



# Gli Argomenti al Main

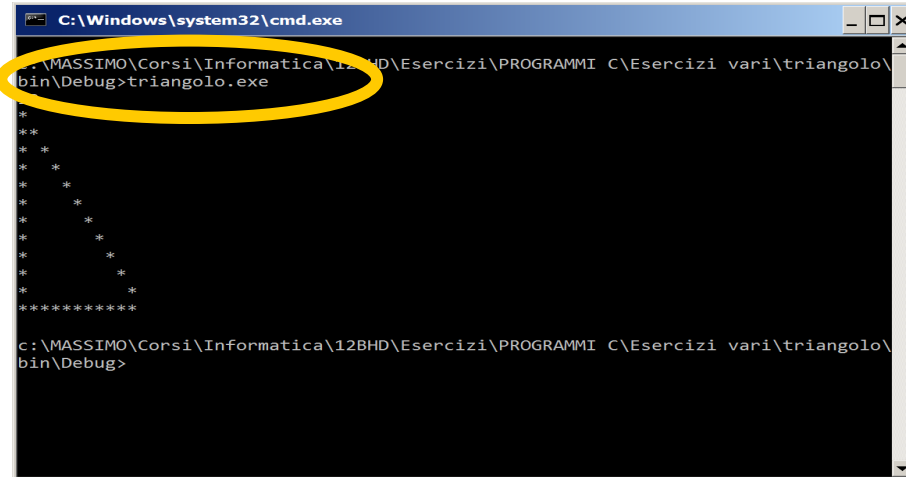
---

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 image shows a screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The command prompt shows the current directory as "c:\MASSIMO\Corsi\Informatica\12BHD\Esercizi\PROGRAMMI C\Esercizi vari\triangolo\" and the command "bin\Debug>triangolo.exe" being entered. The command is highlighted with a yellow oval. Below the command, the output of the program is displayed, showing a pattern of asterisks forming a right-angled triangle. The pattern consists of 10 rows of asterisks, with the number of asterisks in each row increasing from 1 to 10. The output is preceded by a line of asterisks and followed by a line of asterisks.

```
C:\Windows\system32\cmd.exe
c:\MASSIMO\Corsi\Informatica\12BHD\Esercizi\PROGRAMMI C\Esercizi vari\triangolo\
bin\Debug>triangolo.exe
*****
*
*  *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
*****
c:\MASSIMO\Corsi\Informatica\12BHD\Esercizi\PROGRAMMI C\Esercizi vari\triangolo\
bin\Debug>
```

# 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)*

# Gli argomenti in Clion **SENZA** usare Terminal

- In alternativa all'utilizzo di terminal (che non consente ad esempio il debug)
  - E' possibile definire gli argomenti al main anche in modalità di esecuzione IDE (con o senza debug)
  - Menu "Run -> Edit Configurations"
  - Impostare gli argomenti nella riga "Program arguments:"
- Vedi istruzioni disponibili su
  - <https://www.jetbrains.com/help/clion/run-debug-configuration.html#envvars-progargs>
  - sezione "Add program arguments"

# 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)

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

Numero argomenti = 2 (argc=2)  
Argomento 1 = "5"

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

Numero argomenti = 2 (argc=3)  
Argomento 1 = "5"  
Argomento 2 = "K"

# 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[2],"w");  
    ...  
}
```



# 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 = atoi(argv[1]);
    b = atoi(argv[2]);
    sum = a+b;
    printf("il programma %s calcola %f +%f = %f\n", argv[0], a, b, sum);
}
```

# 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 e' 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>
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);
        }
    }
}
```

# 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>

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 [-luh] <nomefile in> <nomefile out>\n");
            }
        }
    converti(argv[argc-2], argv[argc-1], lowercase, uppercase); // svolge il lavoro vero e proprio
}
```