

IL MAIN in C: RIFLESSIONE

In C, il *main* è una funzione come le altre, salvo che:

- ha un nome convenzionale, fissato
- è la funzione invocata per far partire il programma.

Ma... CHI LA INVOCA?

*Visto che è una funzione,
HA DEI PARAMETRI?*

ARGOMENTI DALLA LINEA DI COMANDO

Il *main* è una funzione

- invocata dal sistema operativo
- che riceve un array di stringhe
- che corrispondono agli *argomenti scritti dall'utente sulla linea di comando*

Esempio di invocazione da linea di comando:

C:> prog pippo 12 paperino 23

nome del
programma

I° argomento

II° argomento

III° argomento

IV° argomento

ARGOMENTI DALLA LINEA DI COMANDO

Perciò, *main* ha *due parametri*:

- un intero che rappresenta la lunghezza dell'array
 - *int argc* (*argument counter*)
- l'array di stringhe vero e proprio (ovviamente, si passa il suo indirizzo iniziale)
 - *char* argv[]* (*argument vector*)

Ogni elemento dell'array è un puntatore a carattere, che punta a uno degli argomenti della linea di comando.

ARGOMENTI DALLA LINEA DI COMANDO

Quindi, *l'interfaccia completa del main* è la seguente:

```
int main(int argc, char* argv[])
```

Se non servono, *argc* e *argv* possono essere *omessi*, nel qual caso il main assume la forma semplificata già nota:

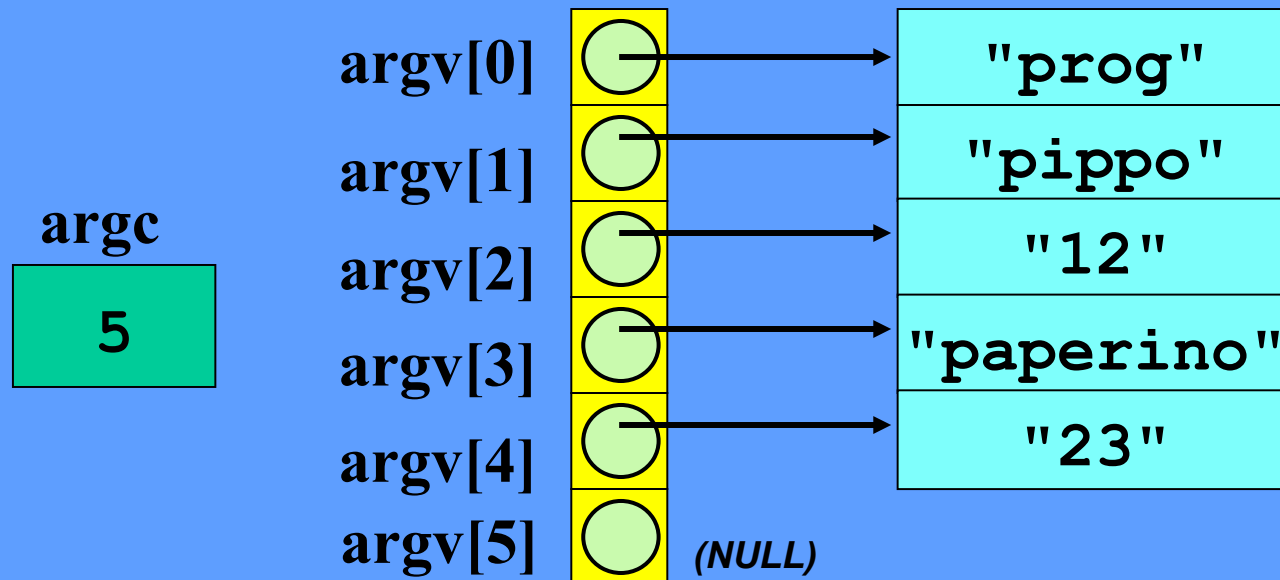
```
int main()
```

Valore di ritorno: può essere usato per restituire al Sistema Operativo un codice numerico di errore.

Convenzione: 0 = OK, ogni altro valore = un tipo di errore

ARGOMENTI DALLA LINEA DI COMANDO

- `argv[0]` è il nome del programma stesso
- da `argv[1]` ad `argv[argc-1]` vi sono gli argomenti passati, nell'ordine
- `argv[argc]` è per convenzione *NULL*

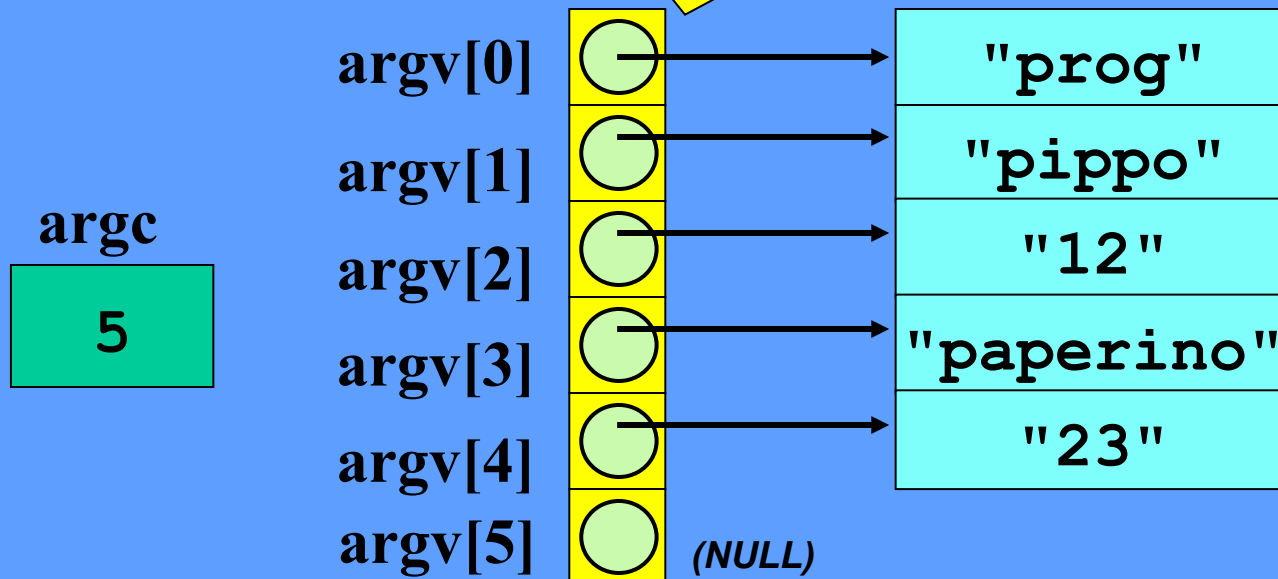


ARGOMENTI DALLA LINEA DI COMANDO

- argv[0] è il nome del programma stesso

Attenzione: sono aree del sistema operativo → disponibili solo in lettura

- argv[argc] è per convenzione NULL



ARGOMENTI DALLA LINEA DI COMANDO

Problema:

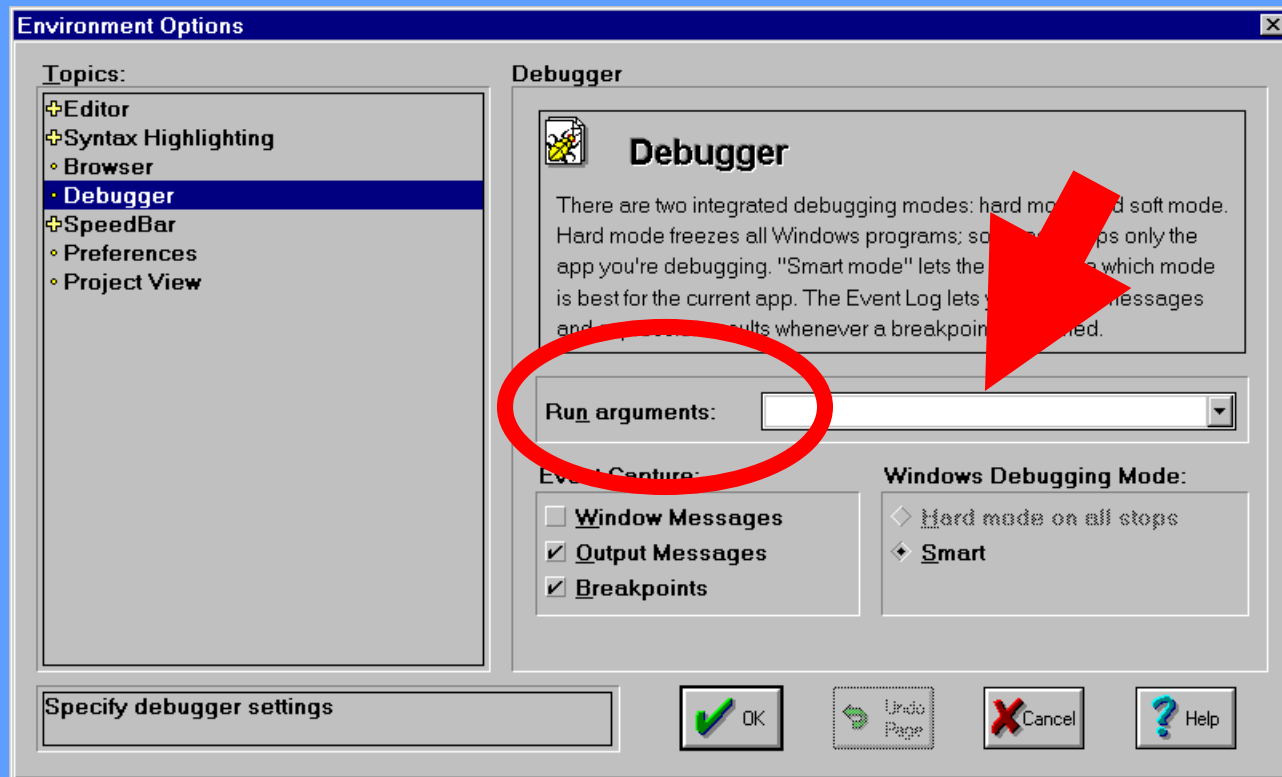
Come passare argomenti dalla linea di comando... quando non c'è una linea di comando, come negli ambienti di sviluppo integrati (Turbo C, RHide, etc) ?

```
C:> prog pippo 12 paperino 23
```

Esiste un'apposita opzione da menù.

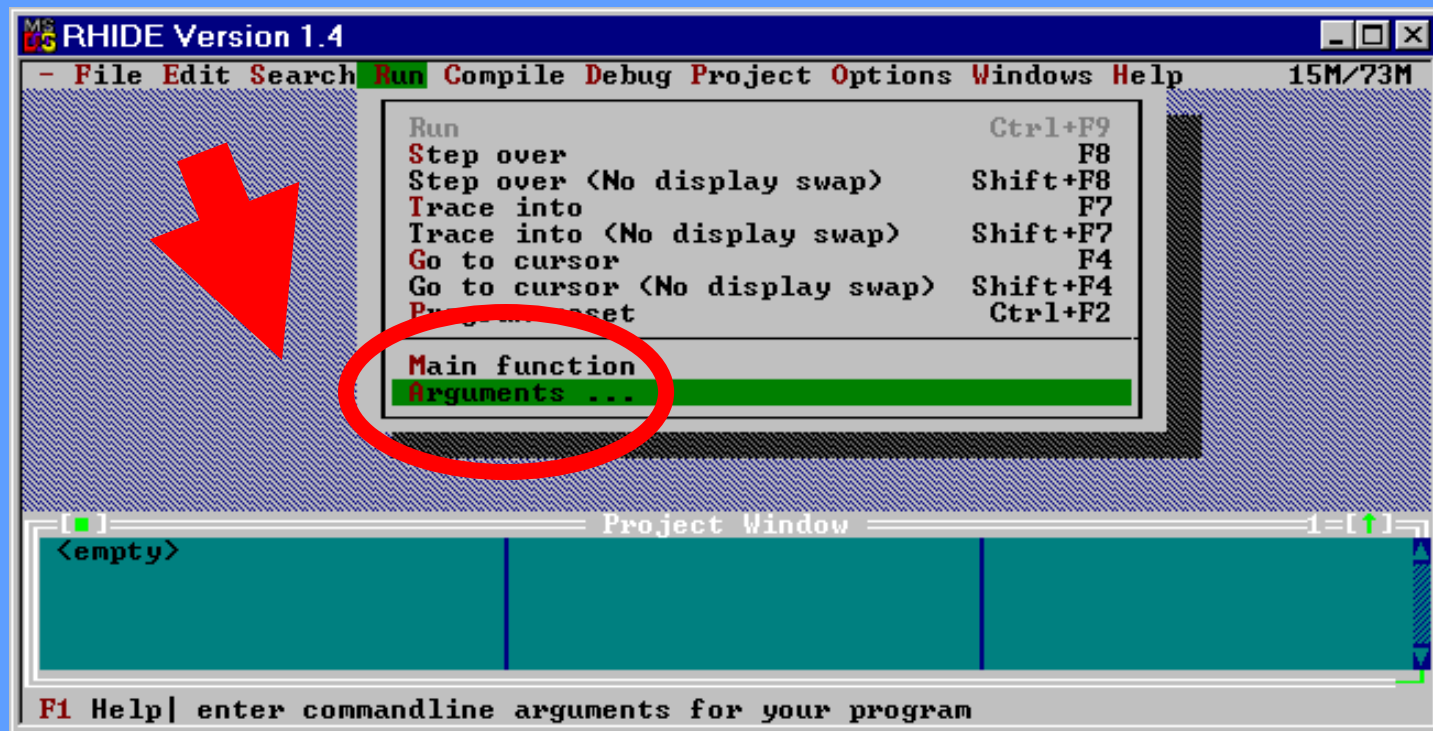
ARGOMENTI DALLA LINEA DI COMANDO

In Turbo C: Options / Environment / Debugger



ARGOMENTI DALLA LINEA DI COMANDO

In RHide: Run / Arguments...



ESEMPIO 1

Problema:

Scrivere un programma che analizzi gli argomenti passati dalla linea di comando e **restituisca come risultato del main** il numero di *lettere minuscole*.

Specifica:

Per ogni argomento da 1 ad argc-1, occorre:

- recuperare l'argomento (una stringa)
- contare le minuscole presenti in tale stringa
- sommare questo valore alla variabile che rappresenta il numero totale di minuscole.

Alla fine si restituisce il valore di tale variabile.

ESEMPIO 1

Codifica del main

```
int contaMinuscole(char s[]);  
  
int main(int argc, char* argv[]) {  
    int sum=0, i;  
    for(i=1; i<argc; i++)  
        sum +=  
        contaMinuscole(argv[i]);  
    return sum;  
}
```

ESEMPIO 1

Codifica della funzione contaMinuscole():

```
#include <ctype.h>  /* islower() */
int contaMinuscole(char *s){
    int n=0,i=0;
    while (s[i]!='\0') if (islower(s[i++])) n++;
    return n;
}
```

Codifica alternativa:

```
#include <ctype.h>  /* islower() */
int contaMinuscole(char *s){
    int n=0;
    while (*s) if (islower(*s++)) n++;
    return n;
}
```

ESEMPIO 1

Uso (ipotesi: il programma si chiama count)

C:> count Paperino Tom Archimede

Nome del
programma

I° argomento

II° argomento

III° argomento

***Alla fine,
Rhide
mostra il
valore
restituito***

```
MS RHIDE Version 1.4
- File Edit Search Run Compile Debug Project Options Windows Help 15M/65M
c:/temp/prova.c 2

int contaMinuscole(char s[]);
int main(int argc, char* argv[]) {
    int sum=0, i;
    for(i=1; i<argc; i++)
        sum += contaMinuscole(argv[i]);
    return sum;
}

#include <ctype.h>
int contaMinuscole(
    int n=0;
    while (*s) if (isl
    return n;

Compiling: prova.c
no errors
Creating: count.exe
no errors

Program exit code: 17 (0x0011)
OK

Message Window
3=1↑

F2 Save F3 Open F5 Zoom F6 Next Alt+F9 Compile F10 Menu Alt+X Quit
```

ESEMPIO 2

Problema:

Scrivere un programma che, dati un carattere e una stringa sulla linea di comando, conti quante volte il carattere compare nella stringa (sia in versione maiuscola che minuscola), e restituisca questo valore come risultato del main.

Specifica:

- occorre per prima cosa recuperare gli argomenti
- poi, occorre scandire la stringa carattere per carattere, contando le occorrenza del carattere dato (facendo attenzione alle maiuscole e minuscole)

Alla fine si restituisce il risultato di tale conteggio.

ESEMPIO 2

Codifica

```
#include <ctype.h>

int main(int argc, char* argv[]) {
    int cont=0;
    char ch = toupper(argv[1][0]);
    char *s = argv[2]; int i=0;
    while (s[i]!='\0')
        if (toupper(s[i++])==ch) cont++;
    return cont;
}
```

ESEMPIO 2

Codifica (stile "hacker")

```
#include <ctype.h>

int main(int argc, char* argv[]) {
    int cont=0;
    char ch = toupper(argv[1][0]);
    char *s = argv[2];
    while (*s)
        if (toupper(*s++)==ch) cont++;
    return cont;
}
```

ESEMPIO 3

Problema:

Scrivere un programma che sommi tutti i numeri passati come argomenti sulla linea di comando, e restituisca questo valore come risultato del main.

Specifica:

- occorre analizzare gli argomenti uno per uno
- per ciascun argomento (una stringa!) occorre *ricavare il numero corrispondente* in base dieci
- tale numero va sommato al totale

Alla fine si restituisce il totale così ottenuto.

ESEMPIO 3

Codifica

```
#include <stdlib.h>

int main(int argc, char* argv[]) {
    int sum=0, i=0;
    while (i<argc)
        sum += atoi(argv[i++]);
    return sum;
}
```

atoi: funzione di libreria che, data una stringa che contiene la rappresentazione di un numero in base dieci, restituisce il numero corrispondente. Ad esempio, **atoi("123")** restituisce l'intero *centoventitre*.

ESEMPIO 3

Codifica

```
#i
```

```
int
```

```
int sum=0;
```

```
while (i<argc)
```

```
    sum += atoi(argv[i++]);
```

```
return sum
```

```
}
```

atoi è usata implicitamente da **scanf** per leggere da tastiera (o da file di testo) la *rappresentazione di un numero*: infatti, **scanf ("%d", &x)** equivale a (st è una stringa):

```
scanf ("%s", st);  x = atoi(st);
```

atoi: funzione di libreria che, data una stringa che contiene la *rappresentazione di un numero in base dieci*, restituisce il numero corrispondente. Ad esempio, **atoi ("123")** restituisce l'intero *centoventitre*.

ESEMPIO 3

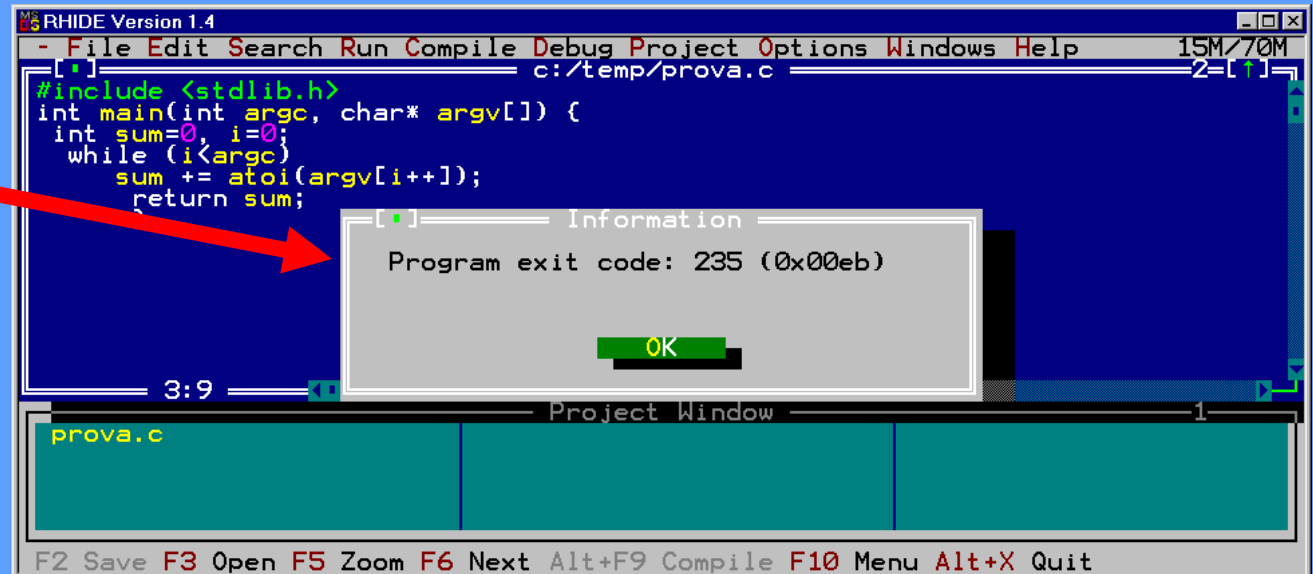
Uso (ipotesi: il programma si chiama sum)

C:> sum 123 45 67

Nome del
programma

Argomenti

Alla fine,
Rhide
mostra il
valore
restituito



ESEMPIO 3 - *RUN in TurboC*

The image shows a screenshot of the TurboC IDE interface. On the left, the 'Environment Options' dialog box is open, with the 'Debugger' tab selected. The 'Run arguments' field contains '123 45 67'. A yellow callout bubble with the word 'Argomenti' points to this field. The 'Event Capture' section has 'Output Messages' and 'Breakpoints' checked. The 'Windows Debugging Mode' section has 'Smart' selected. On the right, the 'C++ - prova1' window shows the following code:

```
#include <stdlib.h>
main(int argc, char* argv[]) {
    sum=0, i=0;
    while (i<argc)
        sum += atoi(argv[i++]);
    return sum;
}
```

The 'Watch' window is open, showing 'sum: 235'. A red arrow points from the text below to this window. At the bottom of the IDE, a status bar says 'Select one or more watch expressions'.

In Turbo C il valore restituito non viene mostrato: bisogna intercettarlo tramite watch expression

ARGOMENTI DALLA LINEA DI COMANDO: PERCHÉ ?

A cosa servono, in C, gli argomenti del **main**?

- Permettono di *scrivere in C "pezzi" del sistema*
- Molti comandi di sistema che si usano da console *non sono altro che programmi scritti* (probabilmente in C) *da qualcuno per noi*
 - in Windows: `comp`, `doskey`, `fc`, `find`, `finger`, `ftp`, `mem`, `net`, `ping`, `print`, `sort`, `subst`, `telnet`, `tracert`, `xcopy`, ...
- E altri possono essere aggiunti (*anche da noi*)

ARGOMENTI DALLA LINEA DI COMANDO: RIASSUNTO

- Gli argomenti sono stringhe
 - il primo (`argv[0]`) è il *nome del programma stesso*
 - i successivi, se esistono (`argc>1`), sono i parametri indicati dall'utente, nell'ordine, separati da spazi
 - per passare una stringa contenente spazi occorre *racchiuderla tra virgolette (es. "mario rossi")*, altrimenti sono considerati due argomenti distinti
- Passando stringhe che rappresentano numeri occorre, all'interno del programma, provvedere alla conversione stringa/numero tramite `atoi` o altra funzione adatta.