



Il primo esperimento

Ver. 3

Lo scopo

- Per esemplificare l'attività di programmazione vengono mostrati quali siano i passi per risolvere il seguente semplice problema:

*scrivere un programma in linguaggio C che **chieda** all'utente di introdurre da tastiera **due numeri interi** e **visualizzi la loro somma** sul video*

Soluzione informale

- Le operazioni da far svolgere al calcolatore costituiscono l'algoritmo di soluzione di questo problema, esse saranno le seguenti (descrizione di tipo pseudocodice):
 1. Chiedi il primo numero
 2. Chiedi il secondo numero
 3. Calcola la somma del primo numero e del secondo numero
 4. Visualizza il risultato

Le variabili

- I numeri richiesti vengono immagazzinati nella memoria centrale del calcolatore (RAM)
- Ciascun numero sarà immagazzinato in una piccola porzione di memoria detta *variabile* costituita da un certo numero di byte
- A ogni variabile il programmatore dà un nome per potersi riferire ad essa (es. A, B, C, ...)

Le variabili

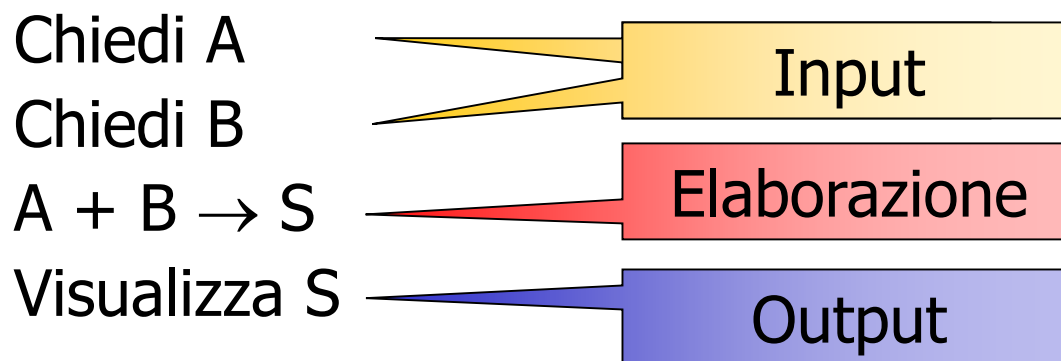
- Allora la sequenza delle operazioni da svolgere può essere descritta più dettagliatamente come segue:
 1. Chiedi il primo numero e mettilo in una variabile di nome A
 2. Chiedi il secondo numero e mettilo in una variabile di nome B
 3. Calcola la somma di A e B
 4. Visualizza il risultato

Le variabili

- Anche il risultato della somma dovrà essere immagazzinato nella memoria del calcolatore in una variabile, quindi avremo:
 1. Chiedi il primo numero e mettilo in una variabile di nome A
 2. Chiedi il secondo numero e mettilo in una variabile di nome B
 3. Calcola la somma di A e B e metti il risultato in una variabile di nome S
 4. Visualizza il contenuto della variabile S

Riduzione all'essenziale

- Riducendo al minimo indispensabile quanto scritto *senza perdere alcuna informazione*, si ha la seguente sequenza di operazioni:



- In generale ogni programma ha una fase di input, una di elaborazione e una di output

Input da tastiera

- *Input* significa "mettere dentro"
- Le operazioni di input sono quelle tramite le quali il calcolatore riceve dei dati dall'esterno
- Per leggere un numero dalla tastiera e memorizzarlo in una variabile si può ad esempio utilizzare la funzione `scanf`:

```
scanf ("%d", &A)
```
- Notare che la variabile **A** è preceduta dal carattere **&** (la motivazione in seguito)
- La funzione dell'esempio assegna ad **A** il valore (che qui deve essere intero) letto dalla tastiera

Output su video

- *Output* significa "mettere fuori"
- Le operazioni di output sono quelle tramite le quali il calcolatore mostra i risultati
- Per visualizzare il valore di una variabile si può utilizzare ad esempio la funzione `printf`:

```
printf("%d", s)
```
- Notare che la variabile `s` non è preceduta dal carattere `&`
- La funzione dell'esempio visualizza un numero (che in questo esempio deve essere intero): quello contenuto nella variabile `s`

Assegnazioni di variabili

- Assegnare un valore ad una variabile significa memorizzare in essa un *valore*
- Per assegnare un valore si usa il simbolo '=':
$$X = 12$$
- Il valore può essere il risultato di un calcolo:
$$S = A+B$$

questo indica che in S viene memorizzato il valore risultante dal calcolo $A+B$
- Il simbolo '=' significa "*prende il valore di*":
S prende il valore del risultato di $A+B$
- Ogni volta che si assegna un valore a una variabile, *il contenuto precedente viene perso*

Bozza di soluzione

- Con quanto visto, la sequenza di operazioni viste precedentemente diventa:

<i>Chiedi A</i>	→	<code>scanf ("%d", &A)</code>
<i>Chiedi B</i>	→	<code>scanf ("%d", &B)</code>
$A + B \rightarrow S$	→	<code>S = A + B</code>
<i>Visualizza S</i>	→	<code>printf ("%d", S)</code>

- Mancano ancora alcune parti perché questo sia un programma C completo

Punto e virgola

- Le istruzioni devono terminare sempre con un punto e virgola:

```
scanf ("%d", &A) ;
```

```
scanf ("%d", &B) ;
```

```
S = A + B ;
```

```
printf ("%d", S) ;
```

Definizione delle variabili

- Tutte le variabili utilizzate nel programma devono essere preventivamente elencate
- Le *definizioni* vanno collocate *tutte insieme* prima delle istruzioni eseguibili:

```
int A, B, S;
```

```
scanf ("%d", &A);
```

```
scanf ("%d", &B);
```

```
S = A + B;
```

```
printf ("%d", S);
```

- **int** indica che A, B ed S sono variabili adeguate a contenere valori **interi**

Il programma principale

- Le istruzioni che costituiscono il programma devono essere racchiuse tra parentesi graffe per formare un *blocco*. Se questo è l'unico che costituisce il programma, deve essere chiamato **main**:

```
main()  
{  
    int A, B, S;  
    scanf("%d", &A);  
    scanf("%d", &B);  
    S = A + B;  
    printf("%d", S);  
}
```

Il programma principale

- Per far terminare il programma, si usa l'istruzione **return** (seguita da un numero, in genere 0) collocata alla fine del blocco `main`:

```
main()  
{  
    int A, B, S;  
    scanf("%d", &A);  
    scanf("%d", &B);  
    S = A + B;  
    printf("%d", S);  
    return 0;  
}
```

I file di intestazione

- Per utilizzare le funzioni di input/output (I/O) contenute nelle librerie del C, è necessario che all'inizio del sorgente ci sia una direttiva che ne definisce l'uso mediante il *file di intestazione* :

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

```
main()
```

```
{
```

```
    int A, B, S;
```

```
    scanf("%d", &A);
```

```
    scanf("%d", &B);
```

```
    S = A + B;
```

```
    printf("%d", S);
```

```
    return 0;
```

```
}
```


int main()

- Molti compilatori richiedono che `main` sia preceduto da `int` (non obbligatorio in C89)

```
#include <stdio.h>
int main()
{
    int A, B, S;
    scanf("%d", &A);
    scanf("%d", &B);
    S = A + B;
    printf("%d", S);
    return 0;
}
```

- Ora il programma è completo ed è possibile compilarlo

Compilazione

- La compilazione del precedente codice sorgente produce un *file eseguibile*
- Mandando in esecuzione l'eseguibile (qui chiamato `primo.exe`), il programma chiederà due numeri (premere il tasto Invio dopo ciascun valore) e darà come risultato il valore della somma

```
C:\>primo.exe
12
23
35
```

Attende l'input (di A)

Attende l'input (di B)

Visualizza S

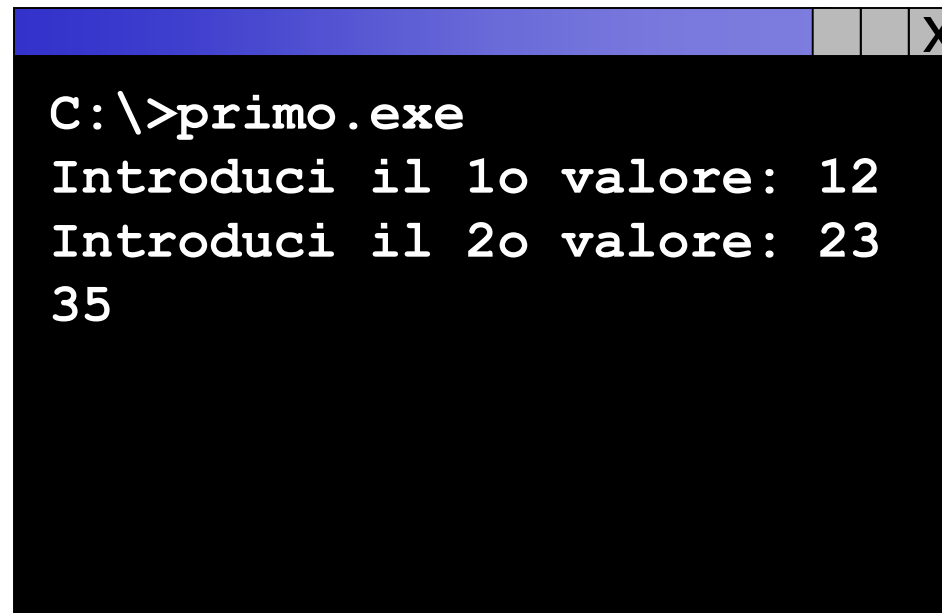
Miglioramento input

- Quando il programma parte, non informa l'utente di essere in attesa che questi introduca due valori da sommare
- E' utile quindi far precedere ciascuna `scanf` da un'istruzione `printf` che informi l'utente su che cosa debba fare:

```
printf("Introduci il 1o valore: ");  
scanf("%d", &A);  
printf("Introduci il 2o valore: ");  
scanf("%d", &B);
```

Miglioramento input

- Dopo i miglioramenti indicati, l'esecuzione produce un output come questo:



```
C:\>primo.exe
Introduci il 1o valore: 12
Introduci il 2o valore: 23
35
```

Miglioramento output

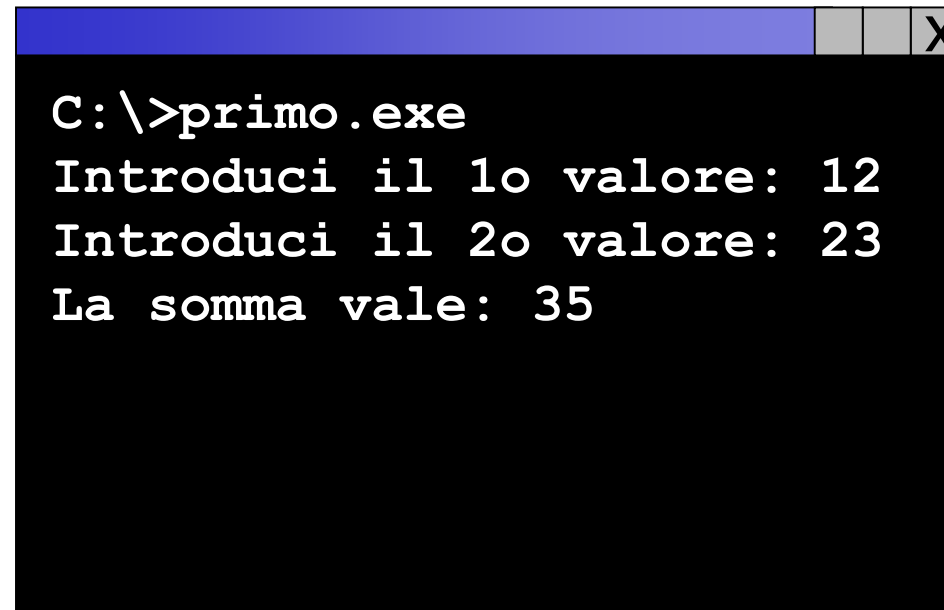
- Il risultato del calcolo viene mostrato a video come semplice numero, ma è preferibile aggiungere una scritta che ne specifichi il significato
- Si può aggiungere tale scritta tra le virgolette della stessa `printf` che visualizza `S`:

```
printf("La somma vale: %d", S);
```

il valore di `S` verrà sostituito al `%d`

Risultato finale

- Dopo i miglioramenti indicati, si ottiene il seguente output:



```
C:\>primo.exe
Introduci il 1o valore: 12
Introduci il 2o valore: 23
La somma vale: 35
```

Soluzione finale completa

```
#include <stdio.h>
int main()
{
    int A, B, S;

    printf("Introduci il 1o valore: ");
    scanf("%d", &A);
    printf("Introduci il 2o valore: ");
    scanf("%d", &B);

    S = A + B;
    printf("La somma vale: %d", S);
    return 0;
}
```

Esercizi

1. Scrivere un unico programma che chieda prima di introdurre 3 numeri A, B e C dalla tastiera e successivamente visualizzi il risultato dei tre seguenti calcoli:
1) $A-B$ 2) $A-B+C$ 3) $A-B+C+C$
2. Scrivere un programma che chieda due numeri e li memorizzi nelle variabili A e B.
Il programma deve ora scambiare il contenuto di A e di B (es. se inizialmente A contiene 12 e B 27, dopo lo scambio A contiene 27 e B 12).
Suggerimento: si immaginino 2 bicchieri, uno blu che contiene acqua e uno rosso che contiene aranciata, travasando opportunamente, si vuole che quello blu contenga aranciata e quello rosso acqua.

Soluzione esercizio 1

1. Prima soluzione:

```
#include <stdio.h>
int main()
{
    int A, B, C, X, Y, Z;
    printf("Primo valore? ");
    scanf("%d", &A);
    printf("Secondo valore? ");
    scanf("%d", &B);
    printf("Terzo valore? ");
    scanf("%d", &C);
```

continua →

Soluzione esercizio 1

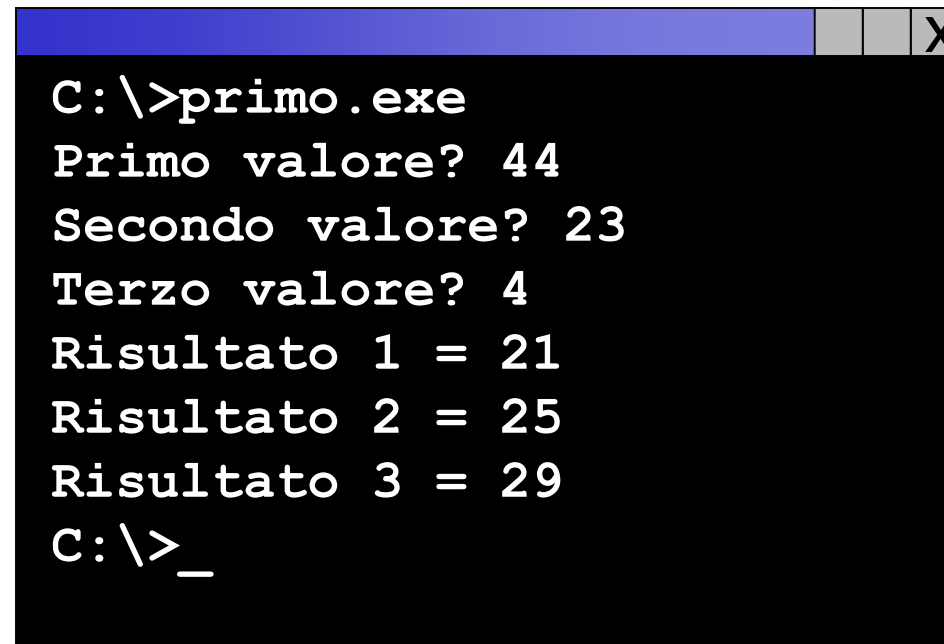
(Continuazione)

```
X = A-B;  
printf("Risultato 1 = %d\n", X);  
Y = A-B+C;  
printf("Risultato 2 = %d\n", Y);  
Z = A-B+C+C;  
printf("Risultato 3 = %d\n", Z);  
return 0;  
}
```

Nota: **\n** fa sì che la riga successiva venga visualizzata a capo (è il "ritorno a capo")

Soluzione esercizio 1

- Schermata risultante dall'esecuzione:



```
C:\>primo.exe
Primo valore? 44
Secondo valore? 23
Terzo valore? 4
Risultato 1 = 21
Risultato 2 = 25
Risultato 3 = 29
C:\>_
```

Variazioni esercizio 1

- Poiché i risultati dei calcoli sono memorizzati in variabili diverse, si possono prima eseguire i 3 calcoli e poi visualizzare i 3 risultati:

```
X = A-B;
```

```
Y = A-B+C;
```

```
Z = A-B+C+C;
```

```
printf("Risultato 1 = %d\n", X);
```

```
printf("Risultato 2 = %d\n", Y);
```

```
printf("Risultato 3 = %d\n", Z);
```

- Chiaramente quello che conta è che PRIMA vengano calcolati i valori e POI questi vengano visualizzati

Variazioni esercizio 1

- In questo esempio le variabili x , y e z dopo essere state visualizzate non vengono più utilizzate, si può allora risparmiare memoria definendo e utilizzando più volte solo la x (ma in questo caso non si possono fare prima tutti i 3 calcoli e poi tutte le 3 visualizzazioni):

```
X = A-B;
```

```
printf("Risultato 1 = %d\n", X);
```

```
X = A-B+C; → notare che il valore precedente di x viene perso
```

```
printf("Risultato 2 = %d\n", X);
```

```
X = A-B+C+C; → il valore precedente di x viene perso
```

```
printf("Risultato 3 = %d\n", X);
```

Variazioni esercizio 1

- Si possono anche utilizzare i valori intermedi:

```
X = A-B;  
printf("Risultato 1 = %d\n", X);
```

qui X vale A-B calcolato qui

```
X = X+C;  
printf("Risultato 2 = %d\n", X);
```

qui X vale A-B+C calcolato qui

```
X = X+C;  
printf("Risultato 3 = %d\n", X);
```

Variazioni esercizio 1

- Si possono indicare i calcoli anche nella `printf()`:

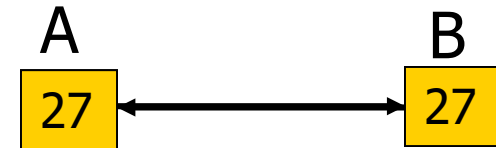
```
printf("Result 1 = %d\n", A-B);  
printf("Result 2 = %d\n", A-B+C);  
printf("Result 3 = %d\n", A-B+C+C);
```
- Quale soluzione sia la migliore dipende dal programma: se il programma continuasse e servissero tutti i 3 valori calcolati, solo le prime due soluzioni sarebbero adeguate
- In questo esempio, poiché i valori devono solo essere visualizzati, tutte le soluzioni presentate sono adeguate, compresa l'ultima che non memorizza i valori in nessuna variabile

Soluzione esercizio 2

- **Non** si può scrivere:

$A = B;$

$B = A;$



Eseguendo l'istruzione $A=B$:

il valore di A (12) viene sovrascritto dal valore di B (27), quindi ora $A=27$

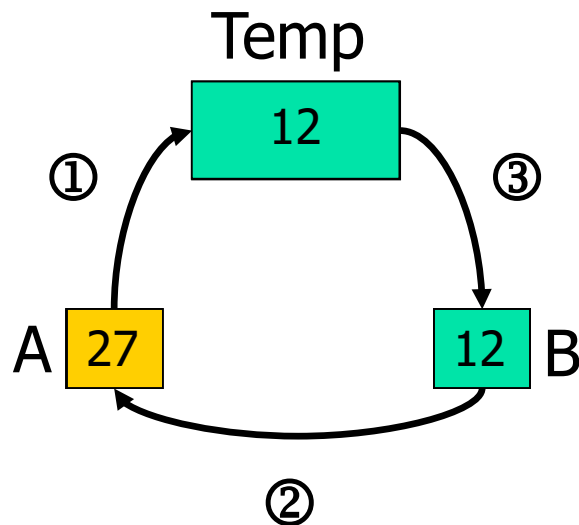
Nella successiva istruzione $B=A$:

B assume il *nuovo* valore di A (27, quello appena assegnato, cioè il *vecchio* B)

Quindi alla fine entrambe le variabili contengono il *vecchio* valore di B (27)

Soluzione esercizio 2

- Serve una variabile temporanea di scambio:
Temp = A; *salva il vecchio A in Temp*
A = B; *copia in A il vecchio B*
B = Temp; *copia in B il vecchio A salvato*
 in Temp



Il codice completo è nel file con le soluzioni

Soluzione esercizio 2

- La soluzione con 2 variabili temporanee è meno efficiente (usa una variabile in più ed esegue un'operazione di assegnamento in più) e quindi non è preferibile:

$$\text{TempA} = A$$

$$\text{TempB} = B$$

$$A = \text{TempB}$$

$$B = \text{TempA}$$

- La soluzione seguente funziona, ma potrebbe dare overflow nei calcoli ed è più lenta

$$A = A + B$$

$$B = A - B$$

$$A = A - B$$