

Tecniche della Programmazione, lez.09

Funzioni e array

File Testuali – textfile (Online)

Un problema problematico

Una società commerciale possiede 5 negozi, ciascuno dei quali produce un guadagno mensile.

Il nostro software deve permettere, ogni mese, di

- ricevere i dati sui guadagni
- stampare la media dei guadagni
- evidenziare quali negozi hanno guadagno $< 1/3$ della media

INPUT

```
caro/a manager, dammi il guadagno del negozio 1: 129
caro/a manager, dammi il guadagno del negozio 2: 333
caro/a manager, dammi il guadagno del negozio 3: 989
caro/a manager, dammi il guadagno del negozio 4: 651
caro/a manager, dammi il guadagno del negozio 5: 68
```

OUTPUT

```
caro/a manager, la media e' 434, e i negozi da controllare sono:
Negozio 1, con guadagno 129
Negozio 5, con guadagno 68
FINE
```

primo livello ...

Una società commerciale possiede 5 negozi, ciascuno dei quali produce un guadagno mensile.

Il nostro software deve permettere, ogni mese, di

- ricevere i dati sui guadagni
- stampare la media dei guadagni
- evidenziare quali negozi hanno guadagno $< 1/3$ della media

se leggo una sequenza di numeri posso calcolare la media (mediante accumulazione e conteggio dei numeri, e divisione della somma per il numero dei numeri) ...

Il problema problematico

Una società commerciale possiede 5 negozi, ciascuno dei quali produce un guadagno mensile.

Il nostro software deve permettere, ogni mese, di

- ricevere i dati sui guadagni
- stampare la media dei guadagni
- evidenziare quali negozi hanno guadagno $< 1/3$ della media

se leggo una sequenza di numeri posso calcolare la media (mediante accumulazione e conteggio dei numeri, e divisione della somma per il numero dei numeri) ...

ma per dire quali negozi sono scarsi, devo ricordarli, e ricordare, quanto hanno guadagnato



Negozi

(affrontiamo il problema ... poi cercheremo una soluzione meno deprimente ...)

Algoritmo

```
0) var guad1, guad2, guad3, guad4, guad5
1) leggere guad1
2) leggere guad2
3) leggere guad3
4) leggere guad4
5) leggere guad5
6) media = la somma dei guad* / 5
7) se guad1 < (media * 1/3) stampare "Negozio 1 con " guad1
8) se guad2 < (media * 1/3) stampare "Negozio 2 con " guad2
9) se guad3 < (media * 1/3) stampare "Negozio 3 con " guad3
10) se guad4 < (media * 1/3) stampare "Negozio 4 con " guad4
11) se guad5 < (media * 1/3) stampare "Negozio 5 con " guad5
12) FINE
```

Negozi - programma

```
#include <stdio.h>
#define N 5          /* non che serva a molto, qui, definire una costante ... */
int main () {
    double guad1, guad2, guad3, guad4, guad5;      /* ... */
    double media;

...
    printf ("... dammi il guadagno del negozio 1: ");
    scanf("%lf", &guad1);
    printf ("... dammi il guadagno del negozio 2: ");
    scanf("%lf", &guad2);
    printf ("... dammi il guadagno del negozio 3: ");
    scanf("%lf", &guad3);
    printf ("... dammi il guadagno del negozio 4: ");
    scanf("%lf", &guad4);
    printf ("... dammi il guadagno del negozio 5: ");
    scanf("%lf", &guad5);

    media = (guad1+guad2+guad3+guad4+guad5)/N;

    printf ("\nOUTPUT\n-----\n");
}
```

Negozi - programma

```
#include <stdio.h>
#define N 5
int main () {
    double guad1, guad2, guad3, guad4, guad5;    /* ... */
    double media;
    printf ("... dammi il guadagno del negozio 1: ");
    scanf("%lf", &guad1);
    ...
    printf ("... dammi il guadagno del negozio 5: ");
    scanf("%lf", &guad5);
    media = (guad1+guad2+guad3+guad4+guad5)/N;
    printf ("\nOUTPUT\n-----\n");
    printf ("...la media è %g, e i negozi ...sono:\n", media);

    if (guad1 < media*1/3)
        printf ("Negozio 1, con guadagno %g\n", guad1);
    if (guad2 < media*1/3)
        printf ("Negozio 2, con guadagno %g\n", guad2);
    if (guad3 < media*1/3)
        printf ("Negozio 3, con guadagno %g\n", guad3);
    if (guad4 < media*1/3)
        printf ("Negozio 4, con guadagno %g\n", guad4);
    if (guad5 < media*1/3)
        printf ("Negozio 5, con guadagno %g\n", guad5);

    printf ("FINE\n");
}
```

Negozi - programma

```
#include <stdio.h>
#define N 5
int main () {
    double guad1, guad2, guad3, guad4, guad5;    /* ... */
    double media;
    printf ("... dammi il guadagno del negozio 1: ");
    scanf("%lf", &guad1);
```

...

P
S
m

INPUT

```
-----
caro/a manager, dammi il guadagno del negozio 1: 129
caro/a manager, dammi il guadagno del negozio 2: 333
caro/a manager, dammi il guadagno del negozio 3: 989
caro/a manager, dammi il guadagno del negozio 4: 651
caro/a manager, dammi il guadagno del negozio 5: 68
```

OUTPUT

```
-----
caro/a manager, la media e' 434, e i negozi da controllare sono:
Negozio 1, con guadagno 129
Negozio 5, con guadagno 68
FINE
```

```
    printf ("Negozio 3, con guadagno %g\n", guad3);
if (guad4<media*1/3)
    printf ("Negozio 4, con guadagno %g\n", guad4);
if (guad5<media*1/3)
    printf ("Negozio 5, con guadagno %g\n", guad5);
```

```
printf ("FINE\n");
```

Tecniche della Programmazione, M.Temperini - lezione 09

che lavoro da cani... Potrebbe andare peggio ...



Potrebbe essere "20 Negozi"

```
#include <stdio.h>
#define N 20

int main () {
    double guad1, guad2, guad3, guad4, guad5;      /* i guadagni dei negozi*/
    double guad6, guad7, guad8, guad9, guad10;     /* i guadagni dei negozi*/
    double guad11, guad12, guad13, guad14, guad15; /* i guadagni dei negozi*/
    double guad16, guad17, guad18, guad19, guad20; /* i guadagni dei negozi*/
    double media;

    printf ("INPUT\n-----\n");

    printf ("caro/a manager, dammi il guadagno del negozio 1: ");
    scanf("%lf", &guad1);
    printf ("caro/a manager, dammi il guadagno del negozio 2: ");
    scanf("%lf", &guad2);
    printf ("caro/a manager, dammi il guadagno del negozio 3: ");
    scanf("%lf", &guad3);
    printf ("caro/a manager, dammi il guadagno del negozio 4: ");
    scanf("%lf", &guad4);
    printf ("caro/a manager, dammi il guadagno del negozio 5: ");
    scanf("%lf", &guad5);
    printf ("caro/a manager, dammi il guadagno del negozio 6: ");
    scanf("%lf", &guad6);
    printf ("caro/a manager, dammi il guadagno del negozio 7: ");
    scanf("%lf", &guad7);
    printf ("caro/a manager, dammi il guadagno del negozio 8: ");
    scanf("%lf", &guad8);
    printf ("caro/a manager, dammi il guadagno del negozio 9: ");
    scanf("%lf", &guad9);
    printf ("caro/a manager, dammi il guadagno del negozio 10: ");
    scanf("%lf", &guad10);
    printf ("caro/a manager, dammi il guadagno del negozio 11: ");
    scanf("%lf", &guad11);
    printf ("caro/a manager, dammi il guadagno del negozio 12: ");
    scanf("%lf", &guad12);
    printf ("caro/a manager, dammi il guadagno del negozio 13: ");
    scanf("%lf", &guad13);
    printf ("caro/a manager, dammi il guadagno del negozio 14: ");
```

Potrebbe essere "20 Negozi"

```
#include <stdio.h>
#define N 20

int main () {
    double guad1, guad2, guad3, guad4, guad5,
    double guad6, guad7, guad8, guad9, guad10,
    double guad11, guad12, guad13, guad14, guad15,
    double guad16, guad17, guad18, guad19, guad20,
    double media;

    printf ("INPUT\n-----\n");

    printf ("caro/a manager, dammi il guadagno del negozio 1: ");
    scanf("%lf", &guad1);
    printf ("caro/a manager, dammi il guadagno del negozio 2: ");
    scanf("%lf", &guad2);
    printf ("caro/a manager, dammi il guadagno del negozio 3: ");
    scanf("%lf", &guad3);
    printf ("caro/a manager, dammi il guadagno del negozio 4: ");
    scanf("%lf", &guad4);
    printf ("caro/a manager, dammi il guadagno del negozio 5: ");
    scanf("%lf", &guad5);
    printf ("caro/a manager, dammi il guadagno del negozio 6: ");
    scanf("%lf", &guad6);
    printf ("caro/a manager, dammi il guadagno del negozio 7: ");
    scanf("%lf", &guad7);
    printf ("caro/a manager, dammi il guadagno del negozio 8: ");
    scanf("%lf", &guad8);
    printf ("caro/a manager, dammi il guadagno del negozio 9: ");
    scanf("%lf", &guad9);
    printf ("caro/a manager, dammi il guadagno del negozio 10: ");
    scanf("%lf", &guad10);
    printf ("caro/a manager, dammi il guadagno del negozio 11: ");
    scanf("%lf", &guad11);
    printf ("caro/a manager, dammi il guadagno del negozio 12: ");
    scanf("%lf", &guad12);
    printf ("caro/a manager, dammi il guadagno del negozio 13: ");
    scanf("%lf", &guad13);
    printf ("caro/a manager, dammi il guadagno del negozio 14: ");
    scanf("%lf", &guad14);
    printf ("caro/a manager, dammi il guadagno del negozio 15: ");
    scanf("%lf", &guad15);
    printf ("caro/a manager, dammi il guadagno del negozio 16: ");
    scanf("%lf", &guad16);
    printf ("caro/a manager, dammi il guadagno del negozio 17: ");
    scanf("%lf", &guad17);
    printf ("caro/a manager, dammi il guadagno del negozio 18: ");
    scanf("%lf", &guad18);
    printf ("caro/a manager, dammi il guadagno del negozio 19: ");
    scanf("%lf", &guad19);
    printf ("caro/a manager, dammi il guadagno del negozio 20: ");
    scanf("%lf", &guad20);

    media = (guad1+guad2+guad3+guad4+guad5);
    media += (guad6+guad7+guad8+guad9+guad10);
    media += (guad11+guad12+guad13+guad14+guad15);
    media += (guad16+guad17+guad18+guad19+guad20);
    media = media/N;

    printf ("\nOUTPUT\n-----\n");
    printf ("caro/a manager, la media è %g, e i negozi da controllare sono:\n", media);

    if (guad1 < media*1/3)
        printf ("Negozio 1, con guadagno %g\n", guad1);
    if (guad2 < media*1/3)
        printf ("Negozio 2, con guadagno %g\n", guad2);
    if (guad3 < media*1/3)
        printf ("Negozio 3, con guadagno %g\n", guad3);
    if (guad4 < media*1/3)
        printf ("Negozio 4, con guadagno %g\n", guad4);
    if (guad5 < media*1/3)
        printf ("Negozio 5, con guadagno %g\n", guad5);
    if (guad6 < media*1/3)
        printf ("Negozio 6, con guadagno %g\n", guad6);
    if (guad7 < media*1/3)
        printf ("Negozio 7, con guadagno %g\n", guad7);
    if (guad8 < media*1/3)
        printf ("Negozio 8, con guadagno %g\n", guad8);
```

Potrebbe essere "20 Negozi"

```
#include <stdio.h>
scanf("%lf", &guad14);
if (guad9<media*1/3) negozio 15: ");
    printf ("Negozio 9, con guadagno %g\n", guad9);
if (guad10<media*1/3) negozio 16: ");
    printf ("Negozio 10, con guadagno %g\n", guad10);
if (guad11< media*1/3) negozio 17: ");
    printf ("Negozio 11, con guadagno %g\n", guad11);
if (guad12<media*1/3) negozio 18: ");
    printf ("Negozio 12, con guadagno %g\n", guad12);
if (guad13<media*1/3) negozio 19: ");
    printf ("Negozio 13, con guadagno %g\n", guad13);
if (guad14<media*1/3) negozio 20: ");
    printf ("Negozio 14, con guadagno %g\n", guad14);
if (guad15<media*1/3)
    printf ("Negozio 15, con guadagno %g\n", guad15);
if (guad16< media*1/3)
    printf ("Negozio 16, con guadagno %g\n", guad16);
if (guad17<media*1/3)
    printf ("Negozio 17, con guadagno %g\n", guad17);
if (guad18<media*1/3)
    printf ("Negozio 18, con guadagno %g\n", guad18);
if (guad19<media*1/3) ozi da controllare sono:\n", media);
    printf ("Negozio 19, con guadagno %g\n", guad19);
if (guad20<media*1/3)
    printf ("Negozio 20, con guadagno %g\n", guad20);

printf("FINE\n");
return 0;
}

scanf("%lf", &guad9);
printf ("caro/a manager, dammi");
scanf("%lf", &guad10);
printf ("caro/a manager, dammi");
scanf("%lf", &guad11);
printf ("caro/a manager, dammi");
scanf("%lf", &guad12);
printf ("caro/a manager, dammi");
scanf("%lf", &guad13);
printf ("caro/a manager, dammi");
scanf("%lf", &guad14);
printf ("caro/a manager, dammi");
scanf("%lf", &guad15);
printf ("caro/a manager, dammi");
scanf("%lf", &guad16);
printf ("caro/a manager, dammi");
scanf("%lf", &guad17);
printf ("caro/a manager, dammi");
scanf("%lf", &guad18);
printf ("caro/a manager, dammi");
scanf("%lf", &guad19);
printf ("caro/a manager, dammi");
scanf("%lf", &guad20);
printf ("caro/a manager, dammi");

if (guad4<media*1/3)
    printf ("Negozio 4, con guadagno %g\n", guad4);
if (guad5<media*1/3)
    printf ("Negozio 5, con guadagno %g\n", guad5);
if (guad6< media*1/3)
    printf ("Negozio 6, con guadagno %g\n", guad6);
if (guad7<media*1/3)
    printf ("Negozio 7, con guadagno %g\n", guad7);
if (guad8<media*1/3)
    printf ("Negozio 8, con guadagno %g\n", guad8);
```

Potrebbe essere "20 Negozi"

```
#include <stdio.h>
scanf("%lf", &guad14);
if (guad9<media*1/3)
    printf ("Negozio 9, con guadagno %g\n", guad9);
if (guad10<media*1/3)
    printf ("Negozio 10, con guadagno %g\n", guad10);
if (guad11<media*1/3)
    printf ("Negozio 11, con guadagno %g\n", guad11);
if (guad12<media*1/3)
    printf ("Negozio 12, con guadagno %g\n", guad12);
if (guad13<media*1/3)
    printf ("Negozio 13, con guadagno %g\n", guad13);
if (guad14<media*1/3)
    printf ("Negozio 14, con guadagno %g\n", guad14);
if (guad15<media*1/3)
    printf ("Negozio 15, con guadagno %g\n", guad15);
if (guad16<media*1/3)
    printf ("Negozio 16, con guadagno %g\n", guad16);
if (guad17<media*1/3)
    printf ("Negozio 17, con guadagno %g\n", guad17);
if (guad18<media*1/3)
    printf ("Negozio 18, con guadagno %g\n", guad18);
if (guad19<media*1/3)
    printf ("Negozio 19, con guadagno %g\n", guad19);
if (guad20<media*1/3)
    printf ("Negozio 20, con guadagno %g\n", guad20);
printf("FINE");
return 0;
}
```

```
INPUT
-----
caro/a manager, dammi il guadagno del negozio 1: 1
caro/a manager, dammi il guadagno del negozio 2: 2
caro/a manager, dammi il guadagno del negozio 3: 3
caro/a manager, dammi il guadagno del negozio 4: 4
caro/a manager, dammi il guadagno del negozio 5: 5
caro/a manager, dammi il guadagno del negozio 6: 6
caro/a manager, dammi il guadagno del negozio 7: 7
caro/a manager, dammi il guadagno del negozio 8: 8
caro/a manager, dammi il guadagno del negozio 9: 9
caro/a manager, dammi il guadagno del negozio 10: 0
caro/a manager, dammi il guadagno del negozio 11: 11
caro/a manager, dammi il guadagno del negozio 12: 11
caro/a manager, dammi il guadagno del negozio 13: 12
caro/a manager, dammi il guadagno del negozio 14: 11
caro/a manager, dammi il guadagno del negozio 15: 22
caro/a manager, dammi il guadagno del negozio 16: 33
caro/a manager, dammi il guadagno del negozio 17: 33
caro/a manager, dammi il guadagno del negozio 18: 44
caro/a manager, dammi il guadagno del negozio 19: 1
caro/a manager, dammi il guadagno del negozio 20: 1
```

```
OUTPUT
-----
caro/a manager, la media e' 11.2, e i negozi da controllare sono:
Negozio 1, con guadagno 1
Negozio 2, con guadagno 2
Negozio 3, con guadagno 3
Negozio 10, con guadagno 0
Negozio 19, con guadagno 1
Negozio 20, con guadagno 1
FINE
```

Potrebbe essere "1000 Negozi" ...



Cosa non va?

- **lunghezza del codice**
 - 1000 volte invece che 20, invece che 5 **ma sono tutte operazioni ripetitive**
- **diverso numero di negozi → "programma diverso"**
 - ...
 - EPPURE il problema non sembra diverso!**

Potrebbe essere "1000 Negozi" ...



Cosa non va?

- **lunghezza del codice** che cresce in modo sconvolgente ... dove invece ci sono chiaramente **operazioni ripetitive che dovrebbero essere messe a fattor comune**
 - ma non possono ... perché dobbiamo usare variabili diverse, con identificatori diversi; identificatori che vanno trattati uno per uno, in istruzioni dedicate ...
- **diverso numero di negozi** → **"programma diverso"** (con modifiche all'algorithmo non fattorizzabili) ... **EPPURE il problema non sembra diverso!**

Osservazione

Nei programmi usiamo le **VARIABILI SEMPLICI** che conosciamo ...

(Una variabile semplice è capace di contenere un singolo dato ...).

Così, tra le variabili **quad-i** ci sono aspetti comuni che non riusciamo a sfruttare per ottenere codice più razionale ...

Ma abbiamo uno strumento che può aiutarci ...

array

Una variabile array è una

VARIABLE STRUTTURATA cioè capace di contenere **più valori**.

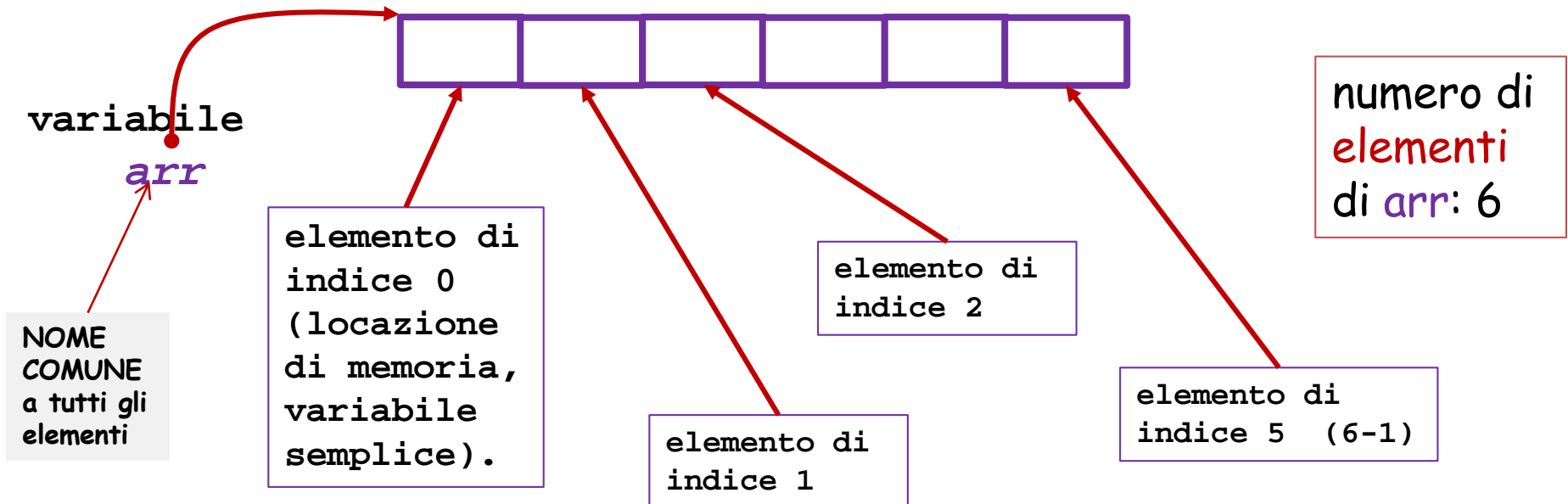
In una variabile array i **VALORI** sono **OMOGENEI**, cioè tutti del medesimo tipo.

Una variabile array è costituita da una

SEQUENZA di ELEMENTI

Gli elementi

- un **NOME COMUNE** (quello della variabile array)
- e un **INDICE** (per distinguerli quello della variabile array)



array

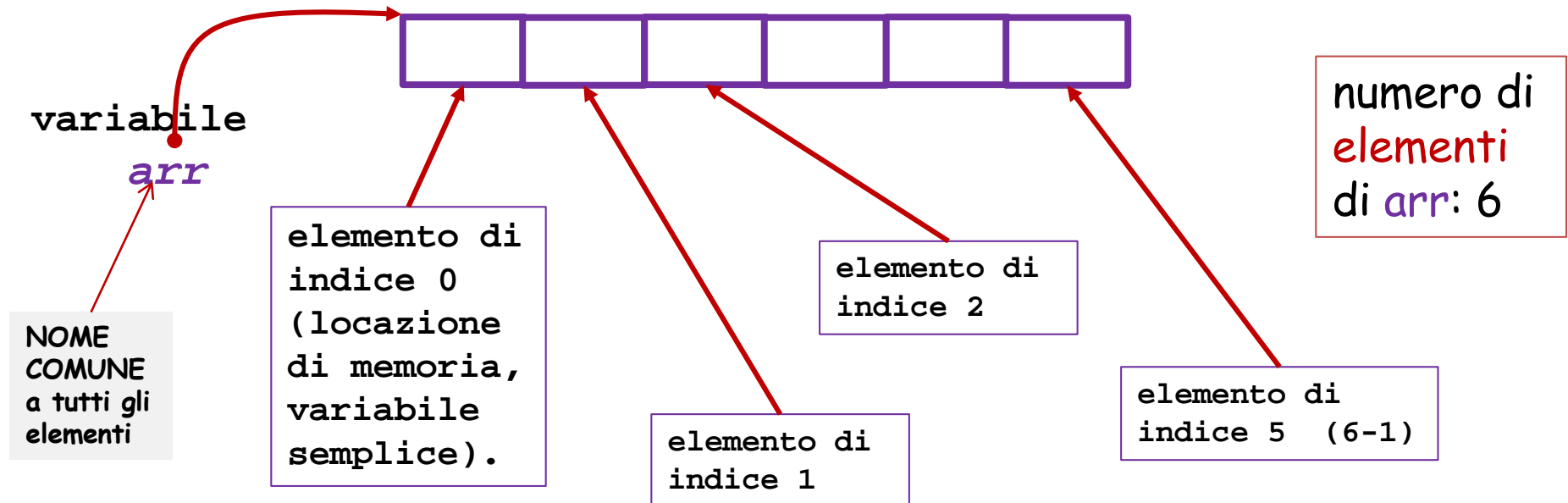
Una variabile array è una **VARIABILE STRUTTURATA**, cioè capace di contenere più valori.

In particolare, i **VALORI** contenuti in un array devono essere **OMOGENEI**, cioè tutti del medesimo tipo.

Una variabile array è costituita da una **SEQUENZA** di variabili semplici omogenee ("elementi" dell'array), che hanno

- un **NOME COMUNE**
- e un **INDICE**

così da poterle distinguere tra loro e con gli elementi di altri array del programma.



array = sequenza di variabili con indice

Quando una variabile array, **arr**, viene dichiarata, come
array di *N* elementi di tipo *Type*,

Type arr[N]

viene allocata in memoria una sequenza di locazioni, aventi ciascuna

- dimensione adatta a contenere valori di tipo *Type*
- NOME COMUNE *arr*
- INDICE tra 0 e *N*-1

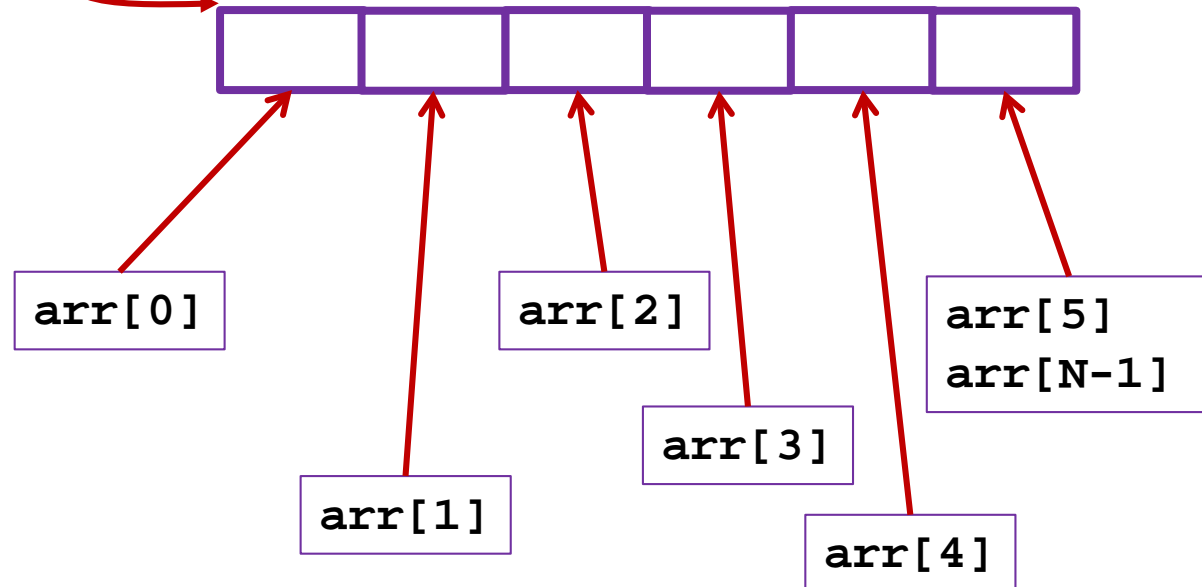
Es. `int arr[N]`
variabile *arr*

`#define N 6` numero di
elementi
di *arr*: 6

NOTAZIONE
per accedere
agli elementi
dell'array *arr*

NOME COMUNE [INDICE]
arr[ind]

0 1 2 3 4 5



array = sequenza di variabili con indice (e relativi valori)

```
#define N 6
```

```
Es. int arr[N]  
variabile arr
```

```
...
```

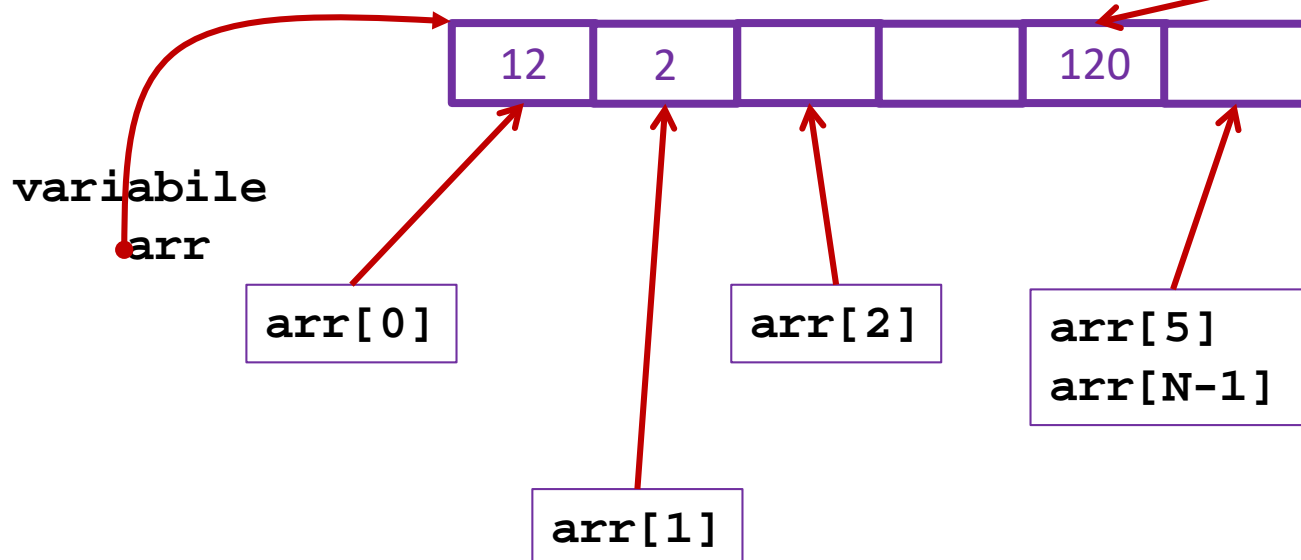
```
arr[0]=12;
```

```
arr[1]=2;
```

```
arr[4]=120;
```

numero di **elementi** di arr: 6

il valore contenuto
in arr[4] è 120



array = sequenza di variabili con indice - **limitata!**

Quando una variabile array, `arr`, viene dichiarata, come array di `N` elementi di tipo `Type`,
`Type arr[N]`

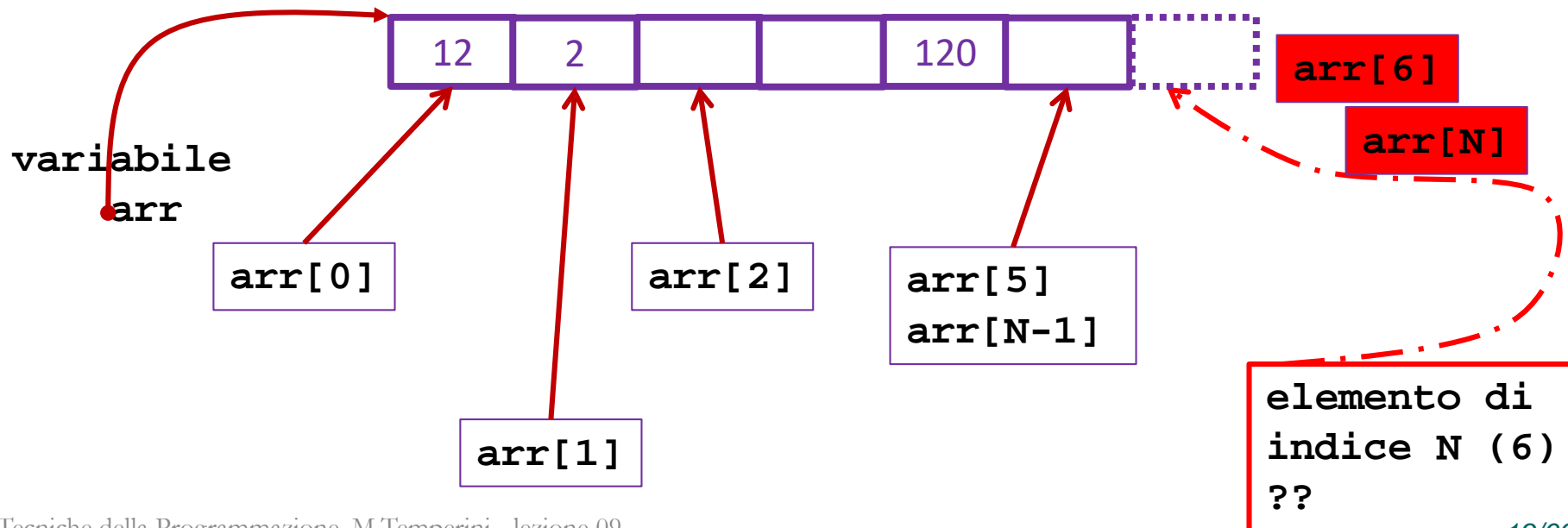
viene allocata in memoria una sequenza di locazioni, aventi ciascuna

- dimensione adatta a contenere valori di tipo `Type`
- NOME COMUNE `arr`
- INDICE tra 0 e `N-1`

Es. `int arr[N]`
variabile `arr`

`#define N 6`

numero di **elementi** di `arr`: 6

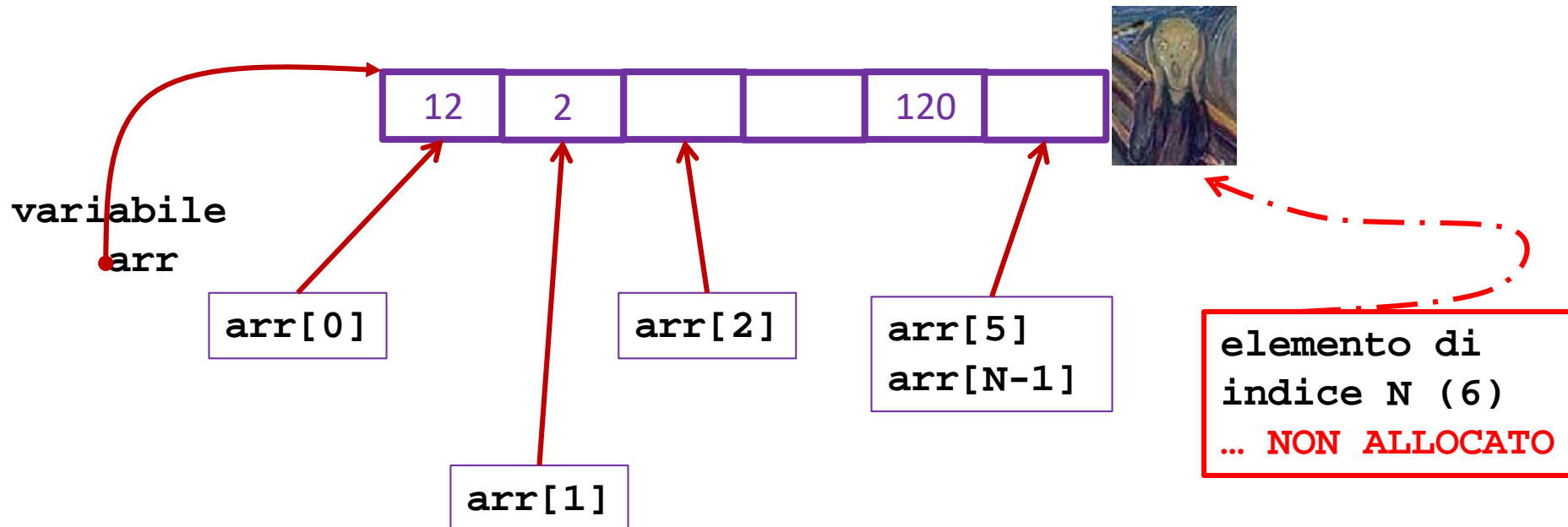


array = sequenza di variabili con indice - **limitata!**

```
#define N 6
```

```
...  
arr[0]=12;  
arr[1]=2;  
arr[4]=120;  
int crash=23;
```

```
arr[6]=crash;
```



array = sequenza di variabili IN MEMORIA

```
#define N 6  
int arr[N]
```

dichiarazione dell'array di
N interi "arr"

arr è una variabile speciale, che
contiene l'indirizzo dell'inizio del
blocco di memoria allocato per
l'array;
le locazioni allocate sono
sequenziali e di tipo int

arr

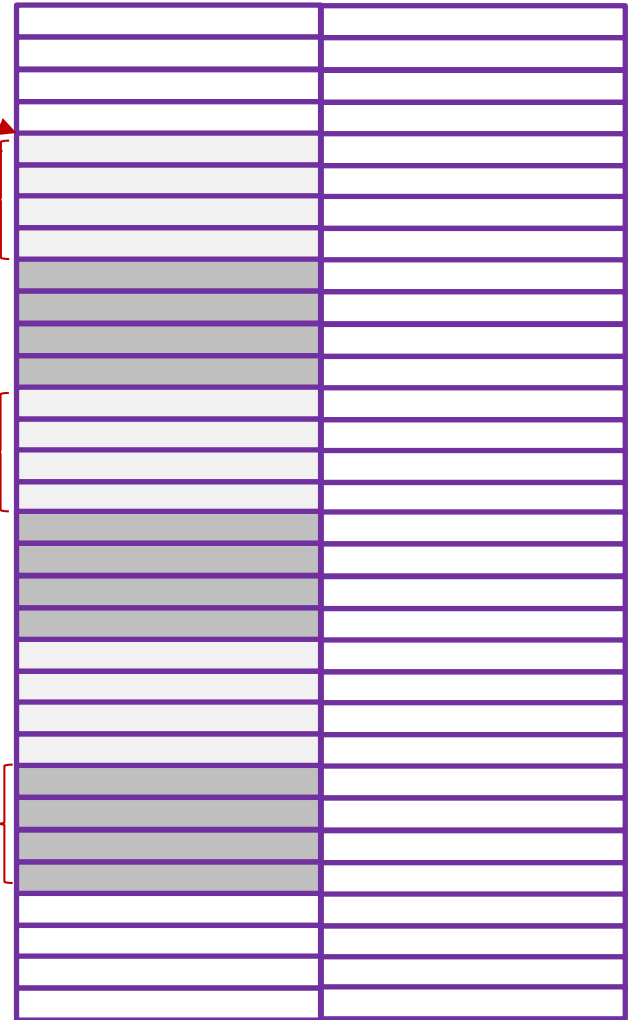
12	9	47	42	61	8
----	---	----	----	----	---

VISUALIZZAZIONE LOGICA

arr



^^



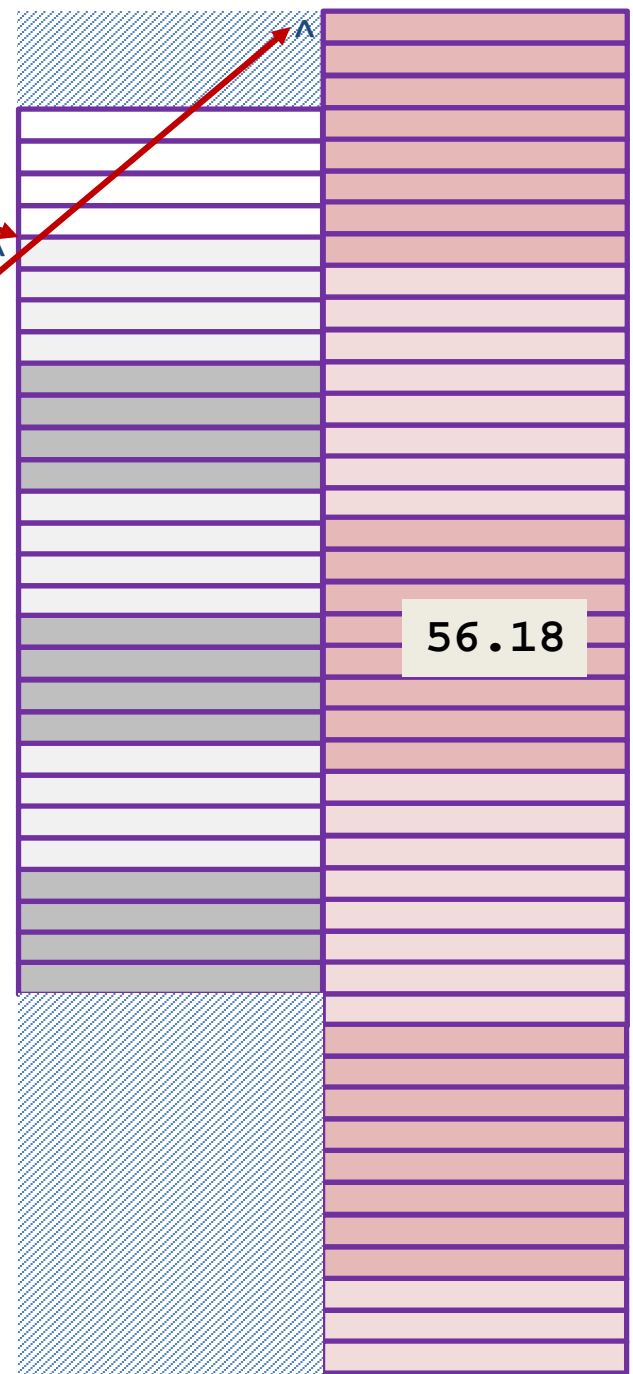
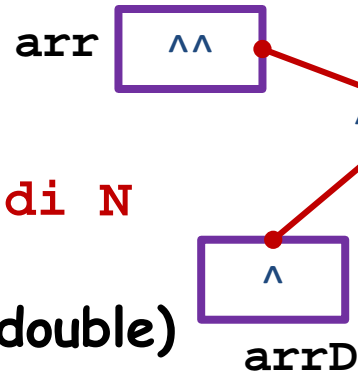
MEMORIA

Qui si vedono due visualizzazioni dell'array: quella "logica" come sequenza di scatole, e quella ben radicata nella RAM.

elementi di un array: locazioni di memoria

```
#define N 6  
int arr[N];  
double arrD[N];
```

dichiarazione dell'array di N
double "arrD"
(locazioni sequenziali e di tipo double)

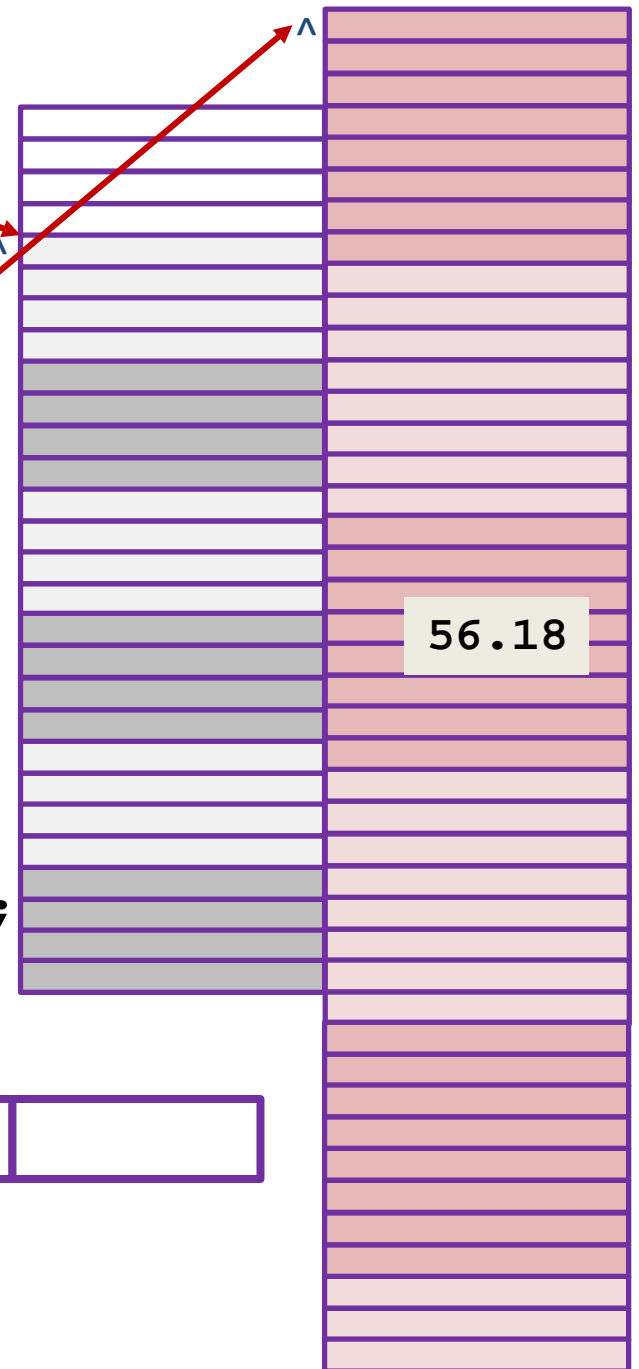
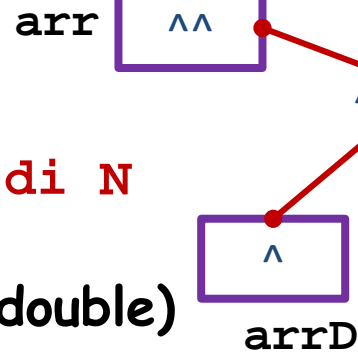


\wedge è l'indirizzo di arrD[0]
 $\wedge\wedge$ è l'indirizzo di arr[0]

elementi di un array: accesso/memorizzazione

```
#define N 6
int arr[N];
double arrD[N];
```

dichiarazione dell'array di N
double "arrD"
(locazioni sequenziali e di tipo double)



Le variabili di un array sono usabili
come qualsiasi variabile semplice:

MEMORIZZAZIONE **ACCESSO**
arrD[2]=56.18 double d = arrD[4];
arrD[1]=32.59

arrD

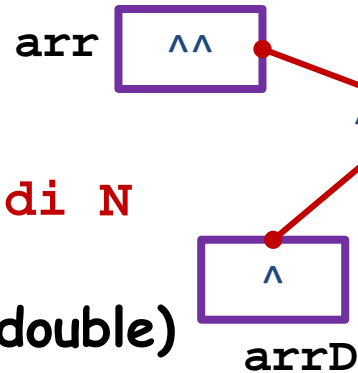


dove sta ?
32.59

elementi di un array: accesso/memorizzazione

```
#define N 6
int arr[N];
double arrD[N];
```

dichiarazione dell'array di N
double "arrD"
(locazioni sequenziali e di tipo double)



Le variabili di un array sono usabili
come qualsiasi variabile semplice:



MEMORIZZAZIONE

```
arrD[2]=56.18
```

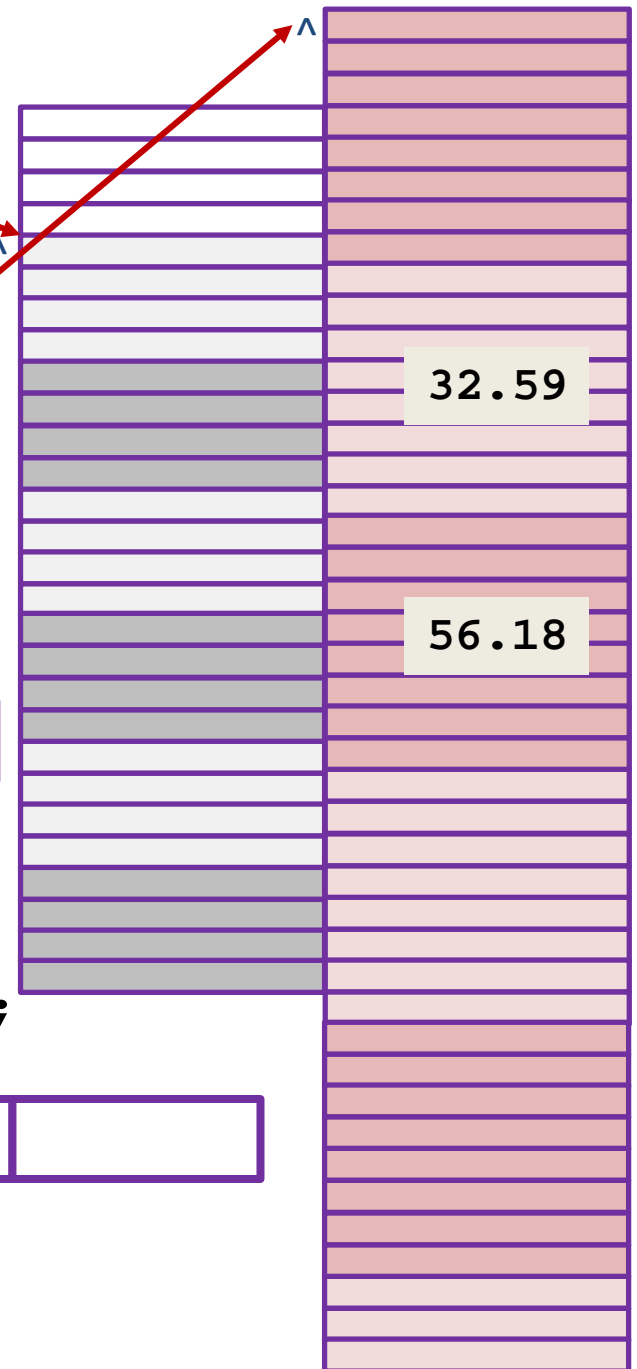
```
arrD[1]=32.59
```

arrD



ACCESSO

```
double d = arrD[4];
```



elementi di un array: accesso/memorizzazione

```
#define N 6
```

```
double arrD[N];
```

dichiarazione dell'array di N

```
double "arrD"
```

(sequenziali e di tipo double)

Le variabili di un array sono usabili
come qualsiasi variabile semplice:

MEMORIZZAZIONE

```
arrD[2]=56.18
```

```
arrD[1]=32.59
```

```
arrD
```



ACCESSO

```
double d = arrD[4];
```

arr

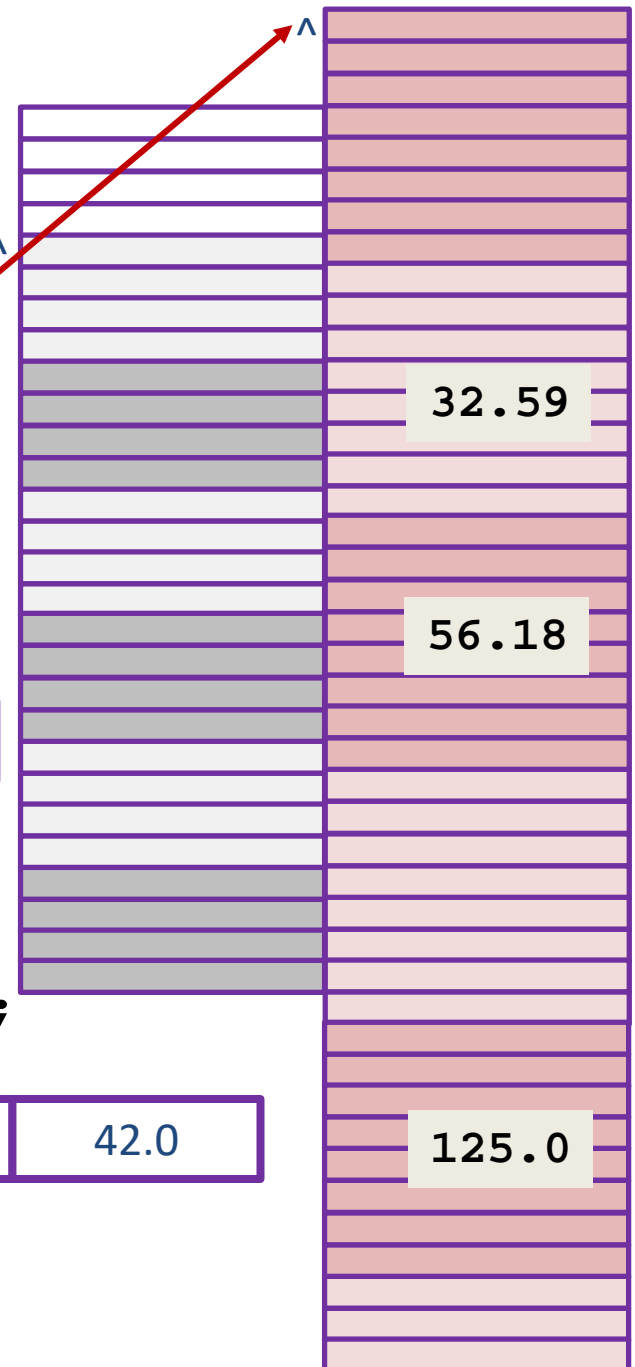


^^



arrD

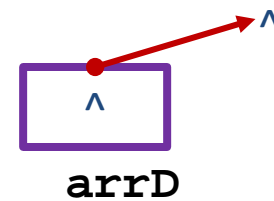
d



elementi di un array: accesso/memorizzazione

```
#define N 6
```

dichiarazione di un array di N elementi double



```
double arrD[N];
```

```
double arrDD[6];
```

dichiarazione di un array di 6 elementi double

```
double arrF[N*2];
```

dichiarazione di un array di 12 elementi ...

```
arrD[i]
```

variabile i-esima di arrD

```
&arrD[i]
```

indirizzo della variabile i-esima di arrD

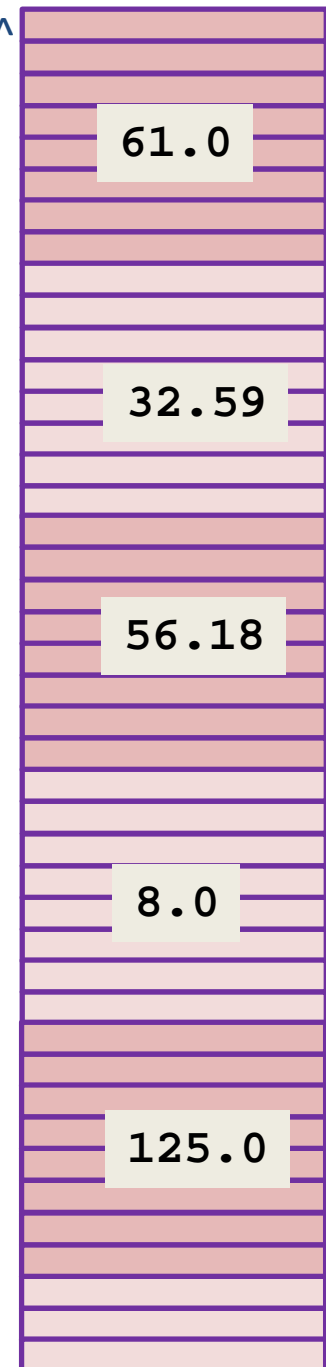
```
scanf("%lf", &arrD[3]);
```

```
printf("%g", arrD[4]);
```

```
printf("%g", arrD[5]);
```

```
arrD
```

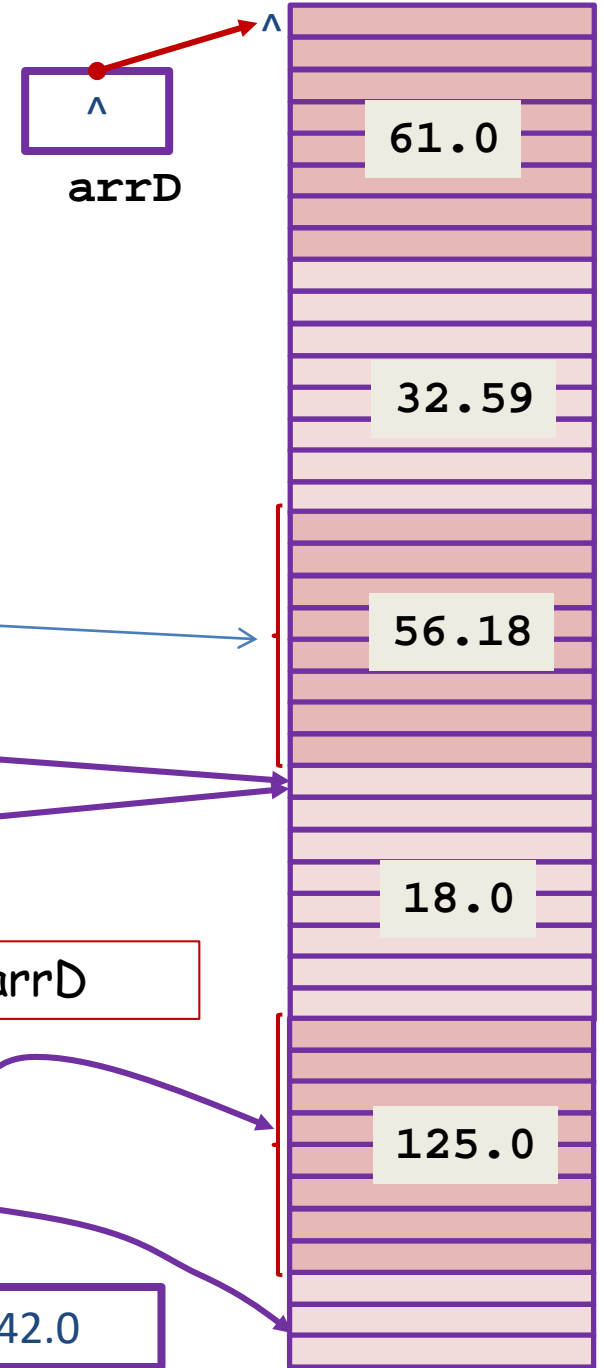
61.0	32.59	56.18	8,0	125.0	42.0
------	-------	-------	-----	-------	------



...

```
#define N 6

double arrD[N];
double arrDD[6];
double arrF[N*2];
```



`arrD[2]` variabile `arrD[2]`

`&arrD[3]` | indirizzo della variabile `arrD[3]` |

`scanf("%lf", &arrD[3]);`
 come abbiamo messo 18.0 nell'elemento di indice 3 di `arrD`

`printf("%g", arrD[4]);` stampa 125.0

`printf("%g", arrD[5]);` stampa 42.0

`arrD`

61.0	32.59	56.18	18,0	125.0	42.0
------	-------	-------	------	-------	------

Torniamo ai Negozi

Algoritmo

- 0) usiamo un array di N (costante, da ora 6) elementi double;
costante $RATIO = 1/3$;
sarà var **double guadagni**[N],
poi ci serve media,

guadagni



supponendo $N=6$

Torniamo ai Negozi

Algoritmo

0) usiamo un array di N elementi double; costante $RATIO = 1/3$;
sarà var **double guadagni[N]**,
poi ci serve media,

ci serve un contatore i per gli indici ...

1) leggere i guadagni, memorizzandoli nell'array

guadagni

&%\$!@	&\$!!@!	&%\$!!@	&%!!@	&%\$!!@	&%\$!!?@
--------	---------	---------	-------	---------	----------

supponendo $N=6$

Torniamo ai Negozi

Algoritmo

0) costante N, costante $RATIO = 1/3$
var **double guadagni[N]**, media
contatore i

1) leggere i guadagni, memorizzandoli nell'array

!! un momento di riflessione
che codice usiamo per questo punto?

```
?  
scanf("%d", &guadagni[0]);  
scanf("%d", &guadagni[1]);  
scanf("%d", &guadagni[2]);  
scanf("%d", &guadagni[3]);  
scanf("%d", &guadagni[4]);  
scanf("%d", &guadagni[5]);  
?
```

guadagni

&%\$!@	&\$!!@!	&%\$!!@	&%!!@	&%\$!!@	&%\$!!?@
--------	---------	---------	-------	---------	----------

macche' ...

supponendo N=6

Torniamo ai Negozi

Algoritmo

- 0) costante N , costante $RATIO = 1/3$
var **double guadagni**[N], media,
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
 - 1.1) per i che va da 0 a $N-1$,
 - 1.1.1) chiedere il guadagno i -esimo
 - 1.1.2) leggere l'elemento i di guadagni

guadagni

&%\$!@	&\$!!@!	&%\$!!@	&%!!@	&%\$!!@	&%\$!!?@
--------	---------	---------	-------	---------	----------

supponendo $N=6$

Torniamo ai Negozi

Vedi Approfondimenti

Algoritmo

- 0) costante N, costante $RATIO = 1/3$
var **double guadagni[N]**, media
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
 - 1.1) per i che va da 0 a N-1,
 - 1.1.1) chiedere il guadagno i-esimo
 - 1.1.2) leggere l'elemento i di guadagni

!!

```
for (i=0; i<N; i++)  
    printf(... richiesta i ...);  
    scanf("%lf", &guadagni[i]);
```

i

0

guadagni

&%\$!!@	&%\$!!@	&%\$!!@	&%\$!!@	&%\$!!@	&%\$!!@
---------	---------	---------	---------	---------	---------

supponendo $N=6$ e che i guadagni dati in input siano 61, 32.59, 56.18, 8, 125, 42
--- inizialmente l'array è "vuoto", cioè c'è roba estranea

Torniamo ai Negozi (ora la media)

Algoritmo

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, **somma**;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array

sbadati ... ce ne siamo accorti solo ora ...

61.0	32.59	56.18	8.0	125.0	42.0
------	-------	-------	-----	-------	------

somma

%@\$

- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma
 - 2.2) $media = somma/N$

media

&%@\$!!@

- 3) scorrere l'array, stampando gli elementi minori di $media * RATIO$

Vedi Approfondimenti

somma e' una variabile usata come "accumulatore" (in generale un accumulatore serve ad ...accumularci dentro dei dati per poi fare dei calcoli sul valore complessivo)

Torniamo ai Negozi (ora il resto)

Algoritmo

supponendo
N=6 e che i guadagni
dati in input siano 61,
32,59, 56,18, 8, 125,
42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma
 - 2.2) $media = somma/N$
- 3) scorrere l'array, stampando gli elementi minori di $media * RATIO$

?

i

6

somma

~~292.77~~ 324.77

media

54,128333

61.0

32.59

56.18

8.0

125.0

42.0

Torniamo ai Negozi (ora il resto)

Algoritmo

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma
 - 2.2) $media = somma/N$
- 3) scorrere l'array, stampando gli elementi minori di $media * RATIO$
cioè
per i che va da 0 a N-1,
se $guadagni[i] < media * RATIO$ ☺
stampare guadagni[i]

i

6

somma

~~292.77~~ 324.77

media

54,128333

61.0

32.59

56.18

8.0

125.0

42.0

Torniamo ai Negozi (ora il resto)

Algoritmo

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma
 - 2.2) $media = somma/N$
- 3) scorrere l'array, stampando gli elementi minori di $media * RATIO$

```
printf(" (media=%g) ecco i negozi che ... poco:\n", media);  
for (i=0; i<N; i++)  
    if (guadagni[i] < media*RATIO)  
        printf("Negozio %d con Euro %g\n", i, guadagni[i]);
```

i

...

OUTPUT?

~~292.77~~ 324.77

somma

media

61.0	32.59	56.18	8.0	125.0	42.0
------	-------	-------	-----	-------	------

54,128333

Torniamo ai Negozi (ora il resto)

Algoritmo

```
0) caro/a manager, dammi il guadagno del negozio 0: 61
   caro/a manager, dammi il guadagno del negozio 1: 32.59
   caro/a manager, dammi il guadagno del negozio 2: 56.18
1) caro/a manager, dammi il guadagno del negozio 3: 8
2) caro/a manager, dammi il guadagno del negozio 4: 125
   caro/a manager, dammi il guadagno del negozio 5: 42
   (media=54.1283) ecco i negozi che guadagnano poco:
3) - negozio 3 con Euro 8
   FINE
```

```
printf("
for (i=0
if (gu
pri
n", media);
dagni[i]);
```

programma per i negozi,
nei complementi didattici
(anche quelli "primitivi" ... con le numerose variabili)

Questo sviuppatelo per conto vostro, seguendo le slides.

Poi controllate nei complementi didattici per una possibile versione.

i ...

somma

7 324.77

media

61.0

32.59

56.18

8.0

125.0

42.0

54,128333

Dichiarazione di array, con inizializzazione

costante N=6

```
double arr[N]={4,5,6,61,42,47}
```

viene allocato un array di N elementi double;

arr[0] viene assegnato a 4 ... arr[5] viene assegnato a 47

disegnare l'array riempito ... 😊

Sempre: i valori devono essere del tipo dichiarato per l'array, o convertibili senza problemi a quel tipo.

(Conversione in assegnazione)

Dichiarazione di array, con inizializzazione

costante N=6

```
double arr[N]={4,5,6,61,42,47}
```

viene allocato un array di N elementi double;

arr[0] viene assegnato a 4 ... arr[5] viene assegnato a 47

4	5	6	61	42	47
---	---	---	----	----	----

Sempre: i valori devono essere del tipo dichiarato per l'array, o convertibili senza problemi a quel tipo.

(Conversione in assegnazione)

Dichiarazione di array, con inizializzazione

```
double vett[]={32.59, 116, 50.08, 56.18, 47, 42};
```

viene allocato un array di tanti elementi double, quanti sono i valori specificati tra le graffe.

vett[0] viene assegnato a **32.59** ... **vett[3]** viene assegnato a **56.18**

vett[5] viene assegnato a ? ... **vett[6]** viene assegnato a ?

[Vedi Approfondimenti](#)

Sempre: i valori devono essere del tipo dichiarato per l'array, o convertibili senza problemi a quel tipo.
(Conversione in assegnazione)

Scusi, dove sono le locazioni di un array?

Lo abbiamo visto in un disegno prima ...



```
#include <stdio.h>
#define N 8
```

```
int main () {
    int arr[N] = {5618, 42, 180, 61, 47, 125, 116, 3783}, i;

    for (i=0; i<N; i++)
        printf("l'elemento arr[%d] è di %d byte, ha
            indirizzo %p e valore %d\n", i,
            sizeof(arr[i]), &arr[i], arr[i]);

    printf("\n\n e adesso HORROR!\n\n");
    for (i=0; i<=N+1; i++)
        printf("arr[%d] = %d\n", i, arr[i]);

    printf("FINE\n");
    return 0;
}
```

Qui ci sono due for: uno stampa l'array in un certo modo, l'altro lo stampa in modalita' horror.

Prevedere l'output del primo for.
(Verificare facendo girare il codice ...)

Cosa stampa il secondo for?
E perche' e' horror allo stato puro?

Per tutto, vedi [Approfondimenti](#)

Prodotto scalare tra vettori

I vettori matematici/fisici hanno una ovvia rappresentazione come array in un programma C.

PROBLEMA: calcolare il prodotto scalare dei due vettori
vettore1 = (32.59, 116, 50.08, 56.18, 47, 42)
vettore2 = (4,5,6,61,42,47)

```
#define N 6
int main () {
    double vettore1[]={32.59, 116, 50.08, 56.18, 47, 42};
    double vettore2[N]={4,5,6,61,42,47},
    pScalare;          /* ... */
    int i;             /* contatore */
    ...
}
```

Prodotto scalare tra vettori (Algoritmo)

```
caro/a utente, primo vettore: ( 32.59 116 50.08 56.18 47 42 )
caro/a utente, secondo vettore: ( 4 5 6 61 42 47 )
ok, prodotto scalare tra i due = 8385.82
FINE
```

Algoritmo

- 0) costante N, vettore1 di N elementi double, vettore2 idem, pScalare, contatore i
- 1) stampare vettore1
 - stampare "... ("
 - per i da 0 a N-1 stampare vettore1[i]
 - stampare ")\n"
- 2) stampare vettore2
- 3) accumulare i prodotti vettore1[i]*vettore2[i] in pScalare
 - pScalare = 0
 - per i da 0 a N-1 pScalare = pScalare + ...
- 4) stampare pScalare
- 5) fine

Fare il programma e provarlo
Vedi Esercizi

Ancora su funzioni:

nomi uguali in scope diversi

"scope" = campo di visibilità ...

in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

Ancora su funzioni:

nomi uguali in scope diversi

in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

```
int main() {
    int result, primo, secondo,;
    ...
    scanf("%d %d", &primo, &secondo);
    ...
    printf("mcd=%d\n", mcd(primo, secondo));
    return 0;
}
```

```
int mcd (int primo, int secondo) {
    int result=0;
    ...
    return result;
}
```

Ancora su funzioni:

nomi uguali in scope diversi

in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

```
int main() {  
    int result, primo, secondo;  
    ...  
    scanf("%d %d", &primo, &secondo);  
    ...  
    printf("mcd=%d\n", mcd(primo, secondo));  
    return 0;  
}
```

scope
delle
funzione
chiamante

```
int mcd (int primo, int secondo) {  
    int result = 0;  
    ...  
    return result;  
}
```

Vedi Approfondimenti

scope
delle
funzione
chiamata

Ancora su funzioni: modifica parametri attuali



Un esempio dice tutto:

```
void adessoModifico (int num) {
    num = num+1;
return;
}

int main () {
    int n=47;

    printf("valore di n prima: %d\n", n);
    adessoModifico(n);
    printf("valore di n dopo: %d\n", n);

    printf("FINE\n");
return 0;
}
```

n viene "passata" alla funzione ...

QUINDI
*potrebbe venire in mente che
l'istruzione*

num=num+1

possa mettere 48 dentro a n

Ancora su funzioni: modifica parametri attuali (tsk)



Un esempio dice tutto:

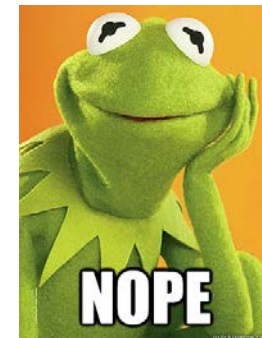
```
void adessoModifico (int num) {  
    num = num+1;  
return;  
}  
  
int main () {  
    int n=47;  
  
    printf("valore di n prima: %d\n", n);  
    adessoModifico(n);  
    printf("valore di n dopo: %d\n", n);  
  
    printf("FINE\n");  
    return 0;  
}
```

Perche' no?
Perche` il passaggio dei parametri e` "PER VALORE"
Vedi Approfondimenti

n viene "passata" alla funzione ...

QUINDI
potrebbe venire in mente che l'istruzione
num=num+1

possa mettere 48 dentro a n



Funzioni con parametri array

Programma per il Prodotto scalare mediante funzioni su array (1/3)

come diventa il programma sul prodotto scalare visto poco fa

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

```
#define N 6
```

```
/* prototipi 😊 */
```

```
int main () {
```

```
double vettore1[]={32.59, 116, 50.08, 56.18, 47, 42};
```

```
double vettore2[N]={4,5,6,61,42,47},
```

```
/* stampa dei vettori */
```

```
printf ("caro/a utente, ecco primo vettore: (");
```

```
stampaArray(vettore1);
```

```
printf (")\n");
```

```
printf ("caro/a utente, ecco secondo vettore: (");
```

```
stampaArray(vettore2);
```

```
printf (")\n");
```

```
printf("ok, prodotto scalare tra i due = %g\n",
```

```
pScalareFun(vettore1, vettore2));
```

FINE

Programma per il Prodotto scalare mediante funzioni su array (1/3)

come diventa il programma sul prodotto scalare visto poco fa

```
#include <stdio.h>
#define N 6
double pScalareFun (double v1[N], double v2[N]);
void stampaArray (double v[N]);

int main () {
    double vettore1[]={32.59, 116, 50.08, 56.18, 47, 42};
    double vettore2[N]={4,5,6,61,42,47},
        /* stampa dei vettori */
    printf ("caro/a utente, ecco primo vettore: (");
    stampaArray(vettore1);
    printf (")\n");
    printf ("caro/a utente, ecco secondo vettore: (");
    stampaArray(vettore2);
    printf (")\n");

    printf("ok, prodotto scalare tra i due = %g\n",
        pScalareFun(vettore1, vettore2));
```

FINE

funzione prodottoScalare

Esempio: funzione che calcola il prodotto scalare tra due vettori

```
#define N 6
double pScalareFun (double v1[N], double v2[N])
{
    int i;                /* contatore*/
    double p = 0.0;      /* accumulatore */

    for (i=0; i<N; i++)
        p += v1[i]*v2[i];
return p;
}
```

verificare, su carta: questo codice funziona?



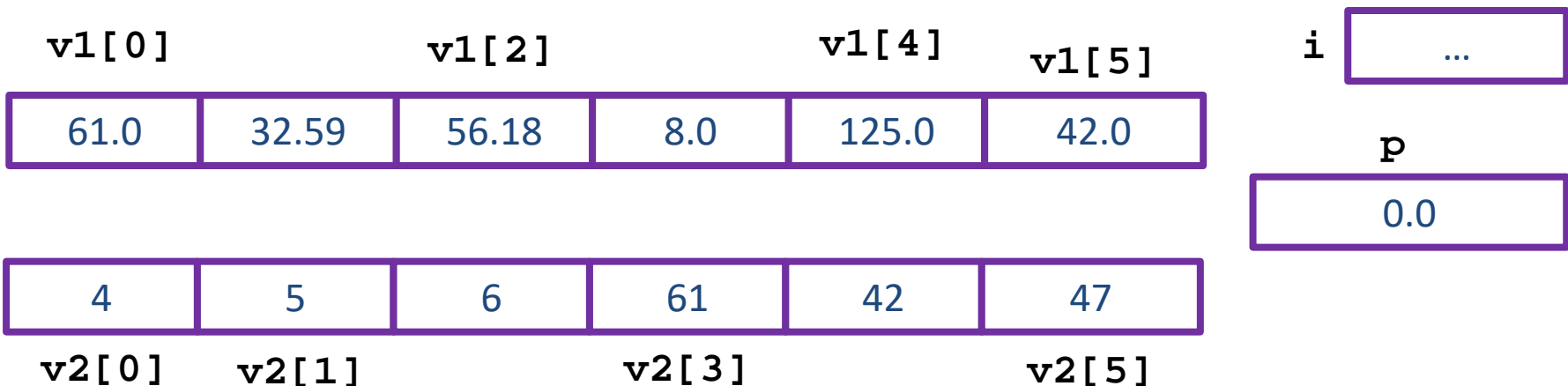
Funzione prodottoScalare

Esempio: funzione che calcola il prodotto scalare tra due vettori

```
#define N 6
double pScalareFun (double v1[N], double v2[N]) {
    int i;          /* contatore */
    double p = 0.0; /* accumulatore */

    for (i=0; i<N; i++)
        p += v1[i]*v2[i];
    return p;
}
```

esecuzione simulata ...

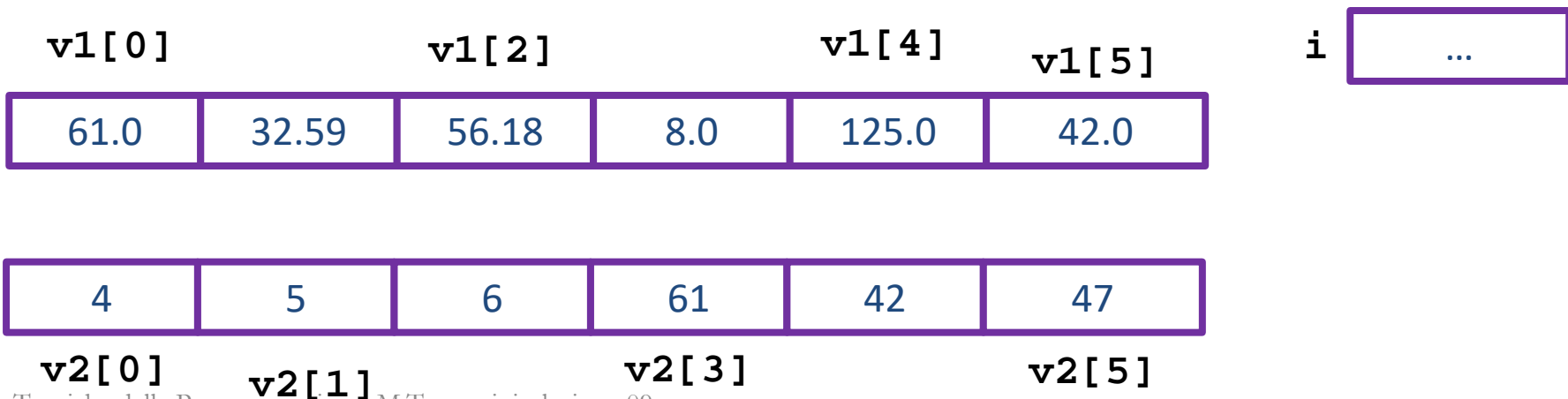


Funzione stampaArray()

Esempio: funzione che **stampa un array** di double

```
#define N 6
void stampaArray (double v[N]) {
    int i;          /* contatore*/

    for (i=0; i<N; i++) {
        printf (" %g ", v[i]);
    }
return;
}
```



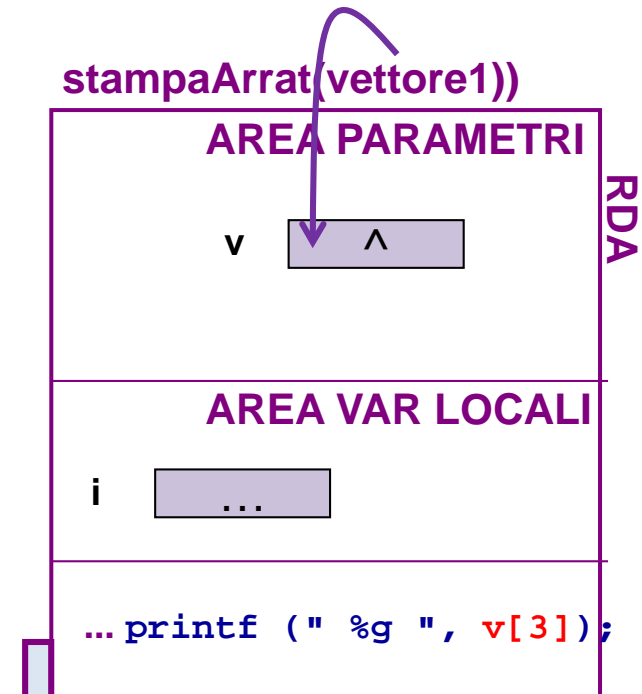
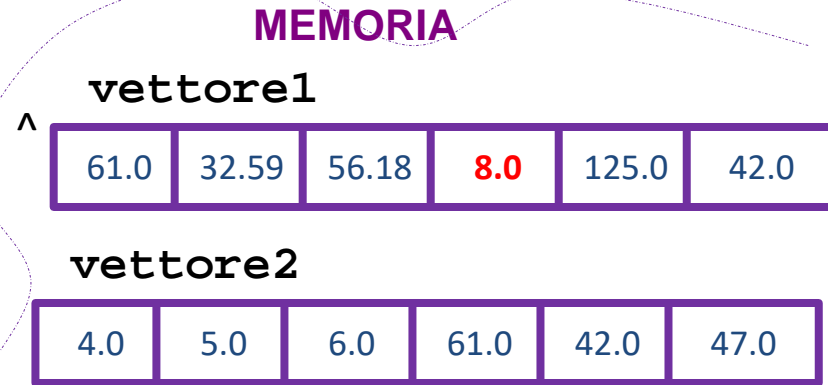
Programma per il Prodotto scalare mediante funzioni su array (2/3)

chiamata

```
stampaArray(vettore1);
```

definizione

```
void stampaArray (double v[N]) {  
    int i;          /* contatore*/  
  
    for (i=0; i<N; i++) {  
        printf (" %g ", v[i]);  
    }  
    return;  
}
```



esecuzione simulata pls



Programma per il Prodotto scalare mediante funzioni su array (2/3)

chiamata

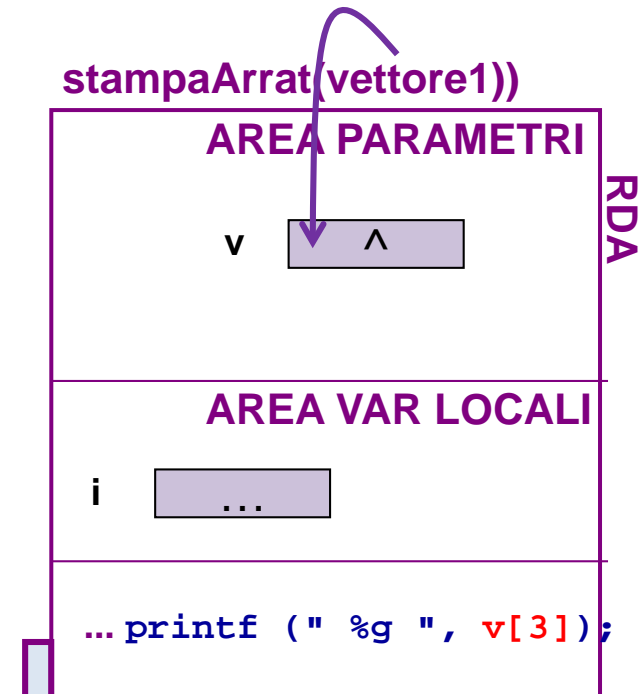
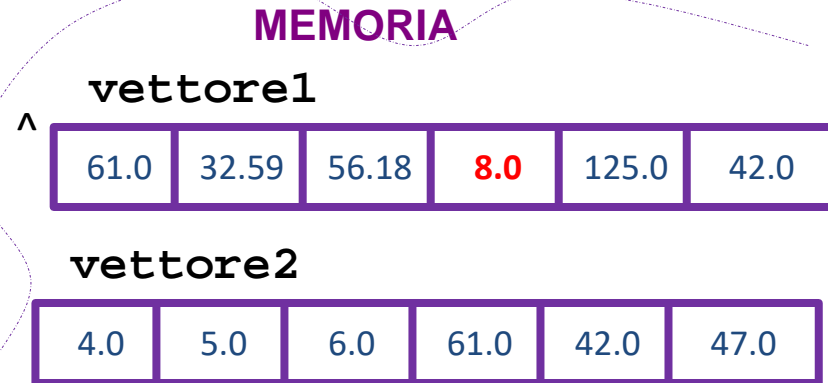
```
stampaArray(vettore1);
```

definizione

```
void stampaArray (double v[N]) {  
    int i;          /* contatore*/  
  
    for (i=0; i<N; i++) {  
        printf (" %g ", v[i]);  
    }  
    return;  
}
```

il passaggio del parametro copia il contenuto della variabile **vettore1** (l'indirizzo del primo elemento dell'array **vettore1**), nel parametro formale **v**;

poi, durante la chiamata, **v[i]** è a tutti gli effetti un accesso alla variabili **vettore1[i]**



Programma per il Prodotto scalare mediante funzioni su array (3/3)

chiamata

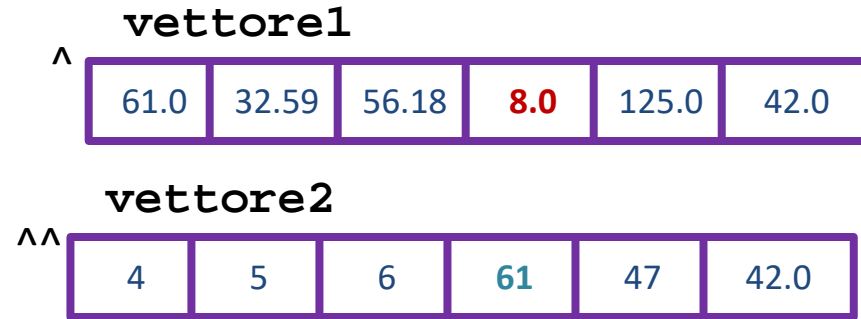
`pScalareFun(vettore1, vettore2)`

```
double pScalareFun (double v1[N],
                    double v2[N]) {
    int i;
    double p = 0.0;

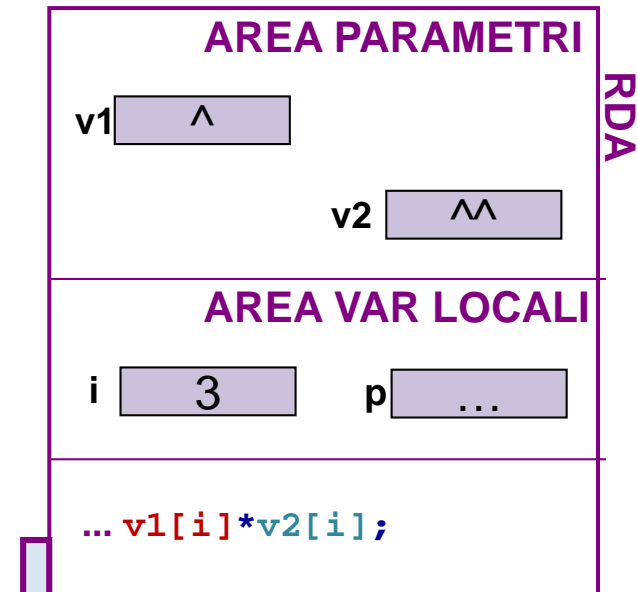
    for (i=0; i<N; i++)
        p += v1[i]*v2[i];
    return p;
}
```

il ...

MEMORIA



`pScalareFun(vettore1, vettore2)`



esecuzione simulata pls



Programma per il Prodotto scalare mediante funzioni su array (3/3)

chiamata

`pScalareFun(vettore1, vettore2)`

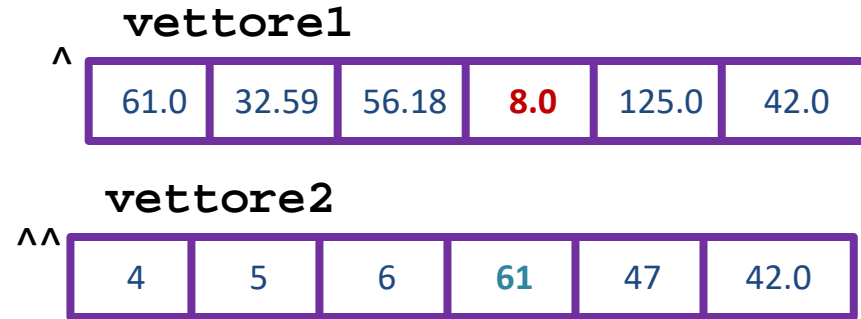
```
double pScalareFun (double v1[N],
                    double v2[N]) {
    int i;
    double p = 0.0;

    for (i=0; i<N; i++)
        p += v1[i]*v2[i];
    return p;
}
```

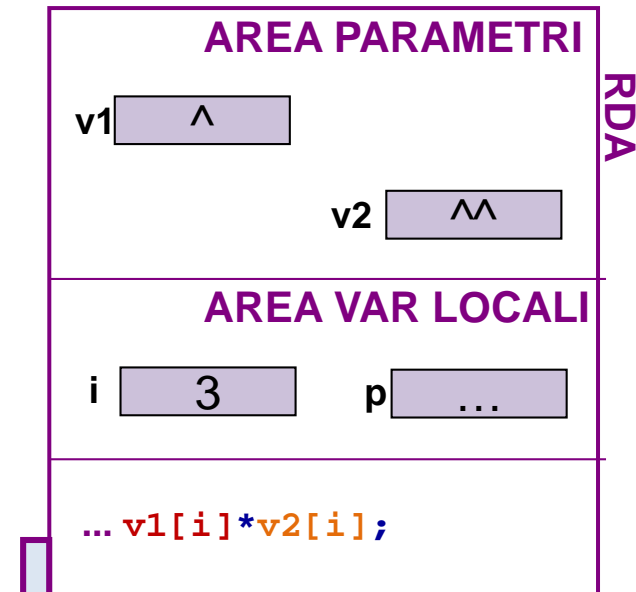
il passaggio del parametro copia il contenuto della variabile **vettore1** (l'indirizzo del primo elemento dell'array vettore1) nel parametro formale **v1**;
idem per v2/vettore2

poi, durante la chiamata,
v1[i] è a tutti gli effetti un accesso alla variabili **vettore1[i]** e **v2[i]** è a tutti gli effetti un accesso alla variabili **vettore2[i]**

MEMORIA



`pScalareFun(vettore1, vettore2)`



NegoziFun

Ecco +/- come diventa il programma sui negozi, se usiamo delle funzioni

```
#define N 6                                /* siamo passati da 5 a 6 ... */  
#define RATIO 1/3
```

```
/* prototipi 😊 */
```

```
int main ()  
{ double guadagni[N], media;  
  int i;  
  
  leggiArray(guadagni);  
  media = mediaArray(guadagni);  
  
  printf(" (media=%g) ecco i negozi ...:\n", media);  
  for (i=0; i<N; i++)  
    if (guadagni[i] < media*RATIO)  
      printf(" - negozio %d con ... %g\n", i, guadagni[i]);  
  printf("FINE\n");  
}
```

NegoziFun

Ecco +/- come diventa il programma sui negozi, se usiamo delle funzioni

```
#define N 6                                /* siamo passati da 5 a 6 ... */
#define RATIO 1/3

/* prototipi delle funzioni usate */
void leggiArray (double a[N]);
/* void stampaArray (double a[N]); non sarebbe male usarla*/
double mediaArray(double a[N]);

int main ()
{ double guadagni[N], media;
  int i;

  leggiArray(guadagni);
  media = mediaArray(guadagni);

  printf(" (media=%g) ecco i negozi ...:\n", media);
  for (i=0; i<N; i++)
    if (guadagni[i] < media*RATIO)
      printf(" - negozio %d con ... %g\n", i, guadagni[i]);
  printf("FINE\n");
}
```

mediaArray() ?



NegoziFun - funzione mediaArray()

```
/* funzione che calcola la media dei valori di un array di double (la  
cui dimensione è data da una costante top level - aka "globale") */
```

```
/* algoritmo fatto prima ... */
```

```
double mediaArray(double a[N]) {  
    double somma = 0.0;    /* accumulatore */  
    int i;                /* contatore */  
  
    for (i=0; i<N; i++)  
        somma += a[i];  
  
    /* potremmo anche assumere che la costante  
    sia non nulla ma andiamo sul sicuro: */  
  
    if (N!=0)  
        return(somma/N);  
    else return 0;  
}
```

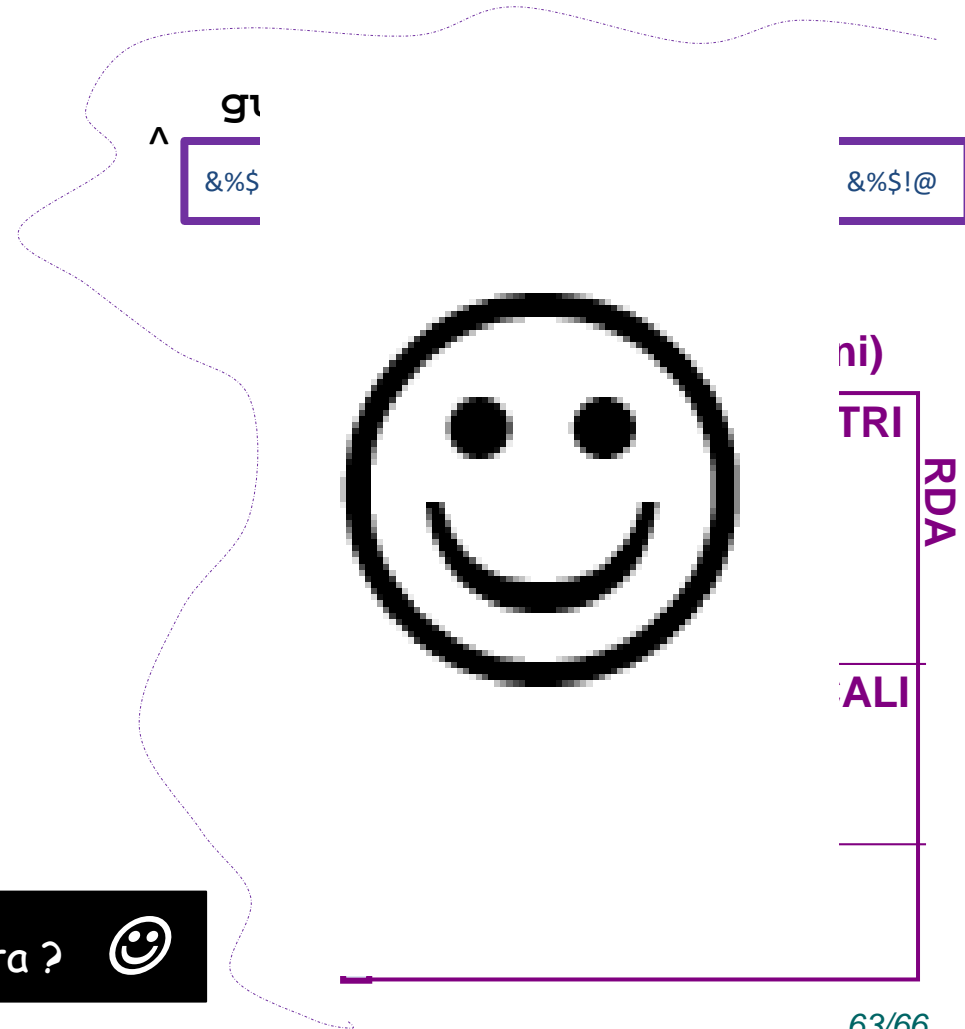
leggiArray() ?



NegoziFun - funzione leggiArray()

```
void leggiArray (double a[N]) {  
    int i;  
    for (i=0; i<N; i++) {  
        printf ("caro/a utente, dammi il valore [%d]: ", i);  
        scanf("%lf", &a[i]);  
    }  
    return;  
}
```

la chiamata è
`leggiArray(guadagni);`



come si svolge la chiamata ?

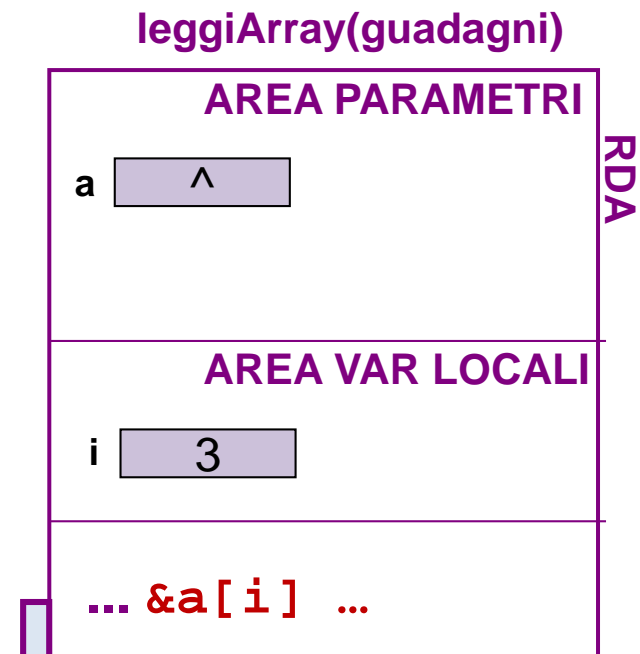
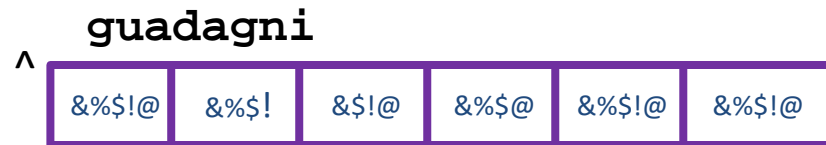


NegoziFun - RDA chiamata leggiArray() - 1/3

```
void leggiArray (double a[N]) {  
    int i;  
    for (i=0; i<N; i++) {  
        printf ("caro/a utente, dammi il valore [%d]: ", i);  
        scanf("%lf", &a[i]);  
    }  
    return;  
}
```

chiamata
`leggiArray(guadagni);`

così guadagni viene modificato ...



NegoziFun - RDA chiamata leggiArray() - 2/3

```
void leggiArray (double a[N]) {  
    int i;  
    for (i=0; i<N; i++) {  
        printf ("caro/a utente, dammi il valore [%d]: ", i);  
        scanf("%lf", &a[i]);  
    }  
    return;  
}
```

chiamata

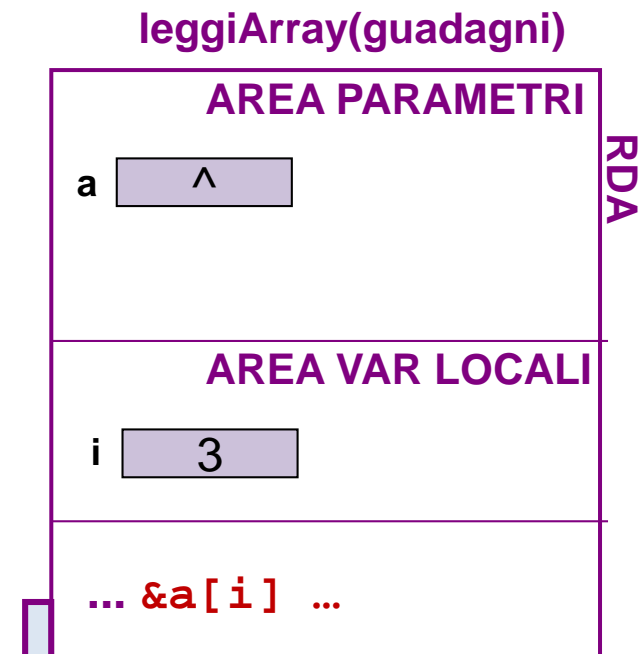
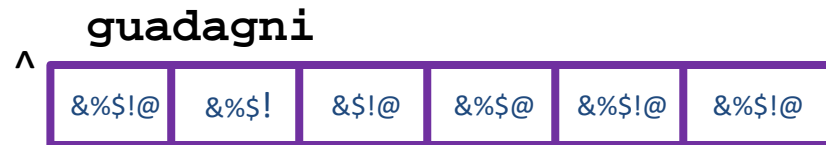
`leggiArray(guadagni);`

durante la chiamata,

`a[i]` è a tutti gli effetti un accesso alla variabile `guadagni[i]`

`&a[i]` è a tutti gli effetti equivalente a `&guadagni[i]`

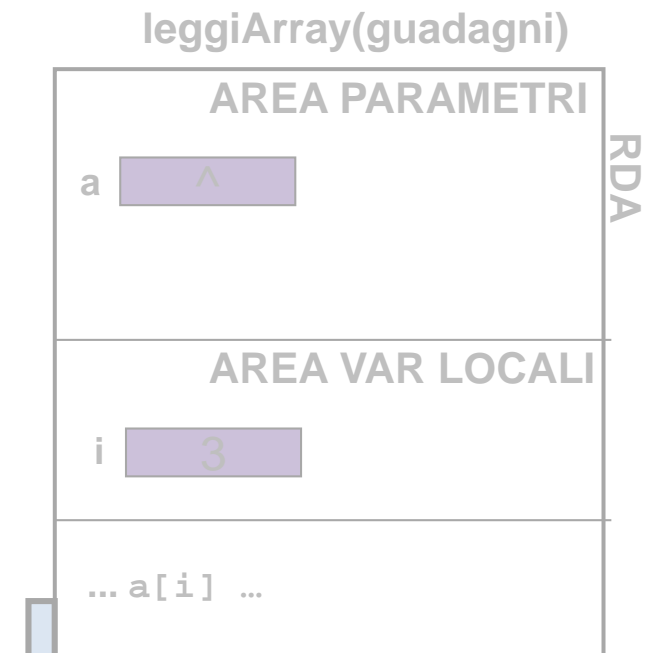
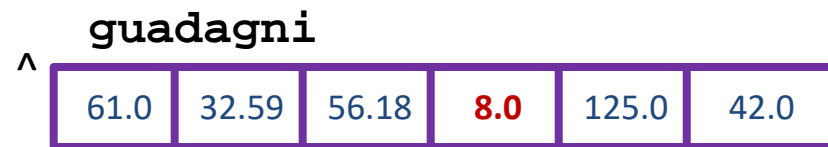
così guadagni viene modificato ...



NegoziFun - RDA chiamata leggiArray() - 3/3

```
void leggiArray (double a[N]) {  
    int i;  
    for (i=0; i<N; i++) {  
        printf ("caro/a utente, dammi il valore [%d]: ", i);  
        scanf("%lf", &a[i]);  
    }  
    return;  
}
```

dopo il termine della chiamata



Vedi Approfondimenti
per uno spunto di riflessione su quello che succede di
diverso ad un parametro array rispetto a quel che
succede ad un parametro "variabile semplice"

Una breve intro sui file TESTUALI

LEZIONE ONLINE
(vedi pagina lezioni)

un file in generale, è

- una sequenza di dati "codificati"
- terminata da EOF
- la cui gestione in C è effettuata mediante

1) **APERTURA** (fopen)

2) **ASSOCIAZIONE** del file ad una variabile *handle* (di tipo **FILE ***)

3) **USO** mediante funzioni applicate alla variabile handle

i passi 1) e 2) ...

```
FILE * f;
```

```
f = fopen(NOMEFILE, "modo")
```

dove

- **NOMEFILE** è una stringa e
- **modo** è
 - r, ... (lettura) ... altro ...
 - w, ... (scrittura) ... altro ...
 - a, ... (append) ... altro ...

buffer file: cursore posizionato lungo il file;
quando viene richiesta una operazione di
lettura o scrittura (USO - fase 3 -),
questa avviene nel punto mirato dal
buffer file (bfile nella slide successiva)

...

EOF

I file di testo hanno una organizzazione particolare (che li rende *editabili* a mano con un editor)
 stdin e stdout sono stream testuali (cioè sequenze trattate come file testuali)

- i **dati** sono **caratteri**
- i modi sono **r/w/a**
- c'è EOF ma nel file possono esserci anche caratteri particolari EOLN (end-of-line)
- gli EOLN ('\n') permettono di dare ai caratteri una **organizzazione in righe**
- le funzioni di uso sono quelle (**lettura/scrittura**) a gestione "**formattata**": fprintf, fscanf

ch p ... a...r...

fin 

i 10

d 7.118

string parola0

```
double d; int i; char ch;
FILE * fin = fopen("dati.txt", "r"); 1
... inizialmente il bfile è sulla 'p'
fscanf (fin, "%c", &ch); 2
/*ora ch contiene 'p' e il bufferfile è sulla 'à */
rewind(fin) /* ora torna sulla 'p' */ 1
fscanf(fin, "%s", string); 3
fscanf(fin, "%d", &i); /* legge il 10 */ 4
fscanf(fin, "%lf", &d); /* legge il 7.118 */ 5
```

I numeri in questo tipo di riquadro simboleggiano la situazione del buffer file dopo l'esecuzione dell'istruzione



Tecniche della Programmazione, lez. 9

- Approfondimenti

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

i

0

prima iterazione (i=0)

guadagni

61.0

&%\$!!@

&%\$!!@

&%\$!!@

&%\$!!@

&%\$!!@

poi i diventa 1

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

i 1

seconda iterazione (i=1)

guadagni

61.0	32.59	&%\$!!@	&%\$!!@	&%\$!!@	&%\$!!@
------	-------	---------	---------	---------	---------

poi i diventa 2

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

i 2

terza iterazione (i=2)

guadagni

61.0	32.59	56.18	&%\$!!@	&%\$!!@	&%\$!!@
------	-------	-------	---------	---------	---------

poi i diventa 3

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

i 3

quarta iterazione (i=3)

supponendo N=6 e che i guadagni dati in input siano 61, 32,59, 56,18, 8, 125, 42

guadagni

61.0	32.59	56.18	8.0	&%\$!!@	&%\$!!@
------	-------	-------	-----	---------	---------

poi i diventa 4

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

i

4

quinta iterazione (i=4)

61.0

32.59

56.18

8.0

125.0

&%\$!!@

poi i diventa 5

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

Torniamo ai Negozi

Algoritmo

```
0) costante N, costante RATIO = 1/3
   var double guadagni[N], media,
       contatore i
1) leggere i guadagni, memorizzandoli nell'array
for (i=0; i<N; i++) {
    printf richiesta guadagno i
    scanf("%lf", &guadagni[i]);
}
```

i

5

sesta iterazione (i=5)

61.0

32.59

56.18

8.0

125.0

42.0

poi i diventa 6 e la condizione di ripetizione fallisce e il ciclo termina

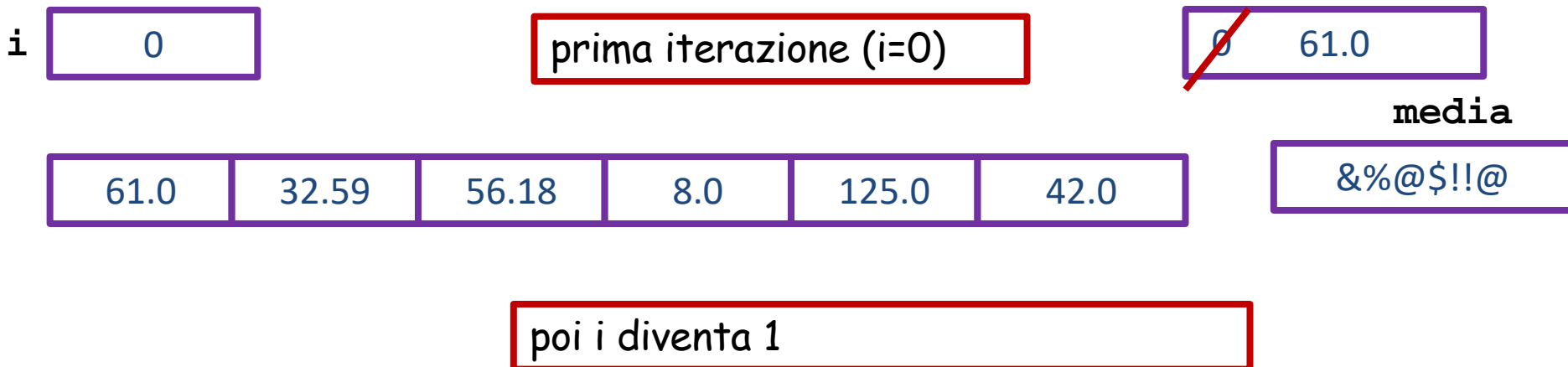
supponendo N=6 e che i guadagni dati in input siano 61, 32,59, 56,18, 8, 125, 42

Torniamo ai Negozi (ora la media)

Algoritmo

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma

```
somma=0.0;  
for (i=0; i<N; i++)  
    somma = somma + guadagni[i];
```



Torniamo ai Negozi (ora la media)

Algoritmo

supponendo
N=6 e che i
guadagni dati in
input siano 61,
32,59, 56,18, 8,
125, 42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma

```
somma=0.0;  
for (i=0; i<N; i++)  
    somma = somma + guadagni[i];
```

i

1

seconda iterazione (i=1)

somma

~~61.0~~ 93.59

media

&%@\$!!@

61.0

32.59

56.18

8.0

125.0

42.0

poi i diventa 2

fare terza e
quarta iterazione,
se presenti ...



Torniamo ai Negozi (ora la media)

Algoritmo

supponendo
N=6 e che i
guadagni dati
in input siano
61, 32,59,
56,18, 8,
125, 42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma

```
somma=0.0;  
for (i=0; i<N; i++)  
    somma = somma + guadagni[i];
```

i

4

quinta iterazione (i=4)

~~157.77~~ 282.77

somma

media

61.0

32.59

56.18

8.0

125.0

42.0

&%@\$!!@

poi i diventa 5

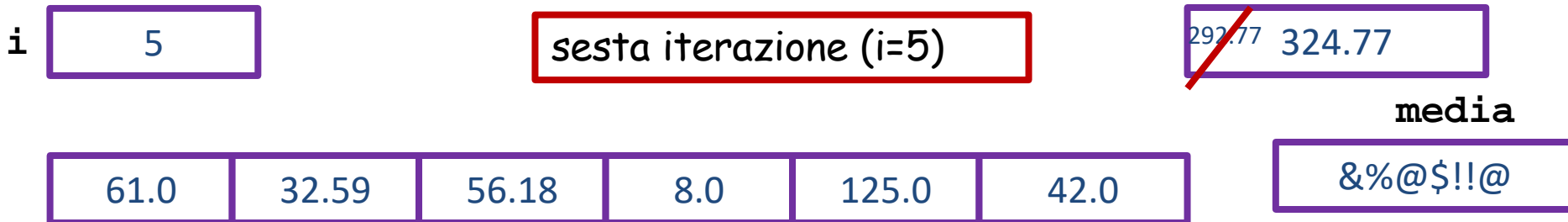
Torniamo ai Negozi (ora la media)

Algoritmo

supponendo
N=6 e che i guadagni
dati in input siano 61,
32,59, 56,18, 8, 125,
42

- 0) costante N, costante $RATIO = 1/3$
var double guadagni[N], media, somma;
contatore i
- 1) leggere i guadagni, memorizzandoli nell'array
- 2) calcolare la media
 - 2.1) accumulare gli elementi dell'array in somma

```
somma=0.0;  
for (i=0; i<N; i++)  
    somma = somma + guadagni[i];
```



poi i diventa 6 e fine ciclo

Dichiarazione di array, con inizializzazione

costante N=6

```
double arr[N]={4,5,6,61,42,47}
```

viene allocato un array di N elementi double;

arr[0] viene assegnato a 4 ... arr[5] viene assegnato a 47

```
double vett[]={32.59, 116, 50.08, 56.18, 47, 42};
```

viene allocato un array di tanti elementi double, quanti sono i valori specificati tra le graffe.

vett[0] viene assegnato a 32.59 ... vett[3] viene assegnato a 56.18

vett[5] viene assegnato a 42.0 ...  vett[6] viene assegnato a un bel niente!
(non è allocato)

Sempre: i valori devono essere del tipo dichiarato per l'array, o convertibili senza problemi a quel tipo.

(Conversione in assegnazione)

Scusi, ...

Lo abbiamo visto in un disegno prima ... sono in memoria: cosa stampa questo programma?

```
#include <stdio.h>
#define N 8
```

```
int main () {
    int arr[N] = {5618, 42, 180, 61, 47, 125, 116, 3783}, i;
```

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

```
        printf("l'elemento arr[%d] è di %d byte, ha
            indirizzo %p e valore %d\n", i,
            sizeof(arr[i]), &arr[i], arr[i]);
```

```
    printf("FINE");
```

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

```
        printf("l'elemento arr[%d] è di %d byte, ha
            indirizzo %p e valore %d\n", i,
            sizeof(arr[i]), &arr[i], arr[i]);
```

```
    printf("FINE");
```

```
    return 0;
```

```
}
```

```
l'elemento arr[0] e' di 4 byte, ha indirizzo 0028FF10 e valore 5618
l'elemento arr[1] e' di 4 byte, ha indirizzo 0028FF14 e valore 42
l'elemento arr[2] e' di 4 byte, ha indirizzo 0028FF18 e valore 180
l'elemento arr[3] e' di 4 byte, ha indirizzo 0028FF1C e valore 61
l'elemento arr[4] e' di 4 byte, ha indirizzo 0028FF20 e valore 47
l'elemento arr[5] e' di 4 byte, ha indirizzo 0028FF24 e valore 125
l'elemento arr[6] e' di 4 byte, ha indirizzo 0028FF28 e valore 116
l'elemento arr[7] e' di 4 byte, ha indirizzo 0028FF2C e valore 3783
e adesso HORROR!
```

Qui c'e' la stampa prodotta dal primo for. Verificarla ...

Cosa stampa il secondo for?
E perche' e' horror allo stato puro?

scusi ... ehm ...

Lo abbiamo visto in un disegno prima ... sono in memoria: cosa stampa questo programma?

```
#include <stdio.h>
#define N 8
```

```
int main () {
    int arr[N] = {5618, 42, 180, 61, 47, 125, 116, 3783}, i;

    for (i=0; i<N; i++)
        printf("l'elemento arr[%d] è di %d byte, ha
            indirizzo %p e valore %d\n",
                sizeof(arr[i]), &arr[i], arr[i]);

    printf("\n\n e adesso HORROR!\n\n");
    for (i=0; i<=N+1; i++)
        printf("arr[%d] = %d\n", i, arr[i]);

    printf("FINE\n");
    return 0;
}
```

what horror?

e adesso HORROR!

```
arr[0] = 5618
arr[1] = 42
arr[2] = 180
arr[3] = 61
arr[4] = 47
arr[5] = 125
arr[6] = 116
arr[7] = 3783
arr[8] = 53
arr[9] = 2
FINE
```

scusi ... ehm ... no excuses

Lo abbiamo visto in un disegno prima ... sono in memoria: cosa stampa questo programma?

```
#include <stdio.h>
#define N 8

int main () {
    int arr[N] = {5618, 42, 180, 61, 47, 125, 116, 3783}, i;

    for (i=0; i<N; i++)
        printf("l'elemento arr[%d] è di %d byte ha
            indirizzo %p e valore %d\n",
                sizeof(arr[i]), &arr[i], arr[i]);

    printf("\n\n e adesso HORROR!\n\n");
    for (i=0; i<=N+1; i++)
        printf("arr[%d] = %d\n", i, arr[i]);

    printf("FINE\n");
    return 0;
}
```



siamo solo noi responsabili
che il codice non visiti aree
di memoria non allocate ...

```
arr[8] = 53
arr[9] = 2
FINE
```


No Excuses: Stare nei limiti


$$\lim_{x \rightarrow -1^+} \frac{2x - x^2}{x + 1} = -\infty$$

è facile scorrere un array senza fermarsi al momento giusto

Quando si cerca di accedere ad una locazione successiva all'ultima allocata per un array, il C non C ferma ... e quindi potremmo avere risultati molesti:

- visitare aree di memoria libera, non dedicata ad altre variabili (ma nemmeno alle nostre)
- visitare la memoria allocata per altre variabili ...
- visitare aree di memoria riservate

Nel primo caso potrebbe non succedere nulla, per ora ... ma se quella memoria venisse allocata per altre variabili? Tipicamente, se non siamo molto fortunati,  avremo errori inattesi e inquietanti nel funzionamento del programma

Nel secondo caso disturbiamo e veniamo disturbati: anche qui potremmo avere errori  inattesi, inspiegabili, inquietanti

Nel terzo caso potremmo semplicemente assistere al blocco dell'esecuzione ("segmentation fault")

Ancora su funzioni:

nomi uguali in scope diversi

in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

```
int main() {  
    int result, primo, secondo;  
    ...  
    scanf("%d %d", &primo, &secondo);  
    ...  
    printf("mcd=%d\n", mcd(primo, secondo));  
    return 0;  
}
```

scope
delle
funzione
chiamante

```
int mcd (int primo, int secondo) {  
    int result = 0;  
    ...  
    return result;  
}
```

Vedi Approfondimenti

scope
delle
funzione
chiamata

Ancora su funzioni:

nomi uguali in scope diversi

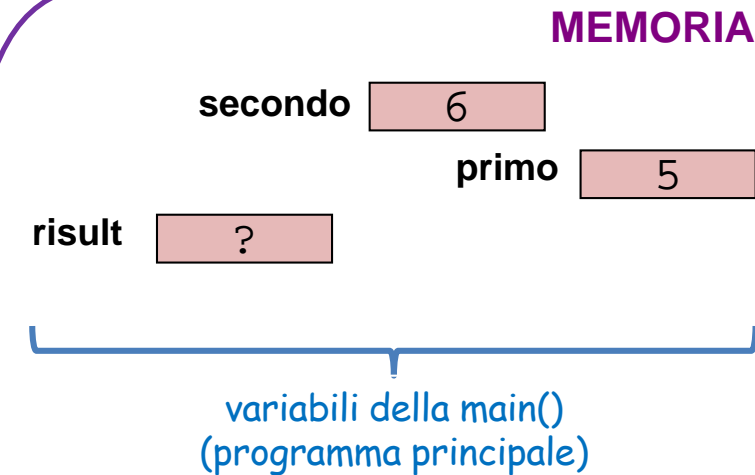
in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

concretamente, che significa?

```
int main() {
    int result, primo, secondo,;
    ...
    scanf("%d %d", &primo, &secondo);
    ...
    printf("mcd=%d\n", mcd(primo, secondo));
    return 0;
}

int mcd (int primo, int secondo) {
    int result=0;
    ...
    return result;
}
```



e le variabili (e parametri) della funzione mcd esistono solo mentre la funzione è in azione ...

Ancora su funzioni:

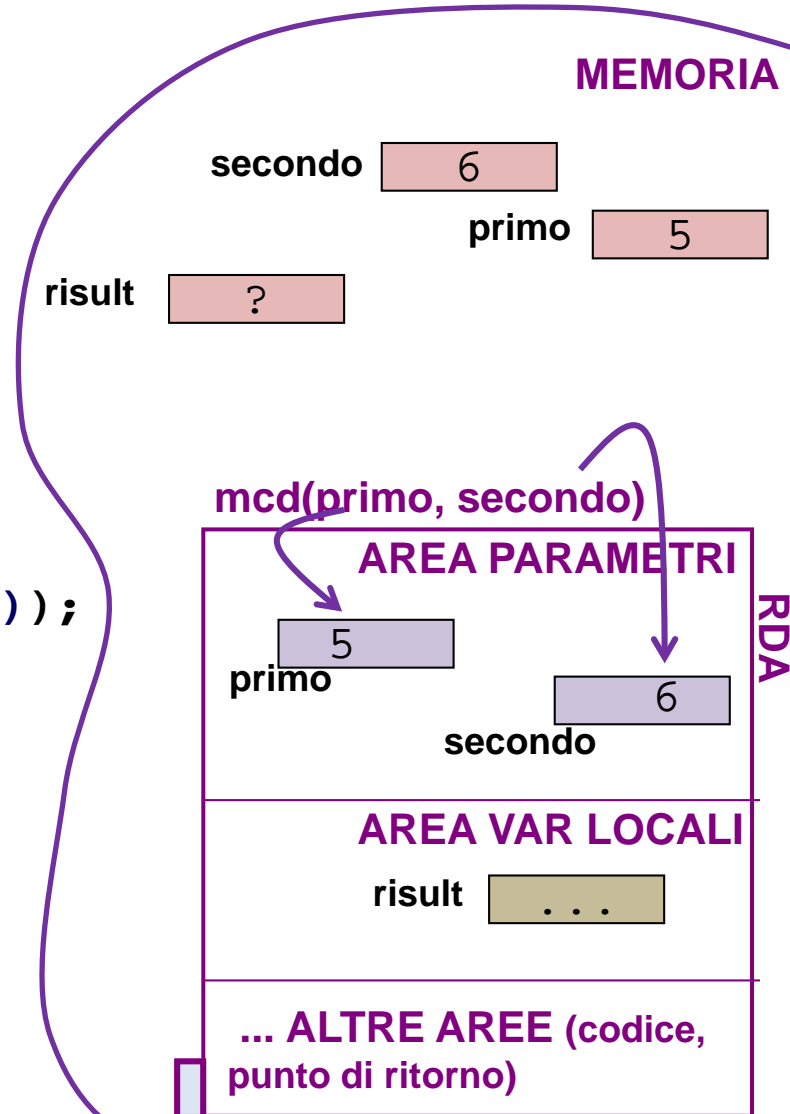
nomi uguali in scope diversi

in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

```
int main() {  
    int result, primo, secondo,;  
    ...  
    scanf("%d %d", &primo, &secondo);  
    ...  
    printf("mcd=%d\n", mcd(primo, secondo));  
    return 0;  
}
```

```
int mcd (int primo, int secondo) {  
    int result=0;  
    ...  
    return result;  
}
```



Ancora su funzioni:

nomi uguali in scope diversi

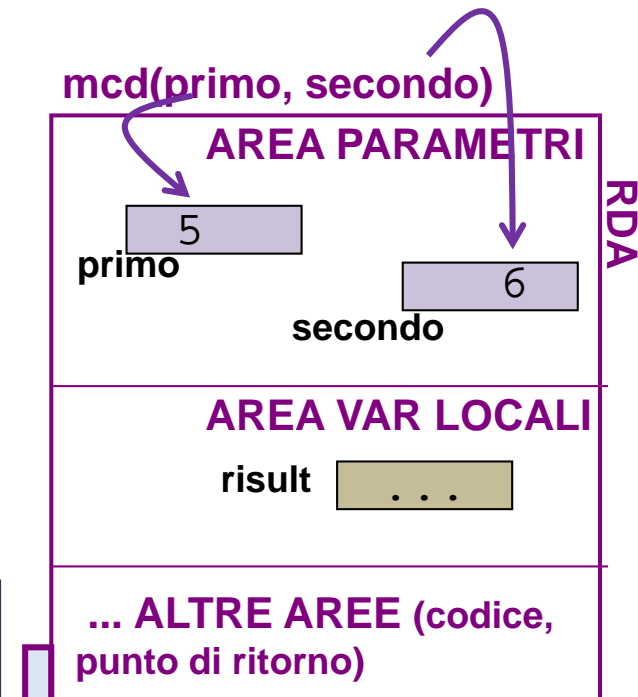
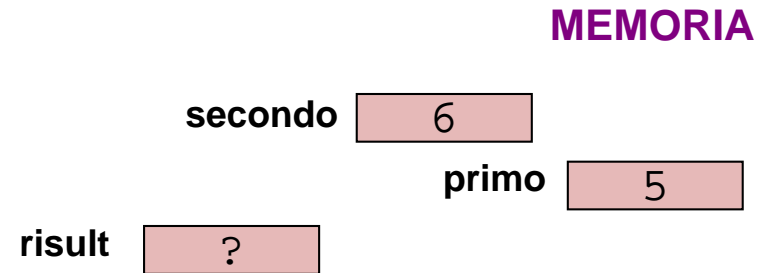
in diversi scope, possono esistere contemporaneamente variabili con il medesimo identificatore, senza darsi fastidio ...

non è consigliabile, ma è possibile

```
int main() {
    int result, primo, secondo;;
    ...
    scanf("%d %d", &primo, &secondo);
    ...
    printf("mcd=%d\n", mcd(primo, secondo));
    return 0;
}

int mcd (int primo, int secondo) {
    int result=0;
    ...
    return result;
}
```

sono diverse entità, diverse locazioni, perché esistono in diversi scope



Ancora su funzioni: nomi uguali in scope diversi (recap)

Abbiamo detto che il RDA fornisce un ambiente protetto per l'esecuzione di una chiamata di funzione.

Parametri formali e variabili locali della funzione esistono fisicamente solo durante l'attivazione (cioè durante l'esistenza del RDA), e sono visibili (usabili) solo nell'ambiente del RDA.

Si dice che il loro SCOPE è il blocco della funzione.

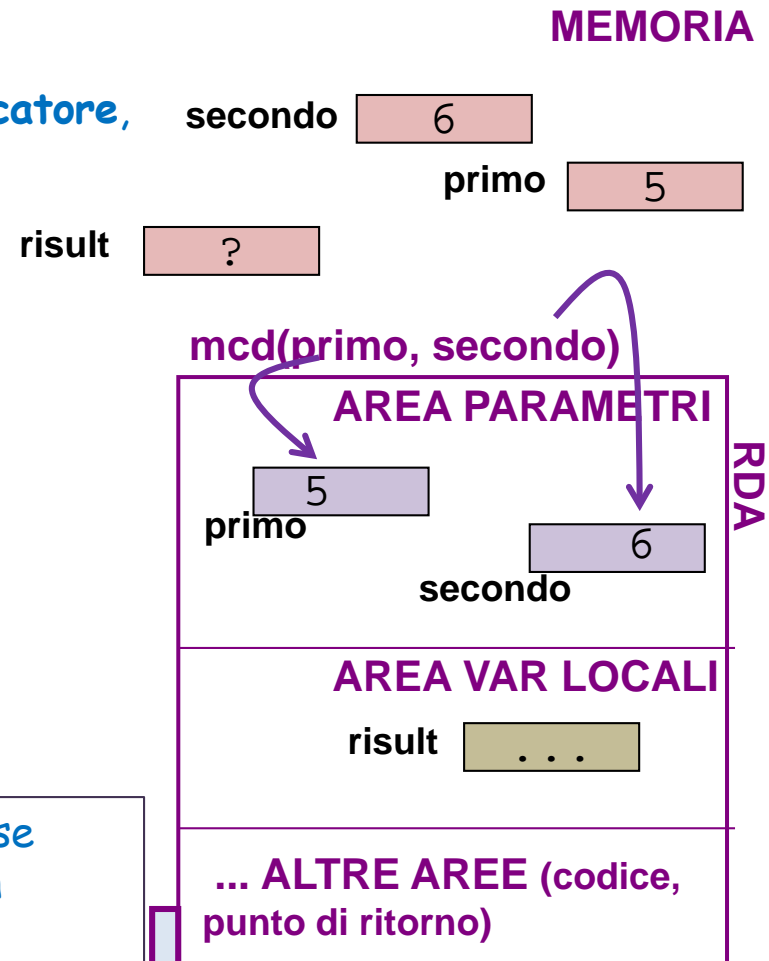
Ne consegue che, **in diversi scope**, possono esistere contemporaneamente **variabili con il medesimo identificatore**, senza darsi fastidio ...

non è consigliabile,
ma è possibile

```
int main() {  
    int result, primo, secondo;  
    ...  
    scanf("%d %d", &primo, &secondo);  
    ...  
    printf("mcd=%d\n", mcd(primo, secondo));  
    return 0;  
}
```

```
int mcd (int primo, int secondo) {  
    int result=0;  
    ...  
    return result;  
}
```

sono diverse variabili, diverse
locazioni, perché esistono in
diversi scope



Ancora su funzioni: modifica parametri attuali (tsk)



Un esempio dice tutto:

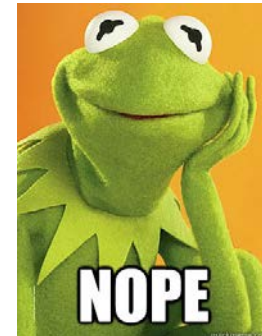
```
void adessoModifico (int num) {  
    num = num+1;  
return;  
}  
  
int main () {  
    int n=47;  
  
    printf("valore di n prima: %d\n", n);  
    adessoModifico(n);  
    printf("valore di n dopo: %d\n", n);  
  
    printf("FINE\n");  
    return 0;  
}
```

Perche' no? Vedi di seguito

n viene "passata" alla funzione ...

QUINDI
potrebbe venire in mente che l'istruzione
num=num+1

possa mettere 48 dentro a n



perché no?



Quando viene chiamata adessoModifico(),
viene creato il RDA
e viene "passato" il parametro

```
void adessoModifico (int num) {  
    num = num+1;  
return;  
}  
  
int main () {  
    int n=47;  
  
    printf("valore di n prima: %d\n", n);  
    adessoModifico(n);  
    printf("valore di n dopo: %d\n", n);  
  
    printf("FINE\n");  
return 0;  
}
```

MEMORIA

n 47

adessoModifico(n)

AREA PARAMETRI

num 47

RDA

AREA VAR LOCALI

... ALTRE AREE (codice,
punto di ritorno)

perché no? perché ecco quel che succede.



Quando viene chiamata adessoModifico(),
viene creato il RDA
e viene "passato" il parametro

Quando viene eseguito num=num+1 nella funzione,
num viene modificato
ma n non viene modificata ...

```
void adessoModifico (int num) {  
    num = num+1;  
return;  
}  
  
int main () {  
    int n=47;  
  
    printf("valore di n prima: %d\n", n);  
    adessoModifico(n);  
    printf("valore di n dopo: %d\n", n);  
  
    printf("FINE\n");  
return 0;  
}
```

MEMORIA

n 47

adessoModifico(n)

AREA PARAMETRI

num ~~47 48~~

RDA

AREA VAR LOCALI

... ALTRE AREE (codice,
punto di ritorno)

perché il passaggio del parametro è "per valore"

Un esempio dice tutto:

```
void adessoModifico (int num) {  
    num = num+1;  
return;  
}
```

```
int main () {  
    int n=47;
```

```
valore di n prima: 47  
valore di n dopo: 47  
FINE
```

```
printf("valore di n prima: %d\n", n);  
adessoModifico(n);  
printf("valore di n dopo: %d\n", n);
```

```
printf("FINE\n");  
return 0;  
}
```

PASSAGGIO PER VALORE:

il parametro attuale è un'espressione che viene valutata; il suo valore viene copiato nella locazione del parametro formale (num dentro al RDA);

l'esecuzione di `num = num+1` comporta una modifica di `num`, MA NON di `n`, che da dentro l'RDA non è nemmeno visibile



MEMORIA

n 47

adessoModifico(n)



NegoziFun - RDA chiamata leggiArray() - 4/3 ... (spunto di riflessione - poi dovremo parlarne ancora ...)

chiamata

```
leggiArray(guadagni);
```

durante la chiamata,

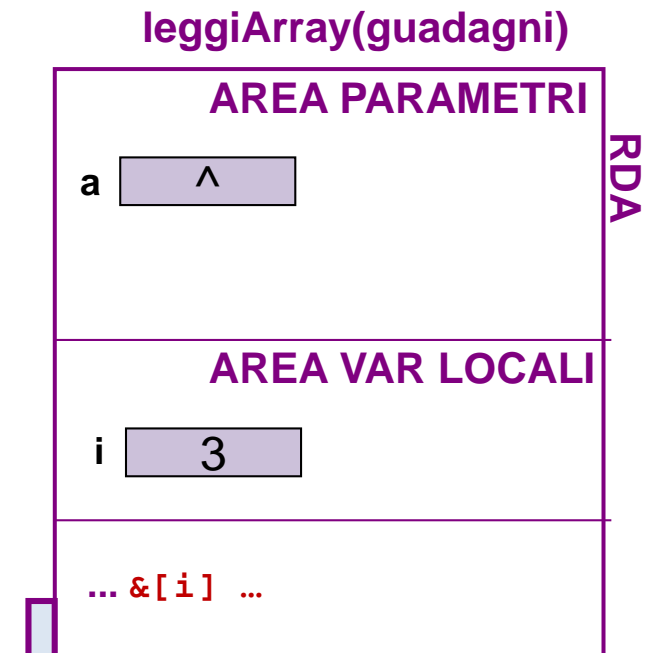
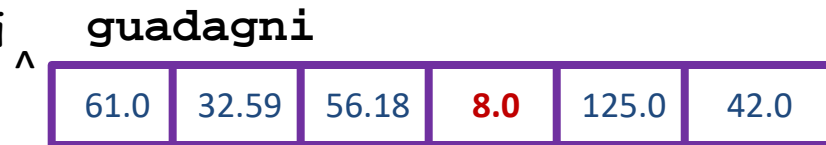
$a[i]$ è a tutti gli effetti un accesso alla variabile $guadagni[i]$

$\&a[i]$ è a tutti gli effetti equivalente a $\&guadagni[i]$

l'array viene modificato, il che è strano per un parametro attuale.

Ma comprensibile, perché in questo caso (array) il parametro attuale è in realtà l'indirizzo del primo elemento dell'array `guadagni`;

la COPIA del parametro attuale nel parametro formale non è una copia dell'intero array, ma solo dell'indirizzo.



Tecniche della Programmazione, lez. 9

- Esercizi

Prodotto scalare tra vettori

```
#define N 6
int main () {
    double vettore1[]={32.59, 116, 50.08, 56.18, 47, 42};
    double vettore2[N]={4,5,6,61,42,47},
    pScalare;          /* ... */
    int i;            /* contatore */

    /* stampa dei vettori */
    printf ("caro/a utente, primo vettore: (");
    for (i=0; i<N; i++) {
        printf (" %g ", vettore1[i]);
    }
    printf (")\n");

    printf ("caro/a utente, secondo vettore: (");
    ...
    /* accumulazione in pScalare dei prodotti tra le componenti corrispondenti dei vettori */
    pScalare = 0.0;
    for (i=0; i<N; i++)
        pScalare += vettore1[i]*vettore2[i];

    printf("ok, prodotto scalare tra i due = %g\n", pScalare);
    printf("FINE\n");
}
```