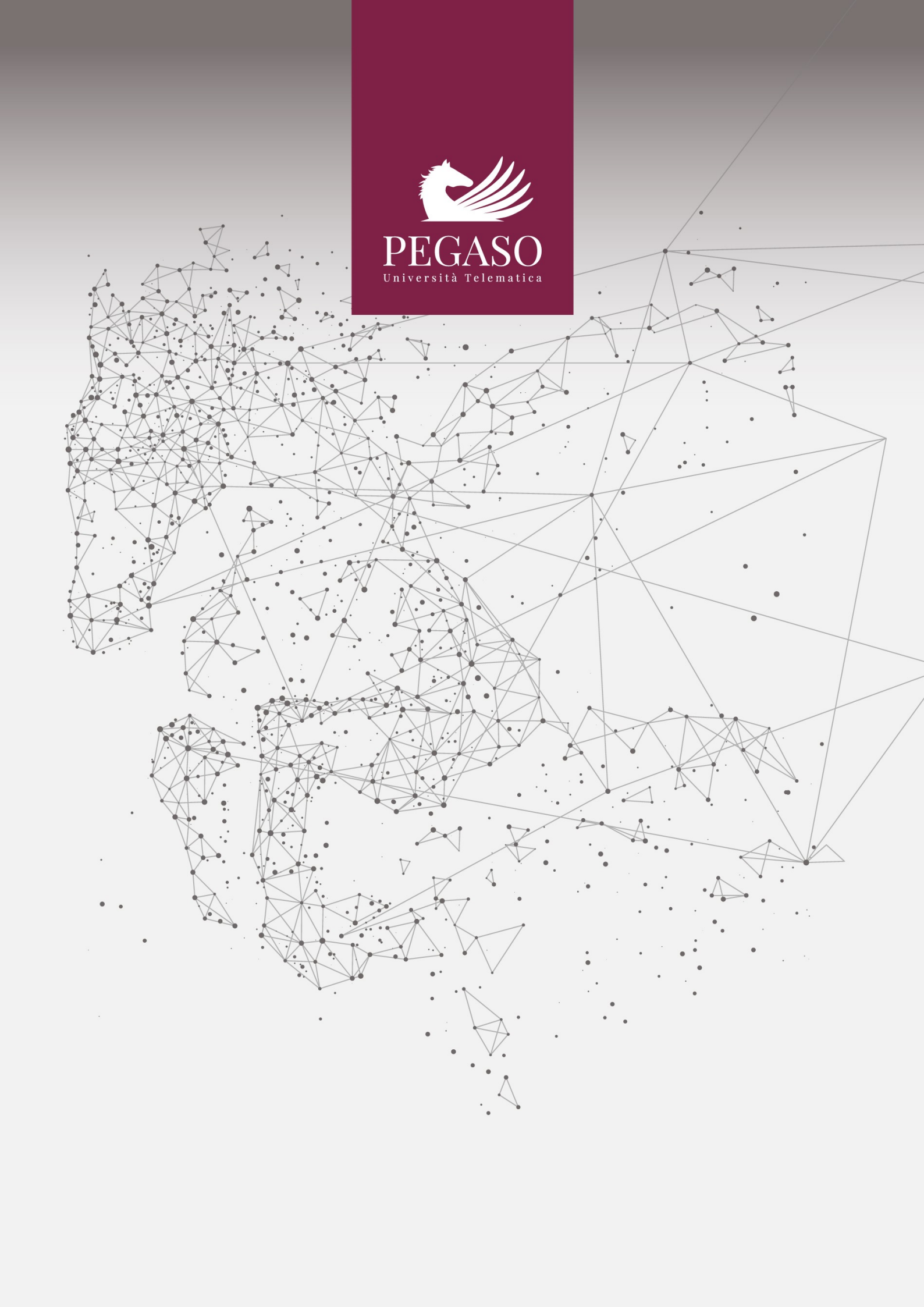
PEGASO
Università Telematica

Indice

1. Struttura del Programma

Consideriamo un semplice programma in C: il nostro primo esempio stampa a video una riga di testo.

```
1  // 03_01.c
2  // Un primo programma in C.
3  #include <stdio.h>
4
5  // la funzione main inizia l'esecuzione del programma
6  int main( void )
7  {
8     printf( "Welcome to C!\n" );
9  } // fine della funzione main
```

```
Welcome to C!
```

Commenti

Pur essendo semplice, questo programma illustra diverse caratteristiche importanti del linguaggio C.

Le righe 1 e 2

// 03_01.c

// Un primo programma in C.

iniziano con // per indicare che queste due righe sono commenti.

Inserite i commenti per documentare i programmi e migliorarne la leggibilità.

- I commenti non portano il computer a compiere alcuna azione durante l'esecuzione del programma.
- I commenti sono ignorati dal compilatore C e non provocano la generazione di alcun codice in linguaggio macchina.
- I commenti precedenti descrivono semplicemente il numero della figura, il nome del file e l'obiettivo del programma.
- I commenti aiutano anche altre persone a leggere e a capire il vostro programma.

Potete usare anche commenti multilinea /*…*/ in cui ogni cosa compresa tra /* nella prima riga e */ alla fine dell'ultima è un commento.

Sono preferibili commenti con // perché sono più brevi ed eliminano gli errori comuni di programmazione che possono capitare con commenti /*…*/, specialmente quando la chiusura */ è omessa.

Direttiva #include per il preprocessore

La riga 3

#include <stdio.h>

È un'istruzione per il preprocessore C. Le righe che cominciano con # sono elaborate dal preprocessore prima della compilazione.

La riga 3 dice al preprocessore di includere nel programma i contenuti del file di intestazione (header) di input/output standard (<stdio.h>).

Il file di intestazione contiene informazioni utilizzate dal compilatore durante la compilazione delle chiamate a funzioni della libreria di input/output standard come printf (riga 8).

Righe vuote e caratteri di spaziatura

La riga 4 è semplicemente una riga vuota.

Usate righe vuote e i caratteri di spazio e di tabulazione (cioè "tab") per rendere i programmi più leggibili.

Questi caratteri sono noti come caratteri di spaziatura.

I caratteri di spaziatura sono normalmente ignorati dal compilatore.

2. La funzione main

La riga 6

int main(**void**)

È parte di ogni programma in C.

Le parentesi dopo il main indicano che main è un blocco costituente di un programma, chiamato funzione.

I programmi in C contengono una o più funzioni, una delle quali deve essere proprio main.

Ciascun programma in C inizia l'esecuzione dalla funzione main.

Le funzioni possono restituire informazioni.

La parola chiave int alla sinistra di main indica che main "restituisce" un valore intero (numero intero).

Le funzioni possono anche ricevere informazioni quando vengono eseguite.

Il void qui tra parentesi vuol dire che main non riceve alcuna informazione.

Una parentesi graffa sinistra, {, inizia il corpo di ogni funzione (riga 7).

Una corrispondente parentesi graffa destra, }, termina ogni funzione (riga 9).

Questa coppia di parentesi graffe e la porzione del programma fra le parentesi è chiamata *blocco*. Il blocco è un importante costrutto sintattico in C.

😊 *Buona pratica di programmazione:*

Ogni funzione deve essere preceduta da un commento che ne descrive l'obiettivo.

3. La funzione printf

La riga 8

printf("Welcome to C!\n");

fa compiere un'azione al computer, ossia fa stampare sullo schermo la stringa di caratteri compresa tra le virgolette.

Una stringa è talvolta chiamata stringa di caratteri, messaggio o letterale.

L'intera riga, comprendente la funzione printf (la "f" sta per "formattato"), il suo argomento all'interno delle parentesi e il punto e virgola (;), è chiamata istruzione.

Ogni istruzione deve terminare con un punto e virgola (noto anche come *terminatore dell'istruzione*).

Quando la precedente istruzione printf viene eseguita, essa stampa sullo schermo il messaggio: Welcome to C!

I caratteri, di norma, vengono stampati così come appaiono nell'istruzione printf tra le doppie virgolette.

Sequenze di escape

Notate che i caratteri \n non sono stati stampati sullo schermo.

Il carattere di *backslash* (\, barra all'indietro) è detto carattere di *escape*.

Esso indica che printf è tenuta a fare qualcosa fuori dall'ordinario.

Quando incontra un backslash in una stringa, il compilatore guarda in avanti al carattere successivo e lo combina con il backslash per formare una sequenza di escape.

La sequenza di escape \n è detta newline (nuova riga).

Quando \n appare nella stringa argomento di una printf, il cursore viene posizionato all'inizio della riga successiva sullo schermo.

Alcune comuni sequenze di escape sono elencate di seguito.

Sequenza di escape	Descrizione
\n	Nuova riga. Posiziona il cursore all'inizio della riga successiva.
\t	Tab orizzontale. Sposta il cursore alla successiva tabulazione.
\a	Alert. Produce un suono o un allarme visibile senza cambiare la posizione corrente del cursore.
\\	Backslash. Inserisce un carattere di backslash in una stringa.
\"	Doppie virgolette. Inserisce un carattere di doppie virgolette in una stringa.

Poiché il backslash ha un significato speciale in una stringa, cioè il compilatore lo riconosce come un carattere di escape, usiamo un doppio backslash (\\) per collocarne uno singolo in una stringa.

Stampare le doppie virgolette costituisce anche un problema, perché esse segnano i confini di una stringa (tali virgolette non vengono stampate).

Usando la sequenza di escape \" in una stringa argomento di printf, indichiamo che printf deve stampare proprio le doppie virgolette.

La parentesi graffa destra, }, (riga 9) indica che è stata raggiunta la fine della funzione main.

☺ *Buona pratica di programmazione:*

Aggiungete un commento alla riga contenente la parentesi graffa destra, }, che chiude ciascuna funzione, compresa main.

Il linker e gli eseguibili

Le funzioni della Libreria Standard come printf e scanf non sono parti del linguaggio di programmazione C.

Ad esempio, il compilatore non è in grado di trovare un errore di ortografia in printf o scanf.

Quando il compilatore compila un'istruzione printf, introduce semplicemente uno spazio nel programma oggetto per una "chiamata" alla funzione della libreria.

Ma il compilatore non sa dove sono le funzioni della libreria, mentre chi lo sa è il linker.

Quando il linker è in esecuzione, localizza le funzioni della libreria e inserisce le chiamate appropriate a queste funzioni nel programma oggetto.

A questo punto il programma oggetto è completo e pronto per essere eseguito.

Per questo motivo, il programma elaborato dal linker è chiamato *eseguibile*.

Se il nome della funzione è scritto male, il linker scoprirà l'errore, perché non sarà in grado di collegare il nome nel programma in C con il nome di una qualsiasi funzione conosciuta nelle librerie.

☹ *Errore comune di programmazione:*

Scrivere in un programma print anziché printf per indicare la funzione di output.

😊 *Buona pratica di programmazione:*

Fate rientrare a destra il testo dell'intero corpo di ciascuna funzione con un livello di indentazione (raccomandiamo tre spazi) all'interno delle parentesi graffe che lo delimitano.

Questa indentazione mette in risalto la struttura funzionale dei programmi e aiuta a renderli più leggibili.

😊 *Buona pratica di programmazione:*

Stabilite una convenzione per l'entità dell'indentazione che preferite e poi applicatela uniformemente. Il tasto tab può essere utilizzato per effettuare le indentazioni, ma le tabulazioni possono variare. Le guide stilistiche professionali spesso raccomandano di utilizzare spazi anziché tabulazioni.

Uso di diversi printf

La funzione printf è in grado di stampare Welcome to C! in tanti modi diversi.

Ad esempio, il programma successivo (03_02.c) produce lo stesso output del primo programma (03_01.c).

Questo funziona perché ogni printf riprende a stampare da dove la printf precedente ha completato la stampa.

La prima printf (riga 8) stampa Welcome seguito da uno spazio (ma senza un carattere di newline), e la seconda printf (riga 9) inizia a stampare sulla stessa riga immediatamente dopo lo spazio.

```
1  03_02.c
2  // Stampare su una riga con due istruzioni printf.
3  #include <stdio.h>
4
5  // la funzione main inizia l'esecuzione del programma
6  int main( void )
7  {
8     printf( "Welcome " );
9     printf( "to C!\n" );
10 } // fine della funzione main
```

```
Welcome to C!
```

Una printf può stampare diverse righe usando ulteriori caratteri di newline come nell'esempio successivo (03_03.c).

Ogni volta che si incontra la sequenza di escape \n (newline), l'output continua all'inizio della riga successiva.

```
1  // 03_03.c
2  // Stampare diverse righe con un singolo printf.
3  #include <stdio.h>
4
5  // la funzione main inizia l'esecuzione del programma
6  int main( void )
7  {
8     printf( "Welcome\nto\nC!\n" );
9  } // fine della funzione main
```

```
Welcome
to
C!
```

Esercizi Proposti

1) Scrivete un programma che stampi a video una freccia utilizzando il carattere *

```
  *
 ***
*****
  *
  *
  *
  *
  *
```

2) Che cosa stampa il seguente codice?

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

Soluzioni

1) Esempio di un programma che stampa a video una freccia utilizzando il carattere *:

Versione 1:

```
#include <stdio.h>

int main( void )
{
   printf( "  *\n" );
   printf( " ***\n" );
   printf( "*****\n" );
   printf( "  *\n" );
   printf( "  *\n" );
   printf( "  *\n" );
   printf( "  *\n" );
```

```
}
```

Versione 2:

```
#include <stdio.h>

int main( void )
{
   printf( "  *\n" );
   printf( " ***\n" );
   printf( "*****\n" );
   printf( "  *\n  *\n  *\n  *\n" );
}
```

2) Il codice

printf("*\n**\n***\n****\n*****\n");

stampa a video:

```
*
**
***
****
*****
```

Bibliografia

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia