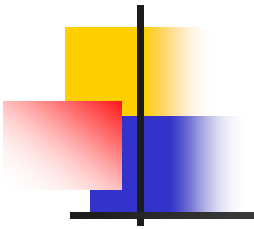
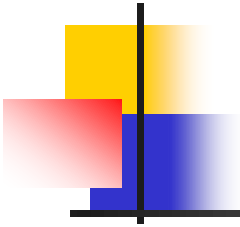


Dal problema al programma: introduzione al problem-solving in linguaggio C

Capitolo 4: Problem-solving su dati vettoriali

G. Cabodi, P. Camurati, P. Pasini,
D. Patti, D. Vendraminetto

MAGGIOLI EDITORE
APOGEO education
2016



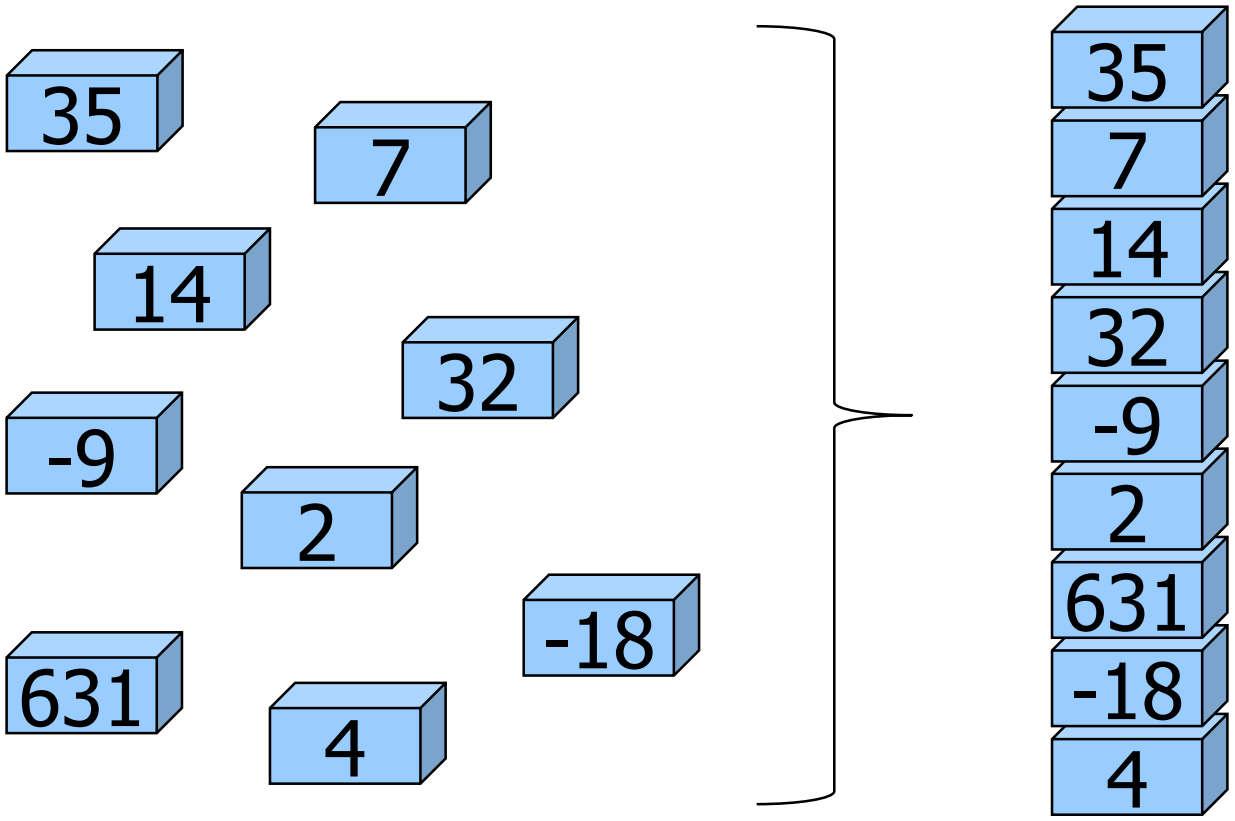
Nota sui puntatori

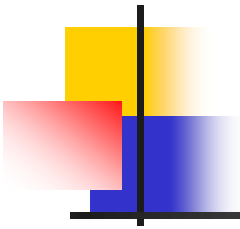
- Le rappresentazioni (figure) di tabelle con stringhe
- I programmi:
  - menu con scelta su parola
  - conversione matricola-nome

sono stati leggermente modificati rispetto alla versione del libro, in modo da evitare l'uso di puntatori. Le versioni con puntatori saranno illustrate dopo l'introduzione del tipo puntatore.

I vettori nel problem solving

- I vettori sono insieme di dati dello stesso tipo
- Un vettore può essere utilizzato come contenitore di dati (omogenei) senza alcun criterio di ordine:





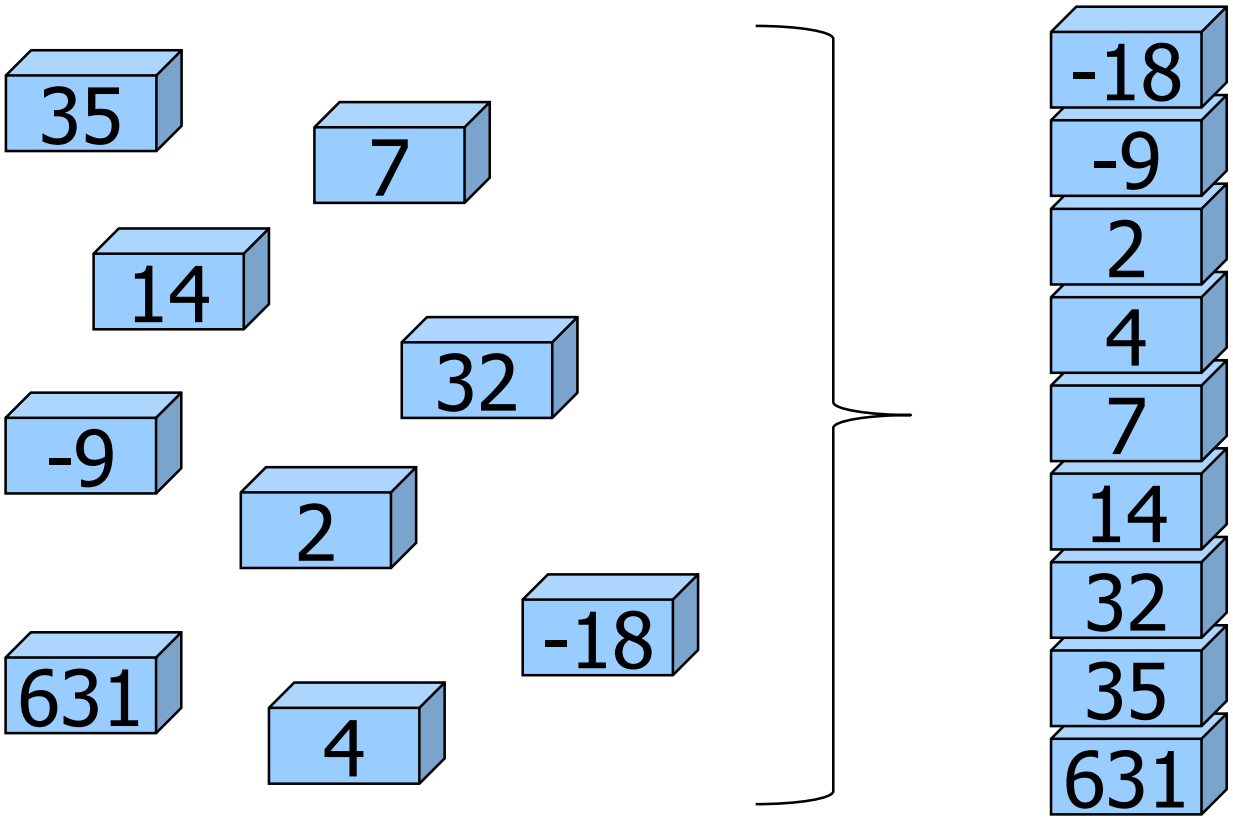
Il vettore come contenitore

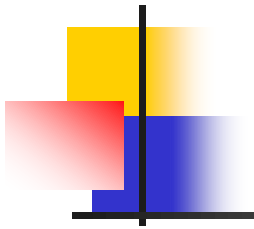
- Utilizzo:
  - problemi in cui è sufficiente raccogliere un insieme di dati, senza relazioni di ordine
  - un dato può essere in qualunque posizione nel vettore
  - accesso ai dati mediante generazione (iterativa) di tutti gli indici

    Esempi:
    - collezionare dati in input, per elaborazioni successive, o preparare dati per l'output
    - operazioni su insiemi di dati

Il vettore ordinato

I dati nel vettore sono organizzati secondo un criterio di ordine (es. valori crescenti):





Utilizzo:

- problemi in cui occorre ordinare e/o mantenere ordinato un insieme di dati, secondo un certo criterio
- il caso più frequente è il vettore mono-dimensionale: ordine lineare (totale o parziale)
- l'indice di un dato indica la posizione nell'ordinamento

Esempi:

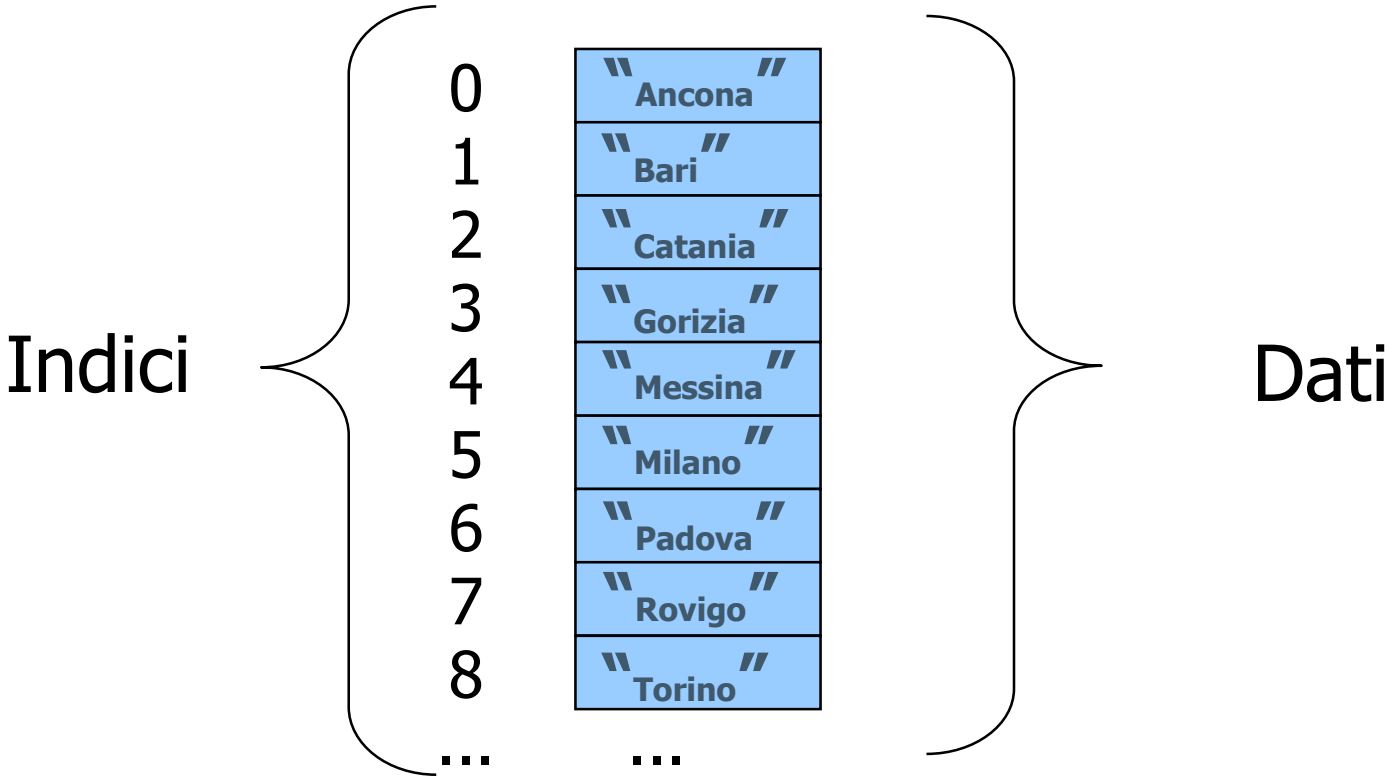
- ordinare dati mediante criterio cronologico (inverso all'input) o secondo valori (crescenti/decrescenti)
- sequenze di campioni (numeri) di grandezze fisiche
- calcoli matematici e/o statistici su successioni di numeri

Corrispondenza indice ↔ dato

Ad ogni indice (intero nell'intervallo 0..NDATI-1) corrisponde un dato (e viceversa)

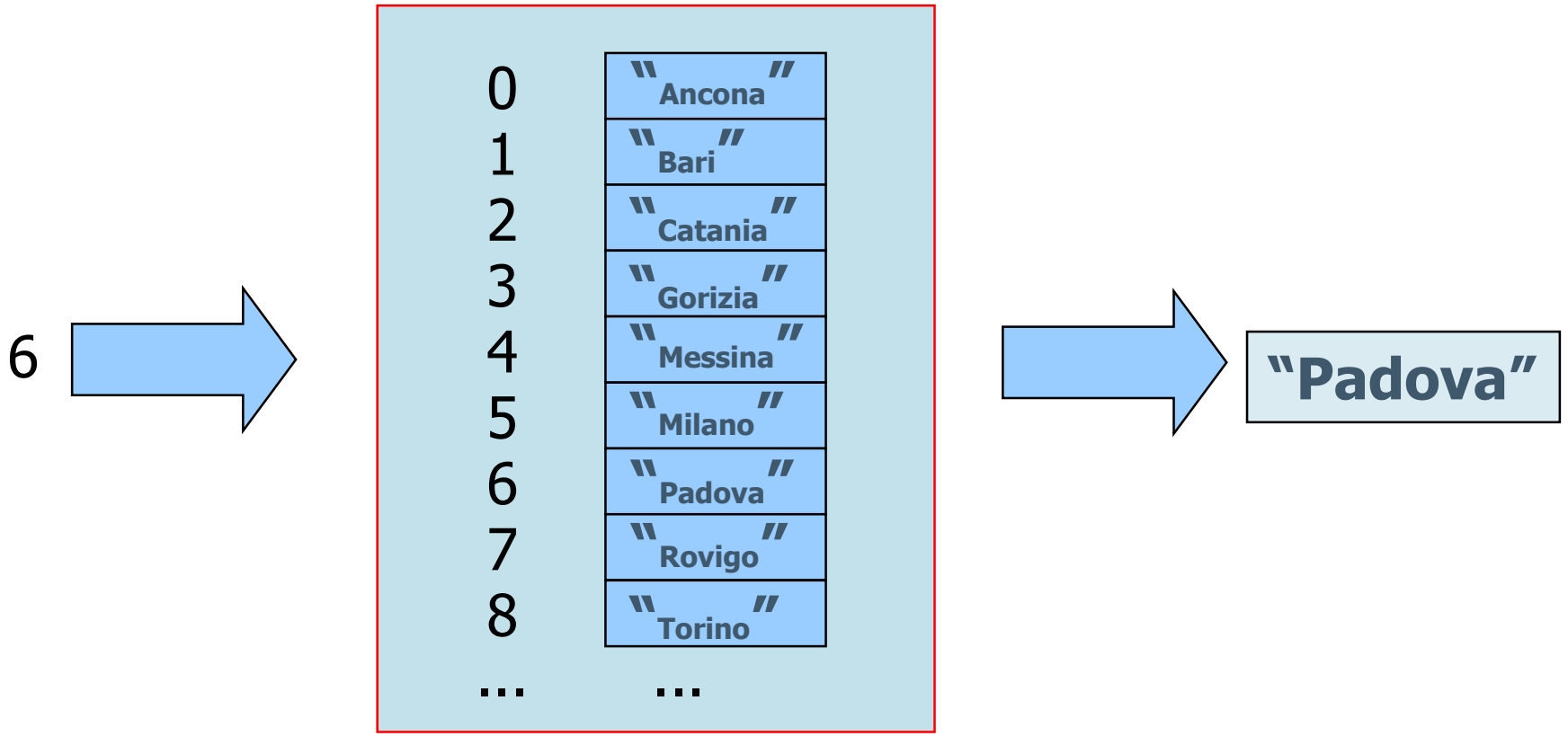
Corrispondenza indice ↔ dato

Ad ogni indice (intero nell'intervallo 0..NDATI-1) corrisponde un dato (e viceversa)



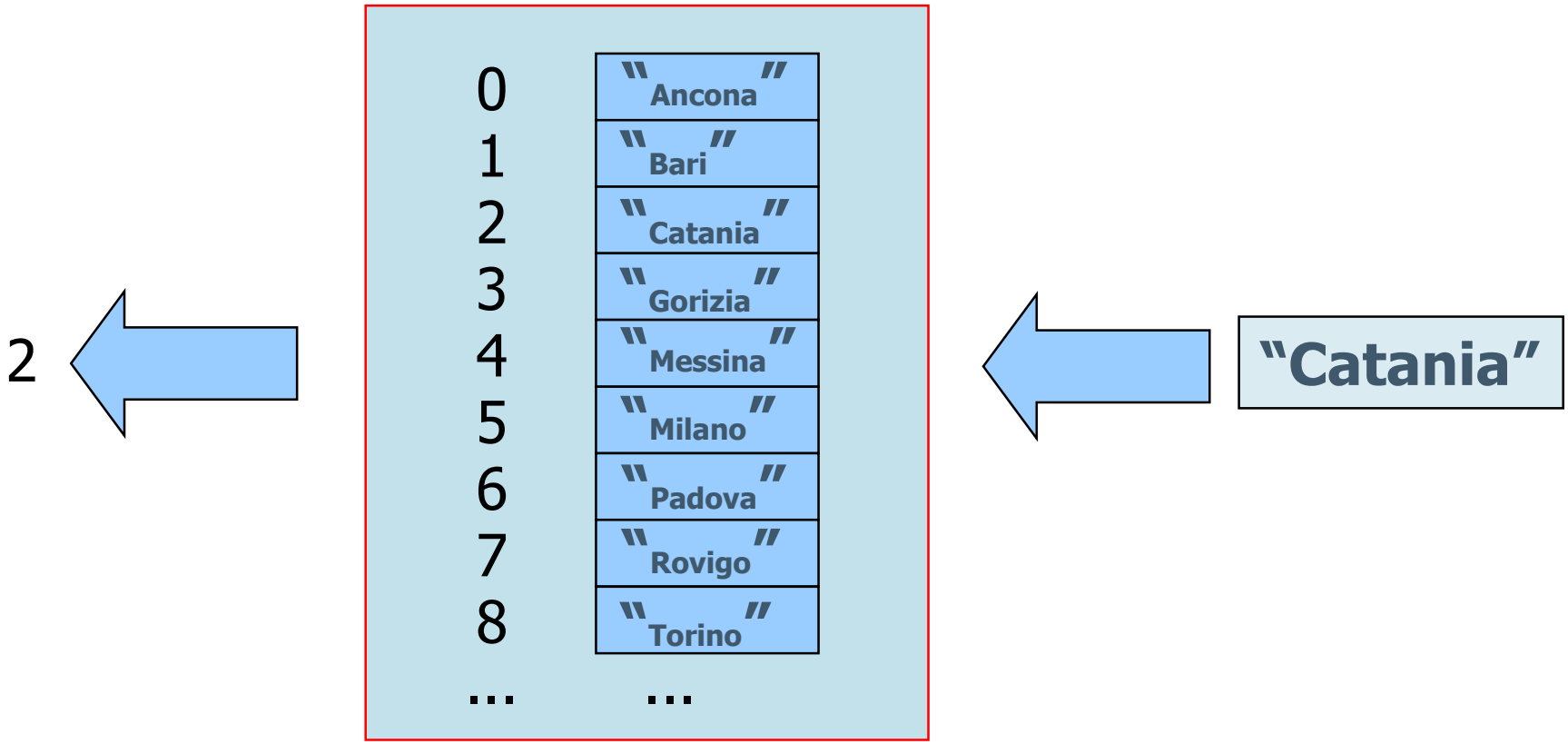
Da indice a dato

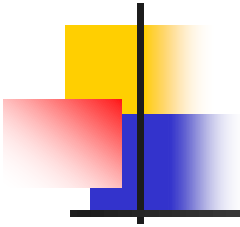
A partire dall'indice, calcolare il dato. Operazione immediata, grazie al meccanismo di accesso ai vettori:



Da dato a indice

A partire dal dato, calcolarne l'indice. Operazione non immediata, in quanto occorre cercare il dato:

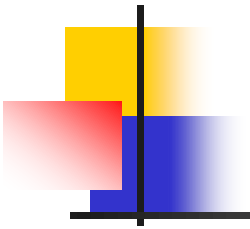




Indici e dati

La relazione indice $\leftrightarrow$ dato:

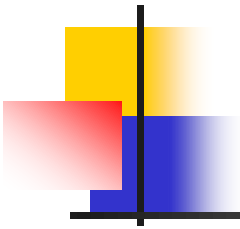
- ogni casella di un vettore è caratterizzata da un indice (o più indici, nel caso multi-dimensionale) e da un dato
- la casella del vettore può essere quindi utilizzata per mettere in relazione indice e dato



- Utilizzo:
  - problemi in cui esistono relazioni/corrispondenze tra numeri interi e dati (numerici o non numerici)
  - l'intero è associato all'indice di una casella, il dato al contenuto
  - attenzione!
    - l'intero (indice) non può essere troppo grande
    - è possibile che occorra un dato vuoto (nullo) per le caselle non utilizzate

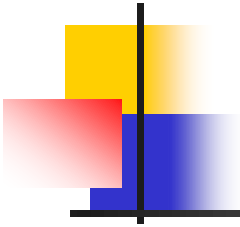
Esempi:

- problemi numerici (statistiche e conteggi in relazione a dati interi, tabulazione di funzione nel piano cartesiano y=f(x), con ascissa intera o riconducibile a intero)
- problemi di codifica/decodifica numerica di informazioni (non numeriche)
- problemi di elaborazione testi (matrice di caratteri in corrispondenza a pagina da visualizzare)
- problemi di verifica (flag logici in corrispondenza a dati interi)
- problemi di selezione (ricerca di indice in corrispondenza a una chiave).



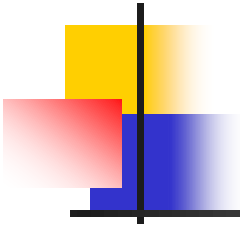
Problemi numerici

- Problemi di algebra, geometria, statistica, ecc., simili a quelli risolti mediante dati scalari
- I vettori possono essere utilizzati:
  - per collezionare e manipolare (insiemi di) numeri
  - per rappresentare dati con struttura lineare (vettori) o tabellare (matrici)
  - per gestire corrispondenze tra numeri (indice e dato).



Problemi su insiemi di numeri

- Problemi nei quali si gestiscono insiemi (gruppi) di dati numerici, con operazioni di I/O, unione, intersezione, …
- La soluzione spesso si basa su costrutti iterativi (eventualmente annidati) tali da:
  - percorrere gli elementi di un insieme
  - percorrere, per ogni elemento di un insieme, tutti gli elementi di un altro insieme.



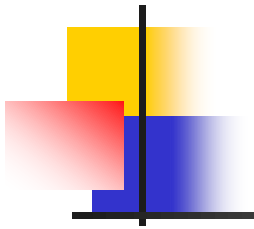
Intersezione tra insiemi di numeri

Formulazione:

- acquisire da tastiera un primo gruppo di 10 interi (privo di numeri ripetuti)
- acquisire da tastiera un secondo gruppo di 10 interi (privo di numeri ripetuti)
- calcolare e visualizzare tutti i numeri che sono presenti in entrambi i gruppi

Soluzione:

- si tratta di un problema di intersezione tra insiemi, che richiede di determinare, per ogni elemento del primo insieme, se appartiene anche al secondo



Algoritmo:

- input: due iterazioni per acquisire i dati dei due gruppi
- elaborazioni: doppia iterazione per confrontare tutti i dati del primo gruppo con tutti quelli del secondo (o viceversa)
- output: iterazione sui dati dell'insieme intersezione.

Le tre parti possono essere distinte, oppure (parzialmente) integrate: mentre si esegue l'input si calcola l'intersezione e si scrivono i risultati in output.

Struttura dati:

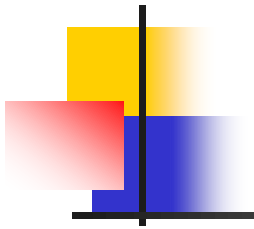
La scelta dipende dall'algoritmo adottato:

- è necessario almeno un vettore per i dati del primo insieme
- l'utilizzo di un secondo vettore dipende dal fatto di adottare uno schema
  - tre fasi separate: input, elaborazioni, output
  - elaborazioni ed output fatte durante l'input del secondo gruppo (per ogni numero del secondo gruppo, si determina direttamente se appartiene anche al primo e si fornisce il relativo output)
- la soluzione proposta utilizza due vettori, con tre fasi separate. Il risultato (intersezione) viene sovrapposto al primo vettore.

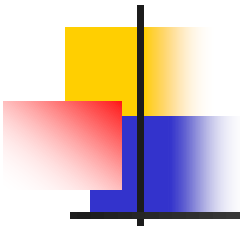
Codice

intersezione_insiemi.c

```c
#define NDATI 10
#include <stdio.h>
void leggiVettore (int dati[], int n);
void scriviVettore (int dati[], int n);

int main(void)  {
   int dati0[NDATI], dati1[NDATI];
   int i, j, ni, trovato;
   /* input */
   leggiVettore(dati0,NDATI);
   leggiVettore(dati1,NDATI);
   /* calcolo intersezione */
   ...
   /* output */
   scriviVettore(dati0,ni);
}
```

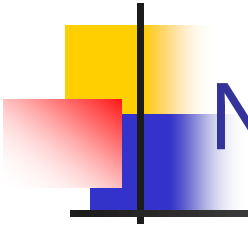


```
...
/* calcolo intersezione */
for (i=ni=0; i<NDATI; i++) {
  trovato=0;
  for (j=0; j<NDATI&&(!trovato); j++) {
    if (dati0[i]==dati1[j])
      trovato=1;
  }
  /* se dato appartiene ad intersezione
     riscrivi nella parte iniziale del
     vettore dati0 (già confrontato) */
  if (trovato)
    dati0[ni++]=dati0[i];
}
```



Problemi su sequenze di numeri

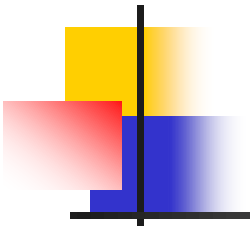
- I problemi interessano sequenze di dati (ordinati) che debbono essere immagazzinati in un vettore prima di venir elaborati, perchè:
  - è necessario attendere l'ultimo dato prima di poter elaborare i dati
  - oppure sono necessarie elaborazioni (ripetute) su tutti i dati
- Non si possono trattare sequenze infinite, ma insiemi (ordinati) finiti di dati, organizzati in vettori mono- o multi-dimensionali (matrici)



Normalizzazione di dati

Formulazione: scrivere una funzione C che:

- acquisisca da file testo una sequenza di dati reali separati da spazi o da a-capo:
  - il numero N di dati non è noto a priori ma può essere sovradimensionato (valore massimo 1000)
- determini, per l'i-esimo dato d_i ($0 \leq i < N$), le medie dei dati predecedenti (p_i) e dei successivi (s_i)
- scriva su un secondo file i dati, normalizzati secondo la seguente regola:
  - ogni dato (d_i) viene sostituito dalla media aritmetica tra d_i, p_i e s_i
- i nomi dei due file sono ricevuti come parametri.



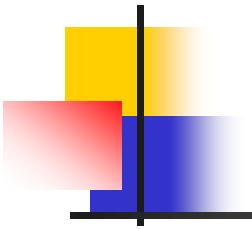
Soluzione:

- occorre calcolare, in modo iterativo, le medie, quindi, con una successiva iterazione, le normalizzazioni dei dati

Struttura dati:

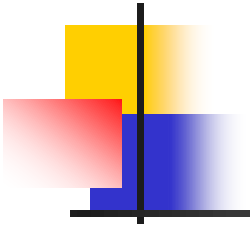
- un vettore per acquisire i dati dal file
- un vettore per calcolare le medie

Si potrebbero evitare i vettori rileggendo più volte il file (soluzione suggerita come esercizio).



Algoritmo:

- input: vettore statico sovradimensionato e lettura iterativa
- elaborazioni: è sufficiente calcolare
  - la sommatoria di tutti dati (STOT)
  - per ogni dato, la somma di se stesso e dei predecessori ($somme_i$). A partire da questa è possibile calcolare:
    $p_i = somme_{i-1}/(i)$,
    $s_i = (STOT-somme_i)/(N-i-1)$
  - i valori normalizzati (sostituendo i dati sul vettore iniziale):
    - per dati[0] non c'è il contributo della media dei predecessori, in quanto $p_0=0$
    - per dati[N-1] non c'è il contributo della media dei successori, in quanto $s_{N-1}=0$
    - per gli altri si itera tenendo conto di tutti e 3 i contributi
- output del vettore ricalcolato.



Esempio:

si supponga di ricevere i seguenti 6 valori:

- 4.0 5.0 3.0 2.0 1.0 7.0

Il vettore somme conterrà:

- 4.0 9.0 12.0 14.0 15.0 22.0

STOT = 22.0

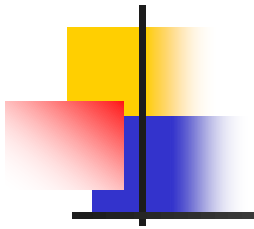
Il risultato sarà:

- dati[0] = (4.0 + (22.0-4.0)/5)/3 = 2.53
- dati[1] = (5.0 + 4.0/1 +(22.0-9.0)/4)/3 = 4.08
- dati[2] = (3.0 + 9.0/2 +(22.0-12.0)/3)/3 = 3.61
- dati[3] = (2.0 + 12.0/3 +(22.0-14.0)/2)/3 = 3.33
- dati[4] = (1.0 + 14.0/4 +(22.0-15.0)/1)/3 = 3.83
- dati[5] = (7.0 + 15.0/5)/3 = 3.33

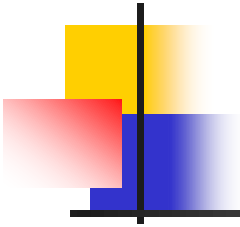
Codice

normalizza_sequenza.c

```
#define NMAX 1000
void normalizzaNum(char nfin[],char nfout[]){
   float dati[NMAX], somme[NMAX];
   int i, j, N, STOT;
   /* input */
   N = leggiDaFile(nfin,dati);
   /* calcola somme parziali */
   somme[0]=dati[0];
   for (i=1; i<N; i++)
      somme[i] = somme[i-1]+dati[i];
   STOT = somme[N-1];
   ...
}
```

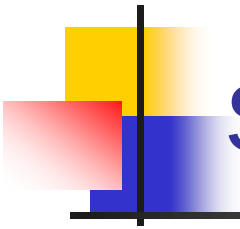


```
void normalizzaNum(char nfin[],char nfout[]){
   ...
   /* normalizza */
   dati[0] = (dati[0] + (STOT-dati[0])/(N-1))/3;
   for (i=1; i<N-1; i++)
     dati[i] = (dati[i] + somme[i-1]/i +
                (STOT-somme[i])/(N-i-1))/3;
   dati[N-1] = (dati[N-1] + somme[N-2]/(N-1))/3;
   /* output */
   scriviSuFile(nfout,dati,N);
}
```



Problemi su statistiche per gruppi

- I problemi sono caratterizzati dalla suddivisione dei numeri in classi/gruppi numerabili ed identificabili da un intero
- La raccolta di conteggi o dati statistici può essere effettuata sulle caselle di un vettore
- A ogni classe o gruppo corrisponde un indice (e una casella del vettore).



Suddivisione in classi

Formulazione: scrivere una funzione C che:

- riceva come parametro un vettore di interi di valore compreso tra 0 e 100 (il secondo parametro indica la dimensione del vettore)
- raggruppi gli interi in decine, calcoli e visualizzi i conteggi dei numeri appartenenti a ciascuna decina.

Esempio:

si supponga di ricevere i seguenti 20 valori:

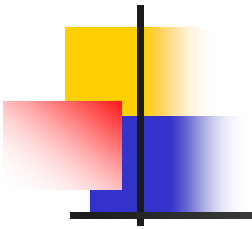
- 3 6 9 16 22 23 30 32 40 48 65 78 7 8 10 15 25 90 27 26

I numeri saranno raggruppati come segue:

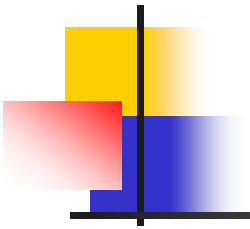
- (3, 6, 9, 7, 8), (16, 10, 15), (22, 23, 25, 26, 27), (30, 32), (40, 48), (-), (65), (78), (-), (90), (-)

I conteggi visualizzati saranno:

- 5, 3, 5, 2, 2, 0, 1, 1, 0, 1, 0



- Soluzione:
  - occorre utilizzare un vettore di contatori, sfruttando la corrispondenza indice-dato. Per ogni dato in ingresso si calcola l'indice corrispondente alla decina di appartenenza, e si aggiorna il relativo contatore
- Struttura dati:
  - un vettore di interi come parametro alla funzione
  - un vettore di contatori (con indici da 0 a 10).

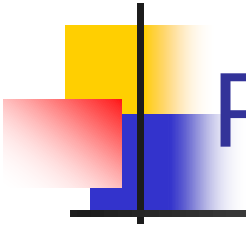


Algoritmo:

- azzeramento dei contatori, per ogni decina (le decine sono numerate da 0 a 10)
- iterazione sui dati interi. Per ognuno:
  - si calcola la decina di appartenenza con una divisione intera `d = dati[i]/10;`
  - si accede al vettore dei conteggi (utilizzando la decina come indice) incrementando il contenuto
- iterazione sui contatori per visualizzare i conteggi.

Codice

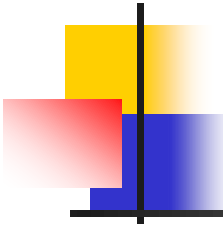

```
void contaPerDecine (int dati[], int n){
  int i, d, conta[11];
  for (i=0; i<=10; i++)
    conta[i]=0;
  for (i=0; i<n; i++) {
    d = dati[i]/10;
    conta[d] = conta[d]+1;
  }
  for (i=0; i<=10; i++)
    printf ("%d dati in decina %d \n",
             conta[i], i+1);
}
```



Problemi di codifica di numeri

Nei problemi di codifica di numeri, i vettori possono essere utilizzati per:

- immagazzinare le cifre in una data codifica
- manipolare i numeri, lavorando a livello di codifica in cifre.

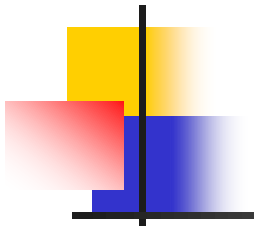


Codifica binaria di un intero

Formulazione: realizzare una funzione C che, ricevuto come parametro un intero ($0 <= n <= 2^{32} - 1$), ne determini la codifica binaria e visualizzi i bit.

Soluzione:

- costruzione iterativa dei bit, a partire dai meno significativi
- divisioni successive per 2, i resti delle divisioni sono i bit della codifica .



Struttura dati:

- un parametro formale (intero): n
- un vettore (bit) per i bit della codifica

Algoritmo:

- iterazione per generare, in bit, i bit, a partire dai meno significativi (al massimo 32 bit)
- iterazione per visualizzare i bit (dal più significativo).

Codice

codifica_binaria.c

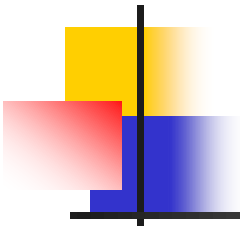
```
void binarioVettore (int n) {
  int i, bit[32];
  i=0;
  do {
    bit[i++] = n%2;
    n = n/2;
  } while (n>0);
  i--;
  while (i>=0) {
    printf("%d",bit[i--]);
  }
  printf("\n");
}
```

```
void binarioVettore (int n) {
  int i, bit[32];
  i=0;
  do {
    bit[i++] = n%2;
    n = n/2;
  } while (n>0);
  i--;
  while (i>=0) {
    printf("%d",bit[i--]);
  }
  printf("\n");
}
```

Calcola bit meno significativo

```
void binarioVettore (int n) {
  int i, bit[32];
  i=0;
  do {
    bit[i++] = n%2;
    n = n/2;
  } while (n>0);
  i--;
  while (i>=0) {
    printf("%d",bit[i--]);
  }
  printf("\n");
}
```

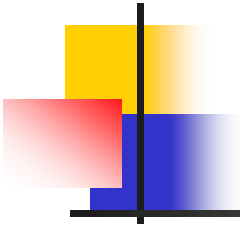
Continua finchè n non è 0
Se inizialmente n=0 calcola 1 bit



Problemi di codifica/decodifica

Problemi di codifica, ricodifica o crittografia applicati a testi, nei quali un vettore può essere utilizzato come:

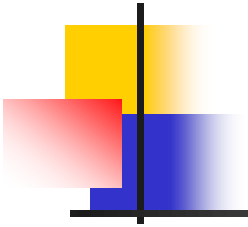
- tabella di codifica o ricodifica
  - mediante la corrispondenza indice-dato
  - come insieme di coppie dato-codice o codice0-codice1
- contenitore (ordinato o non) per dati intermedi.



Crittografia di un file di testo

Formulazione:

- crittografare il contenuto di un file testo, immagazzinando il risultato in un file risultato
- i codici dei caratteri vengono modificati, in base alla tabella di ricodifica contenuta in un file:
  - ogni riga del file contiene due numeri interi (compresi tra 0 e 255), che rappresentano, rispettivamente il codice ASCII di un carattere e il codice ASCII del carattere ricodificato
  - i codici non presenti non vanno modificati
  - la tabella garantisce l'univocità della ricodifica.

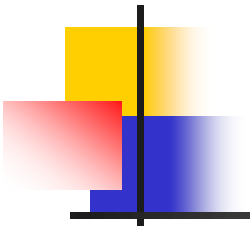


Soluzione:

- leggere la tabella da file, costruendo una vettore di ricodifica indice (codice iniziale) $\rightarrow$ dato (nuovo codice)
- leggere iterativamente i caratteri dal primo file
  - calcolare la ricodifica del carattere
  - scrivere il carattere sul file risultato

Struttura dati:

- tre variabili di tipo puntatore a `FILE` per gestire i due file in lettura e quello in scrittura; una stringa per i nomi dei file
- un vettore di caratteri per la tabella di ricodifica
- una variabile `char` per lettura e ricodifica dei caratteri



Algoritmo:

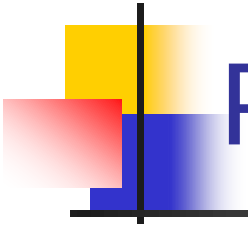
- acquisizione dei nomi di file e loro apertura
- inizializzazione della tabella di codifica (per codici invariati)
- lettura tabella di ricodifica
- iterazione di lettura di un carattere, ricodifica, scrittura nel secondo file
  - la ricodifica viene fatta mediante passaggio indice $\rightarrow$ dato nella tabella

Codice

crittografia.c

```
#define MAXRIGA 30
int main(void) {
  char ch, nomefile[MAXRIGA];
  char tabella[256];
  FILE *fpin, *fpout, *ftab;
  int i, nuovo;
  printf("nome file in ingresso: ");
  scanf("%s", nomefile);
  fpin = fopen(nomefile, "r");
  printf("nome file in uscita: ");
  scanf("%s", nomefile);
  fpout = fopen(nomefile, "w");
  ...
```

```
   ...
   printf("nome file tabella: ");
   scanf("%s", nomefile);
   ftab = fopen(nomefile, "r");
   for (i=0; i<256; i++)
     tabella[i] = (char)i;
   while (fscanf(ftab, "%d%d",&i,&nuovo)==2)
     tabella[i] = (char)nuovo;
   while (fscanf(fpin, "%c", &ch) == 1) {
     fscanf(fpin, "%c", &ch);
     fprintf(fpout, "%c",tabella[(int)ch]);
   }
   fclose(fpin);fclose(fpout);fclose(ftab);
}
```

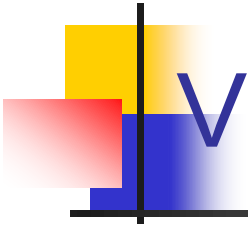


Problemi di text-processing

- Problemi nei quali occorre manipolare sequenze di caratteri e/o stringhe

  Esempio: costruzione o modifica di testo, creazione di messaggio in un dato formato

- Vettori (o matrici) di caratteri (o di stringhe) sono spesso necessari per:
  - generazione di testi a partire da regole o funzioni
  - trasformazione di testi esistenti.



Vettori e selezione su stringhe

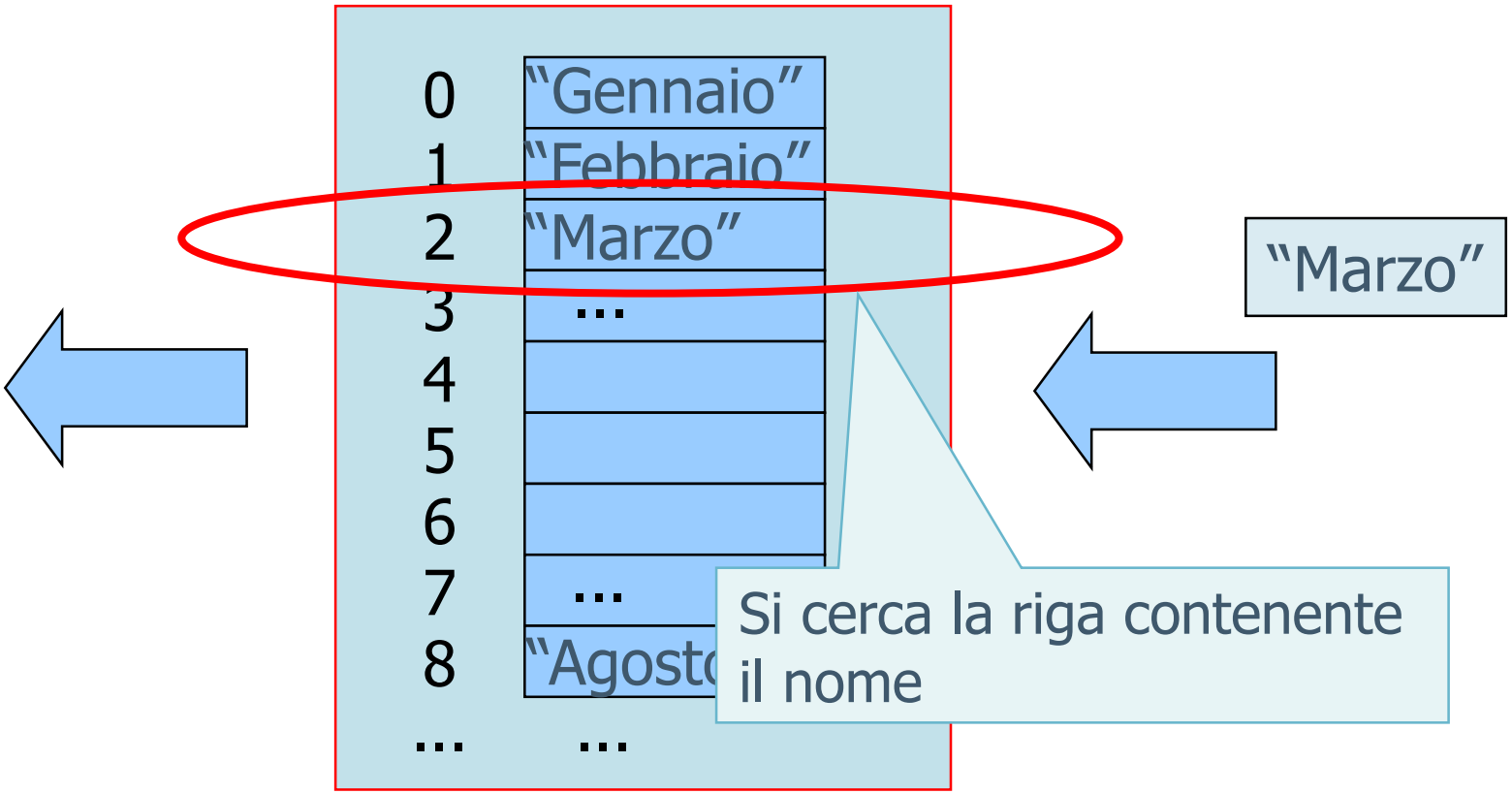
- La selezione basata direttamente su stringhe richiede confronti (`strcmp`) e costrutti condizionali `if` (non sono possibli `switch`)
- I vettori (usati come tabelle) possono consentire la traduzione da codici testuali a codici numerici, con cui:
  - è possibile la selezione mediante `switch`
  - si ottiene una migliore gestione/organizzazione dei casi da trattare
  - si ottengono informazioni più compatte, trasferibili tra moduli, come parametri o valori di ritorno di funzioni (es. codici di errore).

La tabella di conversione (A)

Per la conversione da stringa a intero si può utilizzare la corrispondenza dato → indice

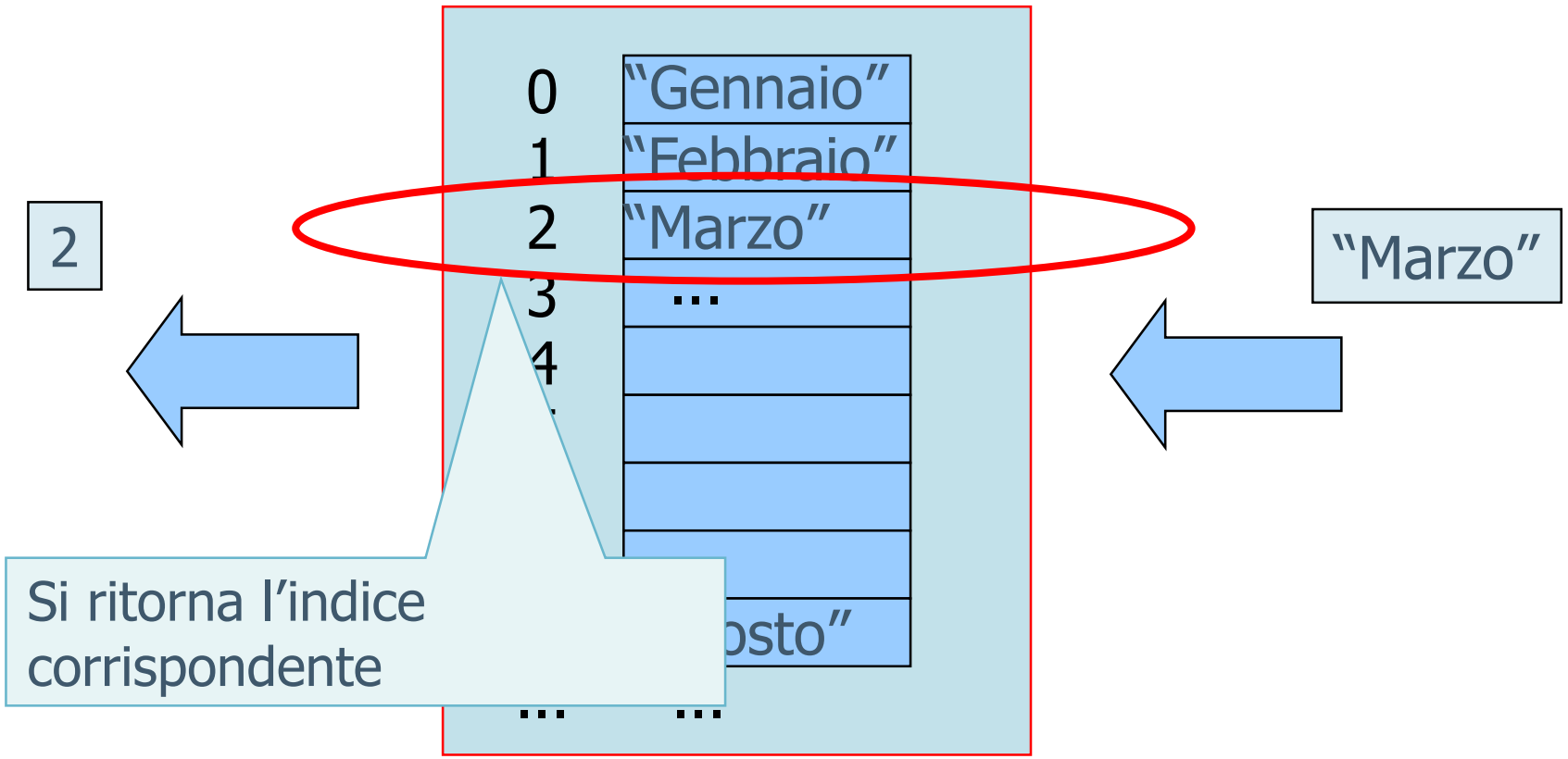
La tabella di conversione (A)

Per la conversione da stringa a intero si può utilizzare la corrispondenza dato → indice



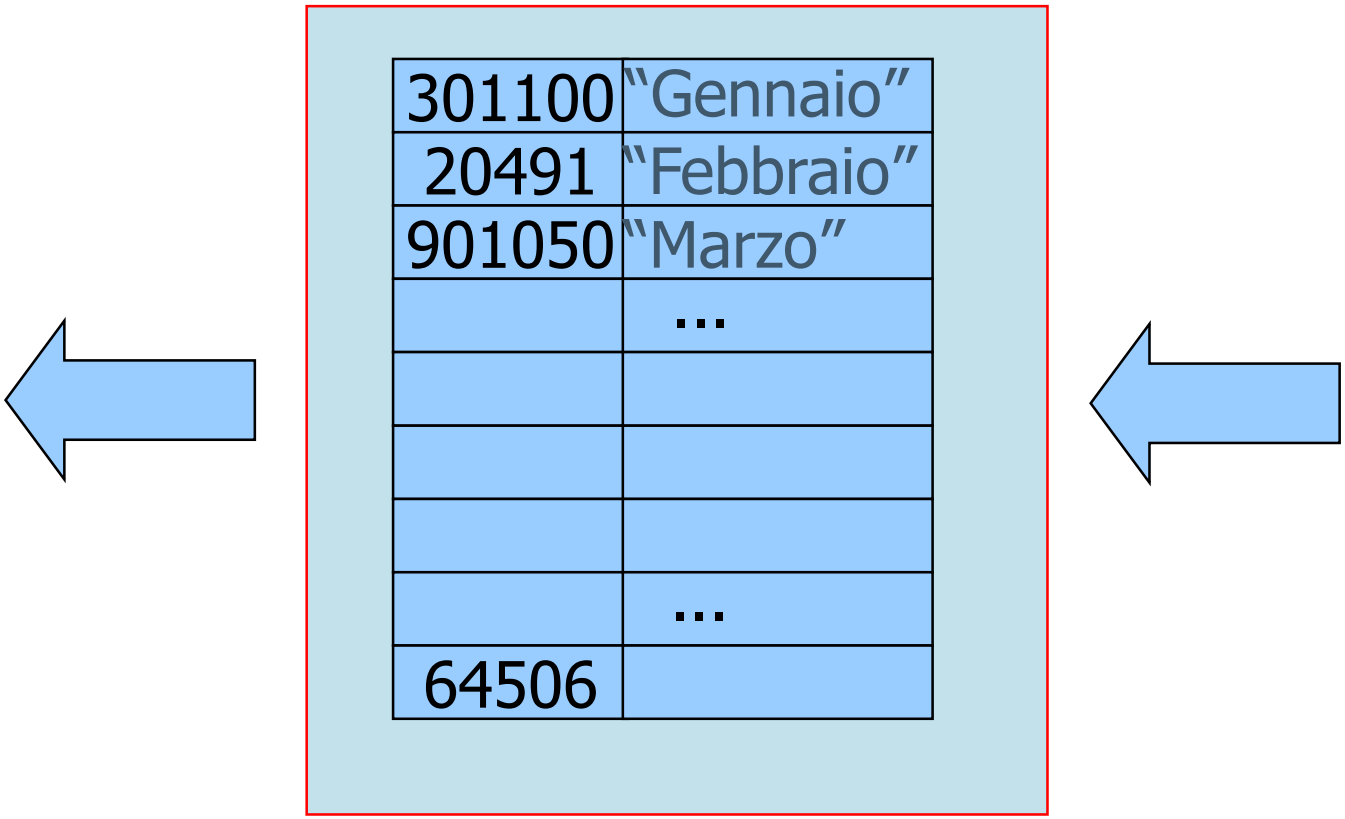
La tabella di conversione (A)

Per la conversione da stringa a intero si può utilizzare la corrispondenza dato → indice



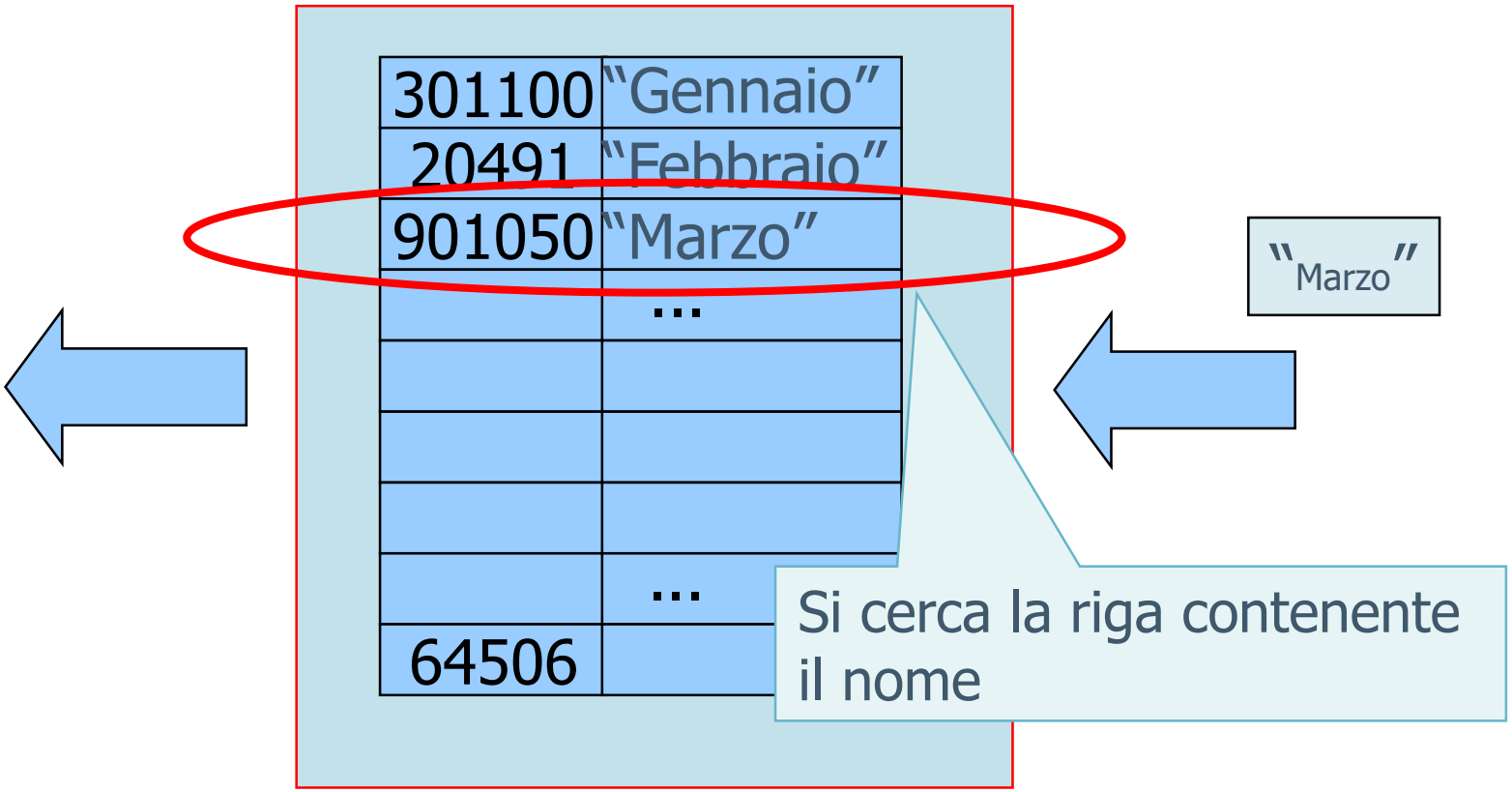
La tabella di conversione (B)

Se i valori interi sono grandi (non adatti come indici) si può realizzare un vettore di struct (codice,nome)



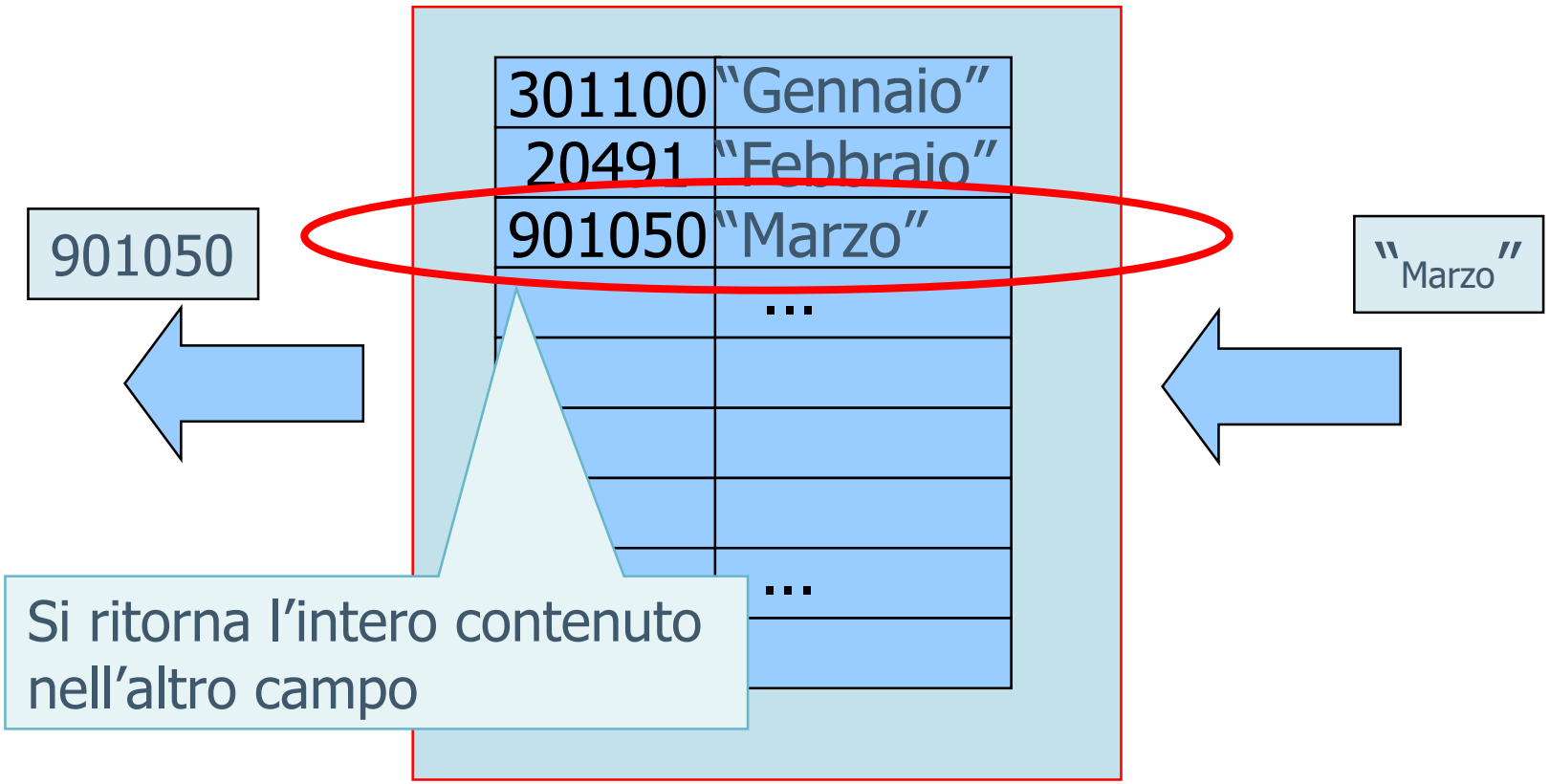
La tabella di conversione (B)

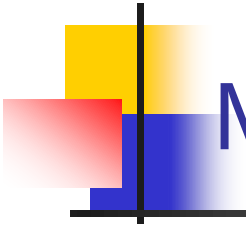
Se i valori interi sono grandi (non adatti come indici) si può realizzare un vettore di struct (codice,nome)



La tabella di conversione (B)

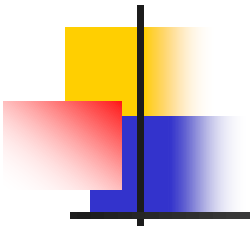
Se i valori interi sono grandi (non adatti come indici) si può realizzare un vettore di struct (codice,nome)





Menu con scelta su una parola

- Formulazione: scrivere una funzione che, iterativamente, acquisisca da tastiera una stringa (al massimo 50 caratteri, contenente eventuali spazi):
  - la prima parola diversa da spazio costituisce il selettore
  - se la parola è “fine”, occorre terminare l’iterazione
  - se la parola è uno tra “cerca”, “modifica”, “stampa” (ignorare differenza maiuscole/minuscole), occorre attivare, rispettivamente, le funzioni cerca, sostituisci, stampa, passando loro come parametro il resto della stringa (oltre la parola di selezione)
  - ogni altra parola va segnalata come errata.



- Soluzione: corrispondenza dato-indice
- Modularizzazione:
  - definizione costanti enumerative (per codici)
  - vettore di stringhe (tabella)
  - funzione di input e conversione da stringa a codice
- Gestione costanti mediante tipo enum:
  - il tipo enum è assimilabile ad un intervallo di valori interi 0..N, con N = numero_di_valori-1
- Tabella:
  - vettore inizializzato mediante costanti stringa
- Conversione da stringa a codice:
  - iterazione di ricerca su vettore

Codice

menu_parola.c

```
const int MAXL=51;
const int MAXC=11;
typedef enum {
   c_cerca, c_modifica, c_stampa, c_fine, c_err
} t_comandi;

void menuParola (void) {
   t_comandi codiceComando;
   char riga[MAXL];
   int continua=1;
   while (continua) {
      codiceComando = leggiComando();
      gets(riga); /* resto della riga */
      ...
   }
}
```

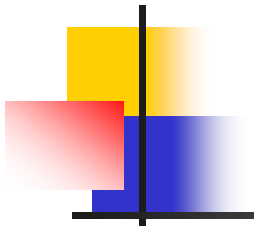
```
const int MAXL=51;
const int MAXC=11;
typedef enum {
   c_cerca, c_modifica, c_stampa, c_fine, c_err
} t_comandi;

void menuPa
   t_comand
   char riga
   int cont
   while (continua) {
      codiceComando = leggiComando();
      gets(riga); /* resto della riga */
      ...
   }
}
```

Definizione tipo enumerativo per i codici di comando corrispondenti agli interi da 0 a 4

```
const int MAXL=51;
const int MAXC=11;
typedef enum {
   c_cerca, c_modifica, c_stampa, c_fine, c_err
} t_comandi;

void menuParola (v
   t_comandi codiceComa
   char riga[MAXL];
   int continua=1;
   while (continua) {
      codiceComando = leggiComando();
      gets(riga); /* resto della riga */
      ...
   }
}
```

Lettura comando e conversione da stringa a codice

```
while (continua) {
   ...
   switch (codiceComando) {
     case c_cerca: cerca(riga);
       break;
     case c_modifica: sostituisci(riga);
       break;
     case c_stampa: stampa(riga);
       break;
     case c_fine: continua=0;
       break;
     case c_err:
     default: printf("comando errato\n");
   }
}
```



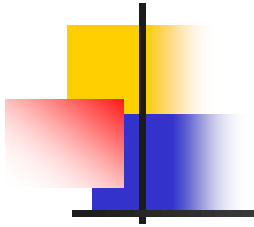
```
t_comandi leggiComando (void) {
  t_comandi c;
  char cmd[MAXC];
  char tabella[c_err][MAXC] = {
    "cerca", "modifica", "stampa", "uscita"
  }
  printf("comando (cerca/modifica");
  printf("/stampa/uscita): ");
  scanf("%s",cmd); strToLower(cmd);
  c=c_cerca;
  while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
  return (c);
}
```

```
t_comandi leggiComando (void) {
  t_comandi c;
  char cmd[MAXC];
  char tabella[c_e
    "cerca", "modi
  }
  printf("comando (cer      difica");
  printf("/stampa/usci      . ");
  scanf("%s",cmd); st     oLower(cmd);
  c=c_cerca;
  while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
  return (c);
}
```

Problema di selezione/ricerca
(vedi sezione successiva)

Elaborazione testi a livello carattere

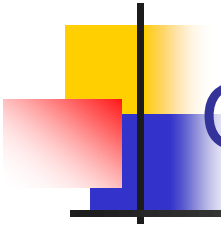
- Un testo può essere costruito o modificato a livello di caratteri utilizzando un vettore o una matrice come:
  - rappresentazione (a caratteri) del testo da esaminare
  - area dati temporanea per costruire o manipolare una stringa o una matrice di caratteri.



- Elaborazione di parole o frasi come:
  - una parola (o una frase), può essere analizzata a livello di singoli caratteri
  - un vettore si rende necessario se occorre accedere direttamente ai caratteri

Esempi:

- verifica di palindromia
- taglia e incolla parte di stringa da una prima ad una seconda collocazione
- ricerca/sostituzione di sottostringa.



Controllo di palindromia

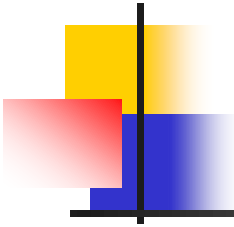
- Formulazione: si realizzi una funzione C in grado di verificare se una stringa, ricevuta come parametro, sia o meno palindroma (trascurando la differenza tra caratteri maiuscoli e minuscoli):
  - una parola si dice palindroma se letta dall'ultimo al primo carattere, risulta invariata

    Esempi: Anna, madam, otto, abcdefgFEDCBA
- Algoritmo:
  - iterazione di confronto tra caratteri corrispondenti (primo-ultimo, secondo-penultimo, …)

- Strutture dati:
  - il vettore è la stringa stessa ricevuta come parametro
  - due indici identificano i caratteri da confrontare
  - un flag implementa la quantificazione

Codice

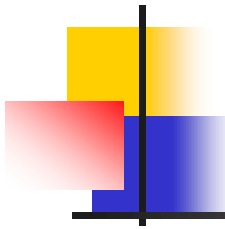

```
int palindroma (char parola[], int n) {
  int i, pal=1;

  for (i=0; i<n/2; i++)
    if (toupper(parola[i]) !=
        toupper(parola[n-1-i]))
      pal = 0;
  return pal;
}
```



Costruzione di figure/grafici

Le figure o grafici rappresentati su video visto come matrice di caratteri di 25 righe e 80 colonne, possono essere preparate su una matrice di caratteri:

- ciò consente di costruire la figura senza rispettare la successione di righe che caratterizza l'output sequenziale (su video o su file testo)
- la figura viene preparata su una matrice di caratteri, sfruttando l'accesso diretto ad ogni casella
- la figura viene successivamente stampata seguendo la successione sequenziale tra righe.



Visualizzazione di parabola

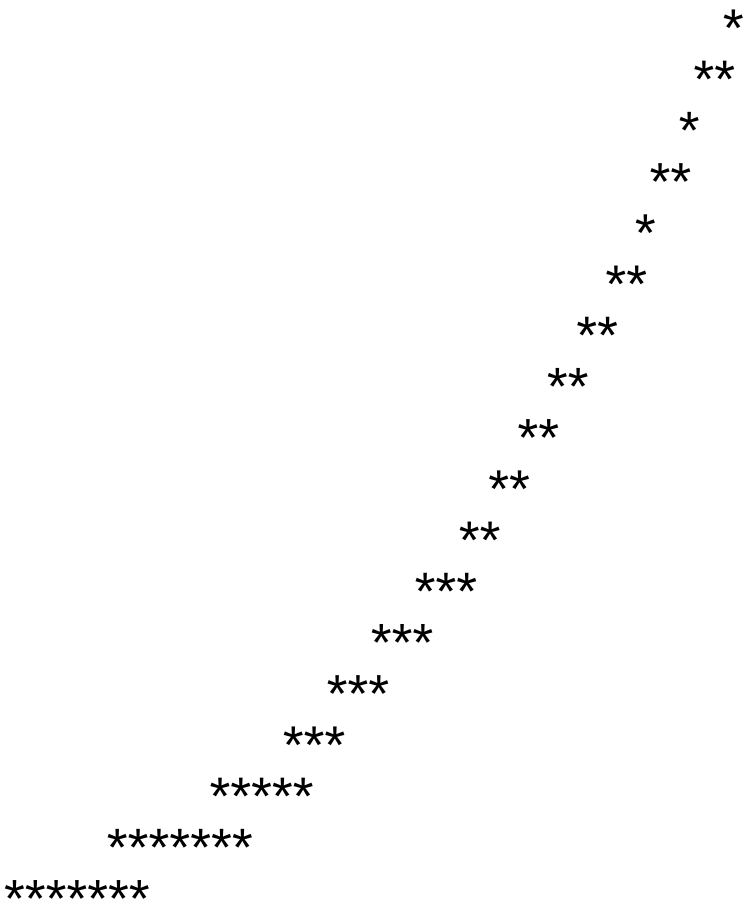
- Formulazione:
  - data la parabola di equazione

$$y = ax^2 + bx + c = 0$$

  - si scriva un programma che:
    - acquisisca da tastiera i coefficienti a, b, c, e i valori degli estremi (xmin, xmax) e (ymin, ymax), rispettivamente di un intervallo per le ascisse e per le ordinate
    - stampi, in un rettangolo di 20 righe per 70 colonne, un grafico (con asse delle ascisse orizzontale) che rappresenti la funzione nel rettangolo del piano cartesiano compreso negli intervalli [xmin,xmax], [ymin,ymax]

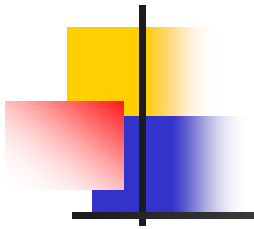
Esempio: se si acquisissero da tastiera i valori:

- a=1.0, b=2.0, c=1.0, x0=-1.0, xn=4.0, ymin=-1.0, ymax=10.0 il contenuto del file sarebbe:



- Struttura dati: sono sufficienti variabili scalari, per rappresentare
  - coefficienti: a, b, c (`float`)
  - intervalli: xmin, xmax, ymin, ymax (`float`)
  - dati intermedi: passoX passoY (lunghezza degli intervalli), x, y (`float`)
  - indici: i, j (`int`)

È necessaria una matrice (di `char`) per costruire il grafico.



- Algoritmo:
  - input dati e calcolo passo (= lunghezza intervalli)
  - inizializzazione a tutti spazi della matrice di caratteri
  - iterazione su valori di x
    - calcolo y(x)
    - se è nell'intervallo [ymin,ymax] converti in intero (j) e assegna '*' nella matrice
  - iterazioni su righe e colonne, per stampare matrice.

Codice

visualizza_parabola.c

```
#include <stdio.h>
#include <math.h>
const int NR=20, NC=80;
int main(void) {
  float a, b, c, x, y, passoX, passoY,
        xmin, xmax, ymin, ymax;
  int i, j;
  char pagina[NR][NC];
  FILE *fpout = fopen("out.txt","w");
  printf("Coefficienti (a b c): ");
  scanf("%f%f%f",&a,&b,&c);
  printf("Intervallo ascisse (xmin xmax): ");
  scanf("%f%f",&xmin,&xmax);
  printf("Intervallo ordinate (ymin ymax): ");
  scanf("%f%f",&ymin,&ymax);
```

```
/* inizializza matrice */
for (i=0; i<NR; i++)
  for (j=0; j<NC; j++)
    pagina[i][j] = ' ';
passoX = (xmax-xmin)/(NC-1);
passoY = (ymax-ymin)/(NR-1);
/* calcola punti della parabola */
for (j=0; j<NC; j++) {
  x = xmin + j*passoX;
  y = a*x*x + b*x + c;
  if (y<ymin || y>ymax)
    continue;
  i = (y-ymin)/passoY;
  pagina[i][j] = '*';
}
```

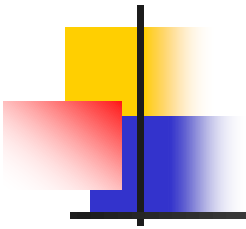
```
  /* stampa matrice per righe */
  for (i=NR-1; i>=0; i--) {
    for (j=0; j<NC; j++)
      fprintf(fpout,"%c",pagina[i][j]);
    fprintf(fpout,"\n");
  }
  fclose(fpout);
}
```

Elaborazione testi a livello di stringhe

Un testo può essere costruito o modificato a livello di stringhe se:

- è possibile identificare sottostringhe (sequenze di caratteri) sulle quali applicare operazioni di tipo unitario
- le operazioni su stringhe debbono trovarsi in libreria, oppure essere chiamate funzioni realizzate dal programmatore.

Un vettore può essere necessario per costruire o immagazzinare temporaneamente stringhe da elaborare.



Formattazione di testo

- Formulazione:
  - è dato un file testo, le cui righe sono scomponibili in sottostringhe (di non più di 20 caratteri) separate da spazi (oppure '\t' o '\n')
  - si realizzi una funzione C che, letto il file, ne copi il contenuto in un altro file, dopo aver:
    - ridotto le sequenze di più spazi ad un solo spazio
    - inserito (in sostituzione di spazi) o eliminato caratteri a-capo ('\n') in modo tale che ogni riga abbia la massima lunghezza possibile, minore o uguale a lmax (terzo parametro della funzione)
    - centrato il testo rispetto alla lunghezza lmax.

- Soluzione:
  - la soluzione è simile al problema del Cap. 3 senza centratura del testo
  - per effettuare la centratura occorre tutta una riga di testo prima di stamparla (per calcolare il numero di spazi da stampare):
    - si può utilizzare un vettore come buffer
    - la centratura (su `lmax` caratteri) di una riga di `l` caratteri, si effettua stampando, prima della riga, `(lmax-l)/2` spazi

Codice

```
const int STRLEN=21;
const int LMAX=255;
```

variabili globali

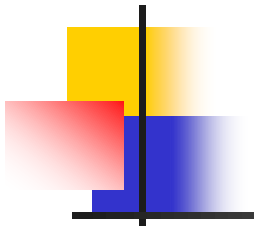
formattazione_testo.c

```
void format(char nin[],char nout[],int lmax){
   FILE *fin=fopen(nin, "r");
   FILE *fout=fopen(nout, "w");
   char parola[STRLEN], riga[LMAX];
   int i,l;

   l=0;
   while (fscanf(fin,"%s",parola)==1) {
      ...
   }
   fclose(fin); fclose(fout);
}
```

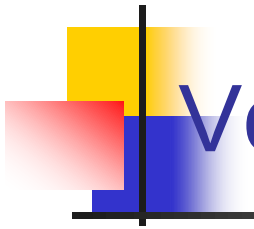
```
while (fscanf(fin,"%s",parola)==1) {
  if (l+1+strlen(parola) > lmax) {
    for (i=0; i<(lmax-l)/2; i++)
      fprintf(fout, " ");
    fprintf(fout, "%s\n",riga);
    strcpy(riga,parola);
    l=strlen(parola);
  }
  else {
    strcat(riga, " "); strcat(riga,parola);
    l+=1+strlen(parola);
  }
}
```

Problemi di verifica e selezione

- I problemi di verifica consistono nel decidere se un insieme di informazioni o dati rispettano un determinato criterio di accettazione
- Selezionare significa verificare i dati e scegliere quelli che corrispondono al criterio di verifica
- La ricerca è una delle modalità di selezione:
  - spesso si cerca il dato che corrisponde a un criterio
  - talvolta i dati possono essere molteplici.

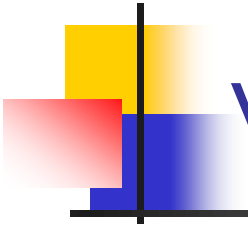


- I vettori possono essere utilizzati:
  - come contenitori per l'insieme di dati, su cui applicare il criterio di verifica
  - come insieme dei dati tra i quali ricercare/selezionare.



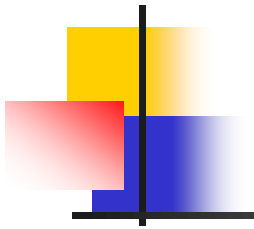
Verifiche su sequenze

- Verificare una sequenza di dati significa decidere se la sequenza rispetta un criterio di accettazione
- Un vettore può essere necessario nel caso di criterio di accettazione che richieda elaborazioni su tutti i dati.



Verifica di dati ripetuti

- Formulazione:
  - un file testo contiene una sequenza di dati numerici (reali)
    - la prima riga del file indica (mediante un intero) quanti sono i dati nella sequenza
    - seguono i dati, separati da spazi o a-capo
  - si scriva una funzione C che, ricevuto come parametro il puntatore al file (già aperto), verifichi che ogni dato sia almeno ripetuto una volta nella sequenza
    - un dato si considera ripetuto se, nella sequenza, se ne trova almeno un altro tale che la loro differenza, in valore assoluto, sia inferiore a 1%

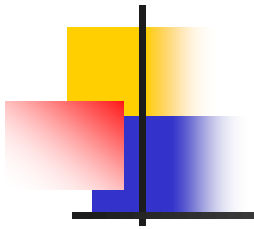


- Soluzione:
  - analizzare i dati, mediante una doppia iterazione, verificando, per ognuno, che ne esista almeno uno uguale (differenza ≤ 1% rispetto al massimo tra i due in valore assoluto)
- Struttura dati:
  - vettore per contenere i dati letti da file
  - variabili scalari: indici, contatore e flag
- Algoritmo:
  - acquisizione dati su vettore statico sovradimensionato
  - verifica mediante doppia iterazione
  - quantificazione con flag.

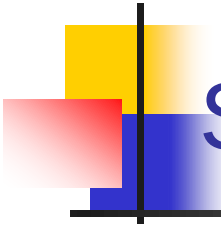
Codice

dati_ripetuti_float.c

```
int datiRipetuti (FILE *fp) {
  float dati[MAXDATI];
  int ndati, i, j, ripetuto;
  fscanf(fp, "%d",&ndati);
  for (i=0; i<ndati; i++)
    fscanf(fp, "%f",&dati[i]);
  for (i=0; i<ndati; i++) {
    ripetuto = 0;
    for (j=0; j<ndati; j++)
      if (i!=j && simili(dati[i],dati[j]))
        ripetuto=1;
    if (!ripetuto) return 0;
  }
  return 1;
}
```



```
int simili (float a, float b) {
   if (fabs(a)>fabs(b))
     return (fabs(a-b)/fabs(a) < 0.01);
   else
     return (fabs(a-b)/fabs(b) < 0.01);
}
```



Selezione di dati

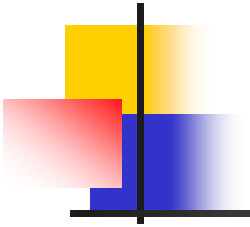
- Contestualmente alla verifica di più dati (o sequenze/insiemi) di dati, è possibile discriminare i dati (o il dato) che corrispondono al criterio di verifica rispetto agli altri
- La selezione può essere vista come una variante della verifica:
  - i dati vengono dapprima verificati
  - quelli (o quello) che corrispondono al criterio di accettazione vengono scelti
- La ricerca è un caso particolare di selezione:
  - si seleziona il dato (se esiste) che corrisponde al criterio di ricerca.

Conversione matricola→nome

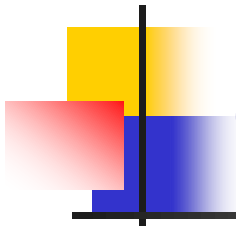
- Formulazione:
  - si realizzi una funzione C in grado di determinare il nome di uno studente, a partire dal numero di matricola (primo parametro alla funzione)

Siccome i numeri di matricola sono grandi (6 cifre decimali, MMAX) non si vuole sfruttare la corrispondenza indice-dato in un vettore. Le matricole sono rappresentate mediante stringhe. Si suppone che il nome abbia lunghezza massima NMAX.

- La tabella di conversione è fornita, come secondo parametro alla funzione, mediante un vettore di `struct`, aventi come campi (stringhe) numero di matricola e nome. Il terzo parametro (intero) è la dimensione della tabella.

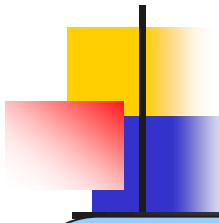


- Soluzione:
  - analizzare iterativamente i dati nel vettore, confrontando di volta in volta la matricola corrente con quella richiesta
- Struttura dati:
  - la tabella è un vettore (fornito come parametro)
  - il numero di matricola richiesto (secondo parametro) è una stringa
- Algoritmo:
  - la ricerca consiste in una verifica dei dati
  - la funzione ricerca la matricola e ritorna l'indice, in tabella, dello studente che corrisponde al numero di matricola (-1 per indicare matricola non trovata)



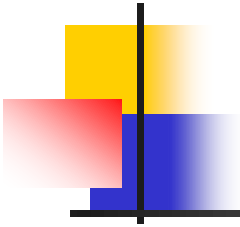
Codice

```
typedef struct {
   char matricola[MMAX], nome[NMAX];
} t_stud;
...
int matrNome(char m[],t_stud tabella[],int n){
   int i;

   for (i=0; i<n; i++) {
      if (strcmp(m,tabella[i].matricola)==0)
         return(i);
   }
   return -1;
}
```



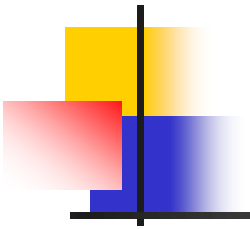
Codice

```
int main(void) {
  t_stud studenti[SMAX];
  int nstud, ind;
  char matr[NMAX];
  ...
  while (...) {
    ...
    scanf("%s", matr);
    ind = matrNome(matr,studenti,nstud);
    if (ind<0)
      printf("matr: %s non trovata\n", matr);
    else
      printf("studente con matricola %s %s\n",
        matr, studenti[i].nome);
  }
}
```



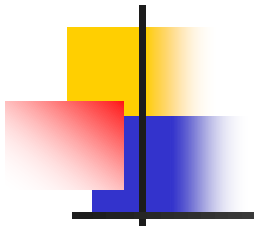
Problemi di ordinamento

- Un problema di ordinamento consiste nella richiesta di permutare una sequenza di dati, in modo tale che (dopo la permutazione) sia verificato un criterio di ordinamento
- Per ordinare dei dati in modo totale, si opera molto spesso su un vettore, adatto a fornire una successione lineare di dati, con valori crescenti (o decrescenti) secondo la progressione crescente degli indici.



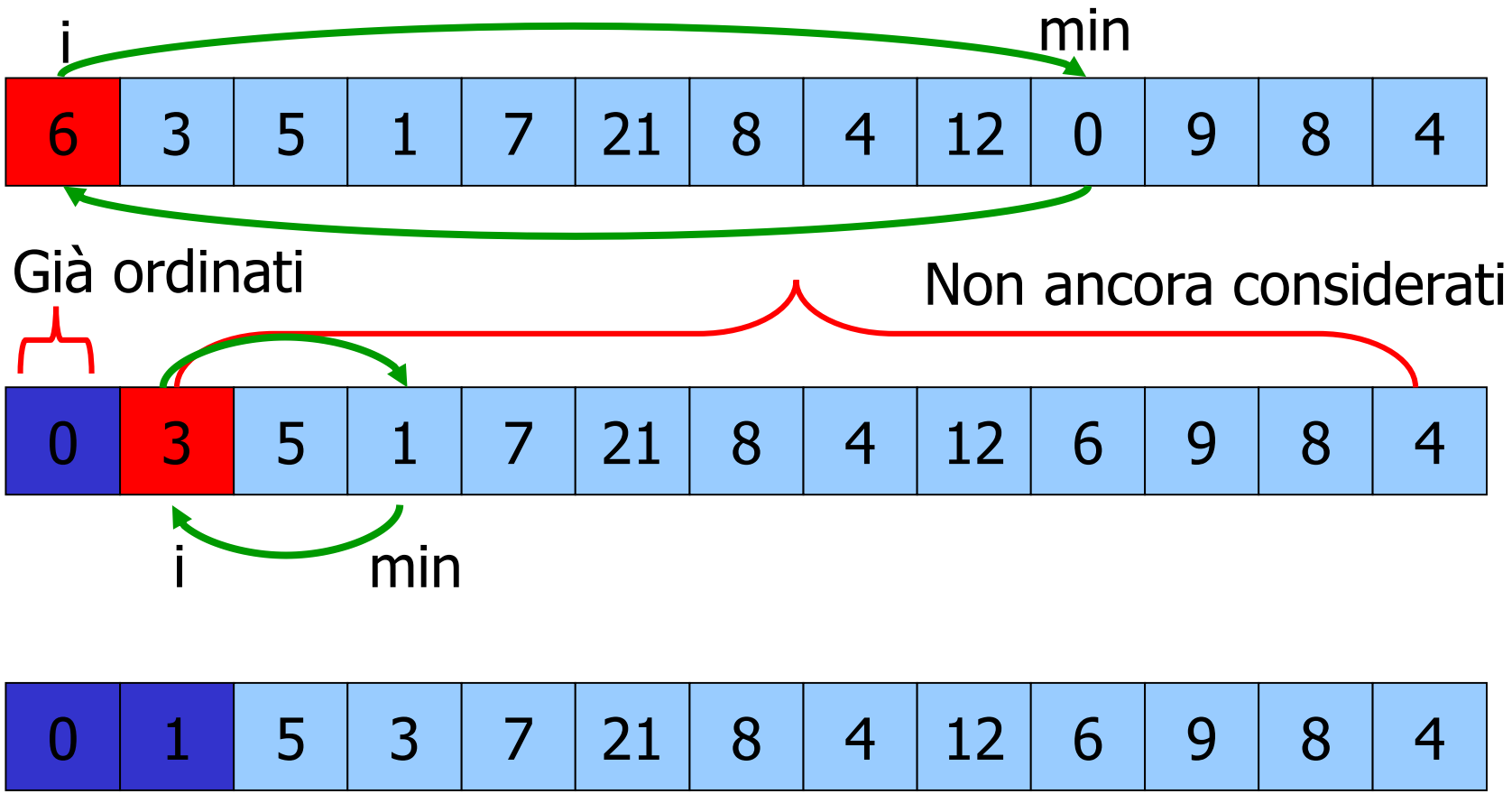
Selection sort

- Formulazione: si scriva una funzione C che:
  - ricevuti come parametri un vettore di numeri interi e la sua dimensione
  - ordini i dati in modo crescente con l'algoritmo di selection sort
- Soluzione: selection sort
  - algoritmo di ordinamento basato su ripetute ricerche di minimo

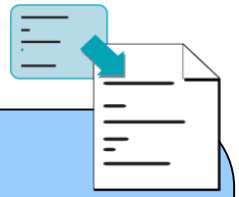


- Struttura dati e algoritmo:
  - dati: vettore A di N interi (A[0] ... A[N-1]), diviso in 2 sotto-vettori:
    - di sinistra: ordinato
    - di destra: disordinato
  - un vettore di un solo elemento è ordinato
  - approccio incrementale:
    - passo i: il minimo del sotto-vettore (A[i] ... A[N-1]) è assegnato a A[i]; incremento di i
  - Terminazione: tutti gli elementi inseriti ordinatamente.

Esempio

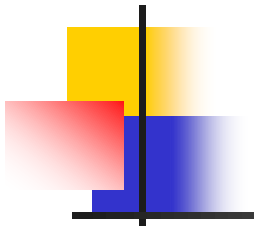

i
min
6 3 5 1 7 21 8 4 12 0 9 8 4
Già ordinati
Non ancora considerati
0 3 5 1 7 21 8 4 12 6 9 8 4
i
min
0 1 5 3 7 21 8 4 12 6 9 8 4

Codice



selection_sort.c

```
void selectionSort (int A[], int N) {
  int i, j, imin, temp;

  for (i=0; i<N-1; i++) {
    /*cerca indice del minimo in A[i]..A[N-1]*/
    imin = i;
    for (j = i+1; j < N; j++)
      if (A[j] < A[imin]) imin = j;
    /*scambia minimo con A[i]*/
    temp = A[i];
    A[i] = A[imin];
    A[imin] = temp;
  }
}
```



```
void selectionSort (int A[], int N) {
  int i, j, imin, temp;

  for (i=0; i<N-1; i++) {
    /*cerca indice del minimo in A[i]..A[N-1]*/
    imin = i;
    for (j = i+1;
      if (A[j] < A
    /*scambia minimo con A[i]*/
    temp = A[i];
    A[i] = A[imin];
    A[imin] = temp;
  }
}
```

Iterazione esterna, eseguita n-1 volte

```
void selectionSort (int A[], int N) {
  int i, j, imin, temp;

  for (i=0; i<N-1; i++) {
    /*cerca indice del minimo in A[i]..A[N-1]*/
    imin = i;
    for (j = i+1; j < N; j++)
      if (A[j] < A[imin]) imin = j;
    /*scambia minimo con A[i]*/
    temp = A[i];
    A[i] = A[imin];
    A[imin] = temp;
  }
}
```

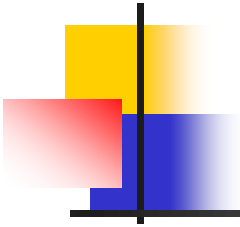
Iterazione interna, eseguita n-i-1 volte

```
void selectionSort (int A[], int N) {
  int i, j, imin, temp;

  for (i=0; i<N-1; i++) {
    /*cerca indice
    imin = i;
    for (j = i+1; j
      if (A[j] < A[im      imin = j;
    /*scambia minimo con A[i]*/
    temp = A[i];
    A[i] = A[imin];
    A[imin] = temp;
  }
}
```

Algoritmo in loco, perché scambia i dati sul vettore

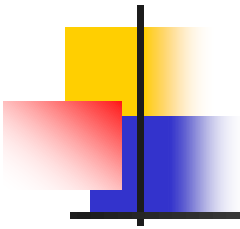
```
void selectionSort (int A[], int N) {
  int i, j, imin, temp;

  for (i=0; i<N-1; i++) {
    /*cerca indice del minimo in A[i]..A[N-1]*/
    imin = i;
    for (j = i+1; j < N; j++)
      if (A[j] < A[imin]) imin = j;
    /*scambia minimo con A[i]*/
    temp = A[i];
    A[i] = A[imin];
    A[imin] = temp;
  }
}
```

Algoritmo stabile, perché non scambia due dati di ugual valore



Sul libro

- Problemi numerici:
  - crivello di Eratostene (numeri primi)
- Problemi di codifica:
  - cifrario di Vigenère
- Problemi di verifica:
  - verifica di primalità
- Esercizi risolti:
  - prodotto matrici, somma in base B, cruciverba, eliminazione di spazi, eliminazione di valori nulli, bubble sort
- Esercizi proposti

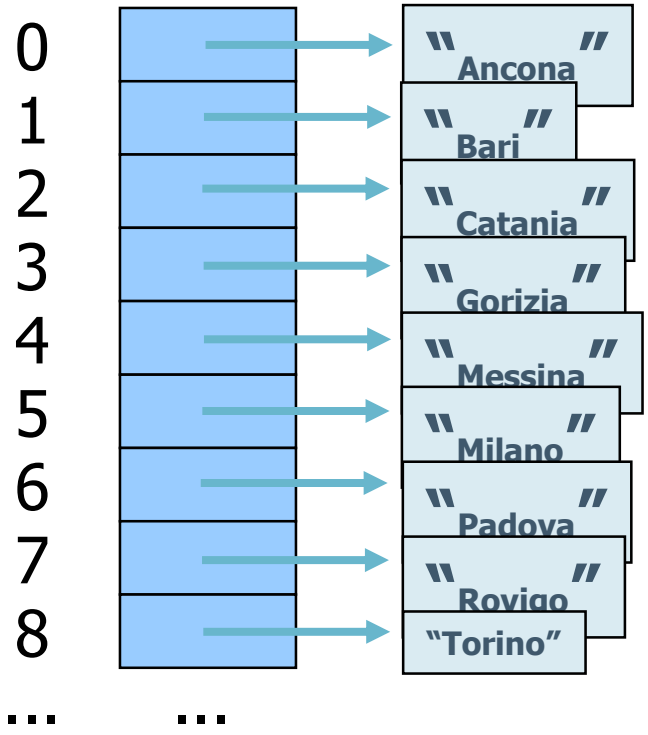


Nota sui puntatori

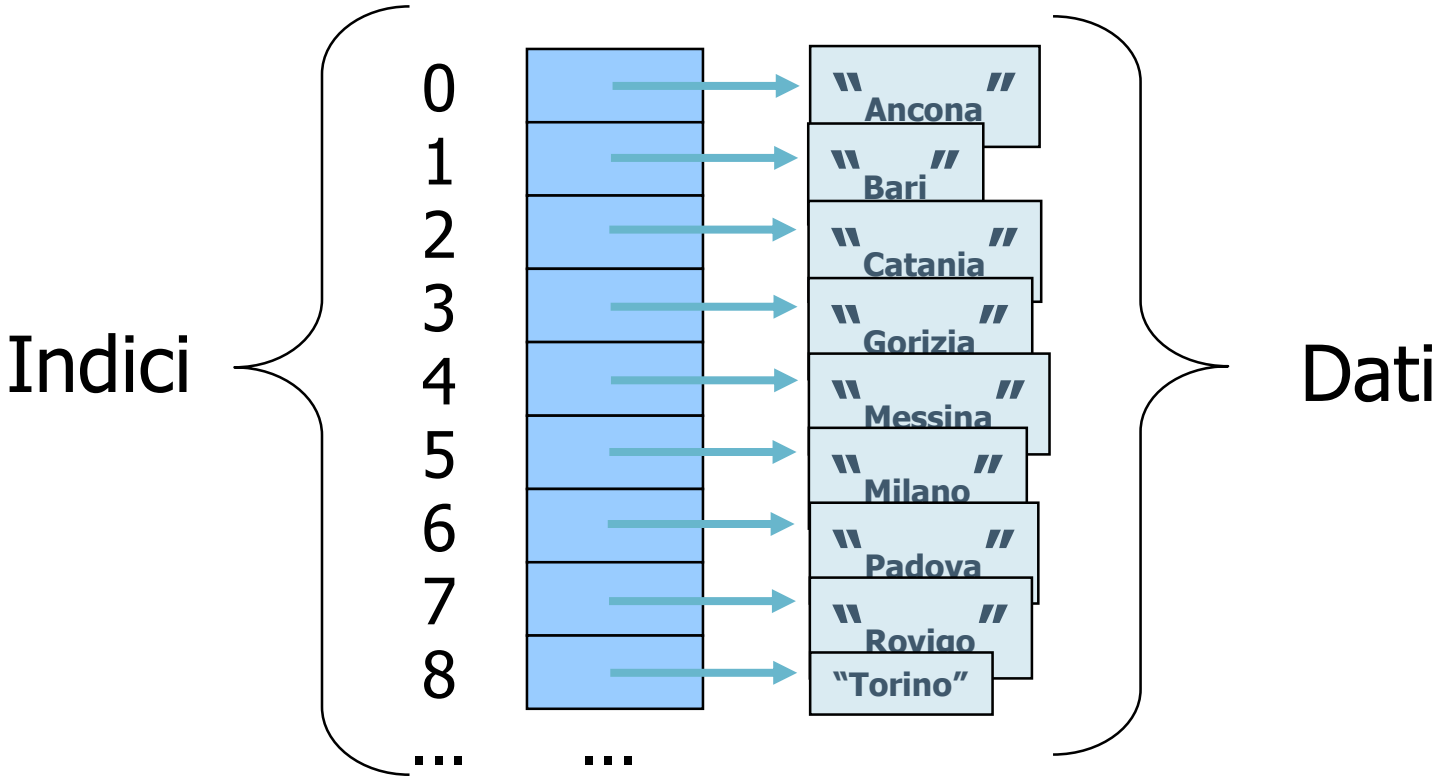
- Seguono le versioni con puntatori nella versione corrispondente al libro

Corrispondenza indice ↔ dato

Ad ogni indice (intero nell'intervallo 0..NDATI-1) corrisponde un dato (e viceversa):



Ad ogni indice (intero nell'intervallo 0..NDATI-1) corrisponde un dato (e viceversa)

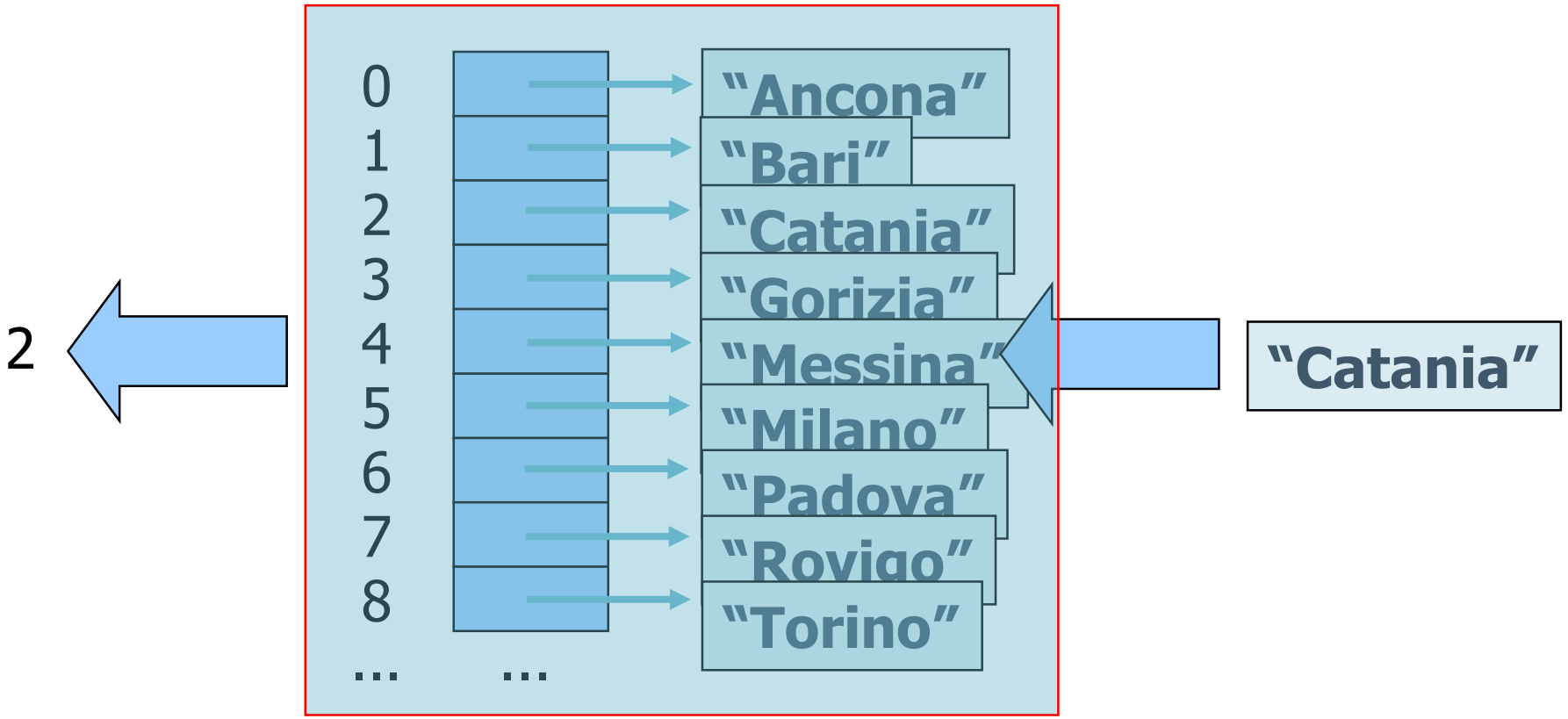


Da indice a dato

A partire dall'indice, calcolare il dato. Operazione immediata, grazie al meccanismo di accesso ai vettori:

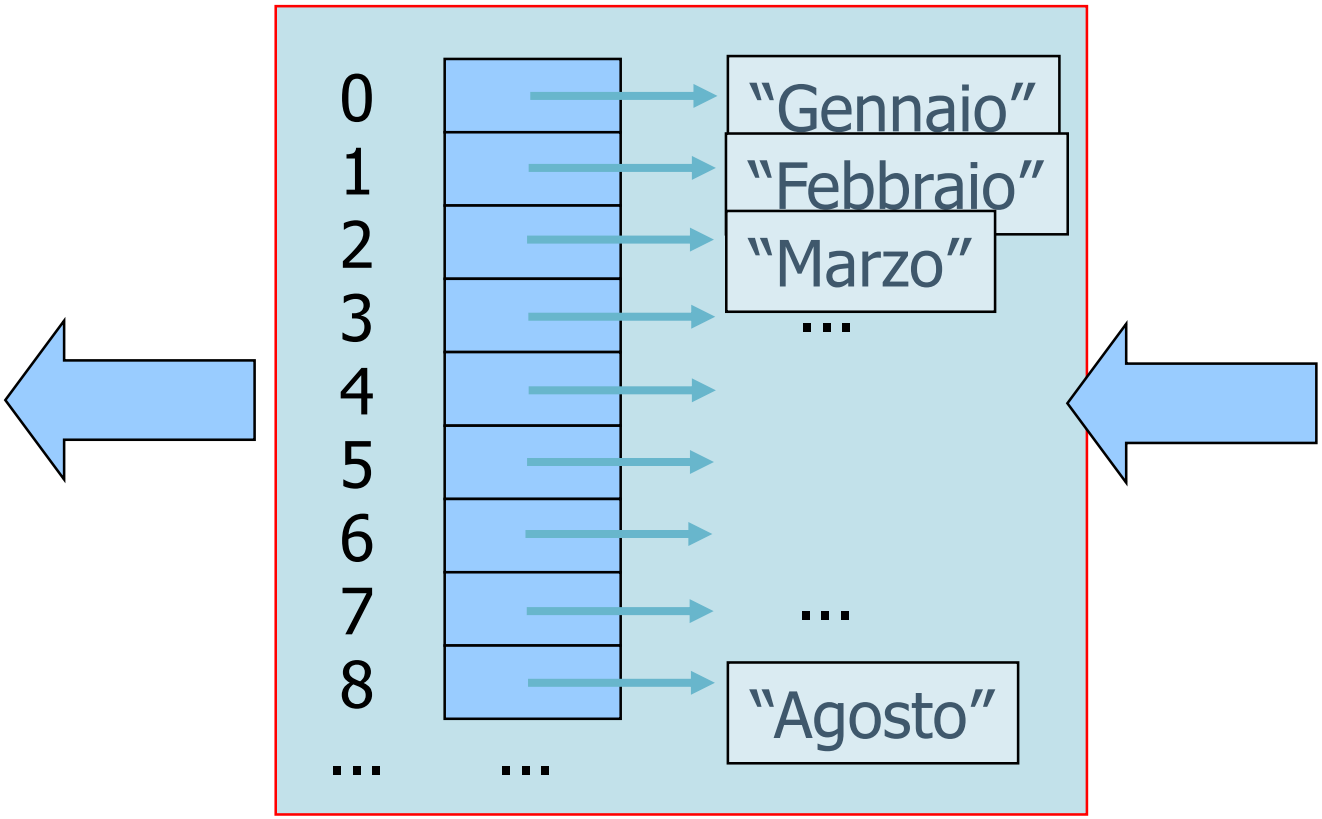
Da dato a indice

A partire dal dato, calcolarne l'indice. Operazione non immediata, in quanto occorre cercare il dato:

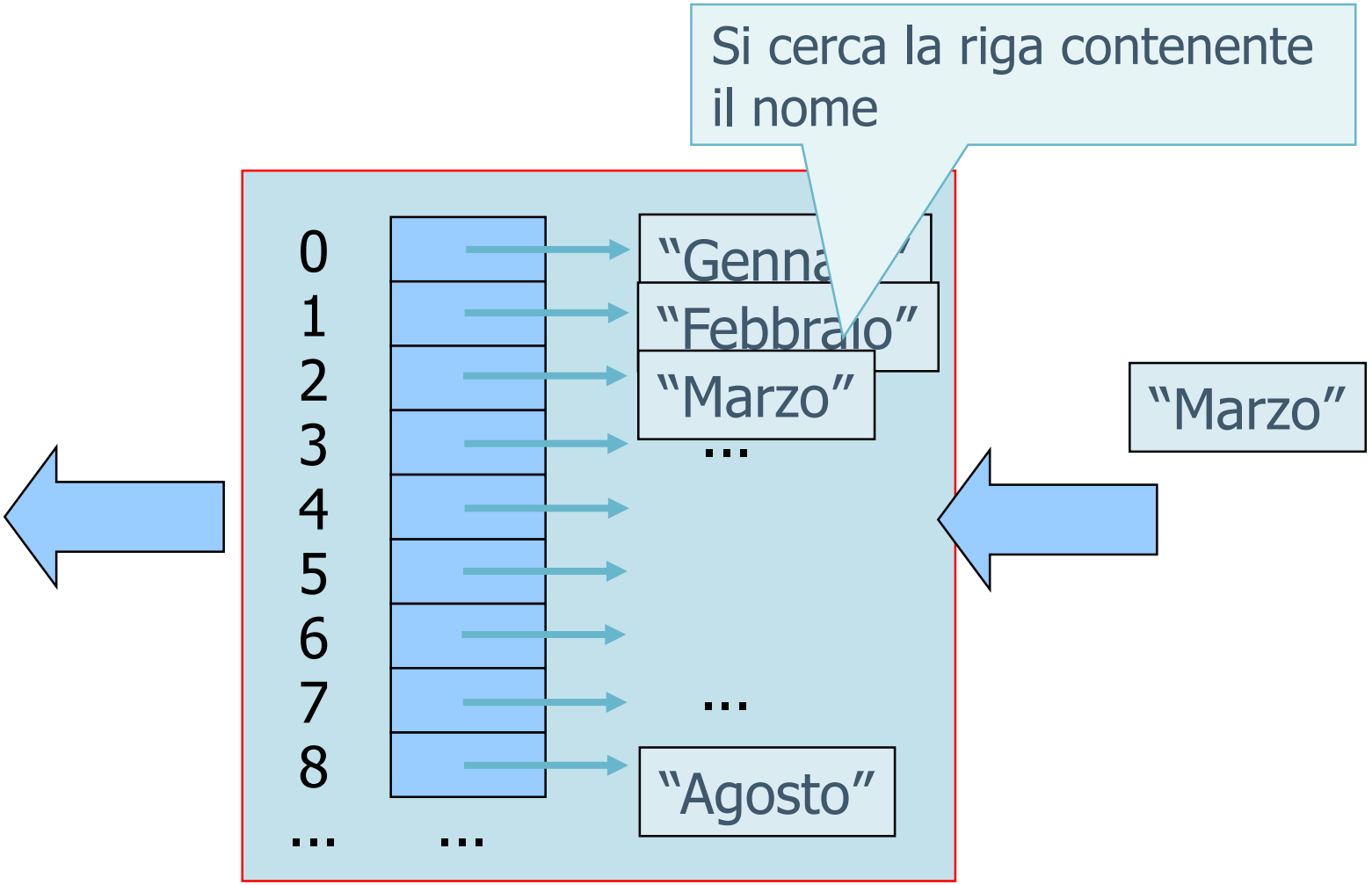


La tabella di conversione (A)

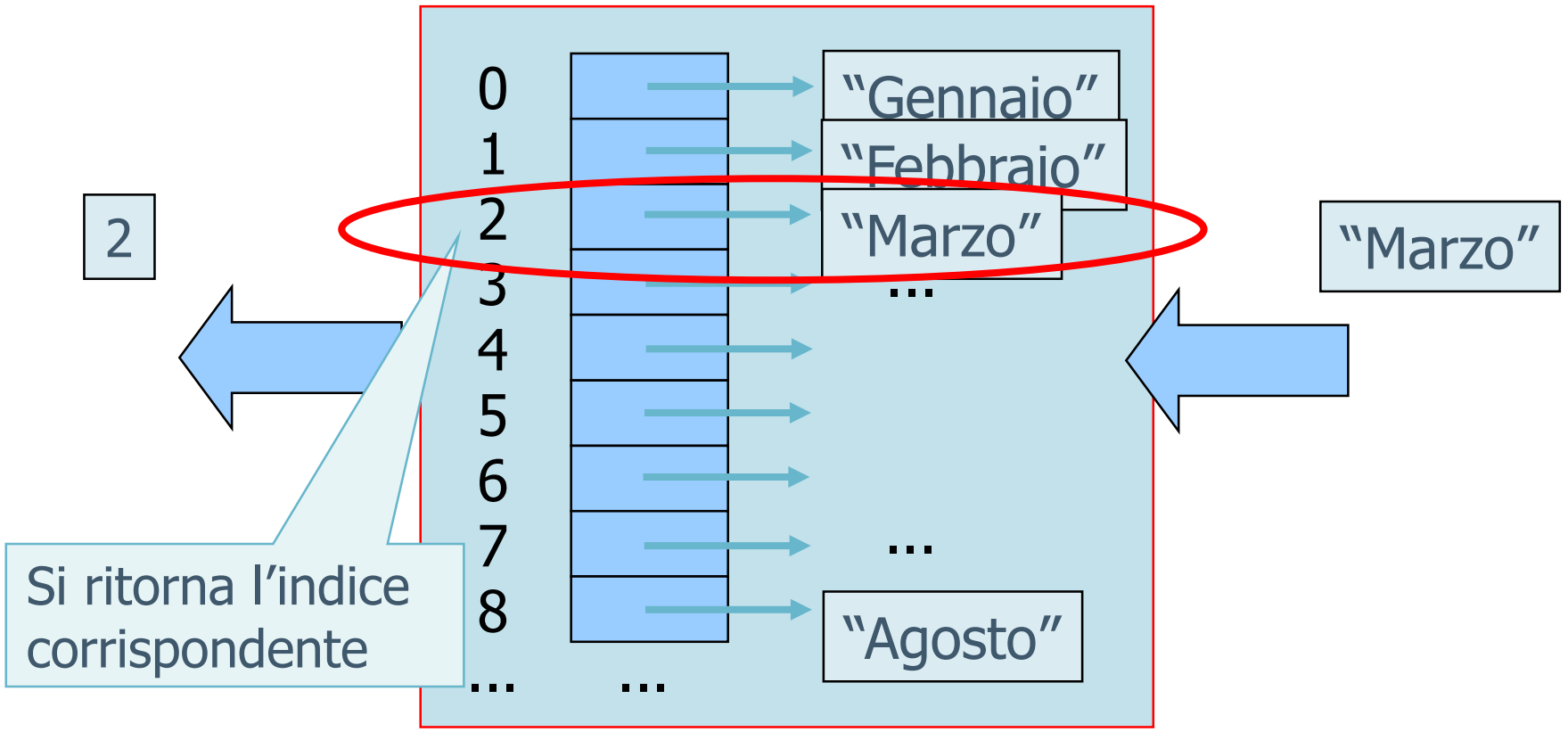
Per la conversione da stringa a intero si può utilizzare la corrispondenza dato → indice



La tabella di conversione (A)

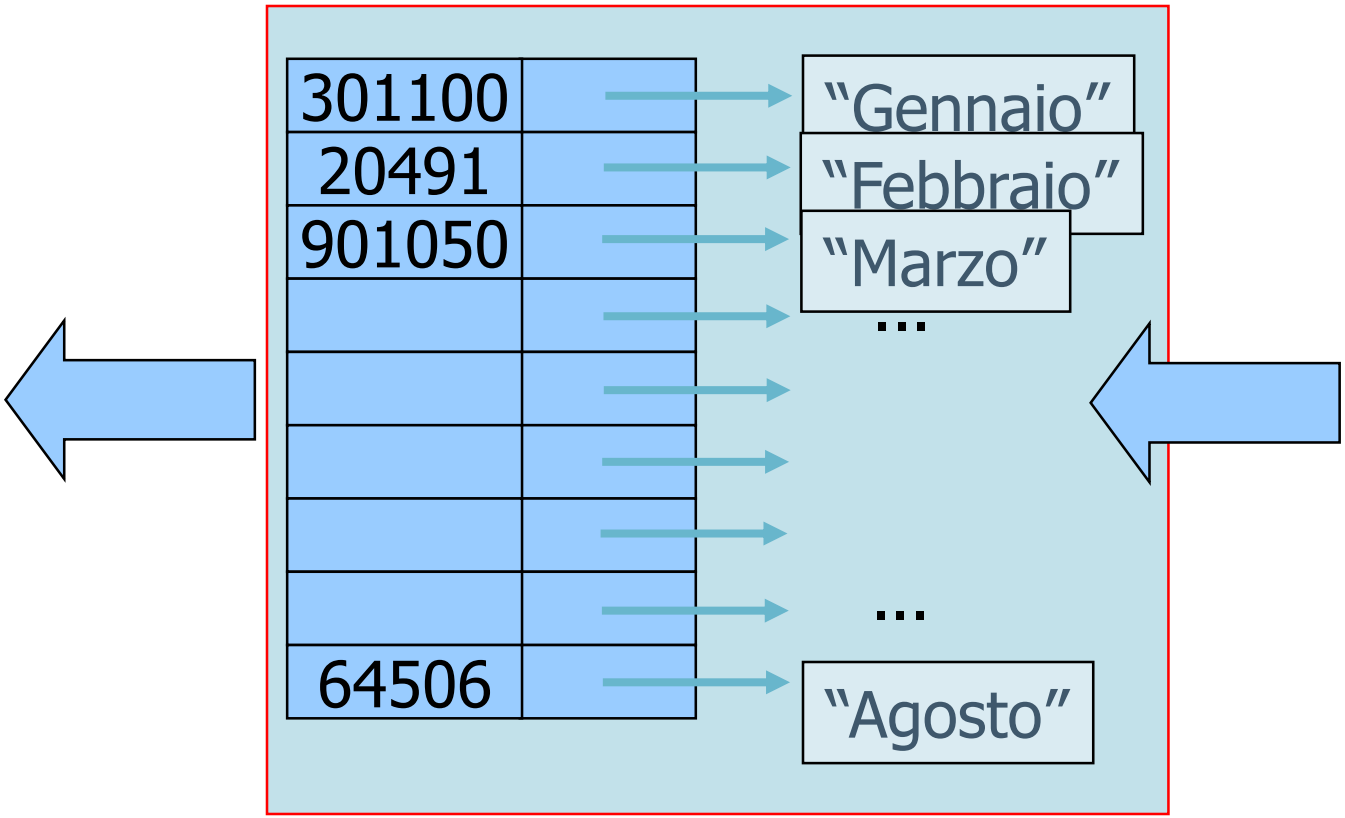


La tabella di conversione (A)

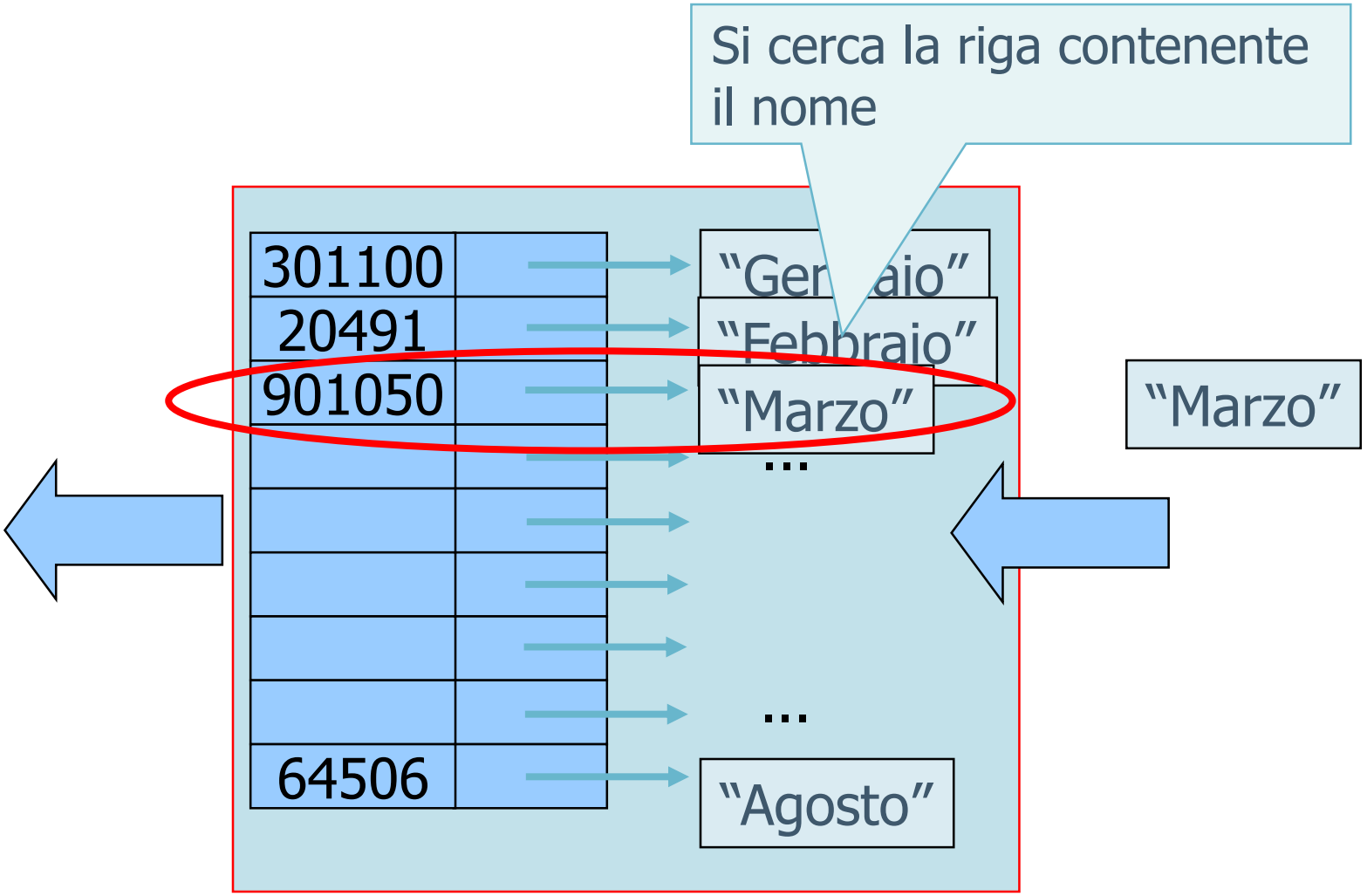


La tabella di conversione (B)

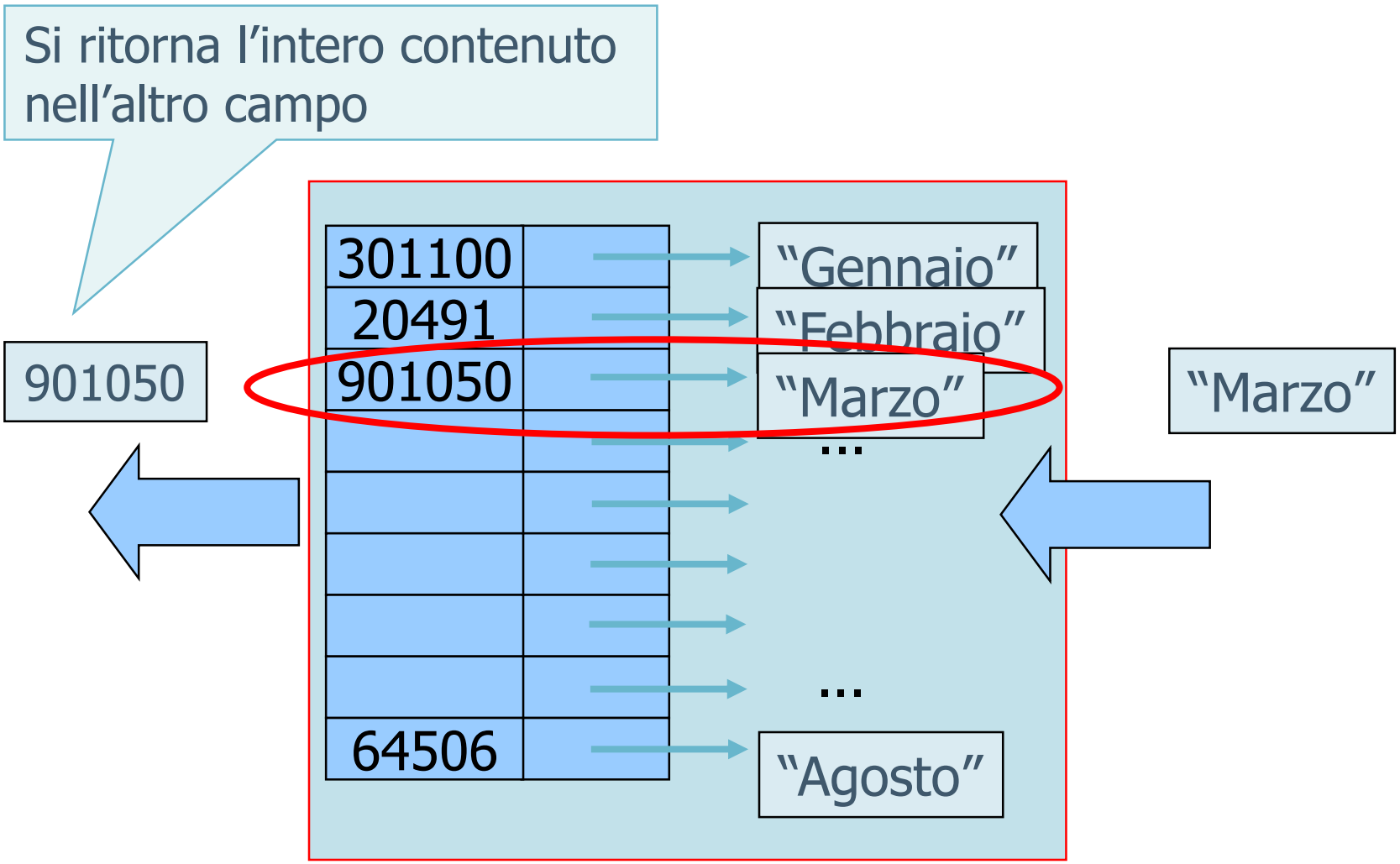
Se i valori interi sono troppo grandi (non adatti come indici) si può realizzare un vettore di `struct` (codice,nome)

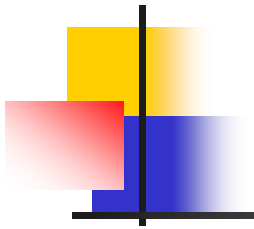


La tabella di conversione (B)



La tabella di conversione (B)

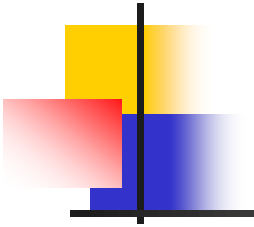




```
t_comandi leggiComando (void) {
  t_comandi c;
  char cmd[MAXL];
  char *tabella[c_err] = {
    "cerca", "modifica", "stampa", "uscita"
  }
  printf("comando (cerca/modifica");
  printf("/stampa/uscita): ");
  scanf("%s",cmd); strToLower(cmd);
  c=c_cerca;
  while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
  return (c);
}
```

```
t_comandi leggiComando (void) {
  t_comandi c;
  char cmd[MAXL];
  char *tabella[c_
    "cerca", "modi
  }
  printf("comando (cer      odifica");
  printf("/stampa/usci      : ");
  scanf("%s",cmd); st     oLower(cmd);
  c=c_cerca;
  while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
  return (c);
}
```

Problema di selezione/ricerca
(vedi lezione successiva)



- Soluzione:
  - analizzare iterativamente i dati nel vettore, confrontando di volta in volta la matricola corrente con quella richiesta
- Struttura dati:
  - la tabella è un vettore (fornito come parametro)
  - il numero di matricola richiesto (secondo parametro) è una stringa
- Algoritmo:
  - la ricerca consiste in una verifica dei dati
  - la funzione ricerca la matricola e ritorna il nome corrispondente (NULL per indicare fallimento).

Codice

conversione_matricola_nome.c

```
typedef struct {
   char matricola[MMAX+1], nome[NMAX+1];
} t_stud;
...
char *matrNome(char m[],t_stud tabella[],int n){
   int i;

   for (i=0; i<n; i++) {
      if (strcmp(m,tabella[i].matricola)==0)
         return(tabella[i].nome);
   }
   return NULL;
}
```