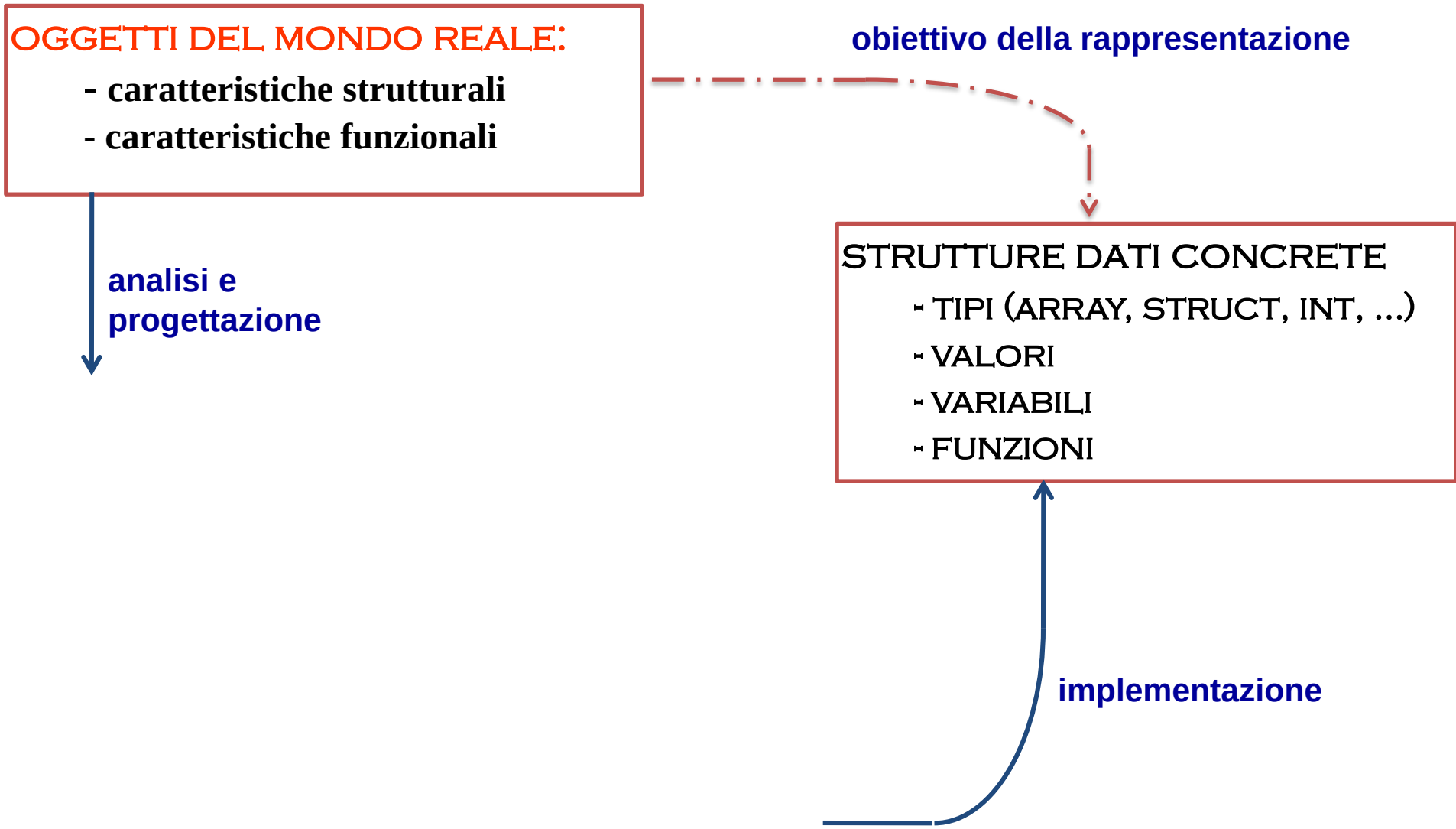
Rappresentazione di dati



formale: su carta: indipendente dal linguaggio

Un esempio di struttura astratta e sua implementazione

L'ADT TABELLA

Un esempio di struttura astratta e sua implementazione

ADT "**collezione di stringhe**"

D0 = dominio di interesse =

D1 = dominio = {le

D2 = dominio = {

funzionalita`

aggiungi: D0 × D1 ⟶ D0

stampaTutto:

ricerca: D0 × D1 D2

implementazione dell'ADT "collezione di stringhe"

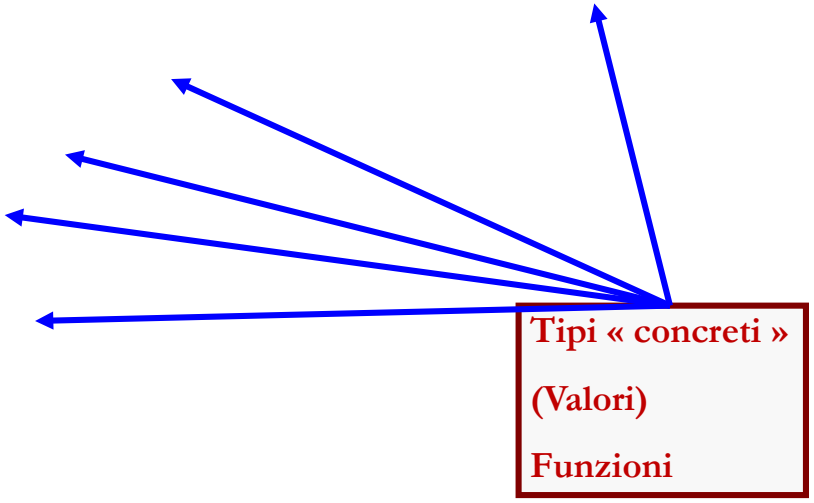
D0 = dominio di interesse = {le collezioni di stringhe} =

D1 = dominio = {le stringhe} =

D2 = dominio = {indici} =

D3 = {gli elenchi di stringhe}

D0 = D3 × D2



funzionalita`

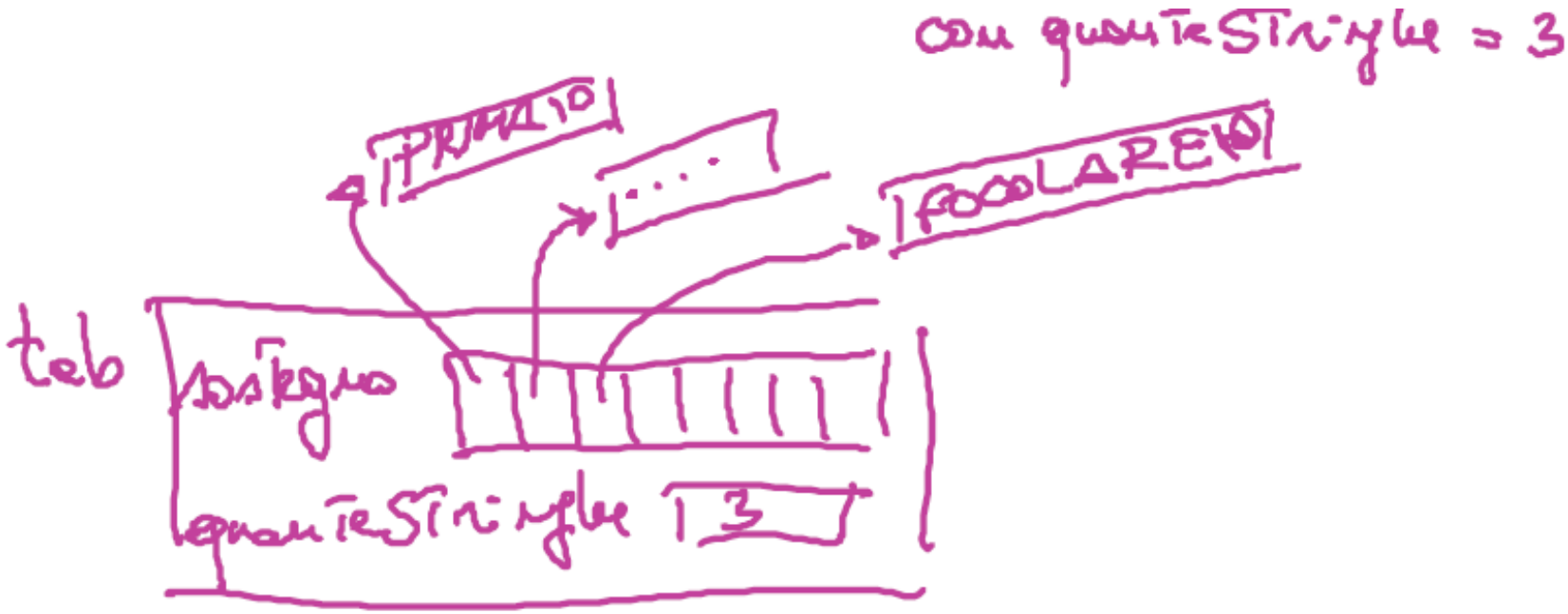
Un esempio di struttura astratta e sua implementazione

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

Un esempio di struttura astratta e sua implementazione

- 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

-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)

Tabella di Voli Aerei

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

- codice
- destinazione
- ora di partenza ( )
- numero di posti liberi (intero)

le funzionalita` disponibili devono essere

- **(AGGIUNTA)**
- **(ELIMINAZIONE)**
- **(USO)**

- **(RIC)**

Tabella di Voli Aerei - ADT (struttura dati astratta)

D_0 = dominio di interesse = { }

D_1 = dominio = { }

altri domini

la struttura teorica viene implementata

1. rappresentando concretamente
2. definendo **variabili di tipi T_D**
3. definendo **funzioni** per

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

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

es.

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

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

stampare la tabella

aggiunta di un volo

`int`

stampa di un volo

`void`

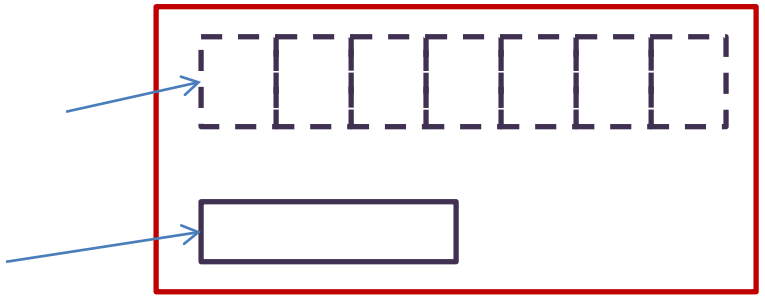
...

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)
```



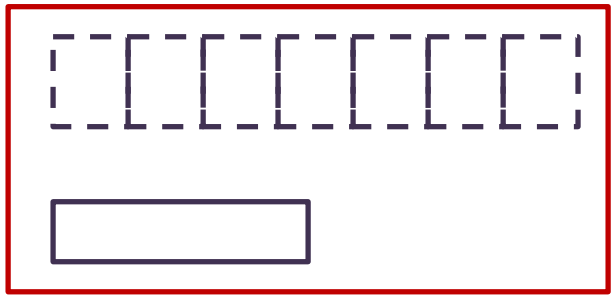
```
struct ora typedef    struct
{
```

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;
```

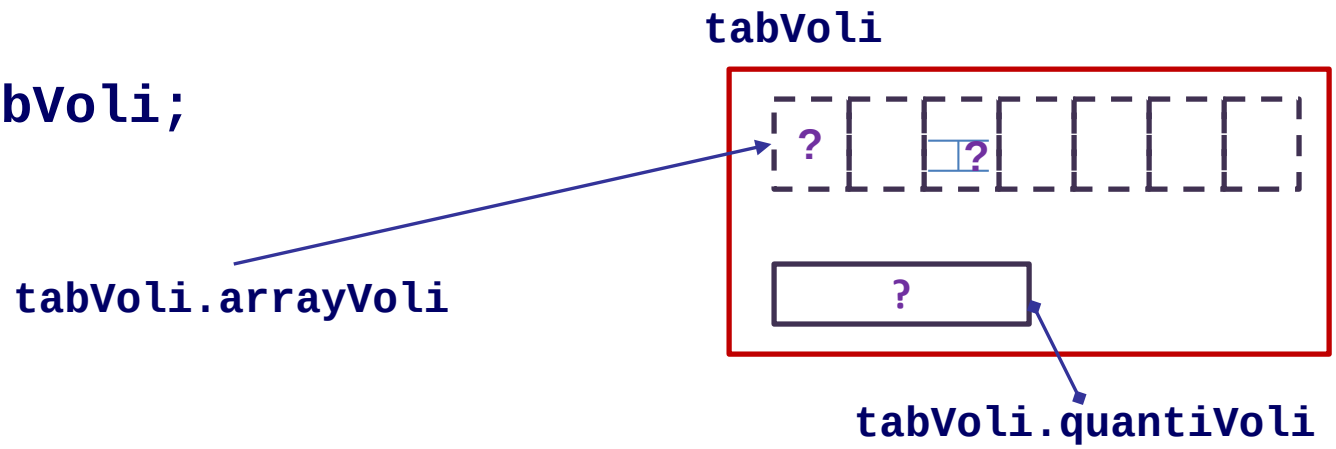


Tabella di Voli Aerei - impl. mediante ARRAY STATICO

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

   tabVoli.quantiVoli=0;   /* inizializzazione */
   do {
     printf(" -      scegli                    -\n");




     printf(" - fine                      (0) -\n");
     scanf("%d", &scelta);
     switch (scelta) {. . . . . .
     } /* fine switch */
  } while (scelta!=0);         /* fine do_while*/     .....
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

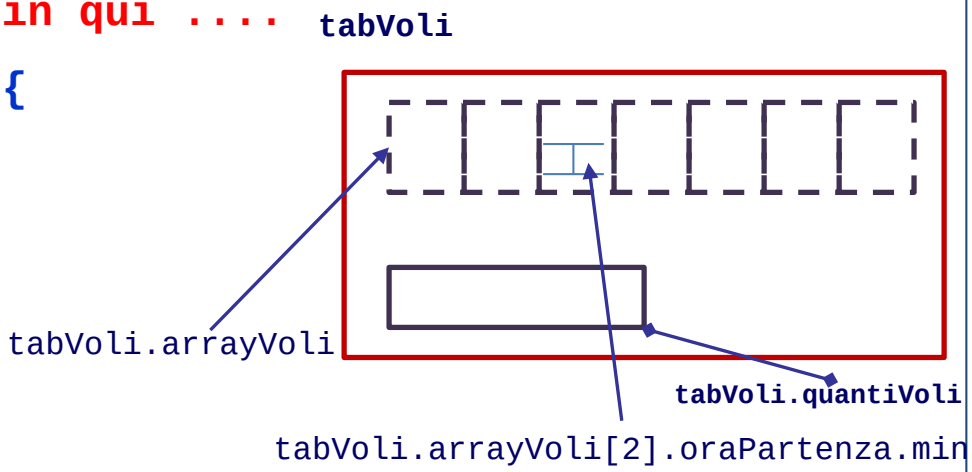
```
int main() {
  TipoTabella tabVoli;
  ....
     switch (scelta) {
       case 1:




       case 2:
          printf(" - codice volo? ");


        break;
       case 3:
          riuscita=aggiungiVolo(              );
.....
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
....
       case 3:
          riuscita=aggiungiVolo(&tabVoli);
          if(!riuscita)

          else



       case 0:

          break;
       default:

     } /* 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) {
```




```
/* 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; }
```

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

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

  if (t->quantiVoli          ) {


    return 0;
  }
  else {
    LETTURA di t->arrayVoli[t->quantiVoli].codice









  /* l'aggiunta ha avuto successo */
    (t->quantiVoli)+=1;
  }                                                    return 1;   }
```

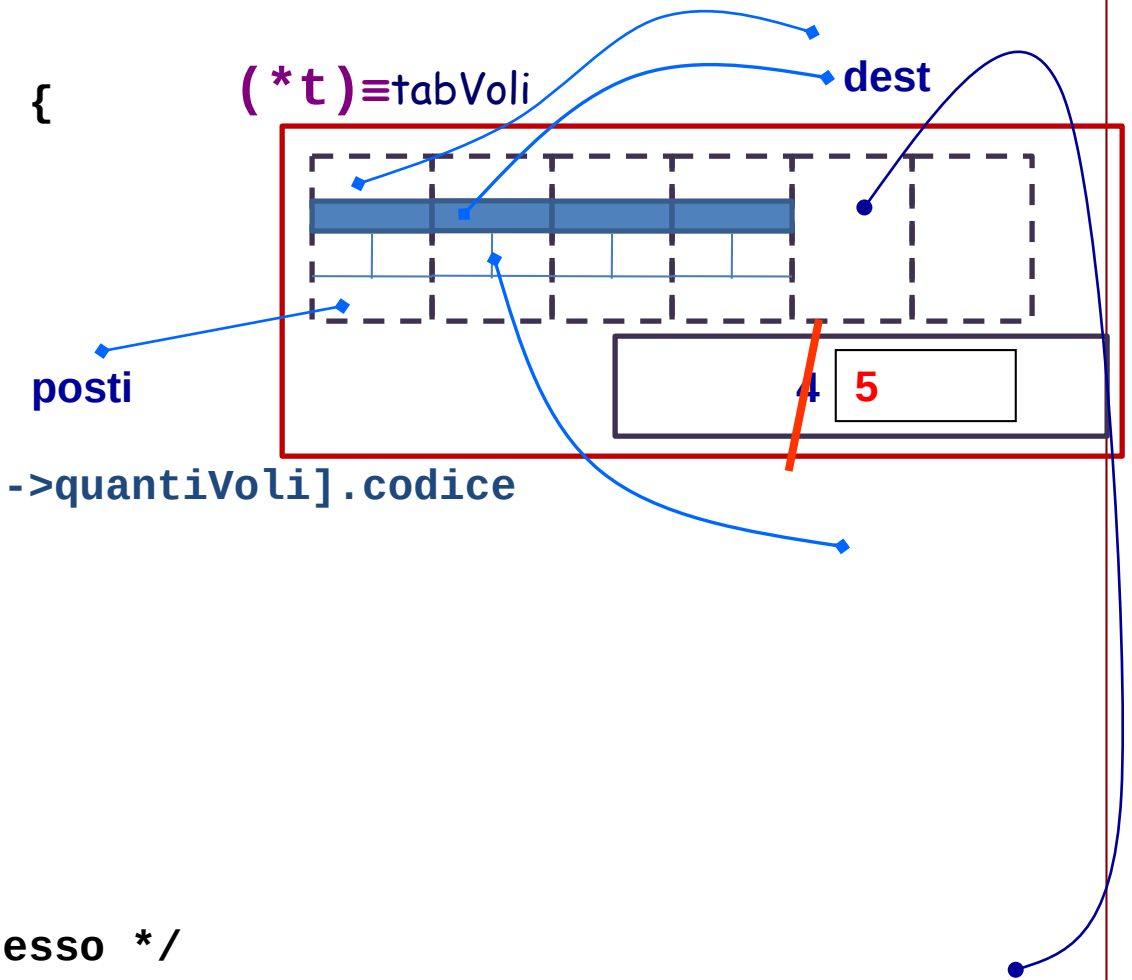


Tabella di Voli Aerei - impl. mediante ARRAY STATICO

```
void stampaQuelVolo (TipoTabella t, char cod[6]);   ☺
```

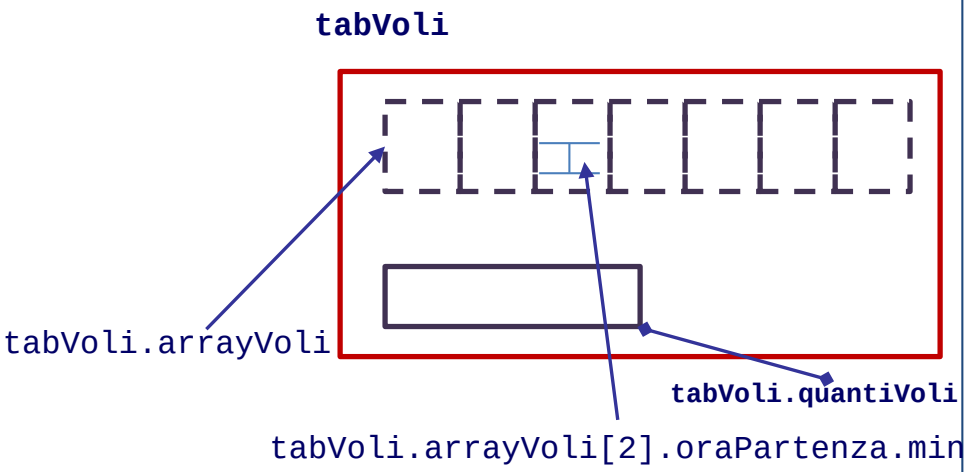
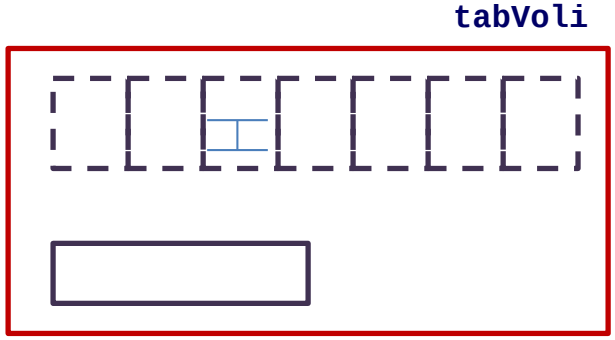


Tabella di Voli Aerei - impl. mediante ARRAY STATICO

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

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



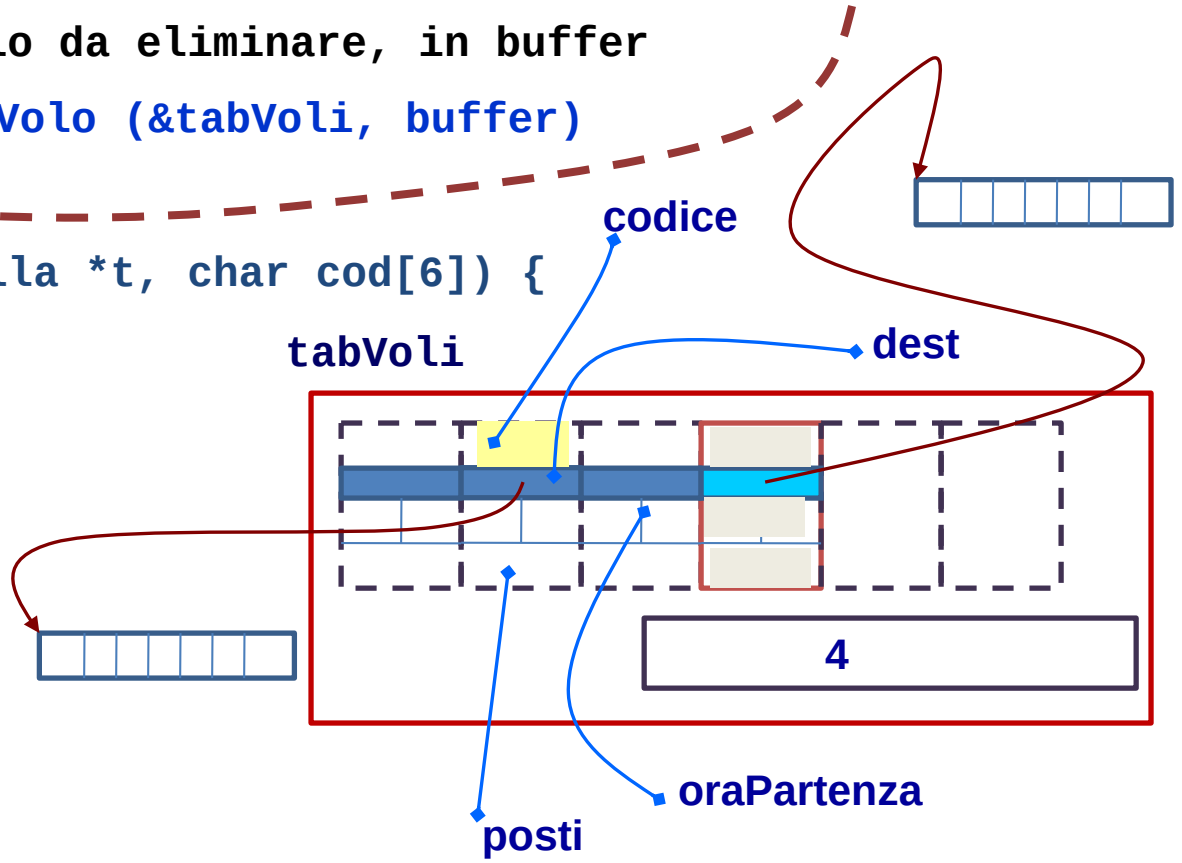
Mentre t.arrayVoli[i] “e` un volo” e "stiamo cercando"

Se e` il volo code trovato!!! esci

Senno` incrementa i

Tabella di Voli Aerei - impl. mediante ARRAY STATICO

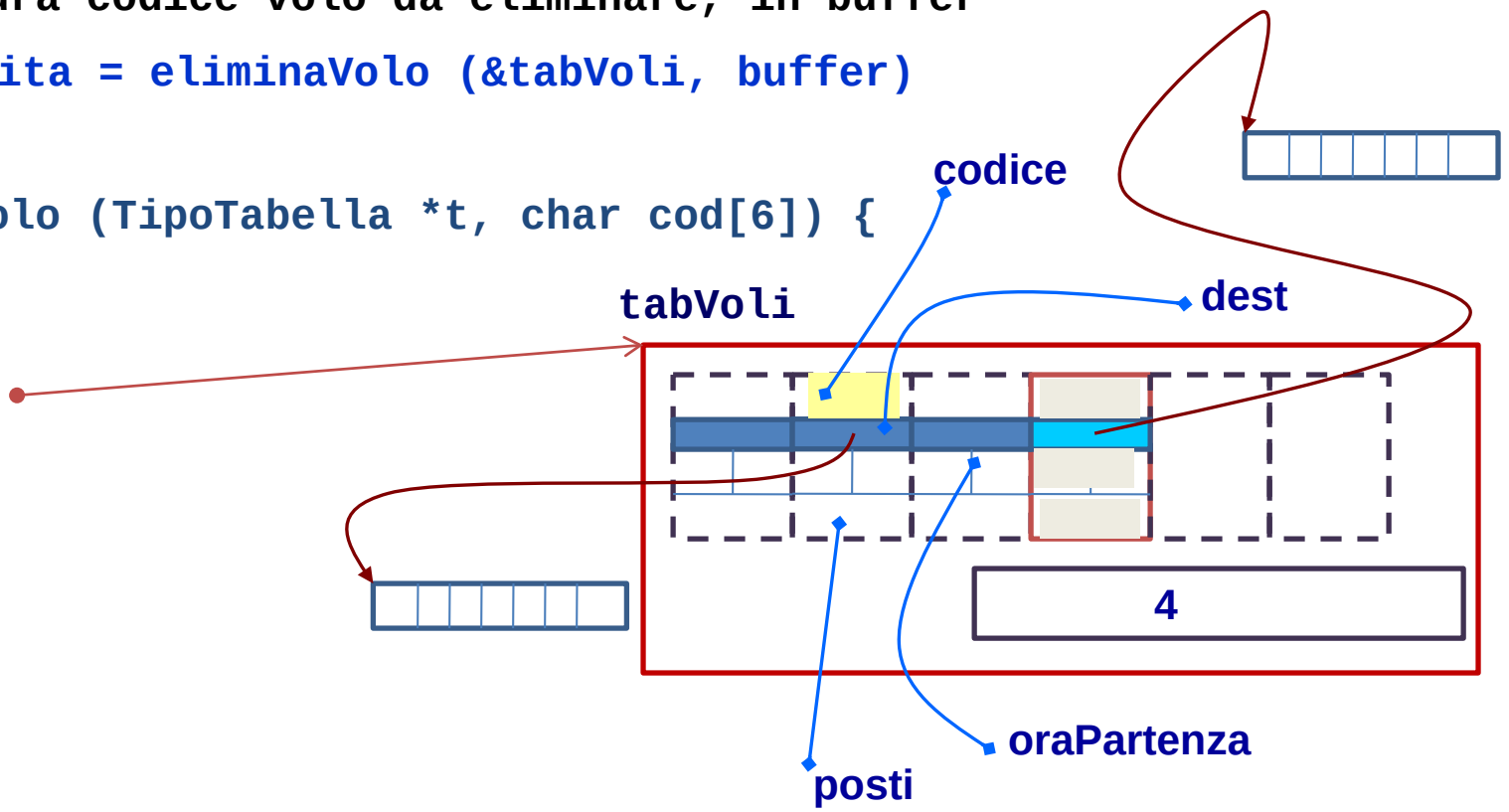
```
case 4: lettura codice volo da eliminare, in buffer
        riuscita = eliminaVolo (&tabVoli, buffer)

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]) {
```



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