

IEIM 2019-2020

Esercitazione VI ***“Puntatori, Matrici, Funzioni”***

Alessandro A. Nacci

alessandro.nacci@polimi.it - www.alessandronacci.it



Esempio: scambio valori



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

- Si scriva una funzione in C che, *dati* due valori interi, gli *restituisce* scambiati



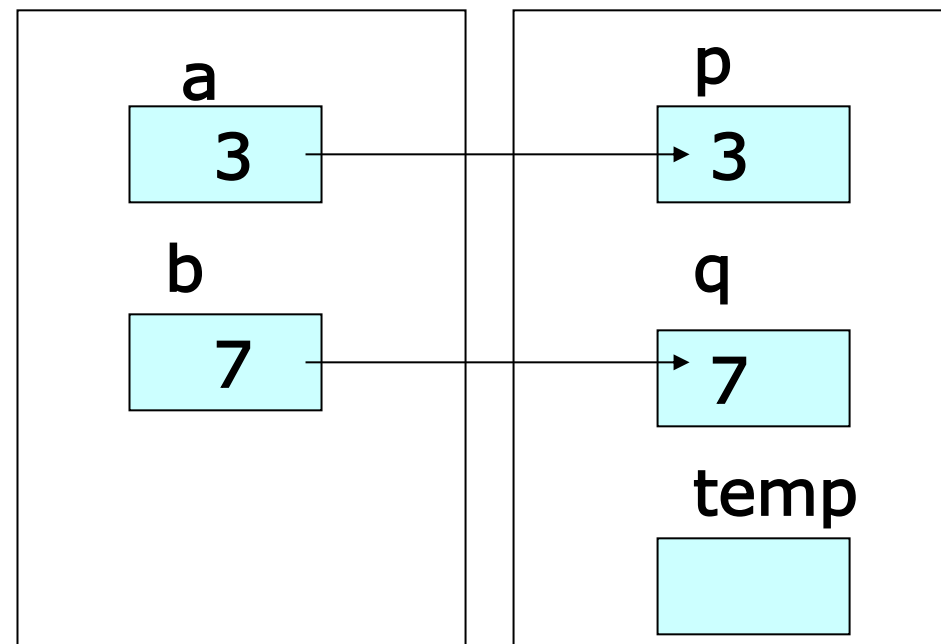
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {  
    int temp;  
    temp = p;  
    p = q;  
    q = temp;  
}
```



- Nel main: swap(a,b)



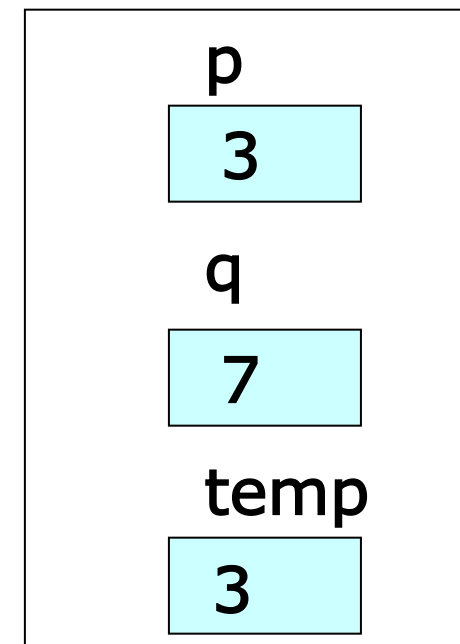
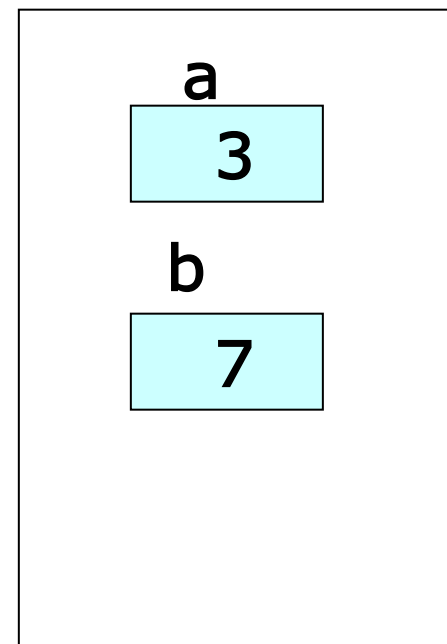
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {  
    int temp;  
    → temp = p;  
    p = q;  
    q = temp;  
}
```



- Nel main: `swap(a,b)`



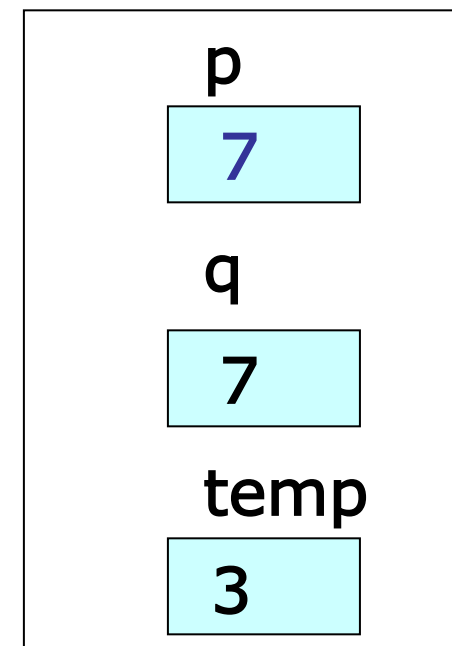
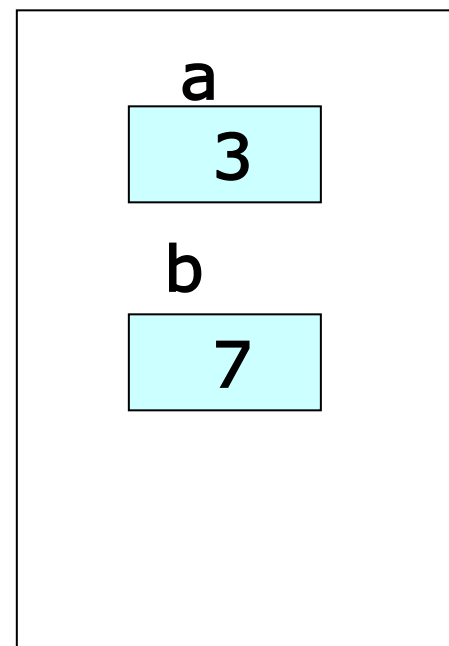
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {  
    int temp;  
    temp = p;  
    → p = q;  
    q = temp;  
}
```



- Nel main: `swap(a,b)`



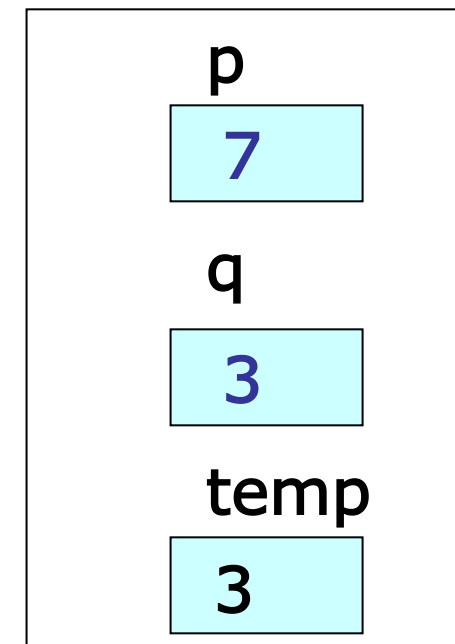
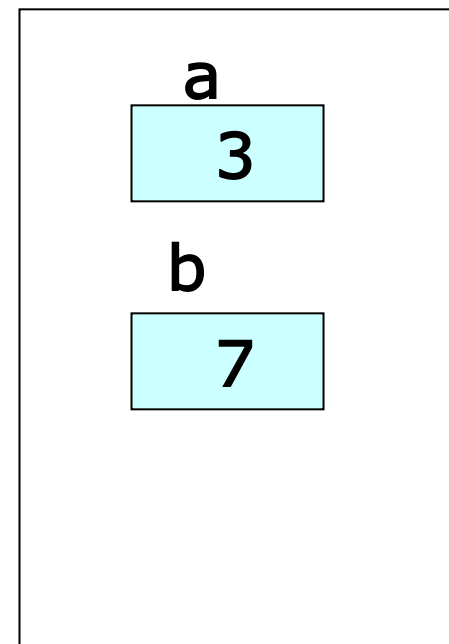
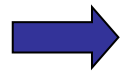
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {  
    int temp;  
    temp = p;  
    p = q;  
    q = temp;  
}
```



- Nel main: `swap(a,b)`



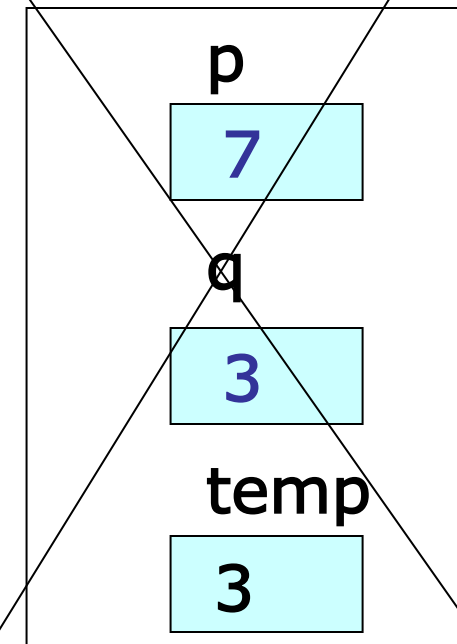
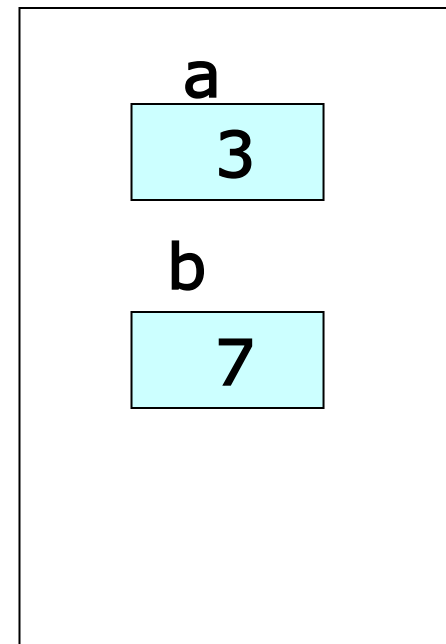
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int p, int q) {  
    int temp;  
    temp = p;  
    p = q;  
    q = temp;  
}
```



**Al termine
dell'esecuzione di
swap le variabili nel
main restano
inalterate!**

ap(a,b)



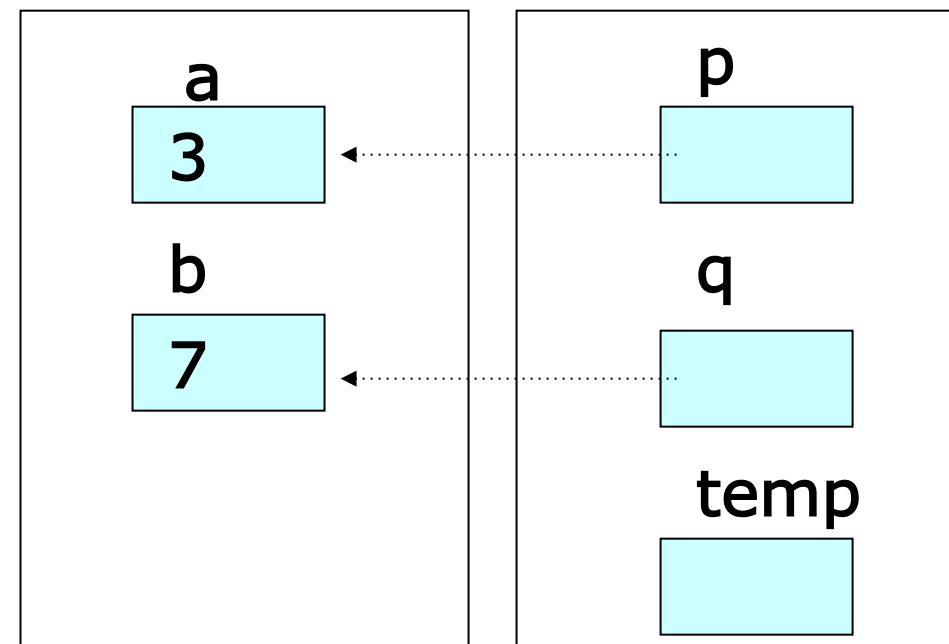
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){  
    int temp;  
    temp = *p;  
    *p = *q;  
    *q = temp;  
}
```



- Nel main: swap(&a, &b)



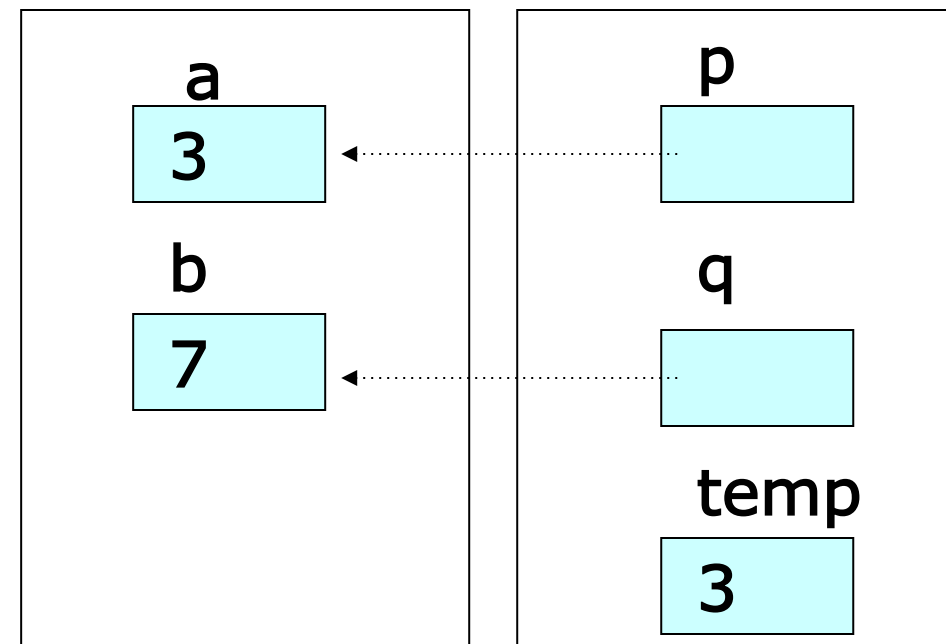
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){  
    int temp;  
    temp = *p;  
    *p = *q;  
    *q = temp;  
}
```



- Nel main: swap(&a, &b)



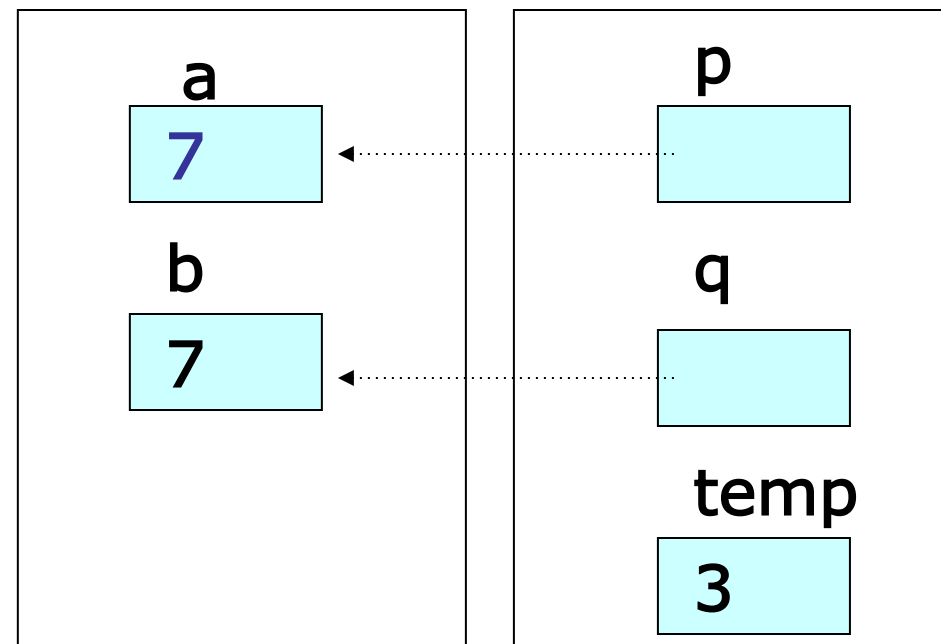
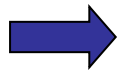
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){  
    int temp;  
    temp = *p;  
    *p = *q;  
    *q = temp;  
}
```



- Nel main: swap(&a, &b)



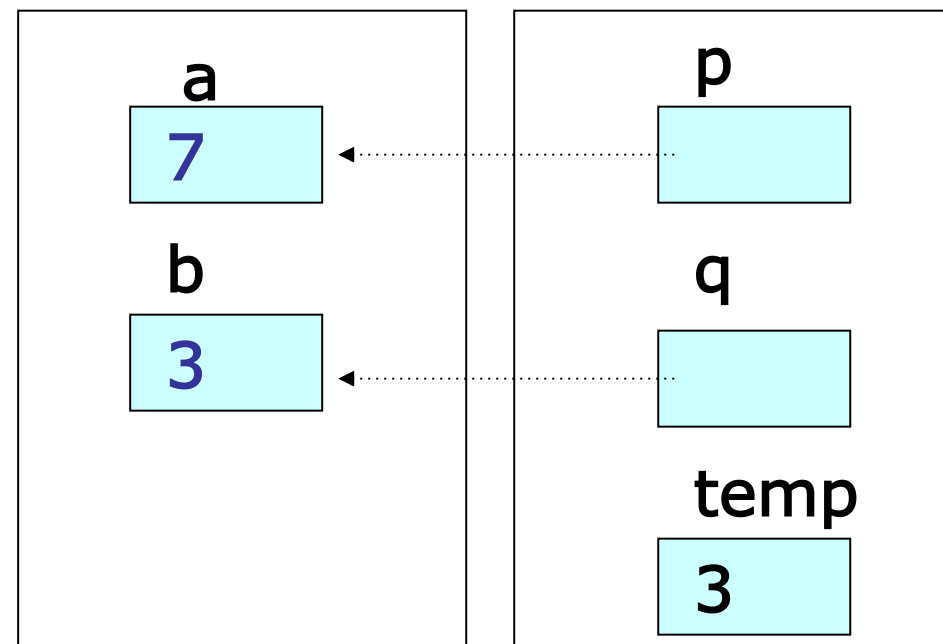
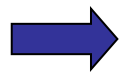
Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){  
    int temp;  
    temp = *p;  
    *p = *q;  
    *q = temp;  
}
```



- Nel main: swap(&a, &b)



Esempio: scambio di 2 valori interi



POLITECNICO
DI MILANO

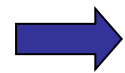
DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

```
void swap (int *p, int *q){
```

```
    int temp;
```

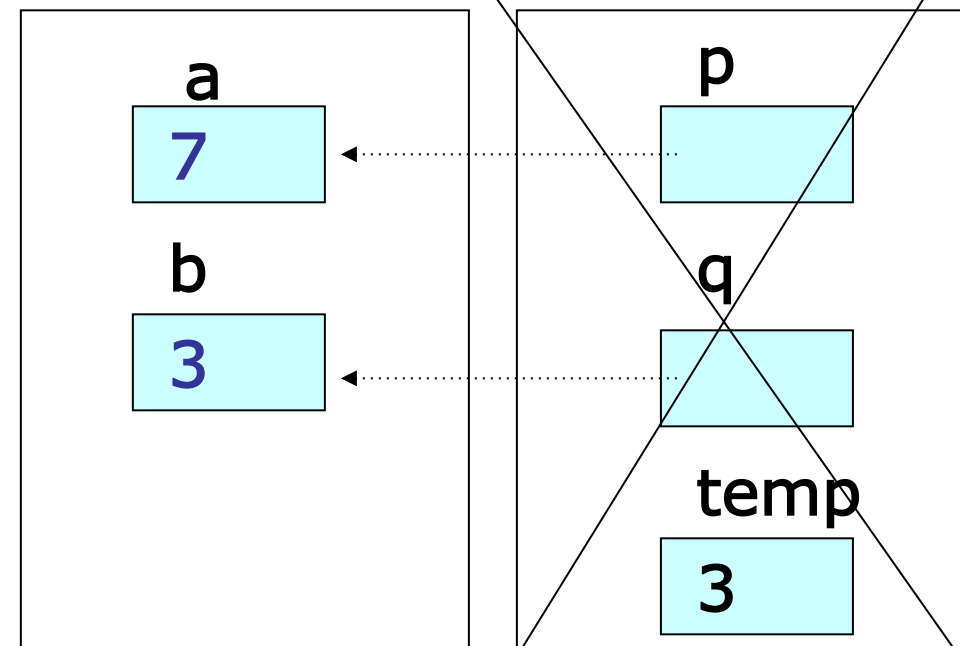
```
    temp = *p;
```

```
    *p = *q;
```



```
    *q = temp;
```

**Al termine
dell'esecuzione di
swap le variabili nel
main vengono
modificate**



```
swap(&a, &b)
```



Fonti per lo studio + Credits



POLITECNICO
DI MILANO

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE

- Fonti per lo studio
 - Binky Pointer Fun Video:
<http://cslibrary.stanford.edu/104/>
- Credits
 - Gianluca Palermo



WARNING

MATRICI E FUNZIONI

Il passaggio di una matrice ad una funzione
e' sempre per indirizzo e mai per copia.



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo1(int *mat)
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            *(mat+i*C+j) = (i+j) * 2;
}
```



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

Puntatore ad intero!

```
void foo1(int *mat)
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            *(mat+i*C+j) = (i+j) * 2;
}
```



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo1(int *mat)
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            *(mat+i*C+j) = (i+j) * 2;
}
```

Puntatore ad intero!

Calcolo manuale dello
spiazzamento



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo1(int *mat)
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            *(mat+i*C+j) = (i+j) * 2;
```

```
}
```

Puntatore ad intero!

Calcolo manuale dello
spiazzamento

Valore contenuto all'indirizzo tra parentesi

*(mat+i*C+j)

INDIRIZZO



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo2(int mat[R][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 3;
}
```



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

Esplicito che il puntatore
punta al primo elemento
di una matrice RxC

```
void foo2(int mat[R][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 3;
}
```



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

Esplicito che il puntatore
punta al primo elemento
di una matrice RxC

```
void foo2(int mat[R][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 3;
}
```

Accesso comodo alla matrice!



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo3(int mat[][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 4;
}
```



Esercizio I: Matrici e Funzioni

Esplicito che il puntatore
punta al primo elemento
di una matrice con C colonne

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

```
void foo3(int mat[][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 4;
}
```



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

Esplicito che il puntatore
punta al primo elemento
di una matrice con C colonne

```
void foo3(int mat[][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 4;
}
```

Accesso comodo alla matrice!



Esercizio I: Matrici e Funzioni

```
#include <stdio.h>

#define R 3
#define C 3

void foo1(int*);
void foo2(int mat[R][C]);
void foo3(int mat[][C]);

int main()
{
    int mat[R][C];
    foo1(mat);
    foo2(mat);
    foo3(mat);
}
```

Esplicito che il puntatore
punta al primo elemento
di una matrice con C colonne

```
void foo3(int mat[][C])
{
    int i, j;
    for(i=0; i<R; i++)
        for(j=0; j<C; j++)
            mat[i][j] = (i+j) * 4;
}
```

Accesso comodo alla matrice!

NOTA BENE

Ai fini del calcolo dello
spiazzamento il numero di
righe non è essenziale!

Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

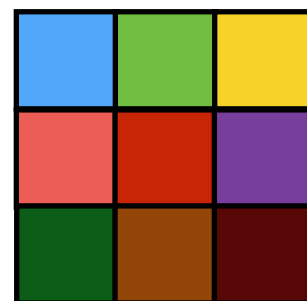
```
    for(i=0; i<R; i++)
```

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!



NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!



Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

x	x	x

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

--	--	--	--	--	--	--	--	--

Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

X	X	X

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

									X	X	X
--	--	--	--	--	--	--	--	--	---	---	---

Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

Il valore di mat cambia

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

X	X	X

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

									X	X	X
--	--	--	--	--	--	--	--	--	---	---	---

Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

Il valore di mat cambia

Il valore di mat cambia

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

X	X	X

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

									X	X	X
--	--	--	--	--	--	--	--	--	---	---	---

Esercizio I: Matrici e Funzioni

Esplicito che il puntatore punta al primo elemento di una matrice con C colonne

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

```
#define R 3
```

```
#define C 3
```

```
void foo1(int*);
```

```
void foo2(int mat[R][C]);
```

```
void foo3(int mat[][C]);
```

```
int main()
```

```
{
```

```
    int mat[R][C];
```

```
    foo1(mat);
```

```
    foo2(mat);
```

```
    foo3(mat);
```

```
}
```

Il valore di mat cambia

Il valore di mat cambia

Il valore di mat cambia

```
void foo3(int mat[][C])
```

```
{
```

```
    int i, j;
```

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

```
        for(j=0; j<C; j++)
```

```
            mat[i][j] = (i+j) * 4;
```

```
}
```

Accesso comodo alla matrice!

X	X	X

NOTA BENE

Ai fini del calcolo dello spiazzamento il numero di righe non è essenziale!

									X	X	X
--	--	--	--	--	--	--	--	--	---	---	---



```
somma_speciale(int a, int b)
{
    int q = 3;
    return d + c + q;
}
```

```
int main()
{
    int a;
    int b;
    int c;

    a = 2;
    b = 5;
```

```
somma_speciale(a,b);
}
```

	Address	Value	Notes
1			
2	0	2	a
3	1	5	b
4	2	10	c
5	3	10	somma_speciale/return
6	4	2	somma_speciale/a
7	5	5	somma_speciale/b
8	6	3	somma_speciale/q
9	7		



Esercizio I

Cosa fa il seguente codice?

Cosa fa il seguente codice?

```
int main() {int a;int b;int  
c;for(a=0;a<10;a++) {if(a==2)  
{for(b=10;b>0;b--) {if(a==5&&b==3)  
{c==1;while(c==<10) {printf("The  
value of c is %d",c);c++;}}}}}}
```

Cosa fa il seguente codice?

```
int main(){
int a;
int b;
int c;
for (a = 0; a < 10; a++){
if (a == 2){for (b = 10; b>0; b--){
if (a == 5 && b == 3){
c == 1;
while(c == < 10){
printf("The value of c is %d", c);
c++;
}}}}}}}
```

Cosa fa il
seguinte
codice?

```
int main()
{
    int a;
    int b;
    int c;

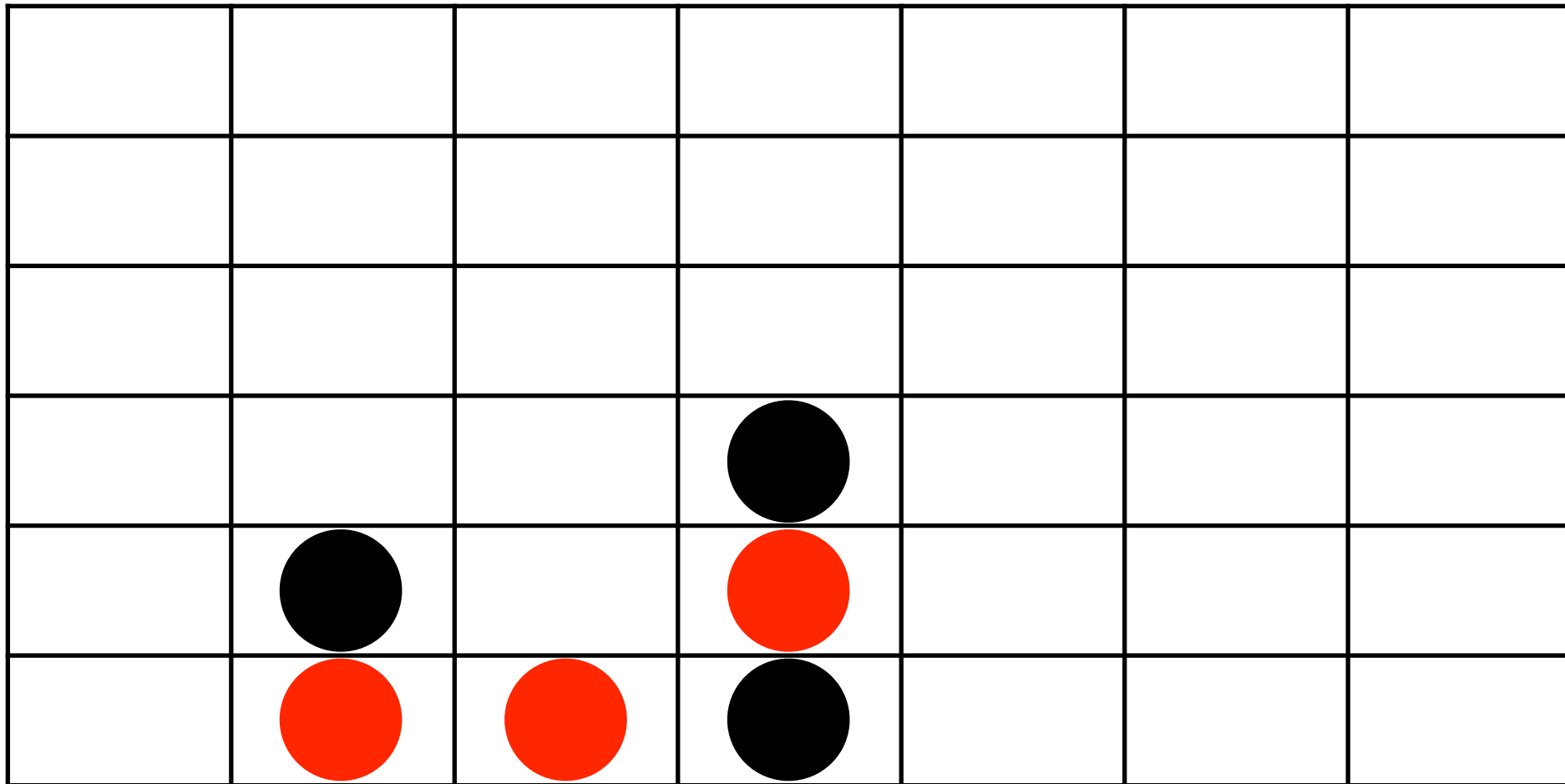
    for (a = 0; a < 10; a++)
    {
        if (a == 2)
        {
            for (b = 10; b>0; b--)
            {
                if (a == 5 && b == 3)
                {
                    c == 1;
                    while(c == < 10)
                    {
                        printf("The value of c is %d", c);
                        c++;
                    }
                }
            }
        }
    }
}
```

Un semplice videogioco

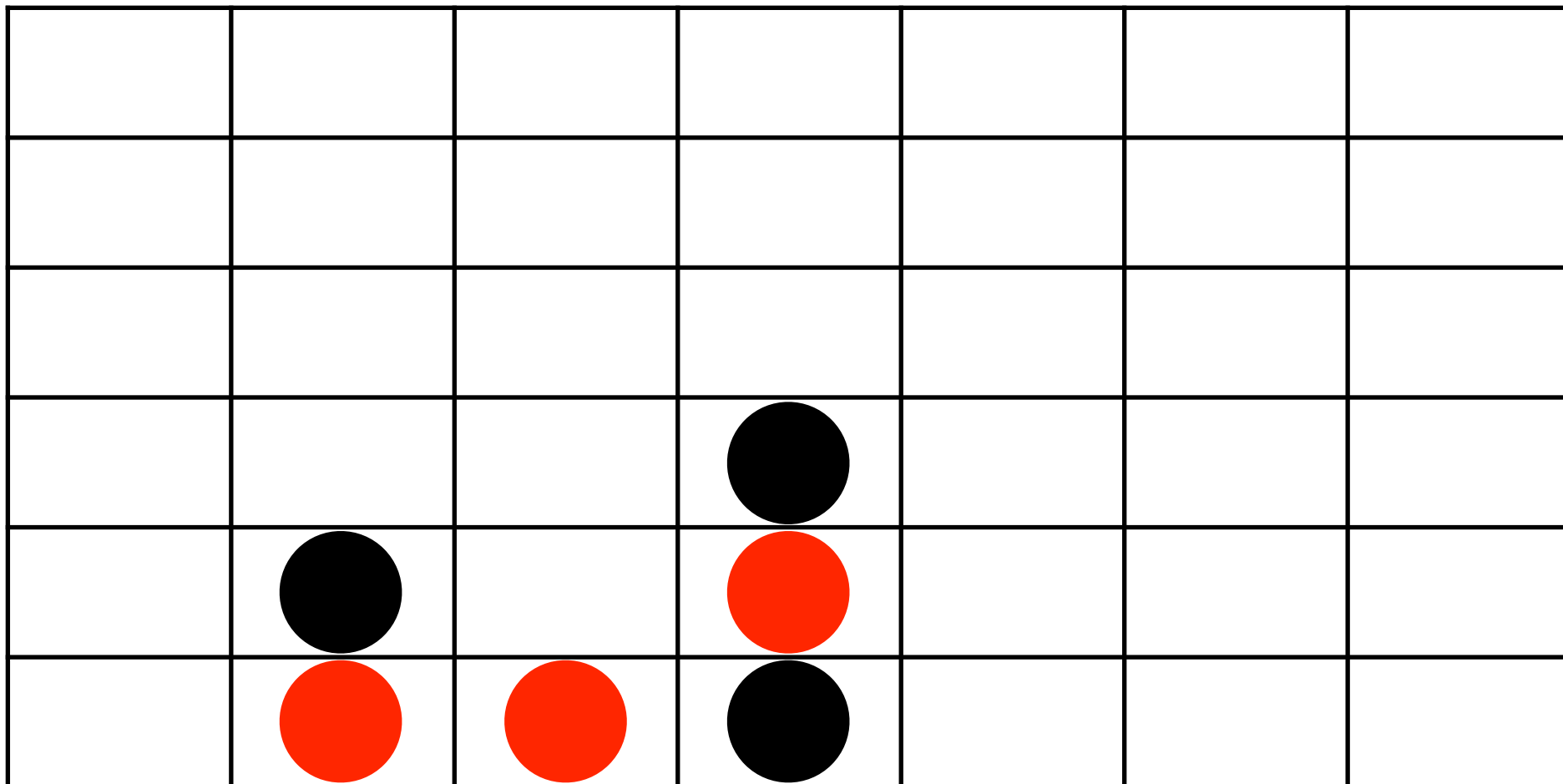
- Scriviamo un programma C che implementi il gioco del **Forza 4**
- Due giocatori, entrambi “reali”
- Il programma deve permettere di giocare
- Il programma deve annunciare il vincitore



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

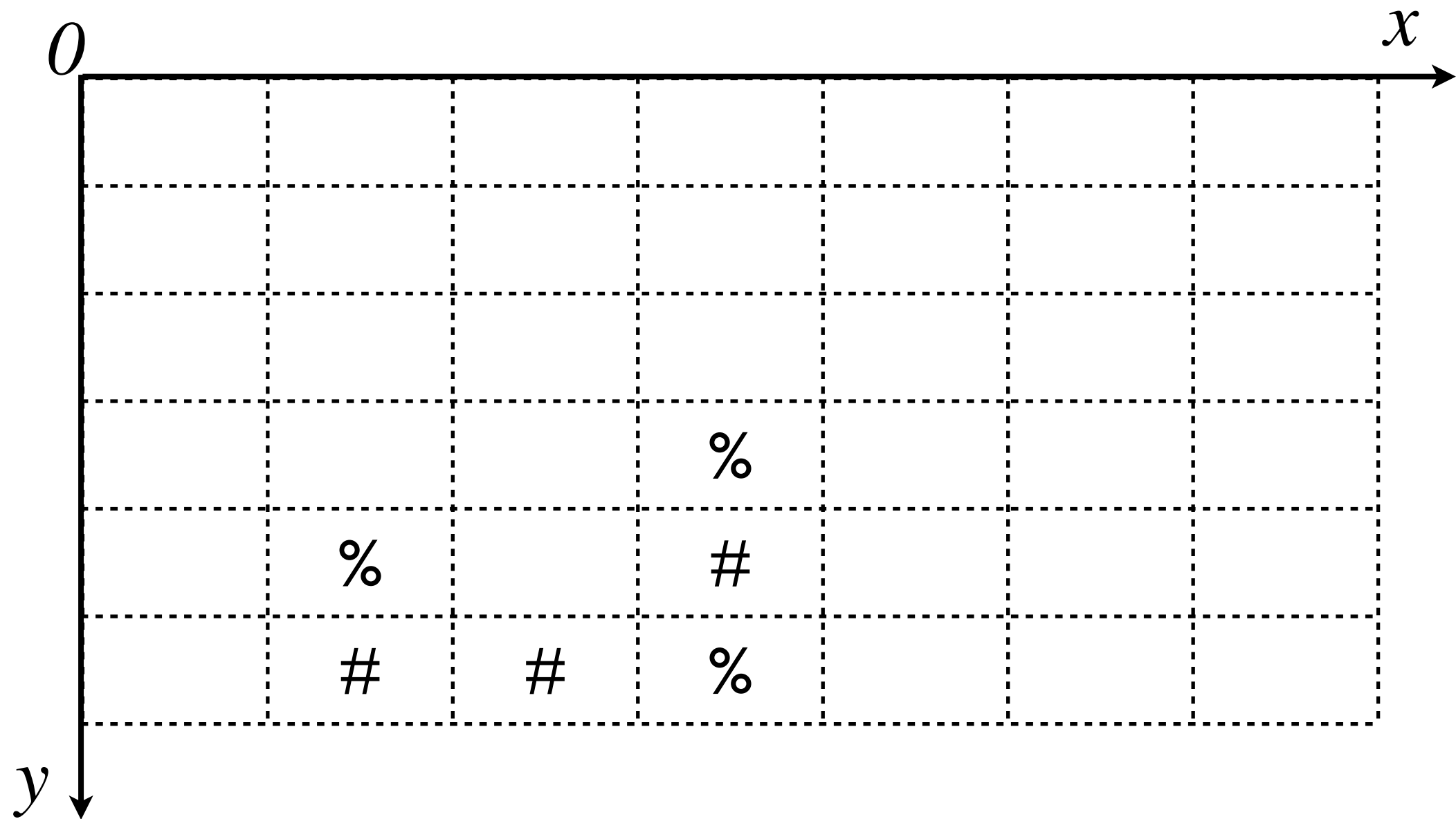
			%			
	%		#			
	#	#	%			

Forza 4: rappresentazione

			%			
	%		#			
	#	#	%			

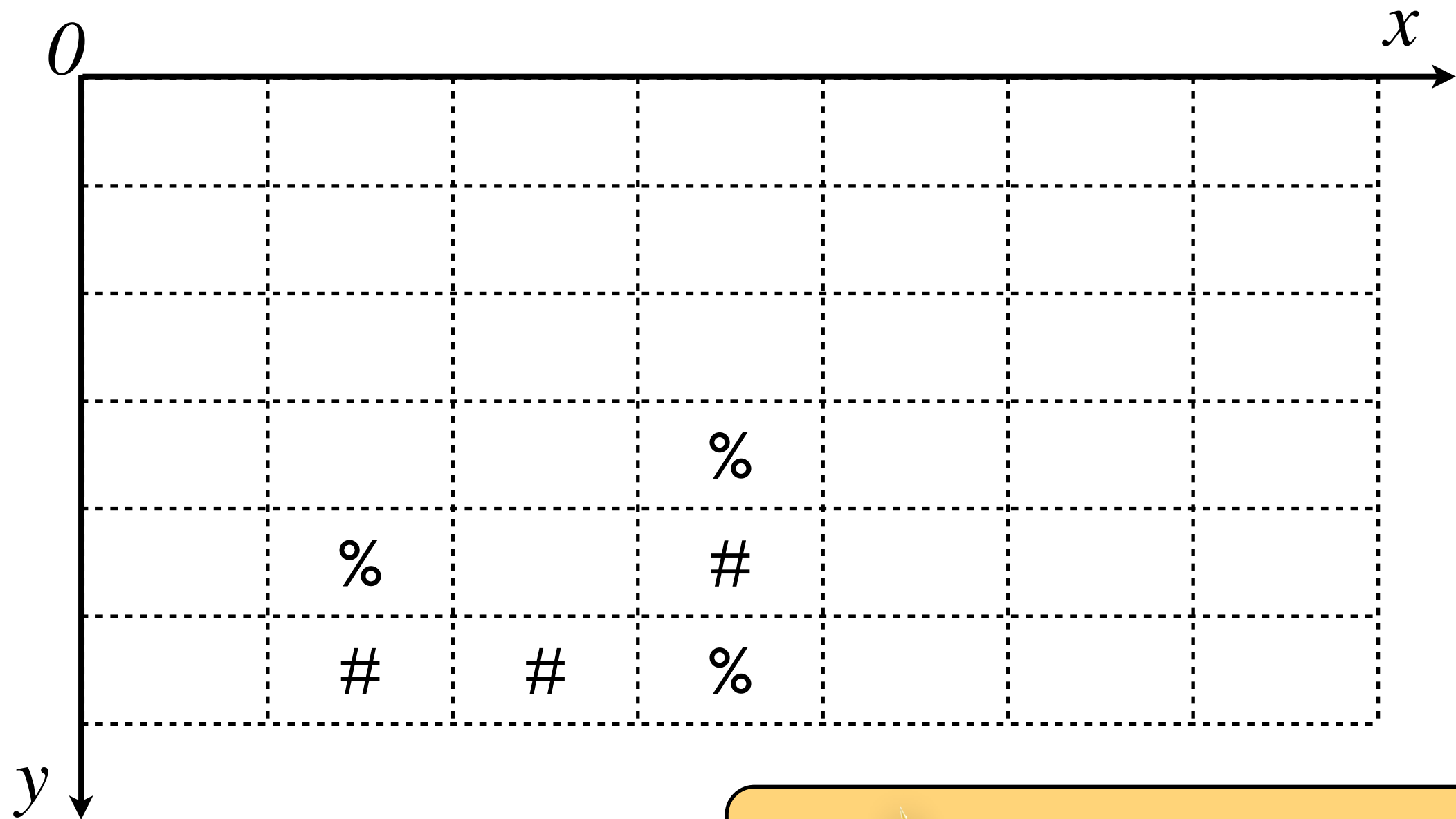
Quindi abbiamo una matrice,
descritta da due indici e che
contiene caratteri...

Forza 4: rappresentazione



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

Forza 4: rappresentazione

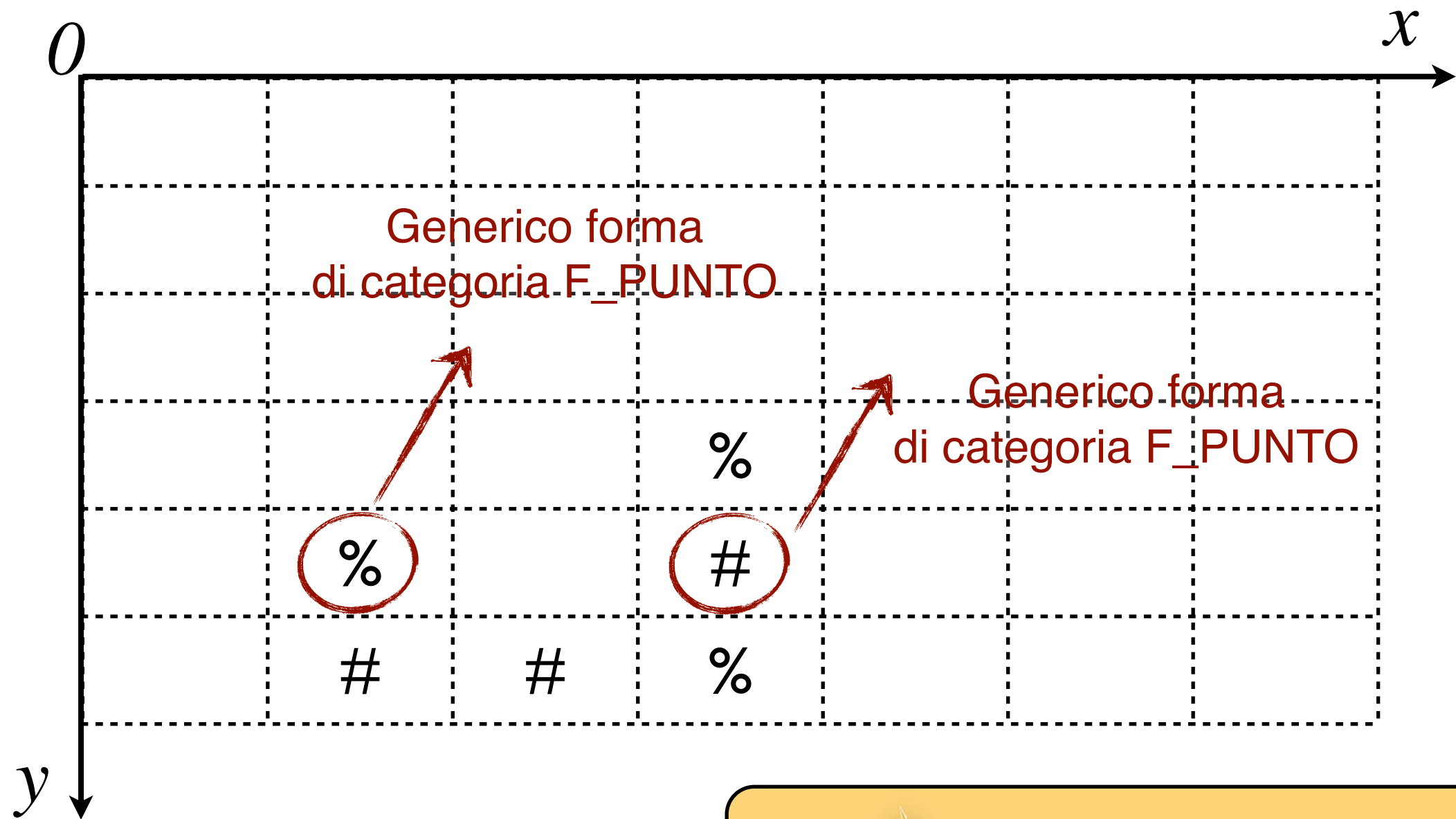


Quindi abbiamo una matrice,
descritta da due indici e che
contiene caratteri...



Ma è il PIANO
CARTESIANO!

Forza 4: rappresentazione



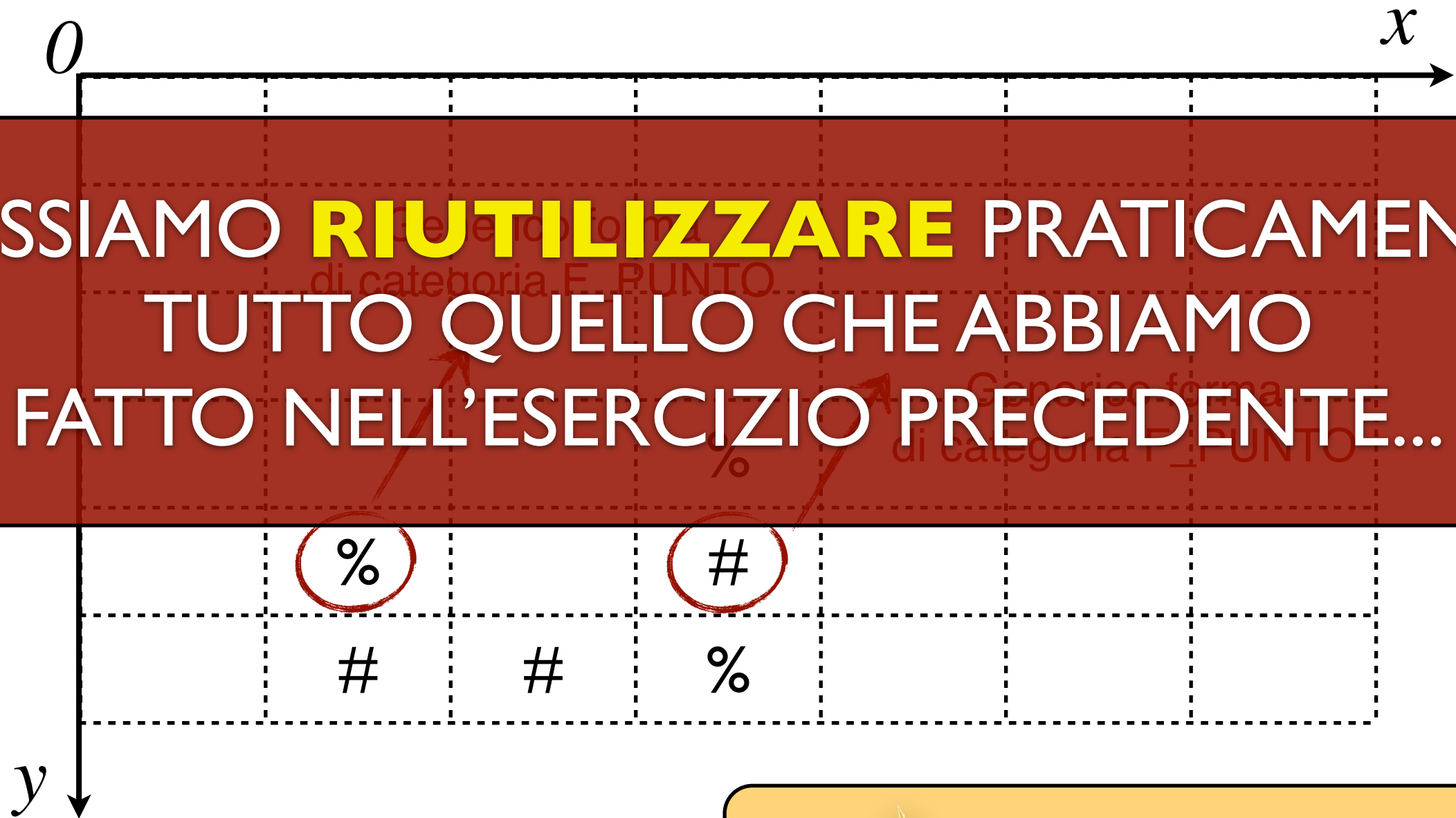
Quindi abbiamo una matrice, descritta da due indici e che contiene caratteri...



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

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

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

- 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?

- 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



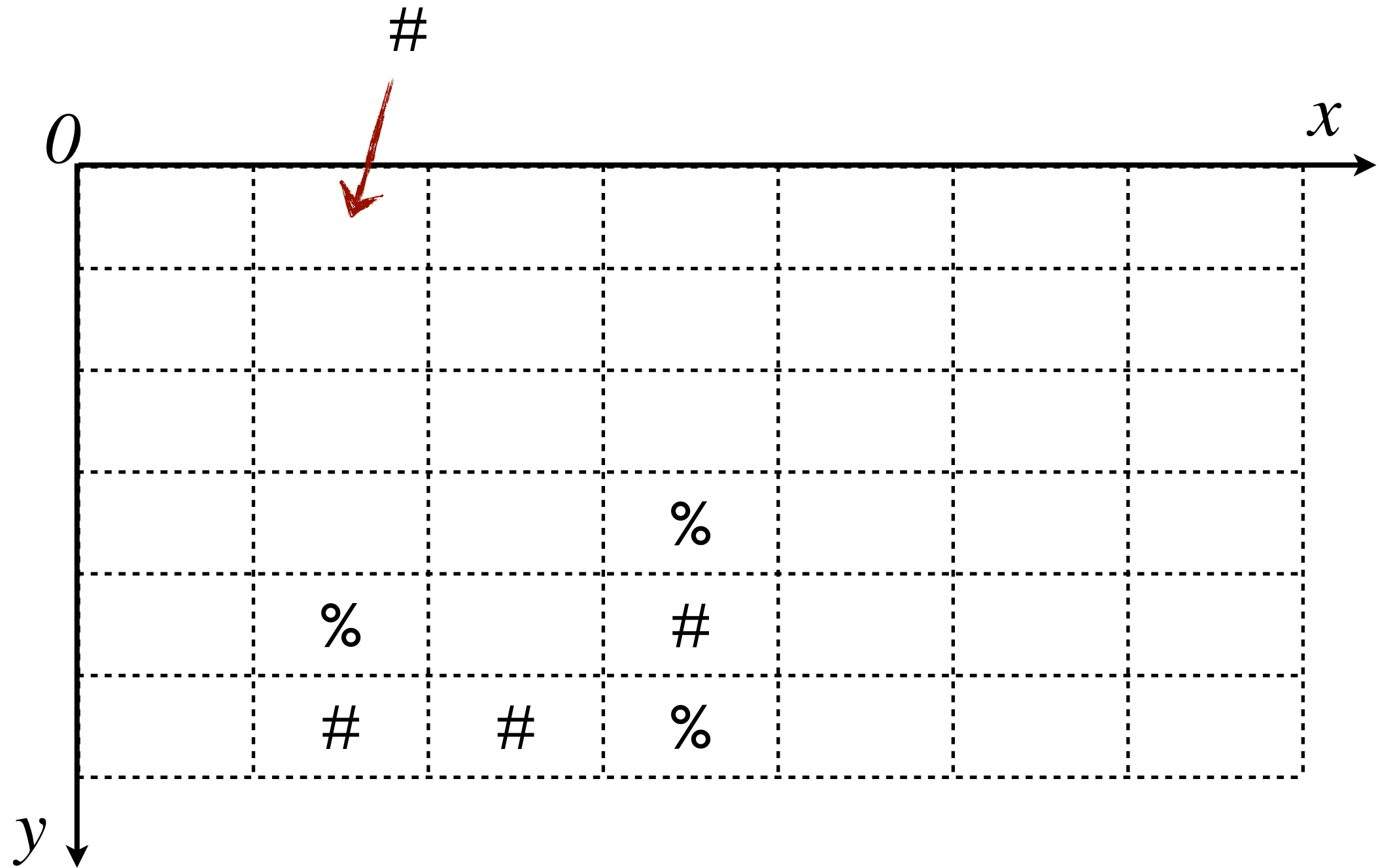
Il giocatore

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

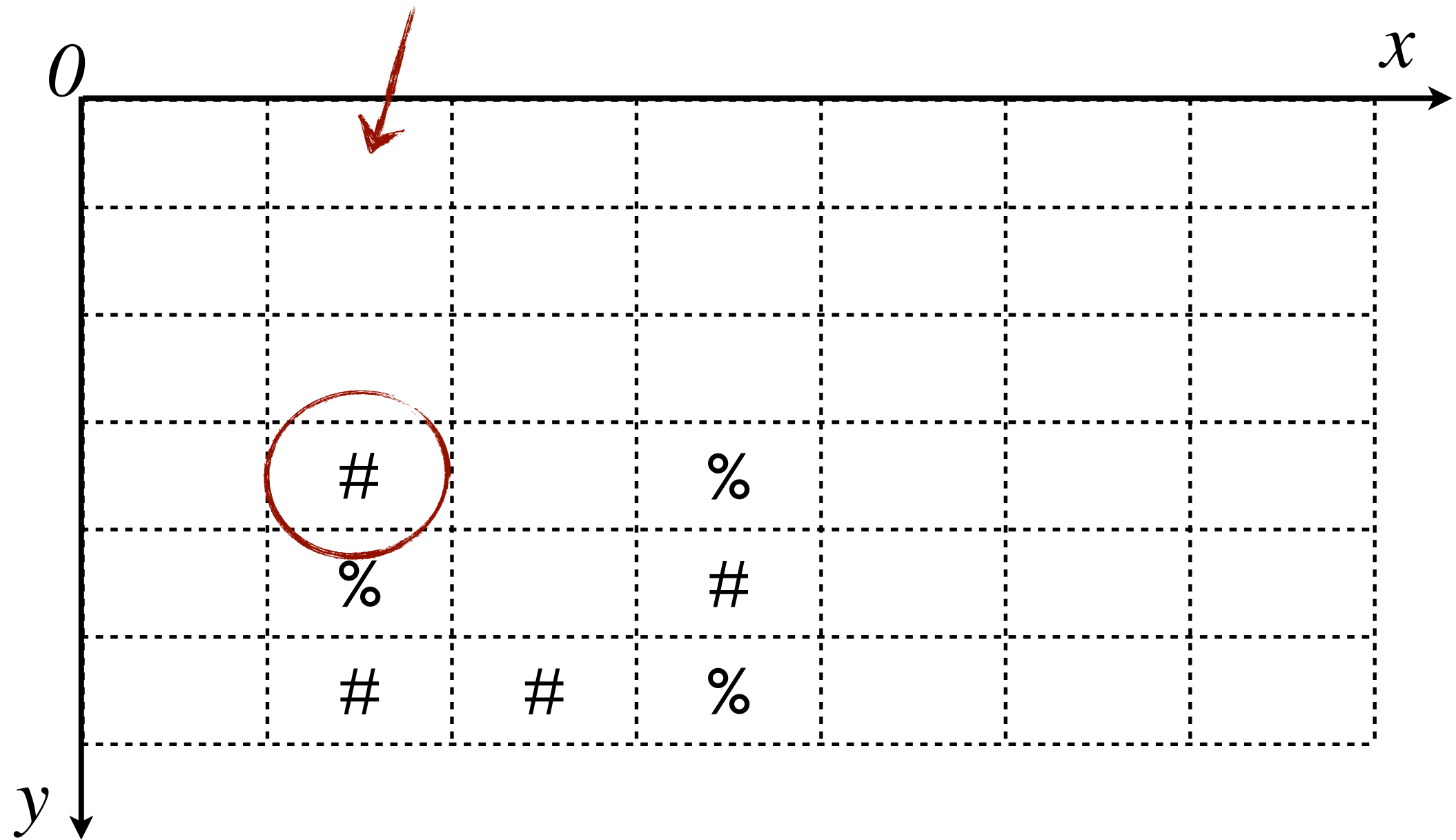
- Un giocatore è identificato da:
 - IL SUO NOME
 - QUALE PEDINA HA SCELTO

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

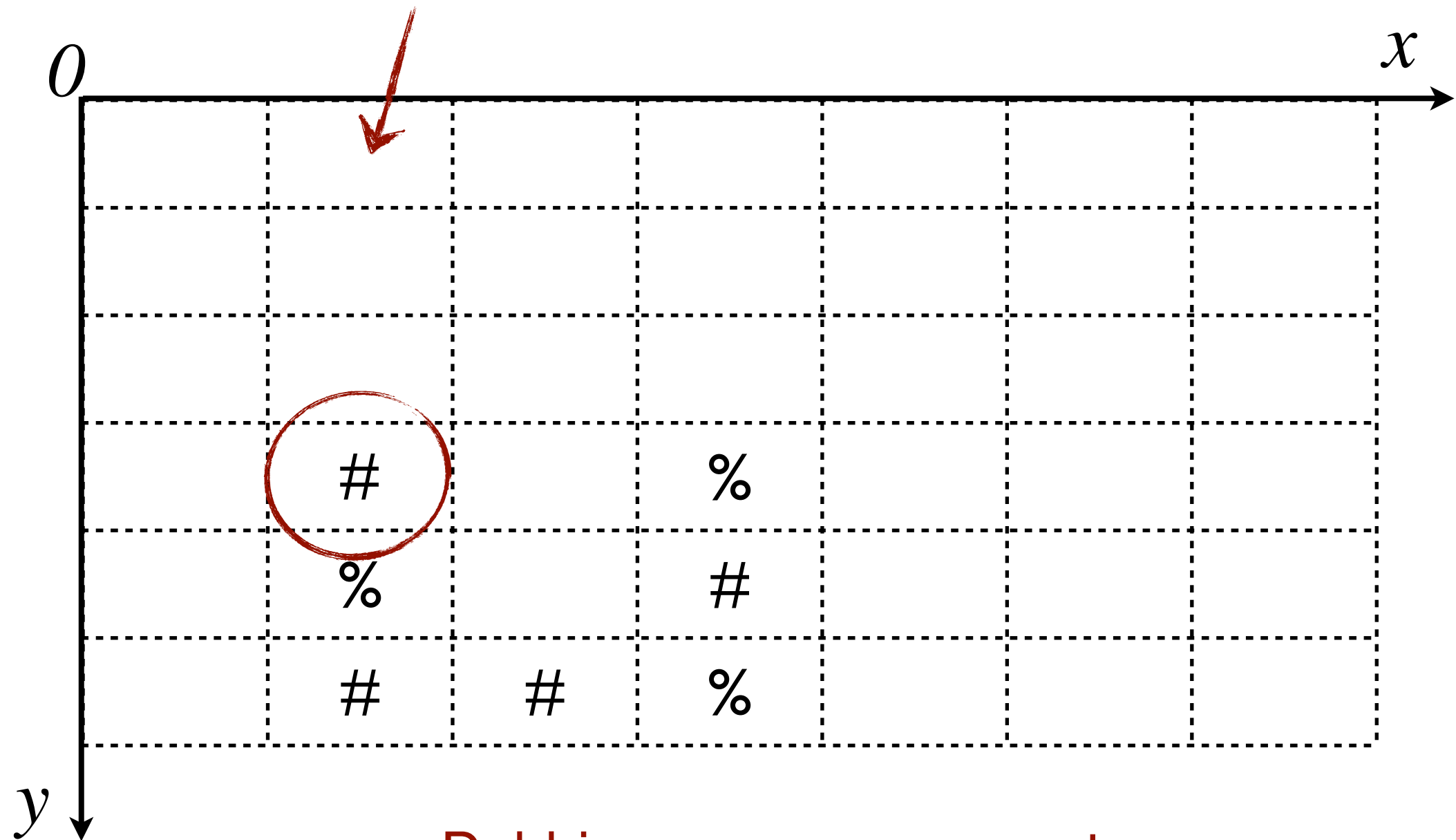
L'inserimento di una pedina



L'inserimento di una pedina



L'inserimento di una pedina



Dobbiamo creare una sorta
di effetto gravita!



```
int inserisci_pedina(char schermo[SCREEN_W][SCREEN_H],
                    int colonna, char simbolo_pedina)
{

```

L'inserimento di una pedina: codice 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;
}
```



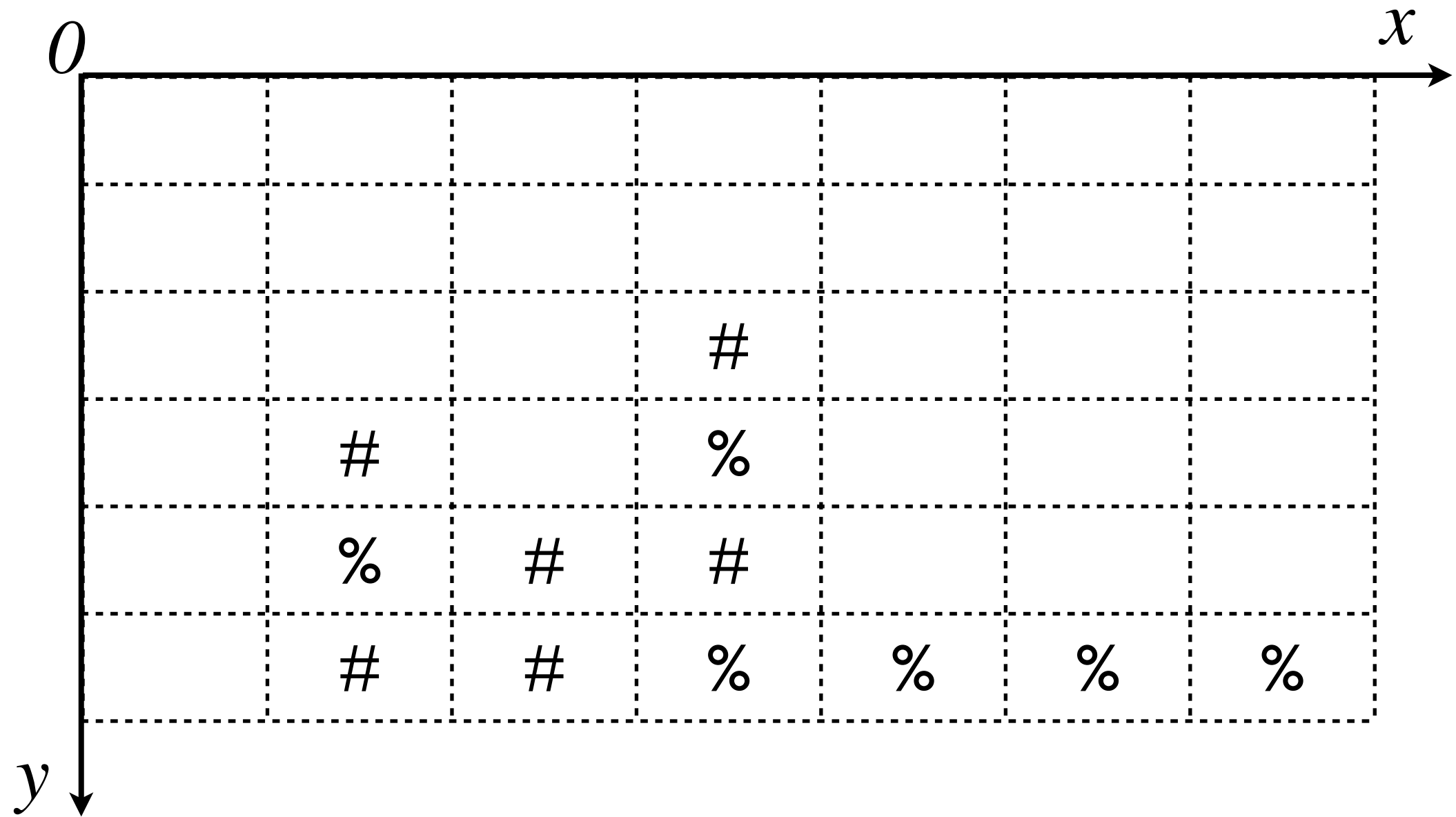
L'inserimento di una pedina: codice C

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

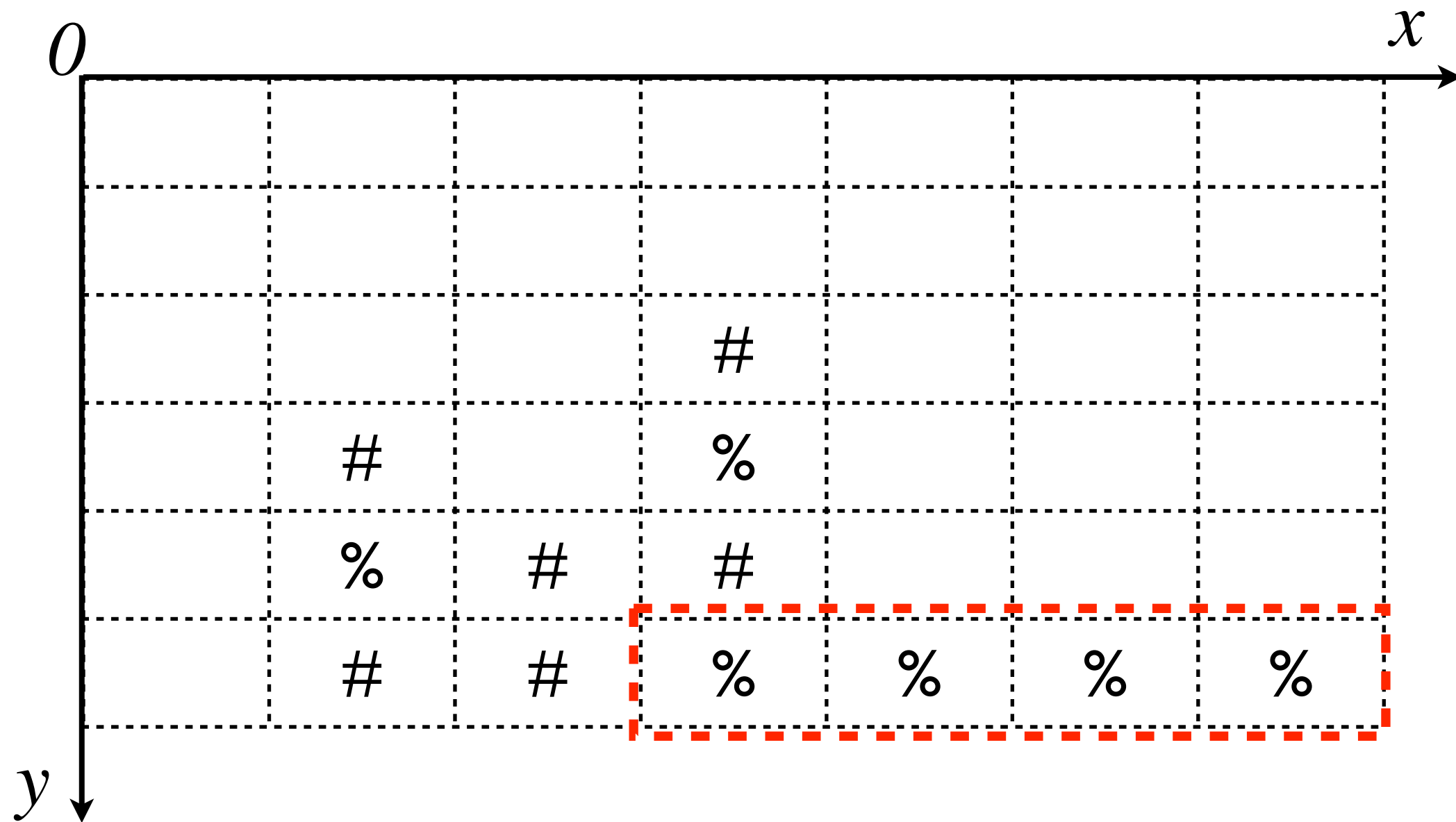
    for (y = SCREEN_H-2; y >= 0; y--)
    {
        if (schermo[colonna][y] == ' ')
        {
            p = crea_punto_schermo(colonna, y, 0);
            pedina = genera_punto(simbolo_pedina);
            disegna_forma(pedina, p, schermo);
            return 1;
        }
    }

    return 0;
}
```

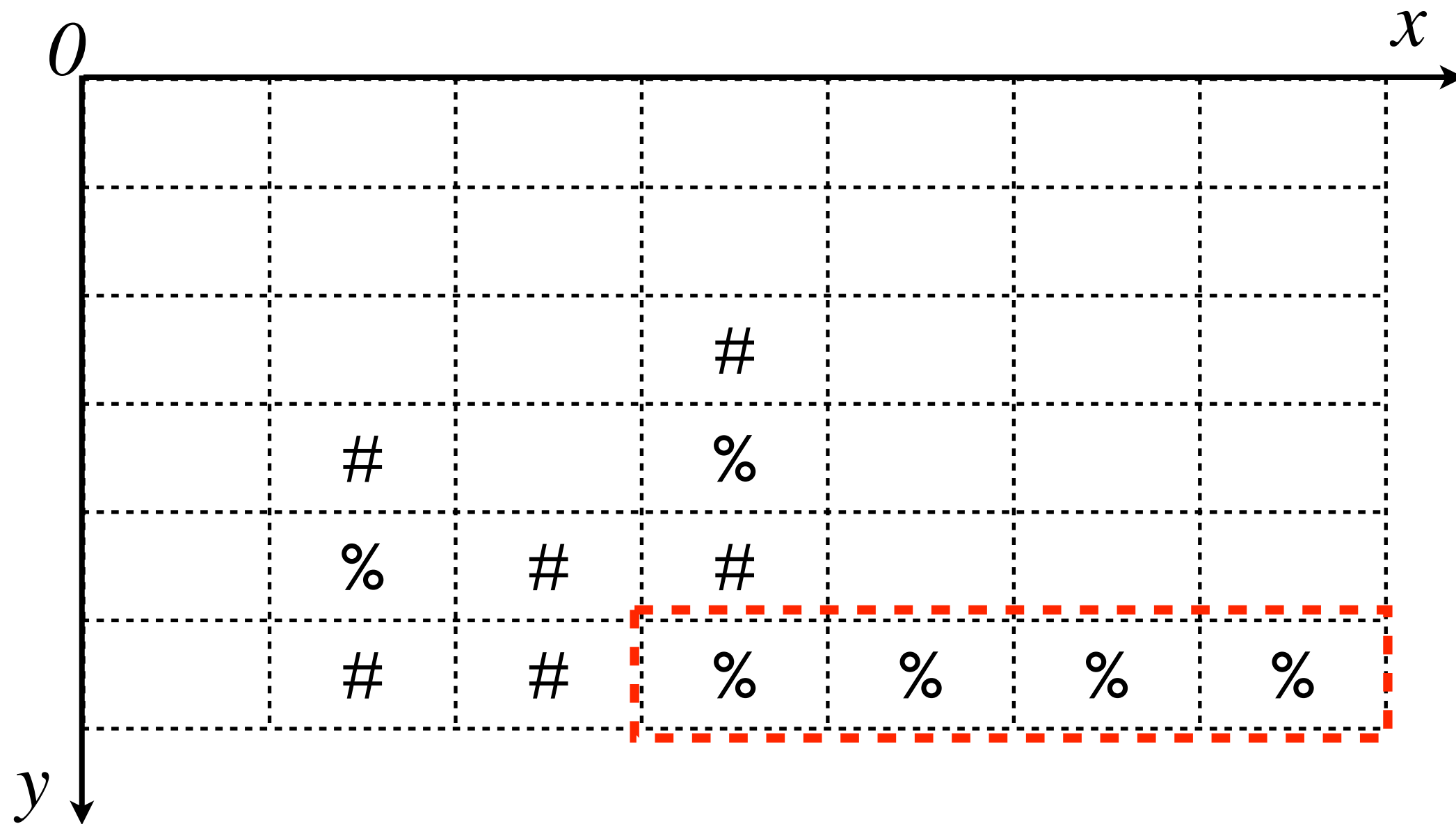
Verifica delle vincita



Verifica delle vincita



Verifica delle vincita



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

- Per verificare una vincita, 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 $A[i] \neq A[i-1]$, *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

```
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;
}
```



Controllo vincita orizzontale

```
int controlla_orizzontale(char schermo[SCREEN_W][SCREEN_H],
                        char simbolo_pedina)
{
    int x,y;
    char vettore[SCREEN_W];
    int risultato = 0;

    for (y = SCREEN_H-2; y >= 0; y--)
    {
        for (x = 0; x< SCREEN_W; x++){
            vettore[x] = schermo[x][y];
        }

        risultato = controlla_vettore(vettore, SCREEN_W, simbolo_pedina);
        if (risultato == 1)
            return 1;
    }

    return 0;
}
```



Controllo vincita verticale

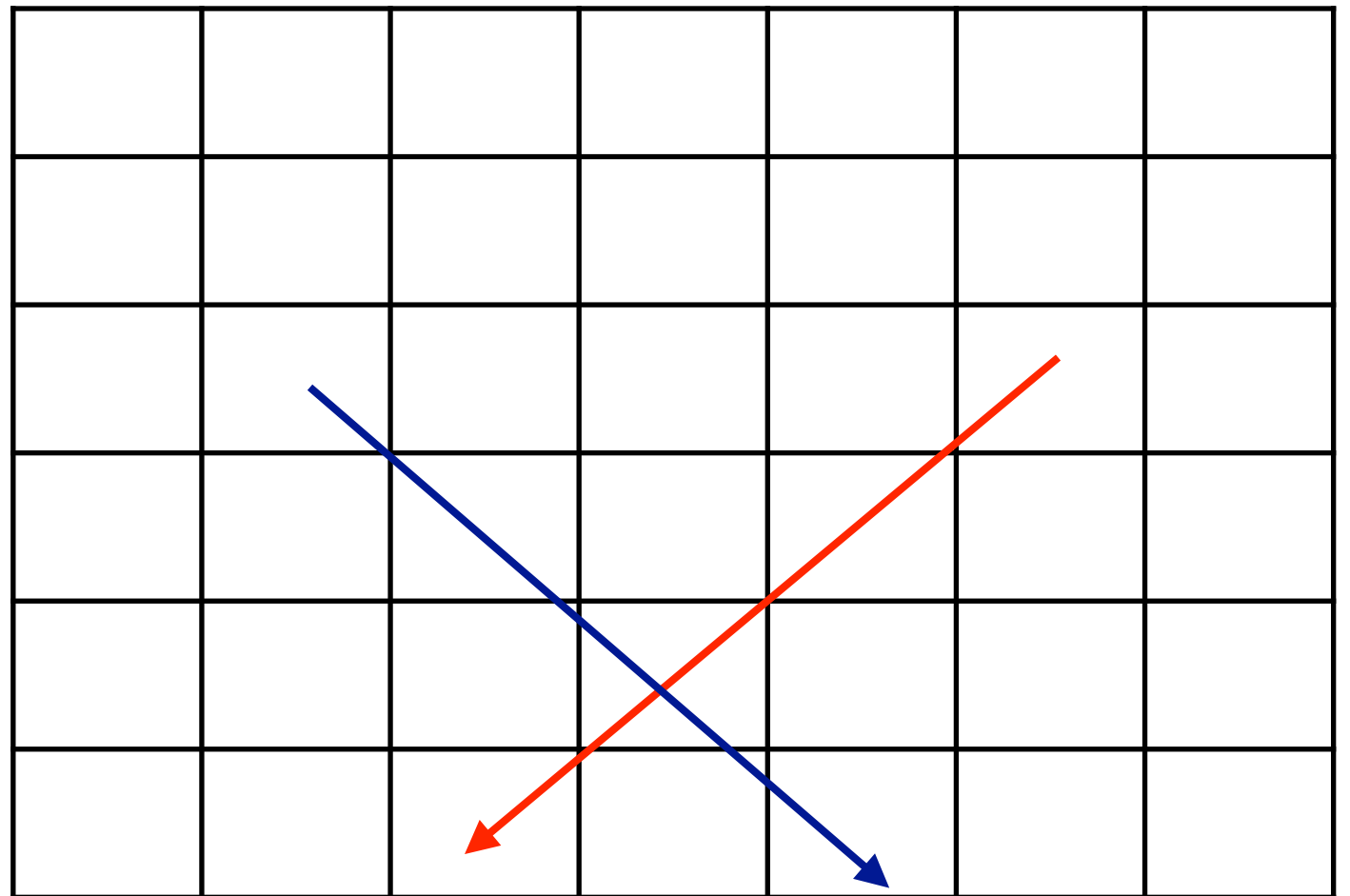
```
int controlla_verticale(char schermo[SCREEN_W][SCREEN_H],
                       char simbolo_pedina)
{
    int x,y;
    char vettore[SCREEN_W];
    int risultato = 0;

    for (x = 0; x < SCREEN_W; x++)
    {
        for (y = SCREEN_H-2; y >= 0; y--)
        {
            vettore[y] = schermo[x][y];
        }

        risultato = controlla_vettore(vettore, SCREEN_W, simbolo_pedina);
        if (risultato == 1)
            return 1;
    }

    return 0;
}
```

- In obliquo abbiamo due direzioni possibili
- Quindi creiamo una funzione parametrica che dato un parametro direzione estra un vettore orizzontale...



```
int estra_controlla_vettore_obliquo(int x, int y, char schermo[SCREEN_W][SCREEN_H],  
                                     char occorrenza, int orientamento)
```

Estrazione array obliquo (codice C)

```
int estra_controlla_vettore_obliquo(int x, int y, char schermo[SCREEN_W][SCREEN_H],  
                                     char occorrenza, int orientamento)  
{  
  
    char vettore[DIAG];  
    int curr_x, curr_y;  
    int i = 0;  
  
    do {  
        curr_x = x + (orientamento) * i;  
        curr_y = y + i;  
        vettore[i] = schermo[curr_x][curr_y];  
        i++;  
    } while (curr_x >= 0 && curr_x < SCREEN_W && curr_y >= 0 && curr_y < SCREEN_H);  
  
    return controlla_vettore(vettore, i-1, occorrenza);  
}
```

- Quindi, usando la funzione appena creata ora possiamo controllare le vincite in obliquo

```
int controlla_obliquo(char schermo[SCREEN_W][SCREEN_H], char occorrenza)
{
    // ...
}
}
```

- Quindi, usando la funzione appena creata ora possiamo controllare le vincite in obliquo

```
int controlla_obliquo(char schermo[SCREEN_W][SCREEN_H], char occorrenza)
{
    int x,y,i;
    int risultato_controllo = 0;

    for (y = 0; y < SCREEN_H; y++)
        for (x = 0; x < SCREEN_W; x++)
        {
            risultato_controllo = estra_controlla_vettore_obliquo(x,y,schermo,occorrenza, 1);
            if (risultato_controllo) return 1;

            risultato_controllo = estra_controlla_vettore_obliquo(SCREEN_W-x,y,schermo,occorrenza, -1);
            if (risultato_controllo) return 1;
        }

    return 0;
}
```

Controllo di una qualsiasi vincita

- Usando le funzioni precedenti possiamo controllare una qualsiasi vincita

```
int controlla_vincita(char schermo[SCREEN_W][SCREEN_H], char simbolo_pedina)
{
    // ...
}
```

Controllo di una qualsiasi vincita

- Usando le funzioni precedenti possiamo controllare una qualsiasi vincita

```
int controlla_vincita(char schermo[SCREEN_W][SCREEN_H], char simbolo_pedina)
{
    int result = 0;

    result += controlla_orizzontale(schermo, simbolo_pedina);
    result += controlla_verticale(schermo, simbolo_pedina);
    result += controlla_obliquo(schermo, simbolo_pedina);

    return result;
}
```

- Prima di scrivere il main(), ci mancano ancora un paio di funzioni comode...

```
// Restituisce una variabile 'giocatore' dato il nome del giocatore e il simbolo della pedina  
giocatore crea_giocatore(char nome[MAX_NOME_GIOCATORE], char pedina);  
  
// Richiede a schermo dove inserire pedina. Restituisce '1' se tutto ok, '0' in caso di errore  
int richiedi_inserimento_pedina();  
  
// Inserisce la pedina 'pedina' nella colonna 'colonna' dello campo di gioco 'schermo'  
int inserisci_pedina(char schermo[SCREEN_W][SCREEN_H], int colonna, char pedina);
```



Crea giocatore: codice C

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



Crea giocatore: codice C

```
giocatore crea_giocatore(char nome[MAX_NOME_GIOCATORE], char simbolo_pedina)
{
    giocatore g;
    strcpy(g.nome, nome);
    g.simbolo_pedina = simbolo_pedina;
    return g;
}
```

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



Richiedi inserimento pedina: codice C

```
int richiedi_inserimento_pedina()
{
    int colonna;
    printf("In quale colonna vuoi inserire la pedina? ==> ");
    scanf("%d", &colonna);
    return colonna;
}
```



Inserisci pedina: codice C

```
int inserisci_pedina(char schermo[SCREEN_W][SCREEN_H], int colonna, char simbolo_pedina)
{
    // ...
}
}
```

Inserisci pedina: codice 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;
}
```

- Ed ora, scriviamo il main:
 - Dobbiamo creare due giocatori
 - Dobbiamo gestire l'alternanza dei due giocatori
 - Dobbiamo richiedere dove inserire la pedina
 - Dobbiamo controllare la vincita

```

int main(){

    char schermo[SCREEN_W][SCREEN_H];
    giocatore giocatori[NUMERO_GIOCATORI];

    int giocatore_corrente = 0;
    int vincitore = -1;
    int colonna_richiesta;

    // Disegniamo un quadrato
    giocatori[0] = crea_giocatore("MARIO", 'o');
    giocatori[1] = crea_giocatore("PAOLO", 'x');

    inizializza_schermo(schermo);
    disegna_schermo(schermo);
    while(vincitore == -1)
    {
        printf ("Giocatore %d \n", giocatore_corrente + 1);
        colonna_richiesta = richiedi_inserimento_pedina();
        if (inserisci_pedina(schermo, colonna_richiesta, giocatori[giocatore_corrente].simbolo_pedina)){
            if (controlla_vincita(schermo, giocatori[giocatore_corrente].simbolo_pedina))
            {
                printf("Il giocatore %d, %s ha vinto!\n\n",
                    giocatore_corrente + 1, giocatori[giocatore_corrente].nome);
                return 0;
            }
            giocatore_corrente = (giocatore_corrente + 1) % NUMERO_GIOCATORI;
        } else {
            printf("La colonna inserita non è valida. Seleziona un'altra colonna.\n");
            aspetta_invio();
            aspetta_invio();    // RISOLVERE QUESTO PROBLEMA
        }
        disegna_schermo(schermo);
    }
}

```

```

}

```

**Tutte il materiale sarà
disponibile sul mio sito
internet!**

www.alessandronacci.it

See You Next Time!

