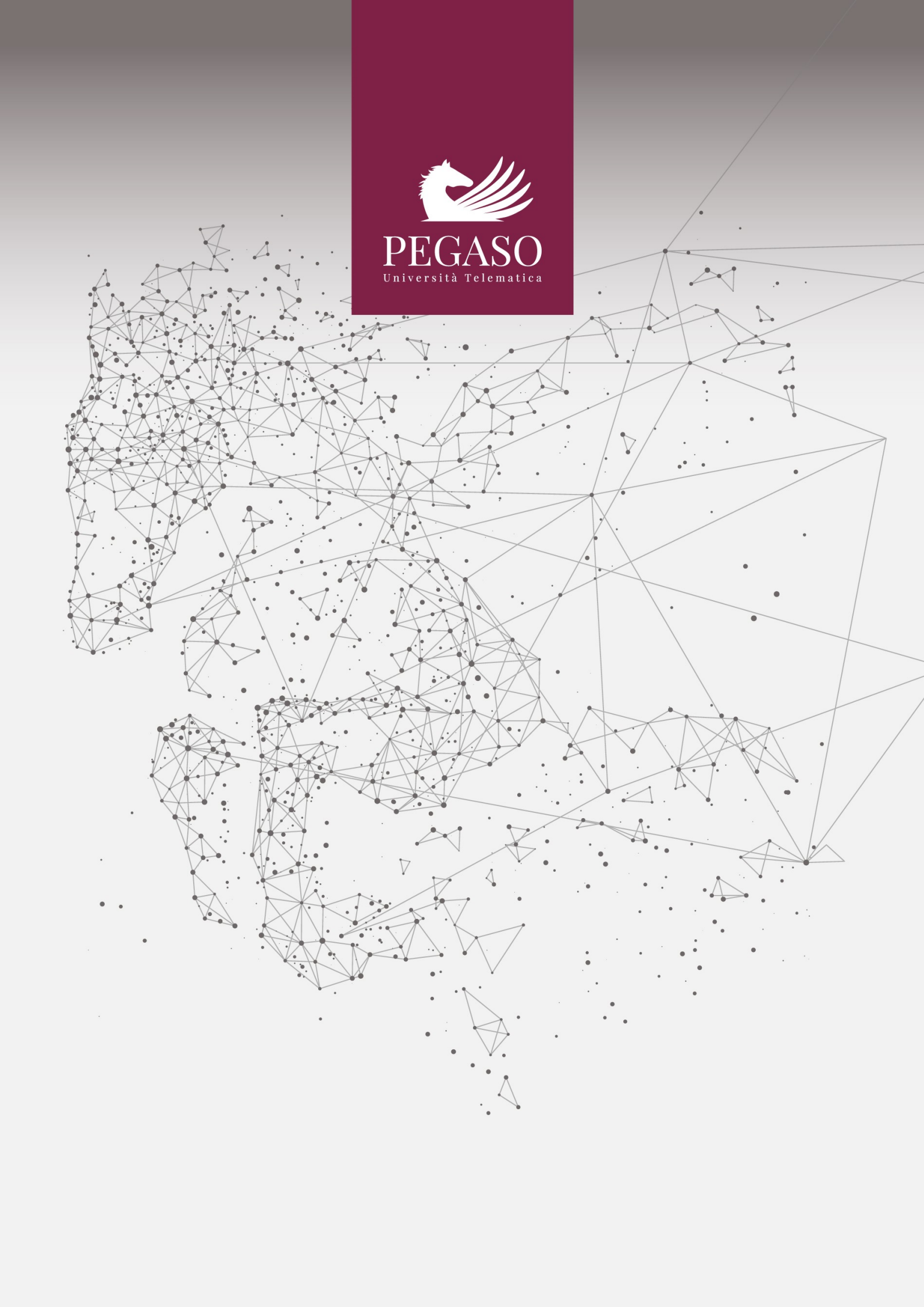
PEGASO
Università Telematica

Indice

1. Un semplice programma in C per addizionare due interi

Il nostro programma usa la funzione scanf della Libreria Standard per ricevere due valori interi scritti da un utente sulla tastiera, calcola la somma di questi valori e stampa il risultato usando printf.

```
1  // 04_01.c
2  // Programma per l'addizione.
3  #include <stdio.h>
4
5  // la funzione main inizia l'esecuzione del programma
6  int main( void )
7  {
8     int integer1; // primo numero inserito dall'utente
9     int integer2; // secondo numero inserito dall'utente
10
11    printf( "Enter first integer\n" ); // prompt
12    scanf( "%d", &integer1 ); // legge un intero
13
14    printf( "Enter second integer\n" ); // prompt
15    scanf( "%d", &integer2 ); // legge un intero
16
17    int sum; // variabile nella quale viene memorizzata la somma
18    sum = integer1 + integer2; // assegna il totale a sum
19
20    printf( "Sum is %d\n", sum ); // stampa la somma
21  } // fine della funzione main
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Il commento nelle righe 1–2 spiega l'obiettivo del programma. Ricordiamo infatti che ogni programma inizia l'esecuzione con main.

La parentesi graffa sinistra { (riga 7) segna l'inizio del corpo di main e la corrispondente parentesi graffa destra } (riga 21) segna la fine di main.

Variabili e definizioni di variabili

Le righe 8–10

int integer1; // primo numero inserito dall'utente

int integer2; // secondo numero inserito dall'utente

sono *definizioni*.

I nomi integer1, integer2 e sum sono nomi di *variabili*, locazioni in memoria dove si possono memorizzare i valori che verranno utilizzati dal programma.

Queste definizioni specificano che le variabili integer1, integer2 e sum sono di tipo int, il che significa che avranno valori interi, cioè numeri interi come 7, –11, 0, 31914 e simili.

Definire le variabili prima che siano utilizzate

Tutte le variabili devono essere definite con un nome e un tipo di dato *prima* di poter essere usate in un programma.

Il C standard permette di collocare dovunque dentro main ogni definizione di variabile prima del primo uso di quella variabile nel codice (sebbene alcuni vecchi compilatori non lo consentano).

È consigliato definire le variabili in prossimità del loro primo utilizzo.

Definire più variabili dello stesso tipo in un'unica istruzione

Le definizioni precedenti avrebbero potuto essere combinate in una singola definizione come segue:

int integer1, integer2;

ma questo avrebbe reso difficile associare i commenti con ciascuna delle variabili, come abbiamo fatto nelle righe 8–9.

Identificatori e maiuscolo/minuscolo

Il nome di una variabile in C può essere un qualunque identificatore valido.

Un identificatore è una serie di caratteri, consistenti in lettere, cifre e trattini (_), che *non* comincia con una cifra.

Il C è sensibile all'uso del carattere maiuscolo/minuscolo: le lettere maiuscole e minuscole sono differenti in C, così a1 e A1 sono identificatori differenti.

☹ *Errore comune di programmazione: Usare una lettera maiuscola quando se ne deve usare una minuscola (ad esempio scrivere Main invece di main).*

☺ *Prevenzione di errori: Evitate di usare come primo carattere di un identificatore il trattino (_) per prevenire conflitti con gli identificatori generati dal compilatore e con gli identificatori della Libreria Standard.*

☺ *Buona pratica di programmazione: Scegliere nomi significativi per le variabili aiuta a rendere un programma auto-documentato e quindi a ridurre la necessità di commenti.*

☺ *Buona pratica di programmazione: La prima lettera di un identificatore usato come semplice nome di una variabile deve essere minuscola. Nel seguito del testo assegneremo uno speciale significato agli identificatori che cominciano con una lettera maiuscola e a quelli che usano tutte lettere maiuscole.*

☺ *Buona pratica di programmazione: I nomi di variabili costituiti da più parole possono essere d'aiuto per rendere un programma più leggibile. Separate le parole con trattini come in*

total_commissions, o, se scrivete le parole attaccate, cominciate ogni parola dopo la prima con una lettera maiuscola, come in totalCommissions. L'ultimo stile – spesso chiamato notazione a cammello perché l'alternarsi di lettere maiuscole e minuscole ricorda la silhouette di un cammello – viene preferito.

Messaggi di prompting

La riga 11

printf("Enter first integer\n"); // prompt

stampa in uscita il letterale "Enter first integer" e posiziona il cursore all'inizio della riga successiva. Questo messaggio è chiamato prompt (letteralmente "richiesta di comandi") perché dice all'utente di compiere una specifica azione.

2. La funzione scanf e gli input formattati

La riga 12

scanf("%d", &integer1); // legge un intero

usa scanf (la "f" sta per "formattato") per ottenere un valore dall'utente. La funzione legge dallo standard input che solitamente è la tastiera.

Questa scanf ha due argomenti, "%d" e integer1.

Il primo, la stringa di controllo del formato, indica il tipo di dato che deve essere inserito dall'utente. Lo specificatore di conversione %d indica che il dato deve essere un intero (la lettera d sta per "decimale intero").

Il % in questo contesto è trattato da scanf (e da printf, come vedremo) come un carattere speciale con cui inizia uno specificatore di conversione.

Il secondo argomento di scanf inizia con il simbolo & – *ampersand*, chiamato operatore di indirizzo – seguito dal nome della variabile.

Il simbolo &, se combinato con il nome della variabile, comunica a scanf la locazione (o l'indirizzo) in memoria in cui è contenuta la variabile integer1.

Il computer quindi memorizza il valore che l'utente inserisce per integer1 in quella locazione. L'uso del simbolo & causa spesso confusione ai programmatori inesperti o a coloro che hanno programmato in altri linguaggi che non richiedono questa notazione. Per adesso ricordate soltanto di far precedere ciascuna variabile in ogni chiamata di scanf con il simbolo &. Alcune eccezioni a questa regola saranno esaminate successivamente.

Quando il computer esegue la funzione scanf di cui sopra, aspetta che l'utente inserisca un valore per la variabile integer1. L'utente risponde scrivendo un intero, poi premendo il tasto Invio per inviare il numero al computer.

Il computer quindi assegna questo numero, o valore, alla variabile integer1.

Qualunque riferimento successivo a integer1 in questo programma userà questo stesso valore.

Le funzioni printf e scanf facilitano l'interazione tra l'utente e il computer. Questa interazione assomiglia a un dialogo ed è spesso chiamata *elaborazione interattiva*.

☺ *Buona pratica di programmazione: Lasciate uno spazio dopo ogni virgola (,) per rendere i programmi più leggibili.*

Prompt e inserimento del secondo intero

La riga 14

printf("Enter second integer\n"); // prompt

stampa sullo schermo il messaggio Enter second integer, poi posiziona il cursore all'inizio della riga successiva. Anche questa printf chiede all'utente di compiere un'azione. La riga 15

scanf("%d", &integer2); // legge un intero

ottiene dall'utente un valore per la variabile integer2.

Nota: è possibile usare una scanf (riga 15) per leggere due interi nelle variabili int num1 e num2.

scanf("%d %d", &num1, &num2); // legge un intero

Ogni specificatore di conversione ha un argomento corrispondente nel quale sarà memorizzato il valore in ingresso. Il primo %d converte il valore da memorizzare nella variabile num1 e il secondo %d converte il valore da memorizzare nella variabile num2.

Definire la variabile sum

La riga 17

int sum; // variabile nella quale viene memorizzata sum

definisce la variabile sum di tipo int appena prima del suo utilizzo nella riga 18.

3. Istruzione di assegnazione

L'istruzione di assegnazione nella riga 18

sum = integer1 + integer2; // assegna il totale a sum

Calcola il totale delle variabili integer1 e integer2 e assegna il risultato alla variabile sum usando l'operatore di assegnazione =. L'istruzione è letta come "a sum è assegnato il valore dell'espressione integer1 + integer2". La maggior parte dei calcoli viene eseguita nelle istruzioni di assegnazione.

L'operatore = e l'operatore + sono chiamati operatori *binari* poiché ognuno ha due operandi. I due operandi dell'operatore + sono integer1 e integer2. I due operandi dell'operatore = sono sum e il valore dell'espressione integer1 + integer2.

☺ *Buona pratica di programmazione: lasciate spazi su entrambi i lati di un operatore binario. Questo fa sì che l'operatore sia in evidenza e rende il programma più leggibile.*

☹ *Errore comune di programmazione: un calcolo in un'istruzione di assegnazione deve essere sul lato destro dell'operatore =. È un errore di compilazione mettere un calcolo sul lato sinistro di un operatore di assegnazione.*

Stampare con una stringa di controllo del formato

La riga 20

printf("Sum is %d\n", sum); // stampa la somma

chiama la funzione printf per stampare il letterale "Sum is" seguito dal valore numerico della variabile sum sullo schermo.

Questa printf ha due argomenti, "Sum is %d\n" e sum.

Il primo è la stringa di controllo del formato. Essa contiene alcuni caratteri da stampare esattamente e lo specificatore di conversione %d che indica che sarà stampato un intero.

Il secondo argomento specifica il valore da stampare. Notate che lo specificatore di conversione per un intero è lo stesso sia in printf che in scanf (ciò vale per la maggior parte dei tipi di dati in C).

Combinare la definizione di una variabile e un'istruzione di assegnazione

È possibile assegnare un valore a una variabile nella sua definizione (questo è conosciuto come inizializzazione della variabile). Per esempio, le righe 17–18 possono essere combinate nell'istruzione

int sum = integer1 + integer2; // assegna il totale a sum

la quale somma integer1 e integer2, poi memorizza il risultato nella variabile sum.

Calcoli all'interno delle istruzioni printf

I calcoli possono anche essere eseguiti all'interno delle istruzioni printf. Per esempio, le righe 17–20 possono essere sostituite con l'istruzione

printf("Sum is %d\n", integer1 + integer2);

nel qual caso la variabile sum non è necessaria.

☹ *Errore comune di programmazione: dimenticare di far precedere una variabile in un'istruzione scanf dal simbolo &, quando invece è necessario farlo, provoca un errore al momento dell'esecuzione. Su molti sistemi questo causa un "errore di segmentazione" o una "violazione di accesso". Un simile errore si verifica quando il programma dell'utente tenta di accedere a una parte della memoria del computer per la quale non ha privilegi d'accesso.*

☹ *Errore comune di programmazione: far precedere una variabile inclusa in un'istruzione printf dal simbolo & quando, invece, non bisognerebbe farlo.*

Bibliografia

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.