Tema 4: Sentencias de control

Departamento de Automática
Universidad de Alcalá

Índice

Bloques de código

- Varias líneas de código se pueden agrupar con "{" y "}"
  - Esas líneas se tratan como una unidad
  - Las llaves nunca se cierran con punto y coma
  - Se usan para definir bucles, funciones, condiciones, etc

```
if (a != b)
  a = c;
  c = c+2;
```

≠

```
if (a != b) {
  a = c;
  c = c+2;
}
```

- Pueden definirse variables dentro de bloques
  - La variable solo existe dentro del bloque
  - Conviene reducir el ámbito de las variables

```
if (a != b) {
  int c;
  c = a;
  a = b;
  b = c;
}
```

Sentencias condicionales

Condiciones if-else

- Permiten la toma de decisiones
  - Se parte de una condición
  - El resultado de una condición es lógico
  - En C no existen variables lógicas
- Por convenio:
  - Valor 0 $\Rightarrow$ FALSO
  - Valor distinto de 0 $\Rightarrow$ VERDADERO

```c
int a = 0;
if (a)
   printf("VERDADERO");
else
   printf("FALSO");
```

=

```c
int a = 0;
if (a != 0)
   printf("VERDADERO");
else
   printf("FALSO");
```

El uso de `else` es opcional, ¡mejor no abusar de él!

Sentencias condicionales

Condiciones else-if (I)

- A veces las decisiones no son binarias
  - La decisión es más compleja que VERDADERO/FALSO
  - La construcción `else if` permite encadenas condiciones
- Evaluación secuencial
  - Sólo se ejecuta el bloque que cumple la condición
  - Si eso no sucede, se ejecuta el bloque `else`
  - `else` es opcional

```
if (condicion1)
  orden1;
else if (condicion2)
  orden2;
else if (condicion3)
  orden3;
else
  orden4;
```

Sentencias condicionales

Condiciones else-if (II)

Ejemplo

```
int main() {
  int mes;
  printf("Introduzca un mes");
  scanf("%d", &mes);
  if (mes == 1)
    printf("Enero");
  else if (mes == 2)
    printf("Febrero");
  .........
  else if (mes == 12)
    printf("Diciembre");
  else {
    printf("Error, mes no reconocido");
    return(-1);
  }
  return 0;
}
```

Sentencias condicionales
Condiciones switch (I)

- Similar al `else if`
  - A veces es más cómodo
- Evalúa una expresión y, según su valor, toma varios caminos
- Compara la expresión con constantes
- Si no satisface ningún `case`, se ejecuta `default`
- La proposición `break` obliga a salir del `switch`

```
switch (expresion) {
  case constante1:
    orden1;
    break;
  case constante2:
    orden2;
    break;
  case constante3:
    orden3;
    break;
  default:
    orden4;
}
```

Sentencias condicionales

Condiciones switch (II)

Ejemplo (1/2)

```
int main() {
  char caracter;
  int numA=0, numE=0, numI=0, numO=0, numU=0;
  printf("Introduzca vocales\n");
  caracter = getchar();
  while(caracter != EOF) {
    switch (caracter) {
      case 'a':
        numA++;
        break;
      case 'e':
        numE++;
        break;
      case 'i':
        numI++;
        break;
```

Sentencias condicionales
Condiciones switch (III)

Ejemplo (2/2)

```
        case 'o':
          numO++;
          break;
        case 'u':
          numU++;
          break;
        default:
          printf("Error: No ha introducido una vocal");
          return -1;
      }
      caracter = getchar();
    }
    printf("a:%d, e:%d, i:%d, o:%d, u:%d", numA, numE,
        numI, numO, numU);
    return 0;
}
```

Bucles

Bucles while y for (I)

Sintaxis for

```
for (expr1; expr2; expr3)
  proposicion
```

=

Sintaxis while

```
expr1;
while (expr2) {
  proposicion
  expr3;
}
```

- Los bucles for y while son equivalentes
  - A veces es más natural usar uno u otro
- El ";" puede omitirse
  - La sentencia `for (;;)` es un bucle infinito
  - Se pueden separar operadores con comas

Bucles

Bucles while y for (II)

Ejemplo: Invertir una cadena

```
void invertir(char cadena[]) {
  int i, j;
  char c;

  for (i=0, j=strlen(cadena)-1; i<j; i++, j--) {
    c = cadena[i];
    cadena[i] = cadena[j];
    cadena[j] = c;
  }
}
```

Bucles
Bucles do-while (I)

- Los bucles `for` y `while` primero evalúan
  - No se garantiza la ejecución
- El bucle do-while evalúa al final
  - Garantiza la ejecución al menos una vez
  - A veces un do-while ahorra código
- Se puede leer como "haz ... mientras ..."

Sintaxis do-while

```
do
   proposicion
while (condicion);
```

Bucles
Bucles do-while (II)

Ejemplo

```
int main() {
  int numero;

  printf("Escoja un numero entre el 10 y el 20\n");

  do {
    scanf("%d", &numero);
  } while ((numero<10) || (numero>20));

  return 0;
}
```

Bucles
Break y continue (I)

- A veces es necesario romper un bucle: break y continue
  - `break`: Obliga a salir del bucle
  - `continue`: Salta a la siguiente iteración
  - Ambas sentencias son válidas dentro de un bucle
  - `break` se usa además con `switch`
  - Afectan al un único bucle

Uso de break

```
for (i=0; i<MAX, i++) {
  orden1;
  if (a == b) break;
  orden2;
}
```

Uso de continue

```
for (i=0; i<MAX, i++) {
  orden1;
  if (a == b) continue;
  orden2;
}
```

Bucles

Break y continue (II)

Ejemplo con break

```
#include<stdio.h>
#include<string.h>
#define MAX_CARACTERES 100

int main() {
   int i;
   char palabra[MAX_CARACTERES];
   scanf("%s", palabra);

   for (i=0; i<strlen(palabra)-1; i++) {
      if (palabra[i] == 'a') break;
   }

   printf("%d\n", i);
   return 0;
}
```

Bucles
Break y continue (III)

Ejemplo con continue

```
#include<stdio.h>

#define MAX 100

int main() {
  int i;

  for (i=0; i<MAX; i++) {
    if (i %2 == 0) continue;
    printf("%d\n", i);
  }

  return 0;
}
```

Bucles
Etiquetas y goto

No usar la sentencia goto ... nunca ... jamás

Cada vez que alguien usa goto, muere un gatito

Ejercicio

1. Un año es bisiesto si es divisible entre 4, salvo que sea divisible entre 100 y no entre 400. Realizar un programa que muestre todos los años bisiestos del siglo XXI.

Ejercicio
Solución

Solución ejercicio

```
#include<stdio.h>

int esBisiesto(int anyo) {
  if (anyo %4 == 0) {
    if (((anyo%100)==0) && ((anyo%400)!=0)) return 0;
    else return 1;
  }
  return 0;
}
int main() {
  int i;
  for (i=2001; i<=2100; i++) {
    if (esBisiesto(i) != 0) printf("%d\n", i);
  }
  return 0;
}
```