



# Gli Argomenti al Main

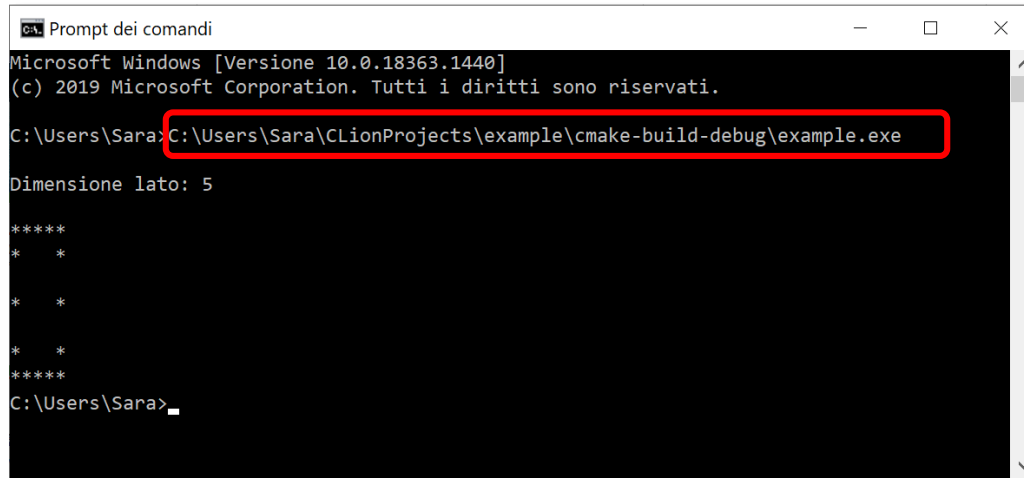
---

COME PASSARE AL MAIN DATI IN  
ESECUZIONE



# Interfaccia utente “a comandi” (testuali)

- Il paradigma standard di invocazione (esecuzione) di un programma è ormai il “doppio click”
- Esiste tuttavia una modalità alternativa che prevede
  - Un’interfaccia testuale (**LINEA DI COMANDO**)
  - L’invocazione di un programma come un comando



The screenshot shows a Windows Command Prompt window titled "C:\> Prompt dei comandi". The window displays the following text:

```
Microsoft Windows [Versione 10.0.18363.1440]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Sara>C:\Users\Sara\CLionProjects\example\cmake-build-debug\example.exe

Dimensione lato: 5

*****
*   *
*   *
*   *
*****
C:\Users\Sara>_
```

The command `C:\Users\Sara\CLionProjects\example\cmake-build-debug\example.exe` is highlighted with a red rectangle.

*Ci sono  
differenze  
sintattiche tra  
i vari sistemi  
operativi!*

# Argomenti sulla linea di comando

- In C, è possibile passare informazioni ad un programma specificando degli argomenti sulla **linea di comando**

- Esempio:

```
C:\> myprog <arg1> <arg2> ... <argN>
```

- Comuni in molti comandi “interattivi”

- Esempio: Windows (*prompt dei comandi*)

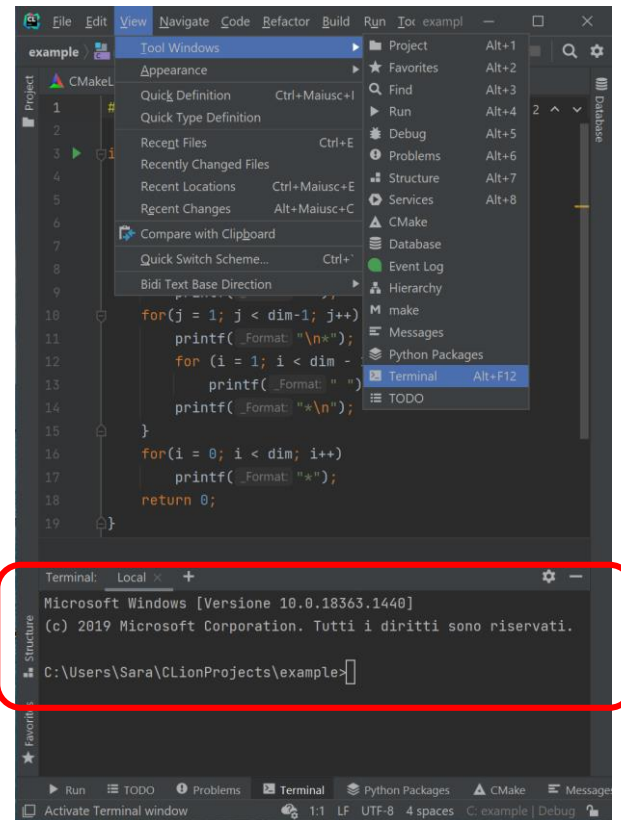
```
C:\home> copy file1.txt dest.txt
```

- Esempio: Os-x (*console*)

```
$ cp file1.txt dest.txt
```

# Come attivare il prompt da CLion

- Si attiva con View -> Tool Windows -> Terminal
- Vedi istruzioni disponibili su
  - <https://www.jetbrains.com/help/clion/terminal-emulator.html>
- Dalla finestra "terminal" è possibile lanciare l'eseguibile mediante comando, specificando eventuali argomenti
  - *Attenzione a verificare la cartella in cui ci si trova (e quella in cui si trova l'eseguibile)*



# Argomenti del main()

- Gli argomenti sulla linea di comando vengono automaticamente visti, all'interno del programma, come argomenti (parametri) della funzione `main()`
- **`int main (int argc, char *argv[])`**
  - **`argc`: Numero di argomenti specificati**
    - Esiste sempre almeno un argomento implicito (nome del programma)
  - **`argv`: Vettore di stringhe**
    - `argv[0]` = primo argomento (il nome del programma)
    - `argv[i]` = generico argomento
    - `argv[argc-1]` = ultimo argomento

# Esempi

```
C:\progr>quadrato
```

Numero argomenti = 1 (argc=1)

- Argomento 0 = "quadrato.exe"

```
C:\progr>quadrato 5
```

Numero argomenti = 2 (argc=2)

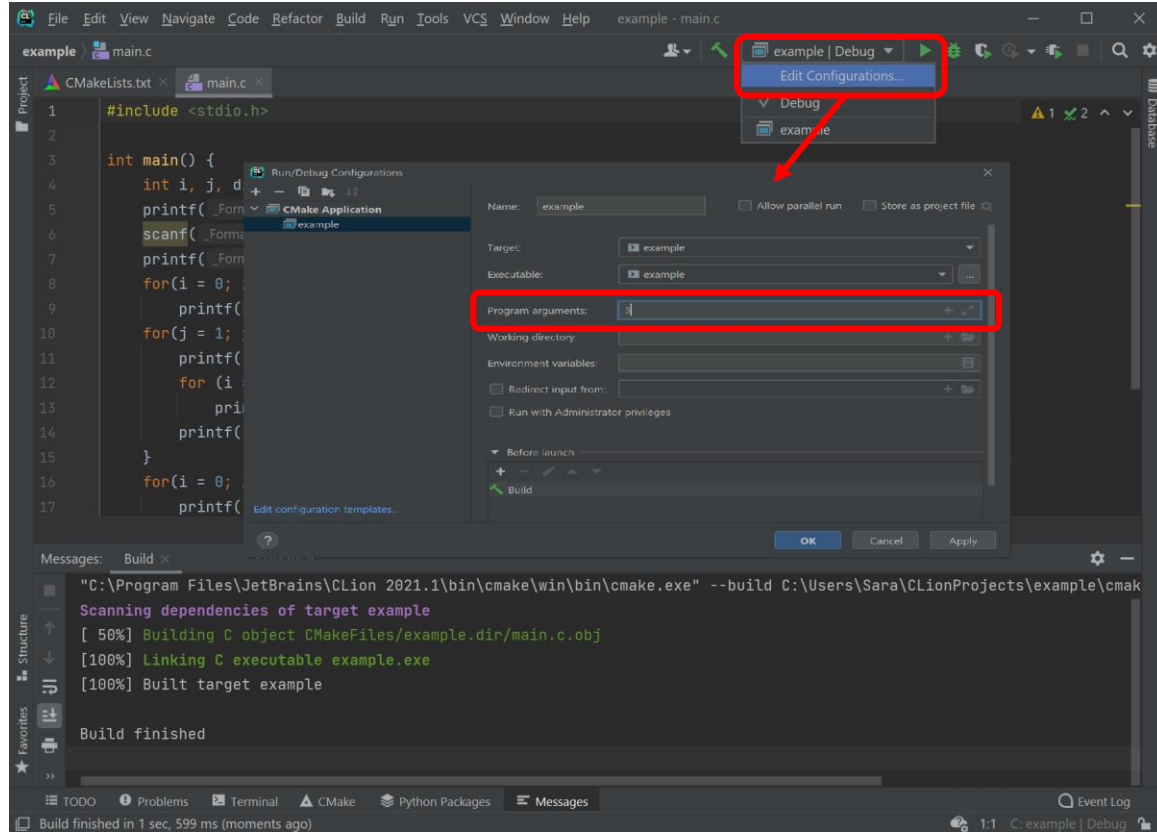
- Argomento 0 = "quadrato.exe"
- Argomento 1 = "5"

```
C:\progr>quadrato 5 K
```

Numero argomenti = 3 (argc=3)

- Argomento 0 = "quadrato.exe"
- Argomento 1 = "5"
- Argomento 2 = "K"

# Esempi



# Argomenti al main

- Due parametri al main:
  - argv (vettore di stringhe)
    - argv[0] sempre presente, rappresenta il nome del file eseguibile (del programma)
  - argc: dimensione del vettore di stringhe (quanti sono gli argomenti)
- Esempio:

```
int main (int argc, char *argv[]) {  
    FILE *fp1, *fp2;  
    if (argc<3) {  
        printf("ERRORE: mancano argomenti al main\n");  
        return 1;  
    }  
    fp1 = fopen(argv[1],"r");  
    fp2 = fopen(argv[1],"r");  
    ...  
}
```



# Come usare `argc` e `argv`

- Ciclo per l'elaborazione degli argomenti

```
for (i=0; i<argc; i++) {  
    /* elabora argv[i] come stringa */  
}
```

- NOTA:
  - Qualunque sia la natura (testo o numero) dell'argomento, è sempre una stringa
  - Necessario quindi uno strumento per “convertire” in modo efficiente stringhe in tipi numerici

# Conversione degli argomenti

- **Gli elementi di argv sono stringhe!**

- Indipendentemente da cosa contengono
- Il C mette a disposizione due funzioni (definite in `<stdlib.h>`) per la conversione di una stringa in valori numerici
  - `int atoi(char s[]);`
  - `double atof(char s[]);`

- **NOTA: `atoi/atof` assumono che la stringa contenga un valore scritto correttamente**

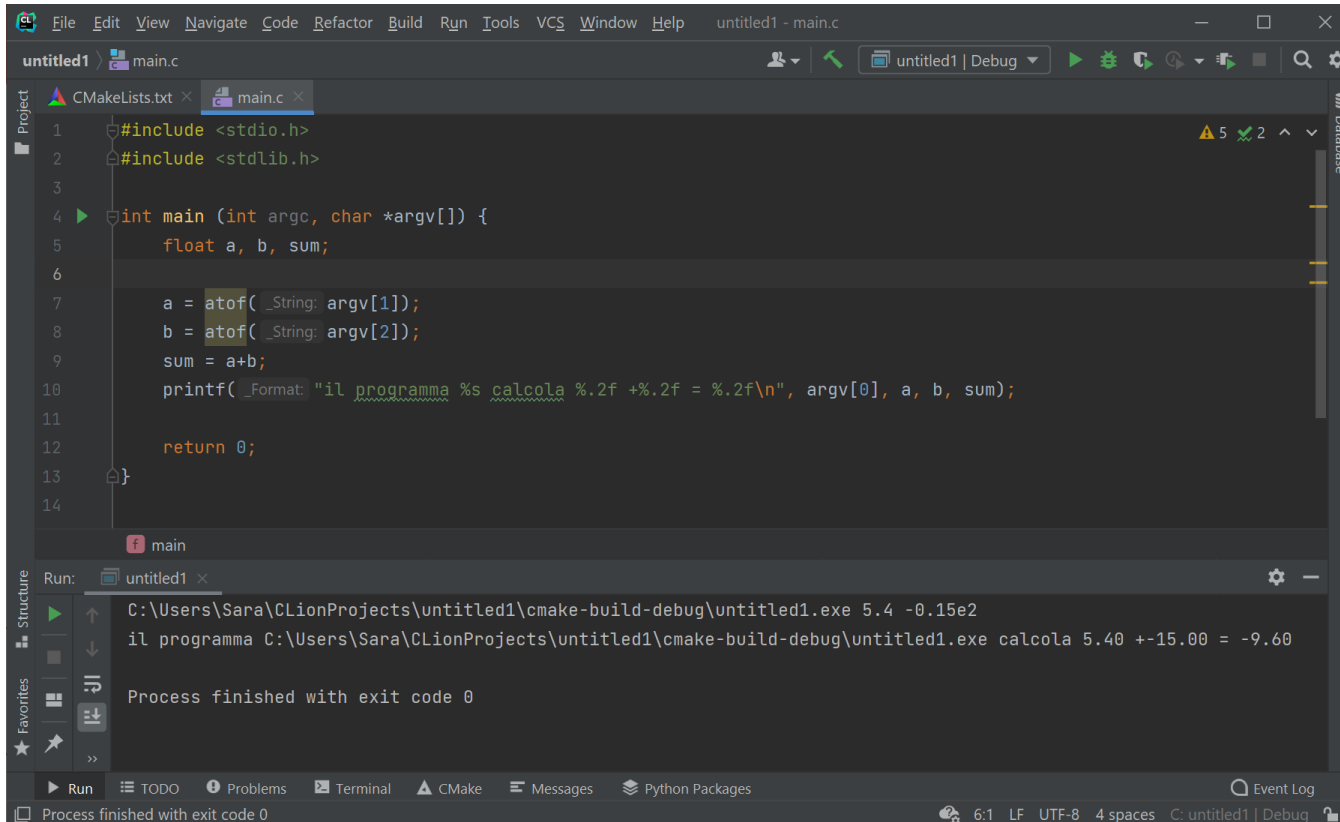
- Le funzioni restituiscono il valore 0 in caso di errore
- Si consiglia di controllare il risultato della conversione!

# Esempi di atoi/atof

```
// Esempi applicati a stringhe costanti (servono solo per capire)
int x = atoi("2");           // x=2
double z = atof("2.35e-2");  // z=0.0235

// Esempio applicato agli argomenti al main
// si suppone che il programma sia chiamato come ad esempio:
// somma 5.4 -0.15e2
int main (int argc, char *argv[]) {
    float a, b, sum;
    a = atof(argv[1]);
    b = atof(argv[2]);
    sum = a+b;
    printf("il programma %s calcola %f +%f = %f\n", argv[0], a, b, sum);
}
```

# Esempi di atoi/atof




The screenshot shows a C code editor with a file named `main.c`. The code defines a `main` function that takes command-line arguments and uses `atof` to convert them to floats. The program calculates the sum of two floats and prints the result.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main (int argc, char *argv[]) {
5     float a, b, sum;
6
7     a = atof(_String: argv[1]);
8     b = atof(_String: argv[2]);
9     sum = a+b;
10    printf(_Format: "il programma %s calcola %.2f +%.2f = %.2f\n", argv[0], a, b, sum);
11
12    return 0;
13 }
```

The output window shows the execution of the program with the following command line and output:

```
Run: C:\Users\Sara\CLionProjects\untitled1\cmake-build-debug\untitled1.exe 5.4 -0.15e2
      il programma C:\Users\Sara\CLionProjects\untitled1\cmake-build-debug\untitled1.exe calcola 5.40 +-15.00 = -9.60
      Process finished with exit code 0
```

# Quali argomenti?

- Cosa passiamo in genere come argomento?
- In teoria possiamo passare qualsiasi cosa
- In pratica, valori tipici sono:
  - **Nomi di file**
    - Es: il nome del file da leggere/scrivere è specificato come argomento
  - **Opzioni del programma**, che può operare in diverse “modalità”
    - Queste “opzioni” (dette flag o switch) sono convenzionalmente specificate (per distinguerle dagli altri argomenti) come -<carattere>
    - Esempio
    - `C:\> myprog -x -u file.txt`  

      - opzioni*
      - argomento*

# Esercizio 1

- Scrivere un programma che legga sulla linea di comando due interi  $N$  e  $D$ , e stampi tutti i numeri minori o uguali ad  $N$  divisibili per  $D$

# Esercizio 1: Soluzione

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    int N, D, i;
    if (argc != 3) {
        printf("Numero argomenti non valido\n");
        return 1;
    }
    N = atoi(argv[1]);
    D = atoi(argv[2]);

    for (i=1; i<N; i++) {
        if (i%D == 0) {
            printf("%d\n",i);
        }
    }
    return 0;
}
```

# Esercizio 2

- Scrivere un programma `m2m` che legga da un file un testo e lo riscriva su un secondo file, eventualmente convertendo tutte le lettere maiuscole in minuscole o viceversa, a seconda dei flag specificati sulla linea di comando
  - `-l`, `-L` → conversione in minuscolo
  - `-u`, `-U` → conversione in maiuscolo
  - Un ulteriore flag `-h` permette di stampare un help
- Utilizzo:
  - `m2m -l a.txt a_minuscolo.txt`
  - `m2m -L a.txt a_minuscolo.txt`
  - `m2m -u b.txt b_maiuscolo.txt`
  - `m2m -U b.txt b_maiuscolo.txt`
  - `m2m c.txt uguale_a_c.txt`
  - `m2m -h`



## Esercizio 2: Soluzione

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    int i, lowercase=0, uppercase=0;
    for (i=1; i<argc-2; i++) {
        switch (argv[i][1]) {
            case 'l': case 'L':
                lowercase = 1;
                break;
            case 'u': case 'U':
                uppercase = 1;
                break;
            case 'h':
                printf("Uso: m2m [-l uh] <nomefile>\n");
        }
    }
    converti(argv[argc-2], argv[argc-1], lowercase, uppercase);
    // svolge il lavoro vero e proprio
}
```