Programmazione in C

Unità
Primo programma in C

Primo programma in C

- Introduzione al linguaggio C
- Struttura minima di un file C
- Sottoinsieme minimale di istruzioni
- Compilare il primo programma
- Esercizi proposti
- Sommario

Riferimenti al materiale

- Testi
  - Kernighan & Ritchie: capitolo 1
  - Cabodi, Quer, Sonza Reorda: capitoli 1, 3
  - Dietel & Dietel: capitolo 1
- Dispense
  - Scheda: “Primo programma in C”

Primo programma in C

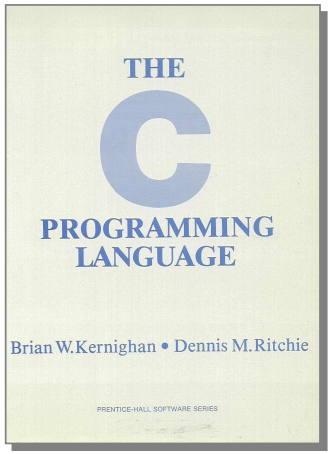
Introduzione al linguaggio C

Genesi del linguaggio C

- Sviluppato tra il 1969 ed il 1973 presso gli AT&T Bell Laboratories
  - B. Kernighan e D. Ritchie
  - Per uso interno
  - Legato allo sviluppo del sistema operativo Unix
- Nel 1978 viene pubblicato "The C Programming Language", prima specifica ufficiale del linguaggio
  - Detto "K&R"

Brian Kernighan

Dennis Ritchie

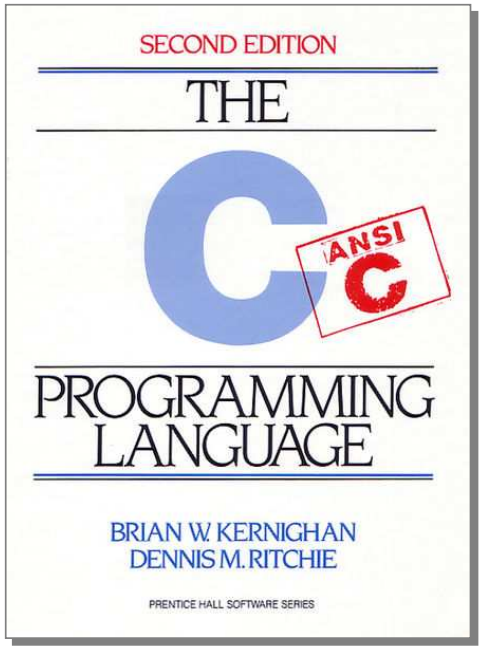


Obiettivi del linguaggio

- Insieme minimale di costrutti base
  - Semplicità del compilatore
- Sintassi estremamente sintetica
  - Talvolta criptica
- Pensato da programmatori per programmatori
  - Elevata efficienza
  - Per nulla user friendly
- Portabile
  - Indipendente dalla macchina
- Disponibilità di una libreria standard di funzioni

Evoluzione del linguaggio (1/2)

- 1978, K&R C
- 1989, ANSI C (o C89)
  - Frutto del lavoro di standardizzazione del comitato X3J11 dell'American National Standards Institute
  - Standard X3.159-1989 "Programming Language C"
  - Seconda edizione del K&R



Evoluzione del linguaggio (2/2)

- 1990, ISO C (o C90)
  - Ratifica da parte della International Organization for Standardization dello standard ANSI C
  - ISO/IEC 9899:1990
- 1999, ISO C99
  - Revisione compiuta negli anni ’90
  - INCITS-ANSI/ISO/IEC 9899-1999
    - 550 pagine
    - http://www.open-std.org/jtc1/sc22/wg14/
  - Supportata da molti (non tutti) i compilatori

Diffusione attuale

- I linguaggi attualmente più diffusi al mondo sono:
  - C
  - C++, un'evoluzione del C
  - Java, la cui sintassi è tratta da C++
  - C#, estremamente simile a Java e C++
- Il linguaggio C è uno dei linguaggi più diffusi
- La sintassi del linguaggio C è ripresa da tutti gli altri linguaggi principali

Principali vantaggi del C

- Basato su relativamente pochi costrutti da apprendere
- Enorme disponibilità di documentazione ed esempi
- Buona disponibiltà di ambienti di sviluppo gratuiti
- Disponibile su qualsiasi configurazione hardware
- Elevata efficienza di elaborazione
- Adatto a vari tipi di applicazioni
  - Programmi di sistema
  - Elaborazione numerica
  - Programmi interattivi

Principali svantaggi del C

- Scarsa leggibilità di alcuni costrutti
- Facilità nel commettere errori di programmazione
  - Molti costrutti "pericolosi" sono permessi dal linguaggio e quindi non vengono segnalati dal compilatore
  - Alcuni errori di digitazione possono causare comportamenti errati
- Difficoltà nella realizzazione di interfacce grafiche
- Complessità nell'elaborazione dei testi

Un esempio

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");

    return 0;
}
```

Primo programma in C

Struttura minima di un file C

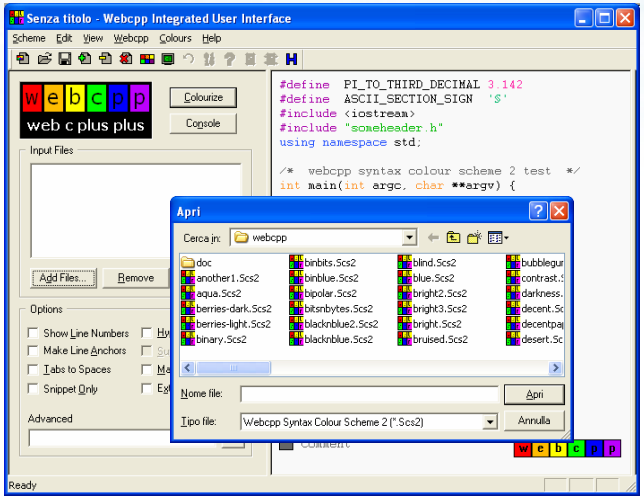
Struttura minima di un file C

- Applicazioni C in modo “console”
- Struttura del programma
- Commenti
- Direttive `#include`
- Definizione di variabili
- Corpo del `main`

Struttura minima di un file C

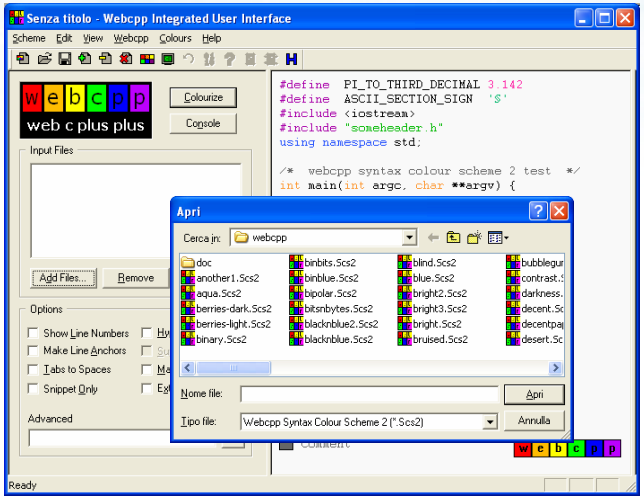
Applicazioni C in modo "console"

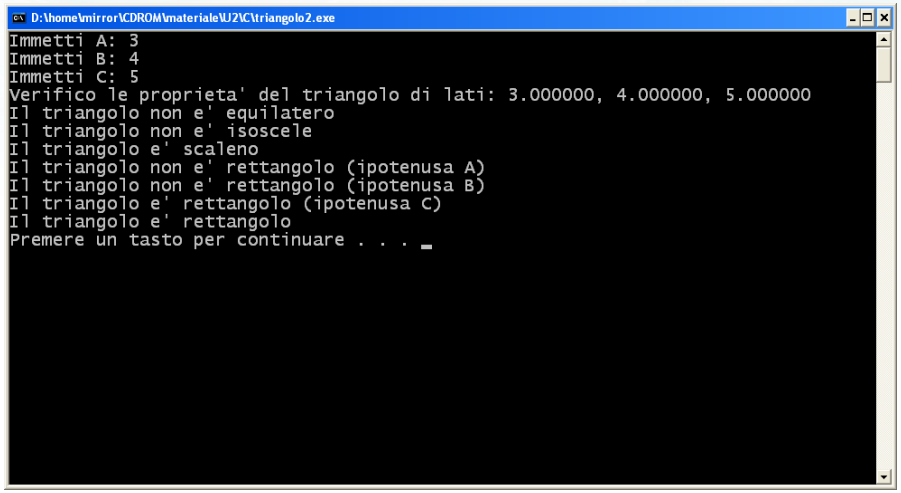
Tipi di applicazioni (1/4)



- Applicazioni grafiche
  - Interazione mediante mouse e finestre
  - Visualizzazione di testi e grafica
  - Elaborazione concorrente

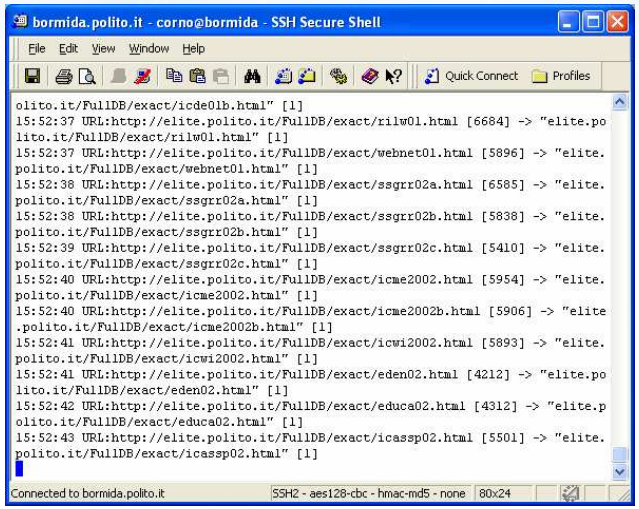
Tipi di applicazioni (2/4)





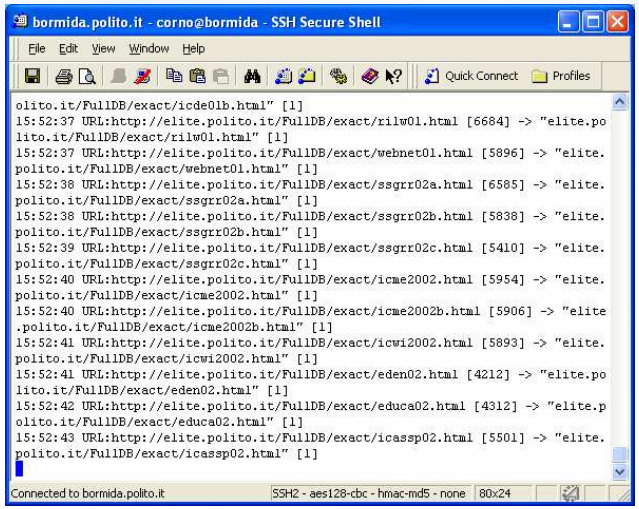
- Applicazioni grafiche
  - Interazione mediante mouse e finestre
  - Visualizzazione di testi e grafica
  - Elaborazione concorrente
- Applicazioni "console"
  - Interazione mediante tastiera
  - Visualizzazione di soli caratteri
  - Elaborazione sequenziale

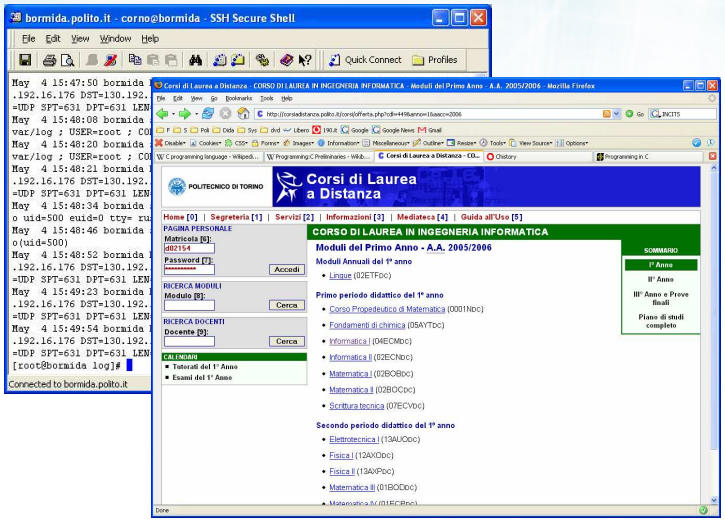
Tipi di applicazioni (3/4)



- Applicazioni batch
  - Nessuna interazione utente
  - Compiti lunghi e ripetitivi
  - Elaborazione numerica, trasferimenti in rete

Tipi di applicazioni (4/4)



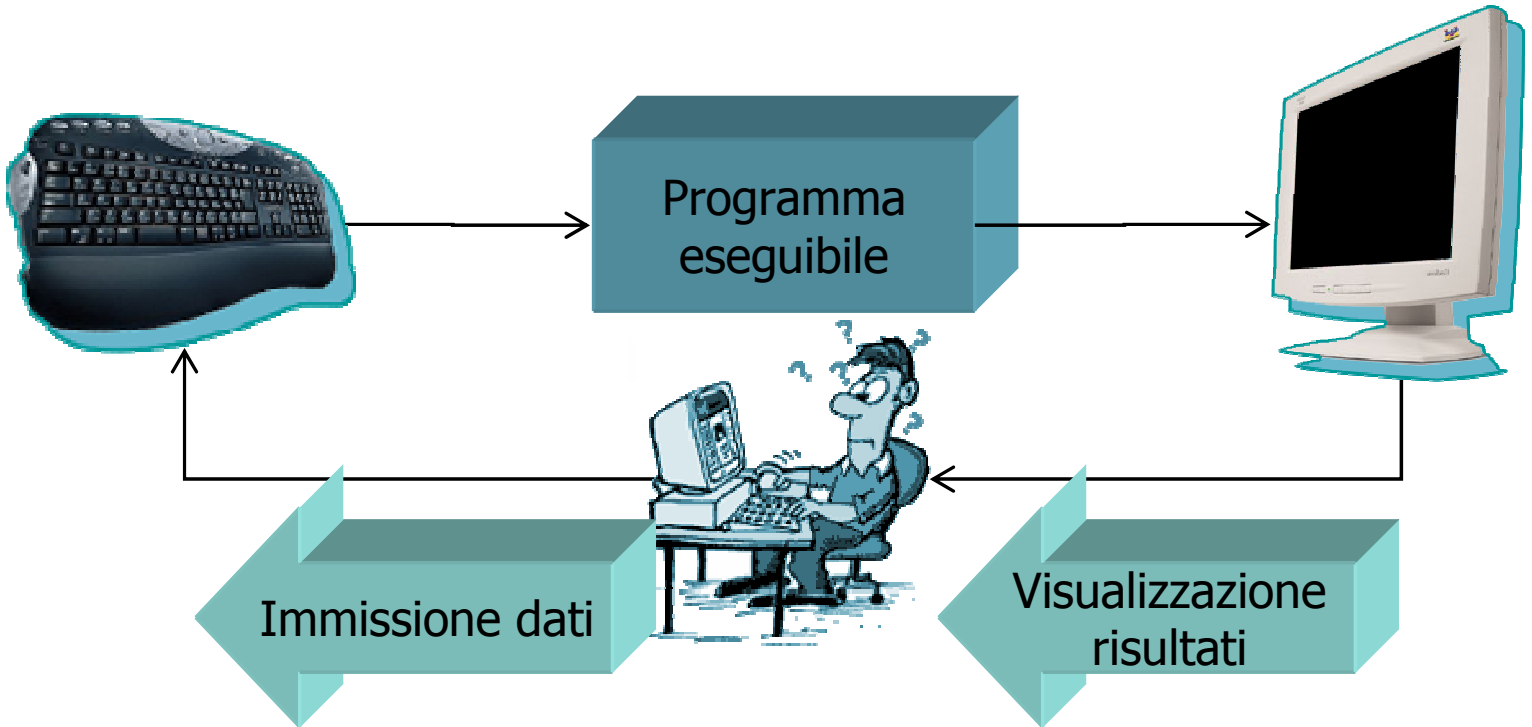


- Applicazioni batch
  - Nessuna interazione utente
  - Compiti lunghi e ripetitivi
  - Elaborazione numerica, trasferimenti in rete
- Applicazioni server
  - Nessuna interazione utente
  - Realizzano funzioni di sistema
  - Server locali o server Internet

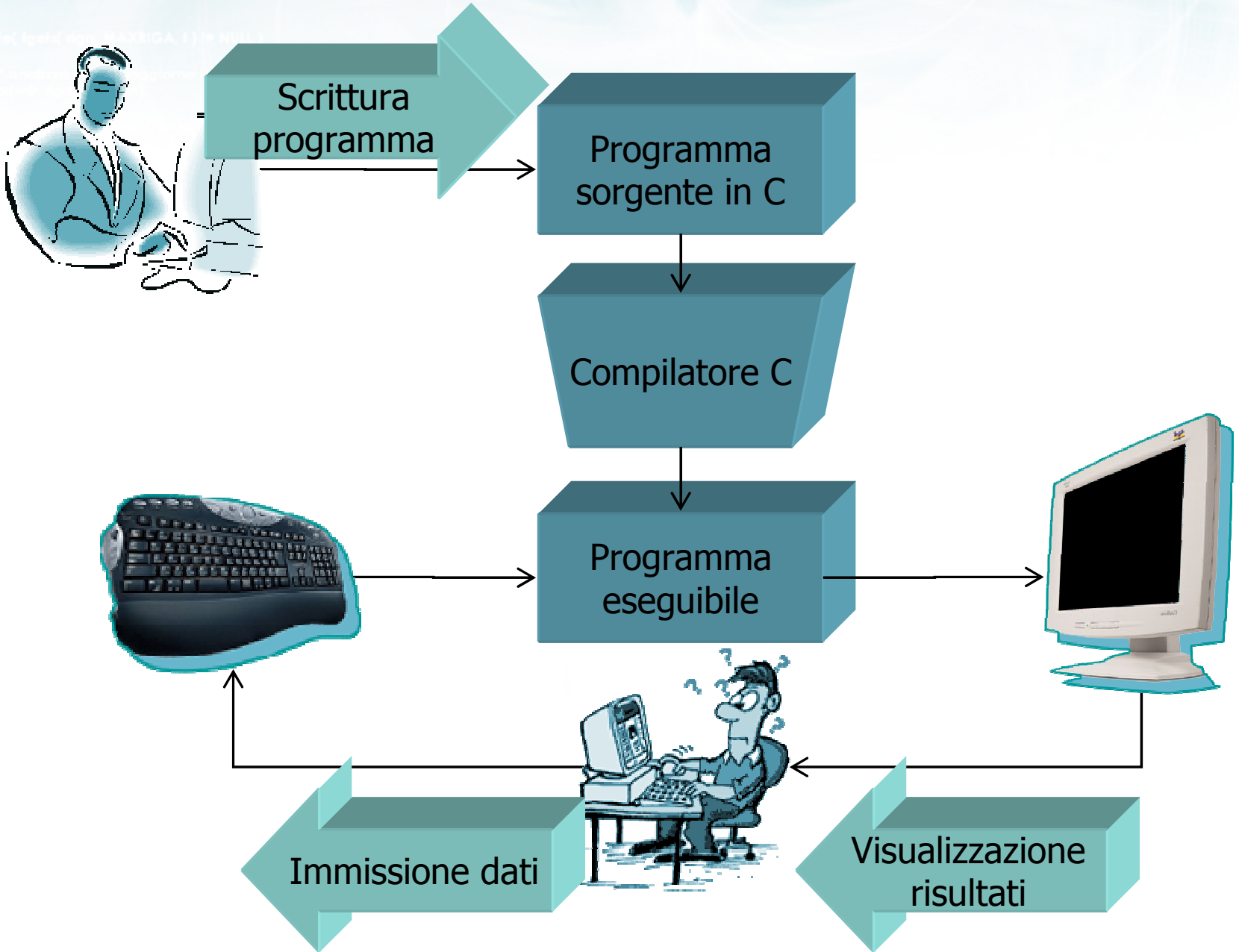
Applicazioni "console"

- Interazione utente limitata a due casi
  - Stampa di messaggi, informazioni e dati a video
  - Immissione di un dato via tastiera
- L'insieme tastiera+video viene detto terminale
- Nessuna caratteristica grafica
- Elaborazione
  - Sequenziale
  - Interattiva
  - Mono-utente

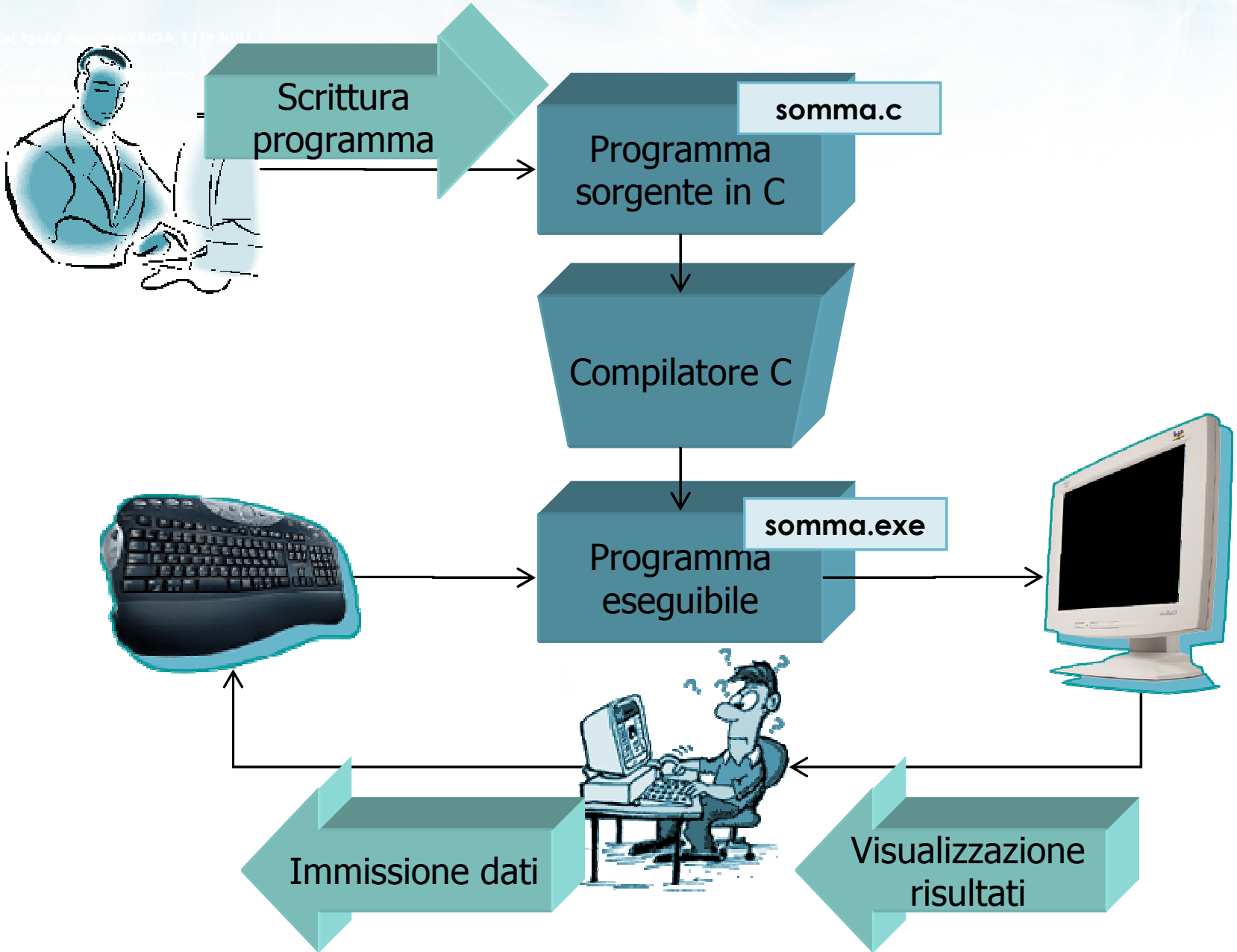
Modello di applicazioni "console"



Modello di applicazioni "console"



Modello di applicazioni "console"

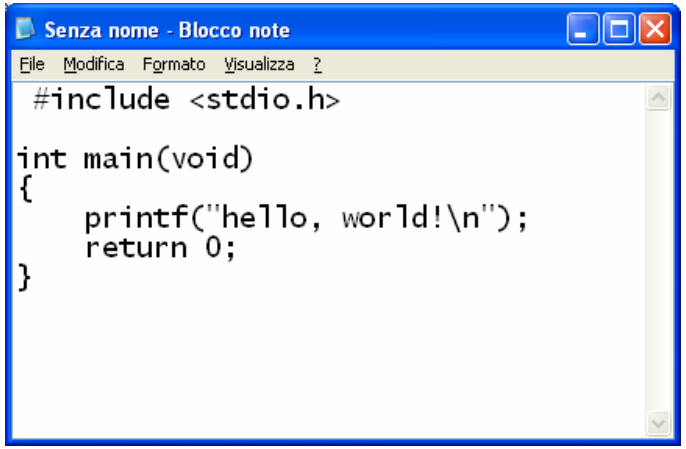


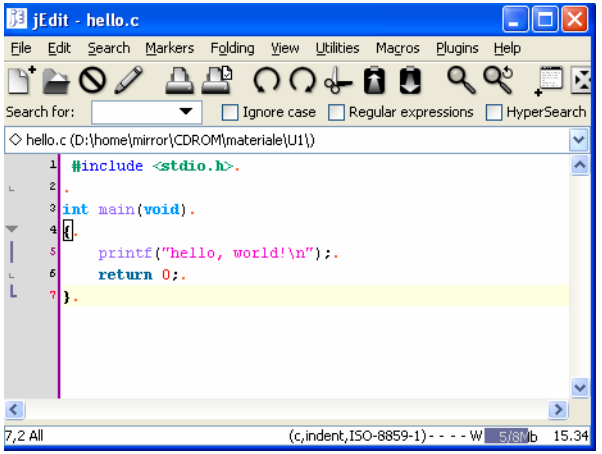
Compilatore C

- Traduce i programmi sorgenti scritti in linguaggio C in programmi eseguibili
- È a sua volta un programma eseguibile, a disposizione del programmatore
- Controlla l'assenza di errori di sintassi del linguaggio
- Non serve all'utente finale del programma
- Ne esistono diversi, sia gratuiti che commerciali

Scrittura del programma

- Un sorgente C è un normale file di testo
- Si utilizza un editor di testi
  - Blocco Note
  - Editor specializzati per programmatori



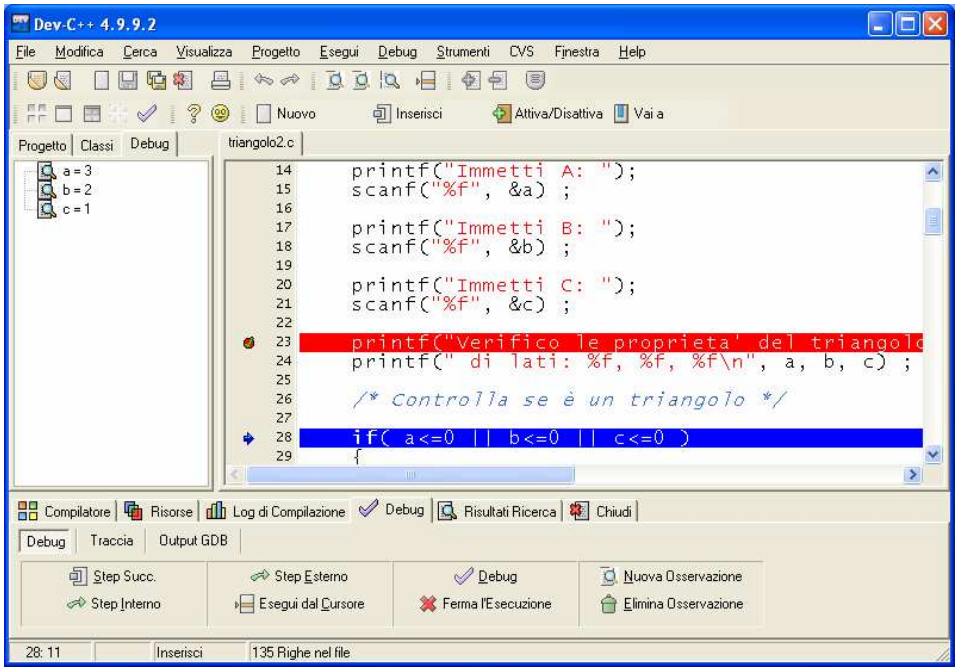


Editor per programmatori

- Colorazione ed evidenziazione della sintassi
- Indentazione automatica
- Attivazione automatica della compilazione
- Identificazione delle parentesi corrispondenti
- Molti disponibili, sia gratuiti che commerciali

Ambienti integrati

- Applicazioni software integrate che contengono al loro interno
  - Un editor di testi per programmatori
  - Un compilatore C
  - Un ambiente di verifica dei programmi (debugger)
- IDE: Integrated Development Environment



Struttura minima di un file C

Struttura del programma

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;

    a = 3 ;

    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

Struttura di un sorgente in C

Programma principale
(funzione `main`)

```
#include <stdio.h>

int main(void)
{
    int a ;

    a = 3 ;

    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;

    a = 3 ;

    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

Parentesi graffe che delimitano il `main`

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
  int a ;

   a = 3 ;

   printf("hello, world\n");
   printf("the magic number is %d\n", a) ;

   return 0;
}
```

Variabili utilizzate dal programma

Struttura di un sorgente in C

```
#include <stdio.h>

int main(void)
{
    int a ;

    a = 3 ;

    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

Istruzioni eseguibili

Struttura di un sorgente in C

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

Richiamo delle librerie utilizzate

```
int main(void)
{
    int a ;

    a = 3 ;

    printf("hello, world\n");
    printf("the magic number is %d\n", a) ;

    return 0;
}
```

In generale

```
#include delle librerie

int main(void)
{

   definizione variabili

   istruzioni eseguibili

}
```

Struttura minima di un file C

Commenti

Commenti

- Il testo presente in un sorgente C deve essere analizzato dal compilatore C, quindi deve sottostare a tutte le regole sintattiche del linguaggio
- Per aggiungere annotazioni, commenti, spiegazioni, note, ... si può usare un commento all'interno del sorgente

```
/* io sono un commento */
```

Sintassi

- Un commento è una qualsiasi sequenza di caratteri (anche su più righe) che:
  - Inizia con la coppia di caratteri /*
  - Termina con la coppia di caratteri */
- Non è permesso annidare commenti
  - All'interno di un commento non devono comparire i caratteri /*
- Tutto ciò che è compreso tra /* e */ viene ignorato dal compilatore C

Esempio

```
/* programma: hello.c
   autore: fulvio corno
 */

/* accedi alla libreria standard */
#include <stdio.h>

int main(void)
{
   int a ;  /* numero magico */

   a = 3 ;  /* assegno un valore */

   /* salutiamo l'utente */
   printf("hello, world\n") ;
   printf("the magic number is %d\n", a) ;

   return 0;
}
```

Spazi bianchi

- Oltre ai commenti, il compilatore ignora tutti gli spazi bianchi
  - Spazi tra un'istruzione e la successiva
  - Spazi ad inizio linea
  - Spazi intorno alla punteggiatura
  - Righe vuote
- La spaziatura viene utilizzata per rendere il sorgente C più ordinato e più facilmente comprensibile

Esempio

```
/* programma: hello.c  autore: fulvio corno */
/* accedi alla libreria standard */
#include <stdio.h>
int main(void)
{ int a ;  /* numero magico */ a = 3 ;
/* assegno un valore */
/* salutiamo l'utente */ printf("hello, world\n") ;
printf("the magic number is %d\n", a) ; return 0;
}
```

Esempio

```
/* programma: hello.c  autore: fulvio corno */
/* accedi alla libreria standard */
#include <stdio.h>
int main(void)
{ int a ;  /* numero magico */ a = 3 ;
/* assegno un valore */
/* salutiamo l'utente */ printf("hello, world\n") ;
printf("the magic number is %d\n", a) ; return 0;
}
```

```
#include <stdio.h>
int main(void)
{ int a ; a = 3 ; printf("hello, world\n") ;
printf("the magic number is %d\n", a) ; return 0; }
```

Struttura minima di un file C

Direttive `#include`

Librerie di funzioni

- Ogni compilatore C dispone di diverse librerie di funzioni già pronte per l'uso
- Il programmatore può utilizzare le funzioni di libreria
- È necessario dichiarare a quali librerie si vuole avere accesso
  - Direttive `#include` ad inizio programma
  - Aggiunge al programma le dichiarazioni di tutte le funzioni di tale libreria, permettendo al programmatore di richiamare tali funzioni

Sintassi

```
#include < NomeLibreria .h>
```

- Librerie principali:
  - `#include <stdio.h>`
    - Funzioni di lettura/scrittura su terminale e su file
  - `#include <stdlib.h>`
    - Funzioni base per interazione con sistema operativo
  - `#include <math.h>`
    - Funzioni matematiche
  - `#include <string.h>`
    - Elaborazione di testi

Avvertenze

- A differenza della regola generale, nelle direttive `#include` la spaziatura è importante
  - Il carattere # deve essere il primo della riga
  - Può esserci una sola `#include` per riga
  - La direttiva `#include` non va terminata con il `;`
- Dimenticare una `#include` potrà portare ad errori nel corpo del `main`, quando si chiameranno le funzioni relative

Suggerimenti

- Iniziare sempre il sorgente C con le seguenti linee:

```
/* programma: NomeFile.c
   autore: NomeAutoreDelProgramma
   BreveDescrizioneDelProgramma
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void)
{

   . . . .

}
```

Struttura minima di un file C

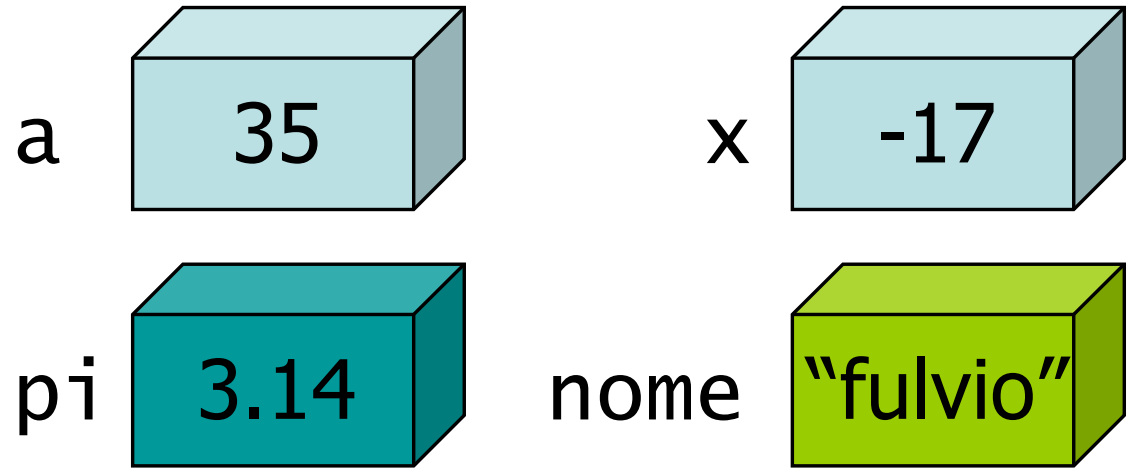
Definizione di variabili

Variabili

- Il programma memorizza le informazioni sulle quali lavora all'interno di variabili
- Ogni variabile è caratterizzata da:
  - Tipo di dato
  - Nome
  - Valore corrente

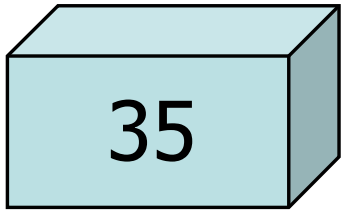
Variabili

- Il programma memorizza le informazioni sulle quali lavora all'interno di variabili
- Ogni variabile è caratterizzata da:
  - Tipo di dato
  - Nome
  - Valore corrente

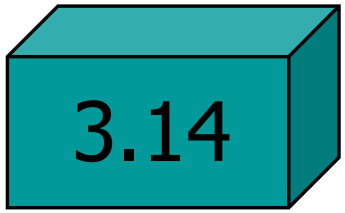


Tipo di dato

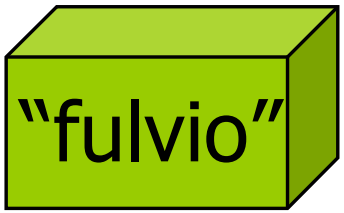
- Definisce l'insieme dei valori ammissibili per la variabile



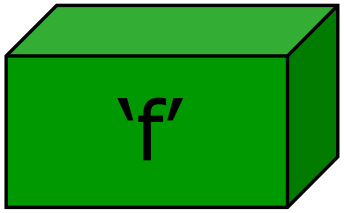
Numeri interi, positivi o negativi



Numeri reali



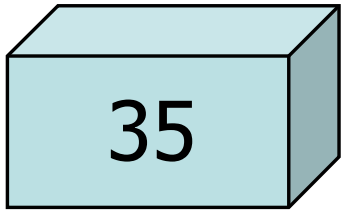
Stringhe di testo



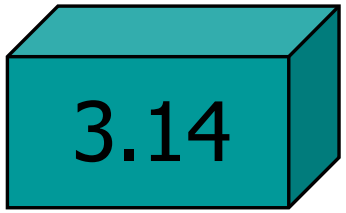
Singoli caratteri di testo

Tipo di dato

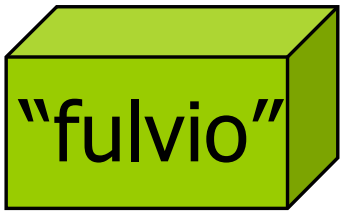
- Definisce l'insieme dei valori ammissibili per la variabile

`int` 

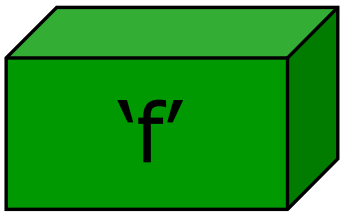
Numeri interi, positivi o negativi

`float` 

Numeri reali



Stringhe di testo

`char` 

Singoli caratteri di testo

Nome

- Il programmatore assegna un nome a ciascuna variabile
- Dovrebbe rappresentare lo scopo dei valori contenuti nella variabile
- Sintetico, rappresentativo, mnemonico, facile da scrivere

Nomi ammissibili

- Il primo carattere deve essere una lettera
- I successivi possono essere lettere o numeri
- Lettere maiuscole e minuscole sono diverse
- Il simbolo _ viene considerato come una lettera
- Non devono essere nomi riservati dal linguaggio

Esempi di nomi

a b a1 a2

Esempi di nomi

a b a1 a2

num n N somma max

Esempi di nomi

a b a1 a2

num n N somma max

area perimetro perim

Esempi di nomi

```
a          b          a1          a2

num        n          N           somma       max

area       perimetro  perim

n_elementi Nelementi  risultato
```

Esempi di nomi

a b a1 a2

num n N somma max

area perimetro perim

n_elementi Nelementi risultato

trovato nome risposta

Definizione di variabili

- Ogni variabile deve essere definita prima di poterla utilizzare
- Definizioni all'inizio della funzione `main`
- Sintassi della definizione
  - *TipoVariabile NomeVariabile* ;

```
int main(void)
{
    int a ;
    int b ;
    float x ;

    . . . . .
}
```

Definizione di variabili

- Ogni variabile deve essere definita prima di poterla utilizzare
- Definizioni all'inizio della funzione `main`
- Sintassi della definizione
  - *TipoVariabile NomeVariabile* ;
  - *TipoVariabile NomeVariabile*, *NomeVariabile* ;

```
int main(void)
{
    int a ;
    int b ;
    float x ;

    . . . . .
}
```

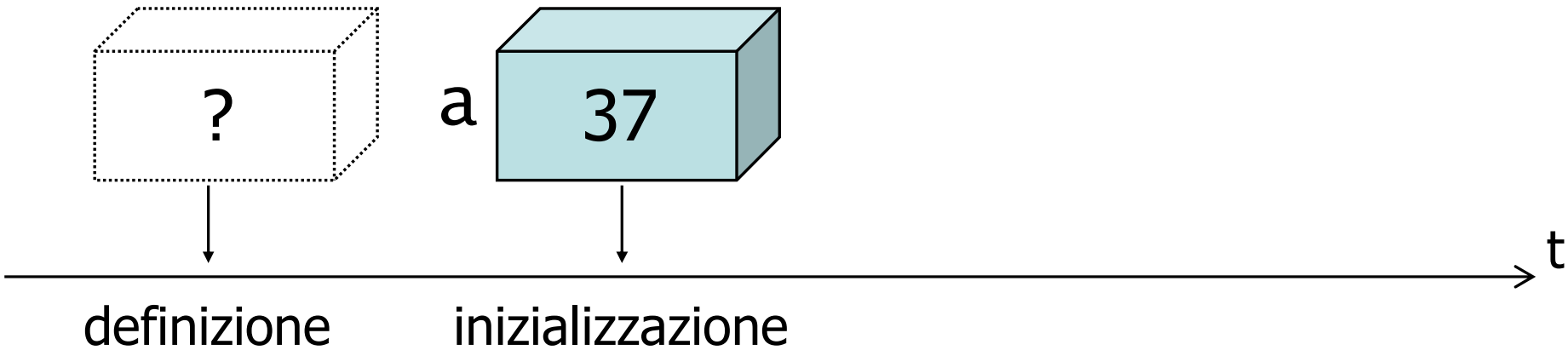
```
int main(void)
{
    int a, b ;
    float x ;

    . . . . .
}
```

Valore contenuto

- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
  - Variabili non inizializzate
- In momenti diversi il valore può cambiare

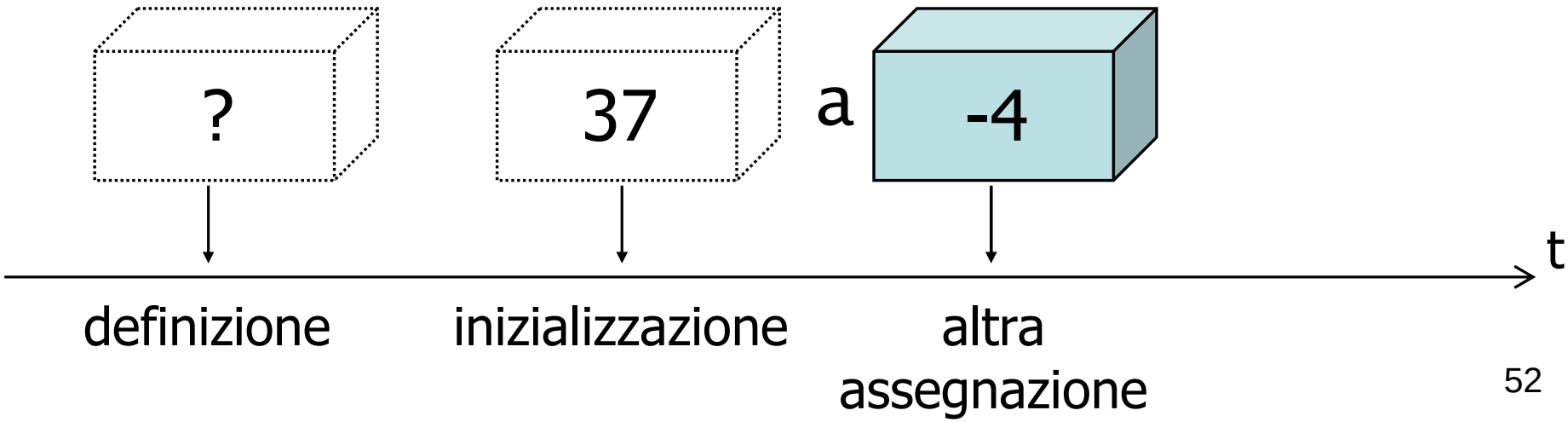
Valore contenuto

- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
  - Variabili non inizializzate
- In momenti diversi il valore può cambiare



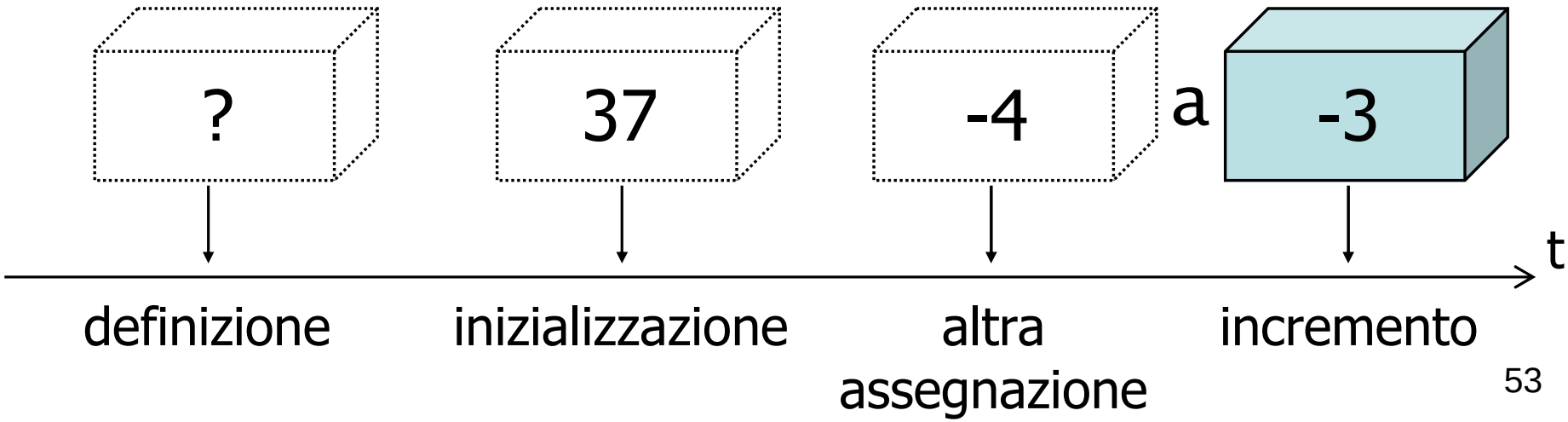
Valore contenuto

- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
  - Variabili non inizializzate
- In momenti diversi il valore può cambiare



Valore contenuto

- Ogni variabile, in ogni istante di tempo, possiede un certo valore
- Le variabili appena definite hanno valore ignoto
  - Variabili non inizializzate
- In momenti diversi il valore può cambiare



Struttura minima di un file C

Corpo del `main`

Istruzioni eseguibili

- La funzione `main`, dopo le definizioni di variabili, contiene le vere e proprie istruzioni eseguibili
- Ciascuna istruzione è terminata da `;`
- Tutte le istruzioni sono comprese nelle `{ ... }`
- Le istruzioni vengono eseguite in ordine
- Dopo aver eseguito l'ultima istruzione, il programma termina

Esempio

```
/* programma: hello.c
   autore: fulvio corno
 */

/* accedi alla libreria standard */
#include <stdio.h>

int main(void)
{
   int a ;  /* numero magico */

   a = 3 ;  /* assegno un valore */

   /* salutiamo l'utente */
   printf("hello, world\n") ;
   printf("the magic number is %d\n", a) ;

   return 0;
}
```

Tipologie di istruzioni

- Istruzioni operative
  - Lettura dati
    - `scanf("%d", &a) ;`
  - Stampa risultati
    - `printf("%d", a) ;`
  - Elaborazione numerica
    - `a = b + c ;`
    - `b = b + 1 ;`
    - `c = 42 ;`
    - `c = sqrt(a) ;`

Tipologie di istruzioni

- Istruzioni operative
  - Lettura dati
    - `scanf("%d", &a) ;`
  - Stampa risultati
    - `printf("%d", a) ;`
  - Elaborazione numerica
    - `a = b + c ;`
    - `b = b + 1 ;`
    - `c = 42 ;`
    - `c = sqrt(a) ;`
- Istruzioni di controllo
  - Modificano il controllo di flusso
    - Scelte
    - Iterazioni
    - Chiamate a funzioni
    - Interruzioni e salti
  - Predefinite dal linguaggio C
    - `if else while for return switch case break continue goto`

Primo programma in C

Sottoinsieme minimale di istruzioni

Sottoinsieme minimale di istruzioni

- I tipi `int` e `float`
- Istruzione `printf` – semplificata
- Istruzione `scanf` – semplificata
- Istruzione di assegnazione
- Semplici espressioni aritmetiche

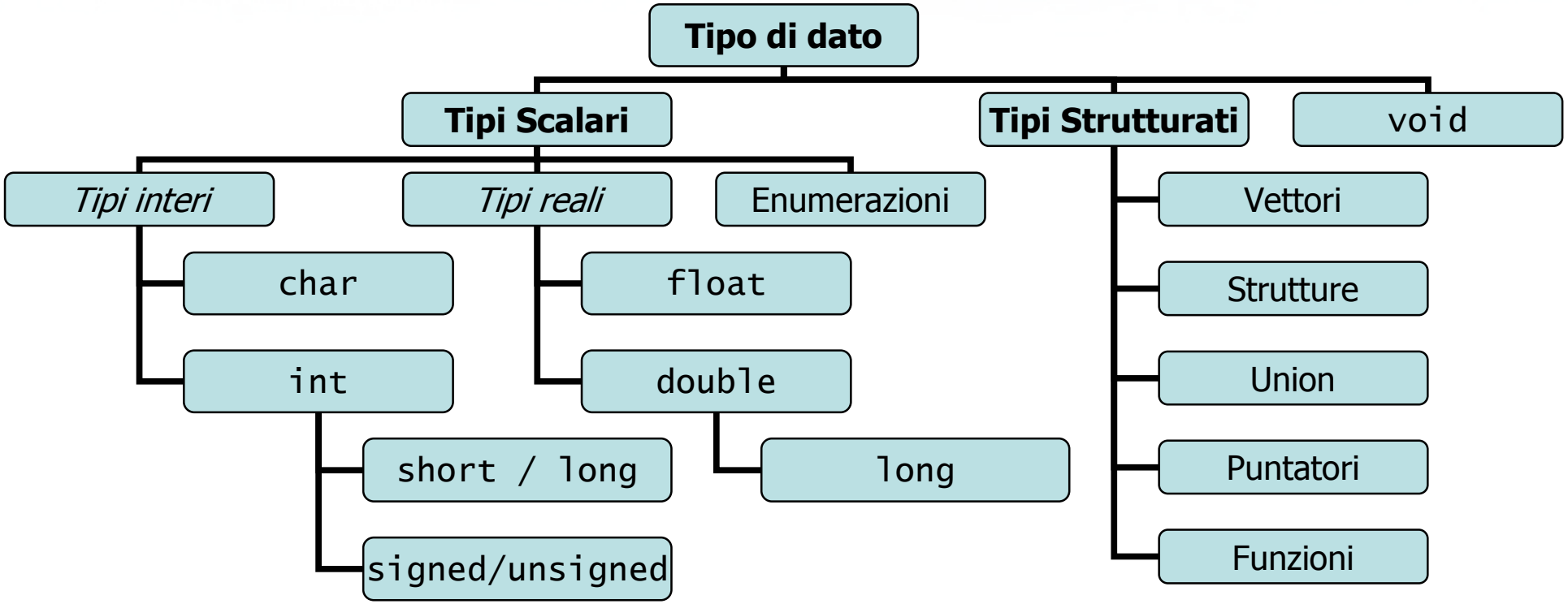
Sottoinsieme minimale di istruzioni

I tipi `int` e `float`

Tipi di dato

- Ogni costante, ogni variabile, ogni espressione appartiene ad un determinato tipo
- Il tipo determina
  - L'insieme dei valori che la costante, variabile o espressione può assumere
  - L'insieme delle operazioni lecite su tali valori
- I tipi possono essere
  - Semplici (o scalari): singoli valori
  - Strutturati: insiemi di più valori semplici

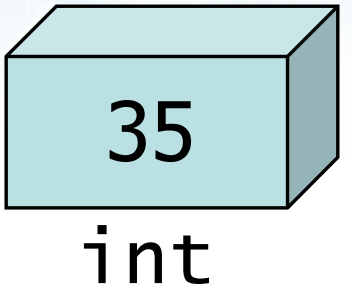
Il sistema dei tipi C



Caratteristiche generali

- I valori ammessi per ciascun tipo non sono fissati dallo standard
- Dipendono dal compilatore e dal sistema operativo
  - Ampiezza dei tipi di dato "naturale" per ogni calcolatore
- Maggior portabilità
- Maggior efficienza
- Nessuna garanzia di uniformità tra piattaforme diverse

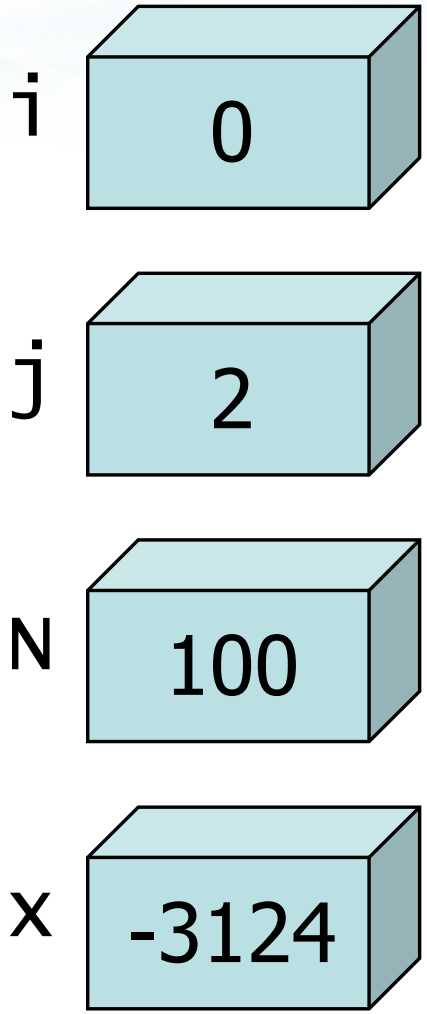
Il tipo `int`



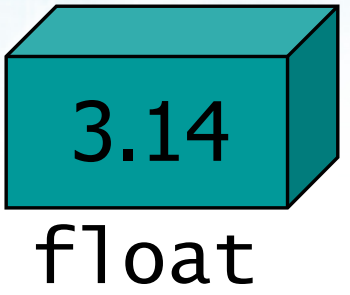
- Il tipo più importante del linguaggio C
- Valori interi, positivi o negativi
- Codificato in complemento a due
- Espresso solitamente su 16 bit oppure 32 bit
  - 16 bit: da –32 768 a +32 767
  - 32 bit: da –2 147 483 648 a +2 147 483 647
  - In generale: da `INT_MIN` a `INT_MAX`
    - `#include <limits.h>`

Esempi

```
int i, j ;
int N ;
int x ;

i = 0 ;
j = 2 ;
N = 100 ;
x = -3124 ;
```



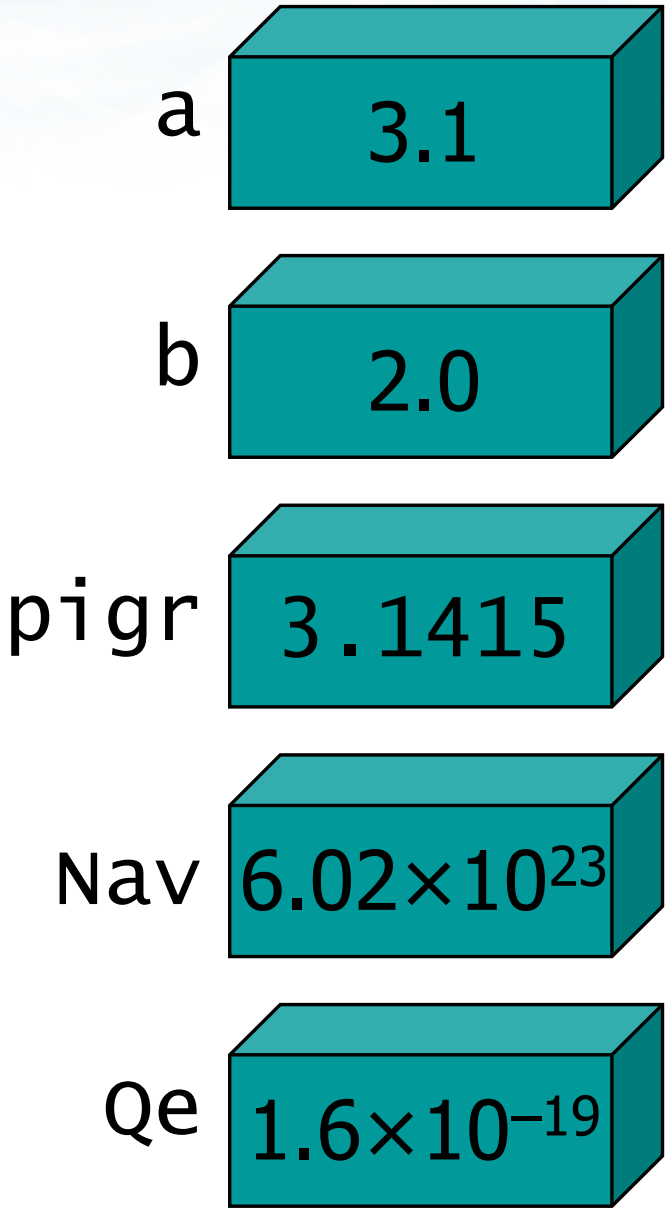
Il tipo `float`



- Valori reali
  - Frazionari
  - Esterni all'intervallo permesso per i numeri interi
- Codificato in virgola mobile, singola precisione
- Espresso solitamente su 32 bit
  - da $\pm 1.17 \times 10^{-38}$ a $\pm 3.40 \times 10^{+38}$
  - circa 6 cifre di precisione
  - In generale: da `FLT_MIN` a `FLT_MAX`
    - `#include <float.h>`

Esempi

```
float a, b ;
float pigr ;
float Nav, Qe ;

a = 3.1 ;
b = 2.0 ;
pigr = 3.1415926 ;
Nav = 6.022e23 ;
Qe = 1.6e-19 ;
```

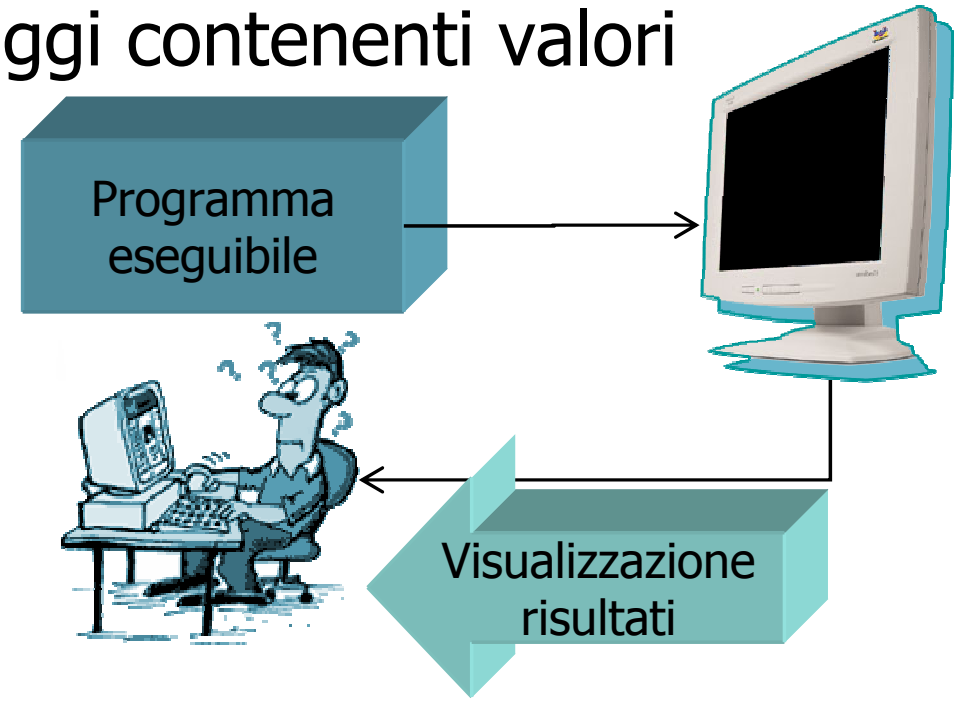


Sottoinsieme minimale di istruzioni

Istruzione `printf` – semplificata

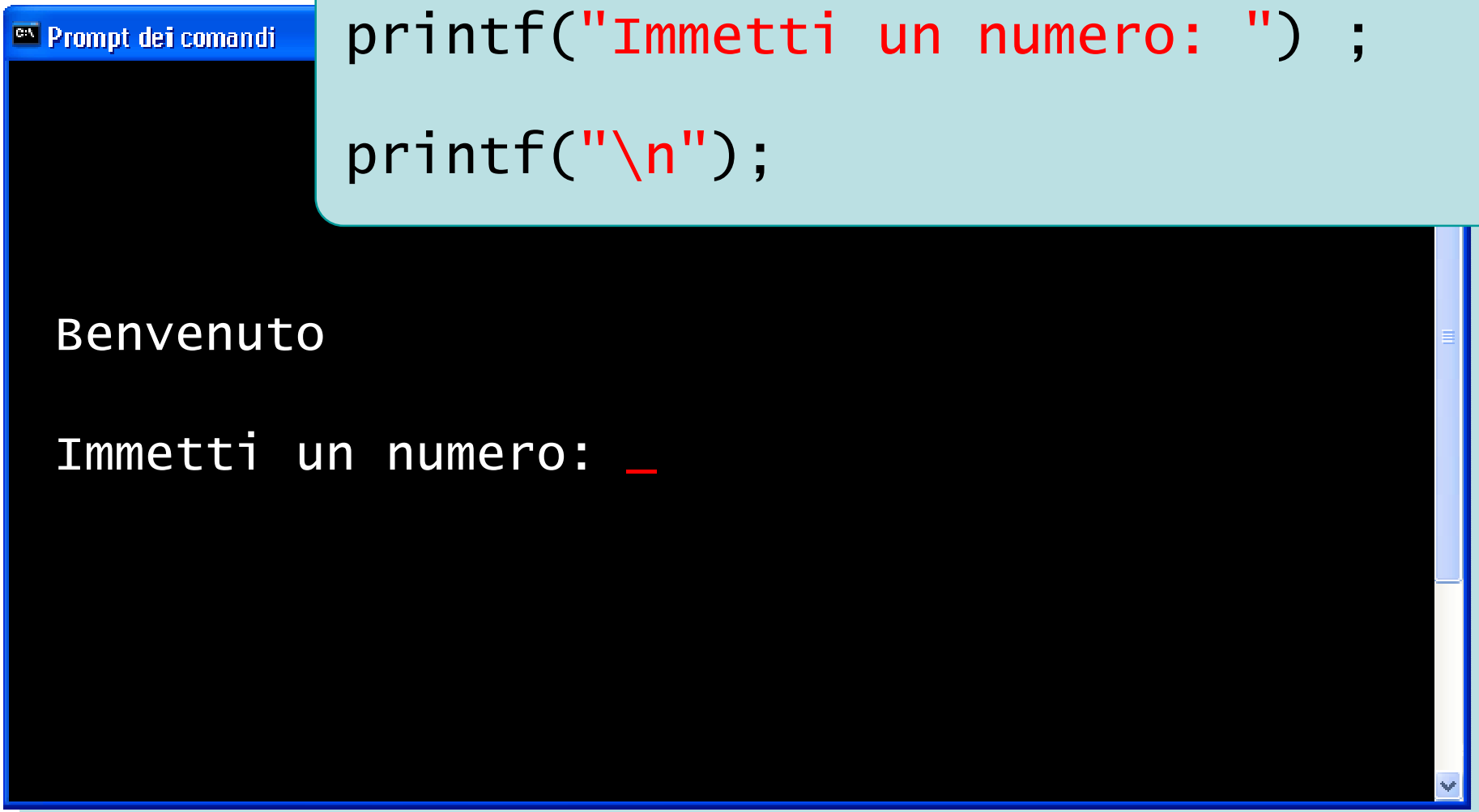
Istruzioni di stampa

- Stampa di messaggi informativi
- Stampa di comando "a capo"
- Stampa di valori di variabili
- Stampa di valori di espressioni calcolate
- Stampa di messaggi contenenti valori



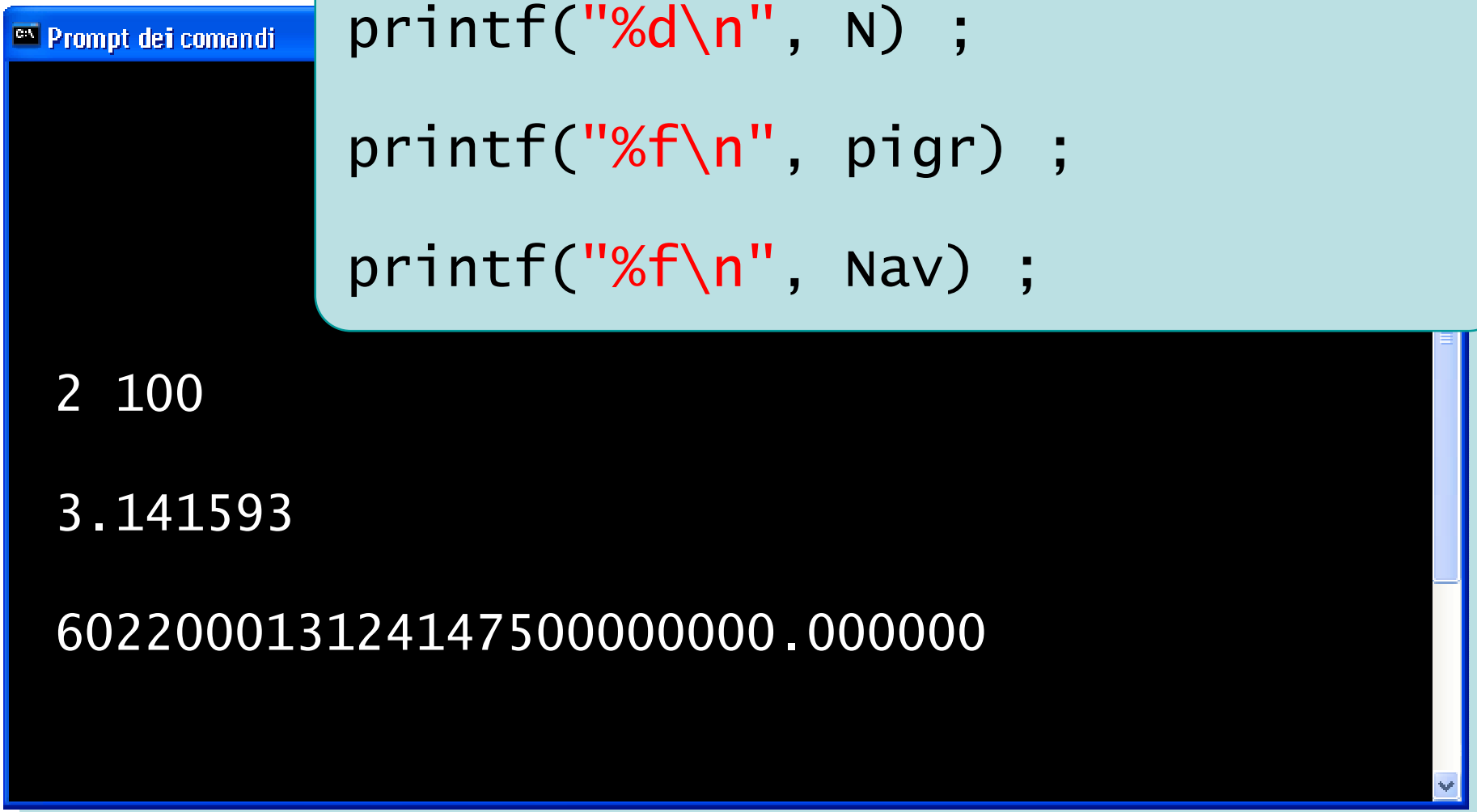
Stampa di messaggi

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

printf("Immetti un numero: ") ;

printf("\n");
```



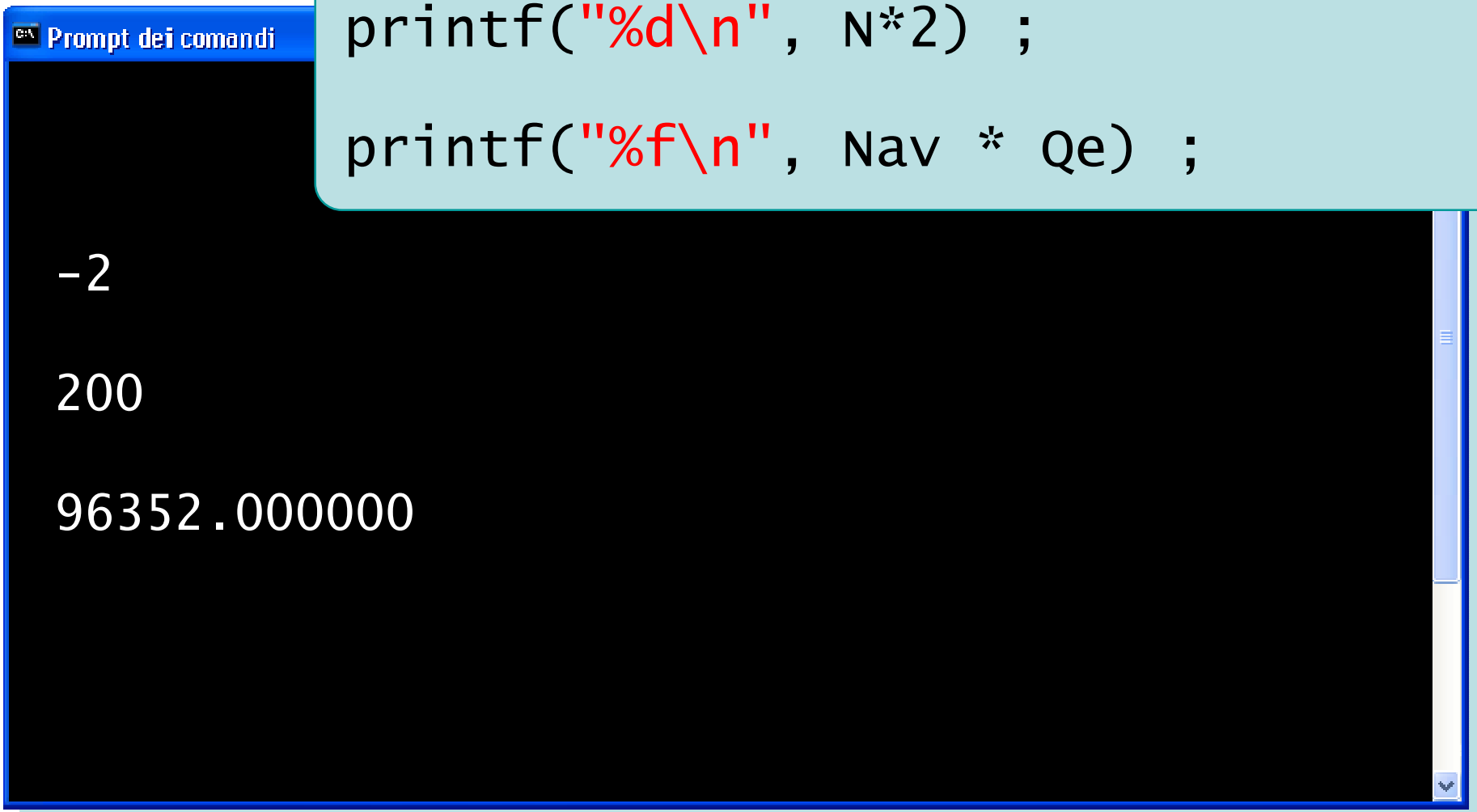
Stampa di variabili

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

printf("%d\n", N) ;

printf("%f\n", pigr) ;

printf("%f\n", Nav) ;
```



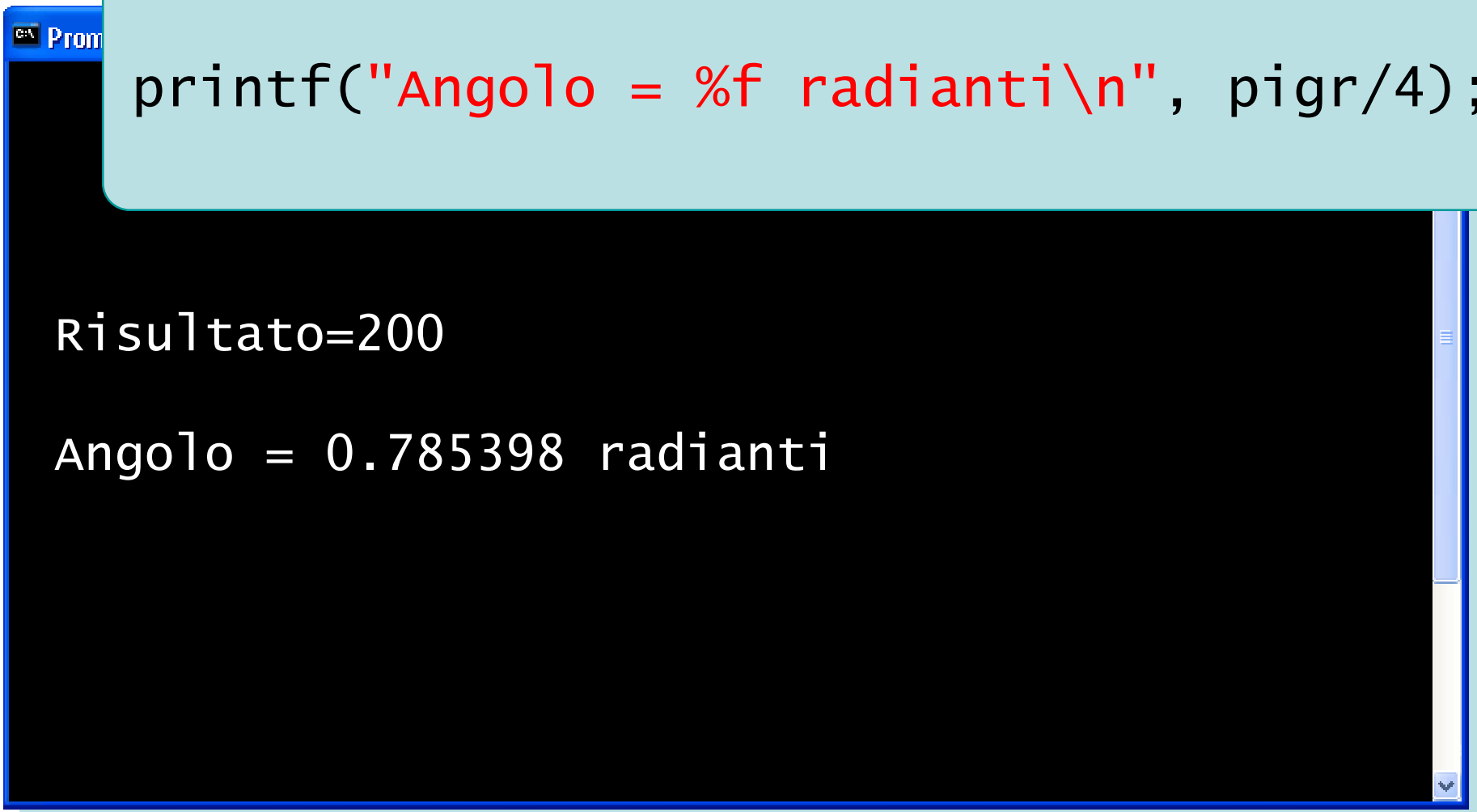
Stampa di espressioni

```
printf("%d\n", i-j) ;

printf("%d\n", N*2) ;

printf("%f\n", Nav * Qe) ;
```



Stampa di messaggi e valori

```
printf("Risultato=%d\n", N*2) ;
printf("Angolo = %f radianti\n", pigr/4);
```



Sintassi istruzione `printf`

- `#include <stdio.h>`
- `printf("`*formato*`", `*valore/i*`) ;`
- Formato:
  - Testo libero (compresi spazi) → viene stampato letteralmente
  - Simboli `\n` → va a capo
  - Simboli %d → stampa un `int`
  - Simboli %f → stampa un `float`
- Valore/i:
  - Variabile o espressione
  - Di tipo `int` o `float`, corrispondente al simbolo %

Casi particolari (1/2)

- Per stampare il simbolo % occorre ripeterlo due volte
  - `printf("Sondaggio: %f%%\n", pSI ) ;`
    - `%f` → stampa `pSI`
    - `%%` → stampa un simbolo %
    - `\n` → va a capo
  - `Sondaggio: 43.12%`

Casi particolari (2/2)

- È possibile stampare più di un valore nella stessa istruzione
  - `printf("Voti: %d su %d\n", voti, tot ) ;`
    - primo %d → stampa `voti`
    - secondo %d → stampa `tot`
  - `Voti: 18 su 45`

Sottoinsieme minimale di istruzioni

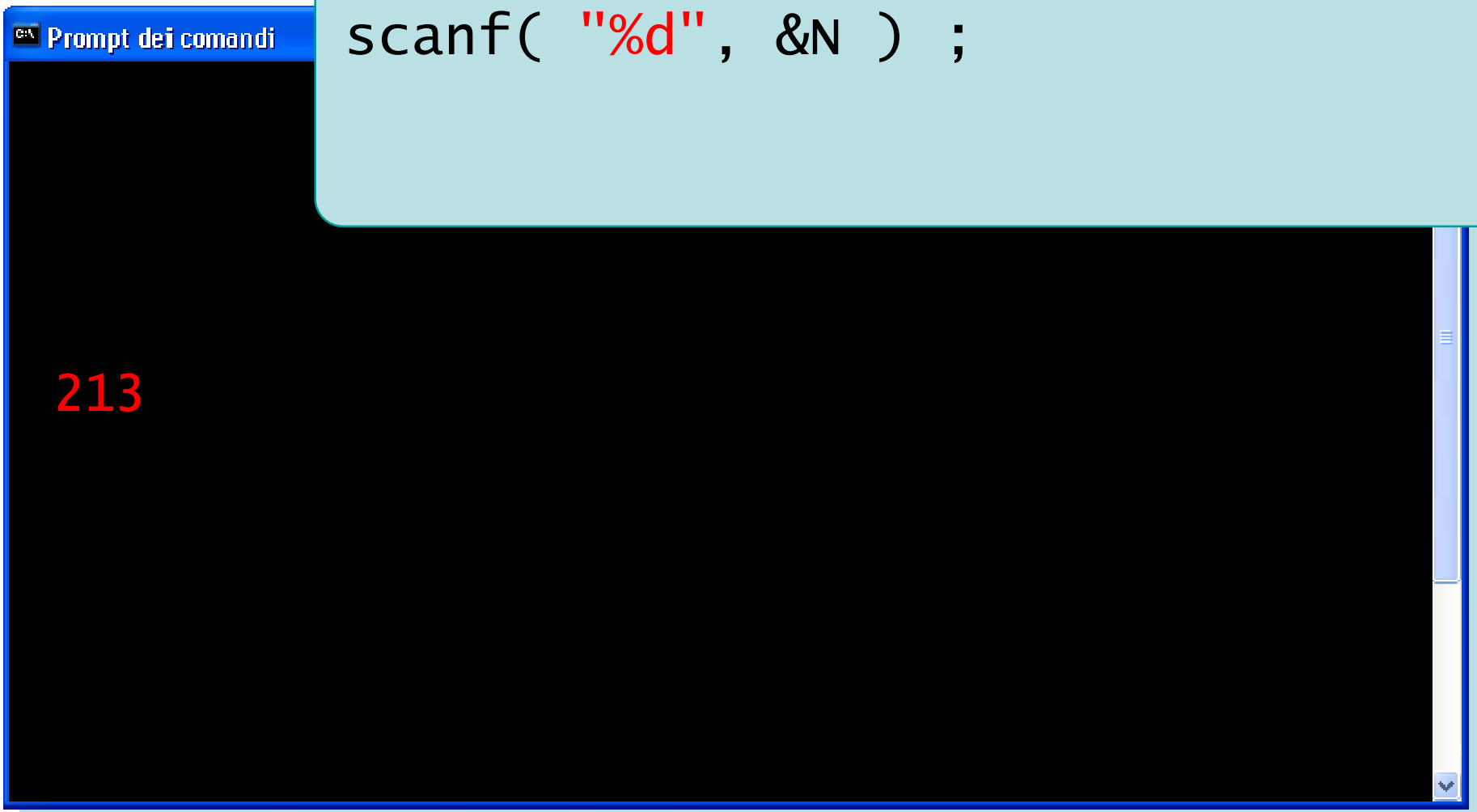
Istruzione scanf – semplificata

Istruzioni di lettura

- Lettura di un valore intero
- Lettura di un valore reale

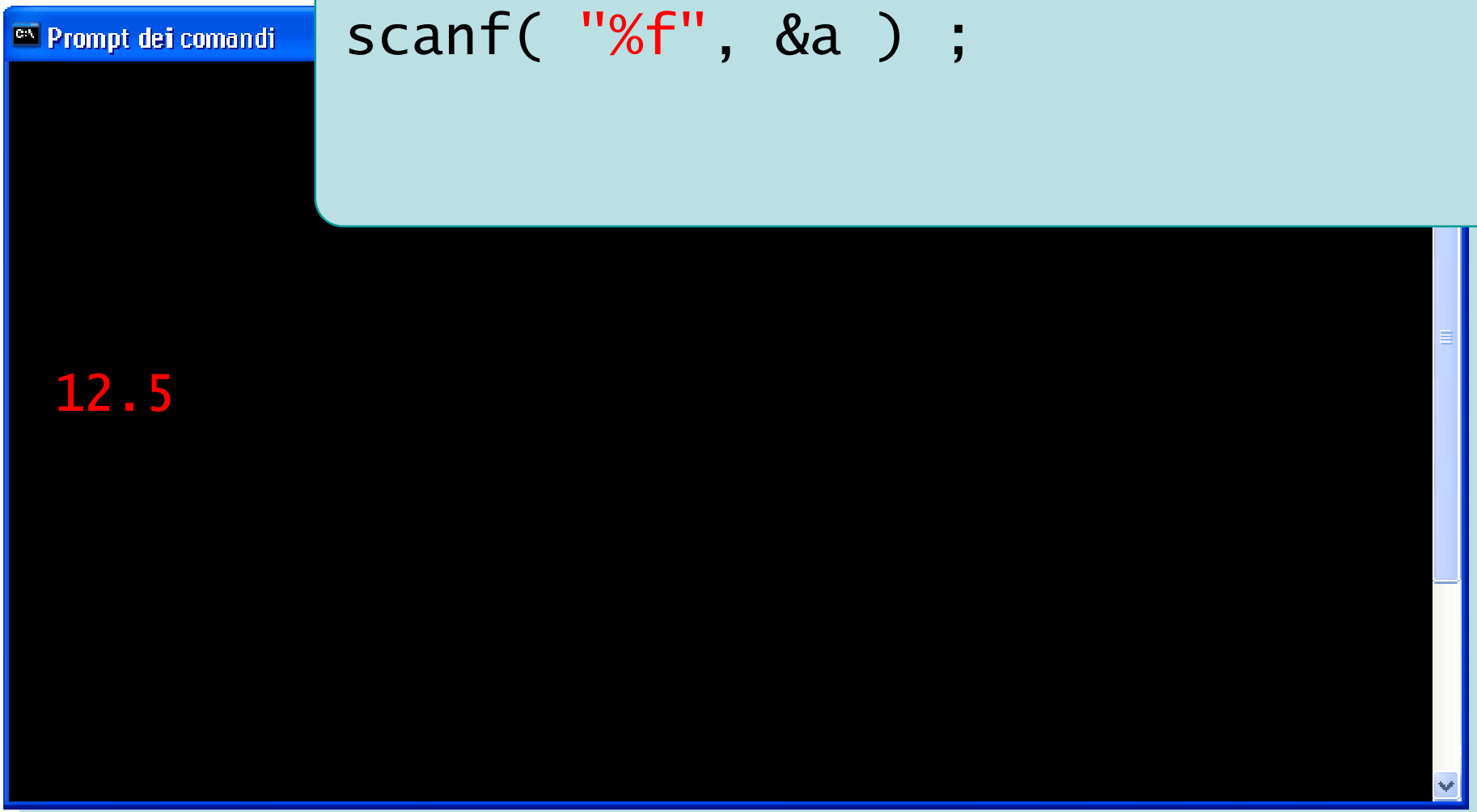
Lettura di un intero

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



Lettura di un reale

```
scanf( "%f", &a ) ;
```



Sintassi istruzione `scanf`

- `#include <stdio.h>`
- `scanf("`*formato*`", &`*variabile*`) ;`
- Formato:
  - Simboli %d → legge un `int`
  - Simboli %f → legge un `float`
- Variabile:
  - Di tipo `int` o `float`, corrispondente al simbolo %
  - Sempre preceduta dal simbolo &

Suggerimento

- Combinare le istruzioni `printf` e `scanf` per guidare l'utente nell'immissione dei dati
  - Ogni `scanf` deve essere preceduta da una `printf` che indica quale dato il programma si aspetta

```
printf("Immetti il numero: ");
scanf("%d", &N) ;
printf("Numero immesso: %d\n", N);
```

Errore frequente

- Dimenticare il simbolo & nelle istruzioni `scanf`

```
printf("Immetti il numero: ");
scanf("%d", N) ;
```

forma corretta

```
printf("Immetti il numero: ");
scanf("%d", &N) ;
```

Errore frequente

- Dimenticare le variabili da stampare nelle istruzioni `printf`

```
printf("Numero immesso: %d\n");
```

forma corretta

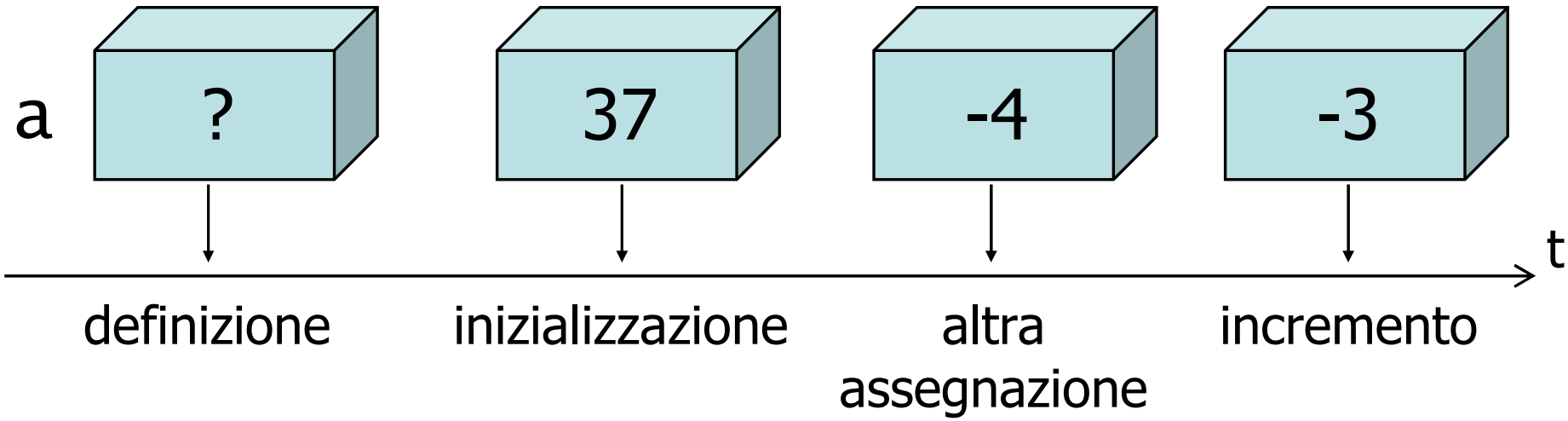
```
printf("Numero immesso: %d\n", N);
```

Sottoinsieme minimale di istruzioni

Istruzione di assegnazione

Assegnazione delle variabili

- Il valore di una variabile
  - Deve essere inizializzato, la prima volta
  - Può essere aggiornato, quante volte si vuole
- Per assegnare un nuovo valore ad una variabile si usa l'operatore =

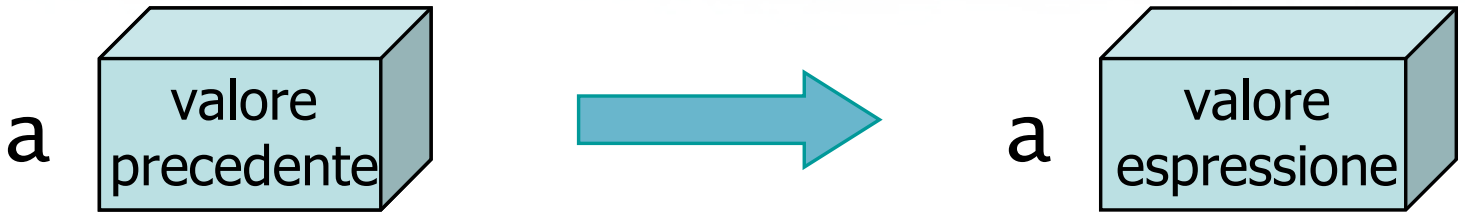


Assegnazione di variabili

- Assegnazione del valore di una costante
  - `i = 0 ;`
  - `a = 3.0 ;`
- Assegnazione del valore di un'altra variabile
  - `i = N ;`
  - `b = a ;`
- Assegnazione del valore di un'espressione
  - `j = N - i ;`
  - `b = a * 2 - 1 ;`

Sintassi dell'assegnazione

- *variabile* = *espressione* ;



- Passo 1: si valuta il valore corrente dell'espressione
  - Per tutte le variabili che compaiono nell'espressione, si usa il valore corrente
  - Può comparire anche la stessa variabile oggetto dell'assegnazione
- Passo 2: si memorizza tale valore nella variabile, cancellando il valore precedente

Esempi

- N = 3 ;

Esempi

- `N = 3 ;`
- `a = b ;`
  - Non confondere con `b = a ;`

Esempi

- `N = 3 ;`
- `a = b ;`
  - Non confondere con `b = a ;`
- `a = a + 1 ;`
  - Incrementa a di un'unità

Esempi

- `N = 3 ;`
- `a = b ;`
  - Non confondere con `b = a ;`
- `a = a + 1 ;`
  - Incrementa a di un'unità
- `a + 1 = a ;`
  - Errore

Quesito

- Che operazione svolge il seguente frammento di programma?

```
a = b ;
b = a ;
```

Risposta

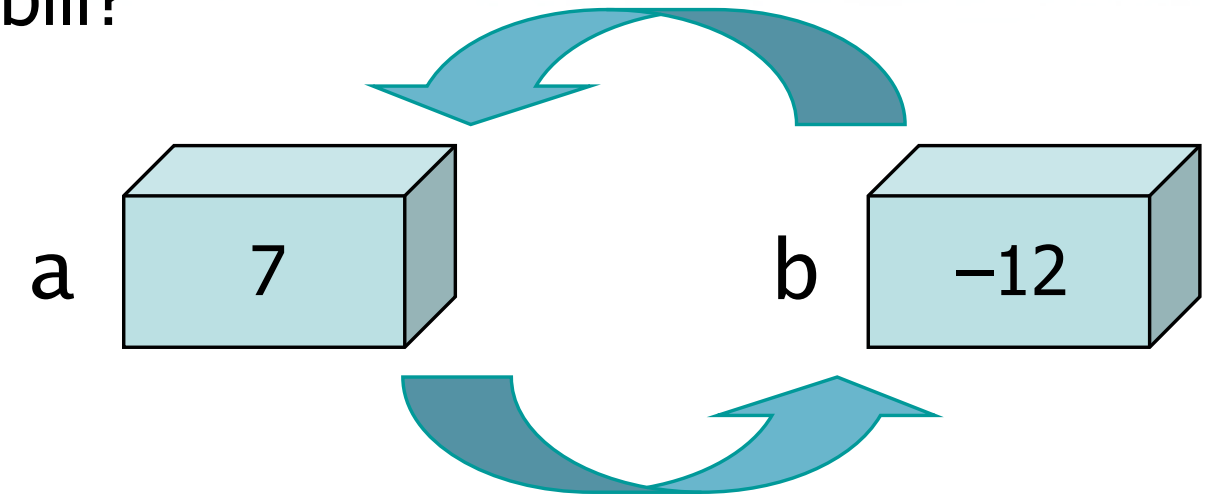
- Che operazione svolge il seguente frammento di programma?

```
a = b ;
b = a ;
```

- Il valore corrente di b viene copiato in a
  - Il valore vecchio di a viene perso
- Il (nuovo) valore corrente di a (uguale a b) viene ricopiato in b (operazione inutile)

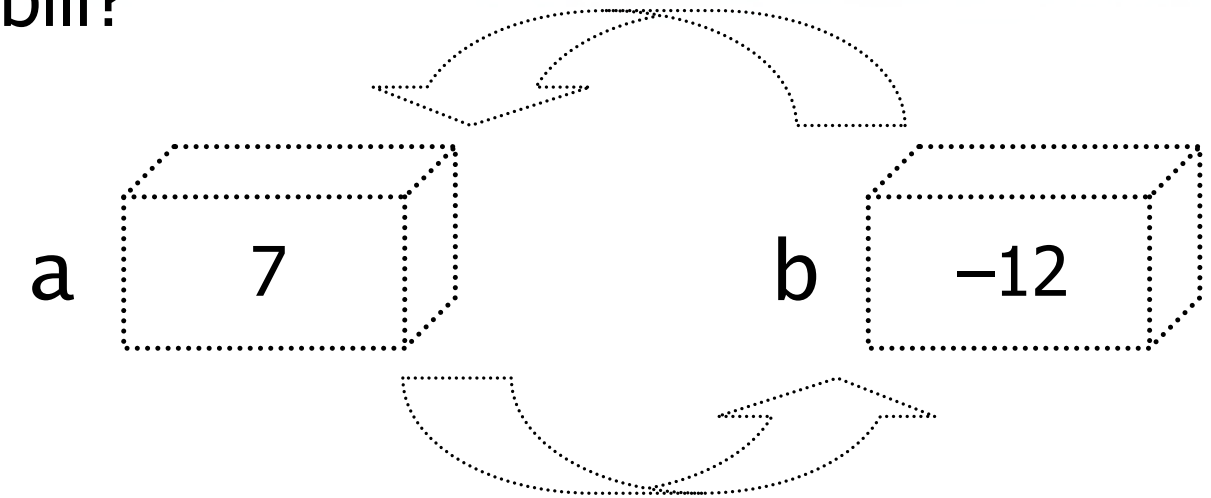
Quesito

- Come fare a scambiare tra di loro i valori di due variabili?



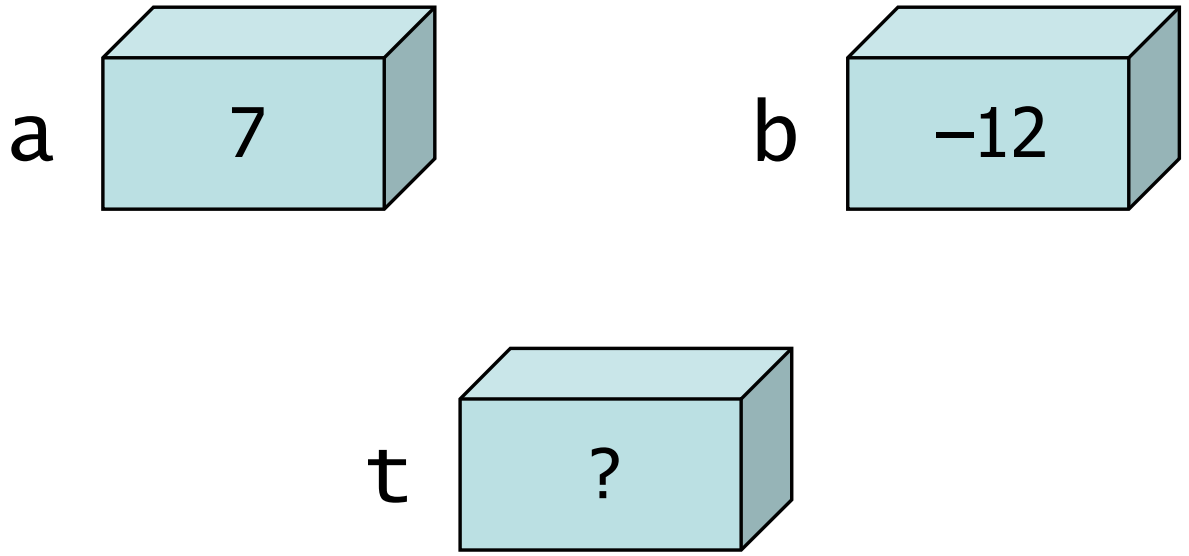
Quesito

- Come fare a scambiare tra di loro i valori di due variabili?



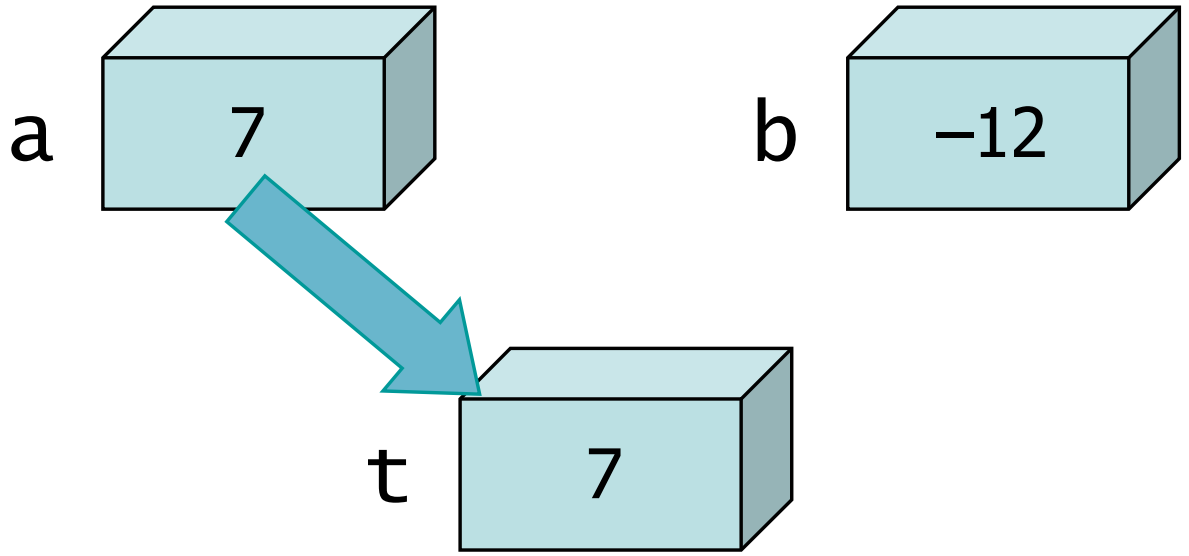
Risposta

```
t = a ;
a = b ;
b = t ;
```



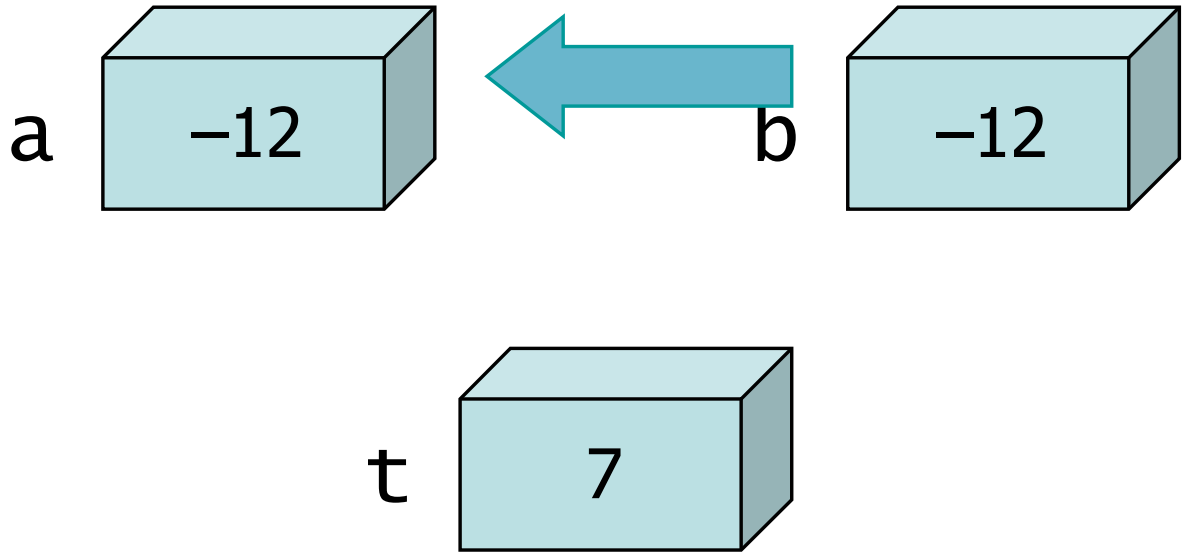
Risposta

```
t = a ;
a = b ;
b = t ;
```



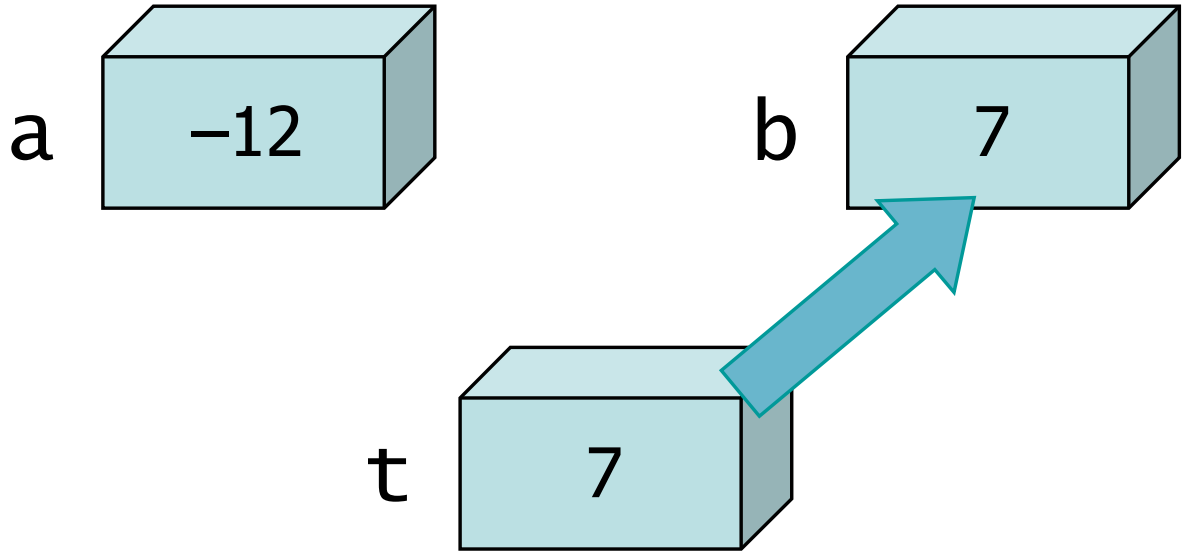
Risposta

```
t = a ;
a = b ;
b = t ;
```



Risposta

```
t = a ;
a = b ;
b = t ;
```



Sottoinsieme minimale di istruzioni

Semplici espressioni aritmetiche

Espressioni aritmetiche

- Ovunque sia richiesto il valore di una variabile, è possibile usare un'espressione aritmetica
  - Nei valori da stampare con la funzione `printf`
  - Nei valori da assegnare ad una variabile
- Le espressioni si possono costruire ricorrendo a:
  - Operatori: `+ - * /`
  - Parentesi: `( ... )`
  - Funzioni di libreria: `sqrt()`, `sin()`, `cos()`, ...

Operatori principali

- Somma: `a+b`
- Sottrazione: `a-b`
- Moltiplicazione: `a*b`
- Divisione: `a/b`
  - Divisione intera (risultato troncato) se entrambi gli operandi sono `int`
- Resto della divisione: `a%b`
  - Solo tra operandi `int`
- Cambio di segno: `-a`

Alcuni operatori avanzati

- Incremento: `i++`
- Decremento: `N--`
- Conversione ad intero: `(int)a`
- Conversione a reale: `(float)N`

Funzioni di libreria

- `#include <math.h>`
- Funzioni algebriche
  - `fabs`, `sqrt`, `cbrt`, `pow`, `hypot`, `ceil`, `floor`, `round`, `trunc`, `fmod`
- Funzioni esponenziali e logaritmiche
  - `exp`, `exp2`, `log`, `log10`, `log2`
- Funzioni trigonometriche e iperboliche
  - `cos`, `sin`, `tan`, `cosh`, `sinh`, `tanh`
- Funzioni trigonometriche e iperboliche inverse
  - `acos`, `asin`, `atan`, `atan2`, `acosh`, `asinh`, `atanh`

Parentesi

- Si possono costruire espressioni complicate a piacere utilizzando le parentesi
- Per maggiore leggibilità, abbondare con le parentesi ed usare la spaziatura e l'incolonnamento in modo ordinato
- Si utilizzano sempre le parentesi tonde

```
x1 = ( -b + sqrt( b*b - 4*a*c ) ) /
     ( 2*a ) ;
```

```
A = sqrt( p * (p-a) * (p-b) * (p-c)) ;
```

Primo programma in C

Compilare il primo programma

Compilare il primo programma

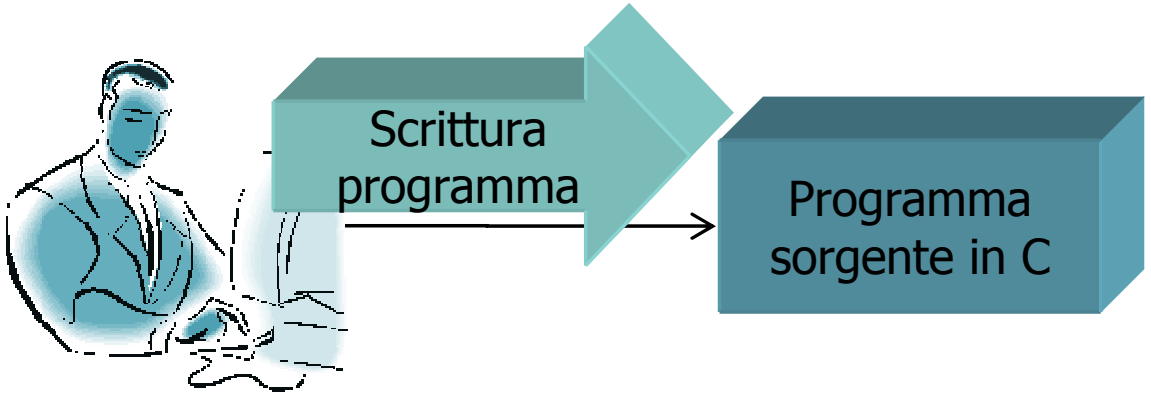
- Un semplice programma
- L'ambiente di sviluppo Dev-C++
- Codifica del programma
- Compilazione e correzione errori
- Esecuzione e verifica

Compilare il primo programma

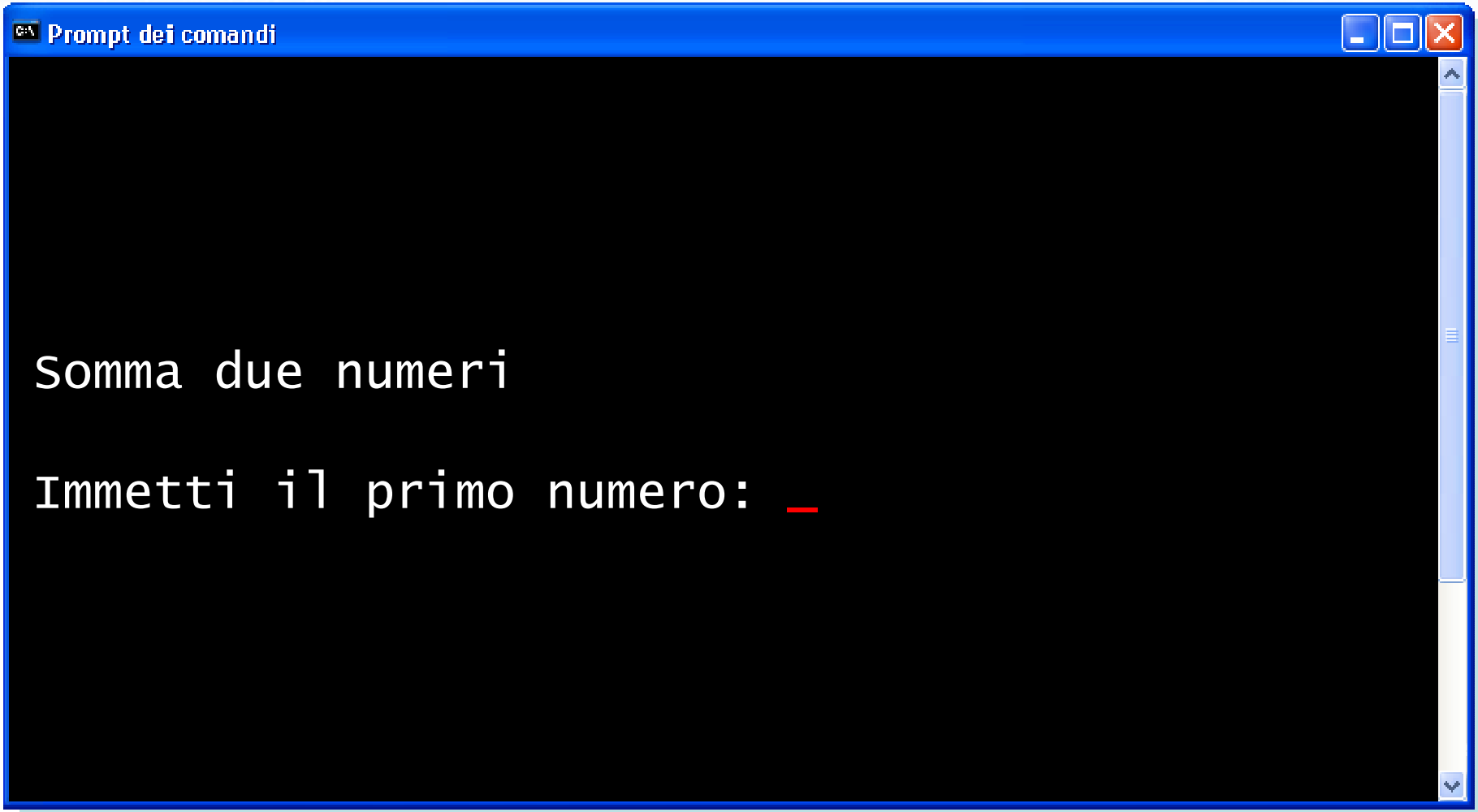
Un semplice programma

Esercizio "Somma due numeri"

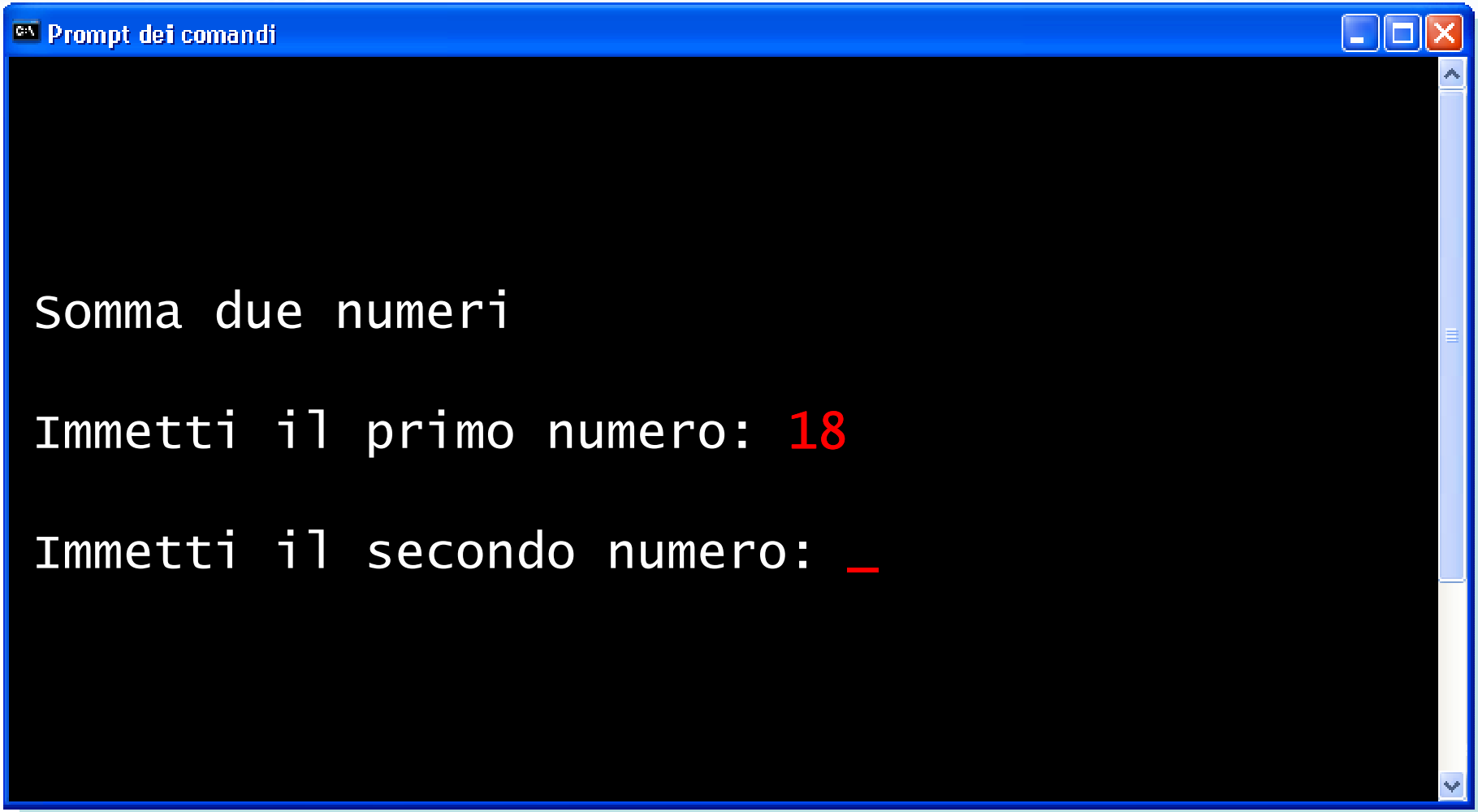
- Si realizzi un programma in linguaggio C che acquisisca da tastiera due numeri interi (detti A e B) e che stampi a video il valore della somma di tali numeri



Analisi


Prompt dei comandi
Somma due numeri
Immetti il primo numero: _

Analisi


Prompt dei comandi
Somma due numeri
Immetti il primo numero: 18
Immetti il secondo numero: _

Analisi

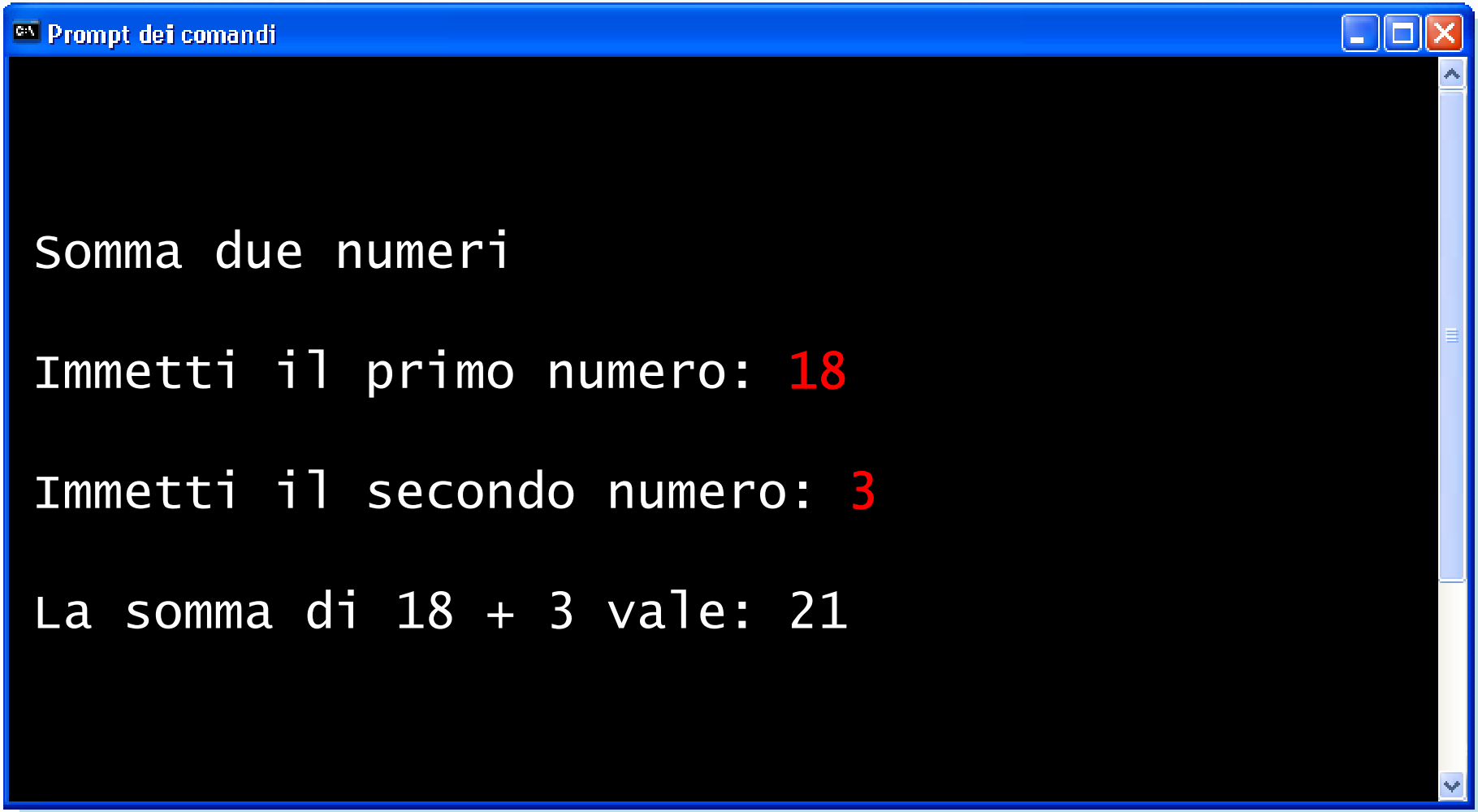
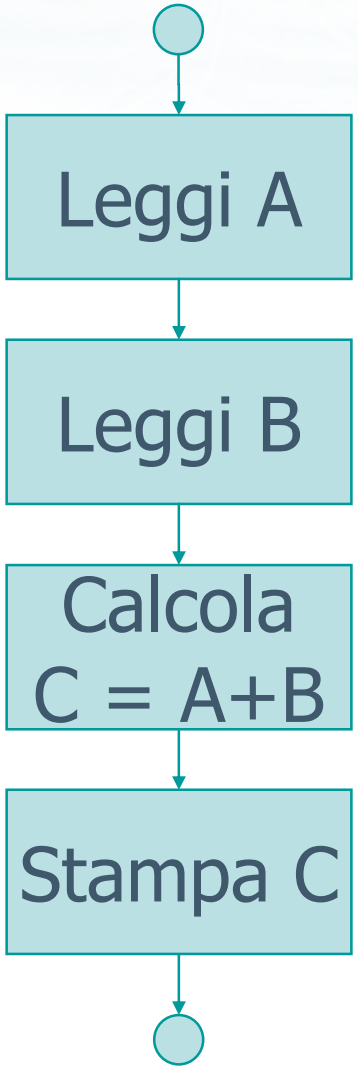
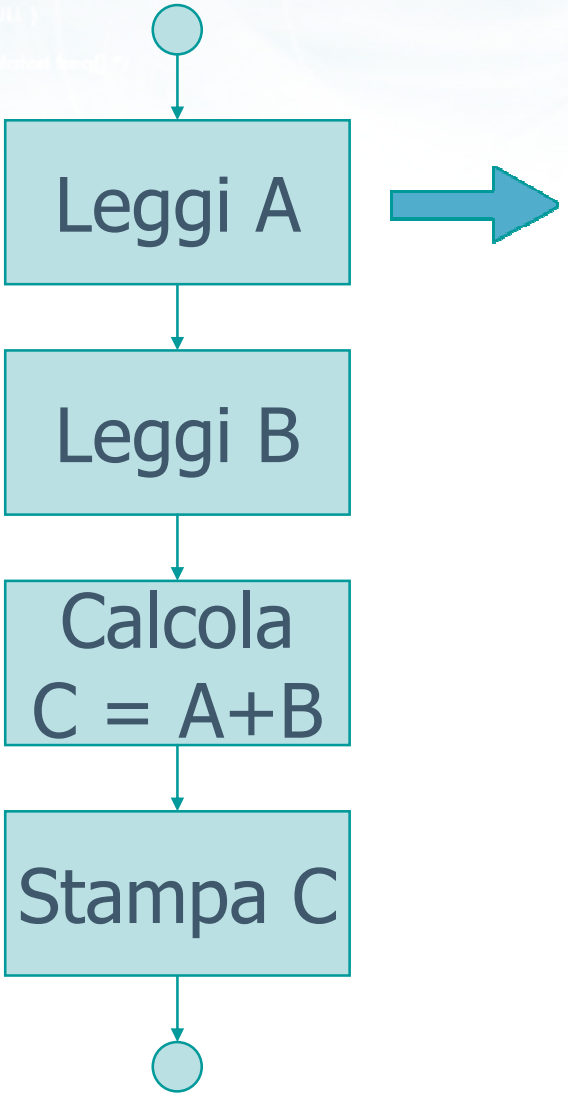

Prompt dei comandi
Somma due numeri
Immetti il primo numero: 18
Immetti il secondo numero: 3
La somma di 18 + 3 vale: 21

Diagramma di flusso

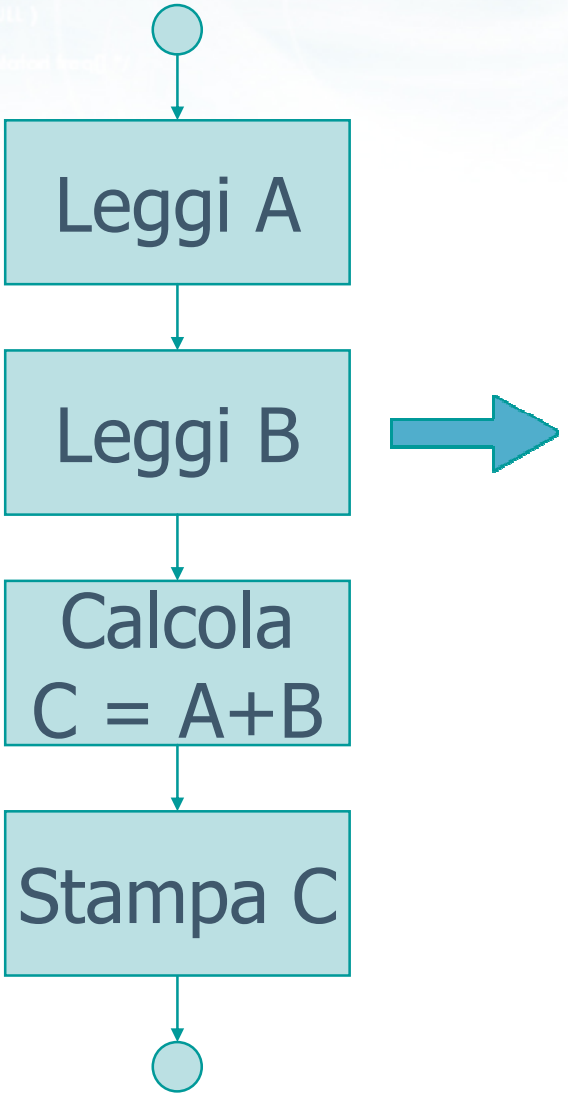


Traduzione in C (1/4)



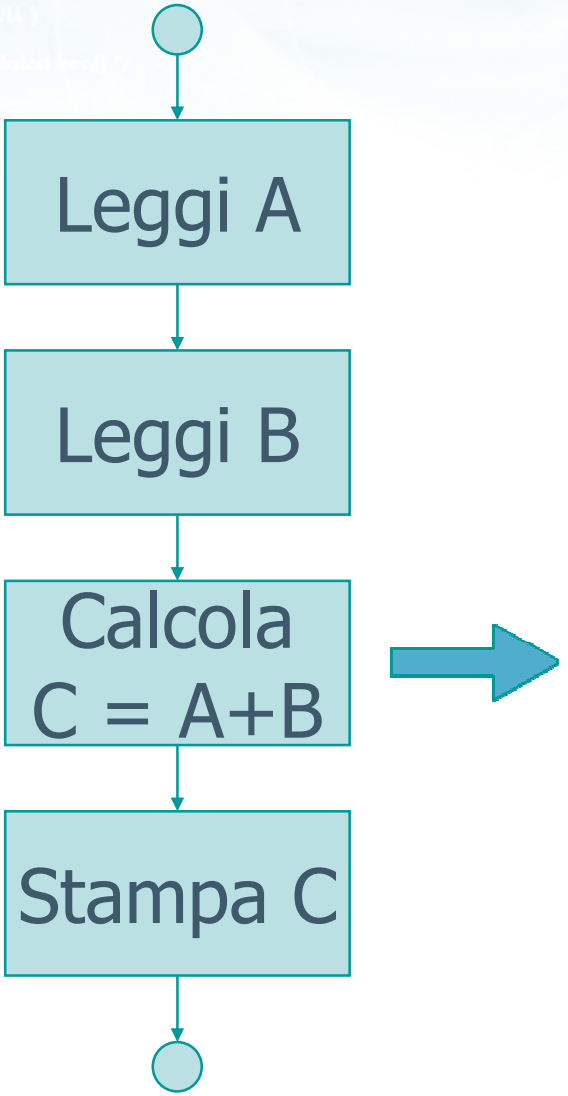

```
int a ;

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

Traduzione in C (2/4)



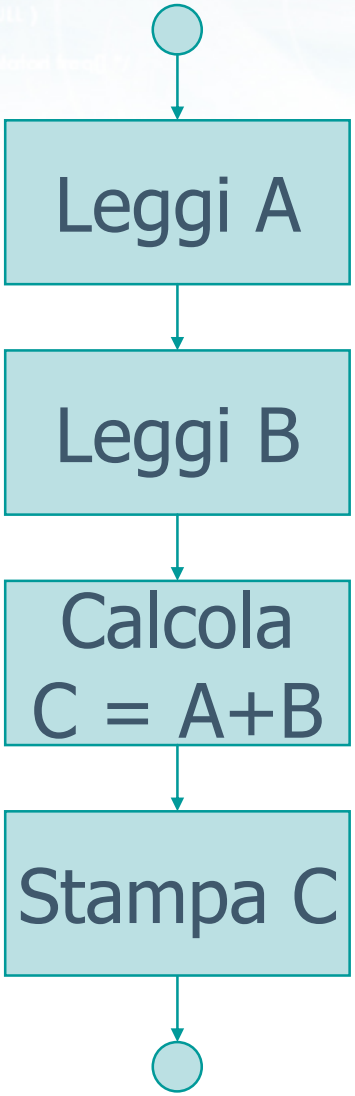

```
int b ;

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

Traduzione in C (3/4)




```
int c ;

c = a + b ;
```

Traduzione in C (4/4)




```
printf("La somma %d + %d ",
        a, b) ;

printf("vale: %d\n", c) ;
```

Compilare il primo programma

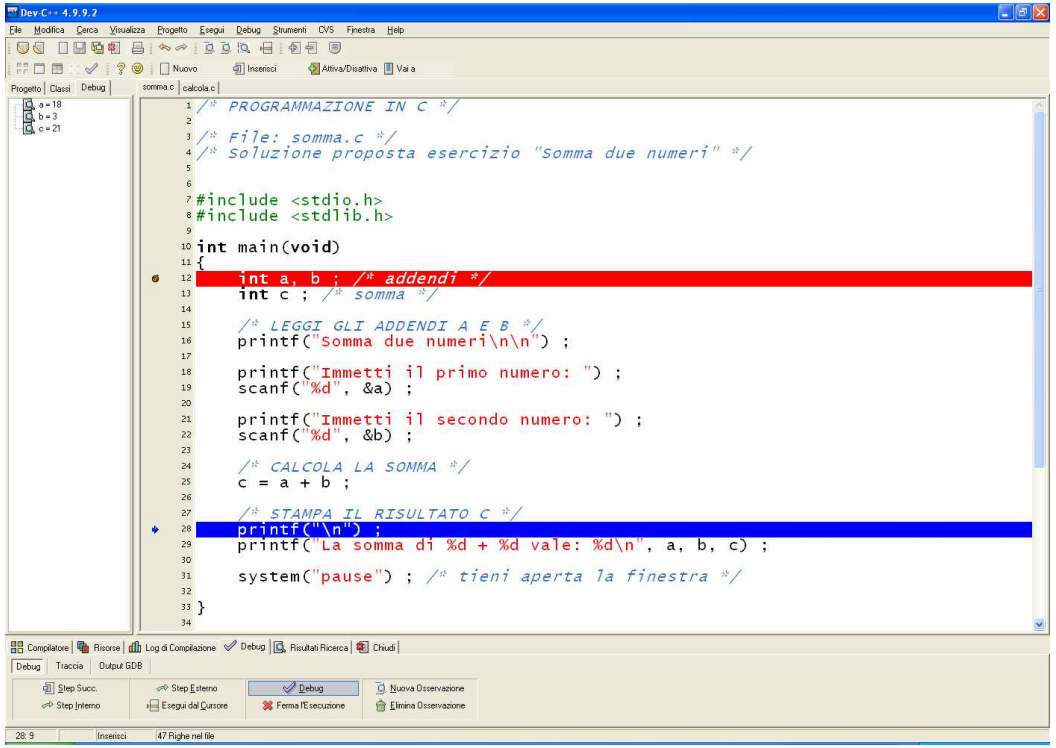
L'ambiente di sviluppo Dev-C++

Compilatori e IDE

- Occorre identificare ed installare
  - Un editor (possibilmente per programmatori)
  - Un compilatore
  - Un debugger
- Oppure trovare un Integrated Development Environment che integri tutte le funzionalità precedenti
- Esistono molte soluzioni gratuite

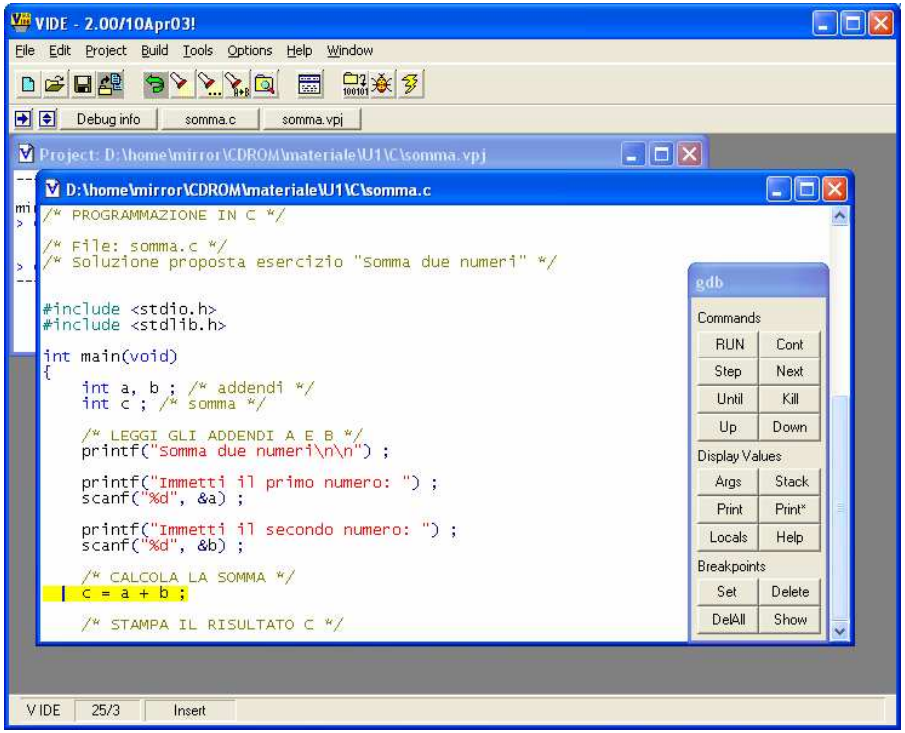
IDE per C in ambiente Windows

- Dev-C++
  - http://www.bloodshed.net



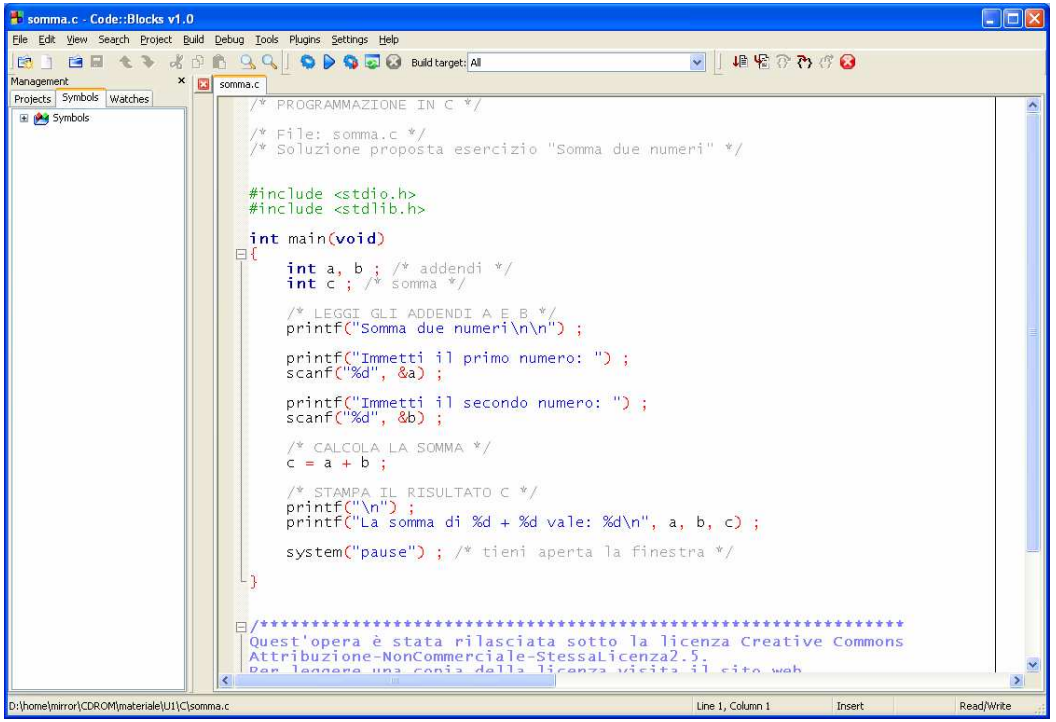
IDE per C in ambiente Windows

- V IDE
  - http://www.objectcentral.org



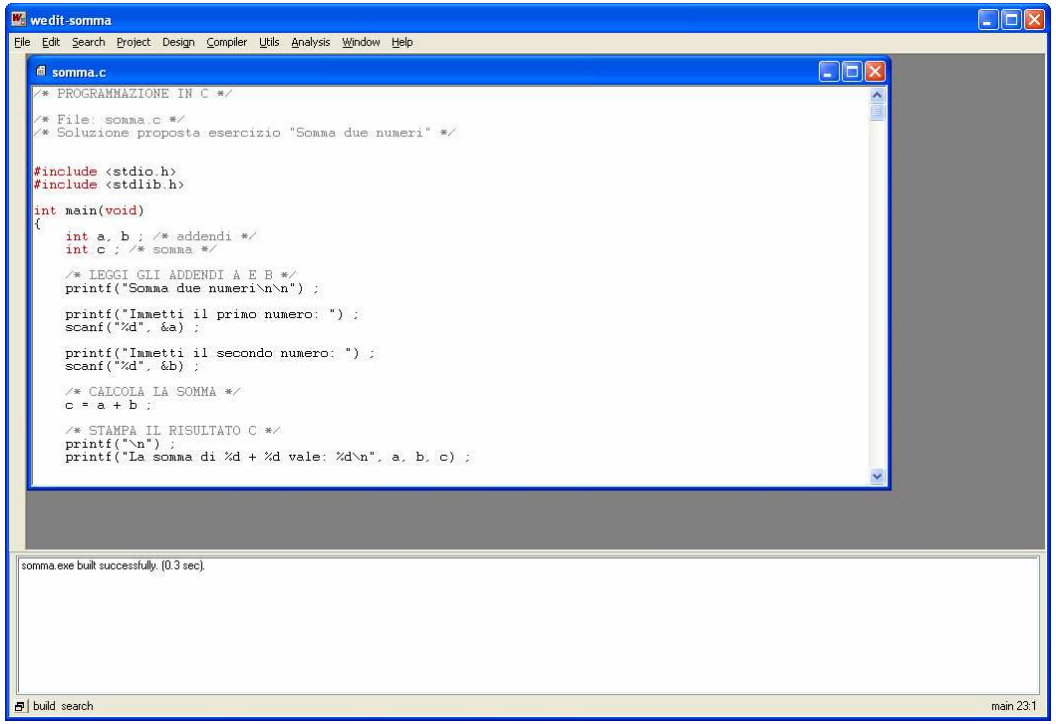
IDE per C in ambiente Windows

- Code::Blocks
  - http://www.codeblocks.org

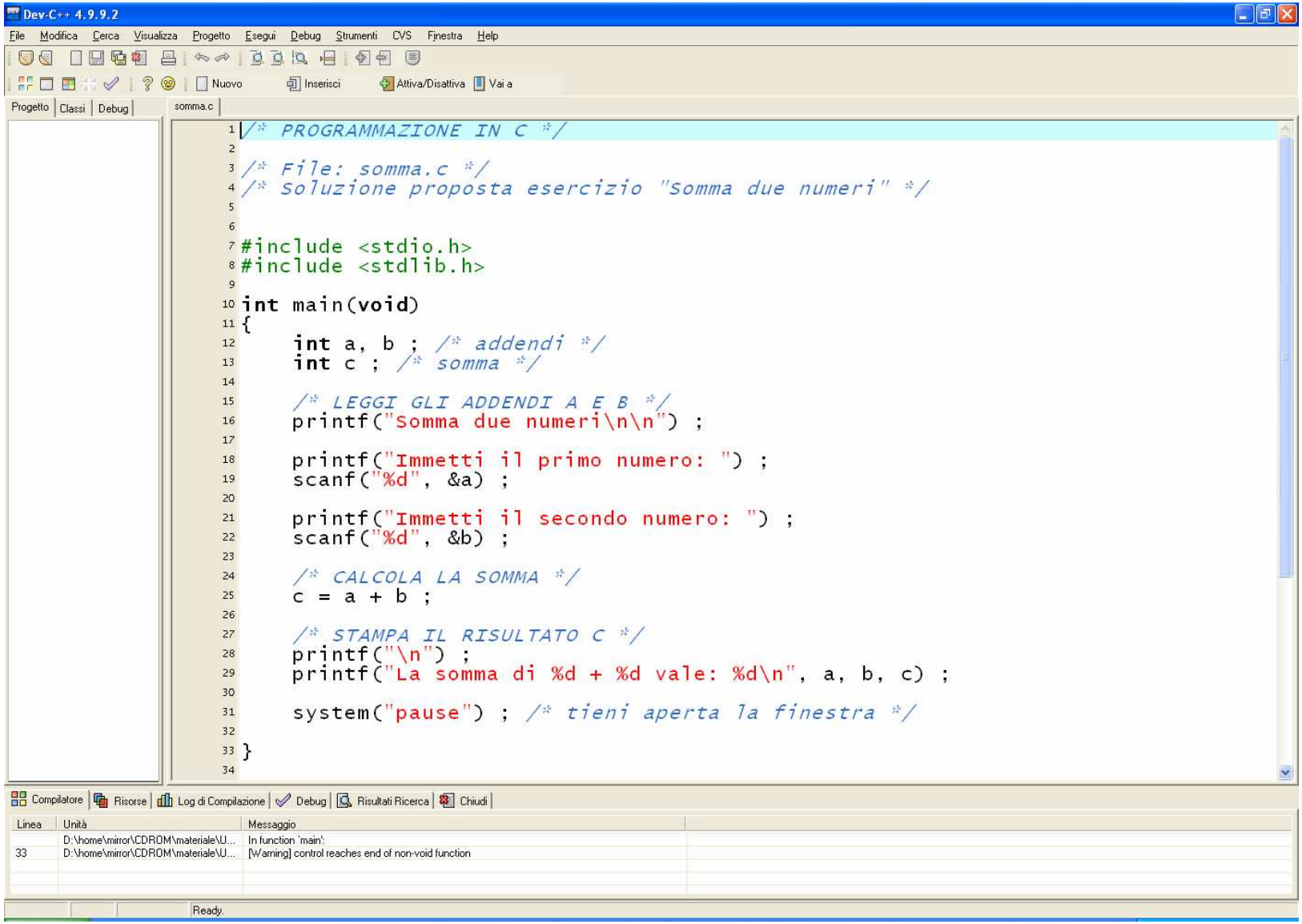


IDE per C in ambiente Windows

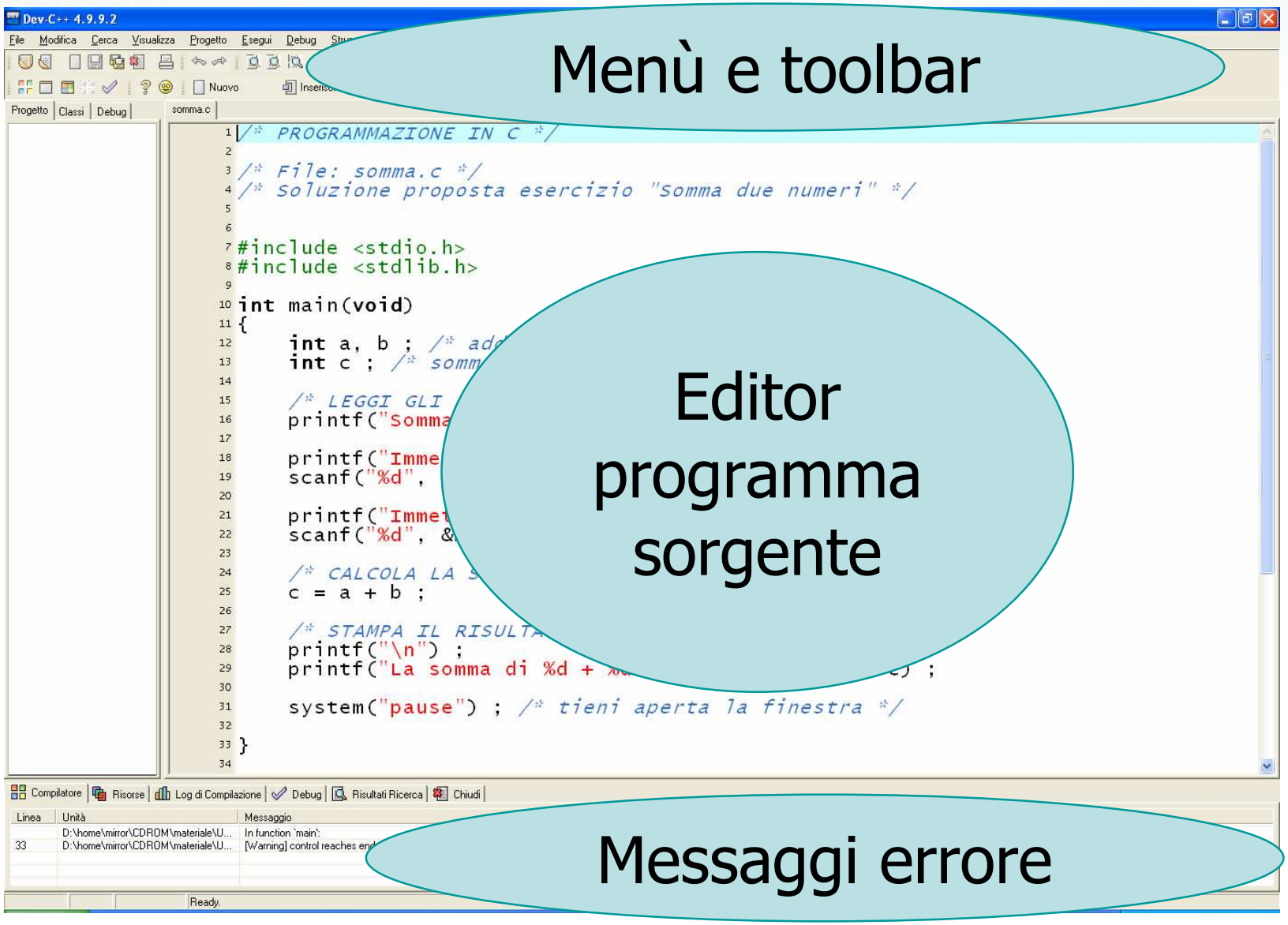
- lcc-win32
  - http://www.cs.virginia.edu/~lcc-win32/



Interfaccia di Dev-C++



Interfaccia di Dev-C++

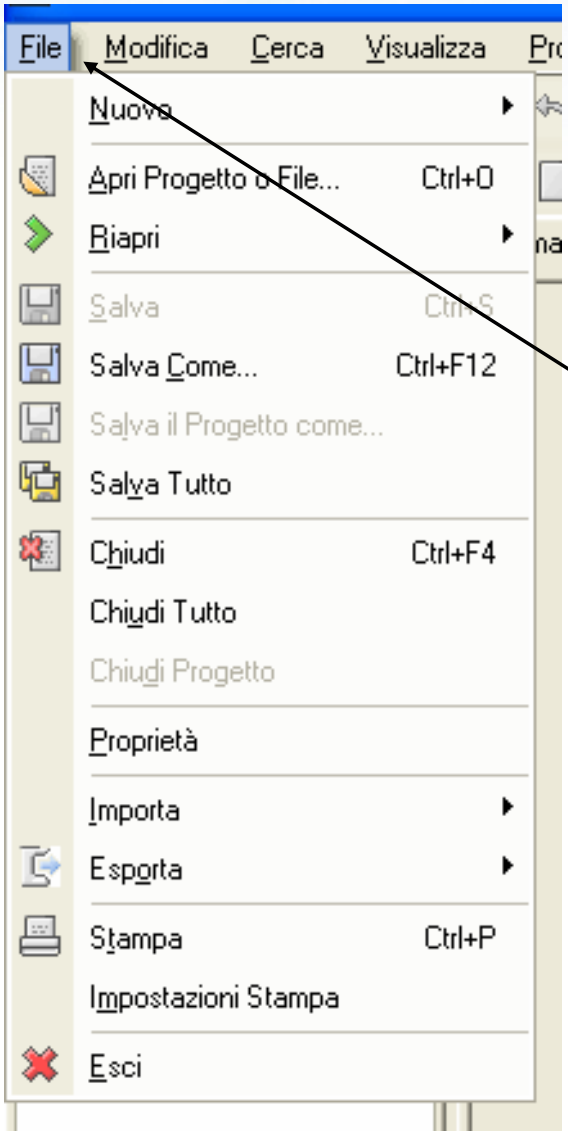


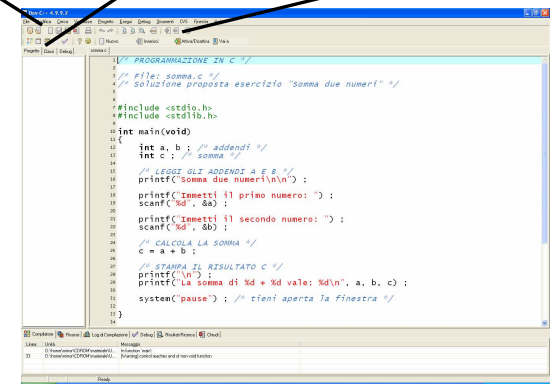
Menù e toolbar

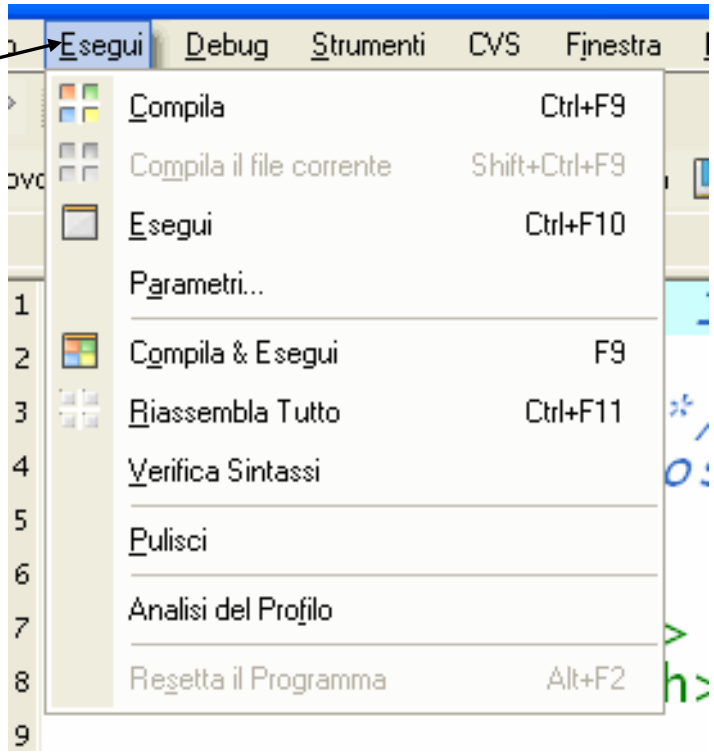
Editor programma sorgente

Messaggi errore

Menu principali





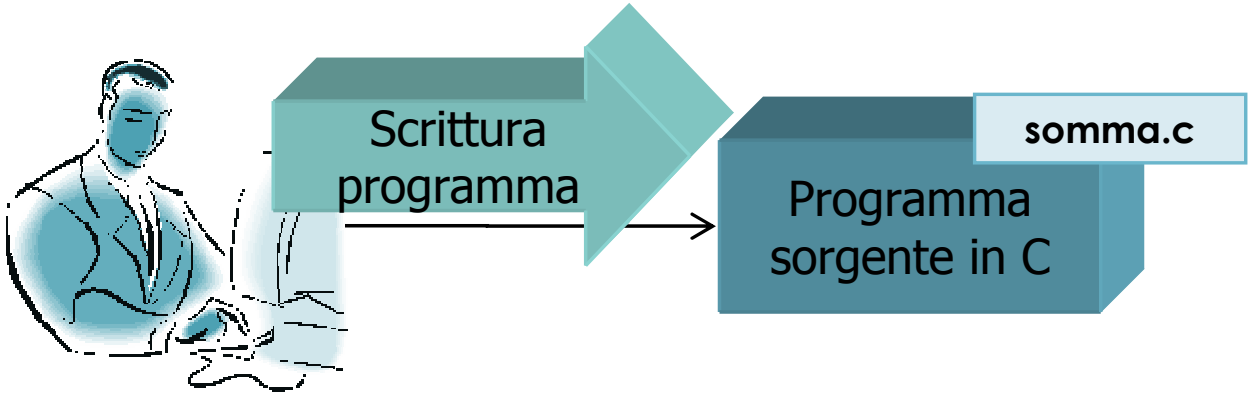


Compilare il primo programma

Codifica del programma

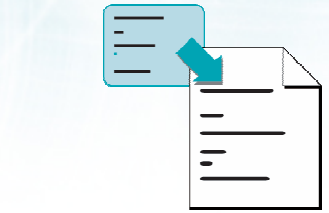
Codifica del programma

- A partire dal diagramma di flusso
- Utilizziamo un editor per immettere le istruzioni C
- Creiamo un file sorgente `somma.c`

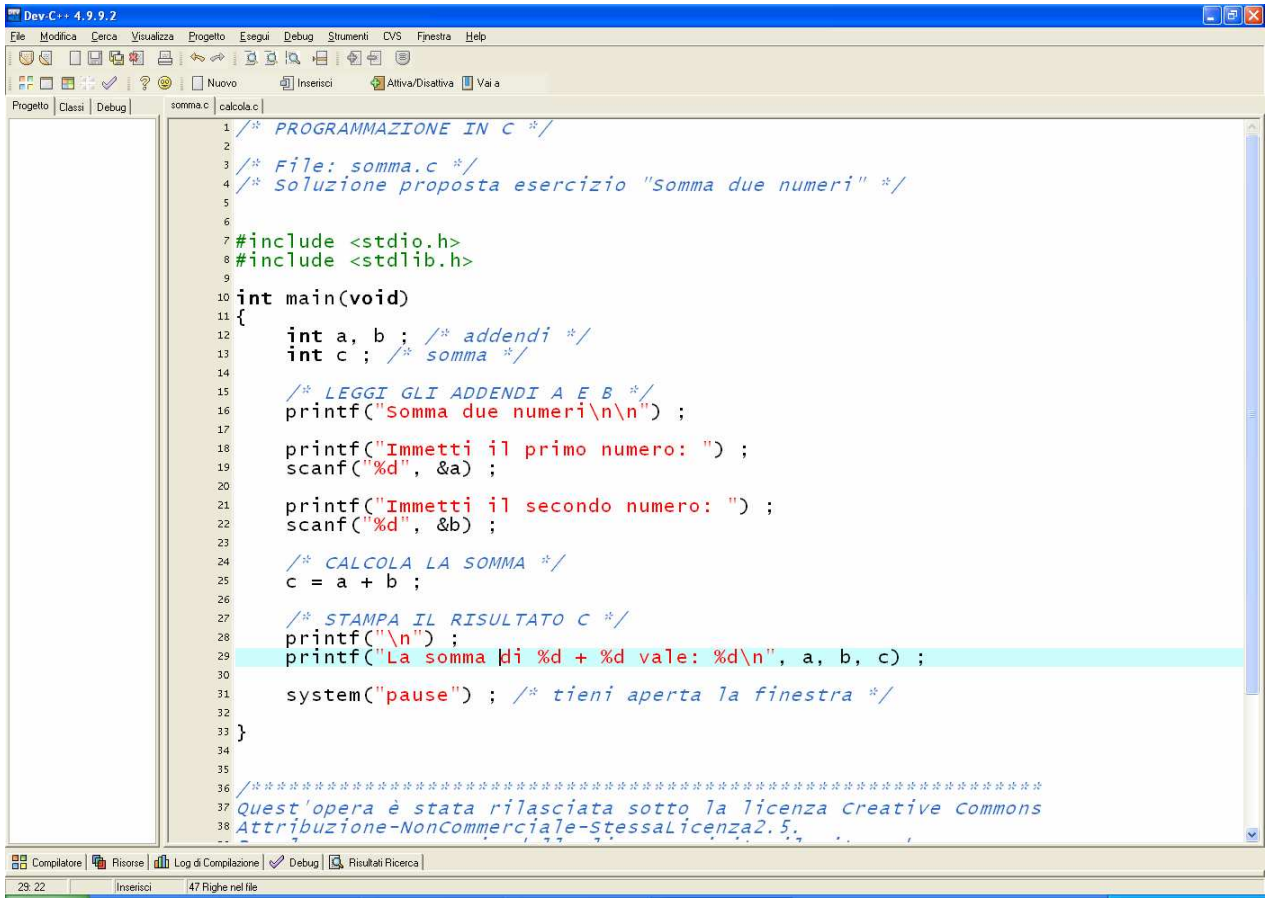


Codifica "Somma due numeri"

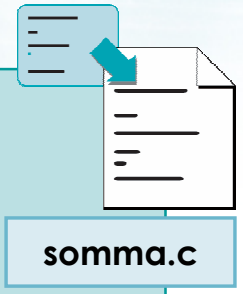
- Codifichiamo il programma in Dev-C++



somma.c

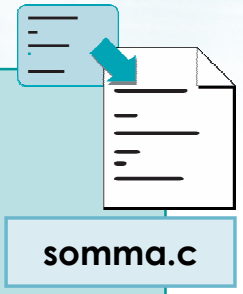


Soluzione proposta (1/2)




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

int main(void)
{
    int a, b ; /* addendi */
    int c ; /* somma */

    /* LEGGI GLI ADDENDI A E B */
    printf("Somma due numeri\n\n") ;

    printf("Immetti il primo numero: ") ;
    scanf("%d", &a) ;
    printf("Immetti il secondo numero: ") ;
    scanf("%d", &b) ;
```

Soluzione proposta (2/2)



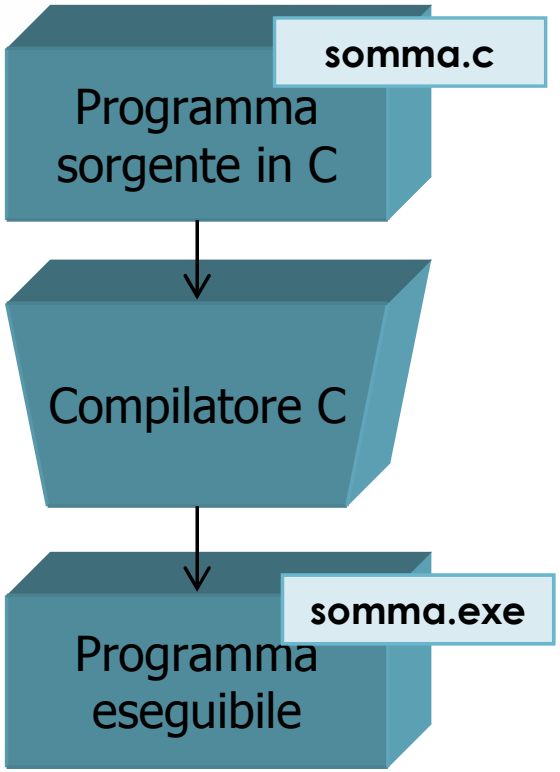

```
    /* CALCOLA LA SOMMA */
    c = a + b ;

    /* STAMPA IL RISULTATO C */
    printf("La somma di %d + %d vale: %d\n",
           a, b, c) ;
}
```

Compilare il primo programma

Compilazione e correzione errori

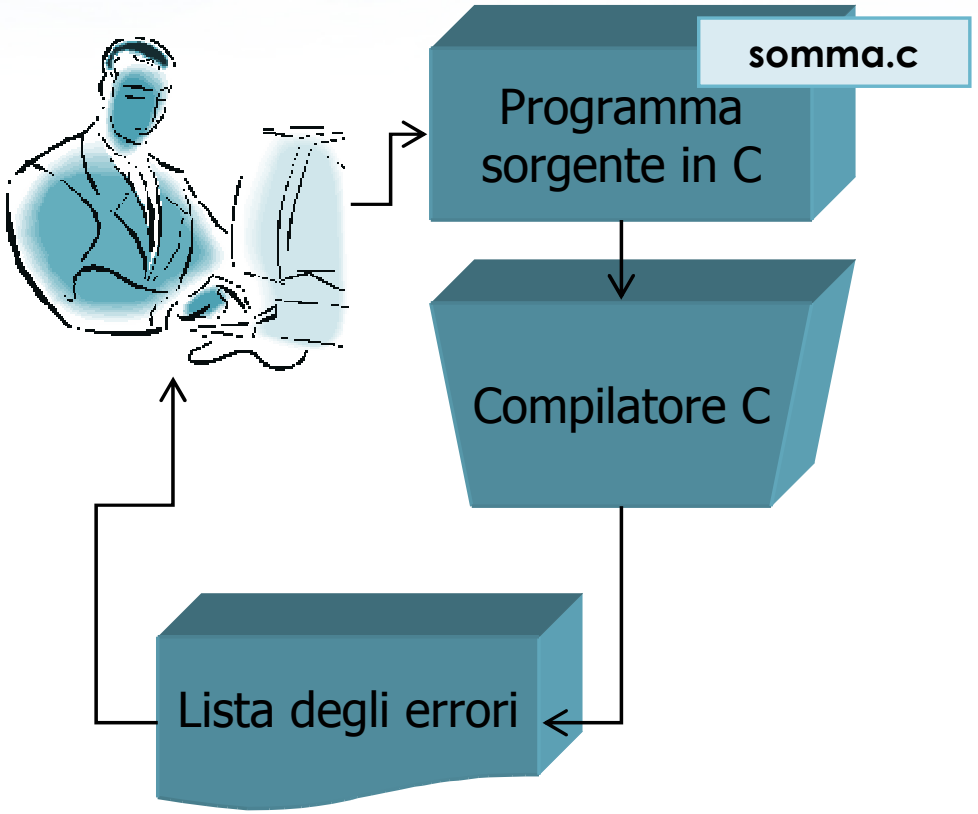
Compilazione del programma

- Attivare il compilatore sul programma sorgente `somma.c`
- Il compilatore verifica che non ci siano errori di sintassi
- In assenza di errori, viene generato il programma eseguibile `somma.exe`



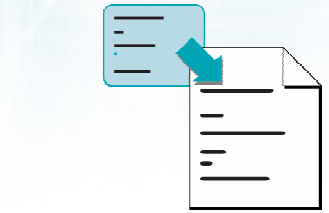
Correzione errori di sintassi

- Il compilatore genera una lista di messaggi di errore
- Capire il messaggio
- Identificare il punto errato nel programma
- Trovare la soluzione
- Correggere il programma
- Generare una nuova versione del file sorgente

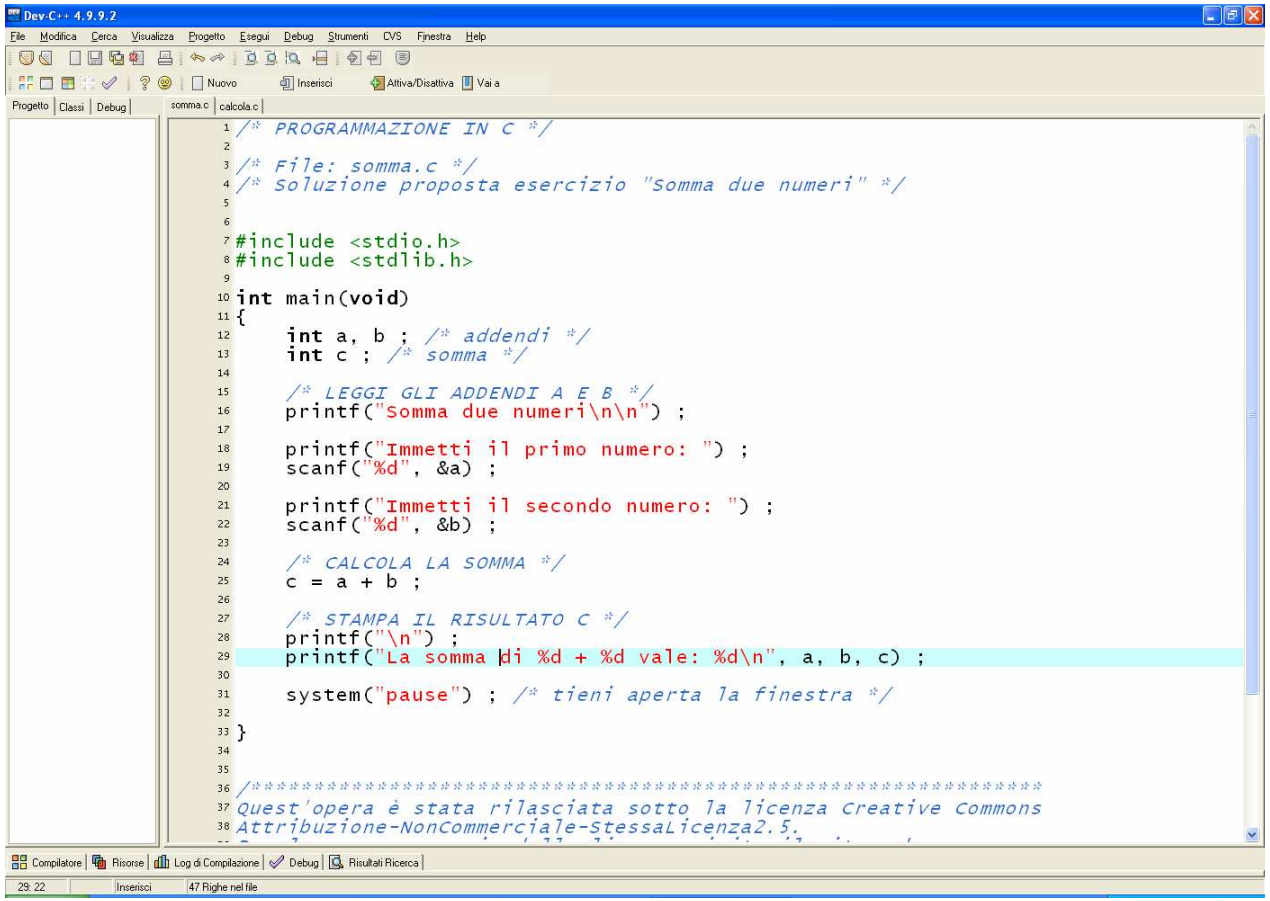


Compilazione "Somma due numeri"

- Compiliamo il programma



somma.c

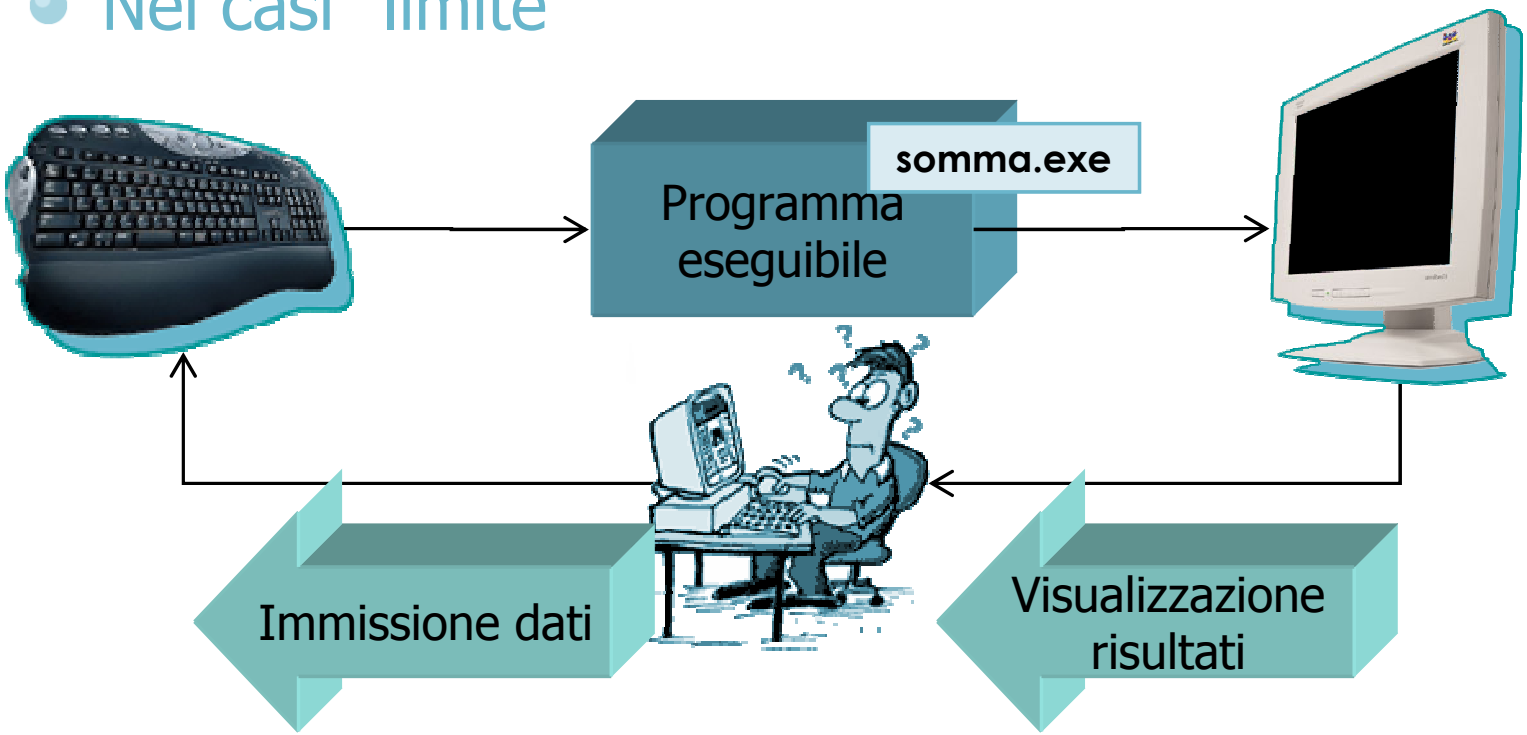


Compilare il primo programma

Esecuzione e verifica

Verifica del programma

- Ci mettiamo nei panni dell'utente finale
- Eseguiamo il programma
- Verifichiamo che funzioni correttamente
  - Nei casi "normali"
  - Nei casi "limite"



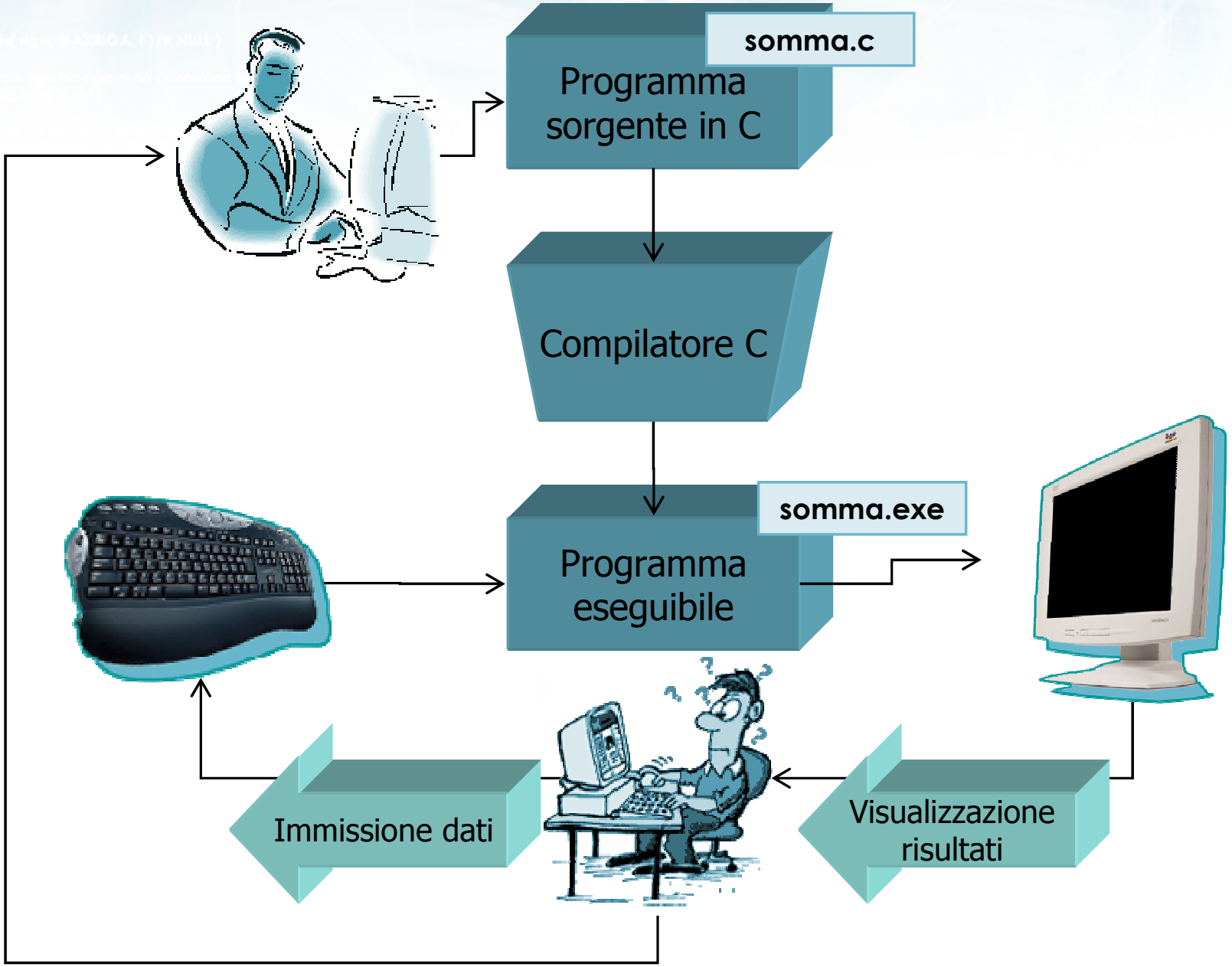
Errori in esecuzione

- Tipologie di errori possibili:
  - Crash del programma
    - Blocco imposto dal sistema operativo
  - Blocco del programma
    - Ciclo "infinito"
  - Risultati errati
    - (Quasi) sempre
    - Solo in alcuni casi (con alcuni dati ma non con altri)

Correzione errori di esecuzione

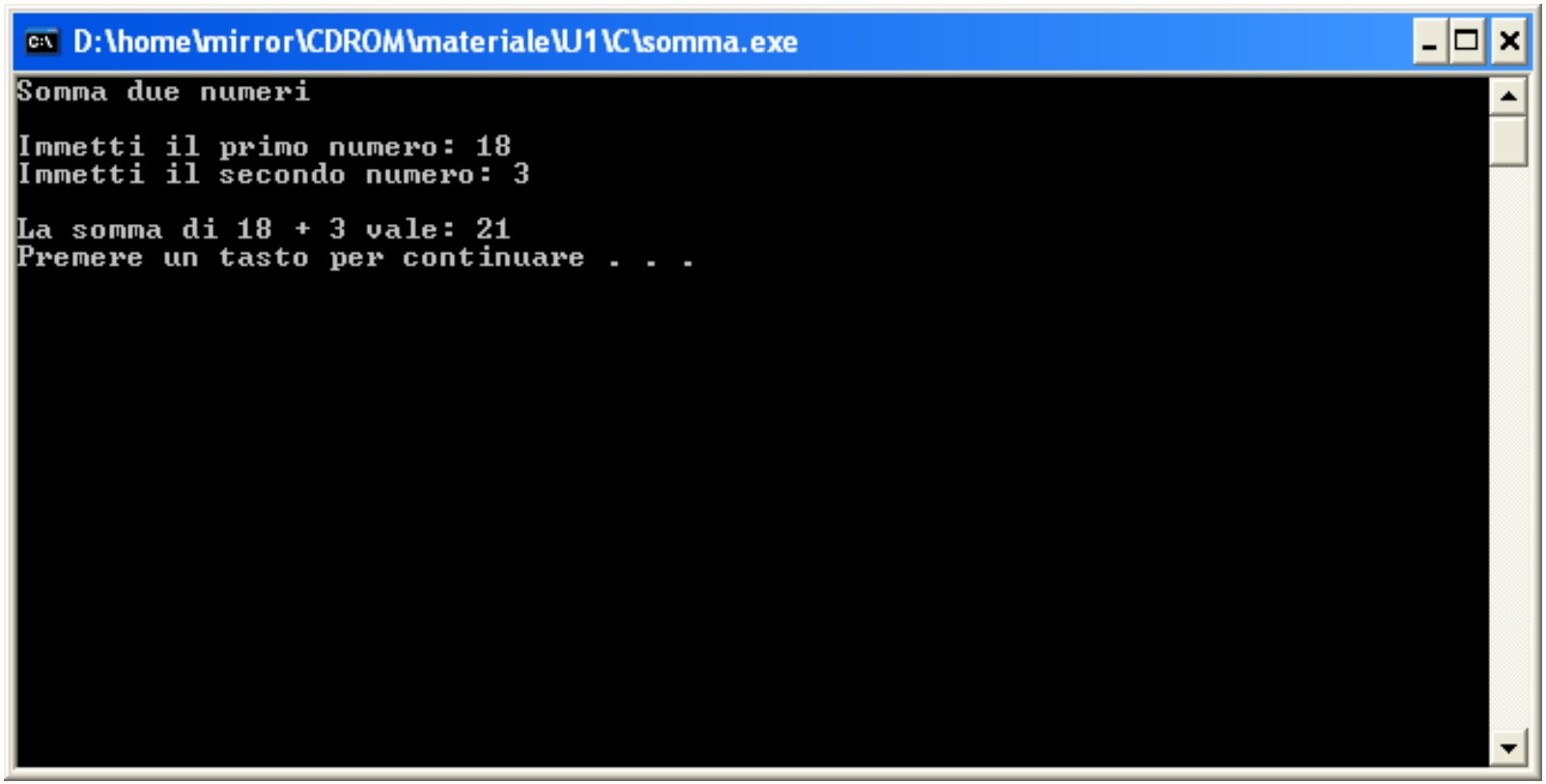
- Lavoro da “detective”
- Risalire dai sintomi alle cause del malfunzionamento
- Formulare delle ipotesi sulla causa dell’errore e verificarle
- Una volta trovato l’errore, cercare una soluzione
- A seconda della gravità, occorrerà modificare
  - Il sorgente C
  - L’algoritmo risolutivo
  - L’approccio generale

Correzione errori di esecuzione



Verifica “Somma due numeri”

- Eseguiamo il programma con alcuni dati di prova, verificandone il comportamento corretto



Primo programma in C

Esercizi proposti

Esercizi proposti

- Esercizio “Equazione di primo grado”
- Esercizio “Calcolo di aree”
- Esercizio “Somma minuti”

Esercizi proposti

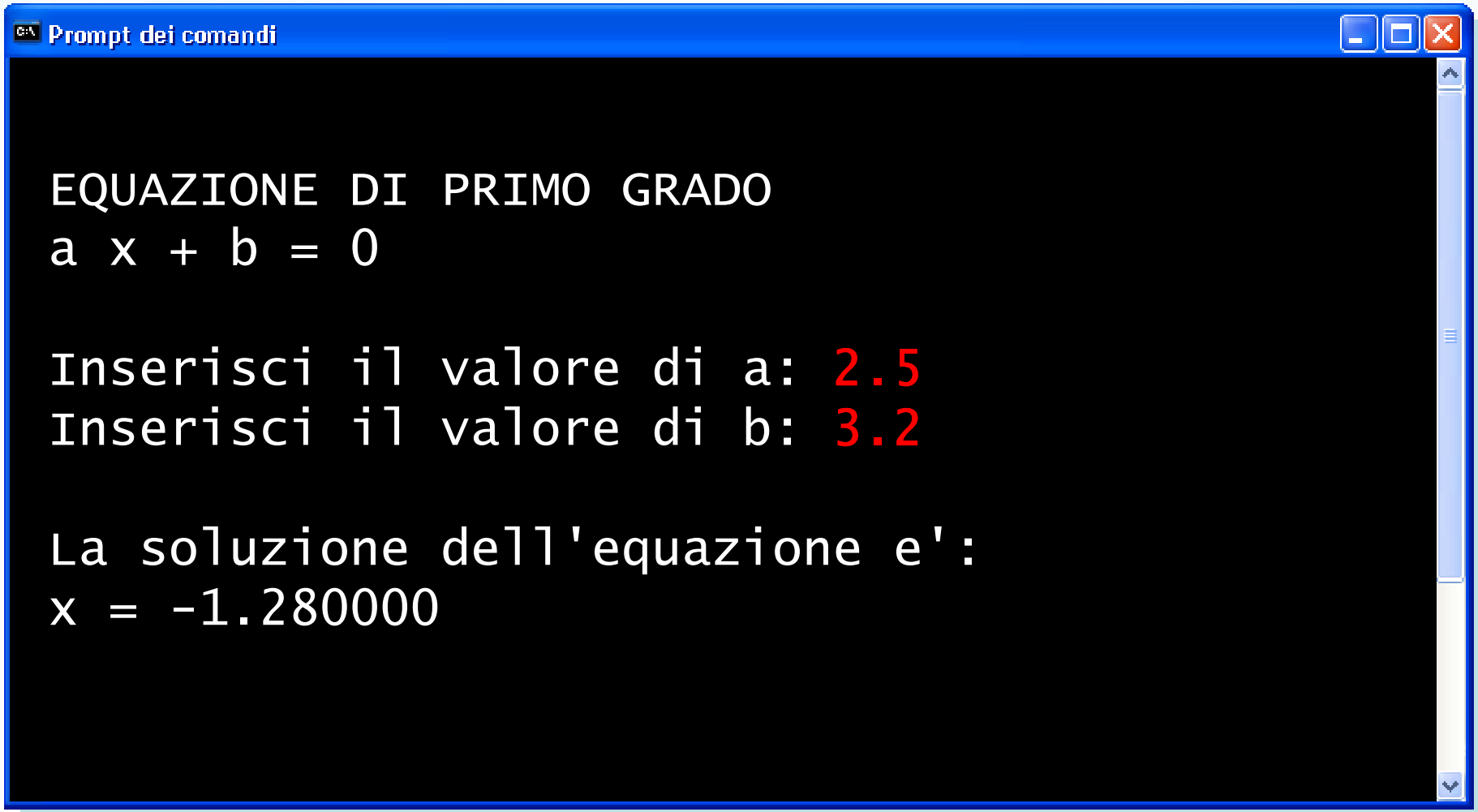
Esercizio "Equazione di primo grado"

Esercizio “Equazione di primo grado”

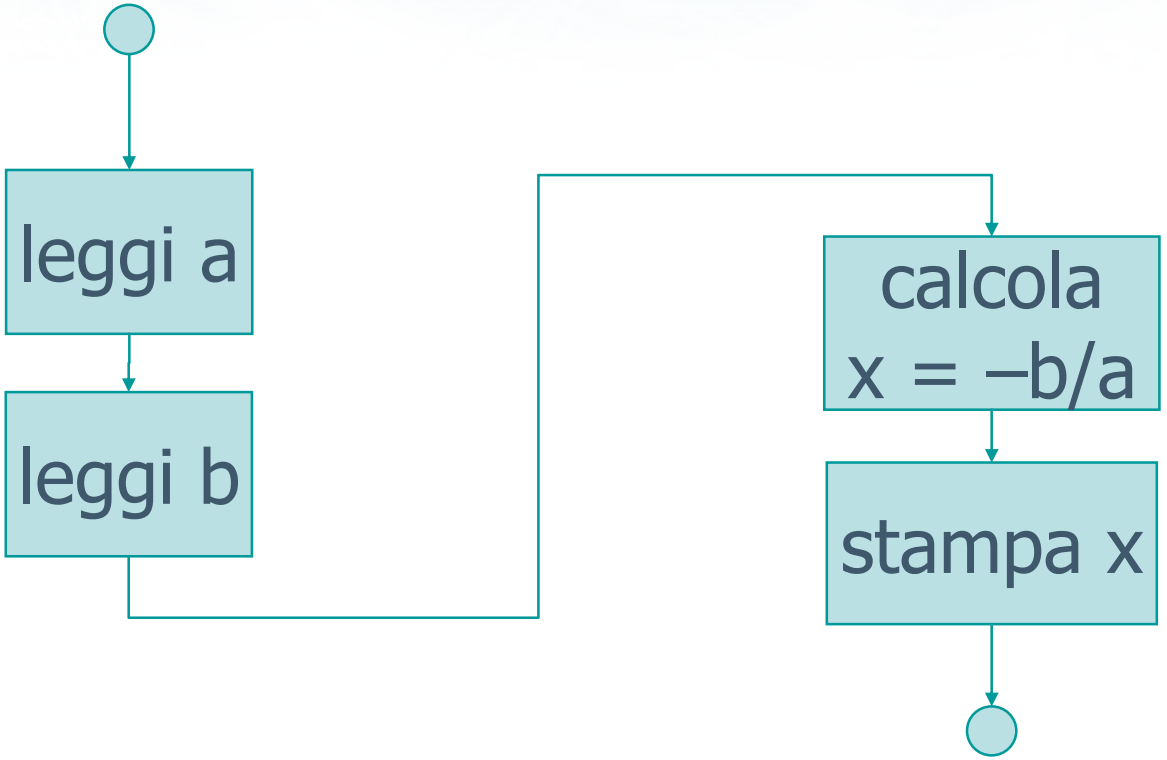
- Data l’equazione
  - $a\,x + b = 0$

  con a e b inseriti da tastiera, determinare il valore di x che risolve l’equazione

Analisi


Prompt dei comandi
EQUAZIONE DI PRIMO GRADO
a x + b = 0
Inserisci il valore di a: 2.5
Inserisci il valore di b: 3.2
La soluzione dell'equazione e':
x = -1.280000

Soluzione



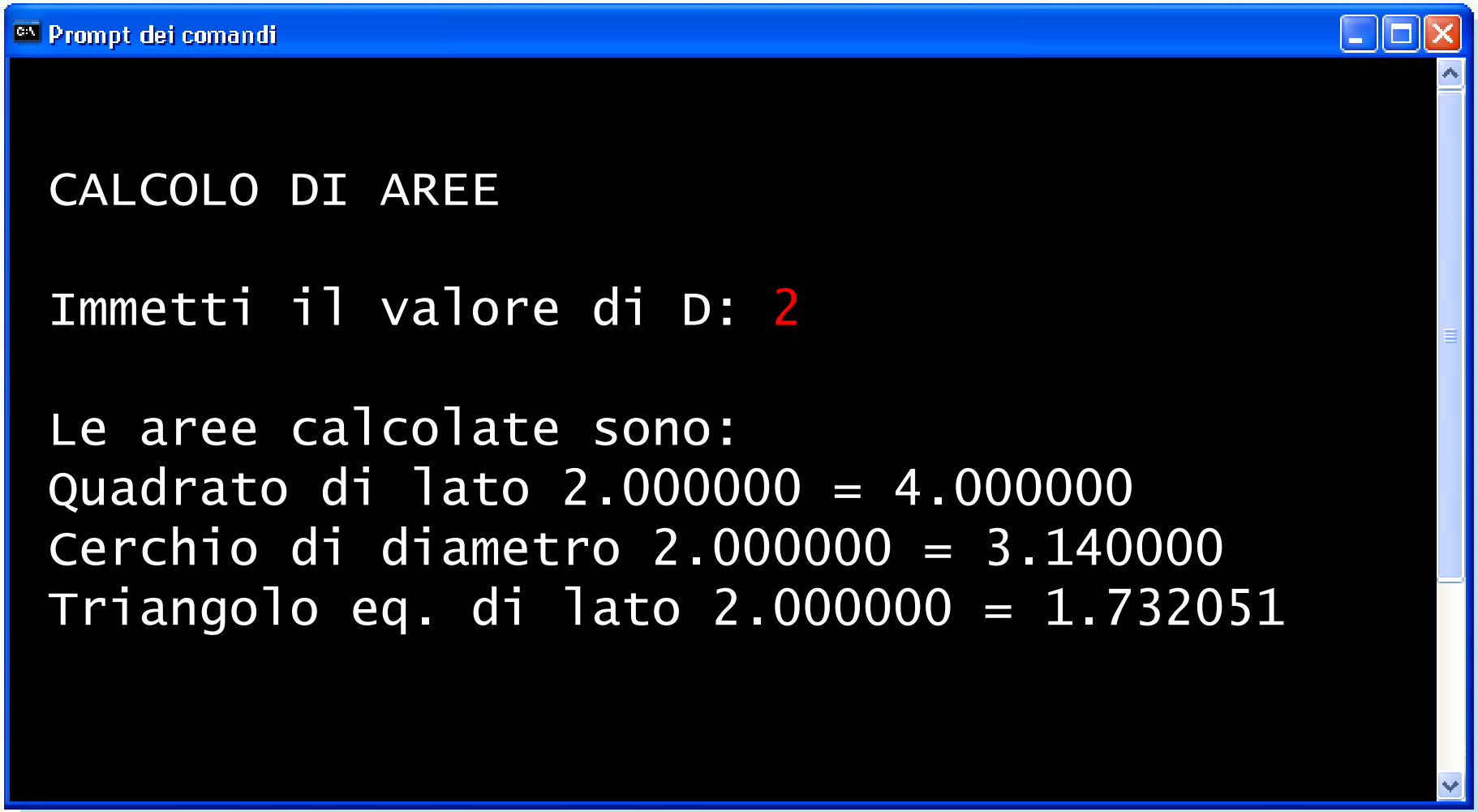
Esercizi proposti

Esercizio “Calcolo di aree”

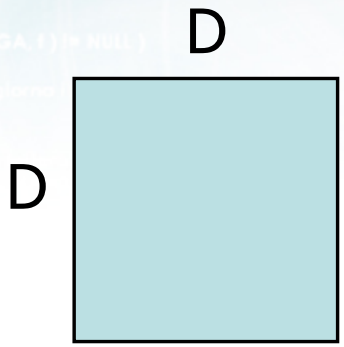
Esercizio "Calcolo di aree"

- Si scriva un programma in linguaggio C che, dato un numero reale immesso da tastiera, detto D, calcoli e stampi:
  - L'area del quadrato di lato D
  - L'area del cerchio di diametro D
  - L'area del triangolo equilatero di lato D

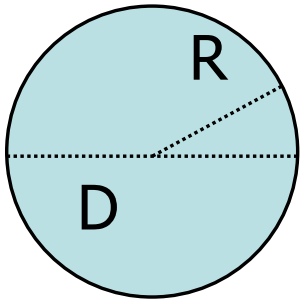
Analisi

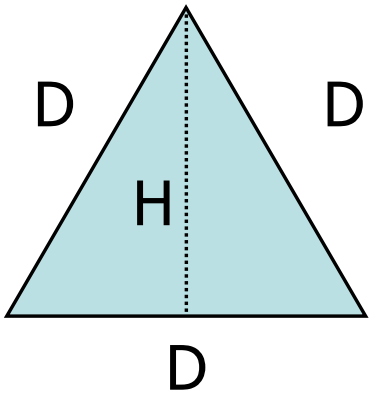

Prompt dei comandi
CALCOLO DI AREE
Immetti il valore di D: 2
Le aree calcolate sono:
Quadrato di lato 2.000000 = 4.000000
Cerchio di diametro 2.000000 = 3.140000
Triangolo eq. di lato 2.000000 = 1.732051

Aree

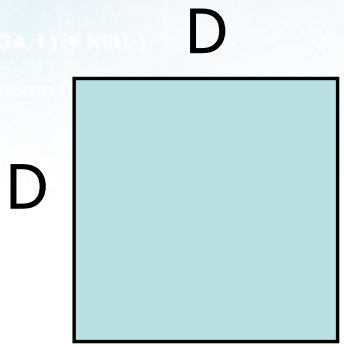


$$A = D^2$$

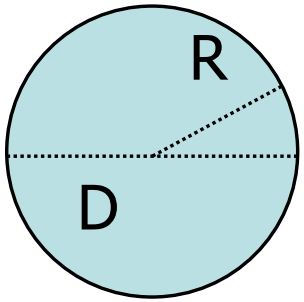




Aree

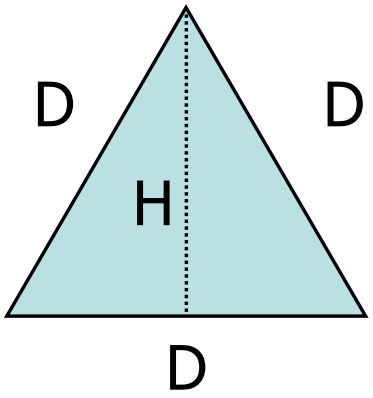


$$A = D^2$$

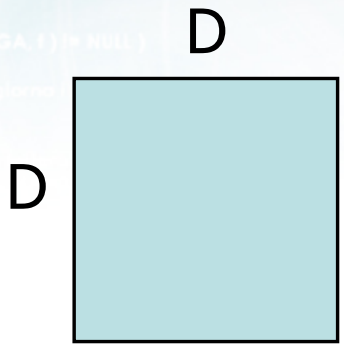


$$A = \pi \cdot R^2$$

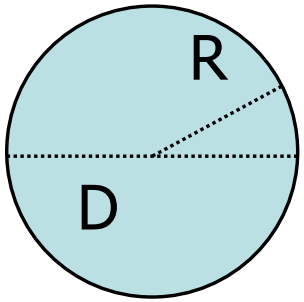
$$R = D/2$$



Aree

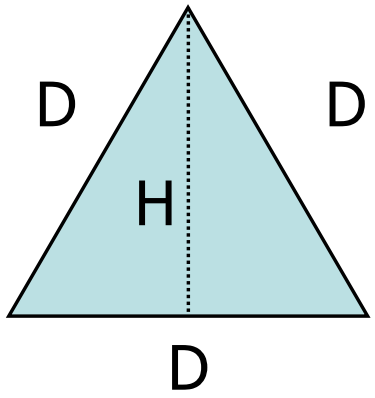


$$A = D^2$$



$$A = \pi \cdot R^2$$

$$R = D/2$$



$$A = \frac{D \cdot H}{2}$$

$$H = D \cdot \sin(60°) =$$

$$= D \cdot \sin\left(\pi/3\right) = D \cdot \sqrt{3}/2$$

Avvertenze

- Per le funzioni matematiche (`sin`, `sqrt`, ...) occorre includere `math.h`
- Gli argomenti delle funzioni trigonometriche (`sin`, `cos`, ...) devono essere espressi in radianti
- Il calcolo del quadrato si ottiene moltiplicando la variabile per se stessa: $D^2 = D \times D$
- Il valore di π deve essere definito dal programmatore in un'apposita variabile
  - La costante `M_PI`, definita in `math.h`, non è più supportata dallo standard ANSI C

Esercizi proposti

Esercizio “Somma minuti”

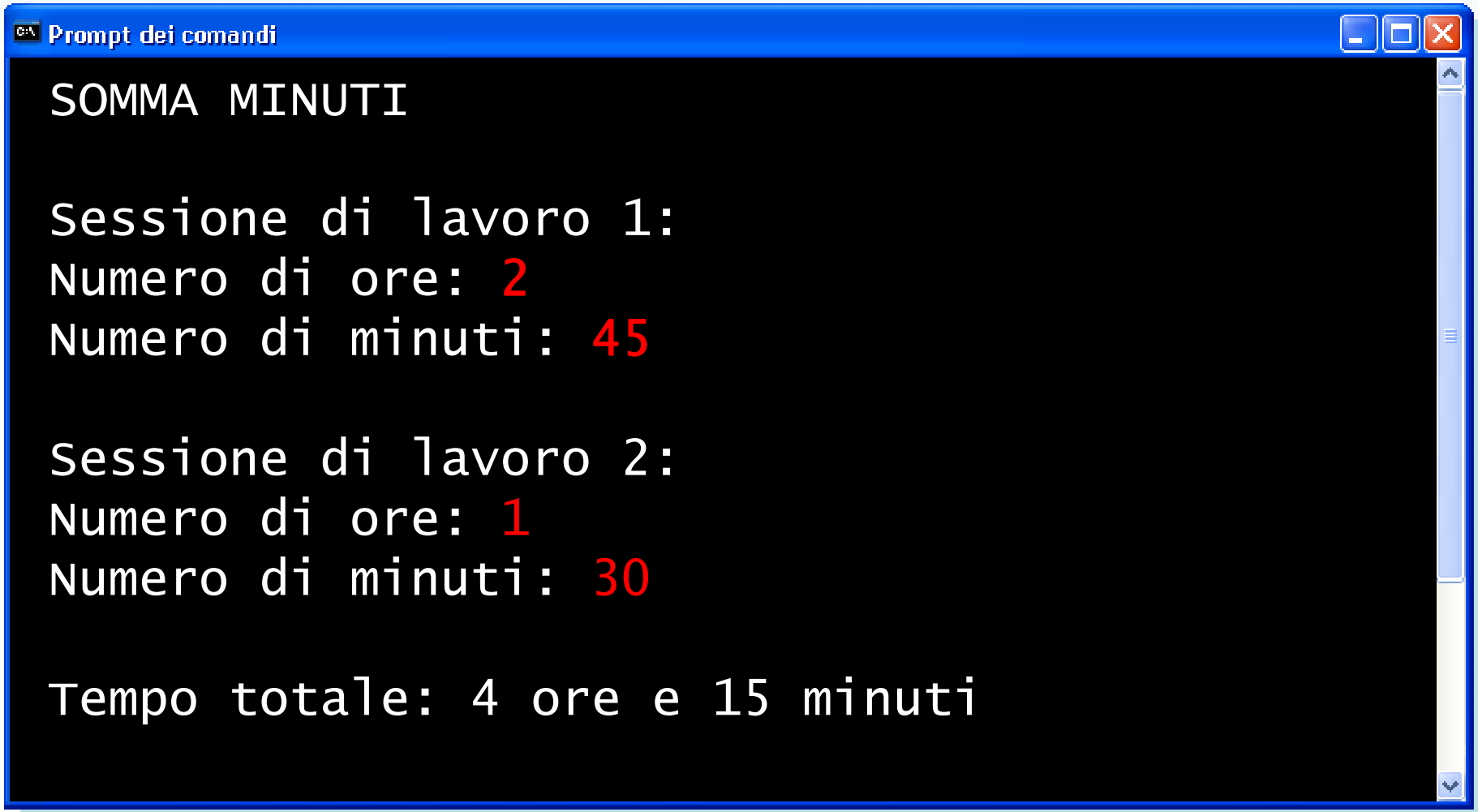
Esercizio "Somma minuti" (1/2)

- Un consulente deve calcolare il numero di ore e minuti per cui ha lavorato per un cliente
- Il consulente ha lavorato in due distinte sessioni di lavoro, per ciascuna delle quali ha annotato il numero di ore e il numero di minuti impiegati

Esercizio “Somma minuti” (2/2)

- Si scriva un programma in C che, a partire dalle ore e minuti della prima sessione di lavoro e dalle ore e minuti della seconda sessione di lavoro, calcoli il numero di ore e minuti complessivi

Analisi


Prompt dei comandi
SOMMA MINUTI
Sessione di lavoro 1:
Numero di ore: 2
Numero di minuti: 45
Sessione di lavoro 2:
Numero di ore: 1
Numero di minuti: 30
Tempo totale: 4 ore e 15 minuti

Aritmetica dell'orologio

- Diciamo:
  - ore1, min1 le ore/minuti della prima sessione
  - ore2, min2 le ore/minuti della seconda sessione
  - oretot, mintot le ore/minuti totali
- Non è possibile semplicemente sommare ore e minuti separatamente, in quanto min1+min2 potrebbe essere maggiore di 59
- Bisogna tener conto del "riporto" nella somma dei minuti

Soluzione

- mintot = (min1 + min2) modulo 60
- oretot = ore1 + ore2 + riporto
  - riporto = parte intera di (min1 + min2) / 60

Soluzione

- mintot = (min1 + min2) modulo 60
- oretot = ore1 + ore2 + riporto
  - riporto = parte intera di (min1 + min2) / 60

```
int ore1, ore2, oretot ;
int min1, min2, mintot, riporto ;

...

mintot = (min1 + min2) % 60 ;

riporto = (min1 + min2) / 60 ;

oretot = ore1 + ore2 + riporto ;
```

Primo programma in C

Sommario

Argomenti trattati

- Presentazione del linguaggio C
- Struttura base di un file sorgente in C
- Istruzioni minime per iniziare a programmare
  - Tipi fondamentali `int` e `float`
  - Istruzioni fondamentali di input/output
  - Istruzione di assegnazione
- Operazioni necessarie per compilare ed eseguire il programma

Suggerimenti

- Analizzare sempre il comportamento previsto del programma prima di iniziare a scrivere il sorgente
  - Interazione con l'utente
  - Risoluzione manuale con carta e penna
- Abbondare con i commenti
- Leggere con attenzione tutti i messaggi di errore e di warning del compilatore, e correggerli
- Verificare il programma con diversi dati di prova

Materiale aggiuntivo

- Sul CD-ROM
  - Testi e soluzioni degli esercizi trattati nei lucidi
  - Scheda sintetica
  - Esercizi risolti
  - Esercizi proposti
- Esercizi proposti da altri libri di testo