

Problemi di codifica/decodifica di testi

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

Crittografia di un file di testo

Soluzione:

- Leggere la tabella da file, costruendo una vettore di ricodifica indice (codice iniziale) → dato (nuovo codice)
- Leggere iterativamente i caratteri dal primo file
 - Calcolare la ricodifica del carattere
 - Scrivere il carattere sul file risultato

- *Esempio:* se il file contiene

65	38
----	----

vuol dire che il carattere 'A' (codice ASCII 65) deve essere codificato come '&' (codice ASCII 46)

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	((72	48	110	H	H	104	68	150	h	h
41	29	051))	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k

Struttura dati:

- [illegible]

- 55

Crittografia di un file di testo

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 → dato nella tabella

Codice

```
#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");
```

```
/* inizializza tabella: ogni carattere
   convertito in se stesso */
for (i=0; i<256; i++)
    tabella[i] = (char)i;
/* leggi da file le conversioni presenti */
while (fscanf(ftab, "%d%d", &i, &nuovo) == 2)
    tabella[i] = (char)nuovo;
/* converti fpin in fpout */
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 ELABORAZIONE TESTI: INPUT,
MODIFICA, OUTPUT

Problemi di text-processing

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

Esempi: input (e comprensione) di un testo, costruzione o modifica di testo, creazione di messaggio in un dato formato

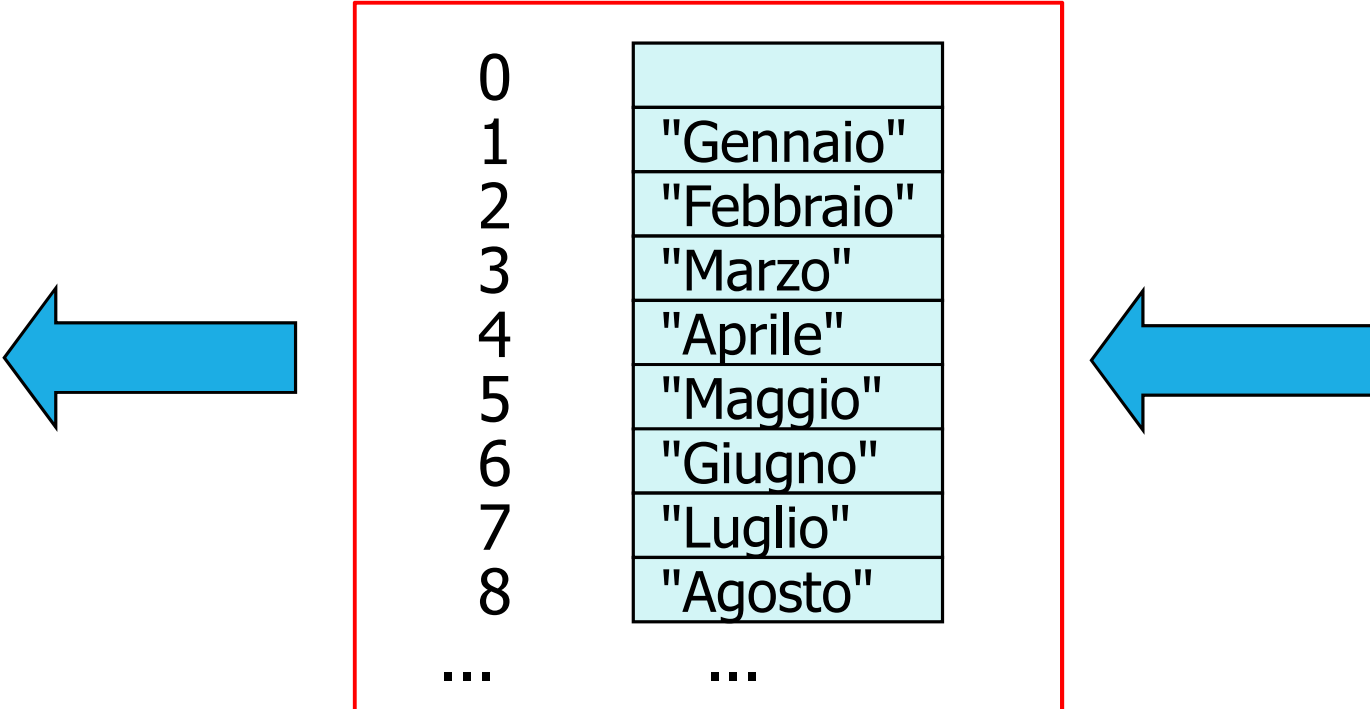
- Vettori (o matrici) di caratteri (o di stringhe) sono spesso necessari per:
 - Generare testi a partire da regole o funzioni
 - Trasformare testi esistenti
 - Acquisire in input o predisporre per output un testo

Vettori di stringhe e selezione

- Un vettore di stringhe è formalmente una matrice di caratteri
 - `char parole[NPAROLE][MAXL];`
- Problema: data una stringa (una parola), cercarla in una tabella (di parole) per "capire" a quale dato (es. un numero) corrisponda
 - La selezione basata direttamente su stringhe richiede confronti (`strcmp`) e costrutti condizionali `if`
 - Non sono possibili `switch`
 - I vettori (usati come tabelle) possono consentire la traduzione da codici testuali a codici numerici, con cui:
 - E' 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)

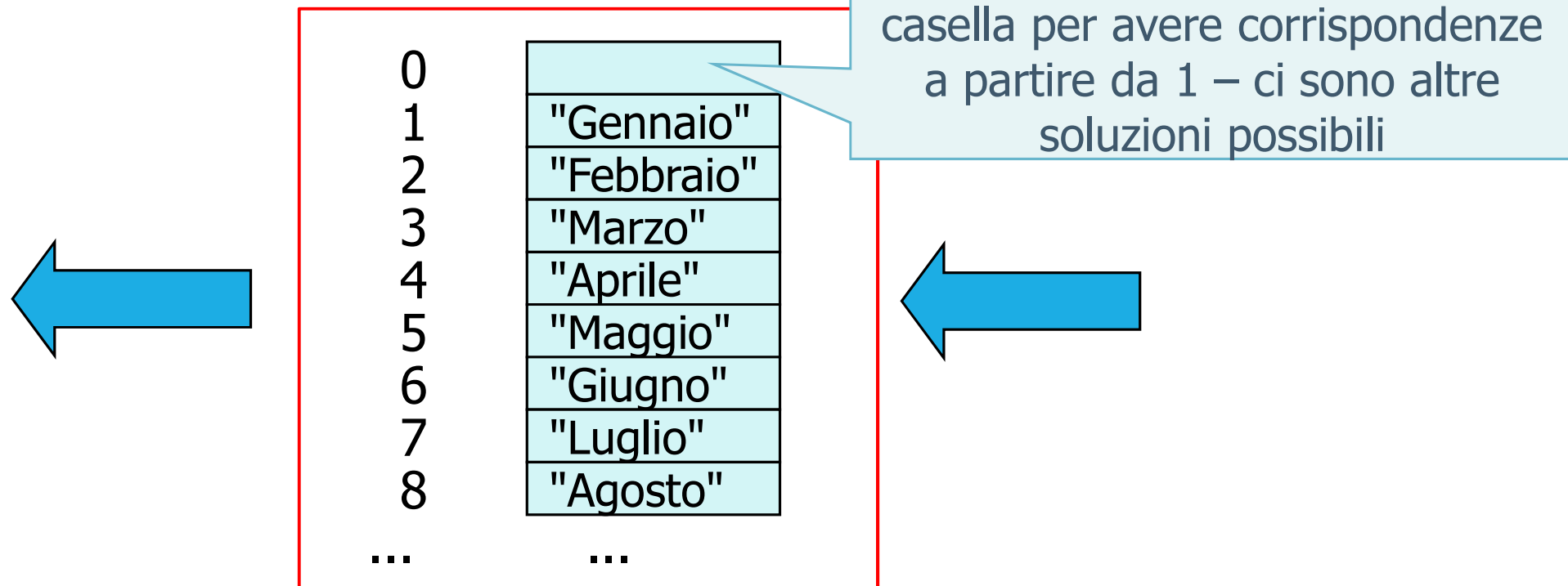
Per la conversione da stringa a intero si può utilizzare la corrispondenza dato \rightarrow indice



0	
1	"Gennaio"
2	"Febbraio"
3	"Marzo"
4	"Aprile"
5	"Maggio"
6	"Giugno"
7	"Luglio"
8	"Agosto"
...	...

La tabella di conversione (A)

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

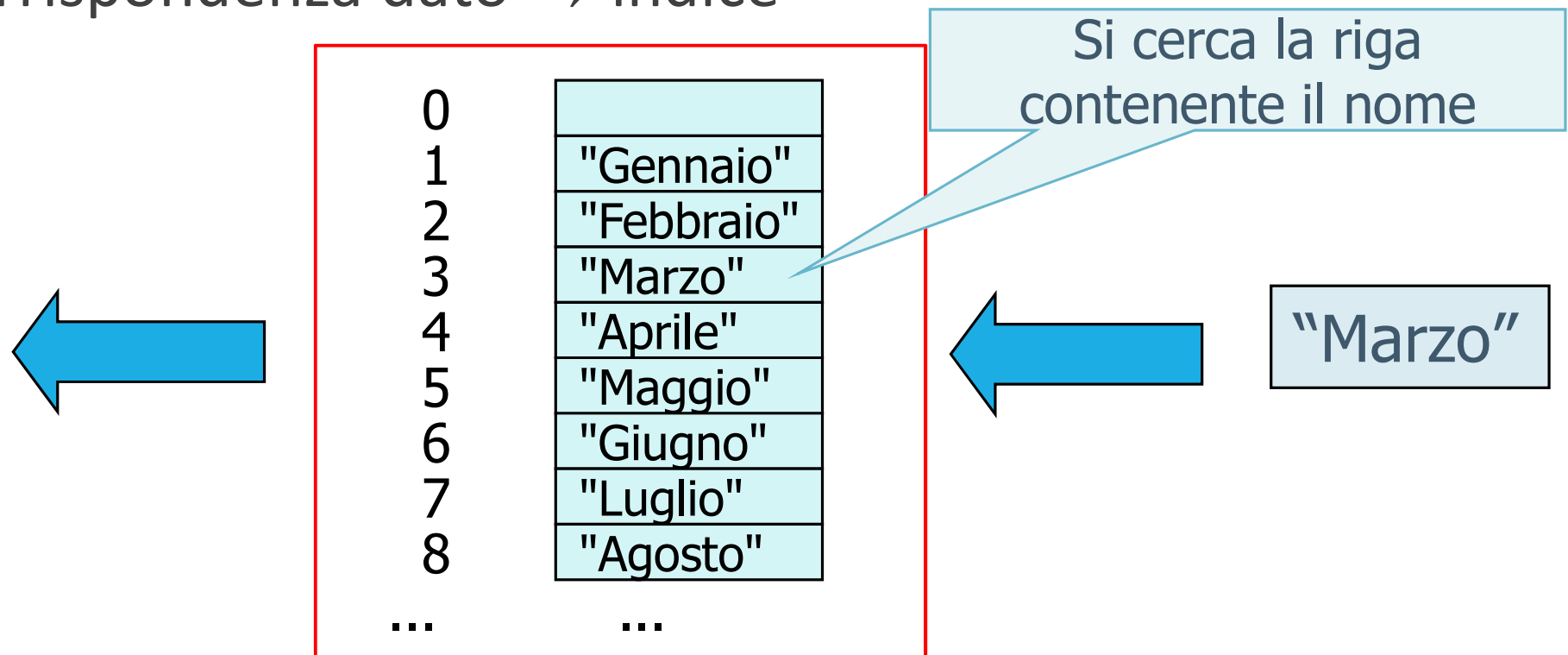


0	
1	"Gennaio"
2	"Febbraio"
3	"Marzo"
4	"Aprile"
5	"Maggio"
6	"Giugno"
7	"Luglio"
8	"Agosto"
...	...

NOTA: non si usa la prima casella per avere corrispondenze a partire da 1 – ci sono altre soluzioni possibili

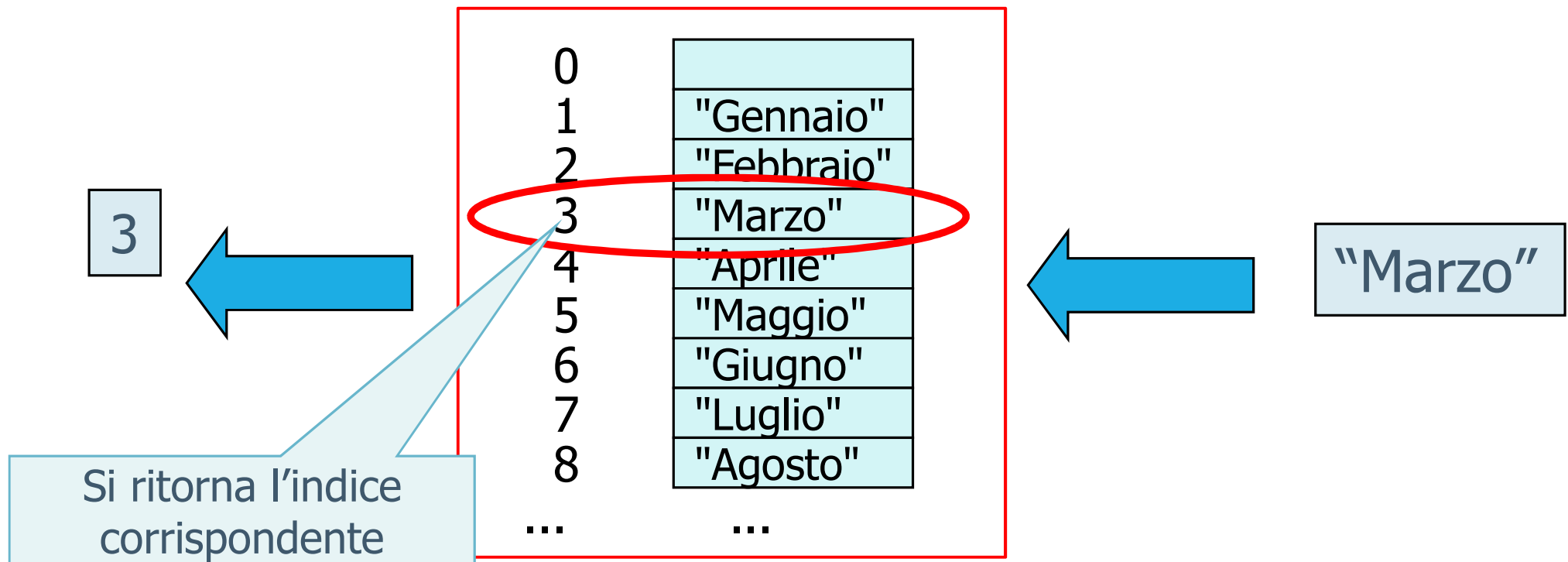
La tabella di conversione (A)

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



La tabella di conversione (A)

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



Codice

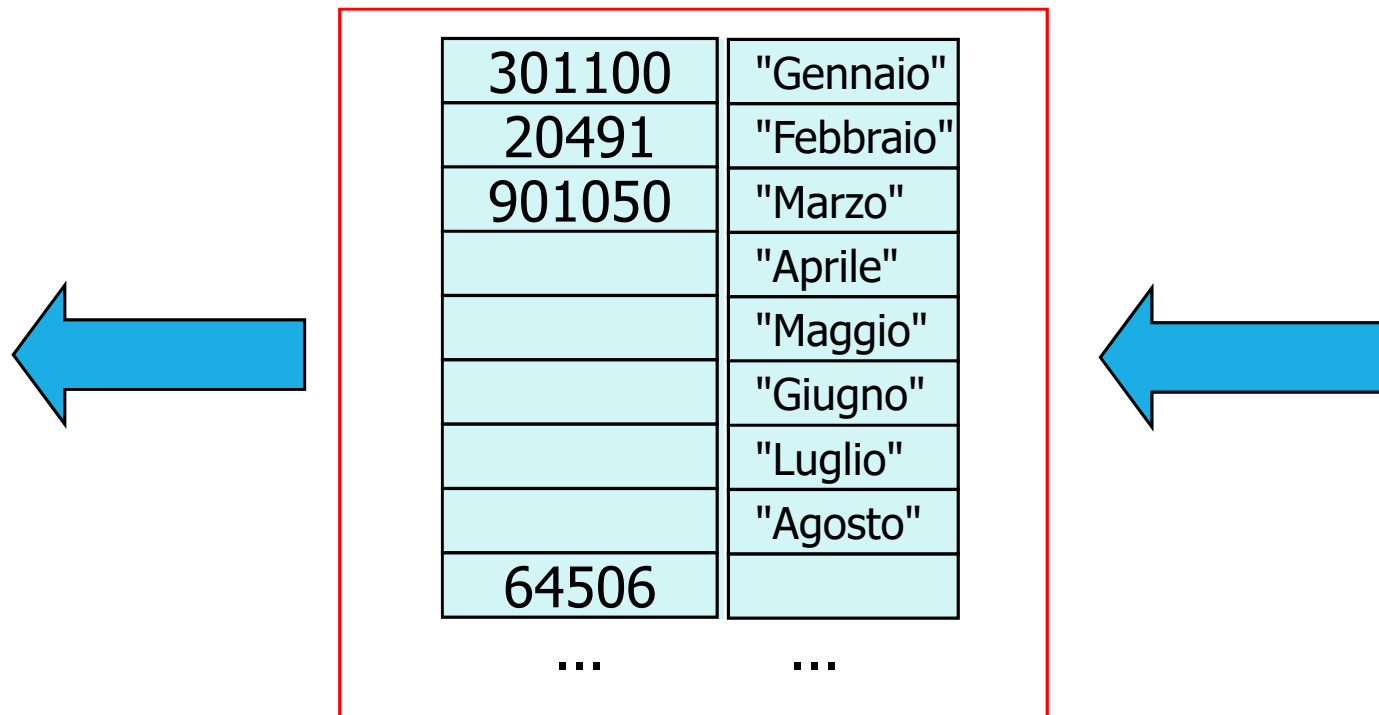
```
#include <string.h>

int monthStringToNum (char month[]) {
    char table[13][10] = {"",
                          "January", "February", "March", "April", "May", "June",
                          "July", "August", "September", "October",
                          "November", "December"};

    int i;
    for (i=1; i<=12; i++) {
        if (strcmp(month,table[i])==0) {
            return i; // found: return index
        }
    }
    return -1; // there is a problem, month not found
}
```

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) o un doppio vettore



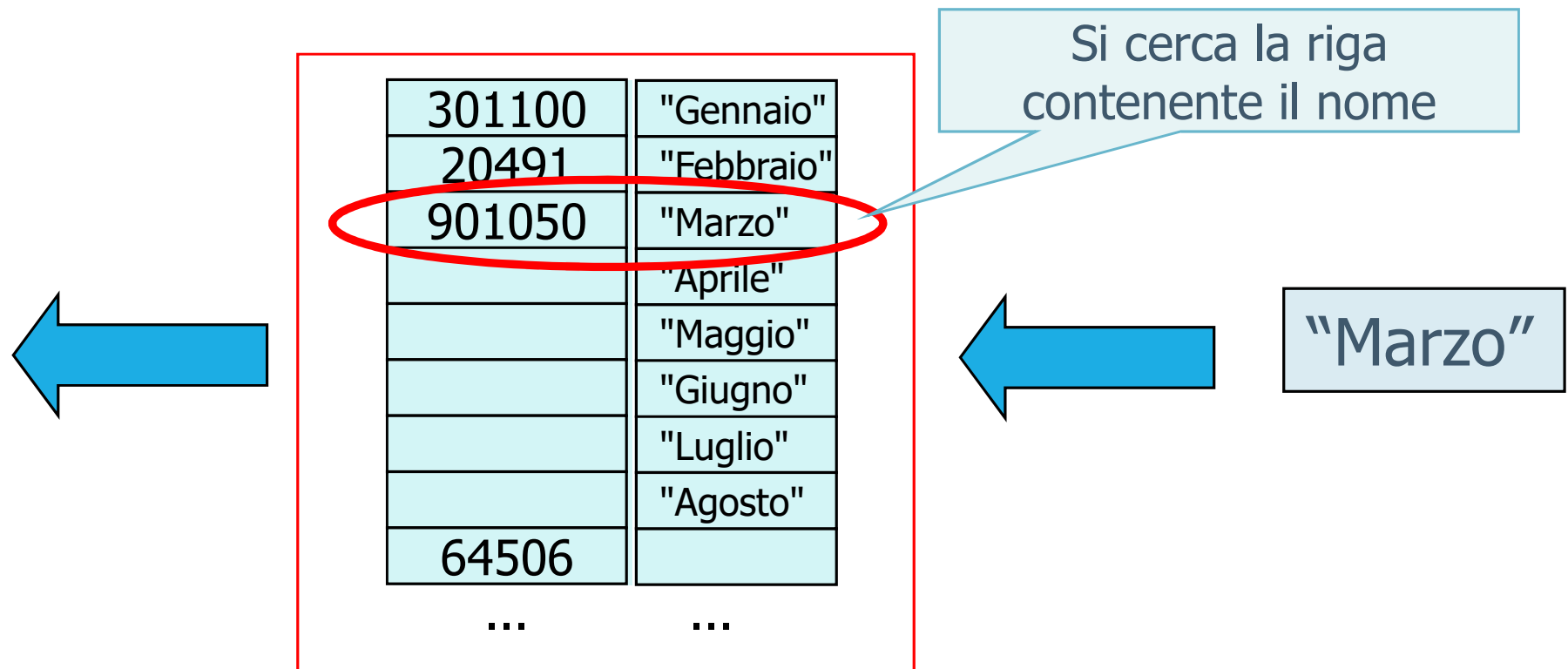
The diagram illustrates a conversion table (B) enclosed in a red rectangular border. The table consists of two columns: the left column contains integer values, and the right column contains month names. The rows are as follows:

301100	"Gennaio"
20491	"Febbraio"
901050	"Marzo"
	"Aprile"
	"Maggio"
	"Giugno"
	"Luglio"
	"Agosto"
64506	
...	...

Two large blue arrows point horizontally away from the table, one to the left and one to the right, indicating the direction of data flow or conversion.

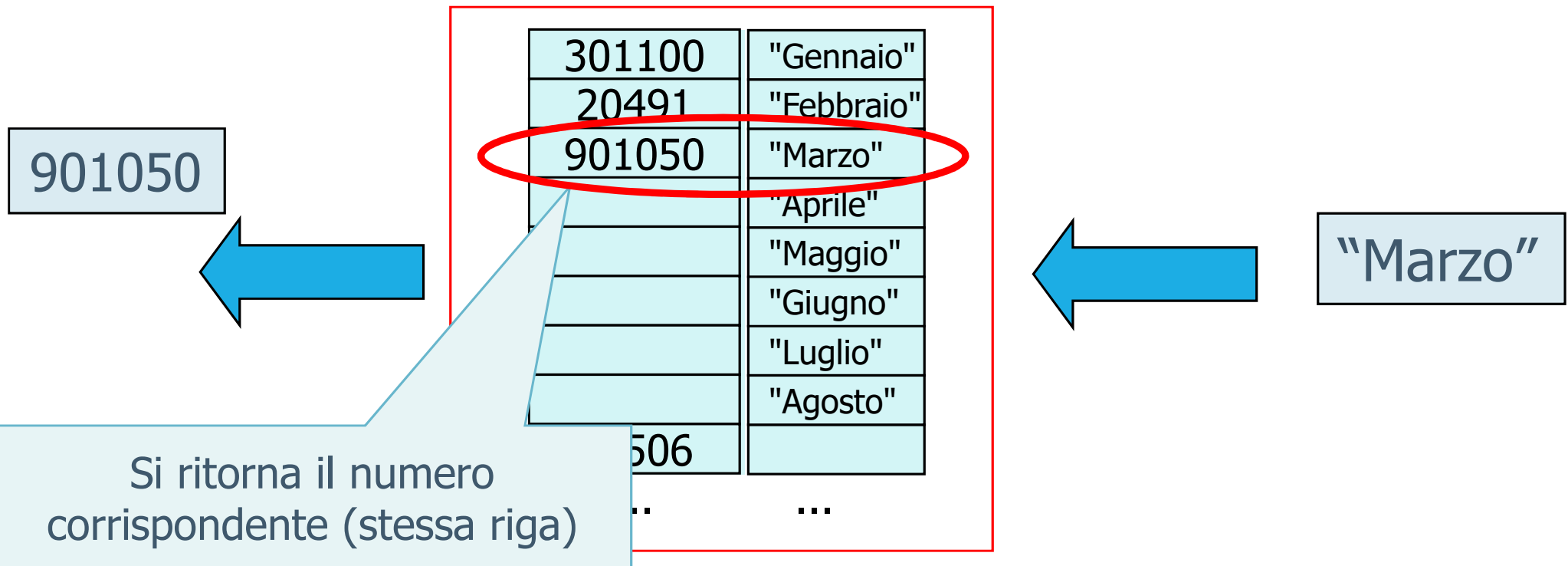
La tabella di conversione (B)

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



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) o un doppio vettore



Codice

```
struct monthEntry {
    int num;
    char name[10];
}

int main (void) {
    int i, num;
    char month[10];
    struct monthEntry table[12];
    if (readTable("table.txt",table) != 0)
        do {
            printf("Write a month"); scanf("%s",month);
            n = monthStringToNum(table,month);
            if (n>=0)
                printf("Month: %s -> num: %d\n");
        } while (n>=0);
    return 0;
}
```

```
int readTable (struct monthEntry t[12]) {
    FILE *fp; int i;
    fp = fopen("table.txt","r");
    if (fp==NULL) {
        printf("Error opening table.txt\n"); return 0;
    }
    for (i=1; i<=12; i++)
        fscanf(fp,"%d%s", &t[i].num, t[i].name);
    fclose(fp); return 1;
}

int monthStringToNum (struct monthEntry t[12], char m[]) {
    int i;
    for (i=1; i<=12; i++) {
        if (strcmp(month,t[i].name)==0)
            return table[i].num; // found: return num
    }
    return -1; // there is a problem, month not found
}
```

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 una 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

Menu con scelta su una parola

- Modularizzazione:

- Funzione di input
- Funzione di conversione da stringa a codice

- Tabella:

- Vettore inizializzato mediante (puntatori a) costanti stringa

- Conversione da stringa a codice:

- Iterazione di ricerca su vettore

Codice

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

#define c_cerca 0
#define c_modifica 1
#define c_stampa 2
#define c_fine 3
#define c_err 4
const int MAXL=51;

int leggiComando (void);
void menuParola (void);

void cerca (char r[]) { printf("cerca: %s\n", r); }
void modifica (char r[]) { printf("modifica: %s\n", r); }
void stampa (char r[]) { printf("stampa: %s\n", r); }
```

```
void strToLower(char s[]) {
    int i, l = strlen(s);
    for (i=0; i<l; i++)
        s[i]=tolower(s[i]);
}

int main(void) {
    menuParola();
}
```

Codice

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

#define c_cerca 0
#define c_modifica 1
#define c_stampa 2
#define c_fine 3
#define c_err 4
const int MAXL=51;

int leggiComando (void);
void menuParola (void);

void cerca (char r[]) { printf("cerca: %s\n", r); }
void modifica (char r[]) { printf("modifica: %s\n", r);}
void stampa (char r[]) { printf("stampa: %s\n", r); }
```

Definizione di costanti più intuitive e facili da usare rispetto agli interi corrispondenti da 0 a 4

```
s[i]=tolower(s[i]);
}

int main(void) {
    menuParola();
}
```

Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto della riga
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("comando errato\n");
        }
    }
}
```

```
int leggiComando (void) {
    int c;
    char cmd[MAXL];
    char tabella[4][9] = {
        "cerca","modifica","stampa","uscita"
    };
    printf("comando (cerca/modifica);");
    printf("/stampa/uscita): ");
    scanf("%s",cmd); strToLower(cmd);
    c=c_cerca; // 0
    while(c<c_err && strcmp(cmd,tabella[c])!=0)
        c++;
    return (c);
}
```

Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto della riga
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("comando errato\n");
        }
    }
}
```

```
int leggiComando (void)
{
    int c;
    char cmd[MAXL];
    char tabella[4][9] = {
        "cerca", "modifica", "stampa", "uscita"
    };
    printf("comando (cerca/modifica);");
    printf("/stampa/uscita): ");
    scanf("%s",cmd); strToLower(cmd);
    c=c_cerca; // 0
    while(c<c_err && strcmp(cmd,tabella[c])!=0)
        c++;
    return (c);
}
```

```
#define c_cerca 0
#define c_modifica 1
#define c_stampa 2
#define c_fine 3
#define c_err 4
```


Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto della riga
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("comando errato\n");
        }
    }
}
```

Lettura comando e conversione
da nome a numero

```
char tabella[4][9] = {
    "cerca","modifica","stampa","uscita"
};
printf("comando (cerca/modifica);
printf("/stampa/uscita): ");
scanf("%s",cmd); strToLower(cmd);
c=c_cerca; // 0
while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
return (c);
}
```

Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto della riga
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("comando errato\n");
        }
    }
}
```

Resto della riga, usato
per eseguire il comando

```
char tabella[4][9] = {
    "cerca","modifica","stampa","uscita"
};
printf("comando (cerca/modifica);
printf("/stampa/uscita): ");
scanf("%s",cmd); strToLower(cmd);
c=c_cerca; // 0
while(c<c_err && strcmp(cmd,tabella[c])!=0)
    c++;
return (c);
}
```

Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("comando errato\n");
        }
    }
}
```

Selezione della funzione da chiamare o azione da eseguire

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

Codice

```
void menuParola (void){
    int comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); // resto della riga
        switch (comando) {
            case c_cerca: cerca(riga); break;
            case c_modifica: modifica(riga); break;
            case c_stampa: stampa(riga); break;
            case c_fine: continua=0; break;
            case c_err:
            default: printf("Comando non valido\n");
        }
    }
}
```

Da nome a numero
mediante ricerca in tabella

```
int leggiComando (void) {
    int c;
    char cmd[MAXL];
    char tabella[4][9] = {
        "cerca","modifica","stampa","uscita"
    };
    printf("comando (cerca/modifica);");
    printf("/stampa/uscita): ");
    scanf("%s",cmd); strToLower(cmd);
    c=c_cerca; // 0
    while(c<c_err && strcmp(cmd,tabella[c])!=0)
        c++;
    return (c);
}
```

MENU: variante con tipo enum

■ Il tipo enum in C:

- Associa automaticamente nomi ai numeri interi a partire da 0
 - Es.: `enum semaforo {verde, rosso, giallo};`
Definisce il tipo "enum semaforo", che associa automaticamente i nomi verde, rosso e giallo ai numeri 0, 1, 2 rispettivamente
 - Si possono definire altre associazioni (saltare degli interi), noi lo evitiamo
- Attenzione, in C si può fare aritmetica, in C++ NO
- Spesso associato a typedef, l'equivalente della #define, applicato ai tipi.
 - Es.: `typedef enum {verde, rosso, giallo} semaforo_e;`

- Possiamo modificare il programma: nel menu usiamo enum invece di definire costanti numeriche mediante #define

Codice

```
typedef enum {
    c_cerca,c_modifica,c_stampa,c_fine,c_err
} t_comandi;
...
void menuParola (void){
    t_comando comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); /* resto della riga */
        switch (comando) {
            case c_cerca: cerca(riga); break;
            ...
        }
    }
}
```

```
t_comandi leggiComando (void) {
    t_comandi c;
    char cmd[MAXL];
    char tabella[c_err][9] = {
        "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);
}
```

Codice

```
typedef enum {
    c_cerca,c_modifica,c_stampa,c_fine,c_err
} t_comandi;
...
void menuParola (void){
    t_comando comando;
    char riga[MAXL];
    int i, continua=1;
    while (continua) {
        comando = leggiComando();
        fgets(riga,MAXL,stdin); /* resto della riga */
        switch (comando) {
            case c_cerca: cerca(riga); break;
            ...
        }
    }
}
```

```
#define c_cerca 0
#define c_modifica 1
#define c_stampa 2
#define c_fine 3
#define c_err 4
```

```
void) {
    char tabella[c_err][9] = {
        "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);
}
```

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 testi a livello carattere

- 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 è palindroma se letta dall'ultimo al primo carattere non varia
Esempi: Anna, madam, otto, abcdefgFEDCBA
- Algoritmo:
 - Iterazione con 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 i, n, pal=1;  
  
    n = strlen(parola);  
    for (i=0; i<n/2 && pal; i++) {  
        if (toupper(parola[i]) != toupper(parola[n-1-i]))  
            pal = 0;  
    }  
    return pal;  
}
```

'm'	'a'	'd'	'a'	'm'
0	1	2	3	4

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

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

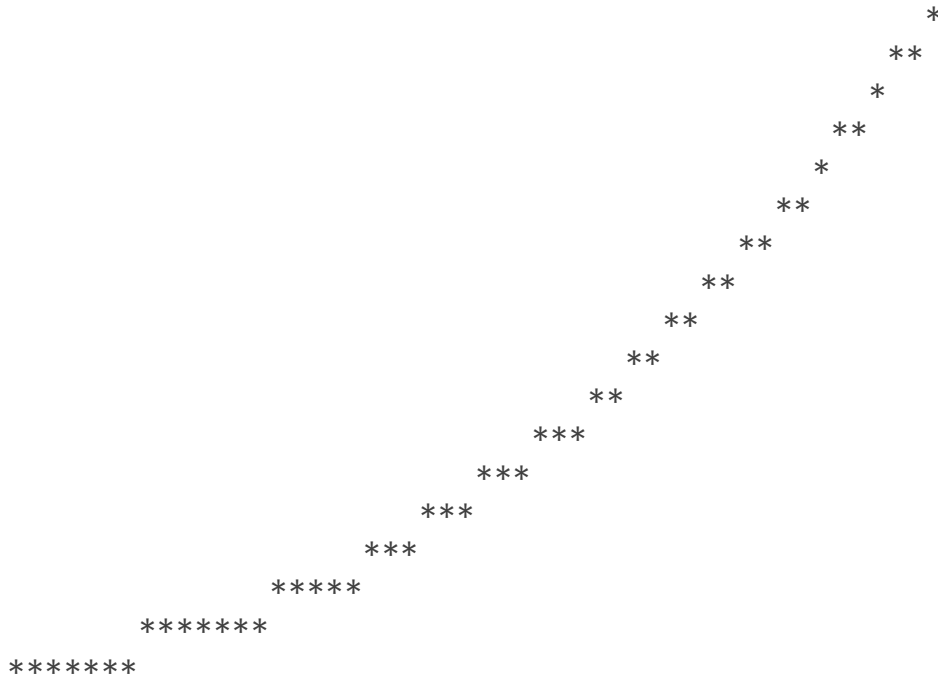
- Si scriva un programma che:

- Acquisisca da tastiera i coefficienti a , b , c , e i valori degli estremi (x_{\min} , x_{\max}) e (y_{\min} , y_{\max}) 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 $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$

Visualizzazione di parabola

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:



$$y = x^2 + 2x + 1$$

Intervalli:

- Asse x: $[-1, 4]$
- Asse y: $[-1, 10]$

Visualizzazione di parabola

- 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

Visualizzazione di parabola

■ Algoritmo:

- Input dati e calcolo passo (= lunghezza intervalli per entrambi gli assi)
- Inizializzazione a tutti spazi della matrice di caratteri
- Iterazione su valori di j (colonne della matrice)
 - Dato i, calcolo $x = x_{\min} + j * \text{passoX}$
 - Calcolo $y(x)$
 - Se è nell'intervallo $[y_{\min}, y_{\max}]$ converti in intero (i)
 - Assegna '*' nella matrice alla posizione corrispondente $[i][j]$
- Iterazioni su righe e colonne, per stampare matrice

Codice

```
#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) {
        i = (y-ymin)/passoY;
        pagina[i][j] = '*';
    }
}
```

Codice

```
/* stampa matrice per righe:  
   il minimo valore di y (prima riga) stampato per ultimo in basso */  
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:

- E' 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
 - Lavorano considerando le stringhe come unità elementari

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

Formattazione di testo

■ Formulazione:

- E' 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

Formattazione di testo

■ Soluzione:

- La soluzione è simile al problema del Cap. 3 (che non prevedeva la 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 (area temporanea)
 - La centratura (su l_{\max} caratteri) di una riga di l caratteri, si effettua stampando, prima della riga, $(l_{\max} - l) / 2$ spazi

Codice

```
#include <string.h>
const int STRLEN=21;
const int LMAX=255;
...
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) {
        if (l+1+strlen(parola) <= lmax) {
            strcat(riga," "); strcat(riga,parola);
            l+=1+strlen(parola);
        }
    }
```

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

Problemi di verifica e selezione

PROBLEMI DI VERIFICA, SELEZIONE E ORDINAMENTO
APPLICATI A VETTORI

Problemi di verifica e selezione

- I problemi di verifica consistono nel decidere se un insieme di informazioni o dati rispettino un dato 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 che corrispondono al criterio 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%

Verifica di dati ripetuti

■ Soluzione:

- Analizzare i dati, mediante una doppia iterazione, verificando per ognuno che ne esista almeno uno uguale
 - Differenza $\leq 1\%$ 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 (dimensione costante!)
- Verifica mediante doppia iterazione
- Quantificazione con uso di flag
 - Appena un dato non rispetta il criterio, interrompo la ricerca e ritorno 0

Codice

```
int datiRipetuti (FILE *fp) {  
    float dati[MAXDATI];  
    int ndati, i, j, ripetuto;  
    fscanf(fp, "%d", &ndati); // assumiamo ndati ≤ MAXDATI  
    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)
- I numeri di matricola sono grandi (6 cifre decimali, MMAX)
 - Non si vuole (non si può) sfruttare la corrispondenza indice-dato in un vettore
 - Le matricole sono rappresentate mediante stringhe
- La tabella di conversione, secondo parametro alla funzione, è un vettore di `struct`, aventi come campi (stringhe) numero di matricola e nome
 - Si suppone che il nome abbia lunghezza massima NMAX
- Il terzo parametro (intero) è la dimensione della tabella
- Il quarto parametro è la stringa (vettore di caratteri) in cui porre il risultato

Conversione matricola→nome

■ 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 di struct (fornito come parametro)
- Il numero di matricola e il risultato (parametri) sono stringhe

■ Algoritmo:

- La ricerca consiste in una verifica dei dati
- La funzione ricerca la matricola e ritorna il nome corrispondente
- Il valore intero ritornato è 1 o 0 (vero o falso) indica successo o no

Codice

```
#include <string.h>
typedef struct {
    char matricola[MMAX+1], nome[NMAX+1];
} t_stud;
...
int matrNome(char m[], t_stud tabella[], int n, char n[]){
    int i;
    for (i=0; i<n; i++) {
        if (strcmp(m, tabella[i].matricola)==0) {
            strcpy(n, tabella[i].nome);
            return 1;
        }
    }
    return 0;
}
```


Codice

```
#include <string.h>
typedef struct {
    char matricola[MMAX+1], nome[NMAX+1];
} t_stud;
...
int matrNome(char m[], t_stud tabella[], int n, char n[]){
    int i;
    for (i=0; i<n; i++) {
        if (strcmp(m, tabella[i].matricola)==0) {
            strcpy(n, tabella[i].nome);
            return 1;
        }
    }
    return 0;
}
```

Si noti typedef per definire un sinonimo (**t_stud**) di **struct studente**
typedef è l'equivalente (applicato ai tipi) di #define per le costanti (es. numeri)

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

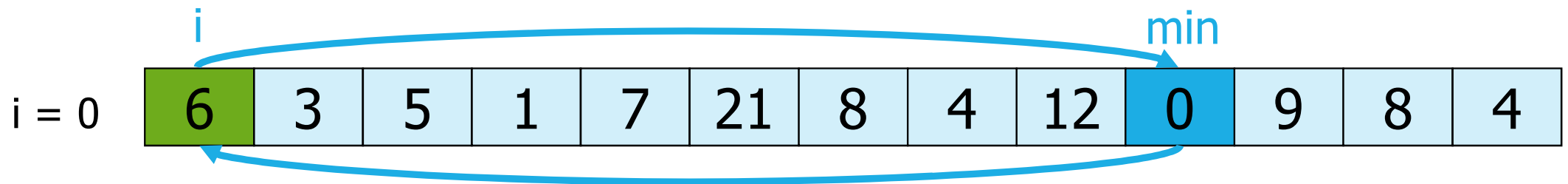
- Formulazione: si scriva una funzione C che:
 - Ricevuti come parametri un vettore di numeri interi e la sua dimensione
 - Ordini i dati in modo crescente con l'algoritmo di *selection sort*
- Soluzione: selection sort
 - Algoritmo di ordinamento basato su ripetute ricerche/selezioni del minimo

Selection sort

- Struttura dati e algoritmo:

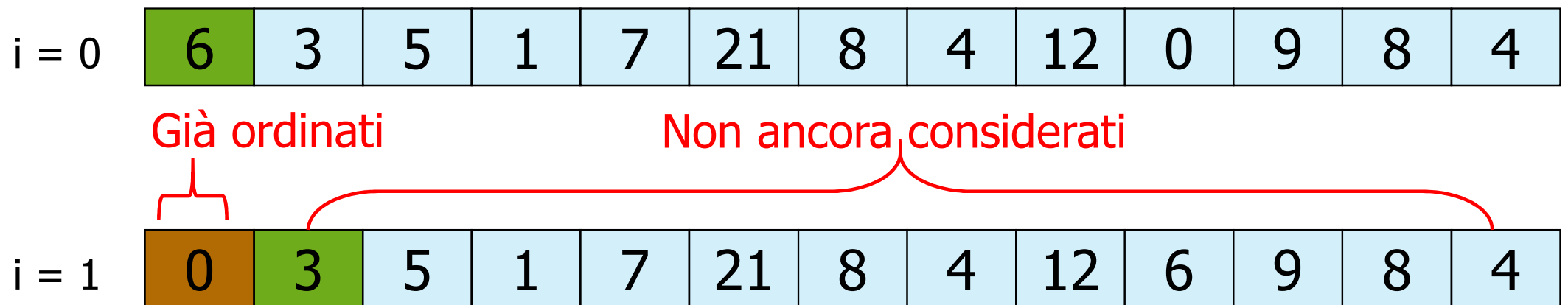
- Vettore A di N interi ($A[0] \dots A[N-1]$), diviso in 2 sotto-vettori:
 - Di sinistra: ordinato (inizialmente vuoto)
 - Di destra: disordinato
- Un vettore di un solo elemento è ordinato
- Approccio incrementale: al passo i
 - il minimo del sotto-vettore ($A[i] \dots A[N-1]$, i.e., di destra) è assegnato ad $A[i]$
 - incremento i
- Terminazione: tutti gli elementi inseriti ordinatamente

Esempio

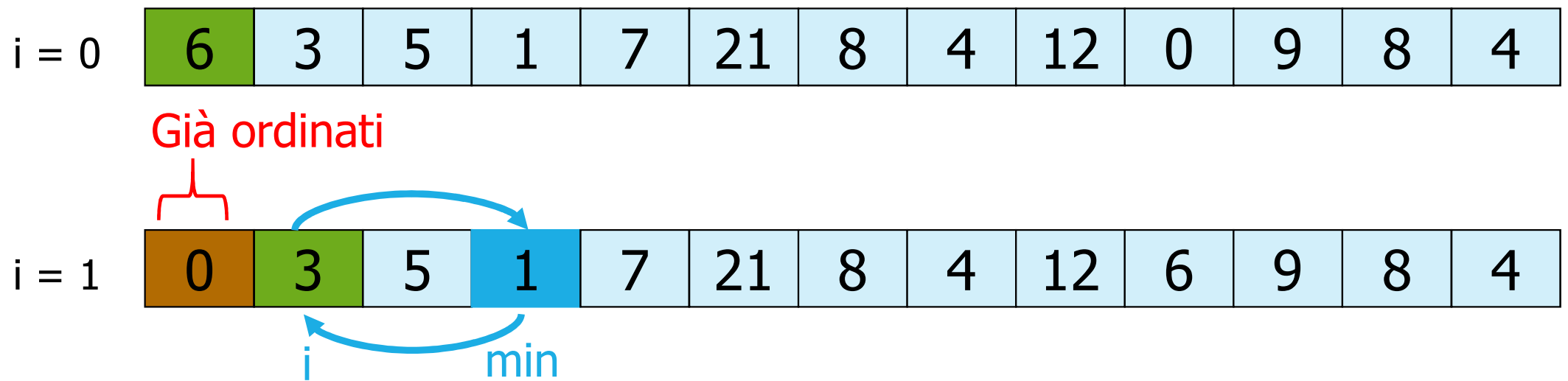


- Considero $i = 0$
- Il minimo di $A[1] \dots A[12]$ sostituisce $A[i]$

Esempio



Esempio



Esempio

$i = 0$

6	3	5	1	7	21	8	4	12	0	9	8	4
---	---	---	---	---	----	---	---	----	---	---	---	---

$i = 1$

0	3	5	1	7	21	8	4	12	6	9	8	4
---	---	---	---	---	----	---	---	----	---	---	---	---

Già ordinati

Non ancora considerati

$i = 2$

0	1	5	3	7	21	8	4	12	6	9	8	4
---	---	---	---	---	----	---	---	----	---	---	---	---

Codice

```
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;
    }
}
```

Codice

```
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 esterna, eseguita N-1 volte

Codice

```
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

Codice

```
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 ***"in loco"***, perché scambia i dati sul vettore (non serve un secondo vettore "di appoggio")

Sorting applicato a vettore di struct

- Esempio: funzione `ordinaStudenti` indicata, ma NON ancora realizzata

```
struct studente {  
    char cognome[MAX], nome[MAX];  
    int matricola;  
    float media;  
};
```

Vettore di **struct**

```
int main(void) {
    struct studente elenco[NMAX];
    int i, n;

    printf("quanti studenti(max %d)? ", NMAX);
    scanf("%d", &n);
    for (i=0; i<n; i++) {
        elenco[i] = leggiStudente();
    }
    ordinaStudenti(elenco, n);
    printf("studenti ordinati per media\n");
    for (i=0; i<n; i++) {
        stampaStudente(elenco[i]);
    }
}
```

```
void ordinaStudenti(struct studente el[],
                    int n) {

    /*
     * funzione da scrivere.
     * MODIFICA il contenuto del vettore
     * riordinando gli studenti per
     * media crescente.
     *
     * Funzione completata alla fine del
     * capitolo (col selectionSort)
     */
}
```

Sorting applicato a vettore di struct

- Esempio: funzione `ordinaStudenti` indicata, ma NON ancora realizzata
- Uno dei campi usato come chiave di ordinamento (confronto)
 - `struct studente`, campo `media`
- Conveniente realizzare funzione di confronto
 - `STUDlt` (less than), oppure `STUDge` (greater or equal), ...

Sorting applicato a vettore di struct

```
// applico selection sort al vettore di studenti
void ordinaParole(char studente el[], int n) {
    int i, j, imin;
    struct studente temp;

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

```
// confronto: ritorna vero (non 0) se la media di
// s1 e' inferiore a quella di s2, falso (0)
// se non lo e'
int STUDlt (struct studente s1, struct studente s2)
{
    return (s1.media < s2.media);
}
```


Sorting applicato a vettore di stringhe

- Un vettore di stringhe è una matrice di char
- Per gestire la matrice di char come vettore occorre:
 - Usare il primo indice per identificare le righe
 - Usare la funzione strcmp per confrontare righe/stringhe
 - Usare la funzione strcpy per assegnare/copiare stringhe

Sorting applicato a vettore di stringhe

```
void ordinaNomi(char nomi[][MAXL], int n) {  
    int i, j, imin;  
    char temp[MAXL];  
  
    for (i=0; i<n-1; i++) {  
        /*cerca indice del minimo in nomi[i]..nomi[n-1]*/  
        imin = i;  
        for (j = i+1; j < N; j++)  
            if (strcmp(nomi[j],nomi[imin])<0) imin = j;  
        /*scambia minimo con el[i]*/  
        strcpy(temp,nomi[i]);  
        strcpy(nomi[i],nomi[imin]);  
        strcpy(nomi[imin],temp);  
    }  
}
```

Sorting applicato a vettore di stringhe

```
void ordinaNomi(char nomi[][MAXL], int n) {  
    int i, j, imin;  
    char temp[MAXL];  
  
    for (i=0; i<n-1; i++) {  
        /*cerca indice  
        imin = i;  
        for (j = i+1; j  
            if (strcmp(n  
        /*scambia minim  
        strcpy(temp, nom  
        strcpy(nomi[i],  
        strcpy(nomi[imin], temp);  
    }  
}
```

La funzione non ha bisogno di sapere quante sono le righe:
Il programma chiamante DEVE conoscere la dimensione (il
numero di righe)

Vantaggio: la funzione può essere chiamata su matrici di
dimensioni diverse (VALE SOLO PER IL NUMERO DI RIGHE)

Sorting applicato a vettore di stringhe

```
void ordinaNomi(char nomi[][MAXL], int n) {  
    int i, j, imin;  
    char temp[MAXL];  
  
    for (i=0; i<n-1; i++) {  
        /*cerca indice del  
        imin = i;  
        for (j = i+1; j <  
            if (strcmp(nomi[  
        /*scambia minimo d  
        strcpy(temp,nomi[i  
        strcpy(nomi[i],non  
        strcpy(nomi[imin],temp);  
    }  
}
```

La funzione ha bisogno di sapere quante sono le righe effettivamente usate (possono essere meno del totale) per fare l'ordinamento

Tocca al programma chiamante passare questa informazione aggiuntiva come argomento (n)

Da fare in autonomia 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