

Introduzione al C

Elementi di Informatica

Un po' di storia...

Nel **1972** Dennis Ritchie sviluppa la prima versione del linguaggio C

Da allora la più significativa estensione è relativa all'introduzione della programmazione orientata agli oggetti (OOP): **C++**

E' un linguaggio di **alto livello** ma con funzionalità più **tipiche del linguaggio macchina**

Unix/Linux

Un po' di storia...

1977: presentazione del linguaggio C

1989: pubblicazione dello standard C - C89

1999: pubblicazione dello standard C - C99

2011: pubblicazione dello standard C - C11

IDE – Integrated Development Enviroment

Linux: *gcc*

Mac OS: dipendenza da *XCode*

```
gcc -o nome_programma mio_codice.c
```

IDE – Integrated Development Enviroment

Windows (multiplatforma)

- DEV-C++ (obsoleto)
- Eclipse
- CodeLite
- Visual Studio (Code)
- **Atom**
- **CLion** (licenza con credenziali di Unica)
- **Qt Creator**



A cross-platform IDE for C and C++

GET FREE 30 DAY TRIAL



TAKE A TOUR

CLion 2018.2 is here. Check out [what's new](#)

Which compilers are supported by the IDE?

CLion supports GCC, Clang and Microsoft Visual C++ compilers. This means that on Windows you can select between [MinGW](#) (or [MinGW-w64](#)), [Cygwin](#) and Microsoft Visual Studio tool sets.

What do I need to start with CLion? What are the system requirements?

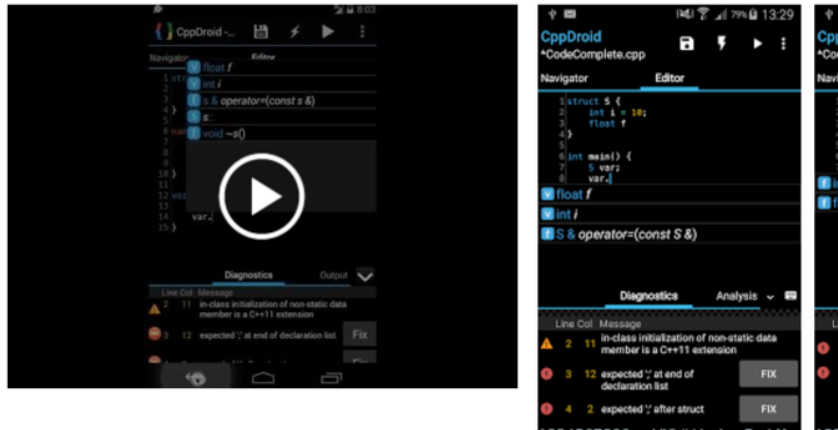
In general to develop in C/C++ with CLion you need:

- GCC/G++ or Clang, which in case of Windows means using toolchains: MinGW (or MinGW-w64), Cygwin 2.8 (minimum required), or Visual Studio if you are going to use Microsoft Visual C++ compiler instead of GCC/C++ or Clang (refer to our [tutorial](#)).

CLion includes bundled GDB (for [MinGW](#) on Windows), recent version of LLDB (on Linux and macOS), JDK 1.8 and CMake so you don't need to install them separately. Check the bundled CMake version number in **File | Settings | Build, Execution, Deployment | Toolchains** (or **CLion | Preferences | Build, Execution, Deployment | Toolchains** if you are macOS user).

You can install any of that packages on your system, including custom versions of CMake, compilers and GDB.

Per smartphone e tablet



Domande?

www.menti.com

Struttura di un programma in C

```
//eventuali definizioni  
  
int main()  
{  
  
    //Dichiarazioni  
  
    //Istruzioni  
  
    return 0;  
}
```

Il mio primo programma

```
//Il mio primo programma in C

#include <stdio.h>

int main()
{
    printf("Hello world");
    return 0;
}
```

Il mio primo programma

```
/*Il mio primo programma in C*/  
  
#include <stdio.h>  
  
int main()  
{  
    printf("Hello world\n");  
    return 0;  
}
```

Passo passo...

```
/*Il mio primo programma in C*/
```

```
//Il mio primo programma in C
```

rappresentano dei **commenti**

Passo passo...

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

- E' una direttiva al preprocessore.
- Le linee che iniziano con `#` vengono elaborate prima della compilazione del programma.
- Nel programma verrà incluso il contenuto del file `stdio.h`.
- `stdio.h` (standard input-output header) è un file di intestazione che definisce gli stream (flussi) di input e output.

Passo passo...

```
int main()
```

- Il `main` è la funzione principale.
- Tutte le funzioni vengono richiamate usando il loro **nome** e tra `()` vengono indicati i parametri.
- `int` indica che la funzione restituisce un intero e le parentesi `{ }` definiscono il **corpo** della funzione

Passo passo...

```
printf("Hello world\n");
```

- La `printf` è funzione che scrive sullo standard output (monitor).
- Tra `"` si può inserire del testo.
- `\n` indica l'inserimento di una nuova linea.
- ogni istruzione deve terminare con un `;`.

Passo passo...

```
return 0;
```

- indica che il programma è terminato con successo.