x < y` (x > y)
x es menor (mayor) o igual que y	`x <= y` (x >= y)

El diseño de condiciones

```
1 int valor=10;
2 double resultado=0.34, resultado2=9.8;
3 char caracter='a';
4 bool apto=false;
```

```
1 valor == 10
2 resultado2 >= resultado
3 resultado2 == valor
4 caracter != 'b'
5 caracter <= 'c'
```

El diseño de condiciones

Condición compuesta

Es una expresión cuyo resultado es un dato booleano o lógico y se compone de condiciones más simples unidas mediante operadores lógicos

Operador lógico

Es una operación que se aplica a valores de tipo lógico y que devuelve otro valor lógico, `true` o `false`

Los operadores lógicos representan la negación, conjunción y disyunción

a ó b	`a || b`
a y b	`a && b`
negación de a	`!a`

El diseño de condiciones

a	b	a || b	a && b	!a
false	false	false	false	true
false	true	true	false	true
true	false	true	false	false
true	true	true	true	false

Tener en cuenta las Leyes de De Morgan

El matemático Augustus De Morgan

```
!(X || Y) ⟶ !X && !Y
!(X && Y) ⟶ !X || !Y
```

El diseño de condiciones

```
1 int valor=10;
2 double resultado=0.34, resultado2=9.8;
3 char caracter='a';
4 bool apto=false;
```

```
1 0 <= valor && valor <= 20
2 0 <= valor <= 20 // Error común
3 carácter == 's'||resultado2 >= resultado
4 caracter != 'n'&& caracter != 'N'
5 caracter == 's'||caracter == 'S'
6 caracter >= 'a'&& caracter <= 'z'
```

El diseño de condiciones

La prioridad de los operadores

- Los paréntesis
- ! (negación)
- – (unario)
- * / %
- + –
- < <= > >=
- == !=
- &&
- ||

En caso de duda, usar paréntesis
Evaluaciones en ciclo corto y ciclo largo (Garrido, A., pg. 59).

La estructura condicional

Una estructura condicional es una estructura que permite la ejecución de una (o más) sentencia(s) dependiendo de la evaluación de una condición.

Existen tres tipos:

- Simple
- Doble
- Múltiple

La estructura condicional simple

Una estructura de control condicional puede alterar el flujo de control y permite ejecutar una sentencia o conjunto de sentencias dependiendo del valor de una condición.

```
if ( <condición>)
      <sentencia>
```

```
if ( <condición>) {
      <sentencias>
}
```

```
1  /* Programa que calcula las raíces de una ecuación
2  de segundo grado (PRIMERA APROXIMACIÓN)
3  Entradas: Los parámetros de la ecuación
4  Salidas: Las raíces de la parábola */
5  #include <iostream>
6  #include <cmath>
7  using namespace std;
8  int main(){
9    double a,b,c;  // Parámetros de la ecuación
10   double x1,x2;  // Raíces obtenidas
11
12   cout << "\nIntroduce coef.  de 2o grado:  ";
13   cin >> a;
14   cout << "\nIntroduce coef.  de 1er grado:  ";
15   cin >> b;
16   cout << "\nIntroduce coef.  independiente:  ";
17   cin >> c;
18   x1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
19   x2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
20   cout << "Las raíces son:  "<< x1 <<
21     " y "<< x2 << endl;
22 }
```

```
1   cout << "\nIntroduce coef.  de 2o grado:";
2   cin >> a;
3   cout << "\nIntroduce coef.  de 1er grado:";
4   cin >> b;
5   cout << "\nIntroduce coef.  independiente:";
6   cin >> c;
7   if (a != 0)   {
8    x1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
9    x2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
10   cout << "Las raíces son:  "<< x1 <<
11          " y "<< x2 << endl;
12   }
```

Flujo de control

a != 0

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, FIN

a == 0

1, 2, 3, 4, 5, 6, 7, FIN

La estructura condicional doble

```
if ( <condición>)
      <sentencia-1>
else
      <sentencia-2>
```

```
if ( <condición>) {
      <sentencias-1>
}
else {
      <sentencias-2>
}
```

```
1   cout << "\nIntroduce coef.  de 2o grado:";
2   cin >> a;
3   cout << "\nIntroduce coef.  de 1er grado:";
4   cin >> b;
5   cout << "\nIntroduce coef.  independiente:";
6   cin >> c;
7   if (a != 0)   {
8    x1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
9    x2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
10   cout << "Las raíces son:  "<< x1 <<
11        " y "<< x2 << endl;
12  }
13  else
14   cout << "Solo una raíz:  "<< -c/b << endl;
```

Flujo de control

a != 0

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, FIN

a == 0

1, 2, 3, 4, 5, 6, 7, 14, FIN

Anidamiento de estructuras condicionales

```
1  if (a<b)
2      if (b<c)
3          <s1>
4      else
5          <s2>
6  else
7      if (b>c)
8          <s3>
9      else
10         <s4>
```

¿Cuándo se ejecuta <s1>?

a<b && b<c

¿Cuándo se ejecuta <s2>?

a<b && b>=c

¿Cuándo se ejecuta <s3>?

a>=b && b>c

¿Cuándo se ejecuta <s4>?

a>=b && b<=c

```
1   cout << "\nIntroduce coef.  de 2o grado:";
2   cin >> a;
3   cout << "\nIntroduce coef.  de 1er grado:";
4   cin >> b;
5   cout << "\nIntroduce coef.  independiente:";
6   cin >> c;
7   if (a != 0)   {
8    if (b*b-4*a*c >= 0)   {
9       x1 = (-b+sqrt(b*b-4*a*c))/(2*a);
10      x2 = (-b-sqrt(b*b-4*a*c))/(2*a);
11      cout << "Las raíces son:  "<< x1 <<
12         " y "<< x2 << endl;
13   }
14   else
15      cout << "Raíces complejas"<< endl;
16  }
17  else
18   cout << "Solo una raíz:  "<< -c/b << endl;
```

Observar la tabulación

La estructura condicional múltiple

Permite elegir entre múltiples alternativas para el flujo de control a la misma vez.

```
switch (<expresión>) {
    case <valor1>:
        <sentencias>
        break;
    case <valor2>:
        <sentencias>
        break;
    case <valor3>:
        <sentencias>
        break;
    default:
        <sentencias>
}
```

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    char c;
5    cout << "Pulsa [S]i/[N]o/[A]yuda:  ";
6    cin >> c;
7    switch (c) {
8    case 's':
9      cout << "Has aceptado";
10     break;
11   case 'n':
12     cout << "Has cancelado";
13     break;
14   case 'a':
15     cout << "Has pedido ayuda";
16     break;
17   default:
18     cout << "Respuesta no permitida";
19   }
20 }
```

```
1   switch (c) {
2   case 'S':
3   case 's':
4      cout << "Has aceptado";
5      break;
6   case'N':
7   case 'n':
8      cout << "Has cancelado";
9      break;
10  case'A':
11  case 'a':
12     cout << "Has pedido ayuda";
13     break;
14  default:
15     cout << "Respuesta no permitida";
16  }
```

Estructuras repetitivas

Las estructuras de control repetitivas, también llamadas bucles, permiten ejecutar mútiples veces una misma sentencia o conjunto de sentencias.

Cuerpo del bucle

El conjunto de sentencias que se repite en un bucle

- Bucles `do while`
- Bucles `while`
- bucles `for`

Bucles `do while`

```
do {
    <sentencias> ← cuerpo del bucle
} while (condición);
```

- Se ejecutan las sentencias mientras la condición sea `true`
- Cuando la condición se hace `false` el bucle termina
- El cuerpo del bucle se ejecuta al menos una vez.

Bucles `do while`

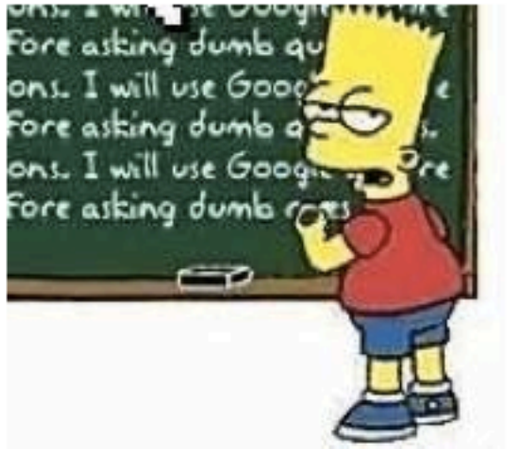
Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo.

`10, 20, 30, 40 50, -1`

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int dato;
5
6    do {
7      cin >> dato;
8      cout << "Recibido:  "<< dato << endl;
9    } while (dato >= 0);
10   cout << "Fin transmisión"<< endl;
11 }
```

FC: 4,5,6,$\overbrace{7,8,9}$,$\overbrace{7,8,9}$,$\overbrace{7,8,9}$,...,$\overbrace{7,8,9}$,10,FIN

Bucles `do while`



Escribir 10 veces
"No debo copiar en clase"

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int cuenta=1;
5
6    do {
7      cout << "No debo copiar en clase"<< endl;
8      cuenta = cuenta + 1;
9    } while (cuenta <= 10);
10 }
```

FC: 4,5,6,7,8,9,7,8,9,7,8,9,...,7,8,9,FIN

Bucles `do while`

Escribir la tabla de multiplicar del 7;

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4    int i=1;
5    do {
6       cout << "7 x "<< i << " = "<< 7*i << endl;
7       i = i + 1;
8    } while (i<= 10);
9 }
```

FC:4,5,6,7,8,6,7,8,6,7,8,...,6,7,8,FIN

Bucles `do while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=0;
5     do {
6        i = i + 3;
7        cuenta = cuenta + 1;
8     } while (i <= 10);
9     cout << cuenta;
10 }
```

Bucles `do while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=12;
5     do {
6        i = i - 3;
7        cuenta = cuenta + 1;
8     } while (i >= 0);
9     cout << cuenta;
10 }
```

Bucles `do while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=12;
5     do {
6        i = i + 3;
7        cuenta = cuenta + 1;
8     } while (i >= 0);
9     cout << cuenta;
10 }
```

Bucles `do while`

Se suelen utilizar para depurar los datos de entrada y eliminar valores incorrectos o no deseados. Filtros de entrada

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5    double dato, res;
6
7    cout << "Introduce un número:  ";
8    cin >> dato;
9    res = sqrt(dato);
10   cout << "Resultado:  "<< res;
11   cout << endl;
12 }
```

Bucles `do while`

Se suelen utilizar para depurar los datos de entrada y eliminar valores incorrectos o no deseados. Filtros de entrada

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5    double dato, res;
6    do {
7      cout << "Introduce un número:  ";
8      cin >> dato;
9    } while (dato < 0);
10   res = sqrt(dato);
11   cout << "Resultado:  "<< res;
12   cout << endl;
13 }
```

Diseñar un filtro de datos de entrada para pedir una confirmación al usuario que sólo acepte las teclas S ó N. El resto se ignoran.

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main() {
5    char tecla;
6    do {
7      cout << "¿ Confirmar (S/N) ?";
8      cin >> tecla;
9    } while (tecla != 's' && tecla !='S'
10      && tecla != 'N' && tecla != 'n');
11   // En este punto tecla es S ó s ó n ó N
12   if (tecla == 's'||tecla == 'S')
13     cout << "ACEPTADO";
14   else
15     cout << "CANCELADO";
16   cout << endl;
17 }
```

Diseñar un filtro de datos de entrada para leer un número entero mayor o igual que -6 y estrictamente menor de 20. El resto se ignoran.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int num;
5     do {
6        cout << "Introduce un número en [-6, 20)";
7        cin >> num;
8     } while (num < -6 ||num >= 20);
9     // En este punto num vale un entero
10    // en el intervalo [-6, 20)
11    cout << "El número "<< num << " es correcto";
12 }
```

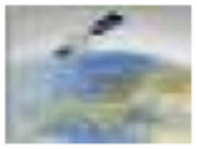
Diseñar un filtro de datos de entrada para leer dos números necesariamente distintos de cero.

Bucles `while`

```
while (<condición>) {
    <sentencias> ← cuerpo del bucle
}
```

- Se ejecutan las sentencias mientras la condición sea `true`
- Cuando la condición se hace `false` el bucle termina
- Si la condición es `false` desde el principio las sentencias nunca se ejecutarían.

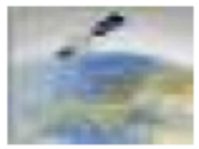
Bucles `while`



Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo**. Entrada de datos controlada por centinela `10, 20, 30, 40 50, -1`

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int dato;
5     do {
6        cin >> dato;
7        cout << "Recibido:   "<< dato << endl;
8        if (dato >= 0)
9           cout << "Procesando:   "<< dato << endl;
10    } while (dato >= 0);
11    cout << "Fin transmisión"<< endl;
12 }
```

Bucles `while`



Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo**. Entrada de datos controlada por centinela `10, 20, 30, 40 50, -1`

```
1  int main() {
2    int dato;
3    cin >> dato;
4    cout << "Recibido:   "<< dato << endl;
5    while (dato >= 0) {
6      cout << "Procesando:   "<< dato << endl;
7      cin >> dato;
8      cout << "Recibido:   "<< dato << endl;
9    }
10   cout << "Fin transmisión"<< endl;
11 }
```

FC:2,3,4,5,6,7,8,5,6,7,8,5,6,7,8,,...,5,6,7,8,5,10,FIN

Lectura anticipada

Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo**. Se necesita saber cuál es la media de los datos procesados.

```
1   int dato, ndatos=0;
2   double suma=0,media; // La media puede ser real
3   cin >> dato; // Lectura adelantada
4   cout << "Recibido:  "<< dato << endl;
5   while (dato >= 0) {
6     cout << "Procesando:  "<< dato << endl;
7     ndatos = ndatos+1;
8     suma = suma + dato;
9     cin >> dato;
10    cout << "Recibido:  "<< dato << endl;
11  }
12  cout << "Fin transmisión"<< endl;
13  media = suma/ndatos;
14  cout << "La media es:  "<< media;
```

Bucles `while`

Escribir 10 veces
"No debo copiar en clase"

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4   int cuenta=1;
5
6   while (cuenta <= 10) {
7     cout << "No debo copiar en clase"<< endl;
8     cuenta = cuenta + 1;
9   }
10 }
```

Bucles `while`

Escribir la tabla de multiplicar del 7;

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4   int i=1;
5   while (i <= 10) {
6     cout << "7 x "<< i << " = "<< 7*i << endl;
7     i = i + 1;
8   }
9 }
```

Bucles `while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=0;
5     while (i <= 20) {
6        i = i + 4;
7        cuenta = cuenta + 1;
8     }
9     cout << cuenta;
10 }
```

Bucles `while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=20;
5     while (i != 4) {
6        i = i - 4;
7        cuenta = cuenta + 1;
8     }
9     cout << cuenta;
10 }
```

Bucles `while`

¿Qué valor se muestra en la pantalla?

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int cuenta=0, i=20;
5     while (i != 4) {
6        i = i - 5;
7        cuenta = cuenta + 1;
8     }
9     cout << cuenta;
10 }
```

Bucles `for`

Es una estructura repetitiva compacta que permite definir una o más variables de control, inicializarlas, definir la condición del bucle y la actualización de las variables de control en la misma sentencia.

```
1    int cuenta=1;
2
3    while (cuenta <= 10) {
4       cout << "No debo copiar en clase"<< endl;
5       cuenta = cuenta + 1;
6    }
```

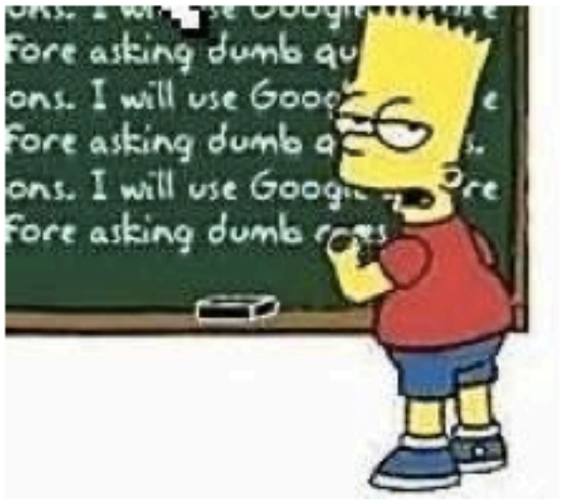
```
1 for (cuenta=1; cuenta <= 10; cuenta = cuenta + 1)
2     cout << "No debo copiar en clase"<< endl;
```

Bucles `for`

```
for (<inicialización>; <condición>; <actualización>) {
    <sentencias> ← cuerpo del bucle
}
```

1. Se ejecuta la inicialización.
2. Se evalúa la condición. Si es falsa, se acaba el bucle. Si es verdadera, continúa.
3. Se ejecuta el cuerpo del bucle.
4. Se ejecuta la actualización.
5. Se repite desde 2.

Bucles `for`



Escribir 10 veces
"No debo copiar en clase"

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4   int cuenta;
5
6   for (cuenta=1; cuenta <= 10; cuenta = cuenta + 1)
7    cout << "No debo copiar en clase"<< endl;
8 }
9
```

FC:4,5,$\overbrace{6(i),6(c),7,6(+)}$,$\overbrace{6(c),7,6(+)}$,$\overbrace{6(c),7,6(+)}$,,...,$\overbrace{6(c),7,6(+)}$,$\overbrace{6(c)}$FIN

Bucles `for`

Escribir la tabla de multiplicar del 7;

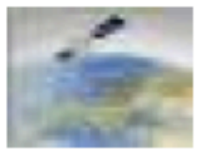
```
1 #include <iostream>
2 using namespace std;
3 int main() {
4   int i;
5   for (i=1; i<=10; i=i+1) {
6     cout << "7 x "<< i << " = "<< 7*i << endl;
7   }
8 }
```

¿Qué valor se muestra en la pantalla?

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4    int i, total;
5    total = 0;
6    for(i=1 ; i<=10; i++)
7       total = total + 3;
8    cout << total << endl;
9 }
```

¿Qué valor se muestra en la pantalla?

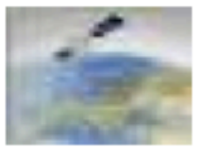
```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int i, total;
5
6     total = 0;
7     for (i=4 ; i<=36 ; i=i+4)
8       total=total+1;
9     cout << total << endl;
10 }
```



Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo. Como mucho se envían 20 datos.** `10, 20, 30, 40 50, -1`

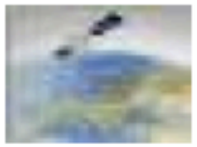
```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int dato, i;
5     for (i=1; i<=20; i=i+1) {
6        cin >> dato;
7        cout << "Recibido:  "<< dato << endl;
8        if (dato >= 0)
9           cout << "Procesando:  "<< dato << endl;
10       else
11          i=21;
12    }
13    cout << "Fin transmisión"<< endl;
14 }
```

Esta es una situación que se debe evitar



Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo. Como mucho se envían 20 datos.** `10, 20, 30, 40 50, -1`

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int dato, i;
5    bool fin=false;
6    for (i=1; i<=20 && fin == false; i=i+1) {
7      cin >> dato;
8      cout << "Recibido:  "<< dato << endl;
9      if (dato >= 0)
10       cout << "Procesando:  "<< dato << endl;
11     else
12       fin = true;
13   }
14   cout << "Fin transmisión"<< endl;
15 }
16
```



Un satélite realiza una transmisión de un número indefinido de datos enteros que terminan con un número negativo. **Todos los datos se procesan en tierra, menos el negativo. Como mucho se envían 20 datos.** `10, 20, 30, 40 50, -1`

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int dato, i;
5    bool fin=false;
6    for (i=1; i<=20 && !fin; i=i+1) {
7      cin >> dato;
8      cout << "Recibido:  "<< dato << endl;
9      if (dato >= 0)
10       cout << "Procesando:  "<< dato << endl;
11     else
12       fin = true;
13   }
14   cout << "Fin transmisión"<< endl;
15 }
16
```

Es importante no modificar las variables controladoras del bucle en el cuerpo del bucle.

Ejemplo: Mostrar los números pares de 1 a número leido (inclusive)

```
1 cin >> tope;
2 for (i = 1; i < tope; i=i+2) {
3    cout << i << endl;
4    if (tope % 2 !=0)
5        tope++;
6 }
```

Solucion correcta:

```
1 cin >> tope;
2 if (tope % 2 !=0)
3     tope++;
4 for (i = 1; i < tope; i=i+2) {
5    cout << i << endl;
6 }
```

Algunas novedades

C++ permite el uso de algunos operadores aritméticos de forma compacta, que son completamente equivalentes pero que simplifican la escritura de los programas.

`i=i+1`	`i+=1`	`i++` `++i`
`i=i-1`	`i-=1`	`i--` `--i`
`i=i+incremento`	`i+=incremento`	
`i=i-incremento`	`i-=incremento`	
`i=i*multiplo`	`i*=multiplo`	
`i=i/divisor`	`i/=divisor`	

Diferencias entre `i++` y `++i` en Garrido, A. pg. 73

Contar cuantos múltiplos de 3 hay en el intervalo [-100, 100]

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    const int LI=-100, LS=100;
5    int multiplos=0, i;
6    for(i = LI; i<=LS; i++)
7      if (i % 3 == 0)
8        multiplos++;
9    cout << "Hay "<< multiplos <<
10     " múltiplos de 3";
11 }
```

Bucle contador

Un satélite ... **Todos los datos se procesan en tierra, menos el negativo. Como mucho se envían 20 datos**. Hallar la media de los datos procesados.

```
1    int dato, i, ndatos=0;
2    double suma=0, media; // La media puede ser real
3    bool fin=false;
4    for (i=1; i<=20 && !fin; i++) {
5      cin >> dato;
6      cout << "Recibido:  "<< dato << endl;
7      if (dato >= 0) {
8        cout << "Procesando:  "<< dato << endl;
9        ndatos ++;
10       suma += dato;
11     }
12     else
13       fin = true;
14   }
15   cout << "Fin transmisión"<< endl;
16   media = suma/ndatos;
17   cout << "La media es:  "<< media;
```

Bucle acumulador

Calcular el factorial de un número leído desde el teclado

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int numero, i;
5     /* int res; // Conceptualmente correcto    */
6     double res=1;  // Pero el resultado será muy grande
7
8     cout << "Introduce el número:  ";
9     cin >> numero;
10    for (i=1; i<=numero; i++)
11       res *= i;
12    cout << numero << "!  = "<< res;
13 }
```

Bucle acumulador

Calcular el valor de

$$\sum_{i=1}^{n} \frac{1-i}{i}$$

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int n, i;
5     double res=0;  //El resultado tiene decimales
6
7     cout << "Introduce n:   ";
8     cin >> n;
9     for (i=1; i<=n; i++)
10       res += (1.0-i)/i;
11    cout << "Resultado:   "<<res;
12 }
```

Bucle acumulador

Anidamiento de estructuras repetitivas

Las estructuras repetitivas se pueden anidar unas dentro de otras.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4     int i, j, iteraciones=0;
5
6     for (i=0; i<3; i++)
7        for (j=0; j<=2 ; j++)
8           iteraciones++;
9     cout << iteraciones;
10 }
```

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int i, j, iteraciones=0;
5
6    for (i=0; i<4; i++)
7      for (j=i; j<4 ; j++)
8        iteraciones++;
9    cout << iteraciones;
10 }
```

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4    int i, j, iteraciones=3;
5
6    for (i=2 ; i<=10 ; i=i+2)
7      for (j=1 ; j<=8 ; j++)
8        iteraciones++;
9    cout << iteraciones;
10 }
```

Una nota final ...

Existen algunas sentencias de control de flujo que permiten hacer algunas “cosas especiales”.

- `goto`
- `break`
- `continue`

Sin embargo estas sentencias van en contra de la programación estructurada y no deben de usarse (Garrido, A. p. 86-87).

Bibliografía

[2005] Garrido, A. "Programación en C++"