

Tema 4

Estructures de control en C

1 Sentències simples i compostes

2 Estructures de control

2.1 Estructura seqüencial

2.2 Estructures de selecció

2.3 Estructures repetitives

5. Sentències simples i compostes

Una sentència és una instrucció o conjunt d'instruccions que fan una acció determinada.

- ✓ Sentència simple: una única instrucció
- ✓ Sentència composta: conjunt o bloc d'instruccions { }

1.1 Sentències simples

Són les formades per una única instrucció.

Al final d'una sentència simple cal posar punt i coma (separador de sentències).

Tipus de sentències simples (instruccions pròpiament dites):

- ✓ Declaratives: declaració de tipus, variables i prototipus de funcions
- ✓ Assignacions: per a donar valors a variables (`x = 10;`)
- ✓ Crides a funció: `printf()`, `maxim()` ...

1.2 Sentències compostes (blocs)

Un bloc està format per una llista de sentències agrupades entre claus { }. Es pot utilitzar en qualsevol lloc on està permès posar una sentència simple.

```
if ( (100 mod num) == 0 )
```

```
{  
    printf("100 és divisible per %d\n", num);  
    q_divisors = q_divisors + 1;  
}
```

Sol utilitzar-se per a dir que més d'una instrucció han d'executar-se si es complix una determinada condició. És a dir, a l'exemple anterior, si 100 és divisible per num, volem que s'execute no una única instrucció, sinó 2: un printf i una assignació. Per tant, estes 2 instruccions cal tancar-les en un bloc.

Una sentència composta pot estar formada per altres sentències compostes. És a dir: un bloc pot contindre altres blocs.

```
num = 1;
```

```
q_divisors = 0;
```

```
while ( num <= 100 )
```

```
{  
    if ( (100 mod num) == 0 )  
    {  
        printf("100 és divisible per %d\n", num);  
        q_divisors = q_divisors + 1;  
    }  
    num = num + 1;  
}
```

```
printf("El 100 té %d divisors\n", q_divisors);
```

Els blocs no acaben en punt i coma. Per tant, darrere de ' } ' no posarem ' ; '.

Recordem que cal anar en compte en l'àmbit i visibilitat de les variables dins dels blocs i quan sortim d'ells.

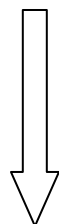
2 Estructures de control

Les estructures de control indiquen l'ordre en què s'executen les sentències. Com en qualsevol llenguatge de programació estructurat, tenim 3 tipus d'estructures:

- Seqüencial
- De selecció
- Repetitives

2.1 Estructura seqüencial

És l'estructura bàsica. Simplement consisteix en una instrucció a continuació de l'altra. Primer s'executarà la primera (la de més amunt) i, a partir d'eixa, s'executaran les altres per ordre d'aparició:



```
printf("Radi:");  
  
scanf("%d", &radi);  
  
printf("L'àrea és %f", 3.14 * radi * radi)
```

Exercicis

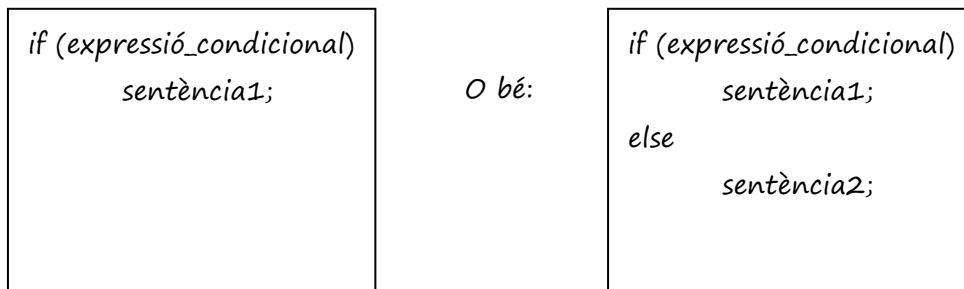
1. Fes un programa que permeti resoldre l'equació de 1r grau $ax + b = 0$. Cal introduir per teclat els valors de a i b . El programa ha de mostrar finalment el valor de x .
2. El servei d'endocrinologia de l'Hospital de La Ribera necessita un programa que calcule el pes recomanat d'una persona. Escriu un programa que llixi l'altura en metres i l'edat d'una persona i mostri el seu pes recomanat segons la fórmula:

$$\text{pes} = (\text{altura_en_cm} - 100 + 10\% \text{edat}) * 0.9$$

2.2 Estructures de selecció

Són les mateixes que veïrem als algorismes: el “SI” i el “EN_CAS_QUE”. És a dir, són les estructures que permeten executar un conjunt d'instruccions només si es complix una determinada condició. Ara vorem com s'implementen en C.

2.2.1 Estructures de selecció simples i dobles



Si sentència1 i/o sentència2 són compostes, hem de tancar-les en un bloc mitjançant { }. Per exemple:

```
if (num > 0)
    printf("La seua arrel val %f\n", sqrt(num));
else
    {
        printf("No té arrel perquè és negatiu\n");
        num = -num;
    }
printf("El número positiu és %d\n", num)
```

Nota:

El valor de l'expressió_condicional es considera vertader quan és distint de 0 i fals en qualsevol altre cas. Per exemple, les següents expressions són equivalents:

```
if (saldo != 0) ...    →   if (saldo) ...

if ((num % 2) != 0) ... →   if (num % 2) ...

if ((num % 2) == 0) ... →   if (!(num % 2)) ...
```

Exercicis

3. Programa que calcule el màxim de 2 números.
4. Escriu un programa que, donat un caràcter alfabètic, el passe de minúscula a majúscula i viceversa.
5. Fes un programa que calcule les solucions reals de l'equació de 2n grau

$$ax^2 + bx + c = 0 \quad \text{mitjançant la fórmula: } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Notes: - Cal comprovar si té solució (allò que està dins l'arrel ha de ser positiu).
- Per a calcular l'arrel usarem la funció `sqrt()`, de la llibreria `<math.h>`.

Esta llibreria (`math.h`) pot donar error quan compilem sota Linux. Per a evitar-ho, cal compilar amb l'opció `"-lm"`. En Geany cal anar a Construir → Selecciona incusions i arguments → Construir: `gcc -Wall -o "%e" "%P" -lm`

2.2.2 Estructures de selecció niuades

Dins d'un `if` van sentències. Per tant, com el `if` també és una sentència, puc posar un `if` dins d'un altre `if` (i així, successivament).

Ara bé, hem d'anar molt amb compte de fer correspondre el `else` al `if` que volem i no a un altre. Per exemple, fixa't en estos 2 programes:

```
if (n>0)
    if (a>b)
        z=a;
else
    z=b;
```

```
if (n>0)
{
    if (a>b)
        z=a;
}
else
    z=b;
```

No són equivalents, ja que en el primer cas, el `else` fa referència al segon `if` (encara que no estiga ben tabulat), mentre que en el segon cas fa referència al primer `if`.

Exercicis

6. Programa que calcule el màxim de 3 números.
7. Escriu un programa que calcule el valor de la següent funció matemàtica per a un valor de `x` passat com a paràmetre:

$$f(x) = \begin{cases} 2x^2 - x - 5 & x < 0 \\ 5x^3 - 4x^2 & 0 \leq x < 5 \\ 3x^5 - 3x^3 + x^2 & 5 \leq x \end{cases}$$

5

Nota: per a calcular a^b usarem funció `pow(a, b)`, inclosa a la llibreria `<math.h>`.

2.2.3 Estructures de selecció múltiples

```
switch (expressió)
{
    case const_1:      sentència_1;
                      [break;]

    case const_2:      sentència_2;
                      [break;]

    ...

    case const_n:      sentència_n;
                      [break;]

    [default:          sentència_d]
}
```

És una estructura d'alternativa múltiple, on s'executen 0, 1 o més branques.

Funcionament:

- ✓ L'expressió i les constants han de ser del mateix tipus: enter o caràcter.
- ✓ Primer que res, s'avalua l'expressió .
- ✓ Si l'expressió és igual a la constant del primer case:
 - S'executen TOTES les sentències fins al final del switch o bé fins a un break.
- ✓ Si no, tornem a comparar l'expressió amb la constant del següent case (i fem el mateix).
- ✓ Els break són opcionals. Servixen per a eixir del switch.
- ✓ La part de default és opcional. S'executarà la seua sentència associada quan l'expressió no siga igual a cap de les constants dels case anteriors.

Exercicis

8. Fes un programa que llija de teclat 2 números i una operació aritmètica (S/R/P/D). El programa farà el càlcul i imprimirà el resultat.
9. Demana número de mes i de dia. Comprova que són correctes i digues a quin número de dia de l'any correspon. Suposem anys no bixestos. Per exemple, per al dia 5 del mes 2, caldrà mostrar que és el dia 36 de l'any.

2.3 Estructures repetitives

Són les mateixes que veïem als algorismes: el “PER”, el “MENTRE-FES” i el “FES-MENTRE”. És a dir, són les estructures que permeten executar un conjunt d'instruccions diverses vegades (mentre es complisca una condició). Ara vorem com s'implementen en C.

2.3.1 Bucle while

```
while (expressió_condicional)
    sentència;
```

Equival al “MENTRE-FES” que veïem al pseudocodi. Com ja sabem, mentre l'expressió_condicional siga certa (distinta de 0), s'executarà la sentència.

Si volem que es repetisca més d'una sentència, hem de tancar-les en un bloc { }.

2.3.2 Bucle do-while

```
do
    sentència;
while (expressió_condicional);
```

Equival al “FES-MENTRE” del pseudocodi. Com ja sabem, s'executarà la sentència mentre es complisca l'expressió_condicional. A diferència del *while*, la sentència s'executarà com a mínim 1 vegada, ja que primer executa i després comprova la condició.

Igual que abans, si la sentència és composta, la tancarem en un bloc { }.

Exercici

10. Programa que ens demane el radi d'una circumferència. Posteriorment ens mostrarà un menú amb 4 opcions:

1. Diàmetre 2. Perímetre 3. Superfície 4. Eixir

L'usuari triarà una opció. El programa calcularà el corresponent valor i tornarà a fer el mateix (demanar el radi, etc) a no ser que s'haja triat l'opció d'eixir.

2.3.3 Bucle for

Equival al "PER" que veiérem al pseudocodi.

```
for ( [inicialització] ; [condició] ; [actualització] )  
    sentència;
```

Per descomptat, si la sentència és composta, també la tancarem amb { }.

Les 3 parts del *for* són opcionals. Però no els ' ; '.

- **Inicialització**. Instrucció o instruccions que s'executaran només abans de la primera iteració del *for* (i abans d'avaluar la condició). Sol usar-se per a inicialitzar la variable que fa de comptador.
- **Condició**. Mentre es complisca eixa condició (que pot ser composta), s'executaran les sentències de dins del *for*. Si no es complix, eixirem del *for*.
- **Actualització**. Instrucció o instruccions que s'executaran després de cada iteració del *for* (després d'executar les sentències de dins del *for*).

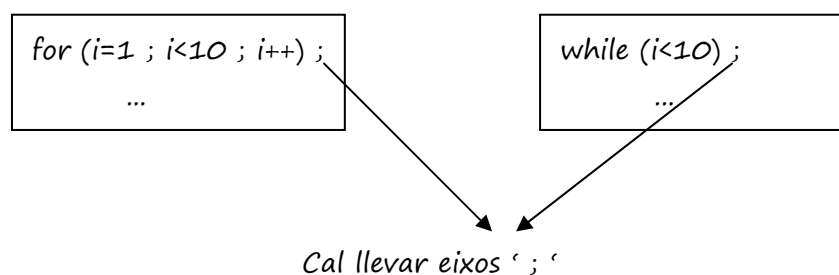
És a dir, el funcionament és el següent:

1r) Executar la inicialització

2n) Avaluar la condició.

- 2.1) Si és certa:
 - 2.1.1) executar sentències
 - 2.1.2) executar l'actualització
 - 2.1.3) tornar al pas 2n)
- 2.2) Si no és certa, el *for* acaba

Alerta! No hem de posar el ' ; ' després del *for*. No dóna error de compilació però no farà el resultat que volem. Si posem eixe ' ; ', la sentència de després queda fora del bucle i, per tant, només s'executarà 1 volta. Pel mateix motiu, tampoc hem de posar el ' ; ' després de la condició del *while*. És a dir:



Exercicis

11. Programa que calcule el màxim de 10 números introduïts per teclat.
12. Programa que calcule el màxim, mínim i mitjana de 10 números entrats per teclat.
13. Programa que calcule el factorial d'un número introduït per teclat ($n!$) tenint en compte que:
 $0! = 1$
$$n! = n * (n-1) * (n-2) * \dots * 2 * 1 \quad (\text{sent } n > 1)$$

Feu-ho amb els 3 tipus de bucles que hem vist, i fent els càlculs incrementant i decrementant el comptador (total: 6 solucions).
14. Programa que mostre la taula de multiplicar del 9.
15. Programa que demane una taula de multiplicar i la mostre.
16. Programa que mostre les taules de multiplicar del 2 al 9. Nota: este exercici és més difícil, ja que suposa incloure un bucle dins d'altre.

2.3.4 Instruccions break i continue

En qualsevol dels 3 tipus de bucles que hem vist, podem posar dins de les sentències les instruccions *break* i/o *continue*.

- *break* interromp l'execució del bucle i seguim per la instrucció posterior al bucle.
- *continue* fa que el programa comence altra iteració, encara que no s'haja acabat l'actual.

Nota: en cas de tindre estes instruccions dins de bucles niuats, només afecten al bucle on estan directament. Per exemple, si tenim un *break* en un bucle (*pare*) i este està dins d'un altre bucle (*iaio*), eixiríem del bucle *pare* però no del *iaio*.

Exercicis

17. Programa que simule un caixer automàtic. Es demana la clau contínuament mentre no siga correcta (1234). Però com a molt són 5 intents. Si s'encerta la clau, cal mostrar "CORRECTE" i acabar. Però si se sobrepassen els intents, cal mostrar "INCORRECTE" i acabar. (Utilitza una única condició per al bucle i ix del bucle amb un *break* quan calga).
18. Programa que demane 10 números (positius i/o negatius) i mostre la mitjana només dels positius. Cal utilitzar la sentència *continue*.