UNIVERSITY OF TRENTO - Italy
Department of Information
and Communication Technology

Laboratorio 9

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2024-2025
Università degli Studi di Trento
pierluigi.roberti@unitn.it

Generazione casuale

- Interi

[0 – max) `rand()%max` →valore max escluso!!

[min – max] `rand()%(max - min + 1 ) + min`

- Caratteri

[minc – maxc] `rand()%('maxc' - 'minc' + 1 ) + 'minc'`

[d - w] `rand()%('w' - 'd' + 1 ) + 'd'`

- Floating point

[min – max] `(rand()%(max*K - min*K +1)+ min*K)/(float)K`

[2.0 – 25.0] `(rand()%(25*100 - 2*100 +1)+ 2*100) / 100.0`

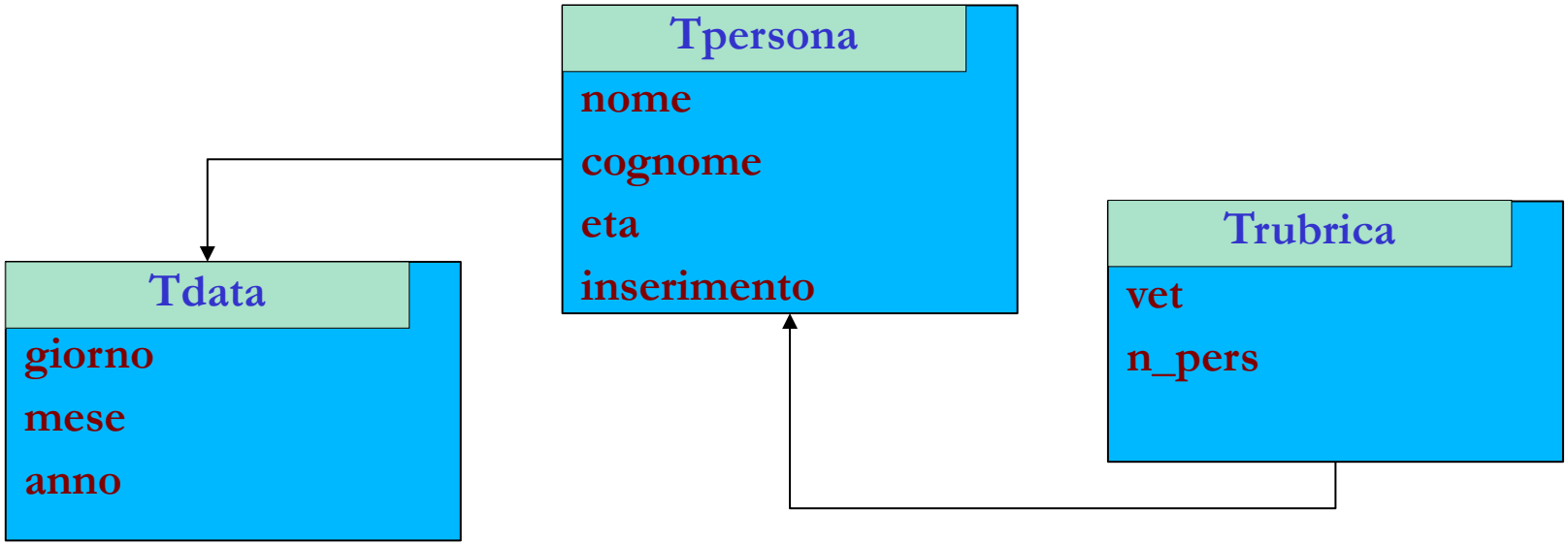
Generazione casuale

- Stringhe
  - Sequenza di caratteri terminate da ‘\0’
  - s: stringa (MAX: 99 caratteri)
  - len: lunghezza stringa (tipo int)

```
char s[100];
int len = 5;
// Nota: caratteri minuscoli
for(i=0 ; i<len ; i++) {
    s[i] = rand()%(‘z’ – ‘a’ + 1 ) + ‘a’;
}
s[i] = ‘\0’ // Nota: i vale len
```

Esercizio 1

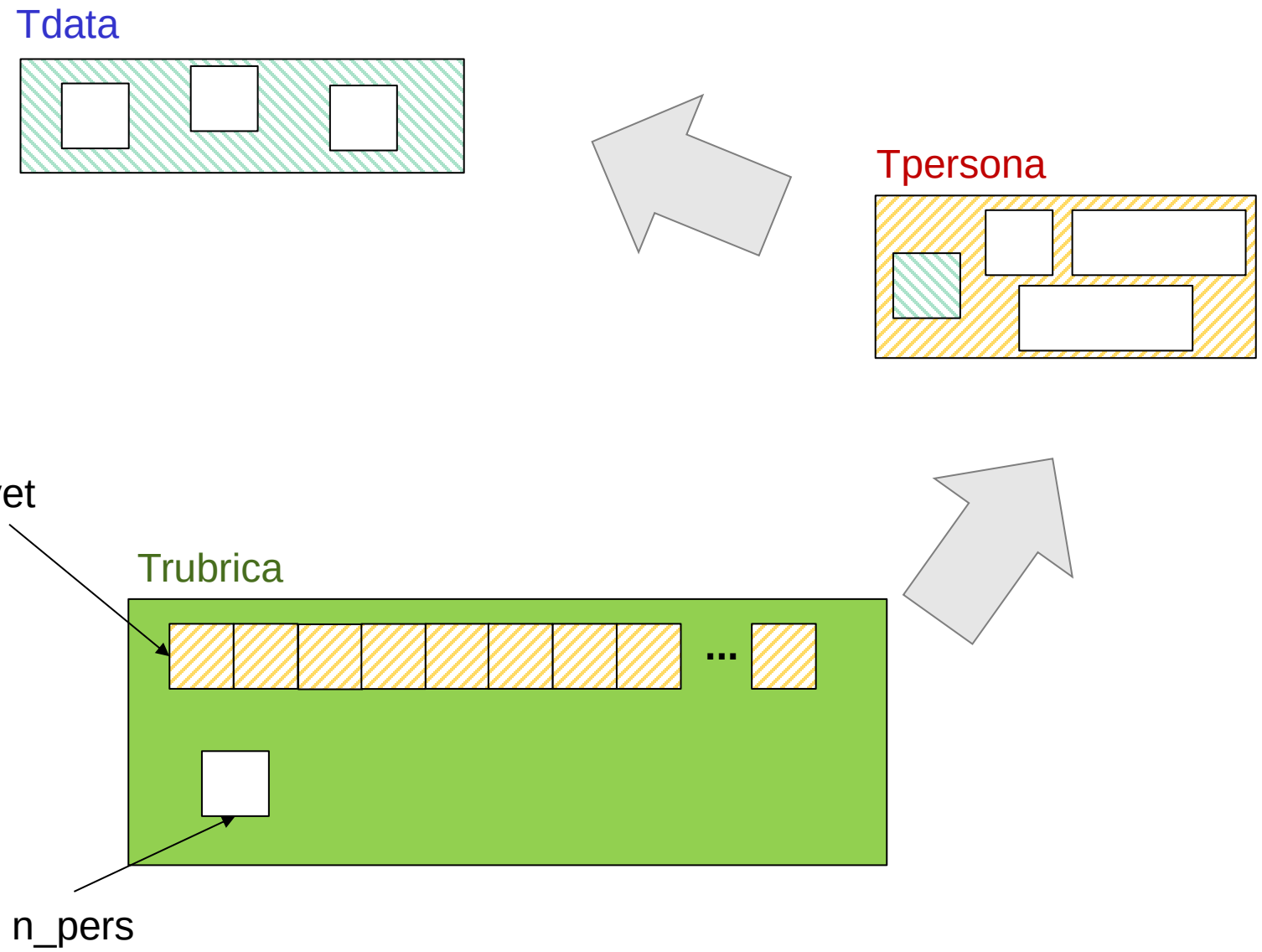
- Si crei un tipo di dato basato su una struttura per memorizzare i dati di una data chiamato «Tdata».
- Si crei un tipo di dato basato su una struttura per memorizzare i dati di una persona chiamato «Tpersona».
- Si crei un tipo di dato chiamato «Trubrica» che contiene
  - un array di NPERSONE (costante pari a 10) di tipo Tpersona
  - un valore intero n_pers (per contare quanti elementi sono stati inseriti nell'array)



Esercizio 1

- Nel main dichiarare una variabile r di tipo Trubrica
- Leggere da tastiera i dati di massimo 10 persone (inserendole nell'array: memorizzare il numero di elementi inizializzato)
  - Inizializzare il campo inserimento usando un numeri casuali per giorno, mese e anno (dopo il 1/1/1950).
  - Inizializzare il campo eta usando un numeri casuali tra 1 e 90.
  - Il campo nome e cognome devono essere letti da tastiera
  - Dopo ogni inserimento domandare se si vuole proseguire con l'inserimento nell'array di una ulteriore Persona
  - NOTA: ricordarsi di modificare il campo n_pers!!!
- Stampare l'array delle persone nella forma:
  [campo nome] [campo cognome] di anni [campo eta]
- Stampare a video nome e cognome del più anziano e del più giovane.
- Mostrare la media delle età delle persone inserite nell'array
- NOTA: Scorrere l'array tenendo conto dei soli elementi inseriti (n_pers)

Esercizio 1



Esercizio 2

- Si creino delle struct adatte per rappresentare:
  A. punti in coordinate bidimensionali (Tvertice)
  B. una figura 2D composta da massimo NVERTICI vertici (Tfigura)
     - NVERTICI pari a 8 (direttiva define)
     - Numero effettivo di vertici casuale (minimo 3)
     - Figura chiusa
     - Spigolo: collega vertice i con vertice i+1
  C. un insieme di figure (Tdisegno)
     - Numero massimo di figure NFIGURE 10
     - Numero effettivo di figure casuale (minimo 4)
- Si scriva un programma che generi le coordinate dei vertici in modo casuale
  - Ogni coordinata nel range -20.0 +20.0
- Identificare la figura la cui somma della lunghezza degli spigoli è massima

Esercizio 2

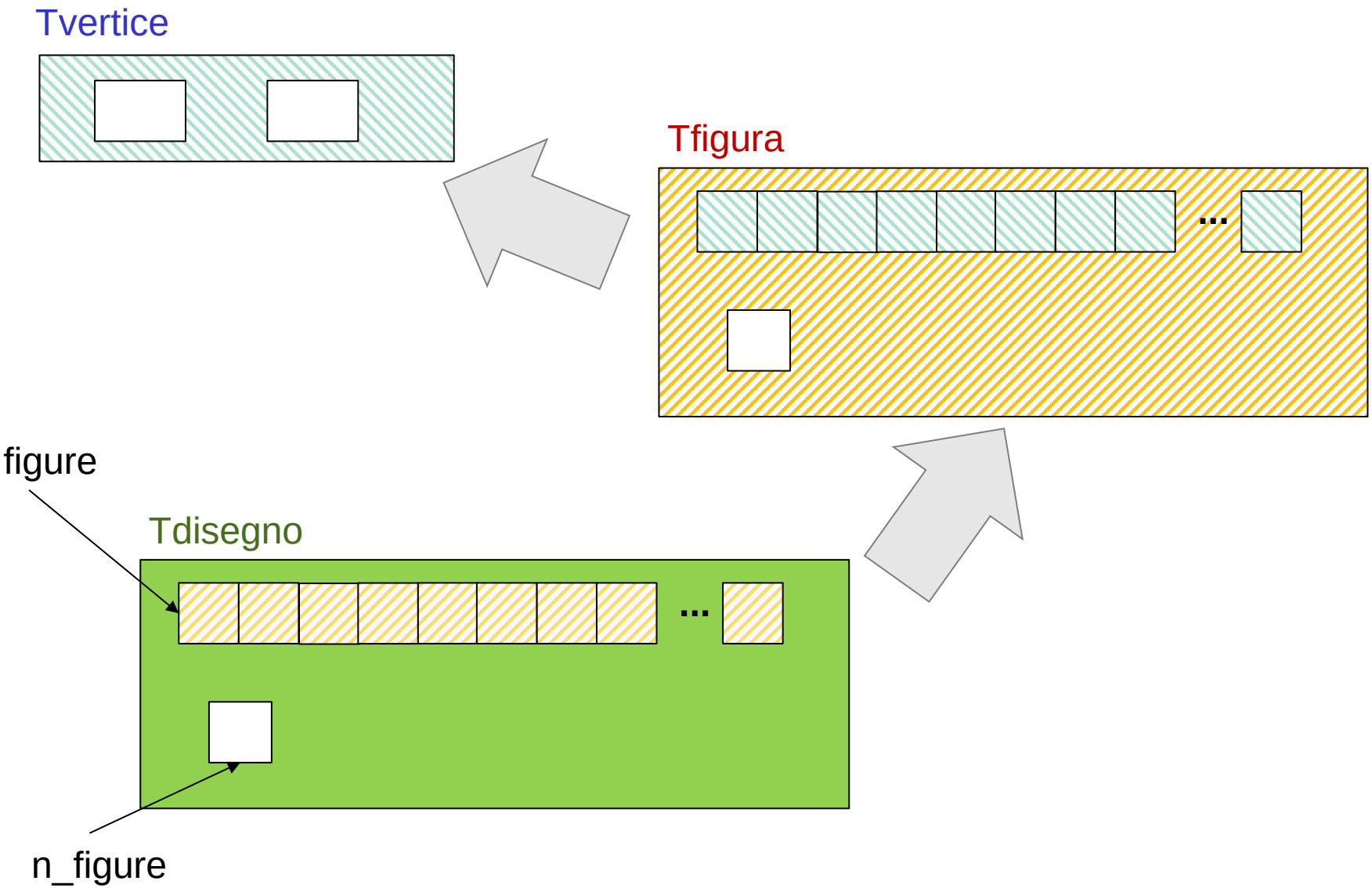
```
#define NFIGURE 10
#define NVERTICI 8
#define VMIN -20
#define VMAX 20

typedef struct Tvertice{
  float x;
  float y;
} Tvertice;

typedef struct Tfigura{
  Tvertice v[NVERTICI];
  int n_vertici;
} Tfigura;

typedef struct Tdisegno{
  Tfigura figure[NFIGURE];
  int n_figure;
} Tdisegno;
```

Esercizio 2



Esercizio 3

- Si creino delle struct adatte per rappresentare i colori di un'immagine:
  A. Colore di un pixel
     - Valori RGB. Ogni valore compreso tra 0-255
  B. una griglia di pixel di dimensione MASSIMA HxL (20x30)
     - Ogni elemento della griglia e' un pixel
     - Dimensione effettiva altezza della griglia (h)
     - Dimensione effettiva larghezza della griglia (l)
- Si scriva un programma che
  - Chiede all'utente le dimensioni effettive h ed l (controllo input)
  - Per ogni pixel, per ogni valore R, G, B
    - Chidere all'utente il valore (controllo input)
    - Oppure generare un numero compreso tra 0 e 255
- Stampare a video la griglia con formato per ogni pixel (r, g, b)
- Stampare a video la posizione del pixel la cui somma dei valori è massima

Esercizio 3

```
#define H 20
#define L 30
#define VMIN 0
#define VMAX 255

typedef struct Tpixel{
  int r, g, b;
} Tpixel;

typedef struct Timmagine{
  Tpixel imm[H][L];
  int h, l;
} Timmagine;
```

Esercizio 3 - b

- I valori R, G, B di solito sono rappresentati con valori ESADECIMALI
- Non modicare la fase di input
  - I valori R, G, B inseriti dall'utente sono in formato DECIMALE
- Modificare il programma precedente per stampare a video le terne in formato esadecimale
  - Formato per ogni pixel: (r_hex, g_hex, b_hex)
  - Ogni valore è identificato da due digits
    - 0 (dec) → 00 (hex)
    - 12 (dec) → 0C (hex)
    - 255 (dec) → FF (hex)

Esercizio 3 - b

- Opzione 1
  - **Mantenere** la struttura Tpixel così com'è e modificare solo la rappresentazione a video
- Opzione 2
  - **Modificare** la struttura Tpixel per memorizzare i valori letti

Esercizio 3 - b

```
#define H 20
#define L 30
#define VMIN 0
#define VMAX 255

typedef struct Tpixel{
  int r, g, b;
} Tpixel;

typedef struct Timmagine{
  Tpixel imm[H][L];
  int h, l;
} Timmagine;
```

Array di tipo char!!

```
typedef struct Tpixel{
  char r[2];
  char g[2];
  char b[2];
} Tpixel;
```

Posso stamparlo solo come %c%c

Serve un posto aggiuntivo per il terminatore \0. Posso stamparlo come %s

Stringa!!

```
typedef struct Tpixel{
  char r[3];
  char g[3];
  char b[3];
} Tpixel;
```

Conversione DEC – HEX – 1

```
int num, n_digit, tmp;
char hex[MAX];
printf("num: "); scanf("%d", &num);
n_digit=0;
tmp = num;
while(tmp>0){
      switch(tmp%16){
            case 0: hex[n_digit] = '0'; break;
            case 1: hex[n_digit] = '1'; break;
            case 2: hex[n_digit] = '2'; break;
            case 3: hex[n_digit] = '3'; break;
            case 4: hex[n_digit] = '4'; break;
            case 5: hex[n_digit] = '5'; break;
            case 6: hex[n_digit] = '6'; break;
            case 7: hex[n_digit] = '7'; break;
            case 8: hex[n_digit] = '8'; break;
```

Conversione DEC – HEX – 1

```
            case 9: hex[n_digit] = '9'; break;
            case 10: hex[n_digit] = 'A'; break;
            case 11: hex[n_digit] = 'B'; break;
            case 12: hex[n_digit] = 'C'; break;
            case 13: hex[n_digit] = 'D'; break;
            case 14: hex[n_digit] = 'E'; break;
            case 15: hex[n_digit] = 'F'; break;
      }
      tmp = tmp/16;
      n_digit++;
}
while(--n_digit>=0){
      printf("%c", hex[n_digit]);
}
```

Conversione DEC – HEX – 2

```
int num, n_digit, tmp;
char hex[MAX];
printf("num: "); scanf("%d", &num);
n_digit=0;
tmp = num;
while(tmp>0){
      if(tmp%16>=0 && tmp%16<=9){
            hex[n_digit] = '0'+ tmp%16;
      }else{
            hex[n_digit] = 'A'+ tmp%16 - 10;
      }
      tmp = tmp/16;
      n_digit++;
}
while(--n_digit>=0){
      printf("%c", hex[n_digit]);
}
```

Conversione DEC – HEX – 3

```
int num, n_digit, tmp;
char hex[MAX];
char alfabeto[]={'0', '1', '2', '3', '4', '5', '6', '7', '8',
'9', 'A', 'B', 'C', 'D', 'E', 'F'};
printf("num: "); scanf("%d", &num);
n_digit=0;
tmp = num;
while(tmp>0){
      hex[n_digit] = alfabeto[ tmp%16 ];
      tmp = tmp/16;
      n_digit++;
}
while(--n_digit>=0){
      printf("%c", hex[n_digit]);
}
```

Conversione DEC – HEX – 4

```
int num, n_digit, tmp;
char hex[MAX];
char alfabeto[]="0123456789ABCDEF";
printf("num: "); scanf("%d", &num);
n_digit=0;
tmp = num;
while(tmp>0){
      hex[n_digit] = alfabeto[ tmp%16 ];
      tmp = tmp/16;
      n_digit++;
}
while(--n_digit>=0){
      printf("%c", hex[n_digit]);
}
```

Conversione DEC – HEX – 5

```
int num;
printf("num: "); scanf("%d", &num);

printf("%x", num);
```

Esercizio 3 - b

- Conversione sapendo che i dati devono essere su due digit ed i valori sono compresi tra 0 e 255 (00 e FF)

```
int num;
char alfabeto[]="0123456789ABCDEF";
printf("num: "); scanf("%d", &num);
do{
        printf("num: "); scanf("%d", &num);
      if(num<0 || num>255){ printf("Errore.\n"); }
}while(num<0 || num>255);
printf("%c%c", alfabeto[num/16], alfabeto[num%16]);
```

Esercizio 4 – a

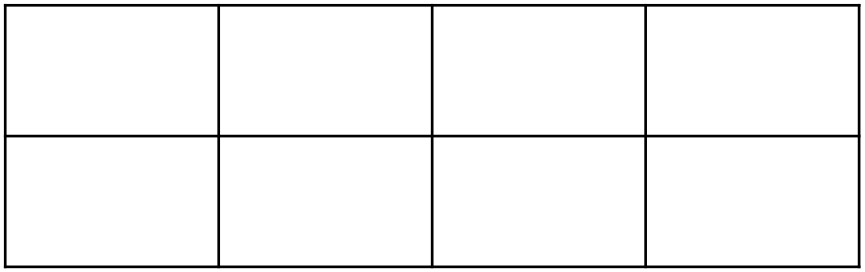
- Considerare uno spazio a **due dimensioni** suddiviso in celle
- C'e' stata un'esplosione in una cella e le schegge sono sparse in tutto il piano.
- Ci sono schegge piccole e grandi.
- Contare quante celle contengono schegge.

- Spazio 2D: matrice m[CELL_X][CELL_Y] di interi
- Inizializzazione matrice
  - 0 se nella cella non è presente una scheggia
  - 1 se nella cella è presente una scheggia piccola
  - 2 se nella cella è presente una scheggia grande

  QUINDI inserire in ogni cella un valore casuale tra 0 e 2
- Stampare a video il numero di schegge presenti nella matrice

Esercizio 4 – a

- Rappresentazione grafica



Esercizio 4 – a

```
int m[CELL_X][CELL_Y];  int x, y, conta;
for(x=0 ; x<CELL_X ; x++){
      for(y=0 ; y<CELL_Y ;y++){
            m[x][y] = rand()%3;
      }
}
conta = 0;
for(x=0 ; x<CELL_X ; x++){
      for(y=0 ; y<CELL_Y ;y++){
            if( m[x][y]>0 ){
                  conta++;
            }
      }
}
```

Esercizio 4 – b

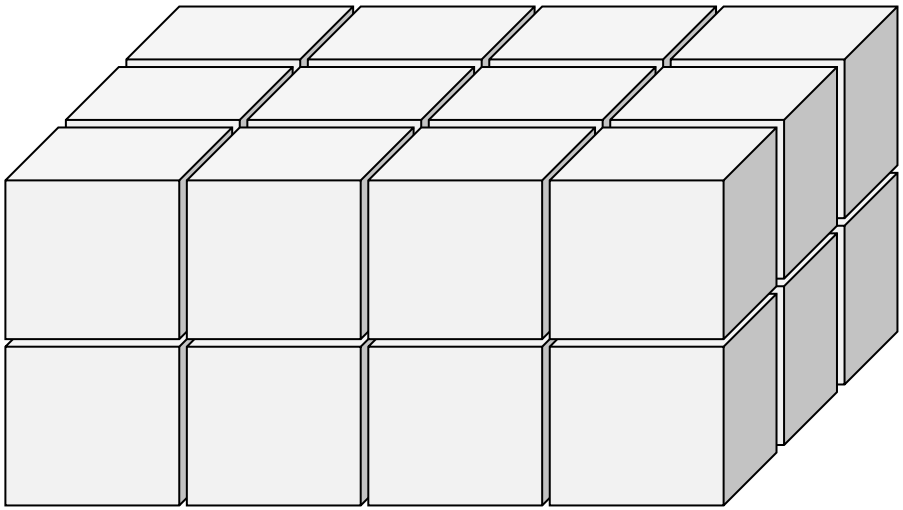
- Considerare uno spazio a **tre dimensioni** suddiviso in celle
- C'e' stata un'esplosione in una cella e le schegge sono sparse in tutto il piano.
- Ci sono schegge piccole e grandi.
- Contare quante celle contengono schegge.

- Spazio 3D: matrice m[CELL_X][CELL_Y][CELL_Z] di interi
- Inizializzazione matrice
  - 0 se nella cella non è presente una scheggia
  - 1 se nella cella è presente una scheggia piccola
  - 2 se nella cella è presente una scheggia grande

  QUINDI inserire in ogni cella un valore casuale tra 0 e 2
- Stampare a video il numero di schegge presenti nella matrice

Esercizio 4 – b

- Rappresentazione grafica



Esercizio 4 – b

```
int m[CELL_X][CELL_Y][CELL_Z];  int x, y, z, conta;
for(x=0 ; x<CELL_X ; x++){
      for(y=0 ; y<CELL_Y ;y++){
            for(z=0 ; z<CELL_Z ;z++){
                  m[x][y][z] = rand()%3;
            }
      }
}
conta = 0;
for(x=0 ; x<CELL_X ; x++){
      for(y=0 ; y<CELL_Y ;y++){
            for(z=0 ; z<CELL_Z ;z++){
                  if( m[x][y][z]>0 ){
                        conta++;
                  }
            }
      }
}
```

Esercizio 4 – c

- Considerare uno spazio a **quattro dimensioni** suddiviso in celle
  - Evoluzione nel tempo di uno spazio 3D
- C'e' stata un'esplosione in una cella e le schegge sono sparse in tutto il piano.
- Ci sono schegge piccole e grandi.
- Contare quante celle contengono schegge.

- Spazio 3D: matrice m[CELL_X][CELL_Y][CELL_Z][MAX_T] di interi
- Inizializzazione matrice
  - 0 se nella cella non è presente una scheggia
  - 1 se nella cella è presente una scheggia piccola
  - 2 se nella cella è presente una scheggia grande
- Stampare a video il numero di schegge presenti nella matrice

Esercizio 4 – c

- Rappresentazione grafica