Programmazione in C

Unità
Primo programma in C

Primo programma in C

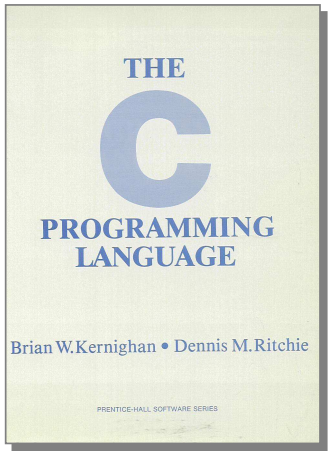
Introduzione al linguaggio C

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



Un esempio

```
#include <stdio.h>

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

    return 0;
}
```

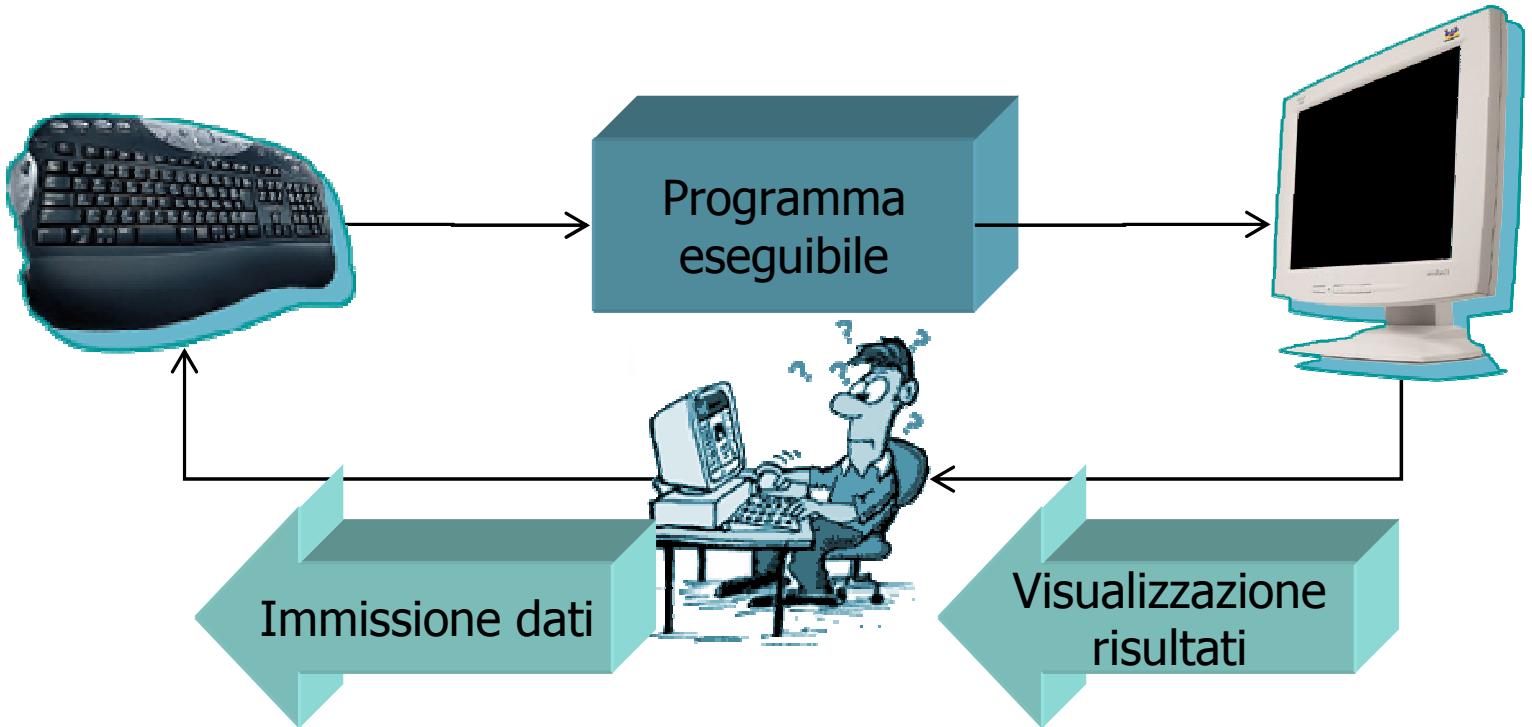
Primo programma in C

Struttura minima di un file C

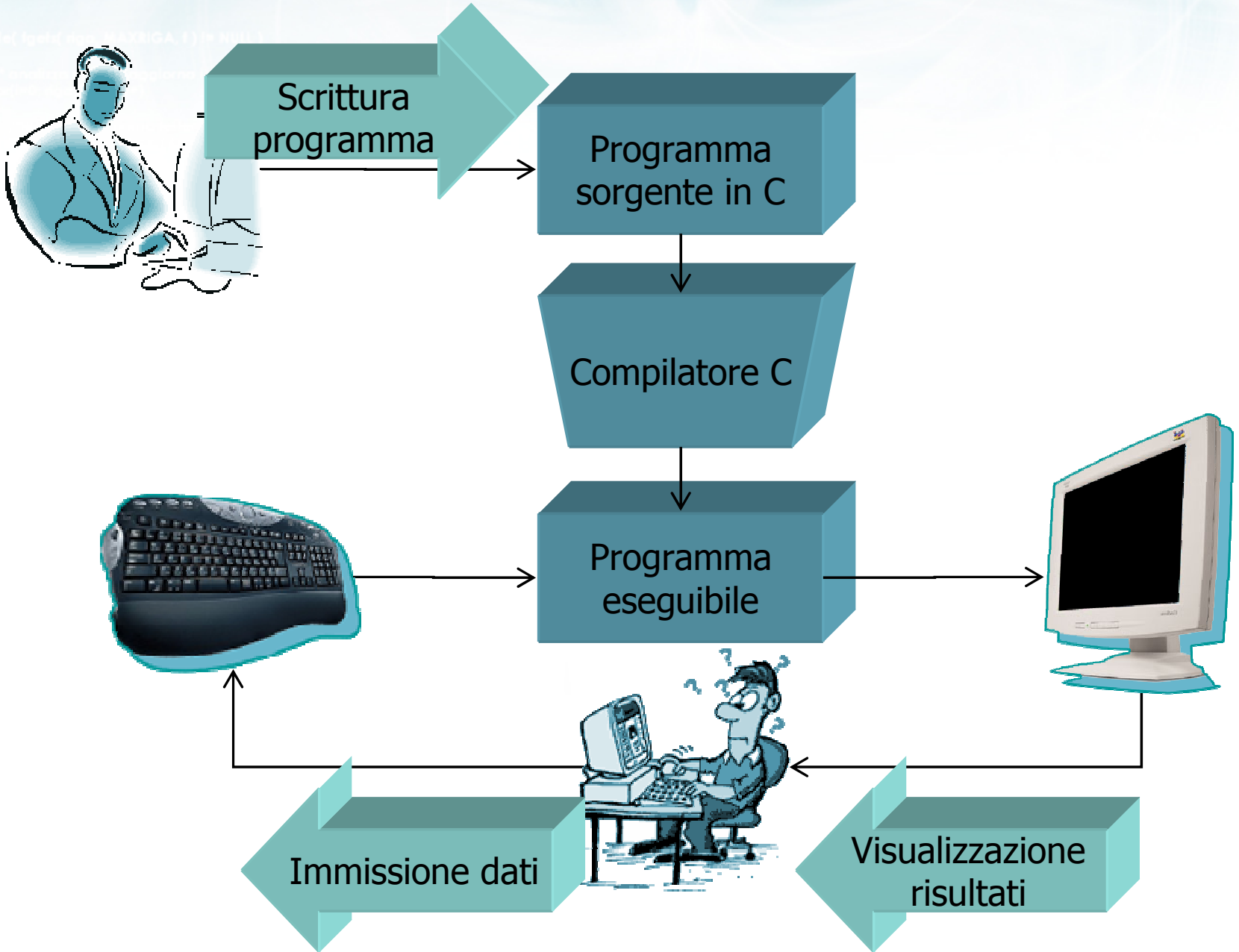
Struttura minima di un file C

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

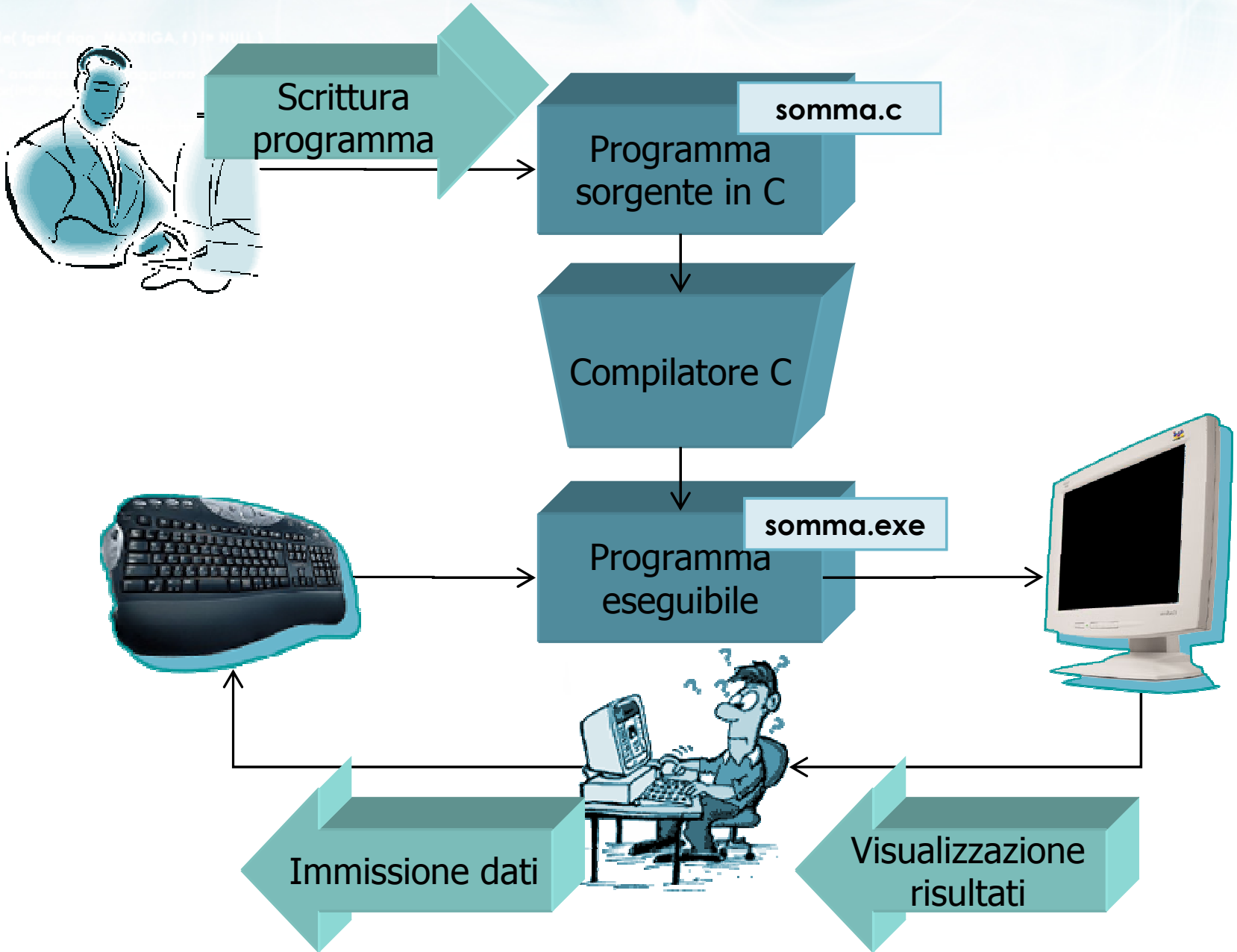
Modello di applicazioni "console"



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

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

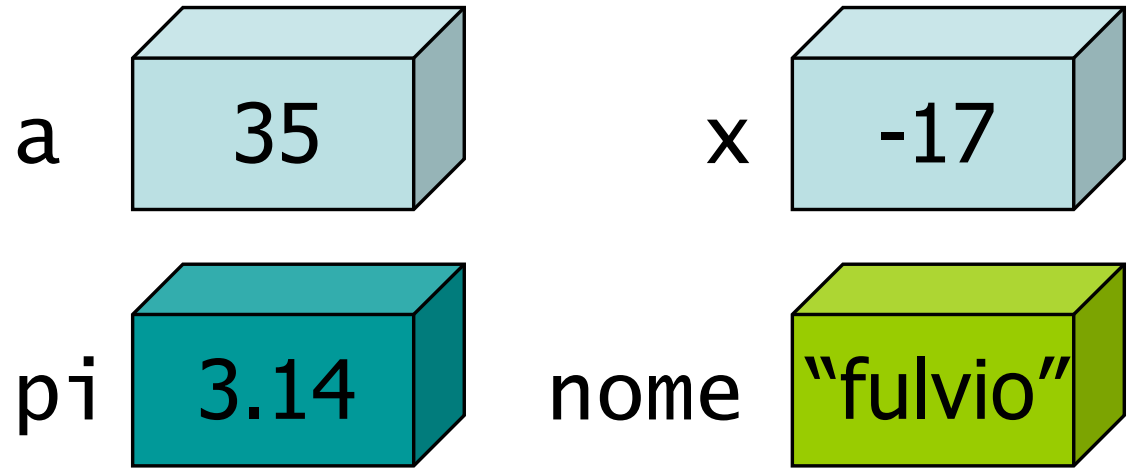
Definizione di variabili

Variabili

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

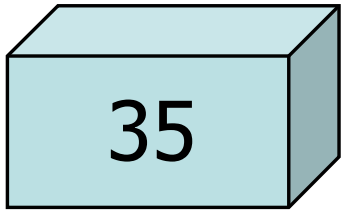
Variabili

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

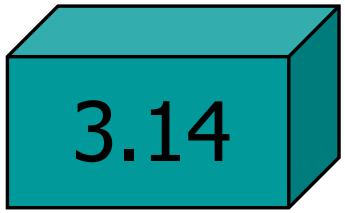


Tipo di dato

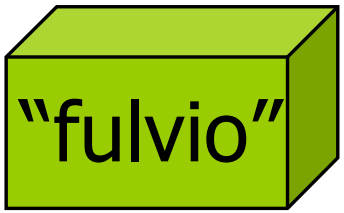
- Definisce l'insieme dei valori ammissibili per la variabile



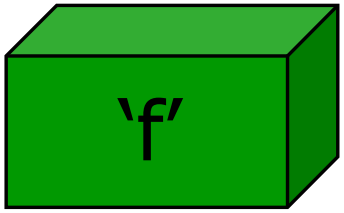
Numeri interi, positivi o negativi



Numeri reali



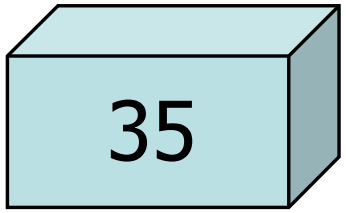
Stringhe di testo



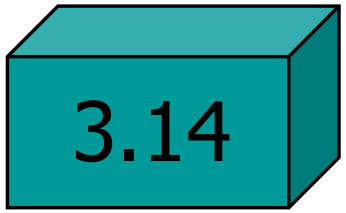
Singoli caratteri di testo

Tipo di dato

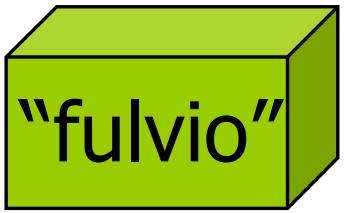
- Definisce l'insieme dei valori ammissibili per la variabile

`int` 

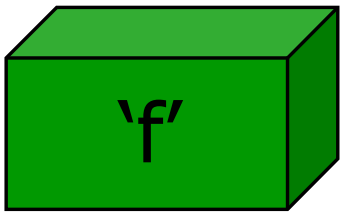
Numeri interi, positivi o negativi

`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

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 ;

    . . . . .
}
```

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

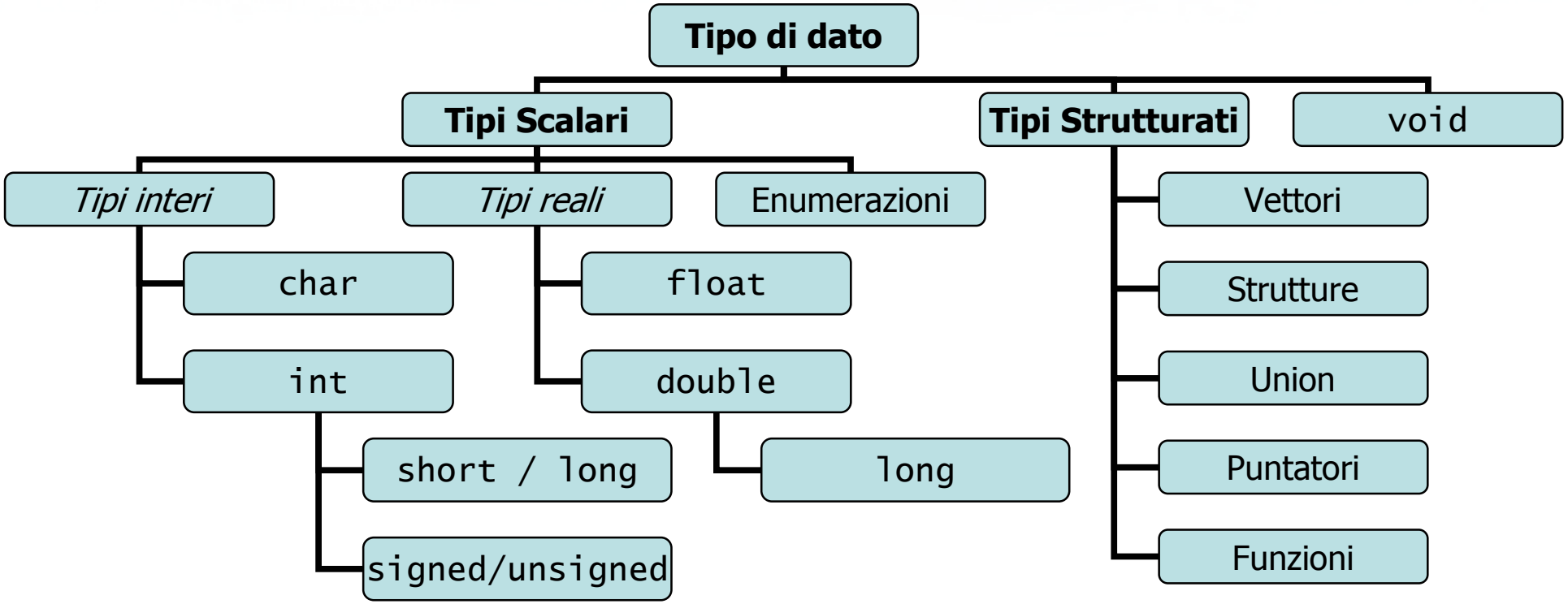
Sottoinsieme minimale di istruzioni

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



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

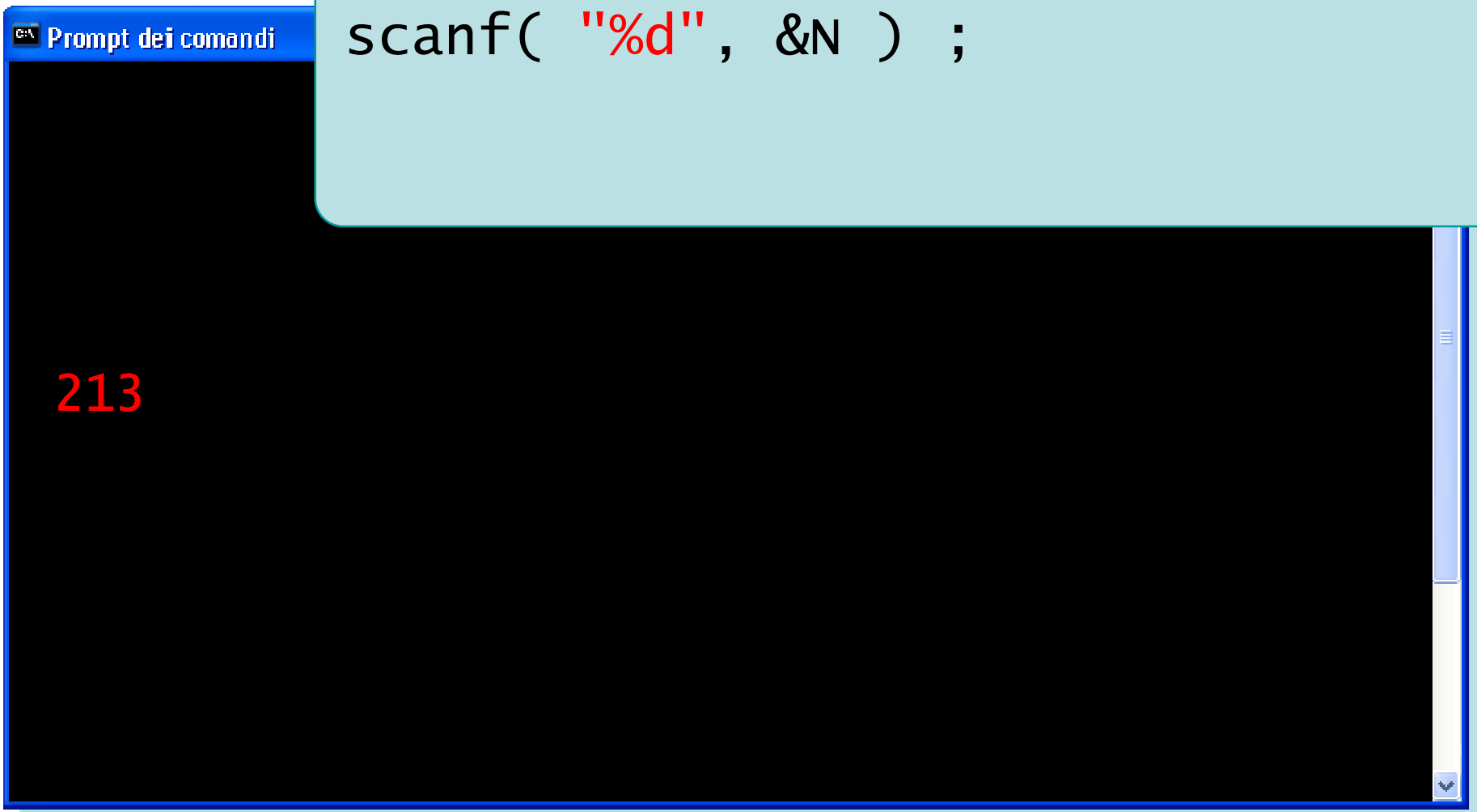
Istruzione scanf – semplificata

Istruzioni di lettura

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

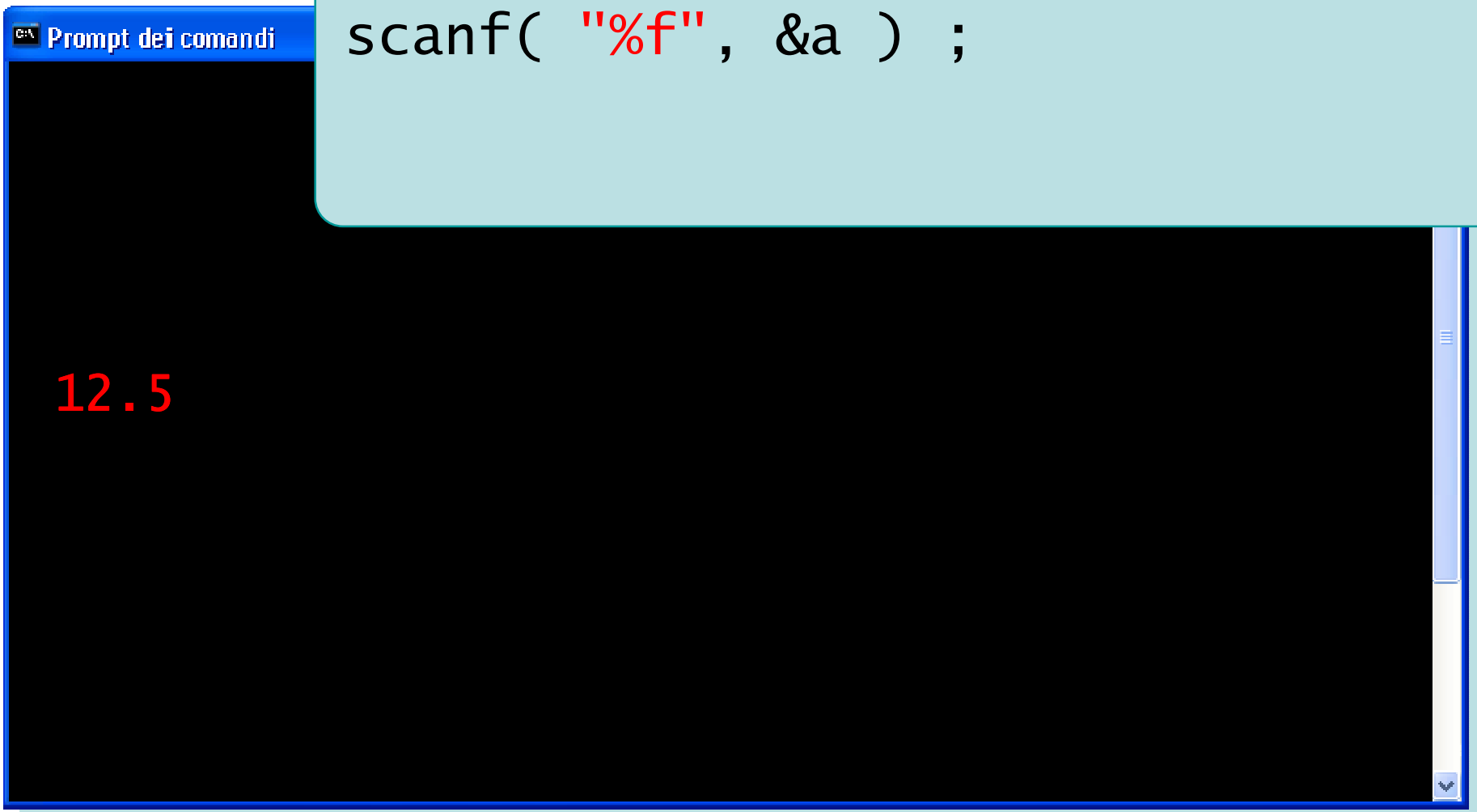
Lettura di un intero

```
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

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`