strutture di dati: astratte e concrete

- **tipo astratto di dati (ADT - Abstract Data Type): definizione teorica di un tipo, indipendente dal linguaggio di programmazione**

- **tipo concreto: realizzazione concreta nel calcolatore; dipendente dal linguaggio usato**

- **ADT *"TABELLA"*, con due esempi di realizzazione concreta: la collezione di stringhe, gia` vista, e la tabella di voli aerei**

Rappresentazione di dati

OGGETTI DEL MONDO REALE:

- **caratteristiche strutturali**
- **caratteristiche funzionali**

obiettivo della rappresentazione

analisi e progettazione

STRUTTURE DATI CONCRETE

- TIPI (ARRAY, STRUCT, INT, ...)
- VALORI
- VARIABILI
- FUNZIONI

dipendenti dal linguaggio di programmazione usato

implementazione

STRUTTURE DATI TEORICHE (ADT)

- **domini** (insiemi di valori che compongono la rappresentazione teorica di oggetti)
- **dominio di interesse** (insieme principale degli oggetti da modellare)
- (costanti)
- **caratteristiche formali** (espresse come funzioni e combinazioni di elementi dei domini)
  - **strutturali** (come gli oggetti sono composti)
  - **funzionali** (quali funzioni si applicano su oggetti)

formale: su carta: indipendente dal linguaggio

Un esempio di struttura astratta e sua implementazione

L'ADT TABELLA

modella tutte le entita` che corrispondono alla definizione (cioe` "le tabelle"):

- e` una collezione di oggetti ("dati"/"elementi")
- memorizzati
- ciascuno identificato univocamente da una CHIAVE (parte dell'elemento)
- fornisce le funzionalita`
  - AGGIUNTA di un elemento nella collezione
  - ELIMINAZIONE di un elemento da una collezione
  - USO di un elemento della collezione (secondo necessita` - vari usi)
  - RICERCA di un elemento nella collezione, data la sua chiave

Ad esempio, la **collezione di stringhe** era una tabella

(l'abbiamo realizzata senza l'eliminazione)

ADT "**collezione di stringhe**"

D0 = dominio di interesse = {le collezioni di stringhe}

D1 = dominio = {le stringhe}

D2 = dominio = {indici}

funzionalita`

aggiungi: D0 × D1 ⟶ D0

stampaTutto: D0 ⟶ **VOID (Ø)**

ricerca: D0 × D1 ⟶ D2

sostituisci: D0 × D1 × D1 ⟶ D0

Un esempio di struttura astratta e sua implementazione

ADT "**collezione di stringhe**"

D0 = dominio di interesse = {le collezioni di stringhe}

D1 = dominio = {le stringhe}

D2 = dominio = {indici}

(alternativa) definendo

D3 = {gli elenchi di stringhe}

si puo` dare una definizione teorica di D0 (che ha aspetti familiari in pratica …):
ogni collezione di stringhe e` composta da un elenco di stringhe e un valore nel dominio degli indici $D0 = D3 \times D2$

funzionalita`

aggiungi: $D0 \times D1 \longrightarrow D0$

aggiungi: $D3 \times D2 \times D1 \longrightarrow D0$

stampaTutto: $D0 \longrightarrow$ **VOID (Ø)**

…

ricerca: $D0 \times D1 \longrightarrow D2$ …

sostituisci: $D0 \times D1 \times D1 \longrightarrow D0$

…

Un esempio di struttura astratta e sua implementazione

implementazione dell'ADT "collezione di stringhe"

D0 = dominio di interesse = {le collezioni di stringhe} = **<array di char *, int>**

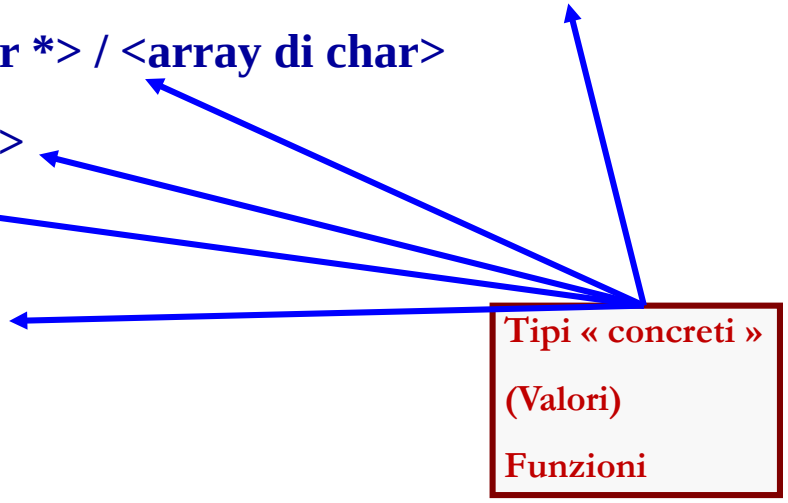
D1 = dominio = {le stringhe} = **<char *> / <array di char>**

D2 = dominio = {indici} = **<int>**

D3 = {gli elenchi di stringhe} **<array di char *>**

D0 = D3 × D2

```
struct {
    char *[MAXSTRINGHE];
    int quanteStringhe;
}
```



funzionalita`

Un esempio di struttura astratta e sua implementazione

implementazione dell'ADT "collezione di stringhe"

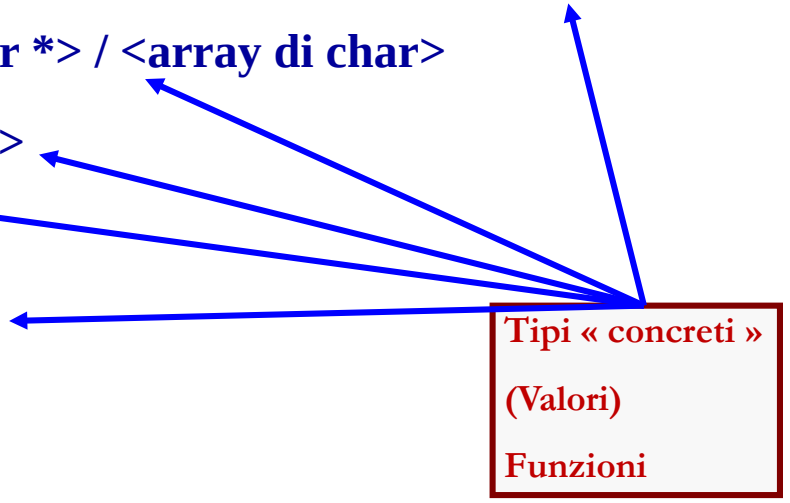
D0 = dominio di interesse = {le collezioni di stringhe} = **<array di char *, int>**

D1 = dominio = {le stringhe} = **<char *> / <array di char>**

D2 = dominio = {indici} = **<int>**

D3 = {gli elenchi di stringhe} **<array di char *>**

D0 = D3 × D2 **struct { char *[MAXSTRINGHE]; int quanteStringhe; }**



funzionalita`

```
void stampaTutto( char *v[], int quanteSono)
```

stampaTutto: D0 ⟶ VOID

```
void aggiungi ( char **v, char *nuovaStringa, int *pQuante)
```

aggiungi: D3 × D2 × D1 ⟶ D0 aggiungi: D0 × D1 ⟶ D0 **Modificato**

```
void sostituisci (char **v, int quanteSono, char *daSost, char *conChi)
```

sostituisci: D0 × D1 × D1 ⟶ D0 **Modificato**

Un esempio di struttura astratta e sua implementazione

-Definizione di TipoCollezione a partire dalla definizione di tipo record precedente

Un esempio di struttura astratta e sua implementazione

-Definizione di TipoCollezione a partire dalla definizione di tipo record precedente

```
typedef
   struct {
      char * sostegno[MAX];
      int quanteStringhe;
   }
TipoCollezione
```

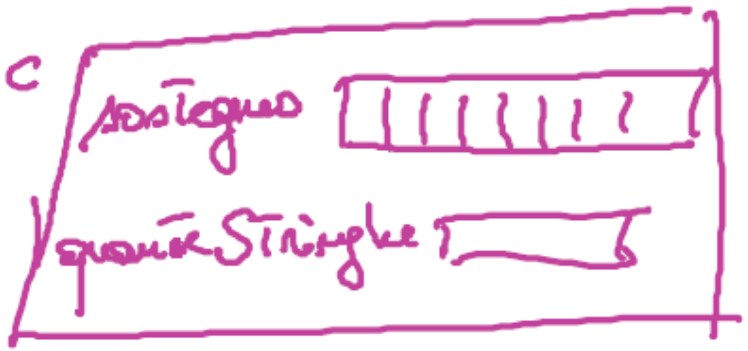
come viene il disegno di una variabile c di tipo TipoCollezione?

Un esempio di struttura astratta e sua implementazione

-Definizione di TipoCollezione a partire dalla definizione di tipo record precedente

```
typedef
   struct {
      char * sostegno[MAX];
      int quanteStringhe;
   }
TipoCollezione
```

TipoCollezione c



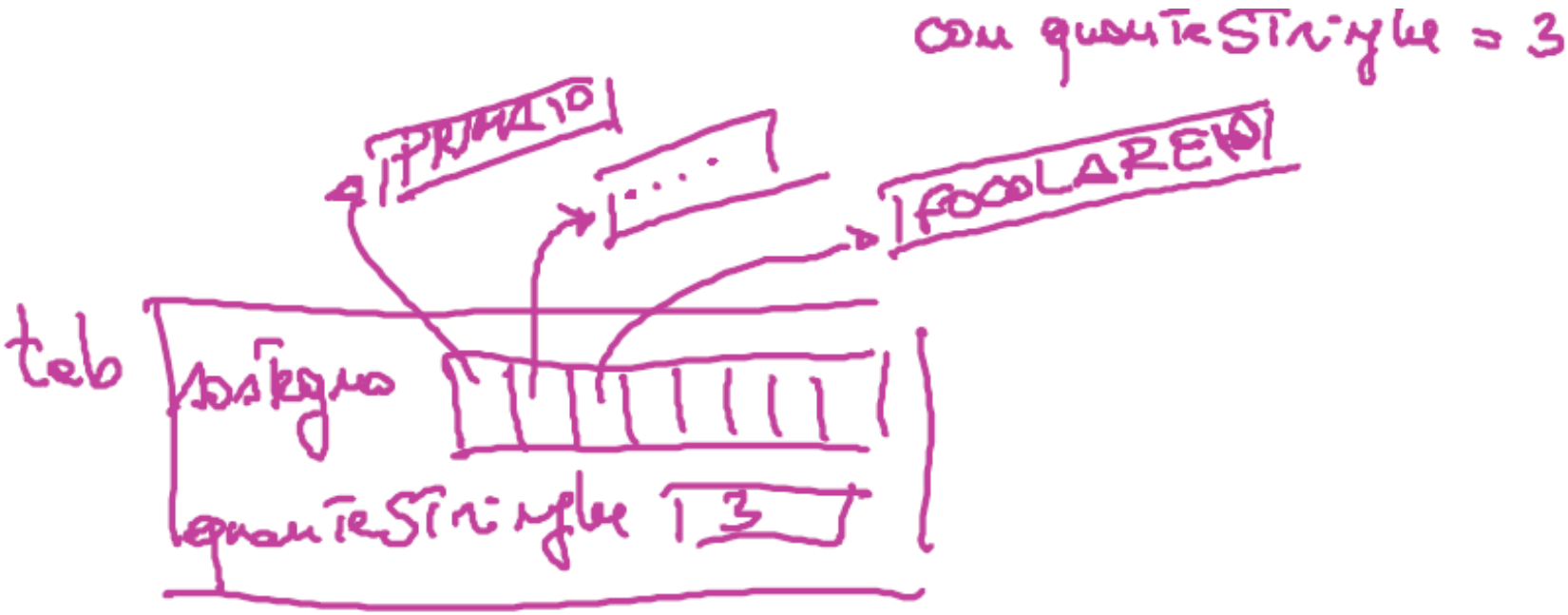
Un esempio di struttura astratta e sua implementazione

-dichiarazione di una variabile tab di tipo TipoCollezione, e sua mappa di memoria (con tre stringhe effettive)

Un esempio di struttura astratta e sua implementazione

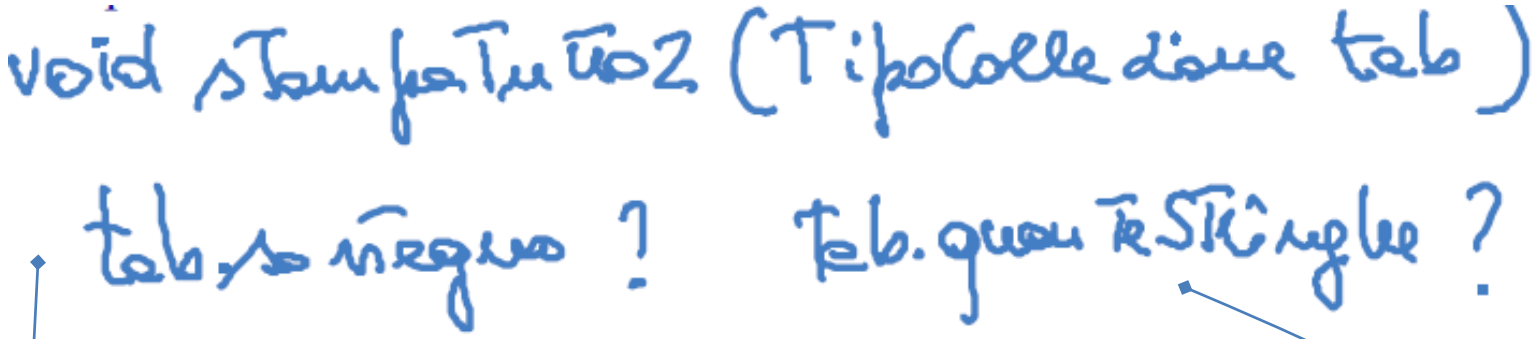
- dichiarazione di una funzione stampaTutto2 che riceve un solo parametro di tipo TipoCollezione per stamparla

-dichiarazione di una variabile tab di tipo TipoCollezione, e sua mappa di memoria (con tre stringhe effettive) `TipoCollezione tab`

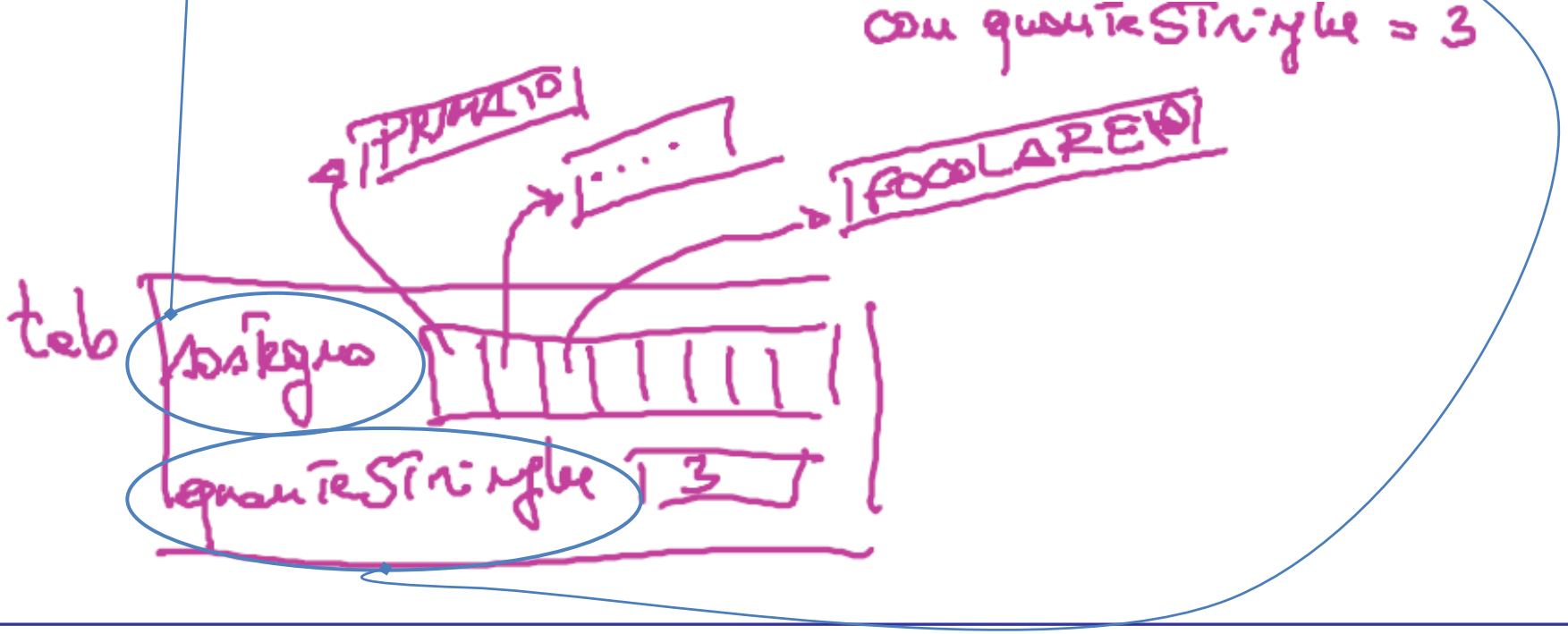


Un esempio di struttura astratta e sua implementazione

- dichiarazione di una funzione stampaTutto2 che riceve un solo parametro di tipo TipoCollezione per stamparla



-dichiarazione di una variabile tab di tipo TipoCollezione, e sua mappa di memoria

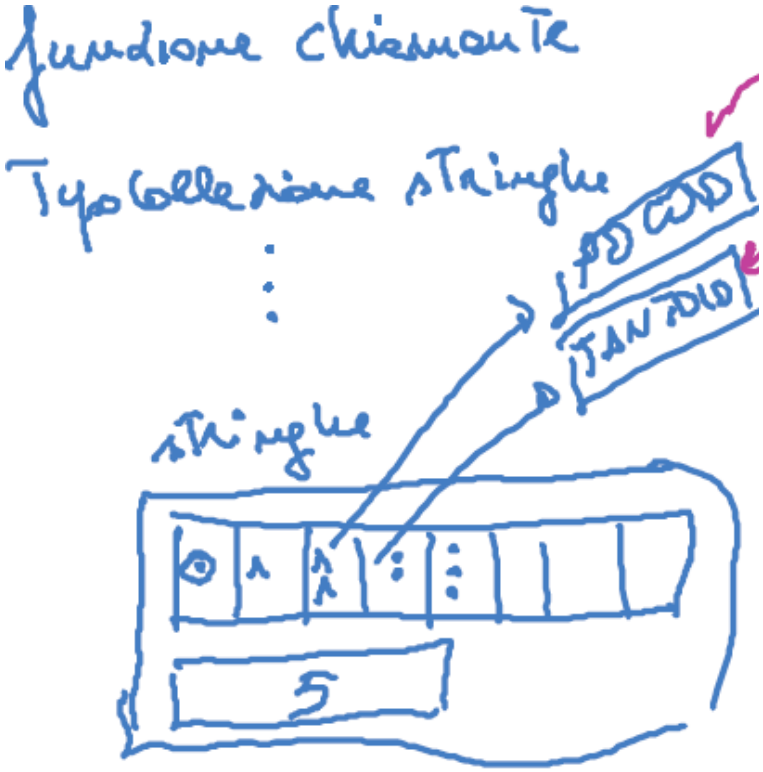


Un esempio di struttura astratta e sua implementazione

-mostrare la chiamata d stampaTutto2 in un ambiente di memoria opportuno e mostrare il RDA

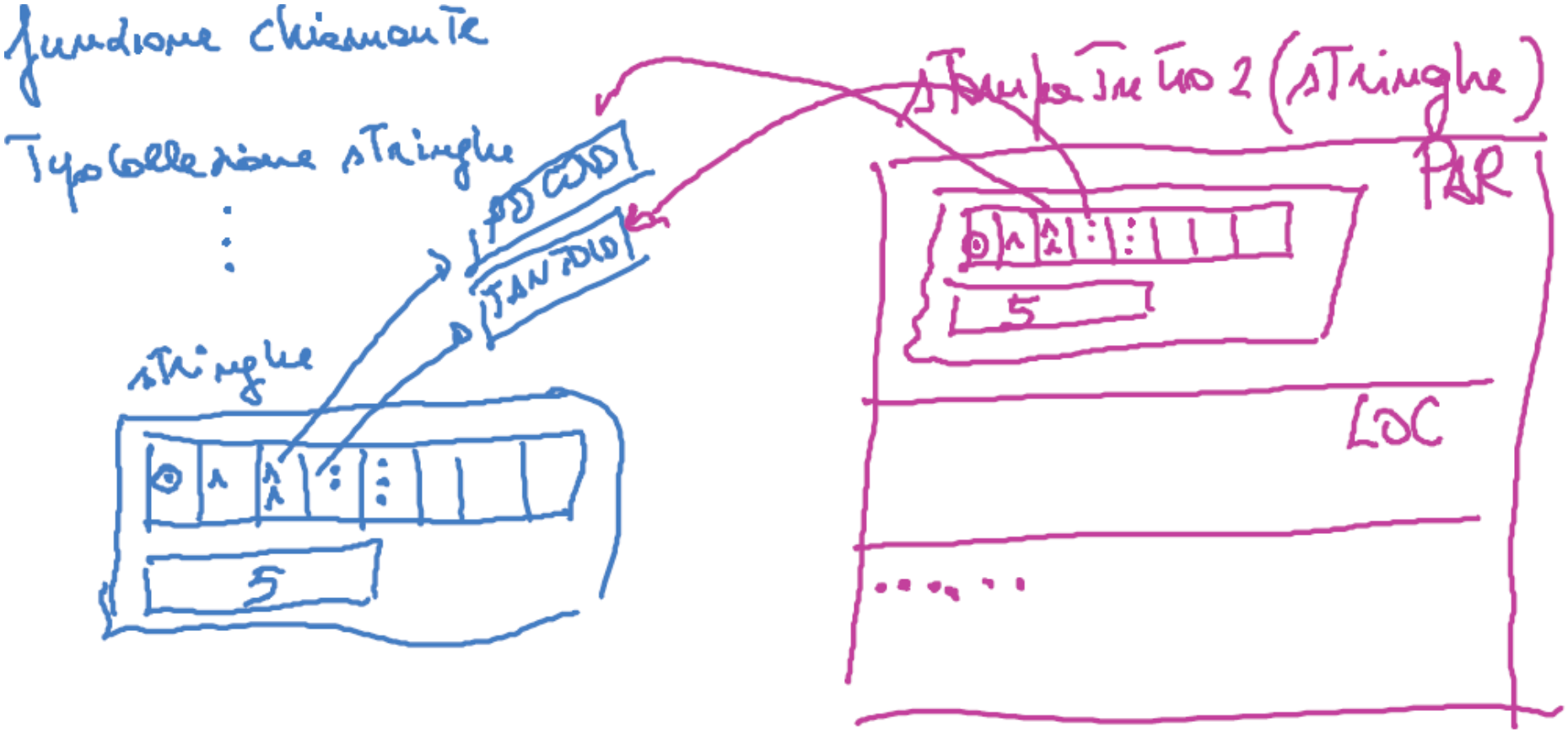
Un esempio di struttura astratta e sua implementazione

-mostrare la chiamata d stampaTutto2 in un ambiente di memoria opportuno e mostrare il RDA



Un esempio di struttura astratta e sua implementazione

-mostrare la chiamata d stampaTutto2 in un ambiente di memoria opportuno e mostrare il RDA



Un esempio di struttura astratta e sua implementazione

- idem come sopra per aggiungi2, che riceve due parametri (una collezione e una stringa da aggiungere): definizione (intestazione solo) ed esempio di chiamata

Un esempio di struttura astratta e sua implementazione

- idem come sopra per aggiungi2, che riceve due parametri (una collezione e una stringa da aggiungere): definizione (intestazione solo) ed esempio di chiamata

void aggiungi2 (TipoCollezione tab, char *str) { ... }

Un esempio di struttura astratta e sua implementazione

- idem come sopra per aggiungi2, che riceve due parametri (una collezione e una stringa da aggiungere): definizione (intestazione solo) ed esempio di chiamata

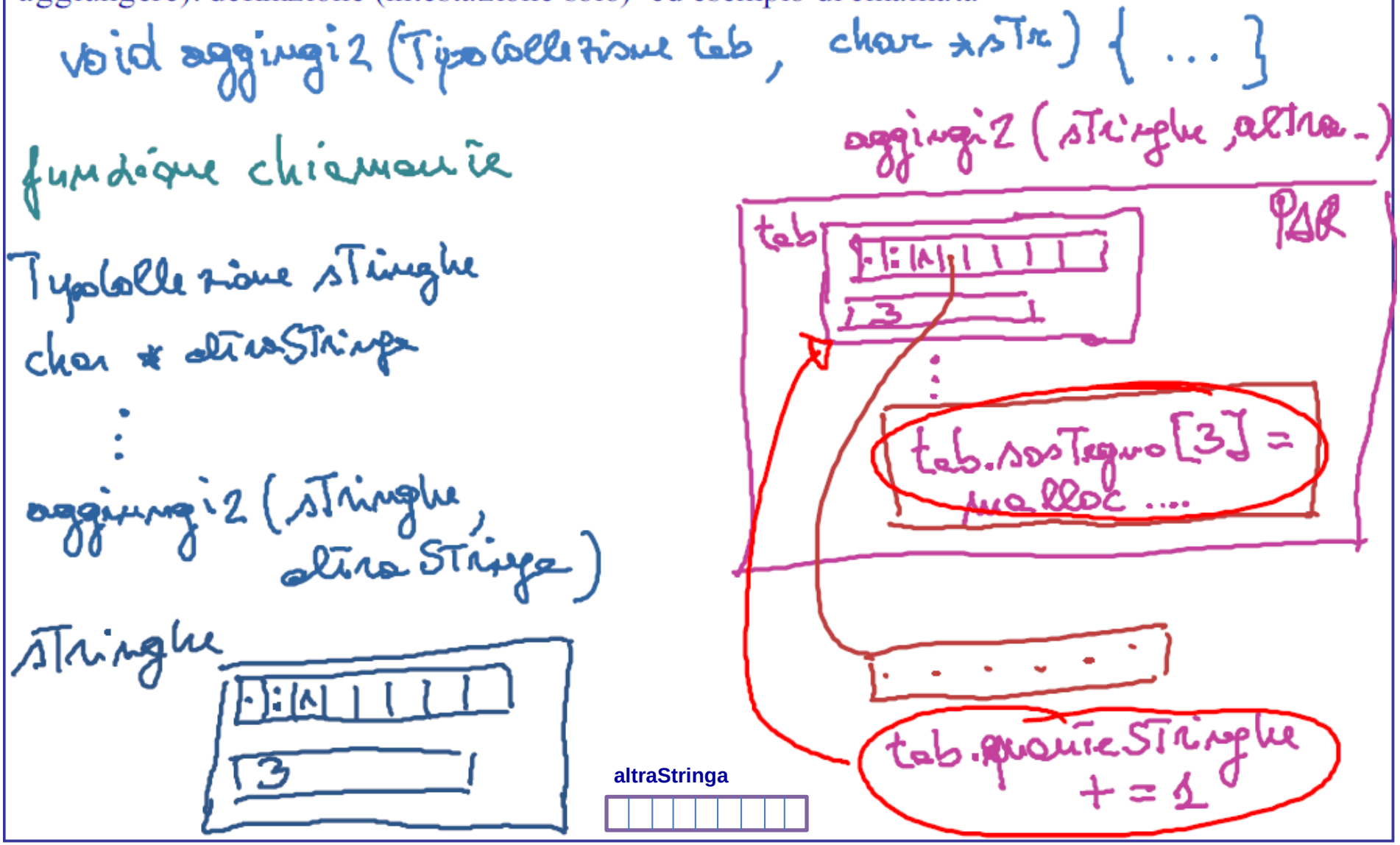
void aggiungi2 (TipoCollezione tab, char *sTr) { ... }

funzione chiamante

TipoCollezione sTinghe
char * altraStringa
:
aggiungi2 (sTinghe,
altraStringa)

Un esempio di struttura astratta e sua implementazione

- idem come sopra per aggiungi2, che riceve due parametri (una collezione e una stringa da aggiungere): definizione (intestazione solo) ed esempio di chiamata



Un esempio di struttura astratta e sua implementazione

- idem come sopra per aggiungi2, che riceve due parametri (una collezione e una stringa da aggiungere): definizione (intestazione solo) ed esempio di chiamata

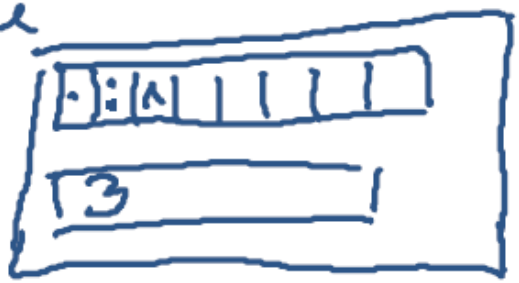
tab, char *str) { ... }

QUI C'E` UN PROBLEMA!!!

QUALE?

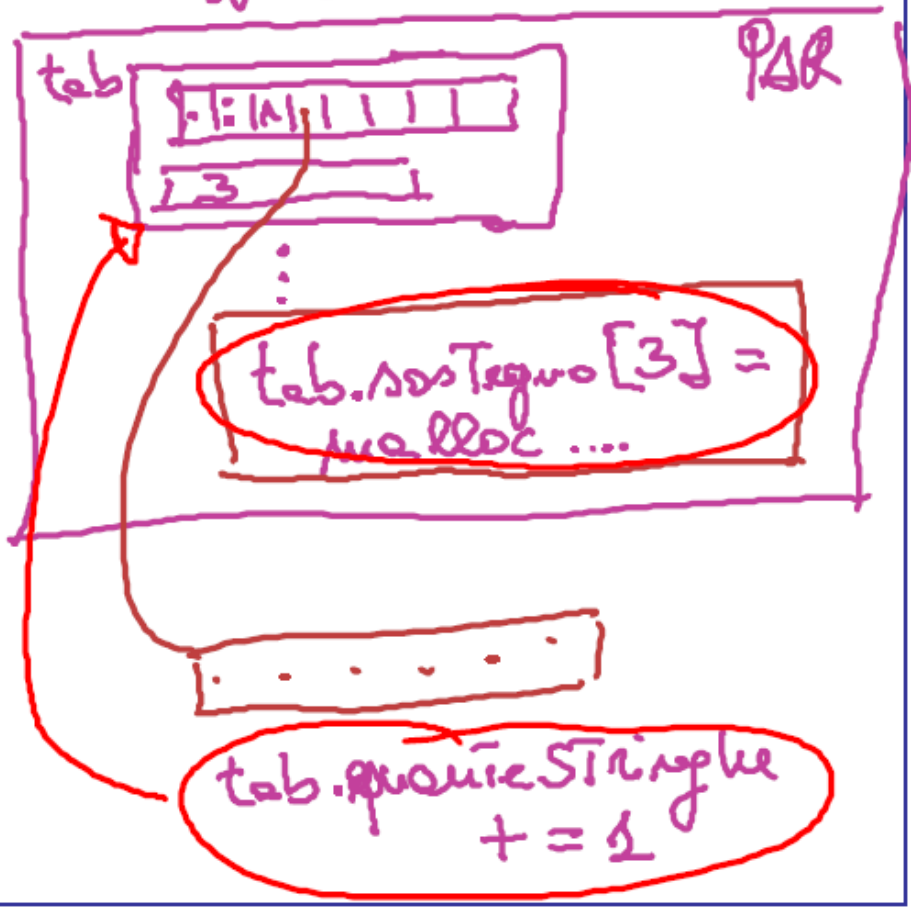
aggiungi2 (stringhe, altra stringa)

stringhe



aggiungi2 (stringhe, altra...)

PSR



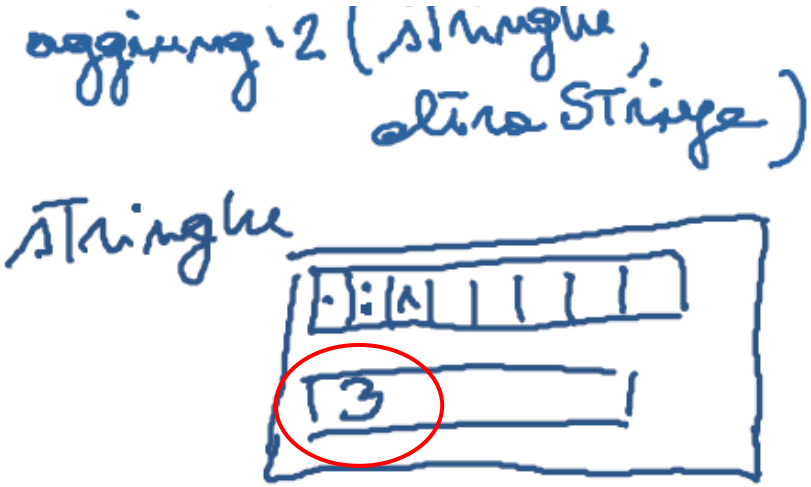
Un esempio di struttura astratta e sua implementazione

Il problema e` che tutte le modifiche vengono confinate nel RDA, sul parametro formale tab,

INVECE DI riflettersi sul parametro attuale stringhe … cioe` non ci sono gli effetti collaterali della funzione che dovrebbero esserci

Questo e` quel che succede se il parametro e` passato per valore: la funzione lavora su una copia del parametro attuale …

Per evitare questo problema, il passaggio deve essere tramite indirizzo – cosi` faremo nel prossimo esempio …



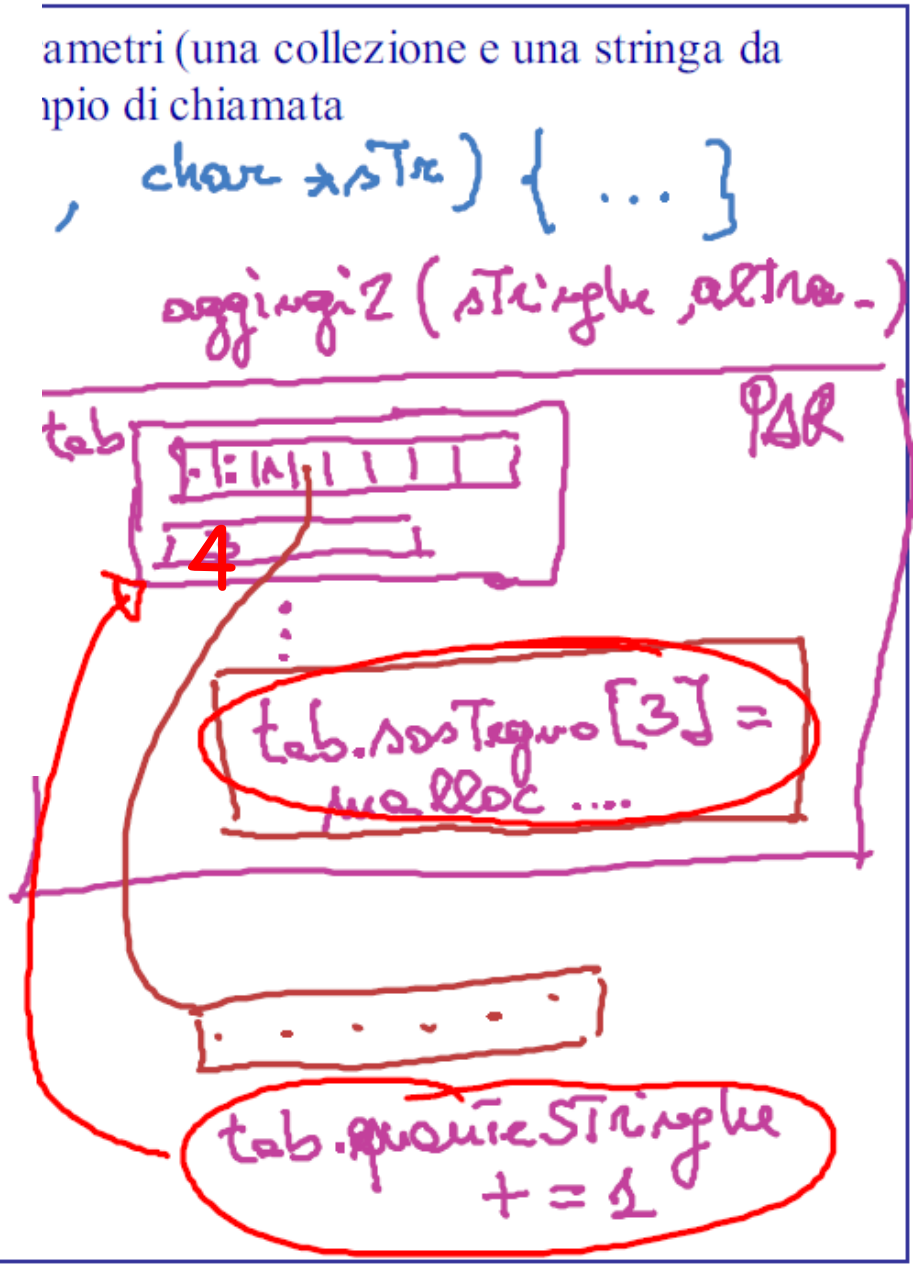


Tabella di Voli Aerei

e` una tabella i cui elementi sono VOLI - Un VOLO e` costituito dalle informazioni

- codice (5 caratteri alfanumerici) --- chiave identificativa
- destinazione (stringa alfanumerica)
- ora di partenza (es. 12:15)
- numero di posti liberi (intero)

Introduzione alla soluzione del problema della tabella dei voli aerei: ADT, implementazione con array statico

le funzionalita` disponibili devono essere

- aggiungere un volo **(AGGIUNTA)**
- eliminare un volo **(ELIMINAZIONE)**
- modificare il numero di posti liberi di un volo (dato codice); **(USO)**

  modificare l'ora di partenza di un volo (dato codice); **(USO)**

  stampare i voli in tabella; stampare un volo (dato codice); **(USO)**

  salvataggio della tabella; caricamento della tabella **(USO)**
- calcolo dell'indice di un volo nel sostegno della tabella, dato il codice **(RIC)**

Tabella di Voli Aerei - ADT (struttura dati astratta)

D_0 = dominio di interesse = {le tabelle di voli aerei}

D_1 = dominio = {i voli}

altri domini

D_2 = dominio = {i codici}

D_3 = dominio = {ore}

...

aggiungiVolo: …

StampaQuelVolo: …

…

…

la struttura teorica viene implementata

1. rappresentando concretamente i **domini D** con **tipi concreti T_D**
2. definendo **variabili di tipi T_D** per avere nel programma i rappresentanti di **elementi dei domini**
3. definendo **funzioni** per programmare le funzionalita` della struttura teorica

Tabella di Voli Aerei - ADT (struttura dati concreta) 1/2

1. rappresentando concretamente i **domini D** con **tipi concreti T_D**

D_1 = dominio = {i voli}

T_1 = tipo concreto = **TipoVolo** (una struct con i dati eterogenei di un volo)

Tabella di Voli Aerei - ADT (struttura dati concreta) 1/2

1. rappresentando concretamente i **domini D** con **tipi concreti T_D**

D_1 = dominio = {i voli}

T_1 = tipo concreto = **TipoVolo** (una struct con i dati eterogenei di un volo)

D_2 = dominio = {i codici}

T_2 = tipo concreto = char [6]

Tabella di Voli Aerei - ADT (struttura dati concreta) 1/2

1. rappresentando concretamente i **domini D** con **tipi concreti T_D**

D_1 = dominio = {i voli}

T_1 = tipo concreto = **TipoVolo** (una struct con i dati eterogenei di un volo)

D_2 = dominio = {i codici}

T_2 = tipo concreto = char [6]

D_3 = dominio = {ore}

T_3 = tipo concreto = **`<ore, minuti>`** coppia di interi (un'altra struct ...) (**TipoOra**)

Tabella di Voli Aerei - ADT (struttura dati concreta) 1/2

1. rappresentando concretamente i **domini D** con **tipi concreti T_D**

D_1 = dominio = {i voli}

T_1 = tipo concreto = **TipoVolo** (una struct con i dati eterogenei di un volo)

D_2 = dominio = {i codici}

T_2 = tipo concreto = char [6]

D_3 = dominio = {ore}

T_3 = tipo concreto = **`<ore, minuti>`** coppia di interi (un'altra struct ...) (**TipoOra**)

D_0 = dominio di interesse = {le tabelle di voli aerei}

T_0 = **TipoTabellaVoli** = **`<array di TipoVolo, intero quantiVoli>`**

Tabella di Voli Aerei - ADT (struttura dati concreta) 1/2

2. definendo **variabili di tipi T_D** per avere nel programma i rappresentanti di **elementi dei domini**

es. `TipoVolo unVolo;` `TipoTabellaVoli tabVoli;` …

Tabella di Voli Aerei - ADT (struttura dati concreta) 2/2

3. definendo **funzioni** per programmare le funzionalita` della struttura teorica

stampare la tabella

```
void stampaTabella (TipoTabellaVoli tab)
```

aggiunta di un volo

```
int aggiungiVolo (TipoTabellaVoli * t, TipoVolo v)
```

stampa di un volo

```
void stampaVolo (TipoVolo v)
```

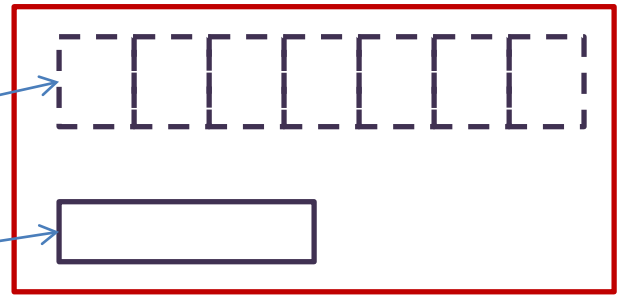
...

Tabella di Voli Aerei - tipi di base

```
struct volo {
   char codice[6];
   char * dest;
   TipoOra oraPartenza;
   int postiLiberi;
}
typedef struct volo TipoVolo;
```

(implementazione mediante array statico)

(questa e` una variabile TipoTabellaVoli tabVoli)




```
struct ora {int ore, min }
typedef struct ora
        TipoOra;
```

```
typedef    struct {
              TipoVolo arrayVoli[MAXVOLI];
              int quantiVoli;
           }
           TipoTabella (o TipoTabellaVoli – piu` lungo ma piu` preciso)
```

Tabella di Voli Aerei - tipi di base

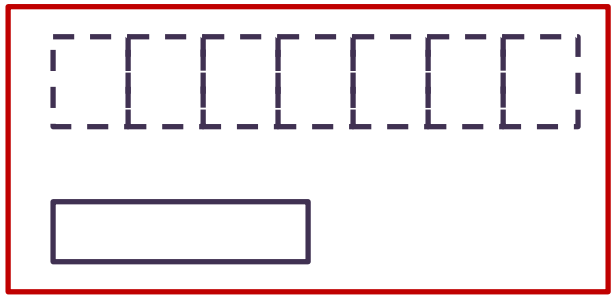
```
struct volo {
   char codice[6];
   char * dest;
   TipoOra oraPartenza;
   int postiLiberi;
}
typedef struct volo TipoVolo;

struct ora {int ore, min }

typedef struct ora   TipoOra;
typedef    struct {
              TipoVolo arrayVoli
              int quantiVoli;
           }
         TipoTabella
```

(implementazione mediante array statico)

(questa e` una variabile
TipoTabellaVoli tabVoli)

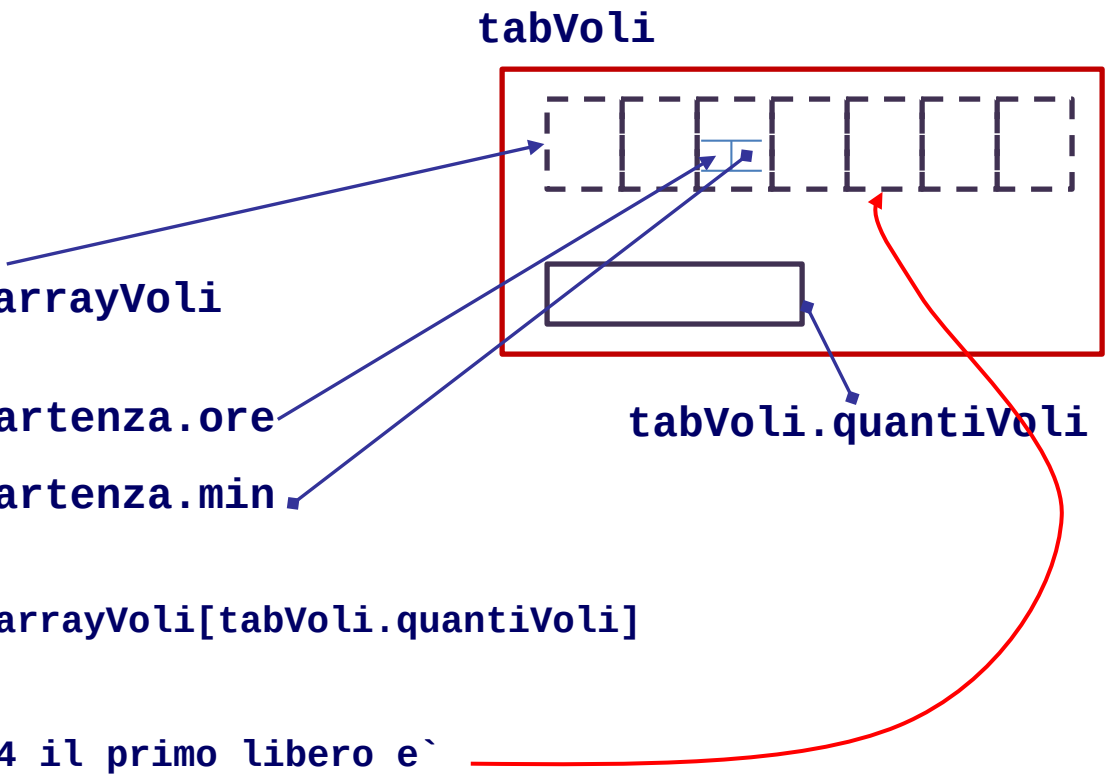


Che cosa e`

- **tabVoli.arrayVoli**
- **tabVoli.quantiVoli**
- **tabVoli.arrayVoli[2].oraPartenza.ore**
- **tabVoli.arrayVoli[2].oraPartenza.min**
- **Il primo elemento vuoto del sostegno**

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
int main() {
  TipoTabella tabVoli;
```



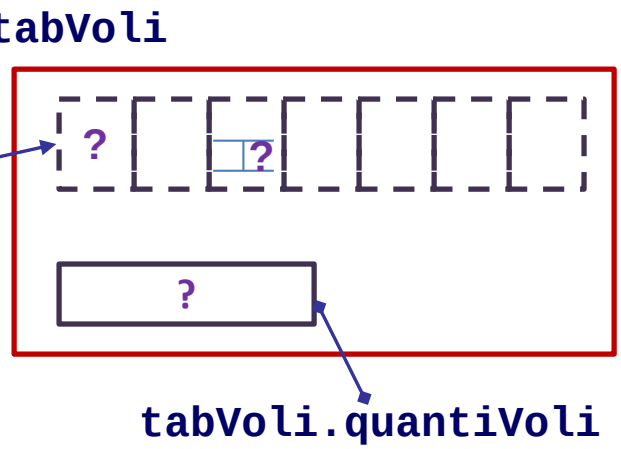
primo elemento libero … tabVoli.arrayVoli[tabVoli.quantiVoli]

esempio: se tabVoli.quantiVoli==4 il primo libero e`

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
int main() {

  TipoTabella tabVoli;
```



inizialmente la tabella e` vuota;

Remember … questa struttura dati va INIZIALIZZATA prima di usarla.

Come?

`tabVoli = 0` ??

… macche`

dopo ogni aggiunta viene modificato **`quantiVoli`** (aggiungendo 1);

`(tabVoli.quantiVoli` e` il numero di voli gia` presenti nel sostegno della tabella)

`(tabVoli.arrayVoli[tabVoli.quantiVoli]` e` invece il primo elemento "vuoto" del sostegno, cioe` il primo disponibile a contenere un nuovo volo)

quindi un'aggiunta si puo` fare in **`tabVoli.arrayVoli[tabVoli.quantiVoli]`**

solo se **`tabVoli.quantiVoli`** e` minore di **`MAXVOLI`**

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
int main() {
  TipoTabella tabVoli;
  int riuscita,              scelta;  /* scelta nel menu` */
  char buffer[40];

   tabVoli.quantiVoli=0;   /* inizializzazione */
   do {
     printf(" -      scegli                    -\n");
     printf(" - stampa dei voli           (1) -\n");
     printf(" - stampa di un certo volo   (2) -\n");
     printf(" - aggiunta di un volo       (3) -\n");
     printf(" - fine                      (0) -\n");
     scanf("%d", &scelta);
     switch (scelta) {. . . . . .
     } /* fine switch */
   } while (scelta!=0);       /* fine do_while*/    .....
```

tabVoli = 0 ??
… macche` … pfui

cosi` si fa ...

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
int main() {
  TipoTabella tabVoli;
  ....
     switch (scelta) {
       case 1:
          printf(" - %d voli in tabella:\n", tabVoli.quantiVoli);
          stampaTabella(tabVoli);
          break;
       case 2:
          printf(" - codice volo? ");
          scanf("%s", buffer);
          stampaQuelVolo(tabVoli, buffer);
          break;
       case 3:
          riuscita=aggiungiVolo(&tabVoli);
.....
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
....
       case 3:
          riuscita=aggiungiVolo(&tabVoli);
          if(!riuscita)
            printf(" - aggiunta non effettuata -\n");
          else
            printf(" - fatto -\n");
          break;
       case 0:
          printf(" - USCITA DAL PROGRAMMA\n");
          break;
       default:
          printf(" - opzione sballata\n");
     } /* fine switch */
   } while (scelta!=0);        /* fine do_while*/
  printf("\nFINE\n");  return 0;  }
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
prototipi/definizioni di funzioni fin qui ....
void stampaTabella (TipoTabella t) {
  int i;
  for(i=0; i< t.quantiVoli; i++) {
      stampaVolo (t.arrayVoli[i]);
      printf("\n");
  }
return;  }


/* stampaQuelVolo, stampaVolo, aggiungi */
void stampaQuelVolo (TipoTabella t, char cod[6]);

void stampaVolo (TipoVolo v) {
 printf(" VOLO %s (%d posti disponibili), partenza alle %d:%2d per %s", v.codice,
     v.postiLiberi, v.oraPartenza.ore, v.oraPartenza.min, v.dest);
return; }

int aggiungiVolo (TipoTabella *t);          (NB … passaggio di indirizzo)
```

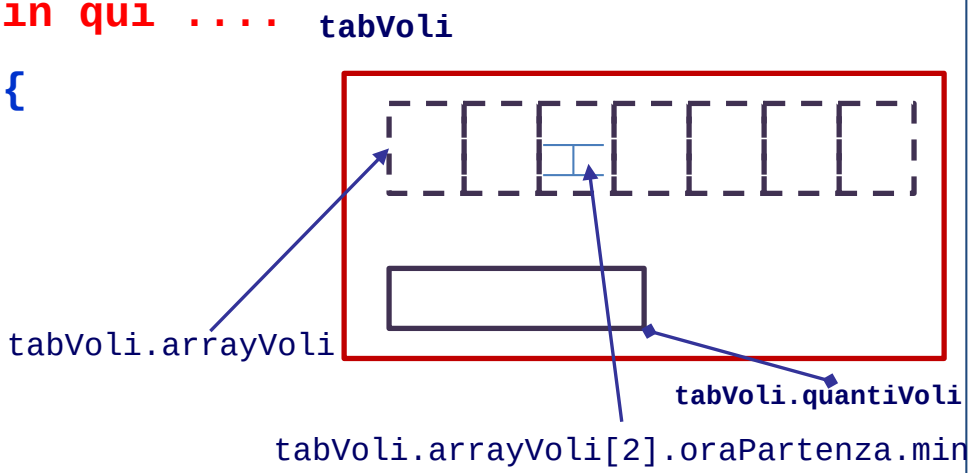
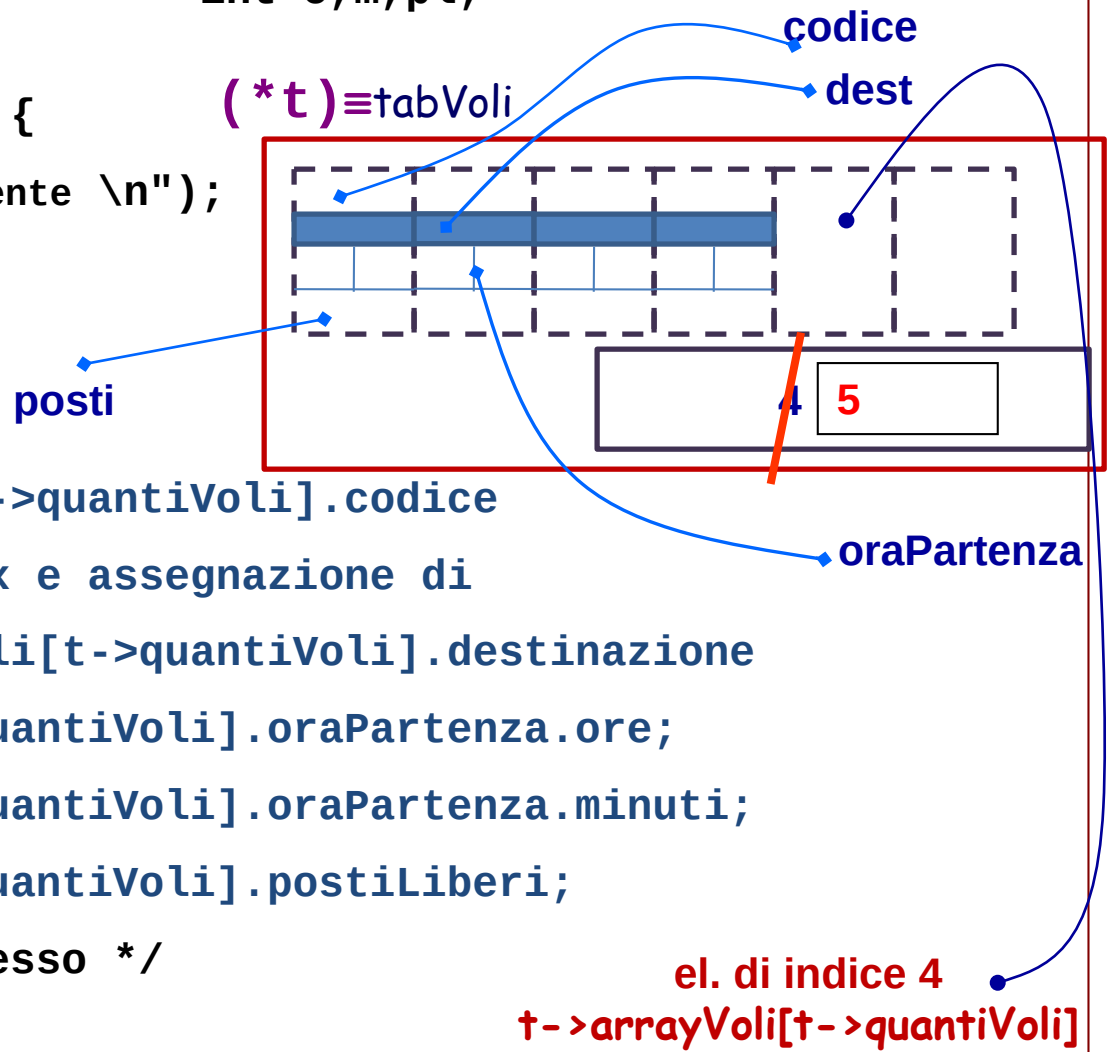


Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
int aggiungiVolo(TipoTabella *t) {
  char buffer[50], *aux;                    int o,m,pl;

  if (t->quantiVoli==MAXVOLI) {
    printf("\n spazio insufficiente \n");
    return 0;
  }
  else {
    LETTURA di t->arrayVoli[t->quantiVoli].codice
    LETTURA della  stringa aux e assegnazione di
                   t->arrayVoli[t->quantiVoli].destinazione
    LETTURA t->arrayVoli[t->quantiVoli].oraPartenza.ore;
    LETTURA t->arrayVoli[t->quantiVoli].oraPartenza.minuti;
    LETTURA t->arrayVoli[t->quantiVoli].postiLiberi;
  /* l'aggiunta ha avuto successo */
    (t->quantiVoli)+=1;
  }
```




```
                                          return 1;   }
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
prototipi/definizioni di funzioni fin qui ....
void stampaTabella (TipoTabella t) {
  int i;
  for(i=0; i< t.quantiVoli; i++) {
      stampaVolo (t.arrayVoli[i]);
      printf("\n");
  }
return;  }

/* stampaQuelVolo, stampaVolo, aggiungi */
void stampaQuelVolo (TipoTabella t, char cod[6]);    ☺

void stampaVolo (TipoVolo v) {
 printf(" VOLO %s (%d posti disponibili), partenza alle %d:%2d per %s", v.codice,
     v.postiLiberi, v.oraPartenza.ore, v.oraPartenza.min, v.dest);
 return; }

int aggiungiVolo (TipoTabella *t);          (NB … passaggio di indirizzo)
```

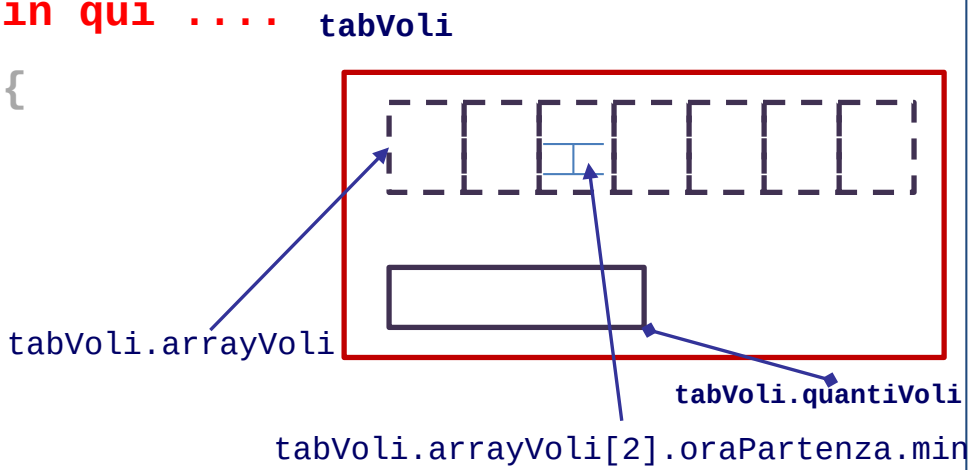
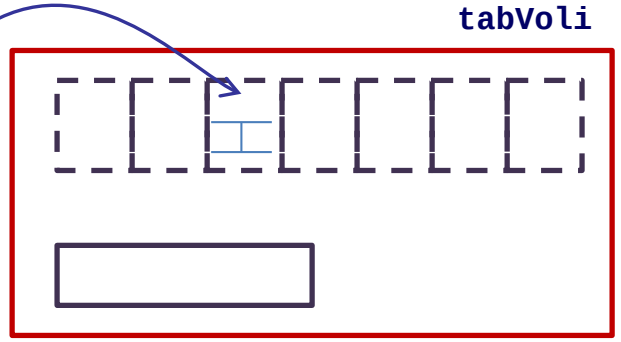


Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
void stampaQuelVolo (TipoTabella t, char cod[6]) {
   int k = indiceVolo(t, cod);


   if (k==-1) printf(...  non c'e` ....);
   else stampaVolo (t.arrayVoli[k]);
return;
}
```

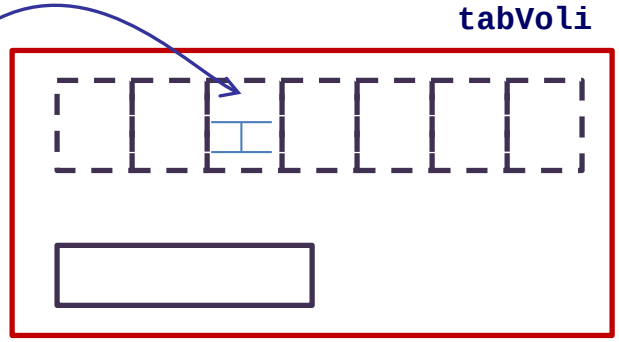



```
                    /* funzione accessoria indiceVolo()*/
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
void stampaQuelVolo (TipoTabella t, char cod[6]) {
   int k = indiceVolo(t, cod);


   if (k==-1) printf(...  non c'e` ....);
   else stampaVolo (t.arrayVoli[k]);
return;
}
```




```
                                /* funzione accessoria indiceVolo()*/
int indiceVolo (TipoTabella tb, char *code) {
 int trovato = 0, i=0;
```

Mentre t.arrayVoli[i] “e` un volo” e "stiamo cercando"

Se e` il volo code trovato!!! esci

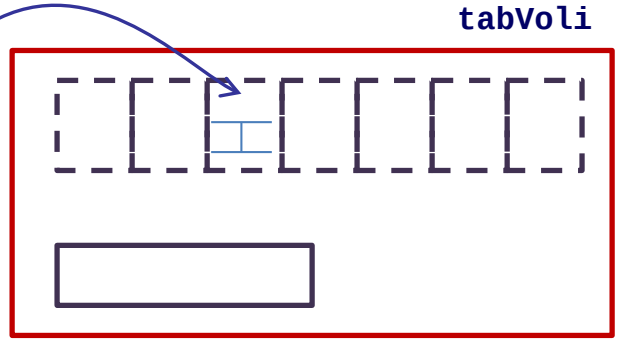
Senno` incrementa i

```
 if (trovato) return i;
 else return -1;                                }
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
void stampaQuelVolo (TipoTabella t, char cod[6]) {
   int k = indiceVolo(t, cod);


   if (k==-1) printf(...  non c'e` ....);
   else stampaVolo (t.arrayVoli[k]);
return;
}
                                /* funzione accessoria indiceVolo()*/
int indiceVolo (TipoTabella tb, char *code) {
 int trovato = 0, i=0;
 while ( (i<tb.quantiVoli) && !trovato )
    if (strcmp(tb.arrayVoli[i].codice, code)==0)
       trovato = 1;
    else i++;
 if (trovato) return i;
 else return -1;                                  }
```



Mentre t.arrayVoli[i] “e` un volo” e "stiamo cercando"

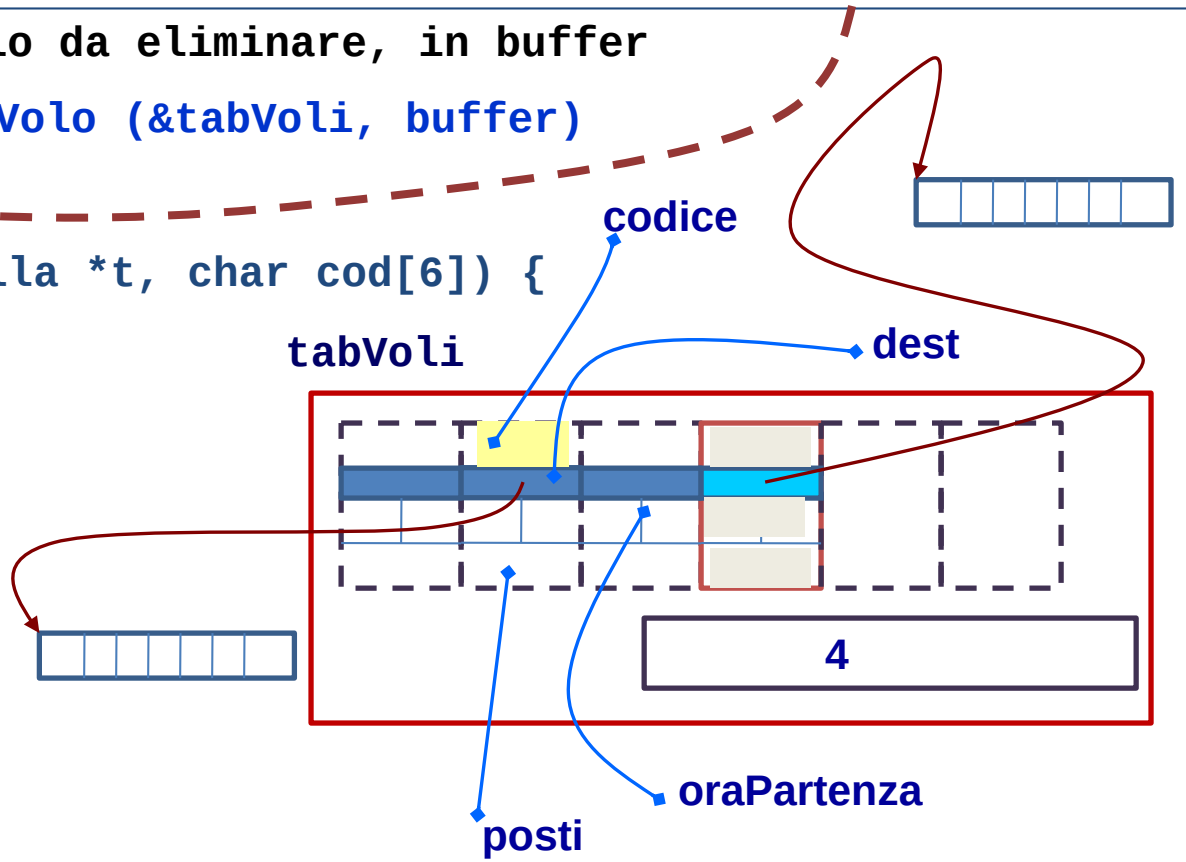
Se e` il volo code trovato!!! esci

Senno` incrementa i

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```

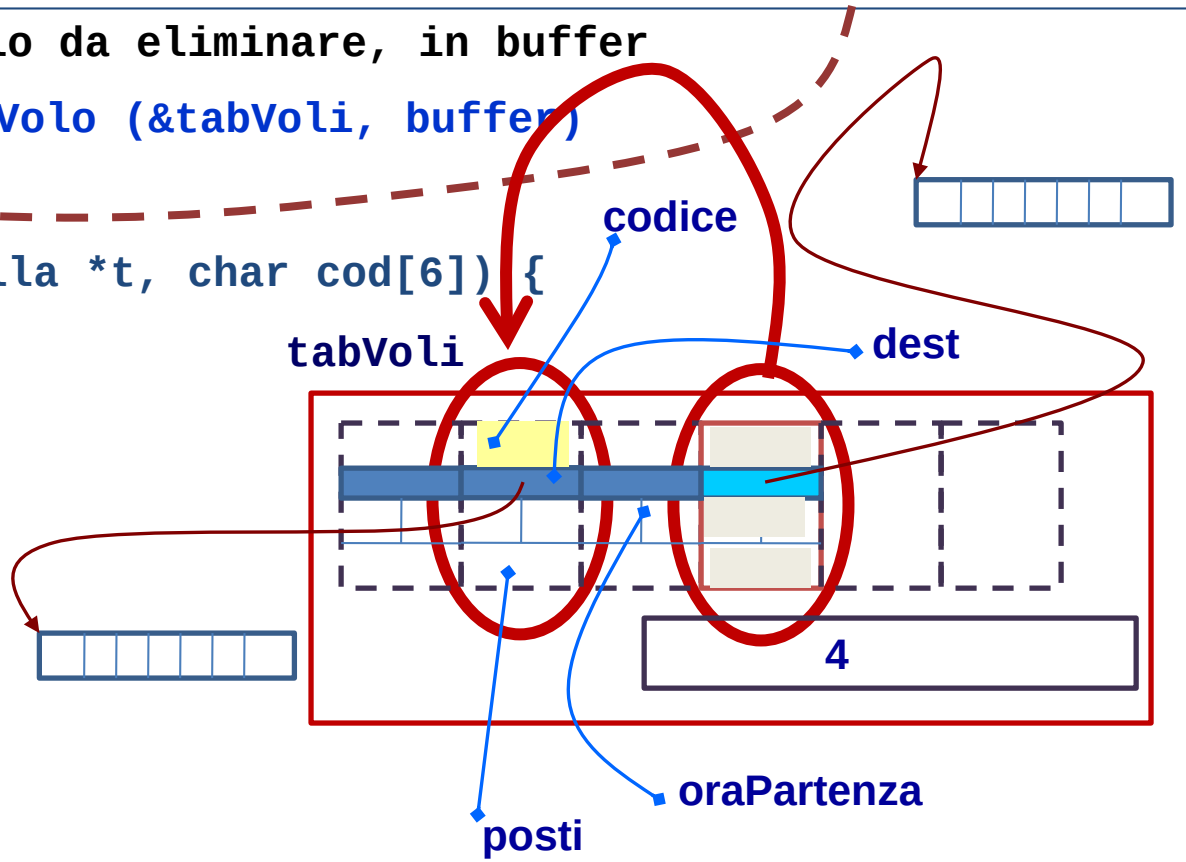


- ricerca volo da eliminare usando indiceVolo) - sia k l'indice del volo da liminare
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```



in sostanza va copiato l'ultimo elemento sopra a quello da eliminare

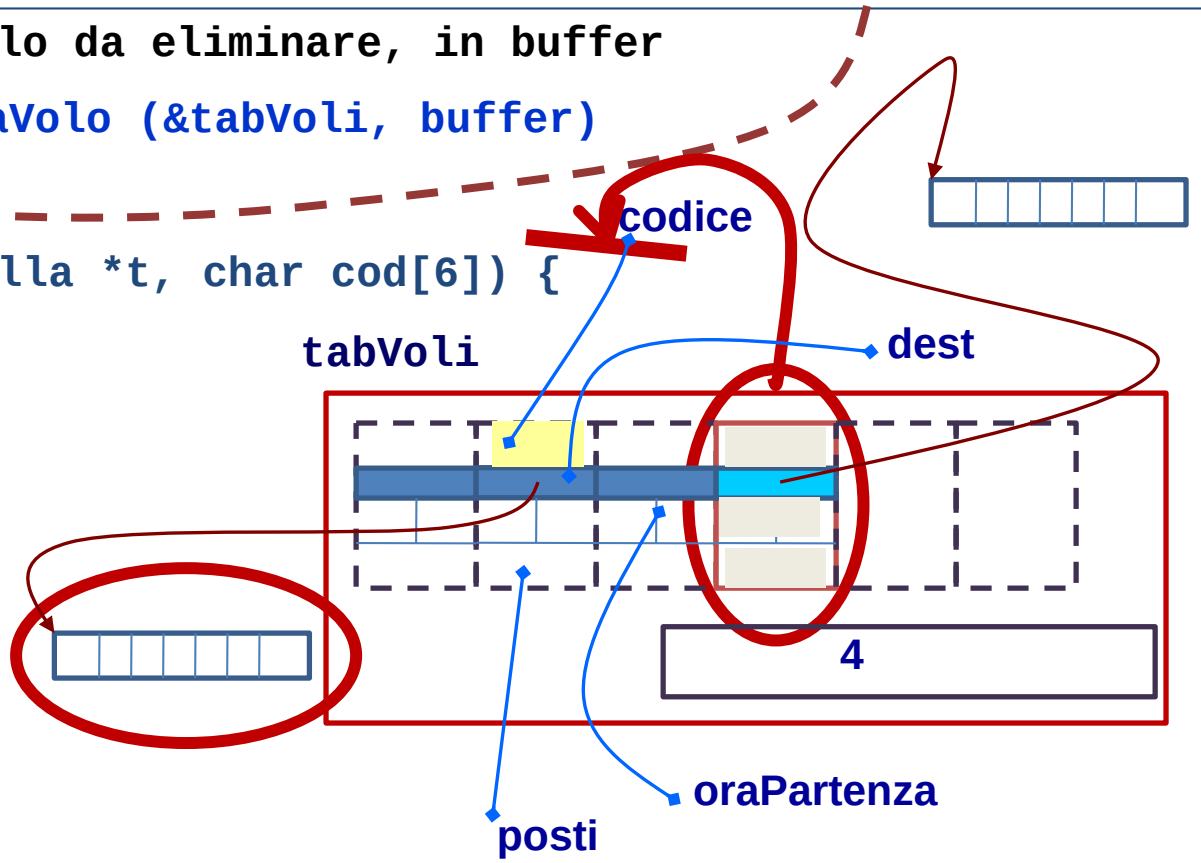
MA …

- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```



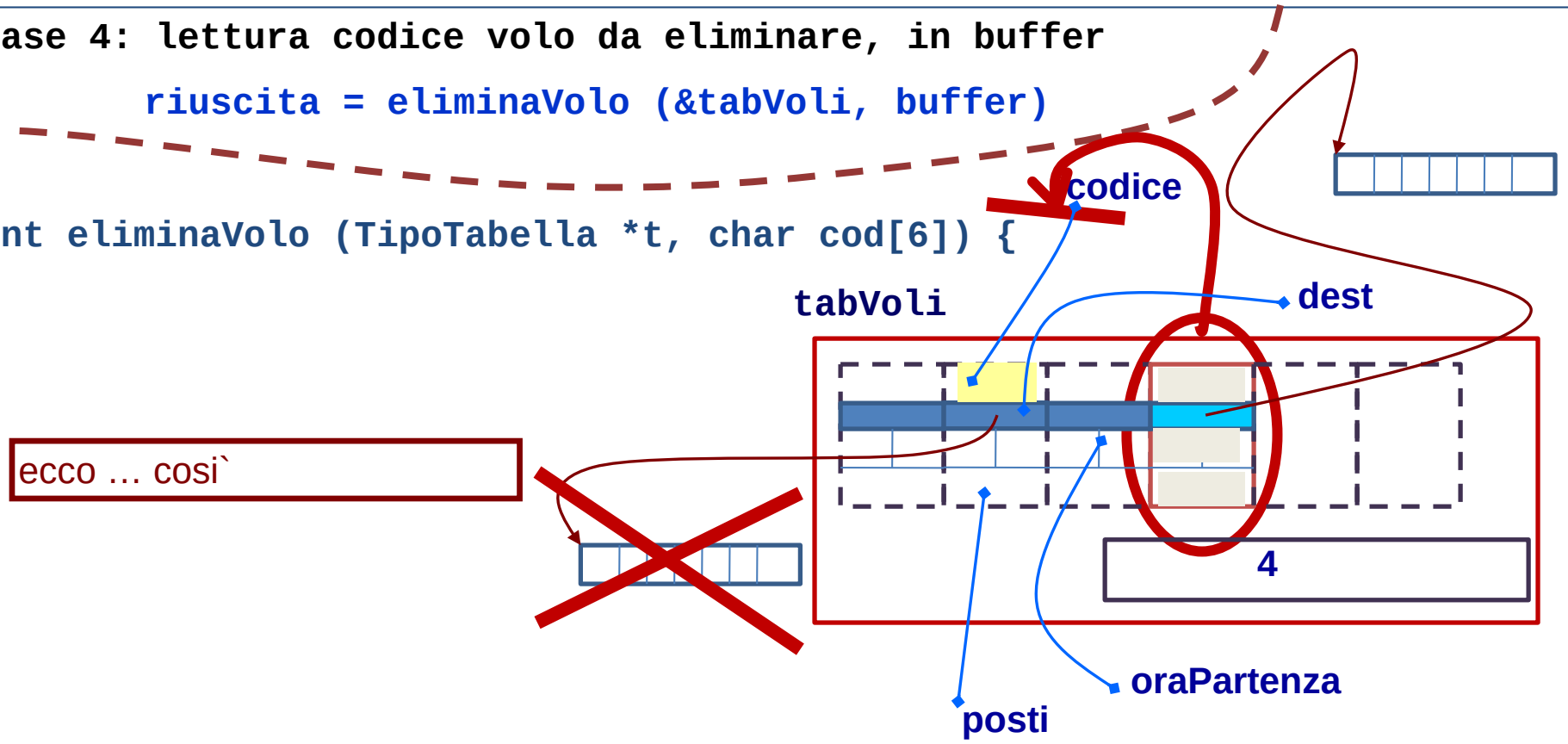
prima di eliminarlo bisogna deallocare la stringa esatta che e` la destinazione del volo da eliminare

- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```

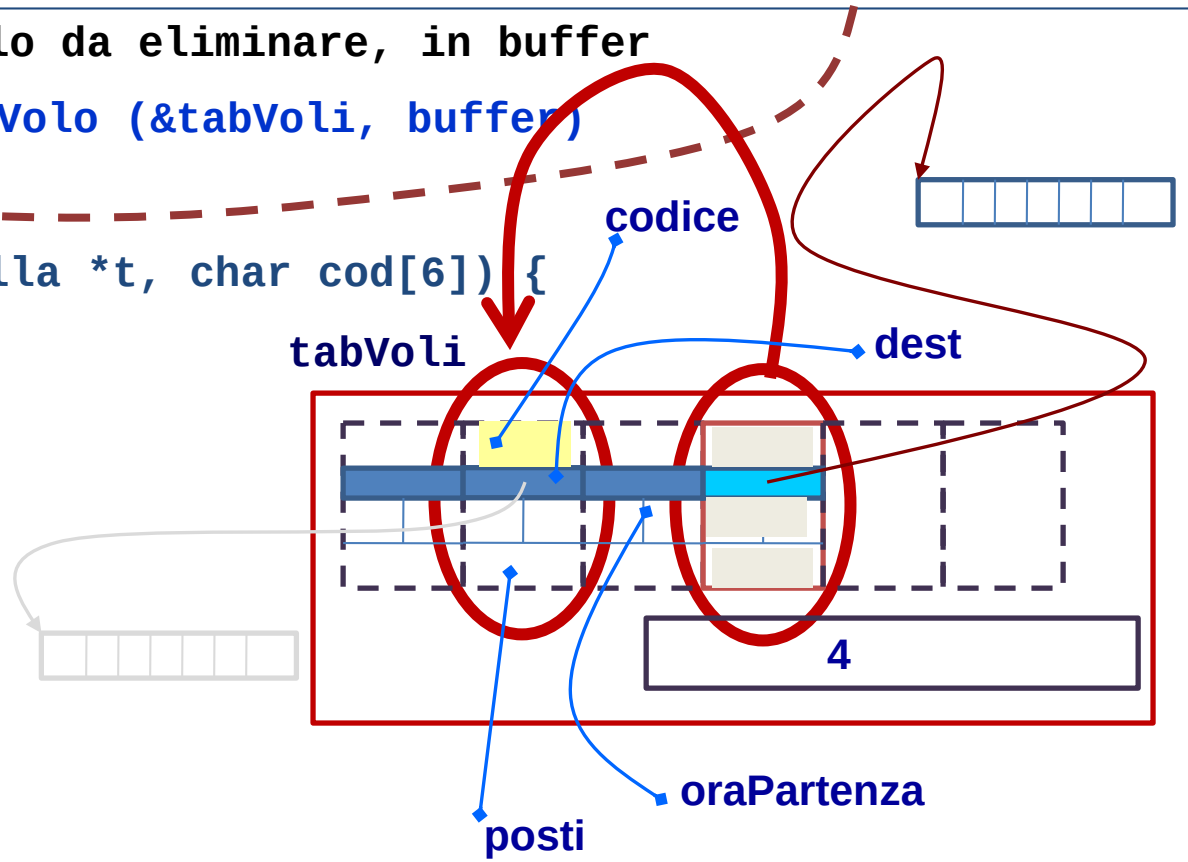


- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```



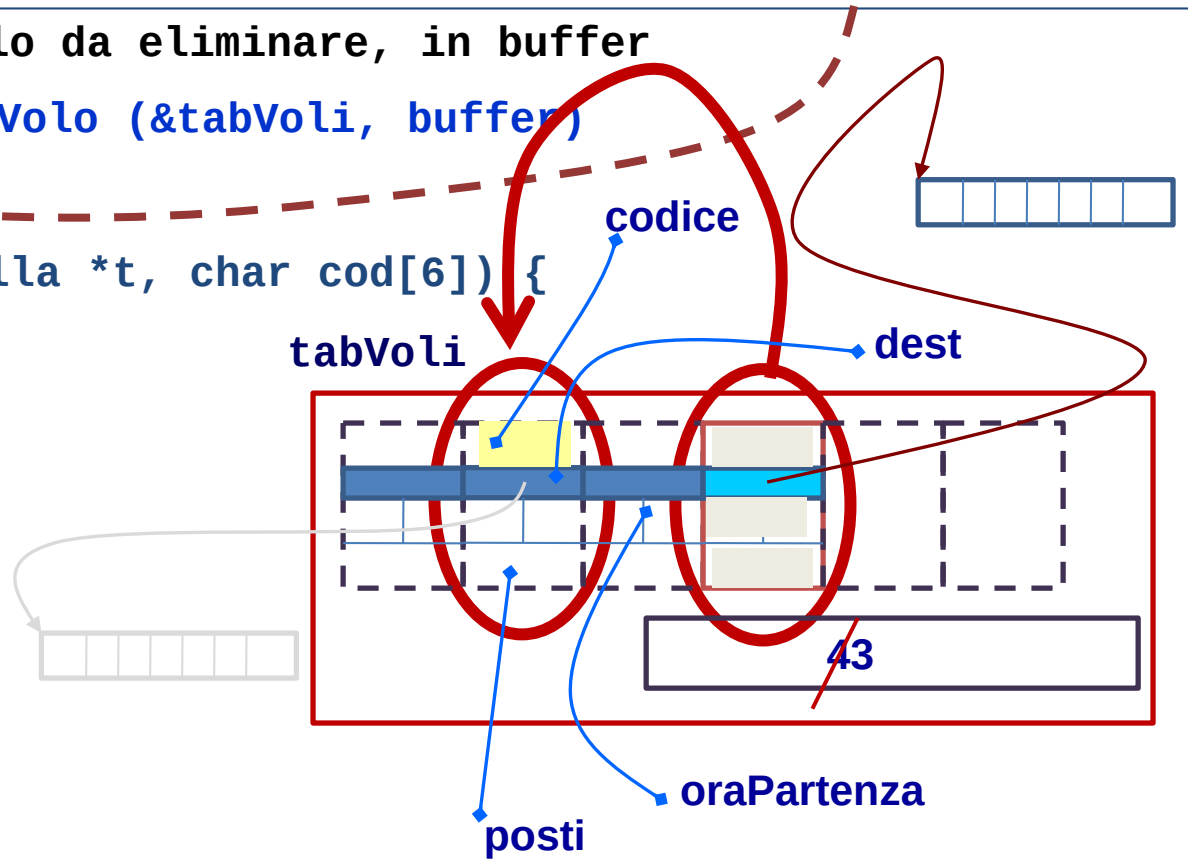
e poi si puo` fare la copia

- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```



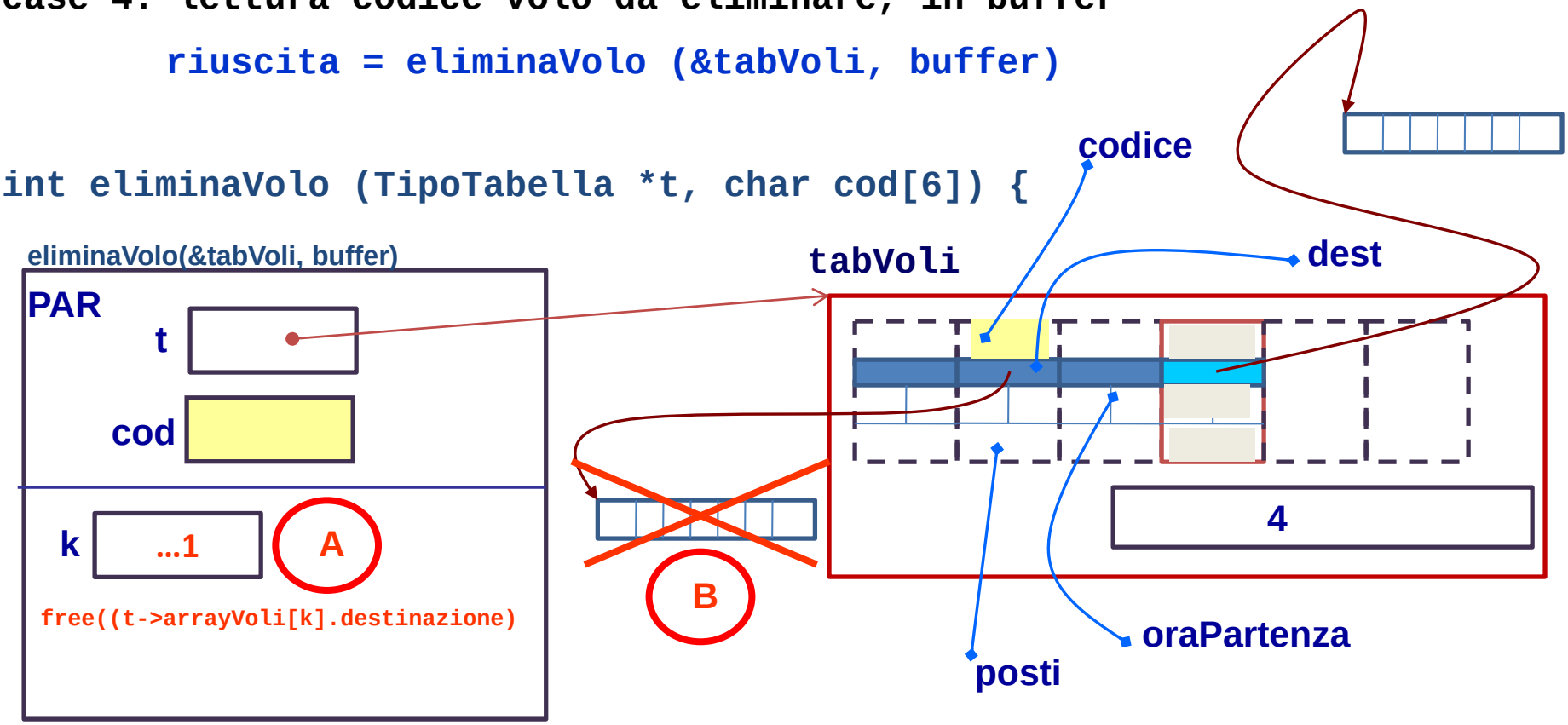
e decrementare quantiVoli

- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```

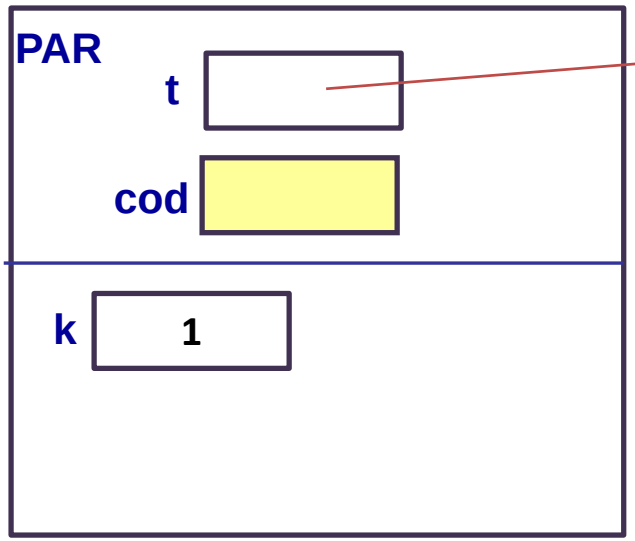


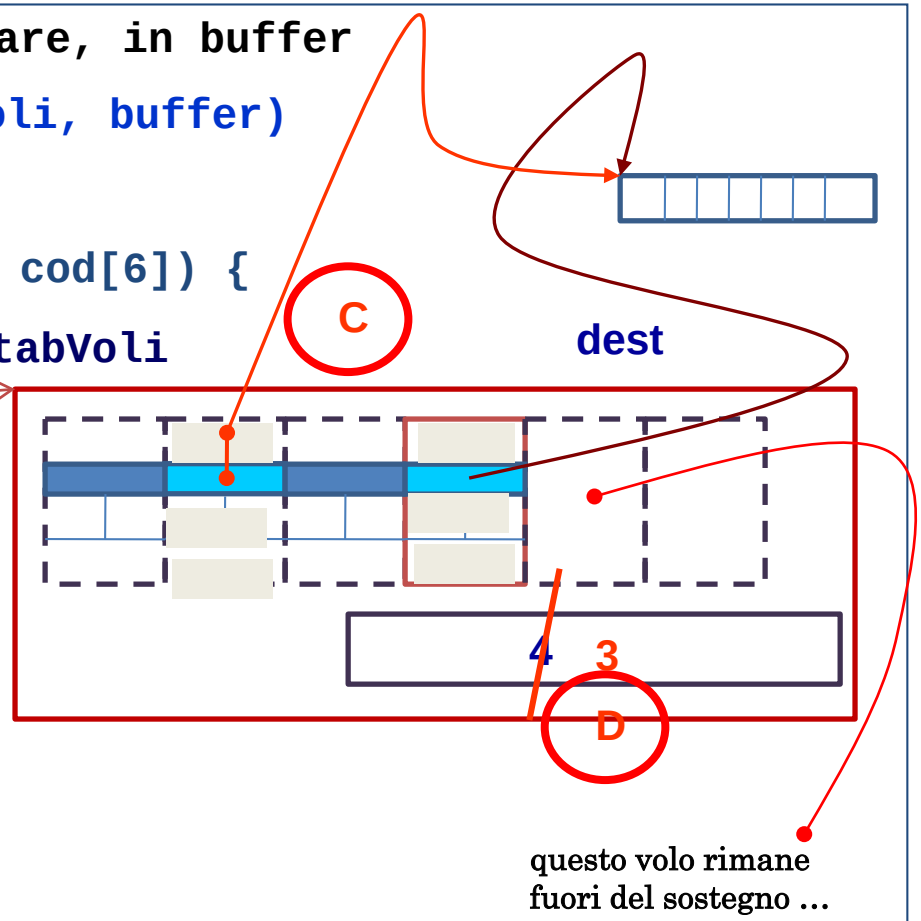
- ricerca volo da eliminare con indiceVOlo) … in indice k (A)
- eliminazione stringa destinazione da t->arrayVoli[k] (B)
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k]
- t->quantivoli -= 1

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

int eliminaVolo (TipoTabella *t, char cod[6]) {
```





- ricerca volo da eliminare con indiceVOlo)- sia codice k
- eliminazione stringa destinazione da t->arrayVoli[k]
- copia di t->arrayVoli[t->quantiVoli-1] in t->arrayVoli[k] C
- t->quantivoli -= 1 D