



Informàtica

**ICB0 Desenvolupament
d'aplicacions multiplataforma**

**ICC0 Desenvolupament
d'aplicacions web**

MPO2 Programació estructurada i modular

**AEA1 Aplica la programació estructurada en la
resolució de problemes en C/C++**

Diari d'activitats

Marc Brufau / Isidre Guixà

Curs 2025/26



Entorns a usar	15-09-2025
<ul style="list-style-type: none"> - CodeBlocks 25.03: Usa la <i>C runtime library</i> de Microsoft que conté funcions d'entrada/sortida segures. Això no és factible en versions anteriors de CodeBlocks. <p>Perill: Debugger no s'activa si el path absolut del projecte conté algun espai o caràcter no ascii!!!</p> <p>v25.03: Problema en instal·lació per Win64bits => No deixa ben instal·lada la ruta del depurador: La instal·lació per Win64 ha deixat la ruta com si fos Win32 i no troba el depurador. Ergo no depura. Per a què el depurador funcioni, fer les comprovacions següents:</p> <ol style="list-style-type: none"> 1) Start CodeBlocks 2) Select the Settings menu 3) Select the Debugger... option in the Settings menu 4) In the Debugger Settings window, select the Default category under GDB/CDB debugger 5) You should see a field named "Executable path:" in the Debugger settings window. Click on the "..." button to the right of this field. 6) In the "Select executable file" window that appears, select the following file and then click on the Open button: <ul style="list-style-type: none"> * En un Windows de 32 bits: C:\Program Files (x86)\CodeBlocks\MinGW\bin\gdb32.exe * En un Windows de 64 bits: C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe 7) Click on the OK button in the Debugger Settings window <ul style="list-style-type: none"> - Visual Studio 	
Sintaxi en C/C++ de conceptes bàsics	16-09-2025
<ul style="list-style-type: none"> - Estructura d'un programa en C/C++: https://www.it.uc3m.es/pbasanta/asng/course_notes/program_structure_es.html Hello World! - Diferència respecte Java: codi font / codi objecte / codi executable - Comentaris al codi <p>Veure projectes:</p> <ul style="list-style-type: none"> - 250915_1_CB_HelloWorldEnC - 250916_1_CB_HelloWorldEnCPP - 250916_2_VS_HelloWorldEnC - 250916_3_VS_HelloWorldEnCPP <p>Els projectes _CB_ són en CodeBlocks, mentre que els _VS_ són en Visual Studio En Visual Studio els fonts sempre són .cpp, encara que el contingut sigui només de C.</p>	
Sintaxi en C/C++ de conceptes bàsics	18-09-2025
<ul style="list-style-type: none"> - Tipus de dades simples https://www.it.uc3m.es/pbasanta/asng/course_notes/data_types_es.html Compte pels valors numèrics! Grandària (i per tant capacitat) segons l'arquitectura (16bits / 32bits / 64bits) - Variables - En C, sortida/escriptura formatada amb la funció <code>printf</code>. Veure documentació aquí i observar els diferents especificadors de format que cal usar en funció del tipus de variable. - Les dades numèriques <code>short-int-long-long long</code> admeten valors positius i negatius. Consulteu aquí els valors màxim i mínim. Per exemple, els valors <code>short</code> van de -32767 a 32767 (pot canviar segons plataforma). El tipus <code>char</code> que emmagatzema caràcters, també es pot usar per gestionar valors numèrics entre -127 i 127. - La funció <code>sizeof(variable/tipus)</code> informa de la llargada en bytes de la variable o del tipus. En el compilador de C de CB retorna un <code>long long unsigned i</code>, per evitar warnings en compilació, cal usar <code>%llu</code> per visualitzar el contingut en la funció <code>printf</code>. Però en VS retorna un <code>unsigned int i</code> es podria usar <code>%u</code>. - unsigned??? També existeixen els 5 tipus de dades <code>char-short-int-long-long long</code>, amb especificador <code>unsigned</code> pel davant (<code>unsigned char,...</code>) i en aquest cas només admeten valors majors o iguals a 0. En aquests casos, en la funció <code>printf</code> cal usar <code>%u</code>. Observar-ho en els projectes 1-3 següents per visualitzar el valor que retorna la funció <code>sizeof</code>. 	

Veure projectes:

- 250918_1_CB_TipusDadesBasicsEnC
- 250918_2_CB_TipusDadesBasicsEnCPP
- 250918_3_VS_TipusDadesBasicsEnC
- 250918_4_VS_TipusDadesBasicsEnCPP

Sintaxi en C/C++ de conceptes bàsics

22-09-2025

- Estructures condicionals & iteratives => Igual que en Java
- Instruccions d'entrada: `scanf` (<https://cplusplus.com/reference/cstdio/scanf>)
- Compte amb la "porqueria" que pot quedar en el buffer del teclat.
Per netejar-ho... hi ha compiladors/líbreries de C en les que `fflush(stdin)` buida el buffer del teclat.
En CodeBlocks 17 i 20 funciona!
Però això no és "oficial" i no funciona en Visual Studio 15+ ni en CodeBlocks 25.
La solució és usar el codi de la funció `netejar_buffer_teclat()` dels projectes adjunts.

Veure projectes:

- 250922_1_CB_ExemplesEntradaTeclat
- 250922_2_VS_ExemplesEntradaTeclat

La funció `scanf` en VS es considera perillosa i no compila generant error *This function or variable may be unsafe. Consider using `scanf_s` instead. To disable deprecation, use `_CRT_SECURE_NO_WARNINGS`.*

Per desactivar aquest avís: "Project -> Properties -> C/C++ -> Preprocessor -> Preprocessor Definitions" i afegir-hi `_CRT_SECURE_NO_WARNINGS`

Sintaxi en C/C++ de conceptes bàsics

23-09-2025

- Cadenes en C/C++: No confondre amb objectes de la classe `string` existent a C++.
Són taules de caràcters, amb caràcter `'\0'` al final
Hi ha diverses maneres de crear cadenes.

Veure projectes:

- 250923_1_CB_ExemplesCadenes
- 250923_2_VS_ExemplesCadenes

Funcions per manipular cadenes (<https://cplusplus.com/reference/cstring>)

- o `strlen`
- o `strcmp`

Sintaxi en C/C++ de conceptes bàsics

25-09-2025

29-09-2025

- Més sobre cadenes!

A més de les formes de crear cadenes vistes en anterior dia, també es poden definir com:

```
char cad3[] = "Pepe Gotera";
```

i aquesta cadena no és constant i podem canviar el seu contingut.

Veure projectes:

- 250925_1_CB_ExemplesCadenes
- 250925_2_VS_ExemplesCadenes

- Noves funcions per manipular cadenes (<https://cplusplus.com/reference/cstring>)

- o `strcpy`
- o `strncpy`

- Utilització d'`scanf` per omplir una cadena per teclat. És un problema, per què si volem controlar la màxima longitud de caràcters a introduir, per exemple 5, necessitem posar:

```
scanf("%5s", cad);
```

i aquest valor 5 és fix en el codi... i no es pot anar canviant en temps d'execució en funció de la capacitat de la cadena que es vulgui emplenar... i en el seu lloc no es pot posar `sizeof(cad)`, que és el què interessaria.

Veure projectes

- 250925_3_CB_ExemplesCadenes
- 250925_4_VS_ExemplesCadenes



- Utilització de `fgets` per omplir una cadena per teclat. Permet indicar la capacitat com a paràmetre!

Veure projectes

- 250925_5_CB_ExemplesCadenes
- 250925_6_VS_ExemplesCadenes

- Utilització de funcions segures, com `scanf_s` per omplir una cadena per teclat.

Veure projectes

- 250929_1_CB_EntradaCadenes
- 250929_2_VS_EntradaCadenes

La funció `scanf_s` (i les seves cosines `gets_s`, `fopen_s`, etc.) no forma part de l'estàndard C original (C89/C99/C11). Va ser introduïda per Microsoft com a extensió pròpia dins del seu compilador MSVC, amb l'objectiu de reduir vulnerabilitats habituals (com els *buffer overflows*) en programes C escrits a Windows. Es pot usar en entorns que usin el compilador MSVC o que enllacen amb la *C runtime library* de Microsoft.

*Estàndards de C i funcions *_s:*

- Amb l'arribada del C11, el comitè de normalització ISO va afegir l'Annex K ("Bounds-checking interfaces"), on apareixen funcions com `scanf_s`, `strcpy_s`, `strncpy_s`, etc.
- Peeeerò... aquest annex és opcional: els compiladors no estan obligats a implementar-lo.
- La majoria de compiladors a Linux/Unix (com GCC i Clang) no implementen Annex K, en part perquè:
 - L'annex es considera controvertit i criticat per la comunitat: les interfícies *_s són verboses, inconsistentes i no resolen tots els problemes de seguretat.
 - Hi ha ja alternatives segures i portables (`fgets`, `snprintf`, `strncat`, etc.).
 - Es vol evitar "contaminar" el llenguatge amb extensions fortament lligades a Microsoft.

Exercici

30-09-2025

Programa que demani una data en format `dd-mm-yyyy` per teclat, informant d'entrada incorrecta o, si és correcta, mostrant el valor introduït pel dia, mes i any.

El dia com a molt ha de tenir 2 dígits, però podria ser només un.

El mes com a molt ha de tenir 2 dígits, però podria ser només un.

L'any com a molt ha de tenir 4 dígits, però podria ser 1 o 2 o 3.

Veure projectes:

- 250930_1_CB_ExerciciEntradaData
- 250930_2_VS_ExerciciEntradaData

Atenció!

- Hem creat un projecte `UtilitatsMila` on anirem incorporant les utilitats i les usarem en diversos programes. Així, incorporem les funcions que hem anat desenvolupant fins ara:

```
void netejar_buffer_teclat();
int llegirTextAmbSeguretat(char *cadena, int mida); // Usant fgets
int llegirTextAmbSeguretatMSVC(char *cadena, int mida); // Usant scanf_s
```

En `utils.h` hi ha els prototipus de les funcions i en `utils.c` hi ha el codi de les funcions.

Observacions sobre els fitxers de capçalera (header: `.h`, `.hpp`,...):

- ✓ En un projecte amb varis `.c/.cpp`, molts d'aquests fitxers amb codi poden necessitar de definicions existents en fitxers de capçalera i, per tant, en diversos `.c/.cpp` podem haver d'incloure els corresponents `#include`, fet que provocaria error de compilació en trobar varies vegades les mateixes definicions.

Per evitar aquest error, en els fitxers de capçalera s'incorpora les següents directives de precompilació:

```
# ifndef CONSTANT_SIMBOLICA // Única per aquest fitxer - S'usa nom del fitxer
# define CONSTANT_SIMBOLICA

// Aquí va el contingut del fitxer de capçalera

# endif
```

- ✓ Les funcions C són compatibles (usables) en C++? En principi NO, a no ser que les "marquem" per a que



puguin ser usades en C++. Hi ha diverses formes de marcatge, que podeu veure en [aquest vídeo](#).

En el nostre cas, com que volem que les funcions definides en `utils.h` puguin ser usades en programes en C++, usarem el primer mecanisme indicat en el vídeo, consistent en posar totes les definicions dins un bloc extern `"C" { ... }`, però això només si estem en cas de codi C++, fet que es pot saber consultant l'existència de la constant simbòlica `__cplusplus`.

- El projecte `250930_1_CB...` conté el `main` i fa servir funcions de les utilitats. Per aconseguir-ho, cal:
 - En el projecte, afegir els fitxers `utils.c` i `utils.h` de l'altre projecte. De fet, els fitxers de capçalera `.h` no és obligatori incorporar-los en el projecte, però és aconsellable per poder-los consultar.
 - En el projecte, anar a *Build Options* i a l'apartat *Search directories* incorporar les rutes de les carpetes que continguin fitxers de capçalera. En el nostre cas, `..\UtilitatsMila` doncs tenim tots els projectes en mateixa ruta.
- **I a partir d'ara, en els projectes per VS no duplicarem codi**, sinó que usarem el codi dels projectes CB. Així, el projecte `250930_2_VS...` s'ha creat buit i, posteriorment:
 - En *Archivos de origen* hem incorporat els fitxers `.c` (`main.c` de `250930_1_CB` i `utils.c` de `UtilitatsMila`)
 - En propietats del projecte, apartat *C/C++ | General | Directorios de inclusion adicionales* incorporem les rutes de les carpetes que continguin fitxers de capçalera.

En els projectes que acompanyen aquest document, s'usarà rutes relatives i no absolutes, suposant que els projectes resideixen en l'estructura de carpetes:

MPO2-AEA1 amb els projectes de l'AEA1 (i aquí és on resideix `UtilitatsMila`)

MPO2-AEA2 amb els projectes de l'AEA2

MPO2-AEA3 amb els projectes de l'AEA3

Així, en un projecte dins MPO2-AEA1, s'haurà incorporat `..\UtilitatsMila`

mentre que en un projecte dins MPO2-AEA2, s'haurà incorporat `..\..\MPO2-AEA1\UtilitatsMila`