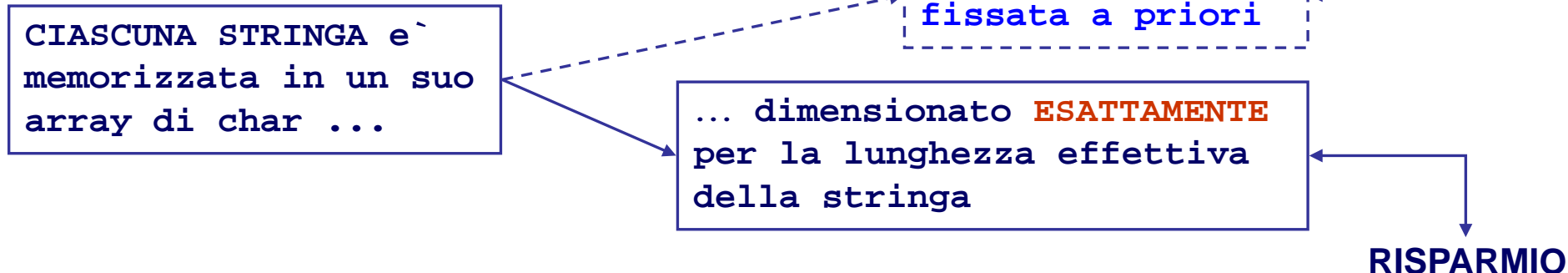


Allocazione Dinamica: Stringhe Esatte

Problema gestione di MOLTE STRINGHE alfanumeriche nel programma; le stringhe possono essere di lunghezza diversa, ma non oltre una lunghezza massima nota



SCHEMA DI REALIZZAZIONE

- vengono definite le diverse stringhe da usare nel programma, come puntatori;

```
char * str, *str2, *str3, *str4; /* 4 stringhe (in potenza) */
```
- viene definita una "stringa buffer" abbastanza grande per contenere qualunque stringa da gestire;

```
char buffer[LUNGMAX+1]
```
- per ogni stringa da memorizzare, prima la si legge usando buffer e poi la si **alloca+riempie/assegna** in corrispondenza di uno dei puntatori, in modo che occupi solo la memoria necessaria

Problema gestione di MOLTE STRINGHE ...

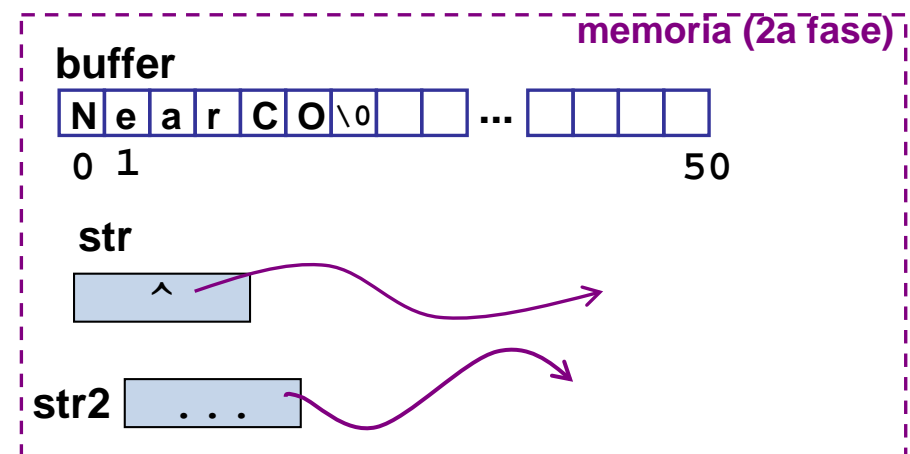
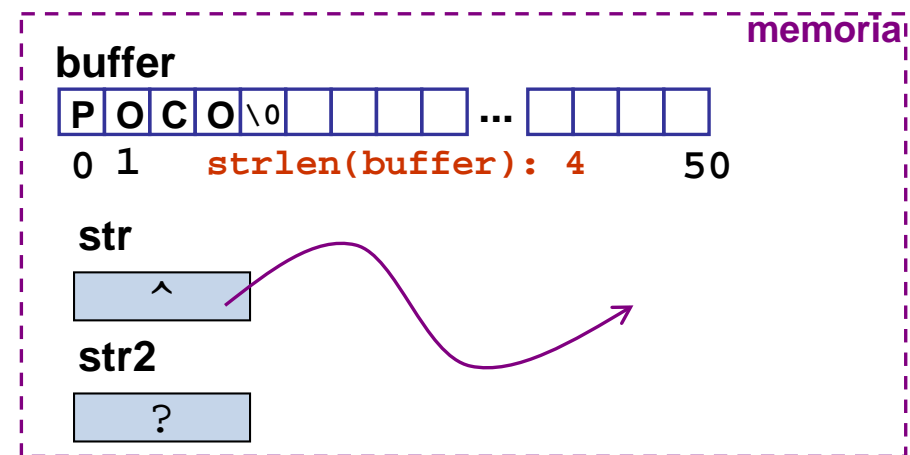
```
#include <stdio.h>
#define LUNGMAX 50 /* stringhe mai piu` lunghe di 50 */
...
char buffer[LUNGMAX+1], *str, *str2 ...
...
scanf(...%s...", buffer);

str=malloc(strlen(buffer)+1);
if (str)

else ... /* messaggio di errore*/
...
scanf(...%s...", buffer);

str2=malloc(
);
if ( )

else ...
```



duplicazione (esatta) di una stringa

esercizio

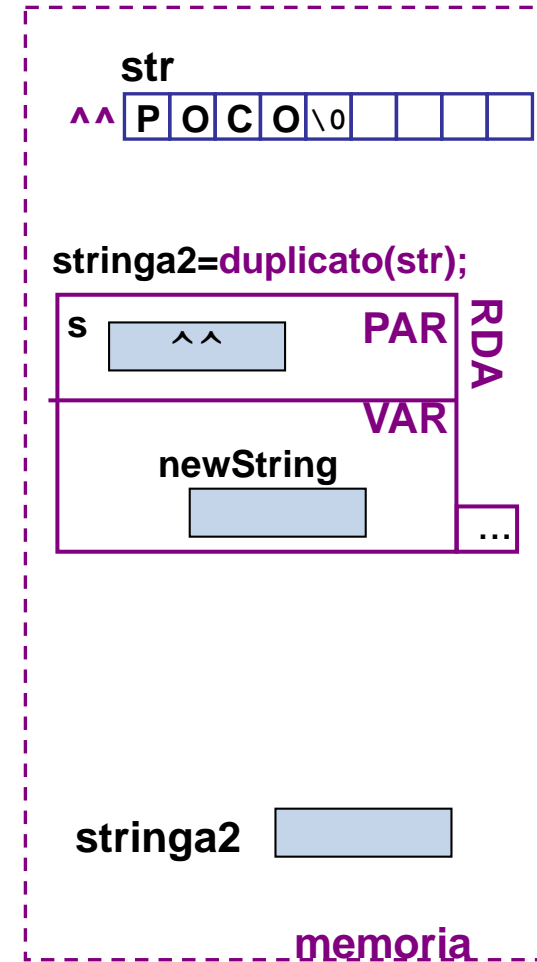
funzione che
ricevendo una stringa s

restituisca

una copia (esatta) di s

```
#include <stdio.h>
#include <stdlib.h>
... (dich.) ...
int main() {
    char str[9], *stringa2;
    .../* "POCO" in stringa2 */
    stringa2 = duplicato(str);
    ...
return 0;
}

char * duplicato (char *s) {
```



Gestione di tante stringhe: Array di stringhe

Array di stringhe

```
char * arrStr[6];
```

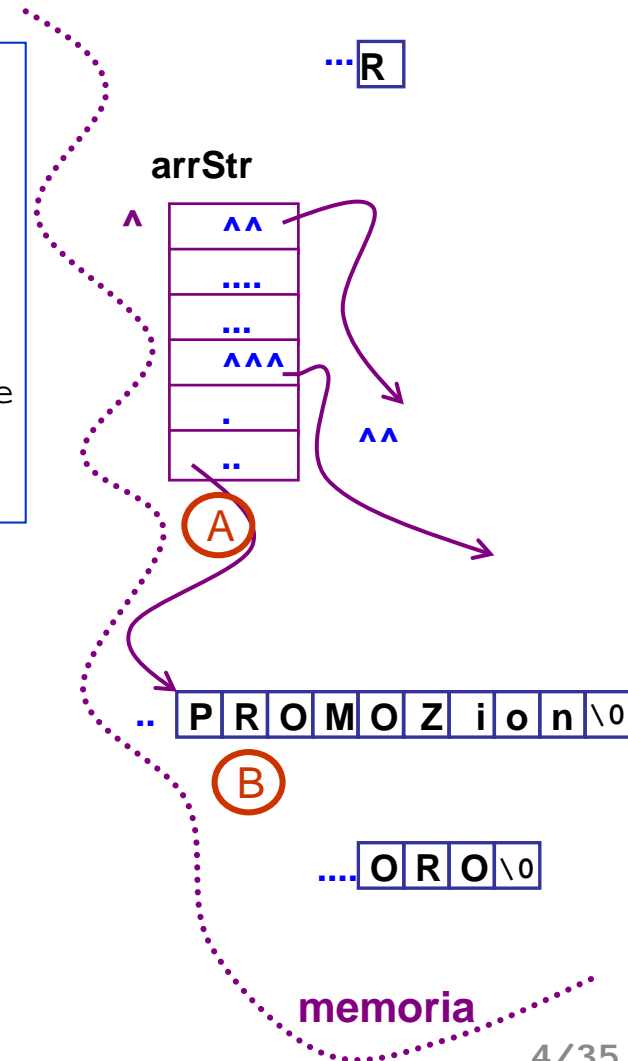
array di puntatori;

ogni elemento punta ad una stringa

(una stringa e` un blocco/array di caratteri)

```
arrStr[5] = malloc(10);      /* allocazione della
                             memoria esattamente
                             necessaria per una delle
                             stringhe (9 char + il '\0') */

if (arrStr[5] == NULL)
    printf("                ");
else
    /* la memoria disponibile viene
       riempita esattamente */
    strcpy(
```



Array di stringhe (lettura) - 1/3 -

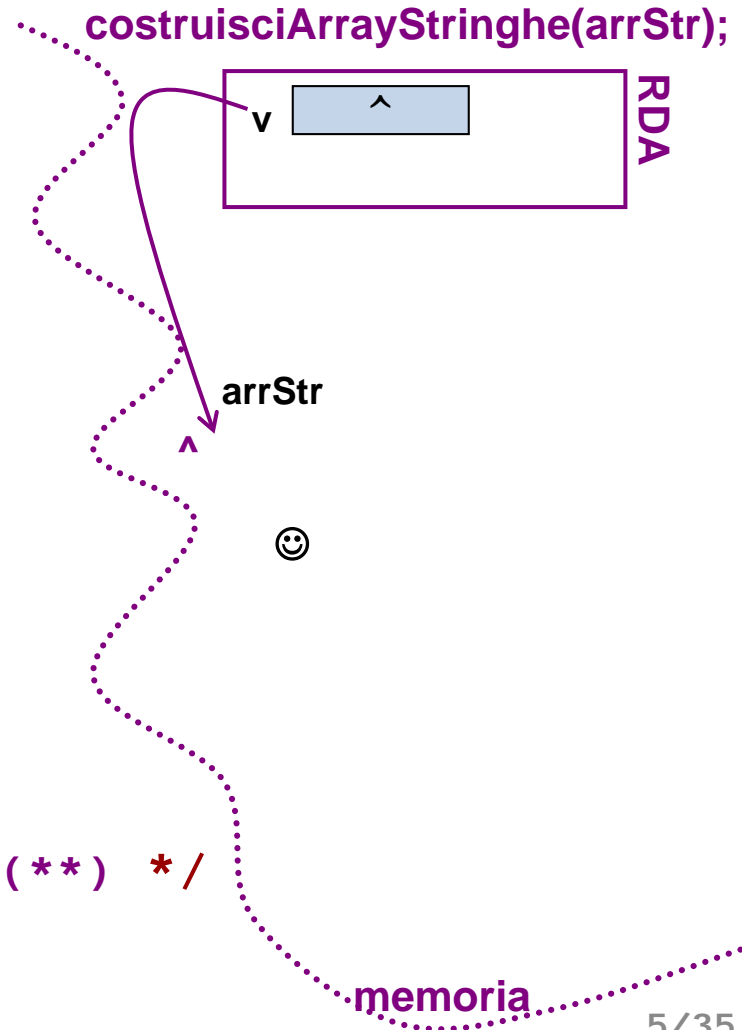
esercizio funzione `costruisciArrayStringhe` che
ricevendo un array di stringhe, `char * v[N]`,
legga **N stringhe**, ciascuna di al più 80 char, e le memorizzi nell'array **(esatte)**

```
/* 1a fase: ambiente di calcolo */
```



```
/* 2a fase: PROTOTIPO (dichiarazione) (**) */
```

```
void costr... ☺
```



Array di stringhe (lettura) - 2/3 -

continua funzione che legge un array di N stringhe, ciascuna di al piu` 80 char

/* 3a fase: definizione funzione */

```
void costruisciArrayStringhe( ☺ ) ;
```

```
char buffer[LUNGMAX+1];
```

```
int i;
```

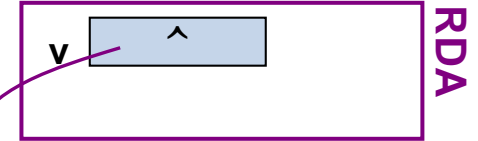
```
for (i=0; i<N; i++) {
```

```
} /* fine for */
```

```
return;
```

```
}
```

costruisciArrayStringhe(arrStr);



arrStr



memoria

Array di stringhe (ricerca) - 1/2 -

esercizio funzione "presentIn" che
ricevendo una stringa **strCercata**, un array di stringhe, **char * v[N]**,
la dimensione di v **dim**
restituisca **1** se **strCercata** e` in **v**, **0** altrimenti

```
/* alg. di ricerca in array, con var. flag */
int presenteIn(
    char *strCercata, char **v, int dim) {

    int trovata, i;
    ...
}
```

memoria

Array di stringhe (ricerca) - 2/2 -

esercizio

funzione che

ricevendo

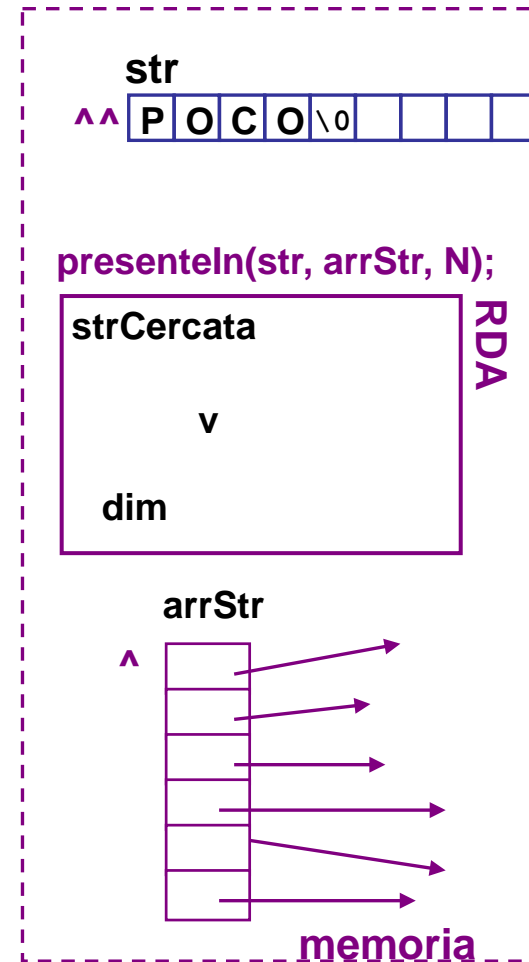
una stringa **strCercata**, un array di stringhe, **char * v[N]**,
la dimensione di v **dim**

restituisca

1 se **strCercata** e` in v, 0 altrimenti

```
/* algoritmo di ricerca in array, con var. flag */
```

```
int presenteIn(  
    char *strCercata, char **v, int dim) {  
  
    int trovata, i;  
  
    for (i=0; (i<dim); i++)  
        if (strcmp(strCercata, v[i])==0)  
            trovata=1;  
  
    return;  
}
```



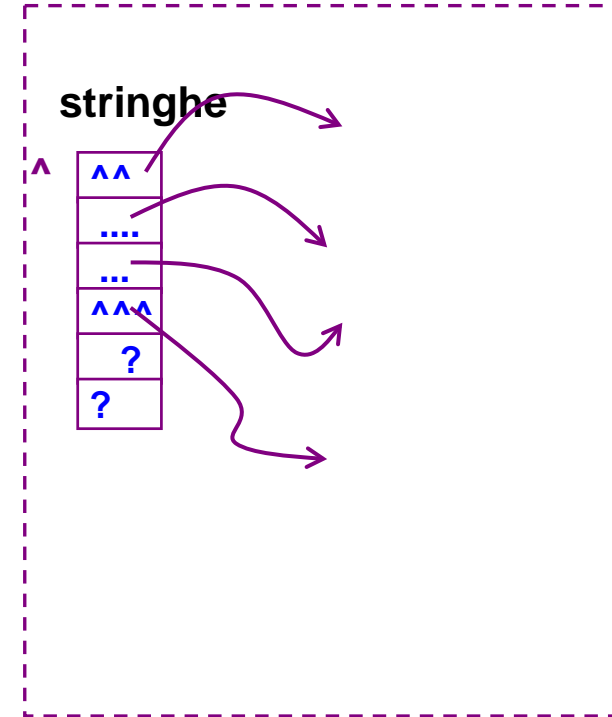
Programma gestione stringhe - introduzione

gestione di un array di al piu` N stringhe, ciascuna di al +
LUNGMAX caratteri
(array usato parzialmente)

Funzionalita` per la gestione di una

- **aggiunta** di una stringa (se possibile)
- **stampa** delle stringhe contenute
- **ricerca** di una stringa e rest. del suo indice (opp. -1)
(*funzione di servizio*)
- **sostituzione** di una stringa con un'altra data

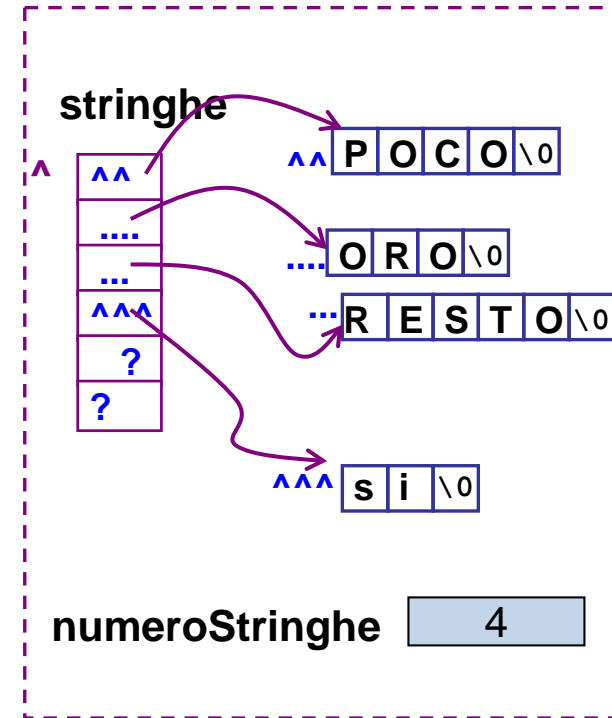
COLLEZIONE di stringhe:



Programma gestione stringhe - introduzione

gestione di un array di al piu` N stringhe, ciascuna di al + LUNGMAX caratteri (array usato parzialmente)

Struttura dati e Funzionalita` per la gestione del TIPO *COLLEZIONE di stringhe*



in sostanza una collezione di stringhe va rappresentata mediante la collaborazione di due variabili: **stringhe** e **numeroStringhe**

- un array di stringhe, che faccia da sostegno per la memorizzazione delle stringhe;
- una variabile intera che dica in ogni momento quante stringhe ci sono nell'array

collezione di stringhe = <array + numerostringhe>

Gestione di tabella (collezione) di stringhe - 1/8 -

SCHEMA DI PROGRAMMA

```
#include ...
#define N ...
... (dich.) ...
int main() {
    scelta      -----(per il menu` ...)
    stringhe, numeroStringhe, -----(per la collezione di stringhe)
    buffer1, buffer2, -----(buffer per leggere stringhe)
do {
    /* ciclo di stampa menu`, lettura scelta funzionalita` da
    eseguire, esecuzione della funzionalita` prescelta */

    ... aggiungi(stringhe, buffer1, &numeroStringhe);    (scelta==1)

    ... stampaTutto(stringhe, numeroStringhe);           (scelta==3)

    ... sostituisci(stringhe, numeroStringhe, buffer1, buffer2);
                                                    (scelta==2)
    ...

} while (scelta!=0)

return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>          #define N ...          #define LUNGMAX
...      (dich.)
int main() { char *stringhe[N], char buffer1[LUNGMAX+1],
            buffer2[LUNGMAX+1];
            int numeroStringhe, scelta;
numeroStringhe = 0; /* init struttura dati array stringhe */
do { stampaMenu(); /* 1=aggiungi 2=sostitui... */
    scanf("%d", &scelta); /* lettura scelta */
    switch(scelta) {
    case 1:
        break;

    case 2:
        break;
    case 3: ...
    }
```

```
#include <stdio.h>
#include <stdlib.h>
... (dich.)
#define N ...
#define LUNGMAX

int main() {
    char *stringhe[N], char buffer1[LUNGMAX+1], buffer2[LUNGMAX+1];
    int numeroStringhe, scelta;

    numeroStringhe = 0; /* init struttura dati array stringhe */

    do {
        stampaMenu();
        scanf("", &scelta);

        switch(scelta) {
            case 1:
                aggiungi(stringhe, buffer1, &numeroStringhe);
            }
            else printf("spazio insufficiente, tsk.\n\n");
            break;
        }
    }
}
```

perche'?



```
#include <stdio.h>
#include <stdlib.h>
... (dich.)
#define N ...
#define LUNGMAX

int main() {
    char *stringhe[N], char buffer1[LUNGMAX+1], buffer2[LUNGMAX+1];
    int numeroStringhe, scelta;

    numeroStringhe = 0; /* init struttura dati array stringhe */

    do {
        stampaMenu();
        scanf("", &scelta);

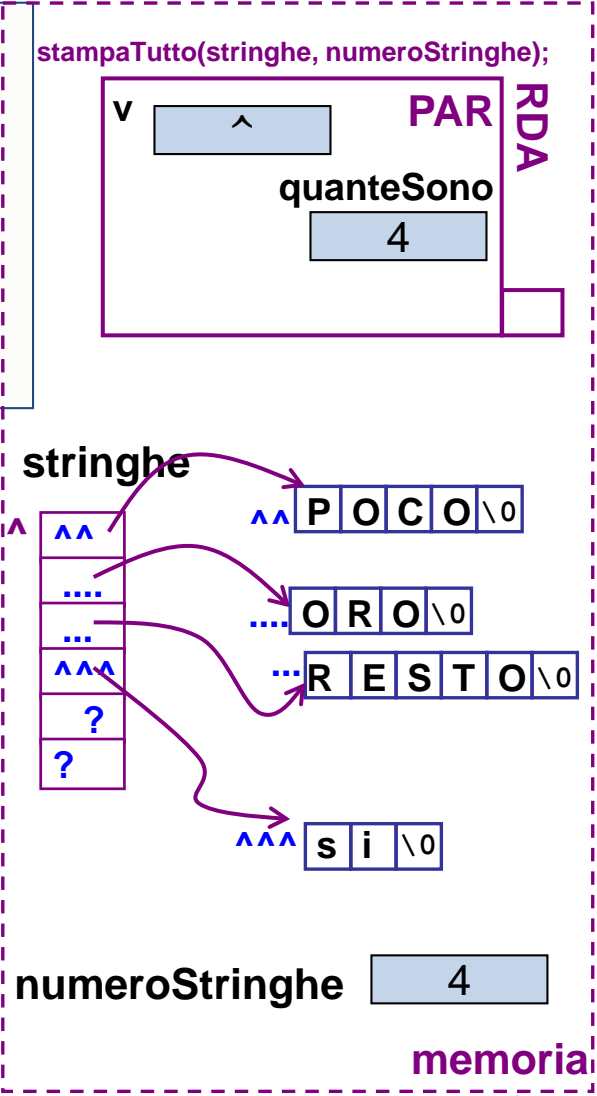
        switch(scelta) {
            case 1:
                if (numeroStringhe<N) {
                    printf("quale stringa da aggiungere? ");
                    scanf("%s", buffer1);
                    aggiungi(stringhe, buffer1, &numeroStringhe);
                }
                else printf("spazio insufficiente, tsk.\n\n");
                break;

            case 2:
                sostituisci(stringhe, numeroStringhe, buffer1, buffer2);
                break;

            ...
        }
    } while (scelta != 0);
}
```

```
...
case 3: stampaTutto(stringhe, numeroStringhe);
       break; ...
```

```
void stampaTutto(char *v[], int quanteSono) {
    int i;
    for (i=0; i<quanteSono; i++)
        printf("%s\n", v[i]);
    return; /* o anche *(v+i) */
}
```

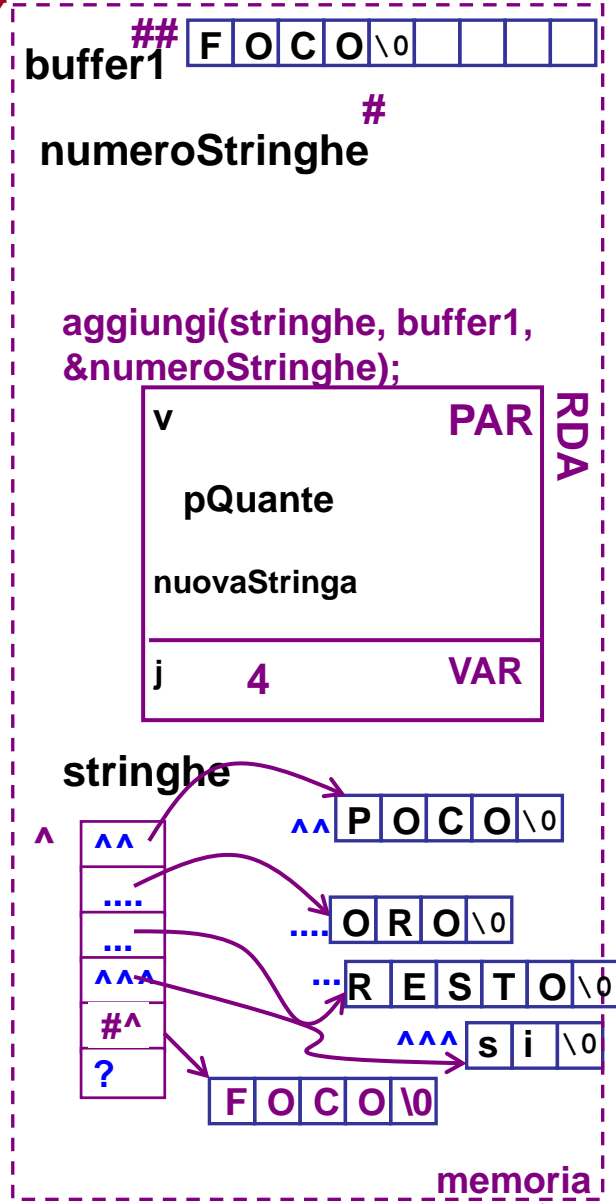


```
void aggiungi ( char **v, char *nuovaStringa, int *pQuante) {
/* il controllo sulla disponibilita` di spazio nell'array
si suppone fatto all'esterno, dalla funzione chiamante
(non e` bello, ma ora ci stiamo concentrando su altro */

    int j = *pQuante; /* solo per comodita` */

    v[j] = malloc (strlen(nuovaStringa)+1);

    if (!v[j])
        printf("
    else {
        strcpy(
        *pQuante
```



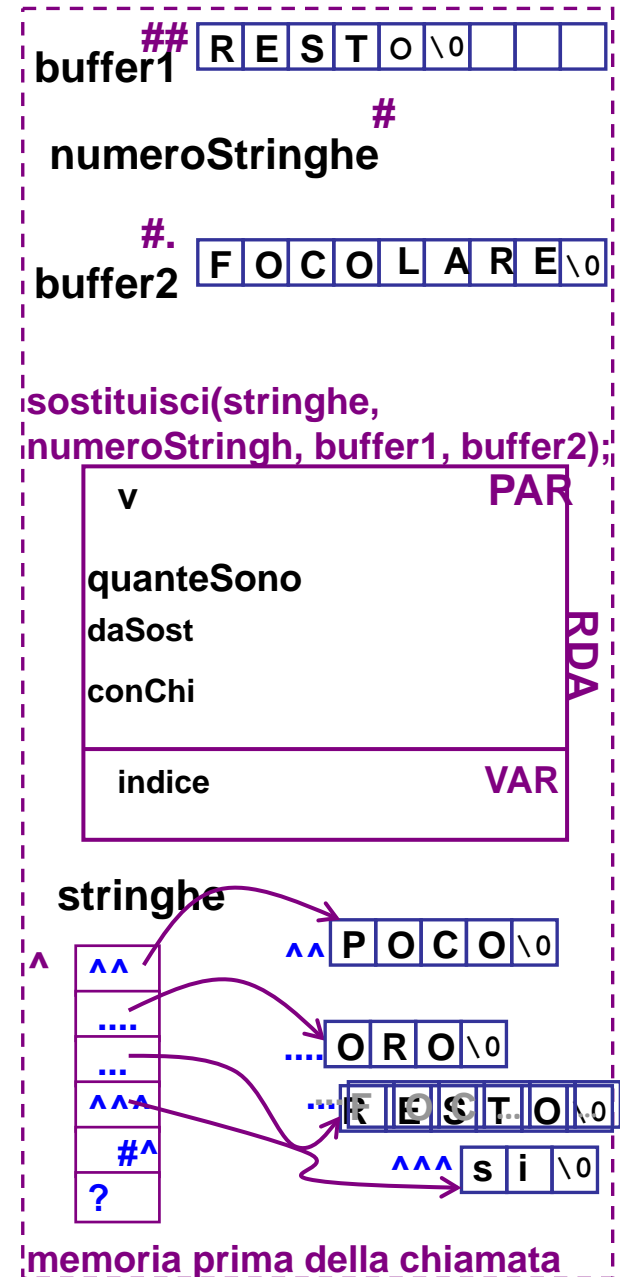

```
void sostituisci (char **v, int quanteSono,
                 char *daSost, char *conChi) {
    /* cerchiamo l'indice della stringa da sostituire con
       una funzione di servizio che restituisce l'indice
       della stringa nell'array, oppure -1 (se non c'e`)*/
    int indice =
        ricerca(v,quanteSono, daSost);

    if (indice== -1)
        printf("non presente\n\n");
    else {

        v[indice]=malloc(strlen(conChi)+1);

        if (
            )

        else
```

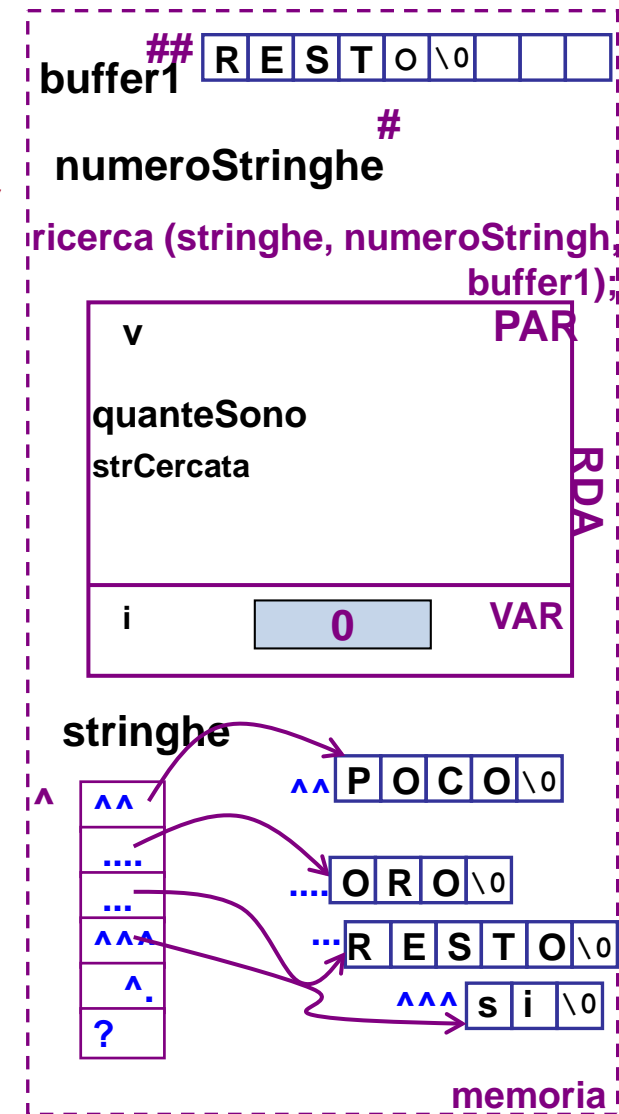


```

int ricerca (char **v, int quanteSono, char *strCercata) {
/* restituisce l'indice di strCercata in v, oppure -1 */
    int i = 0,

    for ( ; i<quanteSono; i++)
        if (strcmp((v[i], strCercata)==0)
            return i; /* stringa trovata: rest. l'indice */

return -1;
}
    
```



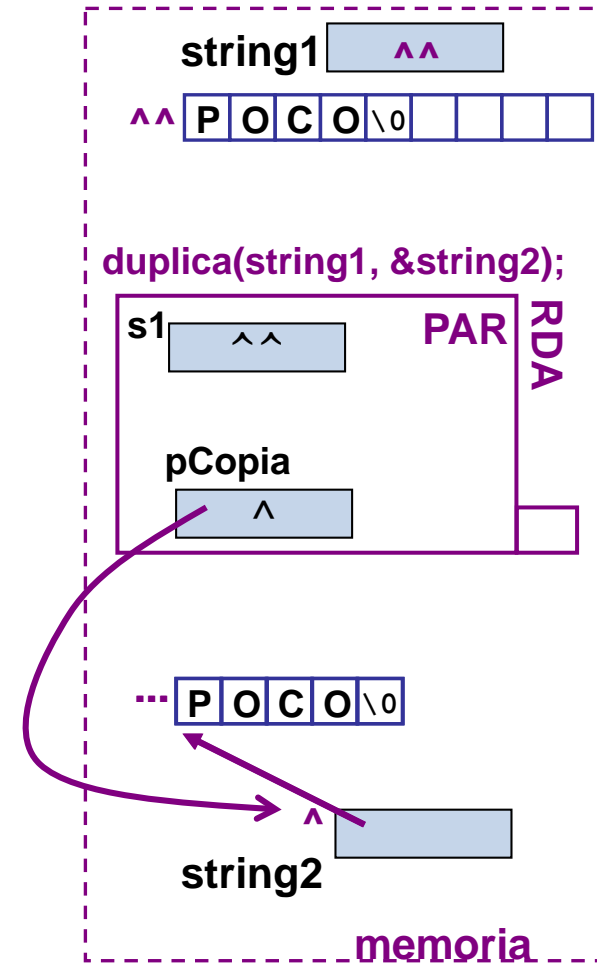
Esercizio (duplica stringa)

```
int main() { char *string1, *string2;
    ...

    ...
    duplica(string1, &string2);
    ...
    return 0;
}

void duplica(char * s1, char **pCopia) {
    *
    = malloc(strlen(s1)+1);

return;
}
```



```
int duplica2(char * s1, char **pCopia) {  
  
    *pCopia = malloc(strlen(s1)+1);  
  
    if (*pCopia) {  
  
        strcpy(*pCopia, s1);  
  
        return 1;          /* e` andata bene */  
  
    } else  
  
        return 0;        /* e` andata male */  
  
}
```

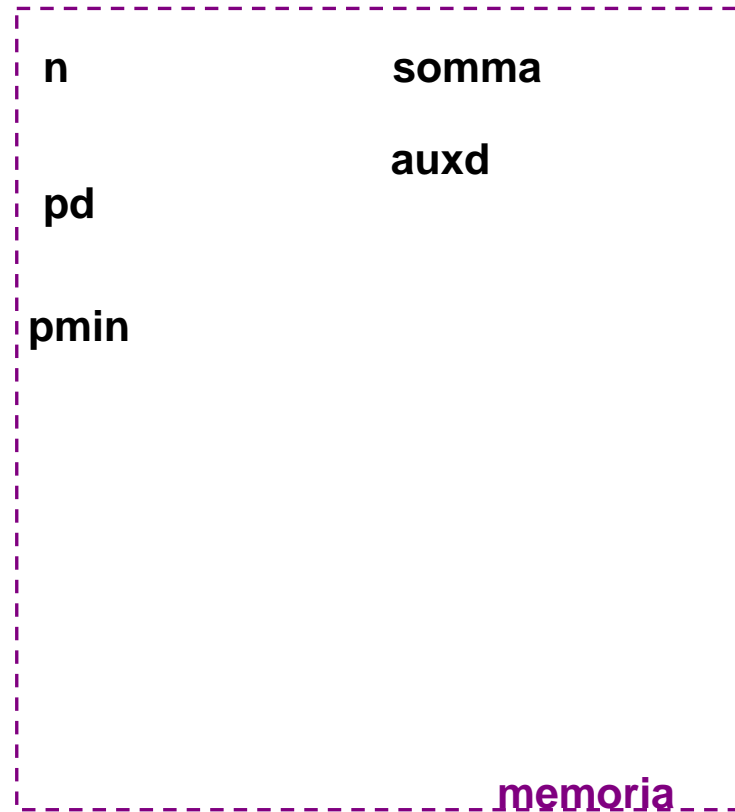
Esercizio

programma che

legge un intero n e poi legge n double;
memorizza i double in un array dinamico esatto,
calcola e stampa minimo, massimo e media dei double

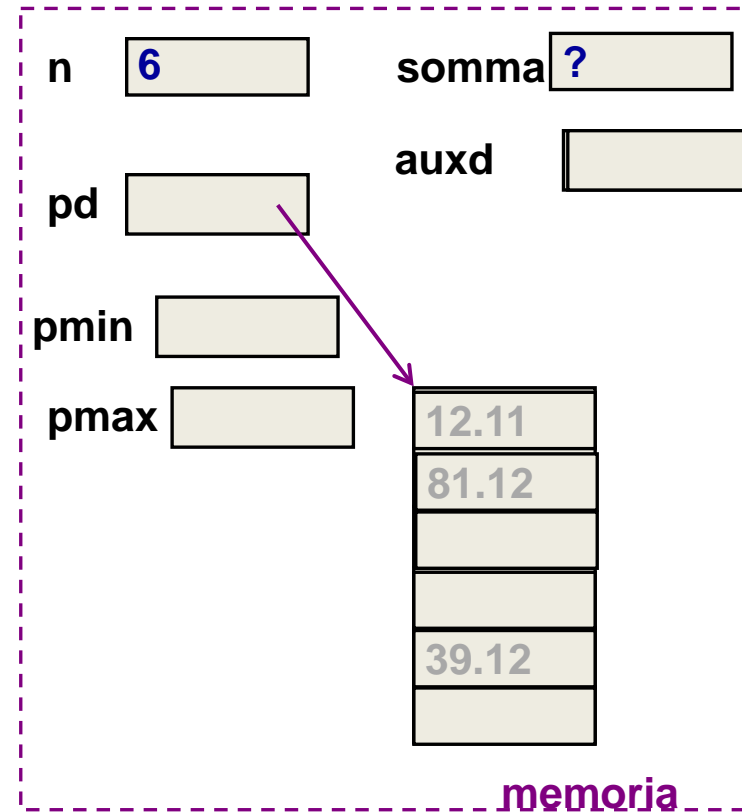
- 1) Allocazione array dinamico, lettura e memorizzazione dei numeri negli elementi $*pd$ $*(pd+n-1)$
- 2) init minimo e massimo parziale, e somma
- 3) scansione a ritroso da "**penultimo**" a "primo" elemento, usando l'algoritmo del massimo (minimo) parziale
e accumulando i double (per poter calcolare la media)
- 4) e poi calcolo media e stampa di min, max e media

La scansione viene realizzata mediante un
puntatore: `auxd`



Esercizio

- 1) Allocazione array dinamico, lettura e memorizzazione dei numeri negli elementi $*pd$ $*(pd+n-1)$
- 2) init minimo e massimo parziale, e somma

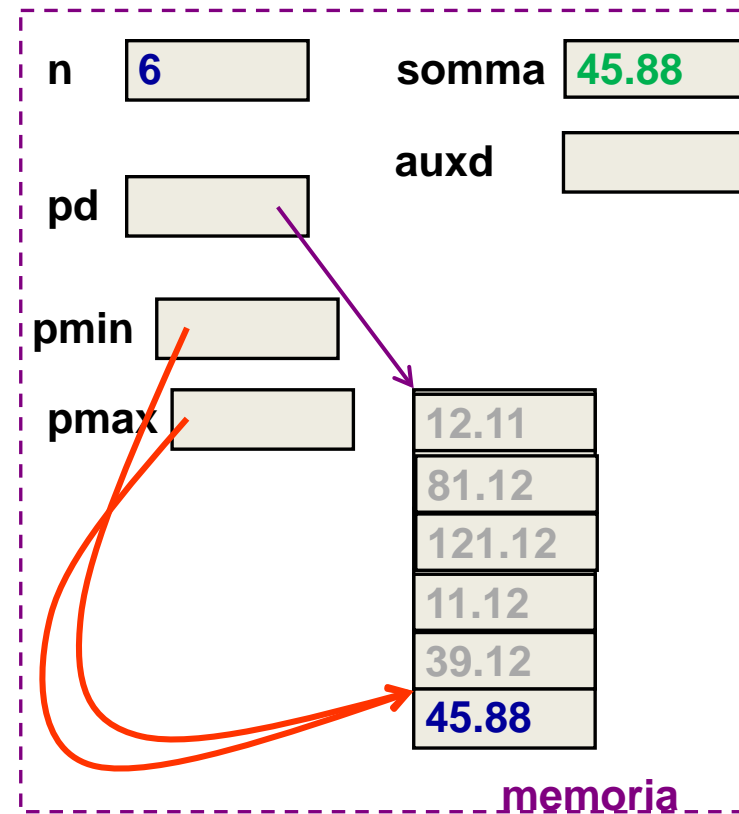


Esercizio (o esempio?)

programma che

legge un intero n e n double;
li memorizza in un array dinamico esatto
calcola e stampa minimo, massimo e media dei double

- 1) Allocazione array dinamico, lettura e memorizzazione dei numeri in $*pd$ $*(pd+n-1)$
- 2) init minimo e massimo parziale, e somma
- 3) scansione a ritroso da "penultimo" a "primo" elemento, trovando max e min, e accumulando

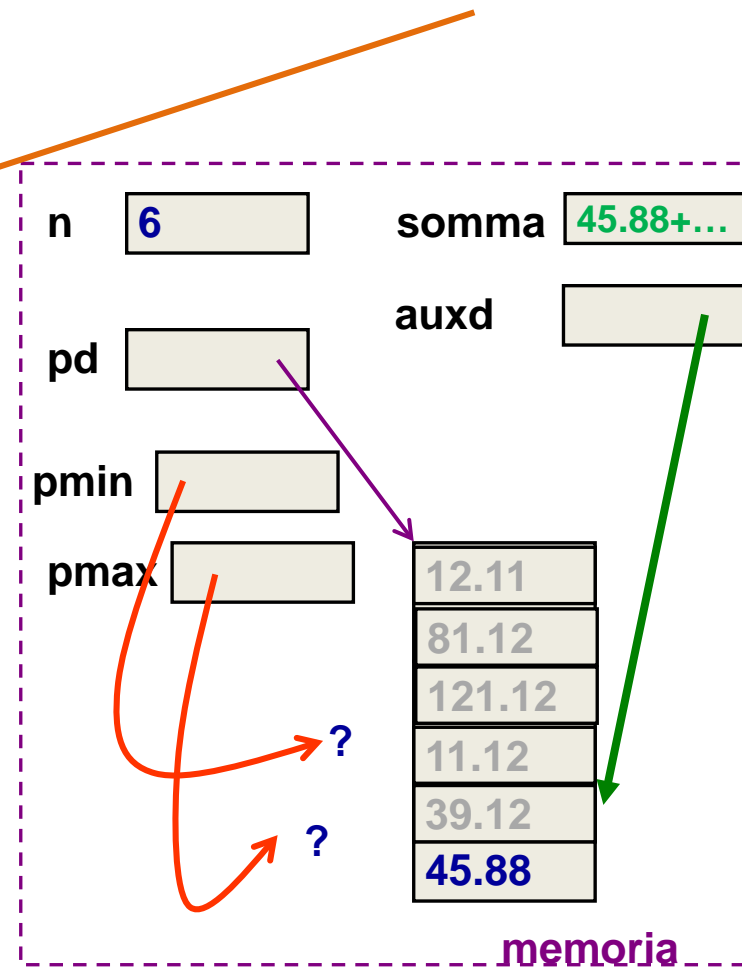


Esercizio

programma che

legge un intero n e n double;
li memorizza in un array dinamico esatto
calcola e stampa minimo, massimo e media dei double

- 1) Allocazione array dinamico, lettura e memorizzazione dei numeri in `*pd *(pd+n-1)`
- 2) init minimo e massimo parziale, e somma
- 3) scansione a ritroso da "penultimo" a "primo" elemento, trovando max e min, e accumulando



esercizio su intero n e n double (coding 1/2)

programma che

legge un intero n e n double;

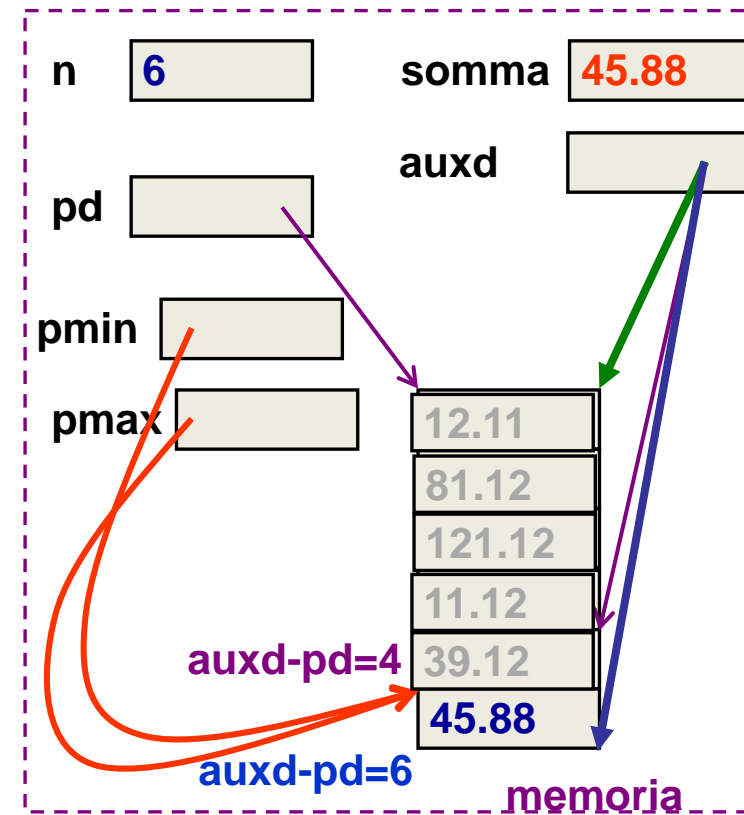
li memorizza in un array dinamico esatto
calcola e stampa minimo, massimo e media dei double

```
#include <stdio.h>
#include <stdlib.h>
int main() {
...

scanf( ... &n);
pd = malloc(n*sizeof(double));

if (!pd)    printf(" ... ");
else {
    for (auxd=pd; auxd-pd < n; auxd++)

        pmax = pmin = --auxd;
        somma = *auxd;
...
}
```



esercizio su intero n e n double (coding 2/2)

```
... pmax = pmin = --auxd;  
somma = *auxd;  
for (auxd--; auxd >= pd; auxd--) {  
    if (*pmax < *auxd)  
        pmax=auxd;  
  
    if (*pmin > *auxd)  
        pmin=auxd;  
  
    somma += *auxd;  
}
```

```
media = somma/n;  
printf( ..., *pmax, *pmin, media);  
return 0;  
}
```

