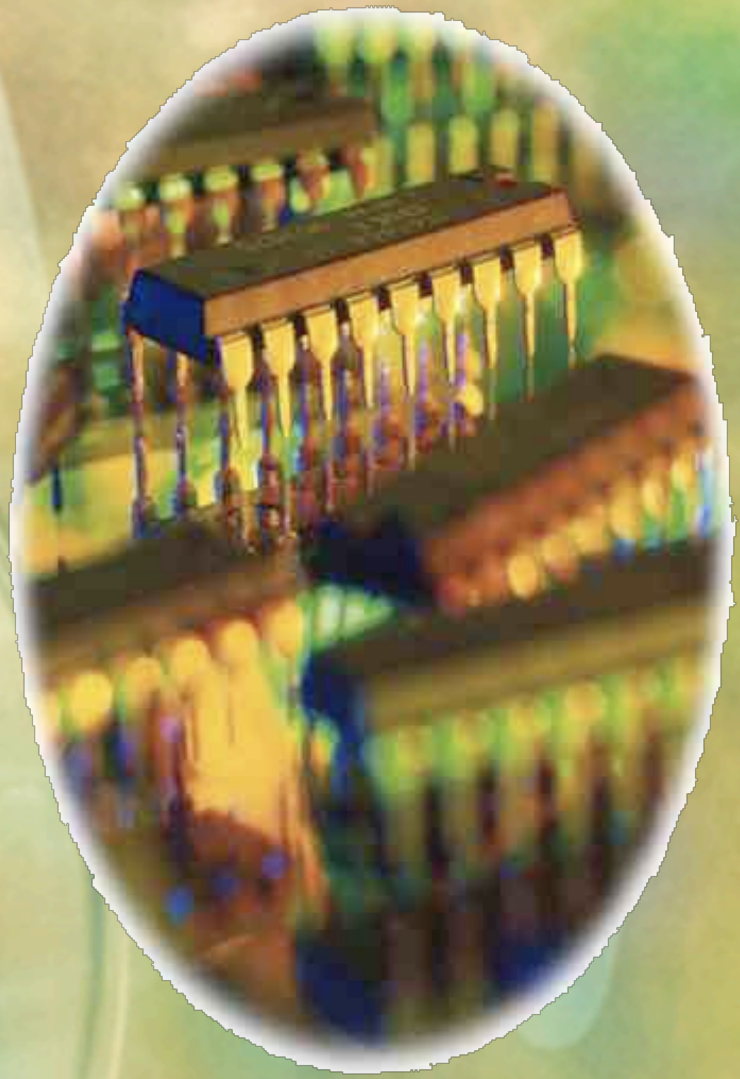
TEMA

2

El lenguaje de programación C (continuación): Estructuras de control



Dept. Ciencias de la Computación e I.A.

Universidad de Granada

Índice

- Estructura secuencial.
  - Ejemplos
- Estructuras condicionales.
  - Condicional Simple
  - Condicional Doble
  - Condicional Múltiple
  - Ejemplos
- Estructuras repetitivas.
  - Bucles Controlados por condición
  - Bucles Controlados por contador
  - Ejemplos

Conceptos Básicos

Las estructuras de control de un lenguaje de programación se refieren al orden en que las instrucciones de un algoritmo se ejecutarán. El orden de ejecución de las sentencias o instrucciones determinán el flujo de control.

Las tres estructuras básicas de control son:

- ***secuencia***
- ***selección***
- ***repetición***

Conceptos Básicos

*Un programa propio puede ser escrito utilizando las tres estructuras de control básicas (*Bôhm y Jacopin (1996)*).*

Un programa se define como *propio* si cumple lo siguiente:

- Posee un sólo punto de entrada y salida o fin para control del programa.
- Existen caminos desde la entrada hasta la salida que se pueden seguir y que pasan por todas las partes del programa.
- Todas las instrucciones son ejecutadas y no existen lazos o bucles infinitos.

Estructura Secuencial

Las sentencias se ejecutan sucesivamente, en el orden en que aparecen. No existen “saltos” o bifurcaciones.

```
/*Calculo de Raices de una ecuacion de grado 2*/
#include <stdio.h>
#include <math.h>
int main(){
 double a,b,c,r1,r2;
 printf("\nIntroduce coeficiente de 2o grado: ");
 scanf("%lf",&a);
 printf("\nIntroduce coeficiente de 1er grado: ");
 scanf("%lf",&b);
 printf("\nIntroduce coeficiente independiente: ");
 scanf("%lf",&c);
 r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
 r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
 printf("Las raices son %lf y %lf \n",r1,r2);
 return 0;}
```

Estructura Secuencial

Si bien el programa anterior es correcto, no es capaz de manejar situaciones excepcionales.

Por ejemplo, ¿qué pasa en la sentencia:

```
r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
```

si `a = 0` o `b*b-4*a*c < 0`?

En el primer caso, obtendríamos un error de ejecución por intentar hacer una división por cero.

Solución: primero comprobar (mediante **estructuras condicionales**), y no efectuar el cálculo si no es posible.

Estructura Condicional

Tambien recibe el nombre de *"estructura de selección"*

Permite elegir entre diferentes cursos de acción en función de condiciones.

```
Si la nota del examen es mayor o igual que 5
     mostrar "Aprobado"
```

Si la condición es *verdadera*, entonces se ejecuta la sentencia ***mostrar,*** y luego el programa continuaría en la sentencia siguiente al ***Si***

Si la condición es *falsa*, la sentencia **mostrar** se ignora y el programa continúa

Estructura Condicional Simple

La instrucción en pseudo código

```
Si la nota del examen es mayor o igual que 5
     mostrar "Aprobado"
```

Se traduce a C mediante una sentencia condicional simple:

```
if ( nota >= 5 )
   printf ("Aprobado");
```

La forma general es:

```
if (<condicion>)
   <bloque if>;
```

Una expresión lógica

Una sentencia o conjunto de sentencias (encerradas entre { })

Cálculo de Raíces

```c
/*Calculo de Raices de una ecuacion de grado 2*/
#include <stdio.h>
#include <math.h>
int main(){
 double a,b,c,r1,r2;
 printf("\nIntroduce coeficiente de 2o grado: ");
 scanf("%lf",&a);
 printf("\nIntroduce coeficiente de 1er grado: ");
 scanf("%lf",&b);
 printf("\nIntroduce coeficiente independiente: ");
 scanf("%lf",&c);
 if (a!=0) {
      r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
      r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
      printf("Las raices son %lf y %lf \n",r1,r2);
 }
 return 0;
}
```

Cálculo de Raíces (2)

Esta aproximación no calcula raíces si ***a=0***.

¿Cómo hacer que también calcule la solución si a=0 (ecuación de primer grado)?

```
if (a==0)
     printf("Tiene una única raíz %lf \n",(-c/b) );
```

Algo a considerar: las condiciones son excluyentes (***a vale cero*** o ***a vale distinto de cero***)

Estructura Condicional Doble

Permite elegir entre 2 cursos de acción diferentes en función del valor de verdad de una expresión lógica.

```
Si la nota del examen es mayor o igual que 5
     mostrar "Aprobado"
Si no
     mostrar "Suspenso"
```

En C:

```
if ( nota >= 5 )
  printf("Aprobado");
else
  printf("Suspenso");
```

La forma general es:

```
if( <condicion> )
 < bloque if >;
else
 < bloque else >;
```

Cálculo de Raíces

En la ecuación de segundo grado, cuando **a = 0**, entonces la raíz es única y vale ***-c/b***

```
if (a!=0) {  // Inicio Bloque
     r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
     r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
     printf("Las raíces son %lf y %lf\n",r1, r2);
   }  // Fin Bloque
else {  // Inicio Bloque
     r1 = -c / b;
     printf("La unica raiz es: %lf \n", r1);
   }  // Fin Bloque
```

Los bloques comienzan con { y terminan con }

Anidamiento de Estructuras Condicionales

```
Si la nota es mayor o igual
que 9
   imprimir "Sobresaliente"
 else
  Si la nota es mayor o
  igual que 7
      imprimir "Notable"
   else
  Si la nota es mayor o igual
  que 5
       imprimir "Aprobado"
     else
        imprimir "Suspenso"
```

```
if (nota>=90)
  printf("A");
else if (nota>=80)
       printf("B");
     else if (nota>=70)
            printf("C");
          else if (nota>=60)
                 printf("D");
               else printf("E");
```

¿ Cuando se ejecuta cada instrucción ?

```
if (condic_1){
   inst_1;
   if (condic_2){
       inst_2;
   } else {
       inst_3;
   }
   inst_4;

} else {
   inst_5;
}
```

	condic_1	***condic_2***
inst_1	*true*	*independiente*
inst_2	*true*	*true*
inst_3	*true*	*false*
inst_4	*true*	*independiente*
inst_5	*false*	*independiente*

Ejemplo: el mayor de tres números

- ¿Cómo hacer para calcular el máximo de tres números *a*, *b* y *c* y almacenar el resultado en una variable *max* cuyo valor se mostrará por pantalla al final?

Ejemplo: el mayor de tres... (2)

```
/* Programa para calcular el máximo de tres números
   que se almacenará en otra variable y se
   mostrará en la pantalla*/
#include <stdio.h>
int main() {
   int a, b, c, max;

   printf("Valor de a: ");
   scanf("%d",&a);
   printf("Valor de b: ");
   scanf("%d",&b);
   printf("Valor de c: ");
   scanf("%d",&c);

   ...
```

Ejemplo: el mayor de tres... (3)

```
  ...
  if ((a>=b) && (a>=c))
      max = a;
  if ((b>=a) && (b>=c))
      max = b;
  if ((c>=a) && (c>=b))
      max = c;

  printf("El mayor es %d \n", max);
  return 0;
}
```

Ejemplo: el mayor de tres... (4)

```
...
if (a>=b)
   if (a>=c)
      max = a;
   else
      max = c;
else
   if (b>=c)
      max = b;
   else
      max = c;
...
```

Opción 2

Ejemplo: el mayor de tres... (5)

```
...
if (a>=b)
   max = a;
 else
   max = b;
if (c>max)
   max = c;
...
```

Opción 3

Ejemplo: el mayor de tres… (6)

	Condiciones	Asignaciones
Opción 1	3	1
Opción 2	2	1
Opción 3	2	1 ó 2

Buscaremos un término medio entre Eficiencia y Elegancia

Ejercicios

- Completar el programa de la ecuación de segundo grado, teniendo en cuenta los valores del discriminante también.
- Implementa un programa que calcule el mayor de cuatro números.

Ejemplo

```c
#include <stdio.h>

int main(){
  int dato1,dato2;
  char opcion;

  printf("Introduce el primer operando: ”);
  scanf(“%d”,&dato1);
  printf("el segundo operando: ”);
  scanf(“%d”,&dato2);
  printf("(S) sumar, (R) restar: ”);
  scanf(“%c”,&opcion);

  if (opcion == 'S')
       printf("Suma = %d \n”,dato1+dato2 );

  if (opcion == 'R')
       printf("Resta = %d \n”,dato1-dato2 );

  if ((opcion != 'R') && (opcion != 'S'))
       printf("Ninguna operación \n");
  return 0;
}
```

En este caso, las condiciones no son excluyentes

Ejemplo

Más eficiente

```
#include <stdio.h>

int main(){
  int dato1,dato2;
  char opcion;

  printf("Introduce el primer operando: ");
  scanf("%d",&dato1);
  printf("el segundo operando: ");
  printf("(S) sumar, (R) restar: ");
  scanf("%c",&opcion);

  if (opcion == 'S')
      printf("Suma = %d \n",dato1+dato2 );

  else if (opcion == 'R')
      printf("Resta = %d \n",dato1-dato2 );

  else /*if ((opcion != 'R') && (opcion != 'S'))*/
      printf("Ninguna operación \n");
  return 0;
}
```

Estructura Condicional Múltiple

Permite definir en una sola estructura, una serie de pares (condición, acción), con condiciones mutuamente excluyentes.

Muy utilizada en la construcción de menús.

```
switch (<expresión>) {
case<constante1>:<sentencias1>
                        break;
case <constante2>:<sentencias2>
                      break;
   ..........................
[default: <sentencias>]
}
```

Estructura Condicional Múltiple

- **<expresión>** es una expresión entera o carácter.
- **<constante1>** ó **<constante2>** es un literal tipo entero o tipo carácter.
- ***switch*** sólo comprueba la igualdad.
- No debe haber dos casos (**case**) con la misma ***<constante>*** en el mismo ***switch***. Si esto ocurre, sólo se ejecutan las sentencias correspondientes al caso que aparezca primero.
- La ejecución de sentencias empieza en la etiqueta de case cuya **<constante>** es igual a la **<expresión>**, y continúa hasta el final del **switch**, o hasta el siguiente **break**.
- El identificador especial ***default*** permite incluir un caso por defecto, que se ejecutará si no se cumple ningún otro. Se suele colocar como el último de los casos.
- En las estructuras condicionales múltiples también se permite el anidamiento.

Ejemplo 1

```
#include <stdio.h>

int main(){
        int dato1,dato2;
        char opcion;

        printf("Introduce el primer operando: ");
        scanf("%d",&dato1);
        printf("Introduce el segundo operando: ");
        scanf("%d",&dato2);
        printf("Selecciona (S) sumar, (R) restar: ");
        scanf("%c",&opcion);

        switch (opcion){
          case 'S': {printf("Suma = %d \n",dato1+dato2);
                     break;}

          case 'R': {printf("Resta = %d \n",dato1-dato2);
                      break;}

          default: printf("Ninguna operacion\n");
        }
        return 0;
}
```

Ejemplo 2

```
printf("Introduce el primer operando: ”);
scanf("%d", &dato1);
printf("Introduce el segundo operando: ”);
scanf("%d", &dato2);
printf("Selecciona (S) sumar, (R) restar: ”);
scanf(”%c", &opcion);

switch (opcion){
 case 's':
 case 'S': {printf("Suma = %d \n”, dato1+dato2);
            break;}

 case 'r':
 case 'R': {printf("Resta = %d \n", dato1-dato2);
            break;}

 default: printf("Ninguna operacion\n”);
}
      }
}
```

Estructuras Repetitivas

Estructuras Repetitivas

Las estructuras repetitivas son también conocidas como ***bucles***, *ciclos o lazos.*

Una estructura repetitiva permite la ejecución de un conjunto de sentencias:

- hasta que se satisface una determinada condición (controladas por condición o controladas por centinela)
- un número determinado de veces (controladas por contador)

Bucles Controlados por Condición

Se ejecuta el conjunto de sentencias de interés mientras la condición sea verdadera.

Existen dos formas básicas de construcción:

Pre - Test

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

Primero se pregunta y luego se ejecuta

Post - Test

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

Primero se ejecuta luego se pregunta

Ejemplo

Escribir 10 líneas con 4 estrellas cada una

```
cont = 1;

do {
 printf("****\n”);
 cont = cont + 1;
}while (cont <= 10);
```

```
cont = 0;

do {
 printf("****\n");
 cont = cont + 1;
}while (cont < 10);
```

```
cont = 1;

while (cont <= 10){
 printf("****\n");
 cont = cont + 1;
}
```

```
cont = 0;

while (cont < 10){
printf("****\n”);
 cont = cont + 1;
}
```

Ejemplo

Escribir *tope* líneas con 4 estrellas cada una

```
scanf("%d",&tope);
cont = 0;

do {
 printf("****\n”);
 cont = cont + 1;
}while (cont < tope);
```

Problema si *tope* <= *0*

Solución

```
scanf("%d",&tope);
cont = 0;

while (cont < tope){
 printf("****\n");
 cont = cont + 1;
}
```

Ejemplo

Escribir los números pares entre [0..*tope*]

```
scanf("%d",&tope);
i = 0;
while (i <= tope){
   printf("%d",i);
   i = i + 2;
}
```

¿Qué ocurre si el código anterior se cambia por :

```
scanf("%d",&tope);
i = 0;
while (i <= tope){
   i = i + 2;
   printf("%d \n",i);
}
```

El 0 no se imprime e imprime un número par de más

Ejemplo: control por centinela

Sumar valores que se leen por teclado hasta que se lee -1. (el valor utilizado para detener el procesamiento se conoce como centinela)

```
suma = 0;
do
{scanf("%d",&valor);
 suma = suma + valor;
}while (valor != -1)
printf("La suma es %d”,
suma);
```

Se procesa el valor centinela. Esta clase de problemas se evita utilizando la técnica de *lectura anticipada*.

Lectura anticipada: se usa un bucle pre-test y se comprueba el primer valor.

```
suma = 0;
scanf("%d",&valor);
while (valor != -1){
   suma = suma + valor;
   scanf("%d",&valor);
 }
printf("La suma es %d”,
suma;
```

Ciclos post-test como Filtros en la Entrada de Datos

Los ciclos post-test son útiles para forzar que los datos de entrada pertenezcan a un rango o conjunto de opciones predeterminados.

```
// filtro para valores positivos
do
 {printf("Introduzca un valor > 0:”);
  scanf("%d”,&valor);
 }while (valor < 0);
```

```
// filtro para 's','S','n','N'
do
 {printf("Desea Salir (s/n)?:”);
  scanf("%c”,&opc);
 }while((opc != 's')&&(opc !='S')
      &&(opc != 'n')&&(opc != 'N'));
```

Ejemplo: lectura anticipada (1)

```
/* Programa que va leyendo números enteros hasta que
   se introduzca el cero. Imprimir el número de
   pares e impares introducidos */

#include <stdio.h>
int main(){
  int ContPar, ContImpar, valor;

  ContPar=0;
  ContImpar=0;
  printf("Introduce valor: \n”);
  scanf("%d",&valor);
  ...
```

Ejemplo: lectura anticipada (2)

```
   ...
   while (valor != 0) {
       if (valor % 2 == 0)
             ContPar = ContPar + 1;
       else
             ContImpar = ContImpar + 1;
       printf("Introduce valor: \n”);
       scanf("%d", &valor);
   }
   printf("Fueron %d pares y %d impares \n”,
   ContPar, ContImpar);
   return 0;
}
```

Bucles sin fin

Bucle al que la evaluación de la condición nunca le permite dejar de ejecutarse.

```
contador = 2;
while (contador < 3) {
      contador = contador - 1;
      printf("%d\n",contador);
      }
```

```
contador = 1;
while (contador != 10) {
      contador = contador + 2;
      }
```

Utilidad: filtros con bucles post-test

```
/* Introducir un único valor, pero positivo.
   Imprimir el coseno.
   Entrada: Un valor entero positivo.
   Salida: El coseno del valor introducido.*/
#include <stdio.h>
#include <math.h>
int main(){
  int valor;

  do {
      printf("\nIntroduzca un valor positivo: ");
      scanf("%d",&valor);
  }while (valor < 0);
  printf("%lf \n",cos(valor));
  return 0;
}
```

Sigue pidiendo el valor mientras sea negativo (hasta que sea positivo)

Utilidad: filtros con bucles… (2)

```
char opcion;
do{
  printf("Introduzca una opción: \n”);
  scanf(”%c", &opcion);
}while ((opcion!='s') && (opcion!='S') &&
      (opcion!='n') && (opcion!='N') );
```

Sigue pidiendo la opción mientras sea distinta de ‘s’, ‘S’, ‘n’ y ‘N’ (hasta que sea una de ellas)

Bucles controlados por contador

Se utilizan para repetir un conjunto de sentencias un número fijo de veces. Se necesita una variable controladora, un valor inicial, un valor final y un incremento.

Ejemplo

```
/* Hallar la media de 5 números enteros.
   Entradas: 5 números enteros (se leen en valor).
   Salidas: La media de los 5 números.*/
#include <stdio.h>
int main(){
int i, valor, suma;
double media;

suma = 0;
i = 1;
while (i <= 5){
   printf("Introduce el número %d :\n",i);
   scanf("%d",&valor);
   suma = suma + valor;
   i++;
}
media = suma / 5.0;
printf("La media es %lf \n",media);
return 0;
}
```

Valor inicial del contador

Prueba de límites del valor del contador

Incremento del valor del contador

Bucles controlados por Contador: FOR

SI conocemos exactamente la cantidad de veces que necesitamos repetir un conjunto de sentencias, entonces podemos usar un bucle FOR.

En general, los bucles controlados por contador requieren

- una variable de control o contador
- Un valor inicial para el contador
- Un valor final para el contador
- Una condición para verificar si la variable de control alcanzó su valor final.
- Un valor de incremento (o decremento) con el cual se modifica la variable de control en cada bucle

Bucles controlados por Contador: FOR

La forma general del bucle FOR es:

for (inicializacion; condicion continuación;incremento)
< conjunto de sentencias>

Ejemplo: imprimir los valores enteros entre 1 y 10

```
for( nro = 1; nro <= 10; nro = nro + 1)
      printf("%d \n",nro);
```

Relación entre FOR y WHILE

Se debe notar que los bucles FOR se pueden reescribir como bucles WHILE

```
for ( inicializacion; condicion continuación, incremento )
   < conjunto de sentencias>
```

```
inicializacion
while (condicion continuacion)
   < conjunto de sentencias>
   incremento
```

Ejemplos

```
/*tabla de multiplicar de un nro*/
int i,nro;
scanf("%d",&nro);
for(i = 1; i <= 10; i = i + 1){
  printf("%d x %d = %d\n",nro,i,nro * i);
}
```

```
/*sumar los nros pares entre 2 y 200*/
int i,suma;
suma = 0;
for(i = 2; i <= 200; i = i + 2){
 suma = suma + i;
}
printf("la suma es %d \n", suma);
```

Ejemplos

```
/*tabla de multiplicar de un nro*/
int i,nro;
scanf("%d",&nro);
i = 1;
while (i <= 10){
  printf("%d x %d = %d\n",nro,i,nro * i);
  i = i+1;
}
```

```
/*sumar los nros pares entre 2 y 200*/
int i,suma;
suma = 0;
i = 2;
while (i <= 200){
   suma = suma + i;
   i = i+2;
}
printf("la suma es %d \n",suma);
```