Esercitazione 09

2026/10/26 – Funzioni

L. Alessandrini

Credits: A. Montenegro per gli esercizi

Funzioni

Recap – Interazione con Array

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- Un array può essere passato in input

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- Un array può essere passato in input
- Questo può avvenire senza specificare il numero di elementi

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- Un array può essere passato in input
- Questo può avvenire senza specificare il numero di elementi
- Il compilatore interpreta `int v[]` come `int * v`

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- Un array può essere passato in input
- Questo può avvenire senza specificare il numero di elementi
- Il compilatore interpreta `int v[]` come `int * v`
- Di fatto viene passato l'indirizzo della prima cella di memoria dell'array

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- La dimensione dell'array non è nota e quindi va passata come argomento se serve!

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- La dimensione dell'array non è nota e quindi va passata come argomento se serve!
- (A meno che non sia definita come costante)

Funzioni

Interazioni con Array

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

- La dimensione dell'array non è nota e quindi va passata come argomento se serve!
- (A meno che non sia definita come costante)

Nelle consegne (anche in esame), potrebbe essere omessa la specifica di alcuni parametri che sapete debbano essere passati (come la dimensione)

Funzioni

Interazioni con Array

- Dato che si agisce su indirizza di memoria, si può modificare un array in una funzione
- Quando l’array è passato dal chiamante e viene modificato in una funzione, il chiamante vede le modifiche

Funzioni

Interazioni con Array

Come sarà il prototipo della funzione di prima?

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

Funzioni

Interazioni con Array

Come sarà il prototipo della funzione di prima?

```
void stampaArray(int v[], int dim){
    int i;
    for(i=0; i<dim ; i++){
        printf("%d", v[i]);
    }
}
```

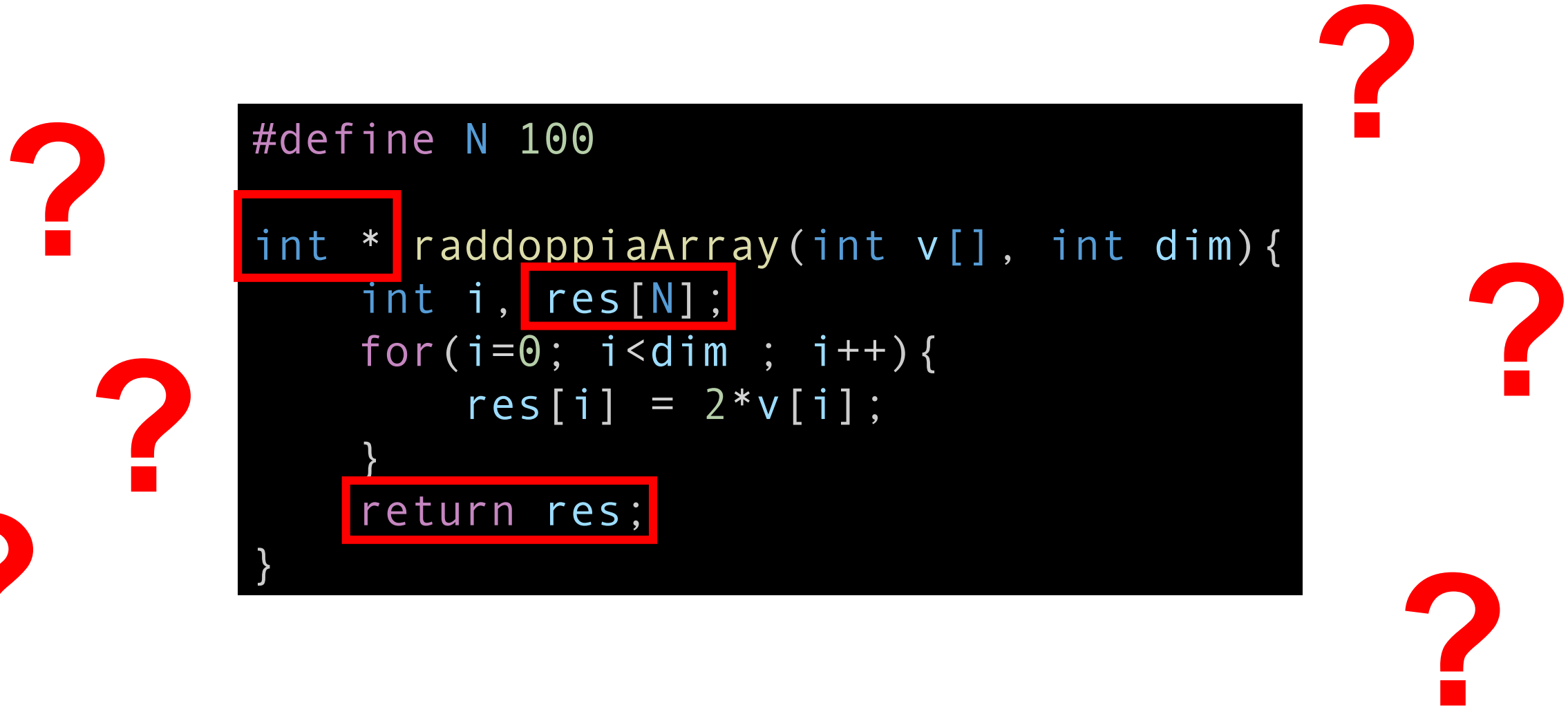
```
void stampaArray(int [], int);
```

Funzioni

E Se volessi restituire un array?

Funzioni

E Se volessi restituire un array?

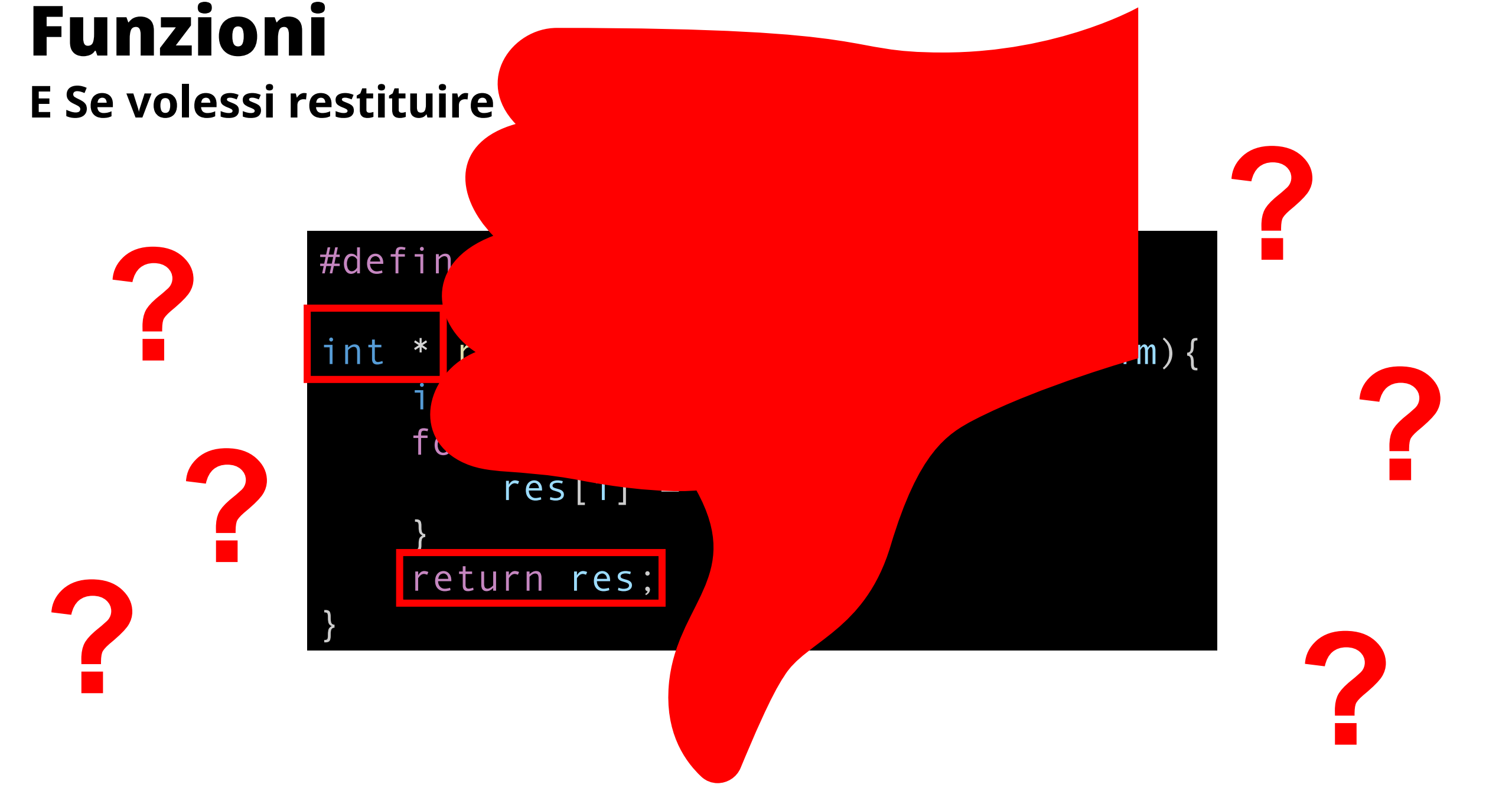


```c
#define N 100

int * raddoppiaArray(int v[], int dim){
    int i, res[N];
    for(i=0; i<dim ; i++){
        res[i] = 2*v[i];
    }
    return res;
}
```

Funzioni

E Se volessi restituire



Funzioni

E Se volessi restituire un array?

- Tutto quello che è **locale alla funzione** viene **distrutto** quando la funzione termina

Funzioni

E Se volessi restituire un array?

- Tutto quello che è **locale alla funzione** viene **distrutto** quando la funzione termina
- Gli array sono **aree di memoria che vengono de-allocate** quando viene invocato il “`return`”

Funzioni

E Se volessi restituire un array?

- Tutto quello che è **locale alla funzione** viene **distrutto** quando la funzione termina
- Gli array sono **aree di memoria che vengono de-allocate** quando viene invocato il “`return`”
- Quindi il **chiamante** vedrebbe un’area di memoria con elementi a caso… un **dangling pointer**!

Funzioni

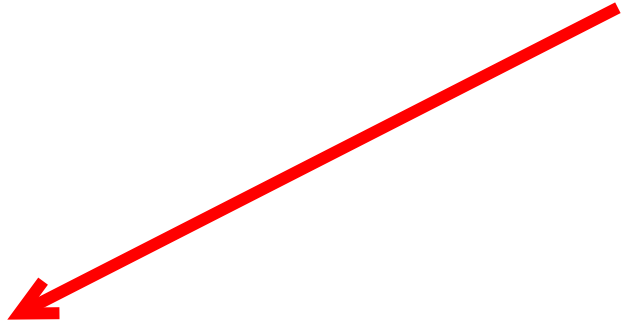
E Se volessi restituire un array?

Come si fa?

Funzioni

E Se volessi restituire un array?

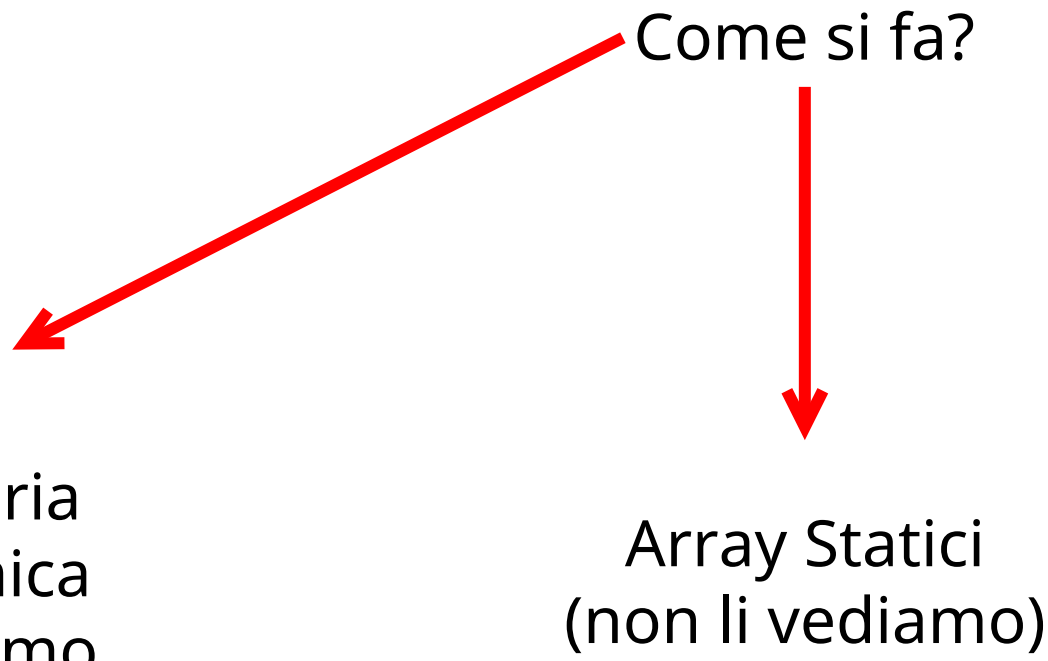
Come si fa?



Memoria
Dinamica
(vedremo
più avanti)

Funzioni

E Se volessi restituire un array?



Funzioni

E Se volessi restituire un array?

Funzioni

E Se volessi restituire un array?

```
#define N 100

void raddoppiaArray(int v[], int dim, int res[]){
    int i;
    for(i=0; i<dim ; i++){
        res[i] = 2*v[i];
    }
    return;
}
```

Funzioni

E Se volessi restituire un array?

```
#define N 100

void raddoppiaArray(int v[], int dim, int res[]){
     int i;
     for(i=0; i<dim ; i++){
          res[i] = 2*v[i];
     }
     return;
}
```

Funzioni

E Se volessi restituire un array?

```
#define N 100

void raddoppiaArray(int v[], int dim, int res[]){
     int i;
     for(i=0; i<dim ; i++){
          res[i] = 2*v[i];
     }
     return;
}
```

Funzioni

Interazioni con Matrici

Funzioni

Interazioni con Matrici

```
#define R …
#define C …
void fMatrice(int m[][C]){
    // …
}
```

Funzioni

Interazioni con Matrici

```
#define R …
#define C …
void fMatrice(int m[][C]){
    // …
}
```

Prototipo

```
#define R …
#define C …
void fMatrice(int [][C]);
```

Funzioni

Interazioni con Matrici

```
#define R …
#define C …
void fMatrice(int m[][C]){
    // …
}
```

Prototipo

```
#define R …
#define C …
void fMatrice(int [][C]);
```

Qui bisogna **esplicitare il numero di colonne** (ma si può omettere quello di righe)
Questo per come è **organizzato** un array multi-dimensionale in **memoria**
Il resto è come per gli array mono-dimensionali

Funzioni

Esercizi – Parte 2

Esercizio 10

Conta Occorrenze in un Array

Testo

Scrivere un programma che legge un array di interi e un numero e stampa il numero di volte che il numero appare all'interno dell'array.

Scrivere le funzioni:

- `leggiInt()`: che legge e restituisce un intero
- `leggiIntN()`: che legge e restituisce un intero minore di N e positivo
- `leggiArrayInt()`: che legge e restituisce un array di interi di una certa dimensione
- `contaOccorrenze()`: che prende in input un array e un numero e restituisce il numero di volte che il numero appare all'interno dell'array

Esercizio 11

Elimina Carattere da Stringa

Testo

Scrivere un programma che legge una stringa senza spazi e un carattere e stampa la stringa inserita senza il carattere inserito.

Scrivere le funzioni (da mappare con quanto possibile fare in C):

- `leggiCarattere()`: che legge e restituisce un carattere
- `leggiStringa()`: che legge e restituisce una stringa
- `rimuoviCarattere()`: che prende in input una stringa e un carattere e restituisce la stringa senza il carattere inserito (in un'altra stringa)

Esercizio 12

Lunghezza Massima Sottosequenza Ascendente

Testo

Scrivere un programma che prende in input nella funzione main una stringa. Il programma stampa la lunghezza della più lunga sottosequenza di caratteri ordinati in ordine ascendente nella stringa.

Scrivere le funzioni:

- `lunghezzaMassimaSottosequenzaAscendente()`: che prende in input un puntatore che punta alla stessa area di memoria della stringa e la lunghezza della stringa. La funzione restituisce un intero pari alla più lunga sottosequenza di caratteri ordinati in ordine ascendente nella stringa.
- `leggiStringa()`: che legge e restituisce una stringa

Esercizio 13

Elevare al Quadrato una Porzione dell'Array

Testo

Scrivere un programma che legge un array di lunghezza definita dall'utente (al massimo una dimensione predeterminata) e che modifica l'array elevando al quadrato una porzione di esso a partire da una posizione decisa dall'utente fino a quando l'array non termina o fino a quando non si incontra uno 0.

Scrivere le funzioni:

- `leggiInt()`: che legge e restituisce un intero
- `leggiIntN()`: che legge e restituisce un intero in [1,N]
- `leggiArrayInt()`: che legge e restituisce un array di interi di una certa dimensione
- `foo()`: che eleva al quadrato gli elementi dell'array da una certa posizione in poi fino ad arrivare alla fine dell'array o fino a quando l'array non termina. Tale funzione fa uso della funzione ausiliaria `subpow()`
- **`subpow()`: che prende l'inizio di una porzione dell'array e la dimensione di tale sotto-array e lo eleva al quadrato**
- `stampaArray()`: che stampa un array passato in input

Esercizio 13

Elevare al Quadrato una Porzione dell'Array

Testo

Scrivere un programma che legge un array di lunghezza definita dall'utente (al massimo una dimensione predeterminata) e che modifica l'array elevando al quadrato una porzione di esso a partire da una posizione decisa dall'utente fino a quando l'array non termina o fino a quando non si incontra uno 0.

Scrivere le funzioni:

- `leggiInt()`: che legge e restituisce un intero
- `leggiIntN()`: che legge e restituisce un intero in [1,N]
- `leggiArrayInt()`: che legge e restituisce un array di interi di una certa dimensione
- `foo()`: che eleva al quadrato gli elementi dell'array da una certa posizione in poi fino ad arrivare alla fine dell'array o fino a quando l'array non termina. Tale funzione fa uso della funzione ausiliaria `subpow()`
- **`subpow()`: che prende l'inizio di una porzione dell'array e la dimensione di tale sotto-array e lo eleva al quadrato NB: tutto quel pezzo dell'array! Partiamo da qui!**
- `stampaArray()`: che stampa un array passato in input

Esercizio 14

Elimina Sottostringa da Stringa

Testo

Scrivere un programma che prende in input una stringa e una seconda stringa. Il programma rimuove dalla prima stringa tutte le sequenze di caratteri uguali alla seconda stringa.

Scrivere le funzioni:

- `leggiStringa()`
- `rimuoviStringa()`

Esercizio 15

Shift di Matrice per Righe

Testo

Scrivere un programma che legge una matrice 3x4 ed effettua uno shift per righe verso il basso e stampa il risultato. L’ultima riga prende la posizione della prima riga.

Scrivere le funzioni:

- `leggiArray()`
- `leggiMatrice()`
- `stampaArray()`
- `stampaMatrice()`
- `copiaArray()`
- `shiftMatrice()`

Esercizio 16

TDE 2026/09/09

Testo

Si consideri una matrice rettangolare che rappresenta le piazzole di un'area di parcheggio per droni. Ogni elemento della matrice è una struttura di tipo:

```
typedef struct {
     int occupata;      /* 1 se la piazzola contiene un drone, 0 altrimenti */
     int ricarica;      /* 1 se la piazzola dispone di ricarica, 0 altrimenti */
     int coperta;       /* 1 se la piazzola e' coperta, 0 altrimenti */
   int idDrone;       } Piazzola;
```

La matrice ha nr righe e nc colonne effettivamente utilizzate.

Esercizio 16

TDE 2026/09/09

Testo

Per ragioni di sicurezza, una disposizione dei droni è considerata corretta se valgono entrambe le condizioni seguenti:

1. Due righe consecutive non possono contenere entrambe almeno un drone.
2. Sulla stessa riga, due droni devono avere indici di colonna che differiscono di almeno 3.

Equivalentemente, dopo una piazzola occupata, le due piazzole immediatamente successive sulla stessa riga devono essere libere, se esistono.

Esercizio 16

TDE 2026/09/09

Testo

Per ragioni di sicurezza, una disposizione dei droni è considerata corretta se valgono entrambe le condizioni seguenti:

1. Due righe consecutive non possono contenere entrambe almeno un drone.
2. Sulla stessa riga, due droni devono avere indici di colonna che differiscono di almeno 3.

Equivalentemente, dopo una piazzola occupata, le due piazzole immediatamente successive sulla stessa riga devono essere libere, se esistono.

Esercizio 16

TDE 2026/09/09

Testo

Non è richiesto che siano occupate solo le righe pari o solo le righe dispari: è vietato esclusivamente che due righe consecutive contengano entrambe droni.

Esempi:

Disposizione corretta:

Indicando con X una piazzola occupata e con . una piazzola libera abbiamo che la disposizione

Esercizio 16

TDE 2026/09/09

Testo

X . . X . .

. . . . . .

. X . . X .

È corretta: le righe occupate non sono consecutive e, sulla stessa riga, i droni distano almeno 3 colonne.

Esercizio 16

TDE 2026/09/09

Testo

I droni sono nelle colonne 0 e 2: la differenza tra gli indici di colonna è 2, quindi la disposizione non è corretta.

Disposizione non corretta per righe consecutive:

X . . . . .

. . . X . .

Le due righe consecutive contengono entrambe almeno un drone.

Esercizio 16

TDE 2026/09/09

Testo

Partendo al file a fine consegna:

1. Si implementi la funzione disposizioneCorretta che presi in ingresso gli opportuni parametri, controlla la disposizione dei droni nel parcheggio. La funzione deve restituire 1 se la disposizione soddisfa entrambe le condizioni precedenti, 0 altrimenti. La funzione non deve modificare la matrice.

2. Per poter rimanere parcheggiato, ogni drone deve inoltre trovarsi su una piazzola dotata di ricarica. Si implementi la funzione: rimuoviSenzaRicaricaLa funzione, presi in input gli opportuni parametri, deve individuare tutti i droni parcheggiati su piazzole prive di ricarica. Per ciascuno di essi deve copiare l'identificativo del drone nell'array idRimossi, liberare la piazzola, ponendo occupata = 0 e idDrone = -1. I campi ricarica e coperta non devono essere modificati. La funzione restituisce il numero di droni rimossi. Si assuma che idRimossi abbia spazio sufficiente.

Esercizio 16

TDE 2026/09/09

Testo

Facendo riferimento al main fornito l’output atteso è

1. A - disposizione corretta

Droni su piazzole senza ricarica: 1

ID rimossi: 103

2. B - disposizione non corretta

Esercizio 16

TDE 2026/09/09

Testo

File di partenza

```c
#include <stdio.h>

#define MAXR 100
#define MAXC 100

typedef struct {
    int occupata,ricarica,coperta,idDrone;
} Piazzola;

int disposizioneCorretta(Piazzola m[][MAXC], int nr, int nc);
int rimuoviSenzaRicarica(Piazzola m[][MAXC],int nr,int nc,int idRimossi[]);
```

Esercizio 16

TDE 2026/09/09

Testo

File di partenza

(inizio del main)

```c
int main() {
    int i,corretta;
    int idRimossiA[MAXR * MAXC],nRimossiA,idRimossiB[MAXR * MAXC],nRimossiB;
    int nrA=3,ncA=6,nrB=2,ncB=6;

    Piazzola A[MAXR][MAXC]={0};
    A[0][0].occupata=1;
    A[0][0].ricarica=1;
    A[0][0].idDrone=101;
    A[0][3].occupata=1;
    A[0][3].ricarica=1;
    A[0][3].idDrone=102;

    A[2][1].occupata=1;
    A[2][1].ricarica=0;    /* drone da rimuovere */
    A[2][1].idDrone=103;
    A[2][4].occupata=1;
    A[2][4].ricarica=1;
    A[2][4].idDrone=104;


    Piazzola B[MAXR][MAXC]={0};
    B[0][0].occupata=1;
    B[0][0].ricarica=1;
    B[0][0].idDrone=201;
    B[1][3].occupata=1;
    B[1][3].ricarica=1;
    B[1][3].idDrone=202;
```

Esercizio 16

TDE 2026/09/09

Testo

File di partenza

(continua, questa è la fine del main)

```
    corretta=disposizioneCorretta(A, nrA, ncA);
    if(corretta){
        printf("A - disposizione corretta\n");
        nRimossiA=rimuoviSenzaRicarica(A, nrA, ncA, idRimossiA);
        printf("Droni su piazzole senza ricarica: %d\n",nRimossiA);

        printf("ID rimossi:");

        for (i=0; i < nRimossiA; i++)
            printf(" %d", idRimossiA[i]);
        printf("\n\n");
    }
    else printf("A - disposizione non corretta\n");

    corretta=disposizioneCorretta(B, nrB, ncB);

    if(corretta){
        printf("B - disposizione corretta\n");
        nRimossiB=rimuoviSenzaRicarica(B, nrB, ncB, idRimossiB);
        printf("Droni su piazzole senza ricarica: %d\n",nRimossiB);

        printf("ID rimossi:");

        for (i=0; i < nRimossiB; i++)
            printf(" %d", idRimossiB[i]);
        printf("\n\n");
    }
    else printf("B - disposizione non corretta\n");

    return 0;
}
```

Esercizio 16

TDE 2026/09/09

Testo

File di partenza

(continua)

```c
int disposizioneCorretta(Piazzola m[][MAXC], int nr, int nc) {
    //MODIFICARE QUI
    return 1;
}


int rimuoviSenzaRicarica(Piazzola m[][MAXC],int nr,int nc,int idRimossi[]) {
    int nRimossi=0;

    //MODIFICARE QUI

    return nRimossi;
}

//AGGIUNGERE QUI EVENTUALI FUNZIONI AGGIUNTIVE
```

Esercizio 17

Implementazione strcpy

Testo

Si implementi la funzione `strcpy(dest, src)`.

Esercizio 18

Prova in Itinere 2025/11/11

Testo

Si considerino le seguenti definizioni di tipi:

```c
typedef struct {
    char piatto[100];
    int calorie;
} portata;

typedef struct {
    portata primo;
    portata secondo;
    portata frutta;
    portata dolce;
} pasto;

typedef struct {
    pasto menu[10];
    int numPasti;
} menu;
```

Esercizio 18

Prova in Itinere 2025/11/11

Testo

Ogni pasto è composto da quattro portate: primo, secondo, frutta e dolce.

Un menu contiene al massimo 10 pasti completi e un intero numPasti che indica quanti pasti sono effettivamente memorizzati.

Si scrivano in linguaggio C le seguenti funzioni:

Esercizio 18

Prova in Itinere 2025/11/11

Testo

1. stampaMenu che stampa a video, in modo leggibile, l’intero menù;

2. sommePasti che prende in ingresso un menu ed eventuali variabili aggiuntive e calcola, per ciascun pasto del menù, la somma totale delle calorie (somma di primo, secondo, frutta e dolce). I totali delle calorie sono memorizzate in un vettore out che deve essere riportato al chiamante e stampato nel main.

Esercizio 18

Prova in Itinere 2025/11/11

Testo

3. eliminaPastiIpercalorici che elimina dal menù tutti i pasti che superano una soglia calorica specificata in input (ad esempio 2000 kcal). La funzione deve spostare i pasti rimanenti nel menu in modo da non lasciare “buchi” nell’array. La funzione deve inoltre deve restituire il numero di pasti eliminati. Si invochi la funzione eliminaPastiIpercalorici e quindi stampaMenu per mostrare il nuovo menù.

Esercizio 18

Prova in Itinere 2025/11/11

Testo

Si parta dal file sottostante, un main() di test.

Usando il file dato l'output sarà quello mostrato nell'immagine seguente:

```
-- SOMME CALORIE PER PASTO --
1695 1795 780 2300 2170 1360 kcal

-- DOPO ELIMINAZIONE: rimossi 2 --
Pasto 0:
  Primo:   Risotto ai funghi       720
  Secondo: Salmone al forno        520
  Frutta:  Pera                     95
  Dolce:   Panna cotta             360
Pasto 1:
  Primo:   Penne al pesto          700
  Secondo: Vitello tonnato         560
  Frutta:  Banana                  105
  Dolce:   Cheesecake              430
Pasto 2:
  Primo:   Minestrone              250
  Secondo: Bresaola rucola         300
  Frutta:  Arancia                  70
  Dolce:   Sorbetto                160
Pasto 3:
  Primo:   Gnocchi burro e salvia  680
  Secondo: Orata alla piastra      410
  Frutta:  Fragole                  50
  Dolce:   Gelato                  220
```

Esercizio 18

Prova in Itinere 2025/11/11

Testo

File di partenza:

Esercizio 18

Prova in Itinere 2025/11/11

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

typedef struct {
    char piatto[100];
    int calorie;
} portata;

typedef struct {
    portata primo;
    portata secondo;
    portata frutta;
    portata dolce;
} pasto;

typedef struct {
    pasto menu[10];
    int numPasti;
} menu;
```

```c
int main(void) {
    menu m = {
        {{{"Risotto ai funghi",720},{"Salmone al
forno",520},{"Pera",95},{"Panna cotta",360} },
        {{"Penne al pesto",700},{"Vitello
tonnato",560},{"Banana",105},{"Cheesecake",430} },
        {{"Minestrone",250},{"Bresaola
rucola",300},{"Arancia",70},{"Sorbetto",160} },
        {{"Maccheroni ala gricia",950},{"Costata di
manzo",850},{"Mela",80},{"Tiramisu",420} },
        {{"Lasagne",820},{"Costata di
manzo",850},{"Kiwi",60},{"Cannolo",440} },
        {{"Gnocchi burro e salvia",680},{"Orata alla
piastra",410},{"Fragole",50},{"Gelato",220} }
        },6};

    return 0;
}
```

Thank You!

Mail: luca1.alessandrini@polimi.it

Website: https://alessandriniluca.github.io/