# Programación I

## Instrucciones de control - bucles

Iván Cantador
Escuela Politécnica Superior
Universidad Autónoma de Madrid

# Contenidos



- Bucles
  - La instrucción for
  - La instrucción while
  - La instrucción do while
- Bucles anidados

# Contenidos



- **Bucles**
  - **La instrucción for**
  - La instrucción while
  - La instrucción do while
- Bucles anidados

# Bucles: la instrucción for (I)



- Un bucle **for** ejecuta un bloque de sentencias de forma iterativa mientras se cumpla una condición de fin, a partir de una sentencia de inicio y considerando una sentencia de paso de iteración
  - Sintaxis

```
for ( <inicio>; <condicion_fin>; <paso_iteracion> ) {
   <bloque_sentencias>
}
```

→ Las sentencias inicio, paso_iteracion y condicion_fin son optativas

# Bucles: la instrucción for (II)



- En general, el número de iteraciones de un bucle **for** viene determinado por valores inicial, de incremento y final de una variable entera (contador), que por motivos históricos suele nombrarse i, j, k, …

```
int numIteraciones = 10;
int i;

// El siguiente bucle se lee "para/desde i=0, mientras que
// i<numIteraciones, incrementar i de 1 en 1"
for ( i = 0; i < numIteraciones; i++ ) {
   printf("Iteracion %d de %d.\n", i, numIteraciones);
}
```

# Bucles: la instrucción for (III)



- Ejemplo

```
#include <stdio.h>

void main() {
    int i;

    // Calculo de la suma de los numeros impares menores o iguales que 99
    suma = 0;
    for ( i = 1; i <= 99; i += 2 ) {
        suma += i;
    }
}
```

# Bucles: la instrucción for (IV)



- Ejemplo

```
#include <stdio.h>
#include <limits.h>

void main() {
    int lista[] = {4, 3, 7, 2, 9, 5};
    int longitudLista = 6;
    int i, valorMedio, valorMinimo, valorMaximo;

    // Calculo de los valores medio, minimo, y maximo de la lista
    valorMedio = 0;
    valorMinimo = INT_MAX;
    valorMaximo = INT_MIN;
    for ( i = 0; i < longitudLista; i++ ) {
        valorMedio += lista[i];

        if ( lista[i] < valorMinimo ) {
            valorMinimo = lista[i];
        }

        if ( lista[i] > valorMaximo ) {
            valorMaximo = lista[i];
        }
    }
    valorMedio /= longitudLista;
}
```

# Bucles: la instrucción for (V)



- Bucle **for** “infinito”

```
for ( ; ; ) {
        printf("Bucle infinito!\n");
}
```

# Bucles: la instrucción for (VI)



- La sentencia **break** "rompe" la ejecución del bucle en el que se encuentra

```
#include <stdio.h>

void main() {
    int lista[] = {4, 3, 7, 2, 9, 5, 8, 1, 2, 6};
    int i, x, n = 10, posicion;

    // Busca la PRIMERA posición en la lista del número introducido o
    // establece la posicion a -1 en caso de que el numero no este en la lista
    printf("Introduce un numero a buscar: ");
    scanf("%d", &x);

    posicion = -1;
    for ( i = 0; i < n; i++ ) {
        if ( lista[i] == x ) {
            posicion = i;
            break;
        }
    }
    printf("La posicion de %d en la lista es %d.\n", x, posicion);
}
```

# Bucles: la instrucción for (VII)



- La sentencia **break** "rompe" la ejecución del bucle en el que se encuentra

```
#include <stdio.h>

void main() {
    int lista[] = {4, 3, 7, 2, 9, 5, 8, 1, 2, 6};
    int i, x, n = 10, posicion;

    // Busca la ULTIMA posición en la lista del número introducido o
    // establece la posición a -1 en caso de que el número no esté en la lista
    printf("Introduce un numero a buscar: ");
    scanf("%d", &x);

    posicion = -1;
    for ( i = n-1; i >= 0; i-- ) {
        if ( lista[i] == x ) {
            posicion = i;
            break;
        }
    }
    printf("La posicion de %d en la lista es %d.\n", x, posicion);
}
```

# Bucles: la instrucción for (VIII)



```
#include <stdio.h>

void main() {
    int lista[] = {4, 3, 7, 2, 9, 5, 8, 1, 2, 6};
    int i, j, x, n = 10, posiciones[10];

    // Busca TODAS las posiciones en la lista del número introducido o
    // estableciéndolas a -1 en caso de que el numero no este en la lista
    printf("Introduce un numero a buscar: ");
    scanf("%d", &x);

    j = 0;
    for ( i = 0; i < 10; i++ ) {
        if ( lista[i] == x ) {
            posiciones[j] = i;
            j++;
        }
    }

    // Imprimimos por pantalla todas las posiciones encontradas
    for ( i = 0; i < j; i++ ) {
        printf("La posición de %d en la lista es %d.\n", x, posiciones[i]);
    }
}
```

# Bucles: la instrucción for (IX)



- La sentencia **continue** "continua" (salta) a la siguiente iteración de un bucle sin ejecutar en la iteración actual aquellas sentencias posteriores al continue

```
#include <stdio.h>

void main() {
    int lista[] = {4, 3, 7, 2, 9, 5, 8, 1, 2, 6}
    int i, x, n = 10;

    // Imprime por pantalla los números de una lista que son mayores a uno dado
    printf("Introduce un numero a buscar: ");
    scanf("%d", &x);

    for ( i = 0; i < n; i++ ) {
        // Si el numero i-esimo de la lista es menor o igual que x se sigue
        // la ejecución del bucle con el siguiente número
        if ( lista[i] <= x ) {
            continue;
        }

        printf("Número mayor que %d en la posición %d: %d.\n", x, i, lista[i]);
    }
}
```

# Contenidos



- **Bucles**
  - La instrucción for
  - **La instrucción while**
  - La instrucción do while
- Bucles anidados

# Bucles: la instrucción while (I)



- Un bucle **while** ejecuta un bloque de sentencias de forma iterativa mientras se cumpla una condición de fin
  - Sintaxis

```
while ( <condicion_fin> ) {
   <bloque_sentencias>
}
```

→ Las sentencia  condicion_fin es obligatoria

# Bucles: la instrucción while (II)



- Ejemplo

```
#include <stdio.h>

void main() {
   char opcion = 'X';

   while ( opcion != 'A' && opcion != 'B' ) {
      printf("Elija una opcion, A o B: ");
      scanf("%c", &opcion);
   }

   printf("Has elegido la opcion %c!\n", opcion);
}
```

# Bucles: la instrucción while (III)



- Bucle **while** “infinito”

```
while ( 1 ) {
   printf("Bucle infinito!\n");
}
```

# Bucles: la instrucción while (IV)



- Al igual que en los bucles for, la instrucción **break** rompe la ejecución del bucle while en el que se encuentra

```
#include <stdio.h>

void main() {
   char opcion;

   while ( 1 ) {
      printf("Elija una opcion, A o B: ");
      scanf("%c", &opcion);

      if ( opcion == 'A' || opcion == 'B' ) {
         break;
      }
   }

   printf("Has elegido la opcion %c!\n", opcion);
}
```

# Bucles: la instrucción while (V)



- Equivalencia entre bucles **for** y bucles **while**

```
int n = 10, i;

for ( i = 0; i < n; i++ ) {
   printf("Iteracion %d.\n", i);
}
```

```
int n = 10, i;

i = 0;
while ( i < n ) {
   printf("Iteracion %d.\n", i);
   i++;
}
```

# Bucles: la instrucción while (VI)



- Al igual que en los bucles for, la instrucción **continue** salta a la siguiente iteración del bucle while en el que se encuentra

```
#include <stdio.h>

#define AS       1
#define DOS      2
// El resto de macros irian aqui
#define SIETE    7
#define SOTA     0.5
#define CABALLO  0.5
#define REY      0.5

void main() {
    double puntos=0, cartas[40] = {SOTA, AS, CUATRO, DOS, REY, /* resto de cartas */ DOS};
    int opcion, c=0;

    while ( puntos < 7.5 ) {
        printf("Quieres carta (s/n)? ");
        scanf("%c", &opcion);
        if ( opcion == 's' || opcion == 'S' ) {
            puntos += cartas[c];
            c++;
            continue;
        }

        printf("Te has plantado!\n");
        break;
    }
    printf("Tienes %f puntos!\n", puntos);
}
```

# Contenidos



- **Bucles**
  - La instrucción for
  - La instrucción while
  - **La instrucción do while**
- Bucles anidados

# Bucles: la instrucción do while (I)



- Un bucle **do while** ejecuta un bloque de sentencias una vez y luego lo ejecuta de forma iterativa mientras se cumpla una condición de fin
  - Sintaxis

```
do {
   <bloque_sentencias>
}
while ( <condicion_fin> );
```

→ Las sentencia  condicion_fin es obligatoria

# Bucles: la instrucción do while (II)



- Ejemplo:

```
#include <stdio.h>

void main() {
   char opcion;

   do {
      printf("Elija una opcion, A o B: ");
      scanf("%c", &opcion);
   }
   while (opcion != 'A' && opcion != 'B' );

   printf("Has elegido la opcion %c!\n", opcion);
}
```

# Bucles: la instrucción do while (III)



- Al igual que en los bucles for y while, en los bucles **do while** también se puede hacer uso de las instrucciones **break** y **continue**

# Contenidos



- **Bucles**
  - La instrucción for
  - La instrucción while
  - La instrucción do while
- **Bucles anidados**

# Bucles anidados (I)



- Un bucle anidado es un bucle que está dentro de otro
  - La anidación de bucles puede ser imprescindible para la realización de cálculos complejos
  - Ejemplo:

```
double A[2][3] = {{1, 2, 3}, {4, 5, 6}};
double media;
int i, j;

// 2 bucles anidados para calcular la media de los numeros
// de una matriz
media = 0;
for ( i=0; i<2; i++ )  {
   for ( j=0; j<3; j++ ) {
      media += A[i][j];
   }
}
media /= 2*3;
```

# Bucles anidados (II)



- Un **bucle anidado** es un bucle que está dentro de otro
  - Puede existir más de un nivel de anidación
  - Ejemplo:

```
double A[2][3] = {{1, 2, 3}, {4, 5, 6}};
double B[3][1] = {{1}, {2}, {3}};
double C[2][1];
int i, j;

// 3 bucles anidados para multiplicar las matrices A y B
for ( i=0; i<2; i++ )  {
   for ( j=0; j<1; j++ ) {
      C[i][j] = 0.0;
      for ( k=0; k<3; k++ ) {
          C[i][j] += A[i][k] * B[k][j];
      }
   }
}
```

# Bucles anidados (III)



- En bucles anidados las instrucciones **break** y **continue** se aplican al bucle más interno al que pertenecen

```
double x, A[2][3] = {{1, 2, 3}, {4, 5, 6}};
int i, j, fila, columna;

// Busca la posicion de una matriz en la que se encuentra un numero dado
printf("Introduce el numero a buscar: ");
scanf("%f", &x);

fila = columna = -1;

for ( i=0; i<2; i++ )  {
    for ( j=0; j<3; j++ ) {
        if ( A[i][j] == x ) {
            fila = i;
            columna = j;
            break;            // Aquí se rompe el bucle j
        }
    }
    if ( fila != -1 ) {
        break;                // Aquí se rompe el bucle i
    }
}
printf("El numero %d esta en la posicion (%d, %d).\n", x, fila, columna);
```