Corso di Informatica Medica

Esercitazione IV

Alessandro A. Nacci
nacci@elet.polimi.it - alessandronacci.com

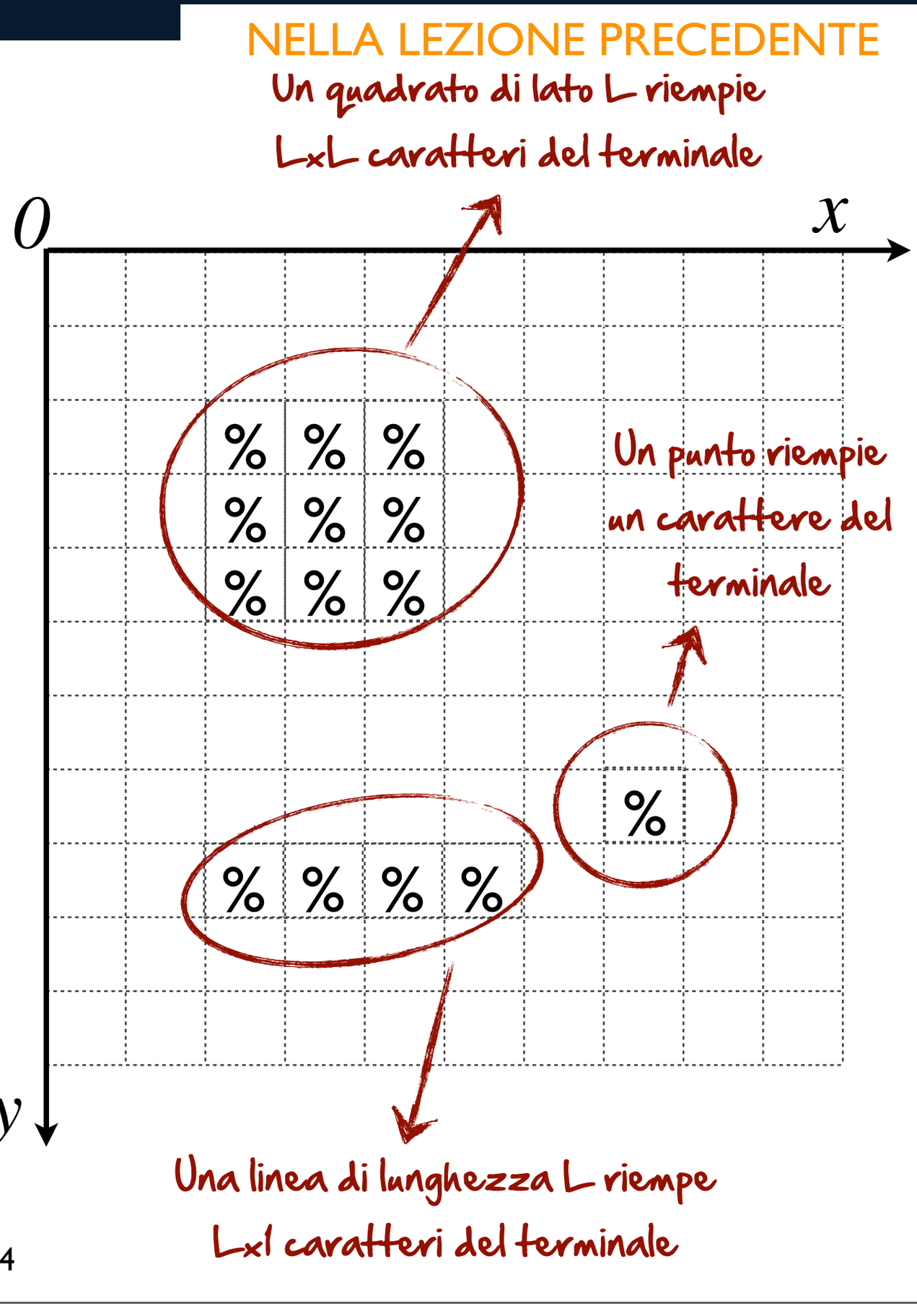
READY...
SET...
GO

WARNING
QUANTO FATTO OGGI
VERRA' UTILIZZATO
NELLE PROSSIME LEZIONI!

Fino ad ora...

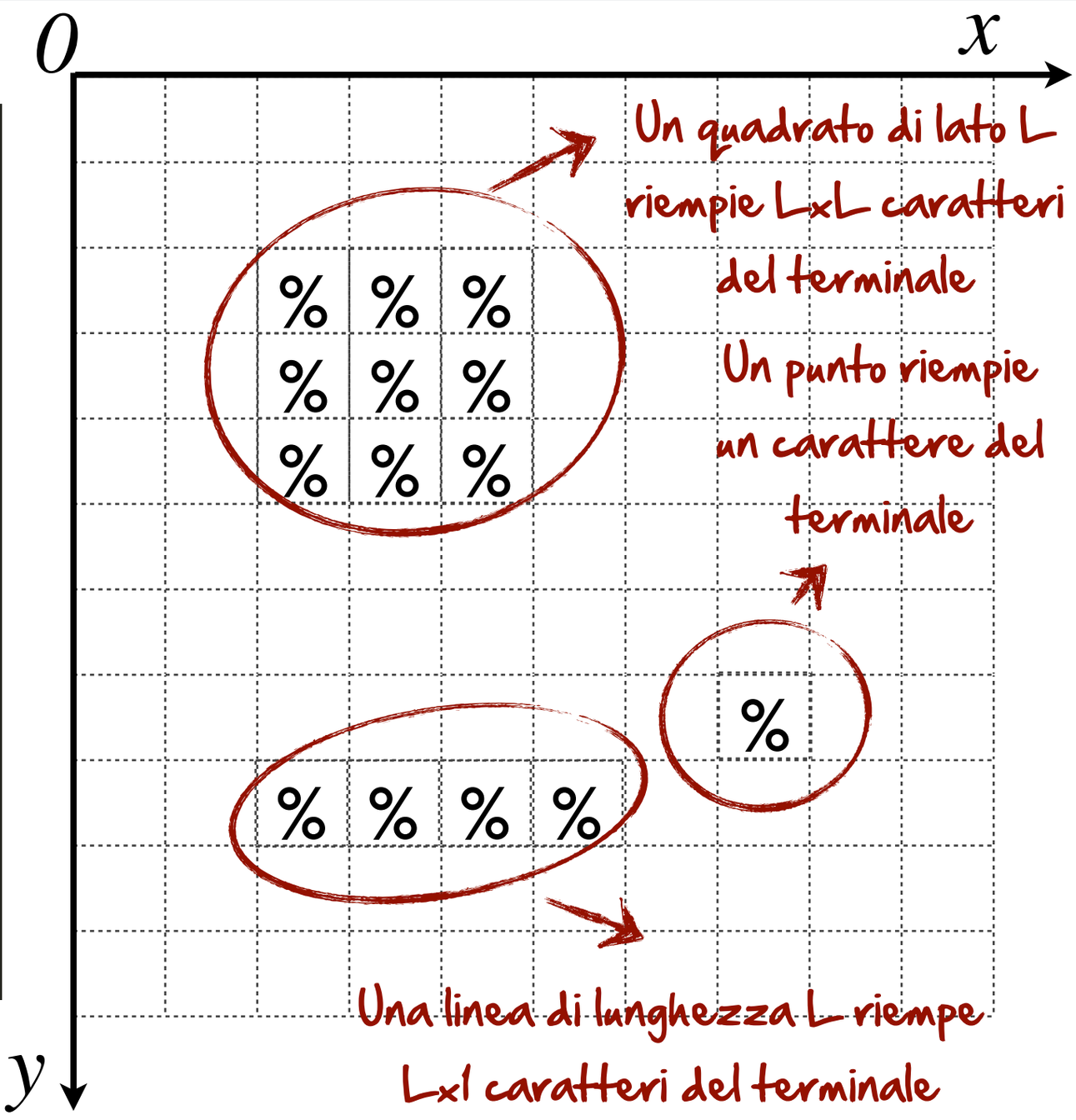
```c
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;

typedef struct {
    int x;
    int y;
    char valore;
} punto_schermo;

typedef struct {
    punto_schermo pixels[MAX_PUNTI_FORMA];
    int numero_pixel;
    categoria_forma categoria;
} forma;

// Funzioni per la visualizzazione a video
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
void aspetta_invio();
void pulisci_terminale();

// Modifica delle matriche che rappresenta lo schemo
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H]);
punto_schermo crea_punto_schermo(int x, int y, char valore);

// Generazione delle forme
forma genera_quadrato(int dim, char carattere);
forma genera_linea(int dim, direzione direzione_linea, char carattere);
forma genera_punto(char carattere);
forma genera_polinomio(int a2, int a1, int a0, int dim, char carattere);

int main(){

    char schermo[SCREEN_W][SCREEN_H];
    forma quadrato;
    punto_schermo p;
    forma linea_or;
    forma linea_vr;
    forma punto;
    // Disegnamo un quadrato
    inizializza_schermo(schermo);
    pulisci_terminale();

    quadrato = genera_quadrato(4, '#');
    p = crea_punto_schermo(1,1,0);

    disegna_forma(quadrato, p, schermo);
    disegna_schermo(schermo);

    aspetta_invio();
```

NELLA LEZIONE PRECEDENTE



Genera quadrato

```c
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;

typedef struct {
    int x;
    int y;
    char valore;
} punto_schermo;

typedef struct {
    punto_schermo pixels[MAX_PUNTI_FORMA];
    int numero_pixel;
    categoria_forma categoria;
} forma;

// Funzioni per la visualizzazione a video
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
void aspetta_invio();
void pulisci_terminale();

// Modifica delle matriche che rappresenta lo schemo
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H]);
punto_schermo crea_punto_schermo(int x, int y, char valore);

// Generazione delle forme
forma genera_quadrato(int dim, char carattere);
forma genera_linea(int dim, direzione direzione_linea, char carattere);
forma genera_punto(char carattere);
forma genera_polinomio(int a2, int a1, int a0, int dim, char carattere);
```

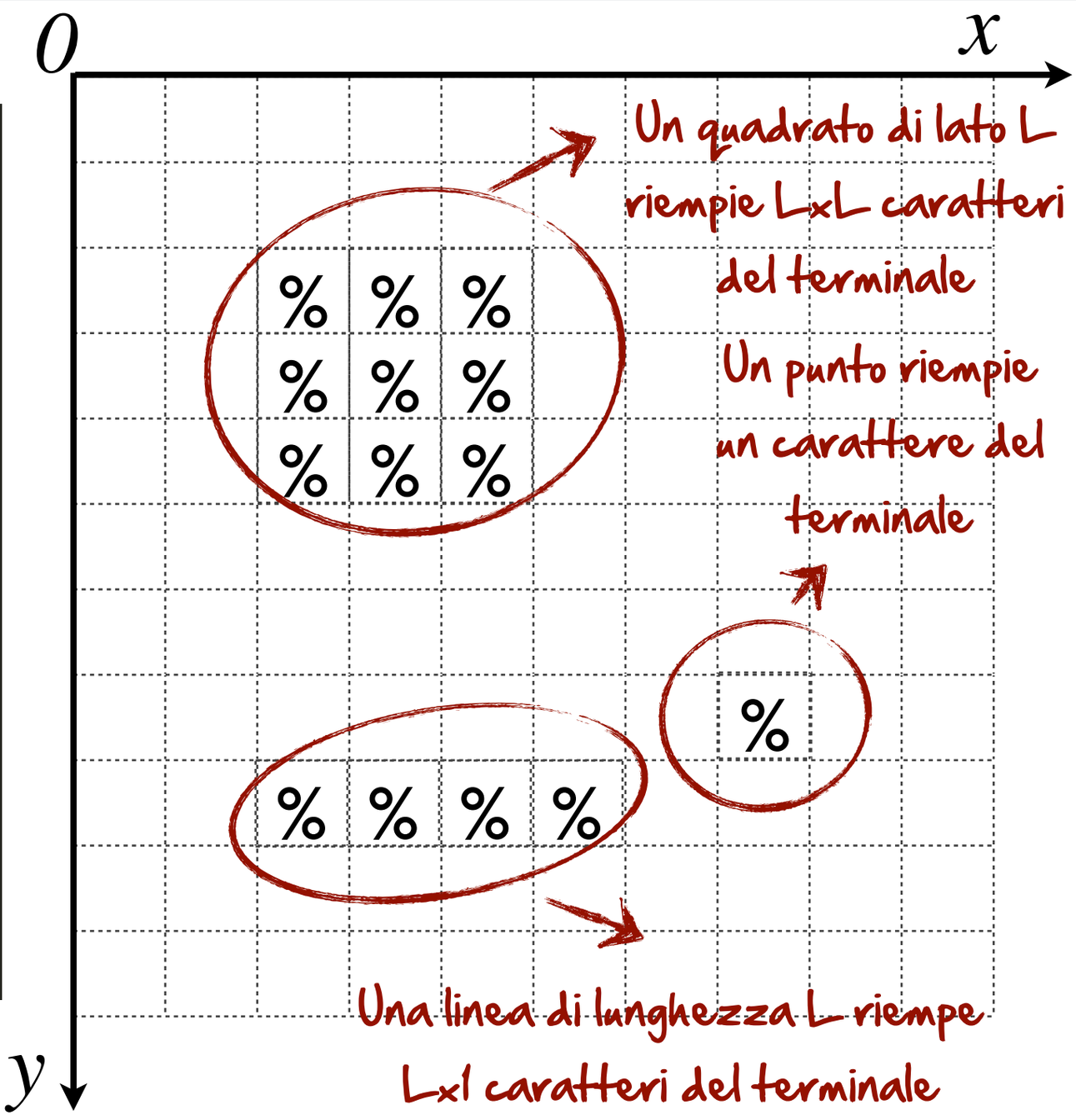



```
forma genera_quadrato(int dim, char carattere);
```

Restituisce una forma quadrata di lato 'dim'

Genera linea

```c
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;

typedef struct {
    int x;
    int y;
    char valore;
} punto_schermo;

typedef struct {
    punto_schermo pixels[MAX_PUNTI_FORMA];
    int numero_pixel;
    categoria_forma categoria;
} forma;

// Funzioni per la visualizzazione a video
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
void aspetta_invio();
void pulisci_terminale();

// Modifica delle matriche che rappresenta lo schemo
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H]);
punto_schermo crea_punto_schermo(int x, int y, char valore);

// Generazione delle forme
forma genera_quadrato(int dim, char carattere);
forma genera_linea(int dim, direzione direzione_linea, char carattere);
forma genera_punto(char carattere);
forma genera_polinomio(int a2, int a1, int a0, int dim, char carattere);
```

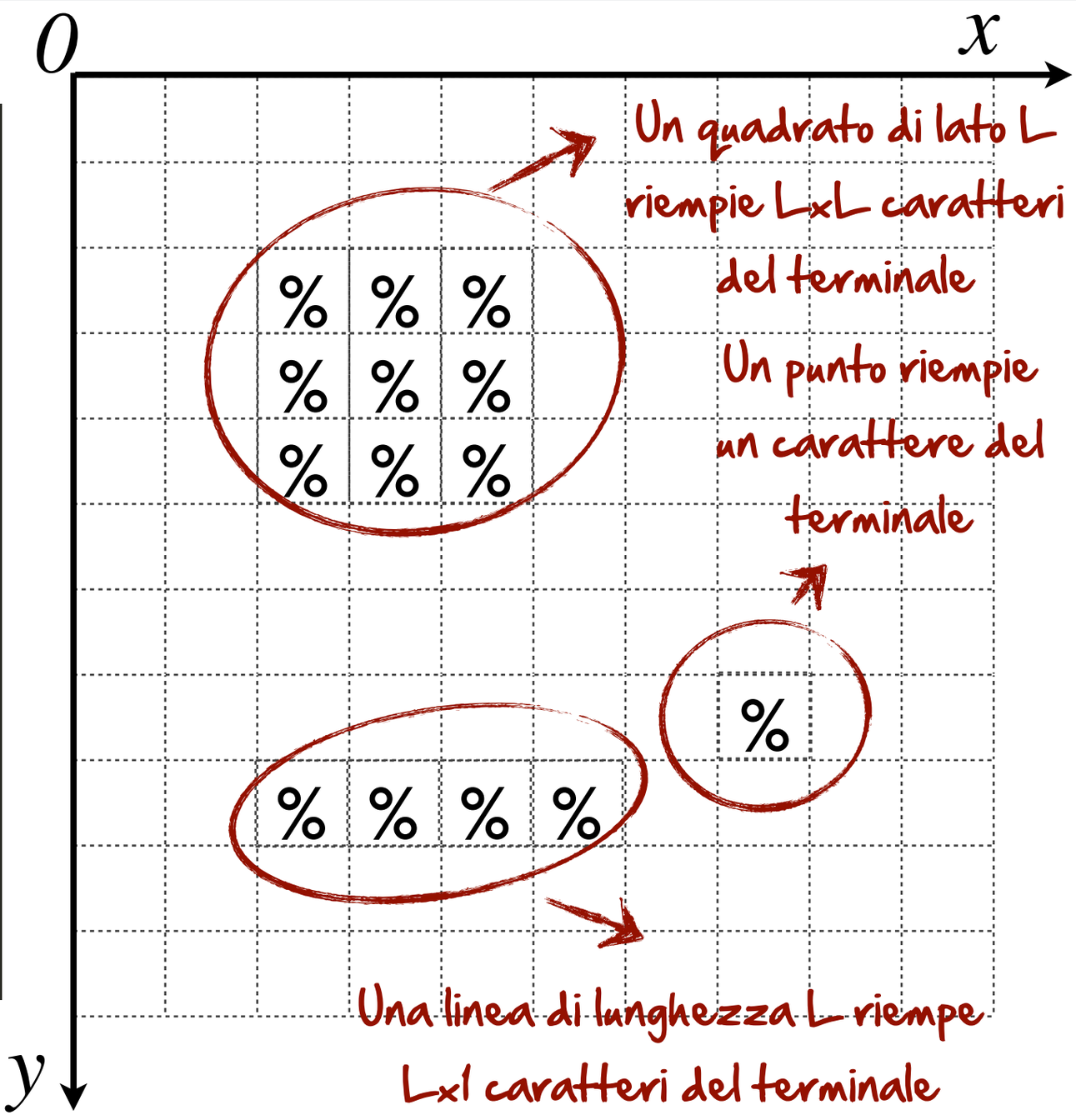



```
forma genera_linea(int dim,
direzione direzione_linea, char carattere);
```

Restituisce una forma linea di lunghezza 'dim' con direzione ‘direzione_linea

Genera punto

```c
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;

typedef struct {
    int x;
    int y;
    char valore;
} punto_schermo;

typedef struct {
    punto_schermo pixels[MAX_PUNTI_FORMA];
    int numero_pixel;
    categoria_forma categoria;
} forma;

// Funzioni per la visualizzazione a video
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
void aspetta_invio();
void pulisci_terminale();

// Modifica delle matriche che rappresenta lo schemo
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H]);
punto_schermo crea_punto_schermo(int x, int y, char valore);

// Generazione delle forme
forma genera_quadrato(int dim, char carattere);
forma genera_linea(int dim, direzione direzione_linea, char carattere);
forma genera_punto(char carattere);
forma genera_polinomio(int a2, int a1, int a0, int dim, char carattere);
```

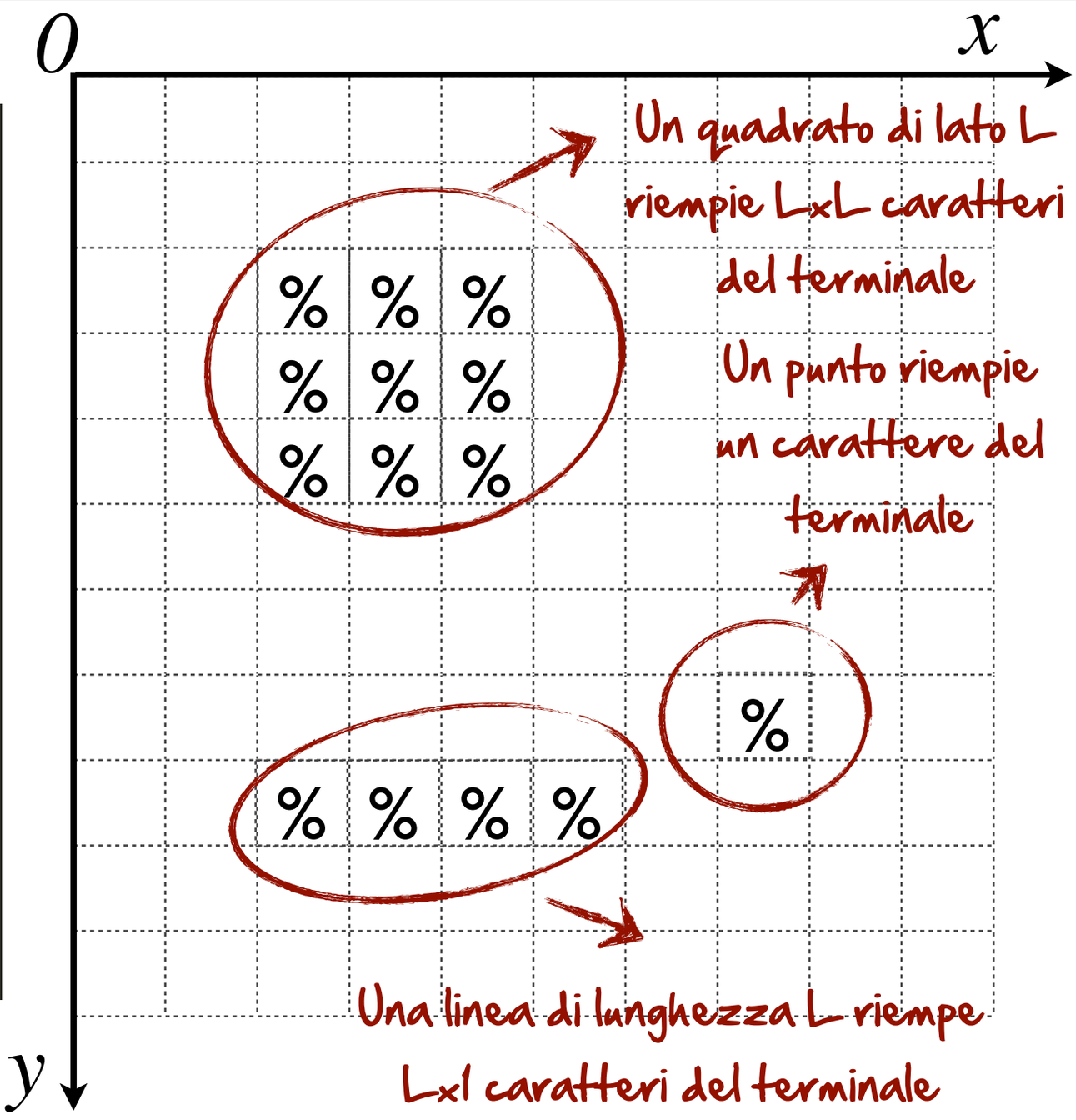



```
forma genera_punto(char carattere);
```

Restituisce una forma punto (linea di lunghezza 1)

Genera polinomio

```c
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;

typedef struct {
    int x;
    int y;
    char valore;
} punto_schermo;

typedef struct {
    punto_schermo pixels[MAX_PUNTI_FORMA];
    int numero_pixel;
    categoria_forma categoria;
} forma;

// Funzioni per la visualizzazione a video
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
void aspetta_invio();
void pulisci_terminale();

// Modifica delle matriche che rappresenta lo schemo
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H]);
punto_schermo crea_punto_schermo(int x, int y, char valore);

// Generazione delle forme
forma genera_quadrato(int dim, char carattere);
forma genera_linea(int dim, direzione direzione_linea, char carattere);
forma genera_punto(char carattere);
forma genera_polinomio(int a2, int a1, int a0, int dim, char carattere);
```




```
forma genera_polinomio(int a2, int a1,
   int a0, int dim, char carattere);
```

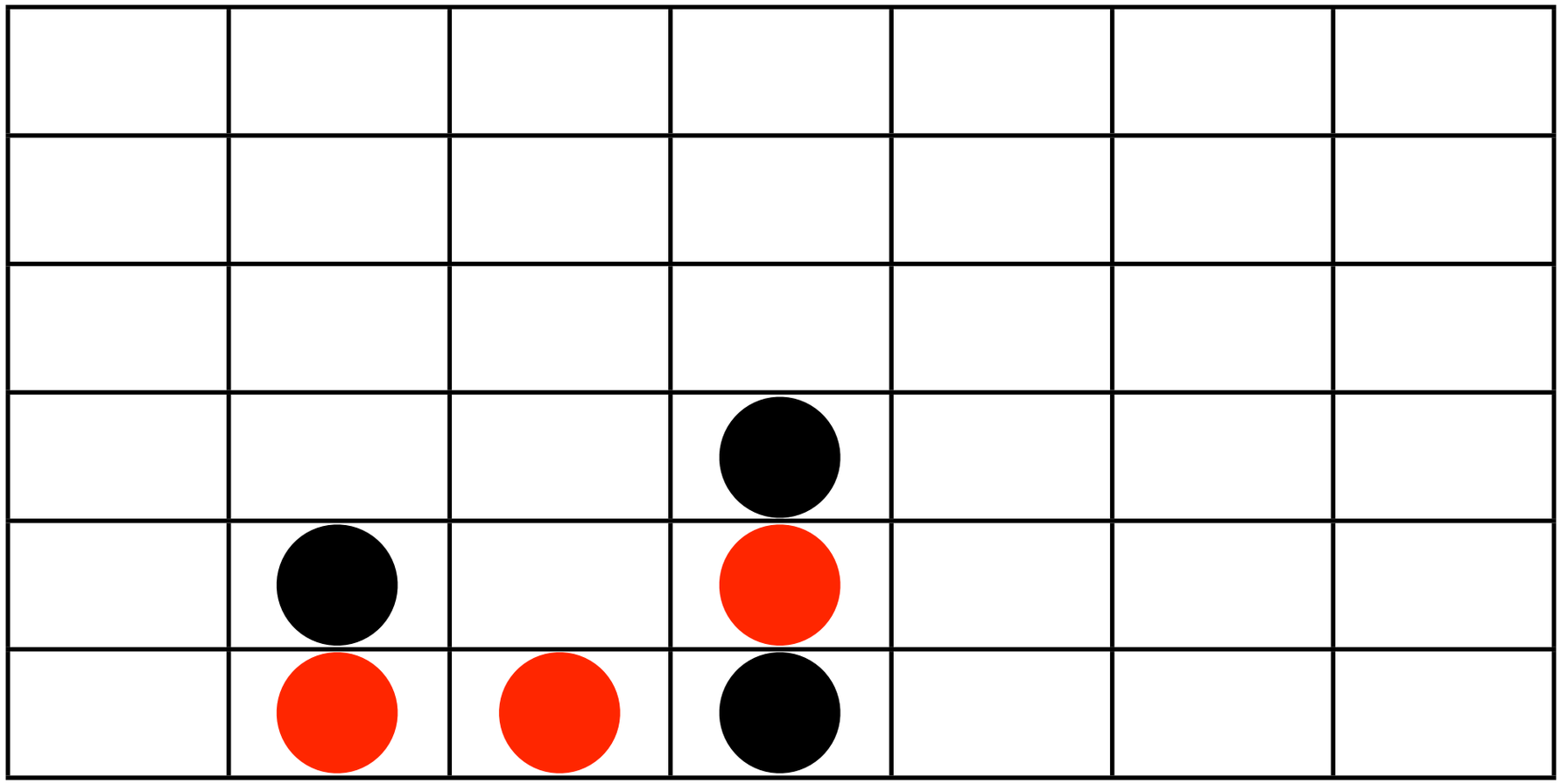
Restituisce la forma che descrive i primi 'dim' punti di un polinomio nella forma y = a2 * x^2 + a1 * x + a0

PIANO CARTESIANO

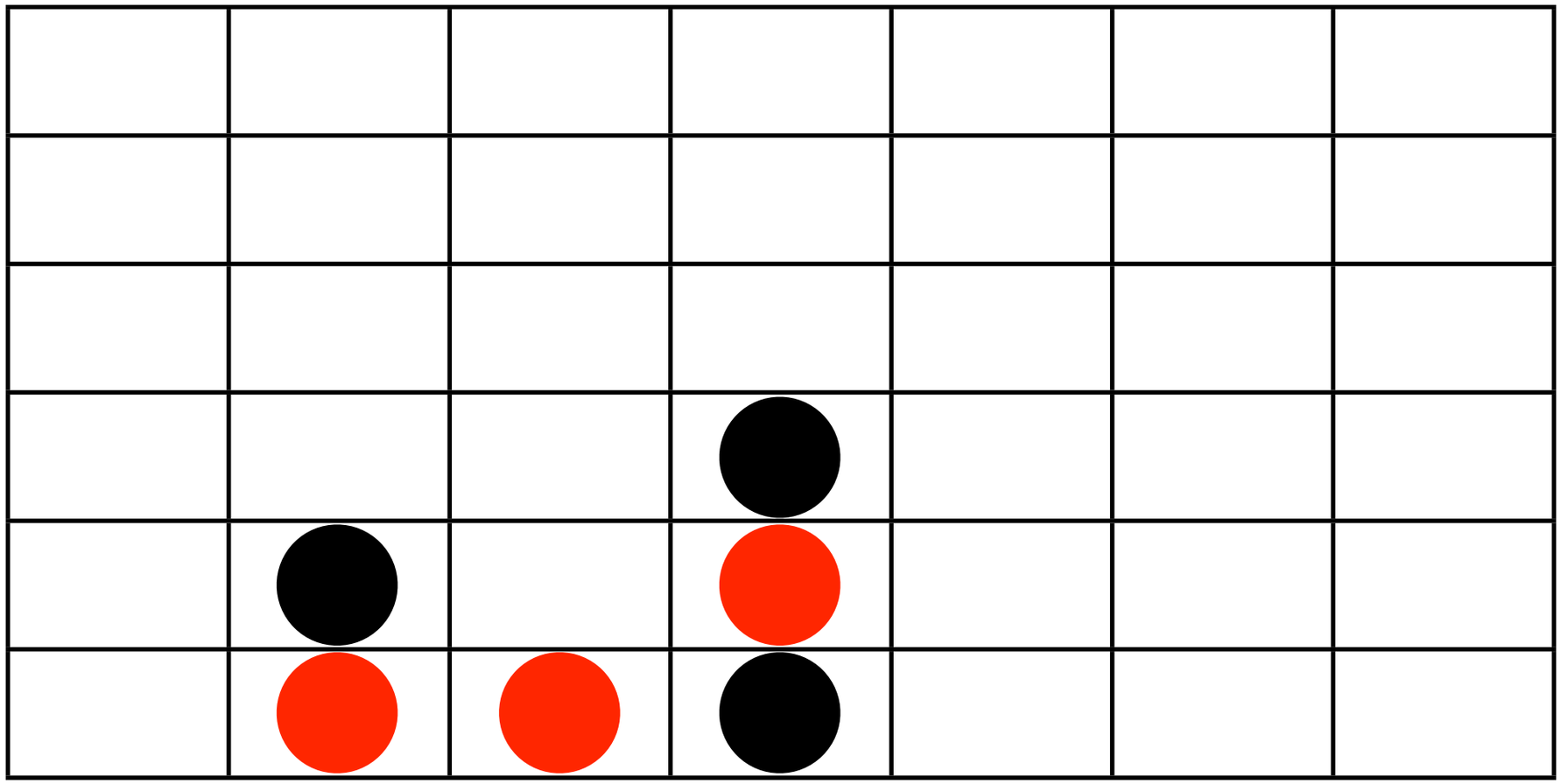
Un semplice videogioco

- Scriviamo un programma C che implementi il gioco del **Forza 4**
- Due giocatori, entrambi "reali"
- Il programma deve permettere di giocare
- Il programma deve annunciare il vincintore

Forza 4: rappresentazione



Forza 4: rappresentazione



Come facciamo a rappresentare nel terminale, con quello che già conosciamo, delle pedine rosse e delle pedine nere?

Forza 4: rappresentazione

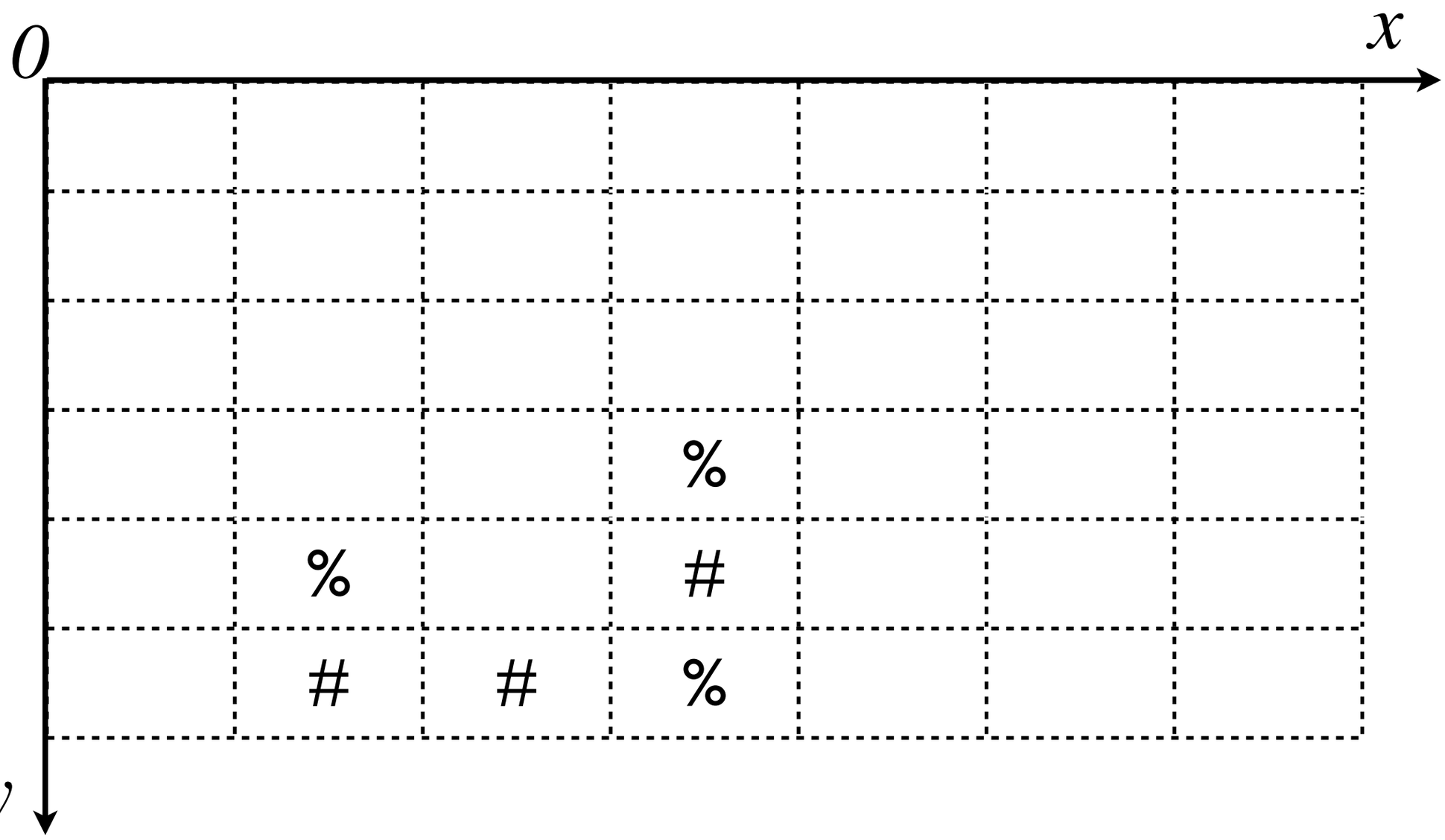
			%			
	%		#			
	#	#	%			

Forza 4: rappresentazione

			%			
	%		#			
	#	#	%			

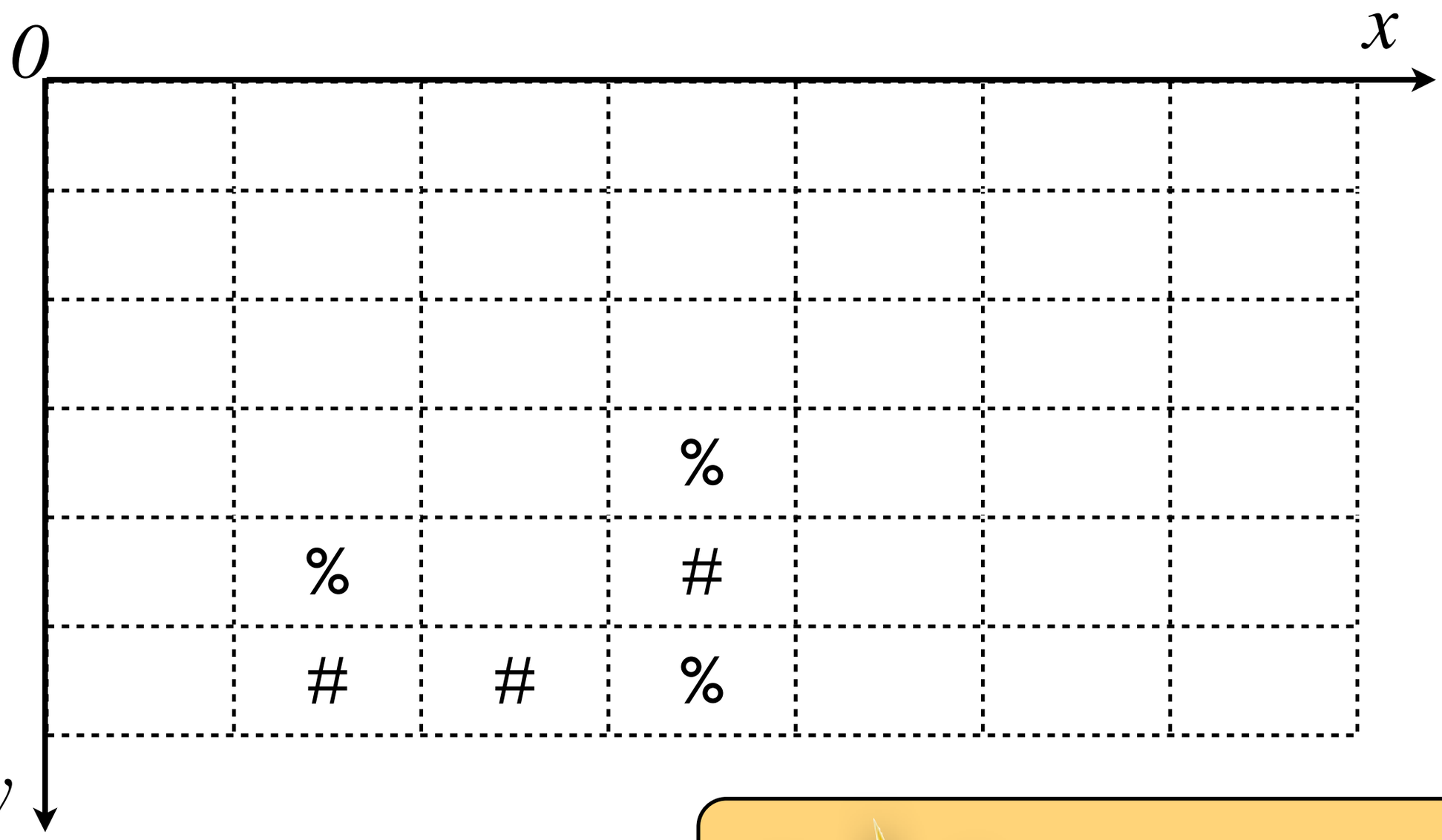
Quindi abbiamo una matrice,
descritta da due indici e
che contiene caratteri...

Forza 4: rappresentazione



Quindi abbiamo una matrice,
descritta da due indici e
che contiene caratteri...

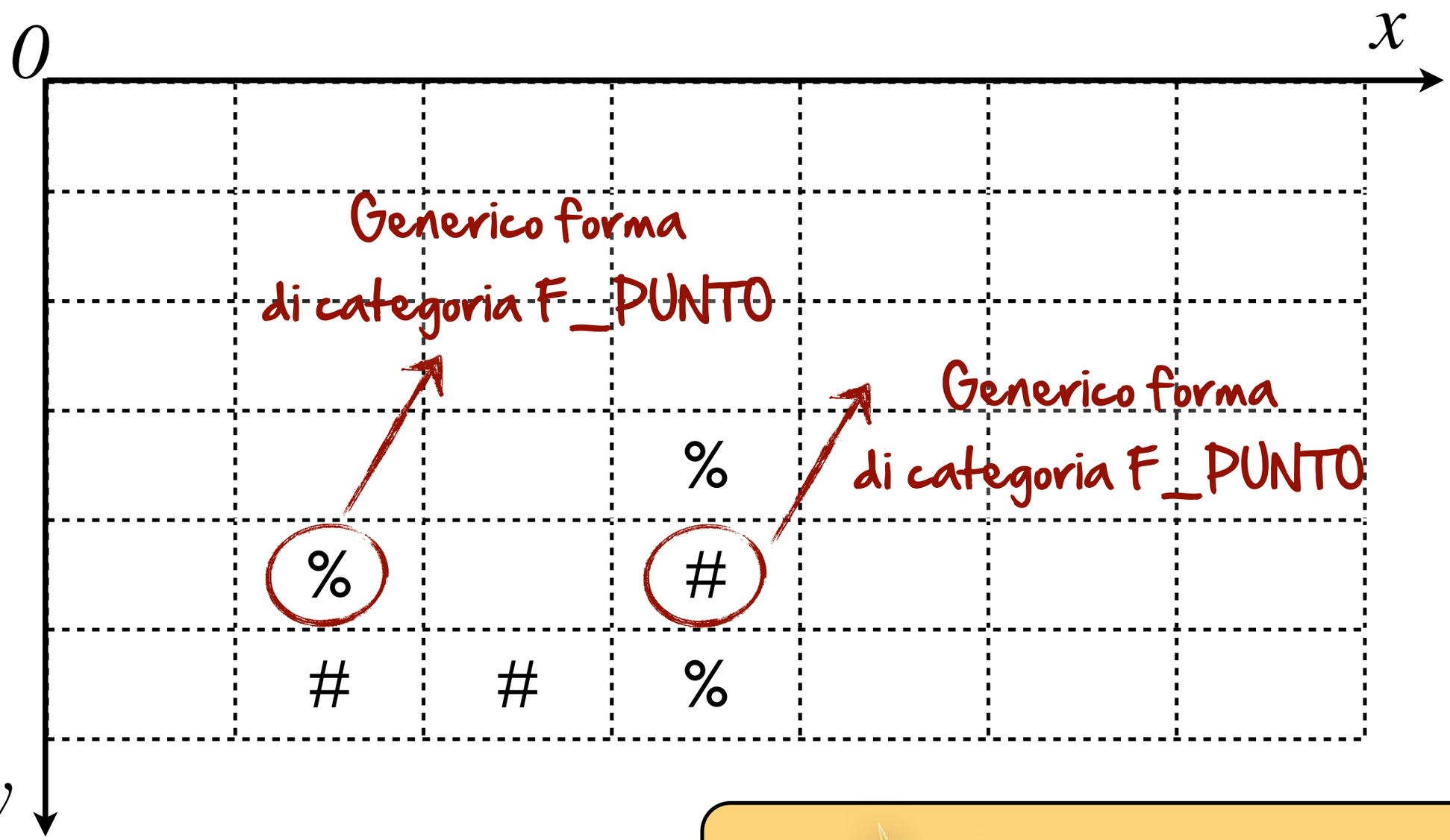
Forza 4: rappresentazione



Quindi abbiamo una matrice, descritta da due indici e che contiene caratteri...

Ma è il PIANO CARTESIANO!

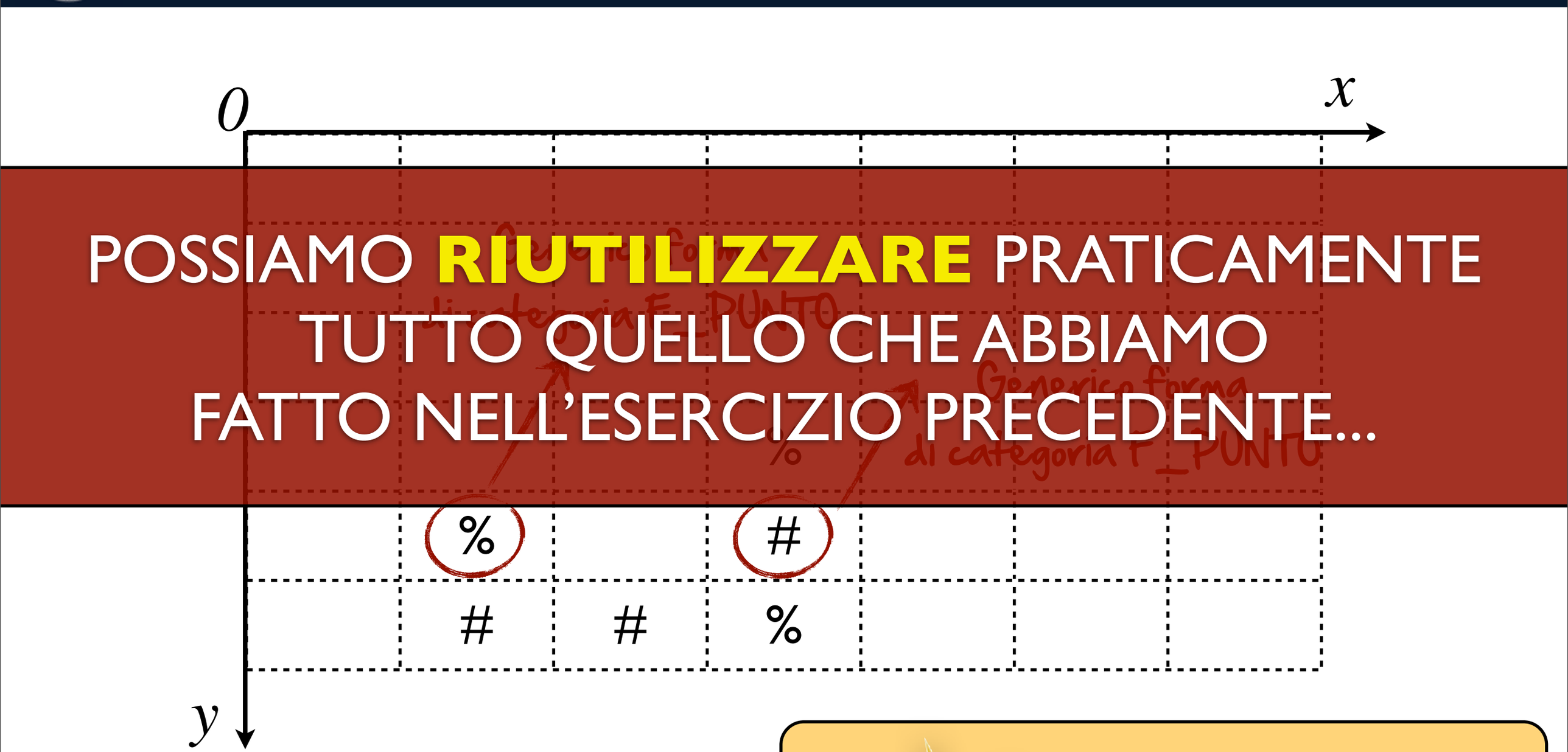
Forza 4: rappresentazione



Quindi abbiamo una matrice, descritta da due indici e che contiene caratteri...

Ma è il PIANO CARTESIANO!

Forza 4: rappresentazione



POSSIAMO **RIUTILIZZARE** PRATICAMENTE TUTTO QUELLO CHE ABBIAMO FATTO NELL'ESERCIZIO PRECEDENTE...

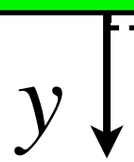
Quindi abbiamo una matrice, descritta da due indici e che contiene caratteri...

Ma è il PIANO CARTESIANO!

Forza 4: rappresentazione

POSSIAMO **RIUTILIZZARE** PRATICAMENTE TUTTO QUELLO CHE ABBIAMO FATTO NELL'ESERCIZIO PRECEDENTE...

ECCO A COSA SERVONO LE FUNZIONI! :D



Quindi abbiamo una matrice, descritta da due indici e che contiene caratteri...

Ma è il PIANO CARTESIANO!

Cosa manca

Cosa manca

- Anche se possiamo utilizzare buona parte del codice già scritto per il “PIANO CARTESIANO”, qualche concetto ancora ci manca:

Cosa manca

- Anche se possiamo utilizzare buona parte del codice già scritto per il “PIANO CARTESIANO”, qualche concetto ancora ci manca:
  - Come rappresentiamo un GIOCATORE?

Cosa manca

- Anche se possiamo utilizzare buona parte del codice già scritto per il “PIANO CARTESIANO”, qualche concetto ancora ci manca:
  - Come rappresentiamo un GIOCATORE?
  - Come gestiamo l’inserimento di una pedina?

Cosa manca

- Anche se possiamo utilizzare buona parte del codice già scritto per il “PIANO CARTESIANO”, qualche concetto ancora ci manca:
  - Come rappresentiamo un GIOCATORE?
  - Come gestiamo l’inserimento di una pedina?
  - Come controlliamo la vincita?

Il giocatore

Il giocatore

- Un giocatore è identificato da:

Il giocatore

- Un giocatore è identificato da:
  - IL SUO NOME

Il giocatore

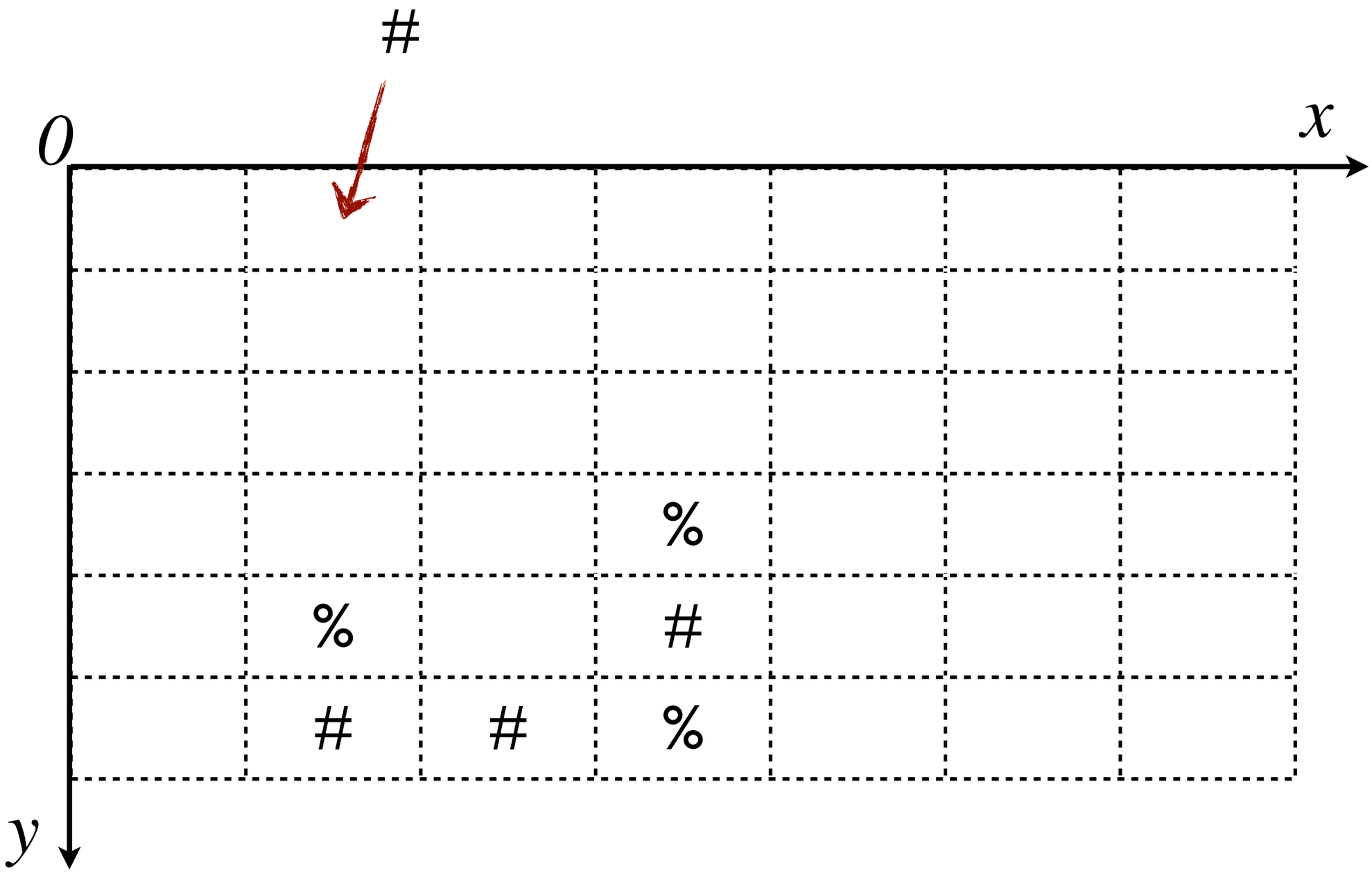
- Un giocatore è identificato da:
  - IL SUO NOME
  - QUALE PEDINA HA SCELTO

Il giocatore

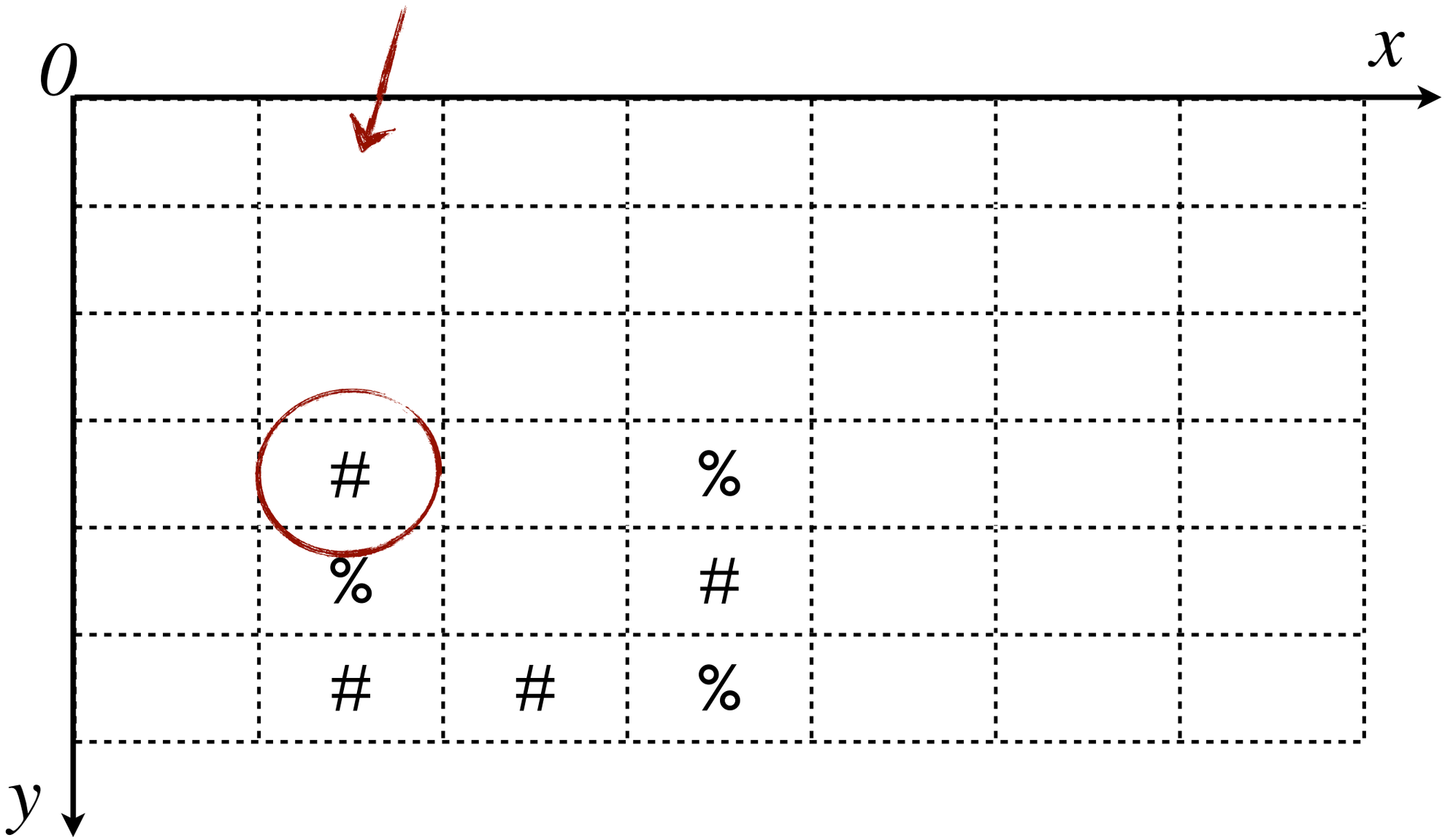
- Un giocatore è identificato da:
  - IL SUO NOME
  - QUALE PEDINA HA SCELTO

```
typedef struct {
    char nome[MAX_NOME_GIOCATORE];
    char simbolo_pedina;
} giocatore;
```

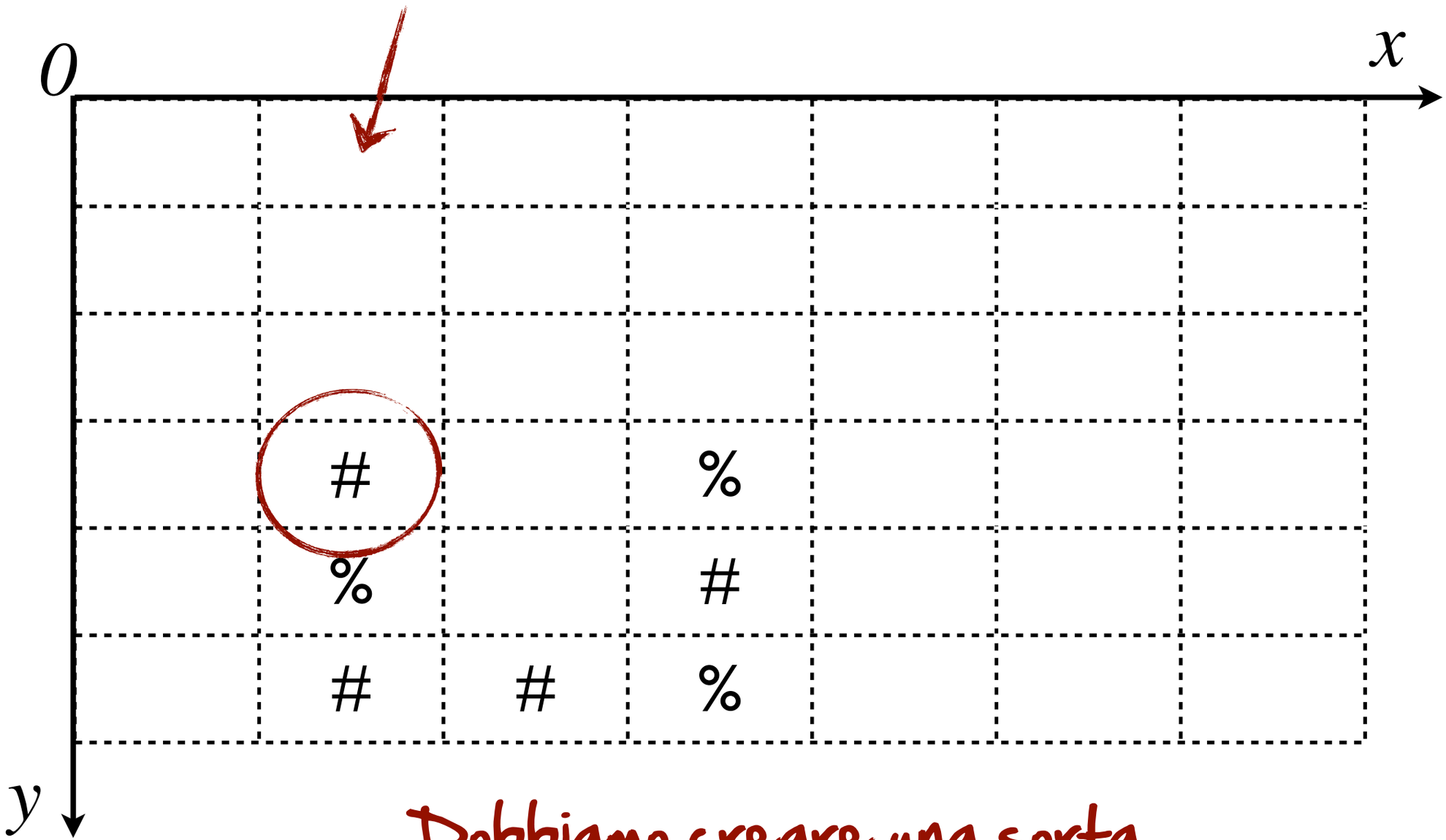
L’inserimento di una pedina



L’inserimento di una pedina



L’inserimento di una pedina

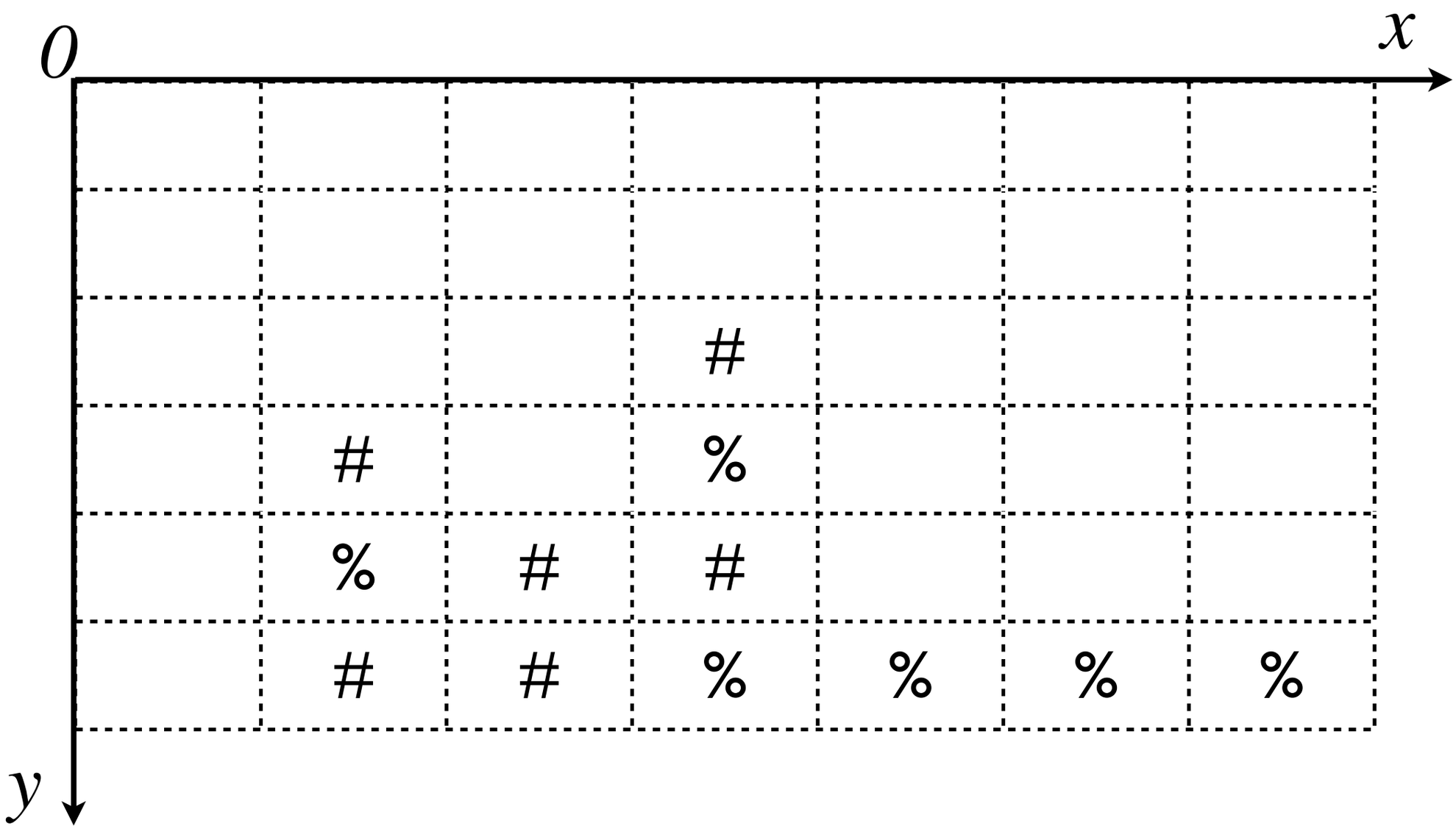


Dobbiamo creare una sorta di effetto gravità!

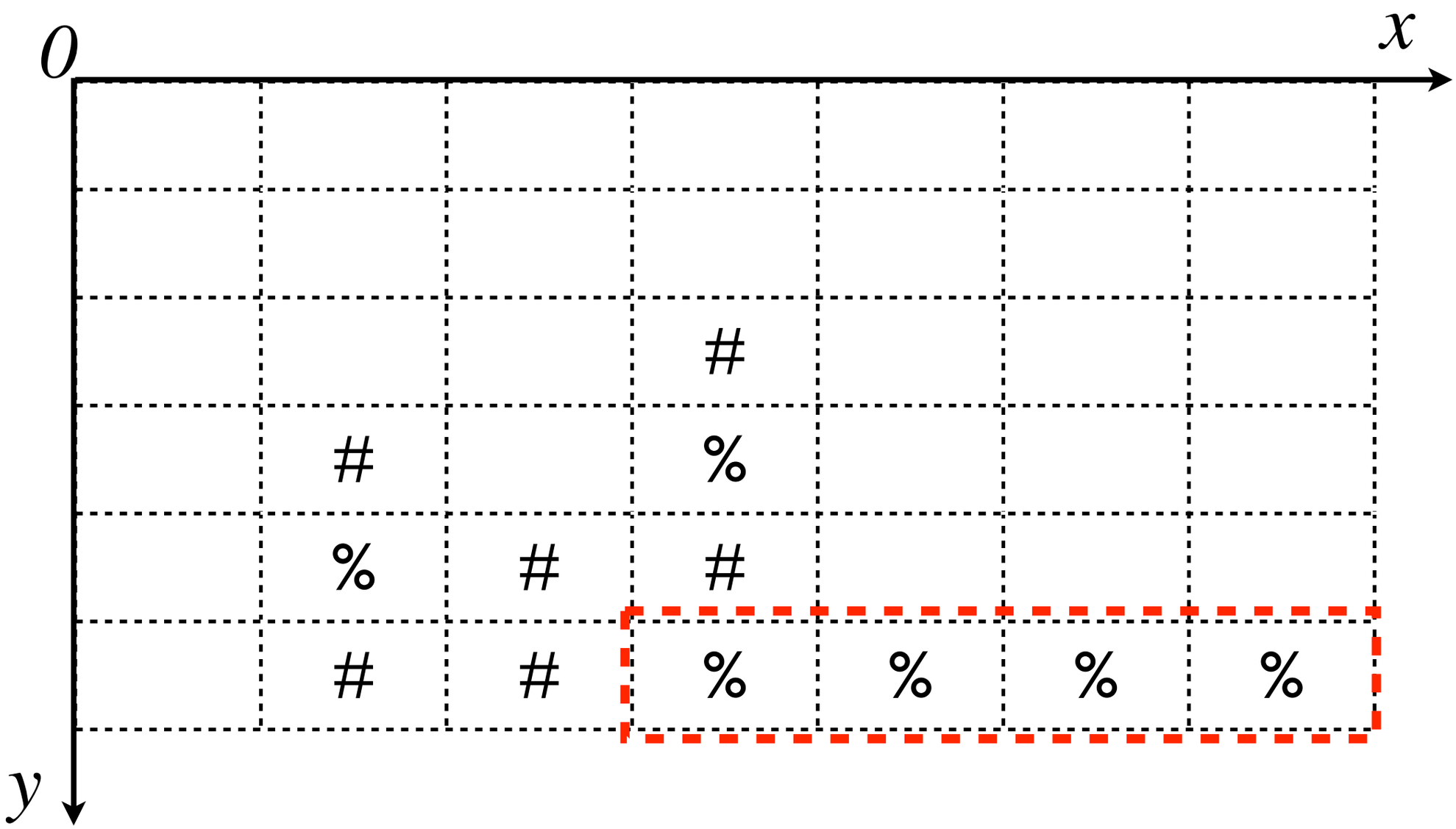
L'inserimento di una pedina: codice C

```c
int inserisci_pedina(char schermo[SCREEN_W][SCREEN_H],
                     int colonna, char simbolo_pedina)
{
    int y;
    punto_schermo p;
    forma pedina;

    for (y = SCREEN_H-2; y >= 0; y--)
    {

        if (schermo[colonna][y] == ' ')
        {
            p = crea_punto_schermo(colonna, y, 0);
            pedina  = genera_punto(simbolo_pedina);
            disegna_forma(pedina, p, schermo);
            return 1;
        }
    }

    return 0;

}
```

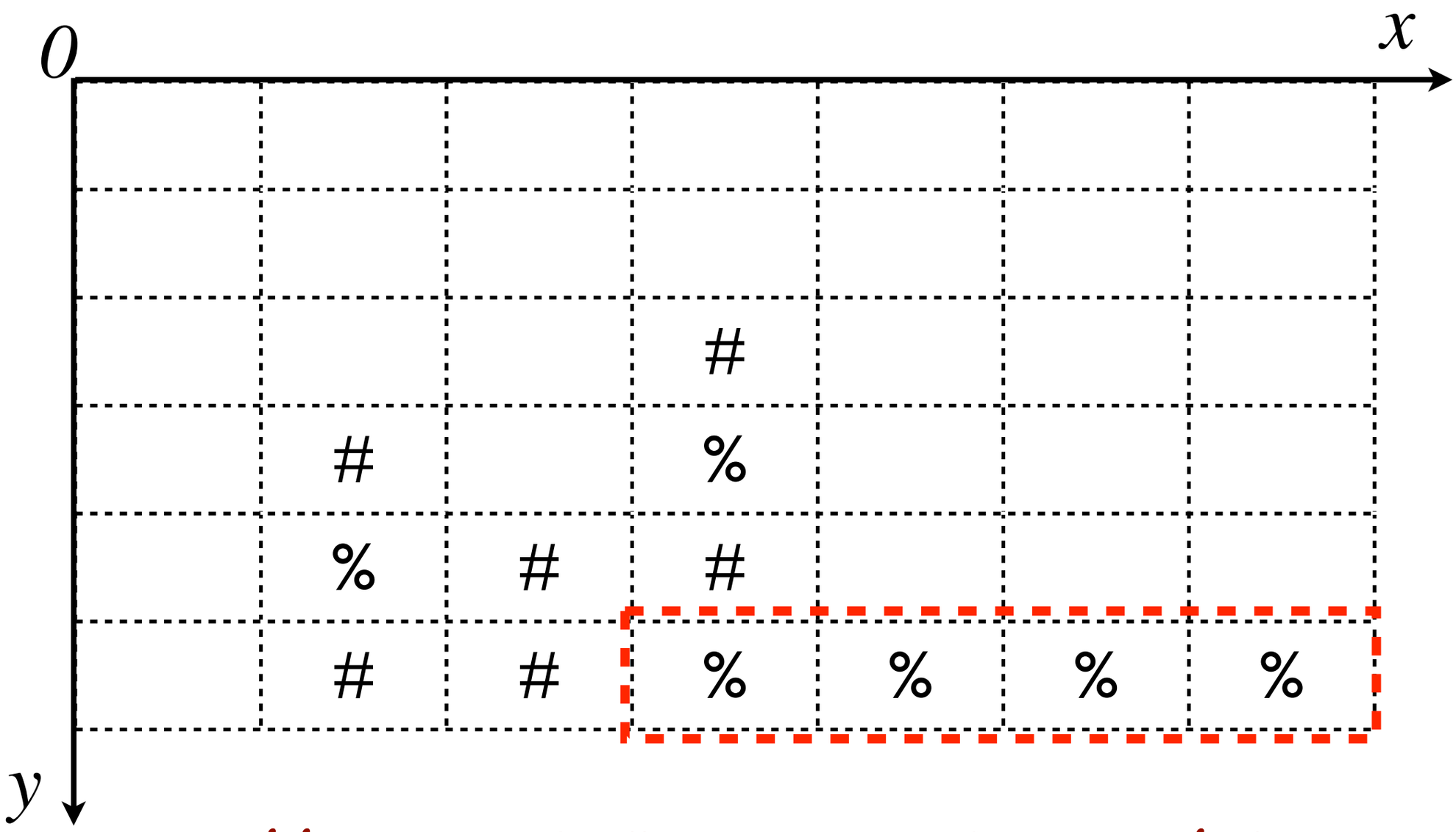
Verifica delle vincita



Verifica delle vincita



Verifica delle vincita



Dobbiamo controllare che, dato un simbolo, un vettore contiene 4 occorrenze di quest'ultimo CONSECUTIVE!

Verifica della vincita

- Per verificare una vincinta, dobbiamo prima di tutto saper controllare se, dato un vettore generico, esistono FORZA=4 occorrenze consecutive di un carattere '*occorrenza*' dato in ingresso

```
int controlla_vettore(char* vettore, int dim_vettore, char occorrenza)
```

- Con questa funzione potremo poi controllare sia vettori verticali, che orizzontali che obliqui

Controllo di un generico vettore

- Se ho un vettore A, e voglio controllare se qualcuno ha vinto, potrei usare il seguente metodo
  - Parto dal secondo elemento e, per ogni elemento
    - Se l'elemento A[i] considerato è uguale al carattere *occorrenza* ed è uguale all'elemento A[i-1], incremento un contatore *occorrenze* (occorrenze++)
    - Se l'elemento A[i] non è uguale ad occorrenza oppure l'elemento A[i] != A[i-1], occorrenze viene riportato a 0 (occorrenze=0)
  - Ogni qual volta che *occorrenze* arriva a FORZA-1, nel nostro caso 4-1, vale a dire 3, ho verificato che c'è una vincita
  - Se non succede mai che *occorrenze* == *FORZA-1*, allora non c'è nessuna vincita

Controllo di un generico vettore: codice C

```c
int controlla_vettore(char* vettore, int dim_vettore, char occorrenza)
{


    int i;
    int occorrenze = 0;

    for (i = 1; i < dim_vettore; i++){

        if (vettore[i] == vettore[i-1] && vettore[i] == occorrenza)
        {
            occorrenze++;
            if (occorrenze == FORZA-1)
                return 1;
        }
        else
        {
            occorrenze = 0;
        }
    }
    return 0;
}
```

Potete lasciare il vostro giudizio qui:

http://tinyurl.com/IEIMExe2013

Tutte il materiale sarà disponibile sul mio sito internet:

alessandronacci.com