



Una panoramica sul C

Francesco Isgrò




Cosa vedremo

- Una veloce panoramica del linguaggio C
- Vedremo dei programmi di esempio e ne spiegheremo gli elementi
- Introduurremo brevemente i principali costrutti
- Approfondimenti durante il resto del corso



Output

- Un programma deve comunicare per essere utile
- Un programma produce sempre un output di qualche tipo
- Il primo esempio che vediamo è un programma che stampa una frase a schermo



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


```
int main(void)
```

```
{
```

```
    printf("from sea  to shining C\n");
```

```
    return 0;
```

```
}
```



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

```
int main(void)
```


```
{
```

```
    printf("from sea  to shining C\n");
```

```
    return 0;
```

```
}
```

- Scriviamo il testo su un file sea.c
- Passi successivi
 - Compilare il programma
 - Mandare in esecuzione il file eseguibile creato



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

```
int main(void)
{
    printf("from sea to shining C\n");

    return 0;
}
```

- Compilazione
 - gcc sea.c
 - Se non ci sono errori viene creato il file a.out
- Esecuzione
 - ./a.out
- Vediamo in pratica



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

```
int main(void)
{
    printf("from sea to shining C\n");

    return 0;
}
```

- Un preprocessor viene implicitamente eseguito prima della compilazione
- Linee di codice che iniziano con # sono direttive per il preprocessor
- Questa richiede di includere una copia dell'header file in quel punto del codice




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

```
int main(void)
{
    printf("from sea to shining C\n");

    return 0;
}
```

- Un preprocessor viene implicitamente eseguito prima della compilazione
- Linee di codice che iniziano con # sono direttive per il preprocessor
- Questa richiede di includere una copia dell'header file in quel punto del codice
- **N.B.** Boccio chiunque dica che include la libreria!!!!



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

```
int main(void)
```

```
{  
    printf("from sea to shining C\n");  
  
    return 0;  
}
```

- Prima riga della definizione della funzione main
- main(...) informa che main è una funzione
- int e void sono parole speciali che forniscono informazioni al compilatore
- void: la funzione non ha parametri
- int: la funzione restituisce un intero



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

```
int main(void)
```


```
{
```

```
    printf("from sea  to shining C\n");
```

```
    return 0;
```

```
}
```

- Ogni programma ha una funzione main
- L'esecuzione parte eseguendo questa funzione
- Il corpo della funzione è delimitato dalle {...}
- Le {...} sono usate per raggruppare istruzioni



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

```
int main(void)
```


```
{
```

```
    printf("from sea  to shining C\n");
```

```
    return 0;
```

```
}
```

- Funzione per stampare a schermo (in generale su stdout)
- Funzione contenuta nella standard library
 - un insieme di librerie che forniscono al programmatore funzioni tipizzate
 - gli header file di utilizzare l'insieme di funzioni di libreria associate
 - stdio.h fornisce le informazioni per la printf



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

```
int main(void)
```


```
{
```

```
    printf("from sea to shining C\n");
```

```
    return 0;
```

```
}
```

- Stringa costante
- Argomento della printf
- Controlla cosa viene stampato
- \n è un singolo carattere newline
 - Avanza il cursore all'inizio della riga successiva



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

```
int main(void)
```

```
{
```

```
    printf("from sea to shining C\n");
```

```
    return 0;
```

```
}
```

- Istruzione
- Le istruzioni terminano con un ;
- L'effetto di questa istruzione è stampare a schermo la stringa costante



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

```
int main(void)
```


```
{
```


```
    printf("from sea  to shining C\n");
```

```
    return 0;
```

```
}
```

- Istruzione di return
- Ritorna 0 al sistema operativo all'uscita del programma
- Il valore può essere usato per verificare l'uscita del programma
- La } indica la fine della funzione main()

- 
- La funzione printf stampa su tutto lo schermo
 - Da sinistra a destra
 - Dall'alto verso il basso
 - Muove il cursore accapo quando trova il carattere newline
 - Per essere leggibile l'output deve essere opportunamente spaziato sullo schermo



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

```
int main(void)
```

```
{
```


```
    printf("from sea to ");
```

```
    printf("shining ");
```

```
    printf("C\n");
```

```
    return 0;
```

```
}
```





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

```
int main(void)
{
    printf("from sea to ");
    printf("shining ");
    printf("C\n");

    return 0;
}
```

- Programma diverso da quello precedente
- Stampa lo stesso output
- Ogni chiamata della printf inizia da dove la precedente ha lasciato il cursore
- Se vogliamo scrivere su tre linee dobbiamo introdurre dei caratteri newline




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

```
int main(void)
{
    printf("from sea to\n");
    printf("shining\n");
    printf("C\n");

    return 0;
}
```

- In esecuzione stamperà
from sea to
shining
C




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

```
int main(void)
{
    printf("from sea to\n");
    printf("shining\nC\n");

    return 0;
}
```


- Si possono mettere più `\n` nella stessa `printf`



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

```
int main(void)
```

```
{  
    printf("\n\n\n\n\n\n\n\n\n\n\n");  
    printf("          *****\n");  
    printf("          *   from sea       *\n");  
    printf("          *   to shining C    *\n");  
    printf("          *****\n");  
    printf("\n\n\n\n\n\n\n\n\n\n\n\n");  
    return 0;  
}
```



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

```
int main(void)
{
    printf("\n\n\n\n\n\n\n\n\n\n\n");
    printf("          *****\n");
    printf("          *   from sea       *\n");
    printf("          *   to shining C    *\n");
    printf("          *****\n");
    printf("\n\n\n\n\n\n\n\n\n\n\n");
    return 0;
}
```

- L'output di questo programma:
 - La stessa frase
 - In una cornice di asterischi

Variabili, espressioni e assegnamento

- Vediamo un programma che converte una distanza
 - Input distanza maratona in miglia e iarde
 - Output distanza maratona in Km
- Regole
 - 1760 iarde in un miglio
 - 1.609 Km in un miglio



```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Compilazione

```
gcc -o marathon marathon.c
```

- L'opzione -o specifica il nome del file di output

- Esecuzione

```
./marathon
```


/* The distance of a marathon in kilometers. */

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Commento
 - Tutto ciò che è compreso fra /* e */ viene ignorato dal preprocessore e dal compilatore
 - Anche le linee che iniziano con // sono ignorate
- Un commento all'inizio serve a comunicare scopo del programma o delle funzioni
- Si può aggiungere commenti sulle varie versioni e autori

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)  
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Dichiarazione

- int è una parola chiave del C

- Indica un tipo di dato fondamentale

- Quelle che seguono sono variabili

- Si indica al compilatore che le variabile sono intere e assumono valori interi.

- Dichiarazioni terminano con un ;

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Dichiarazione

- float è una parola chiave del C
- Indica un tipo di dato fondamentale
- Si indica al compilatore che le variabile sono float e assumono valori reali.
- La variabile kilometers assume valori reali

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Istruzioni di assegnazione

- = operatore di assegnazione

- 26 e 385 sono costanti intere

- 26 assegnato alla variabile miles

- 285 assegnato alla variabile yards

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Istruzioni di assegnazione
 - Valore assegnato calcolato da una espressione
 - Operazioni dentro le parentesi eseguite per prime
 - Divisione ha precedenza su somma
 - Prima yards/1760.0
 - Il risultato viene sommato a miles e poi moltiplicato per 1.609



```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- 1.609 e 1760.0 costanti double
- yards/1760.0 diventa una divisione fra double con un risultato double
-

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Chiamata alla funzione printf
- Argomento è la stringa di controllo
- I parametri liberi vengono stampati con la conversione
 - %f specifica che stamperà un float
 - La variabile corrispondente deve essere float
 - In questo caso %f è associato a kilometers

```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Ora 1760 è una costante intera
- yards/1760 divisione fra interi



```
/* The distance of a marathon in kilometers. */
```

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

```
int main(void)
```

```
{
```

```
    int    miles, yards;
```

```
    float  kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

```
}
```

- Ora 1760 è una costante intera
- yards/1760 divisione fra interi
- Il resto della divisione si perde e di conseguenza il risultato non è corretto