IEIM 2015-2016

Esercitazione VII

"Forza quattro"

Alessandro A. Nacci
alessandro.nacci@polimi.it - www.alessandronacci.it

Cosa facciamo oggi?

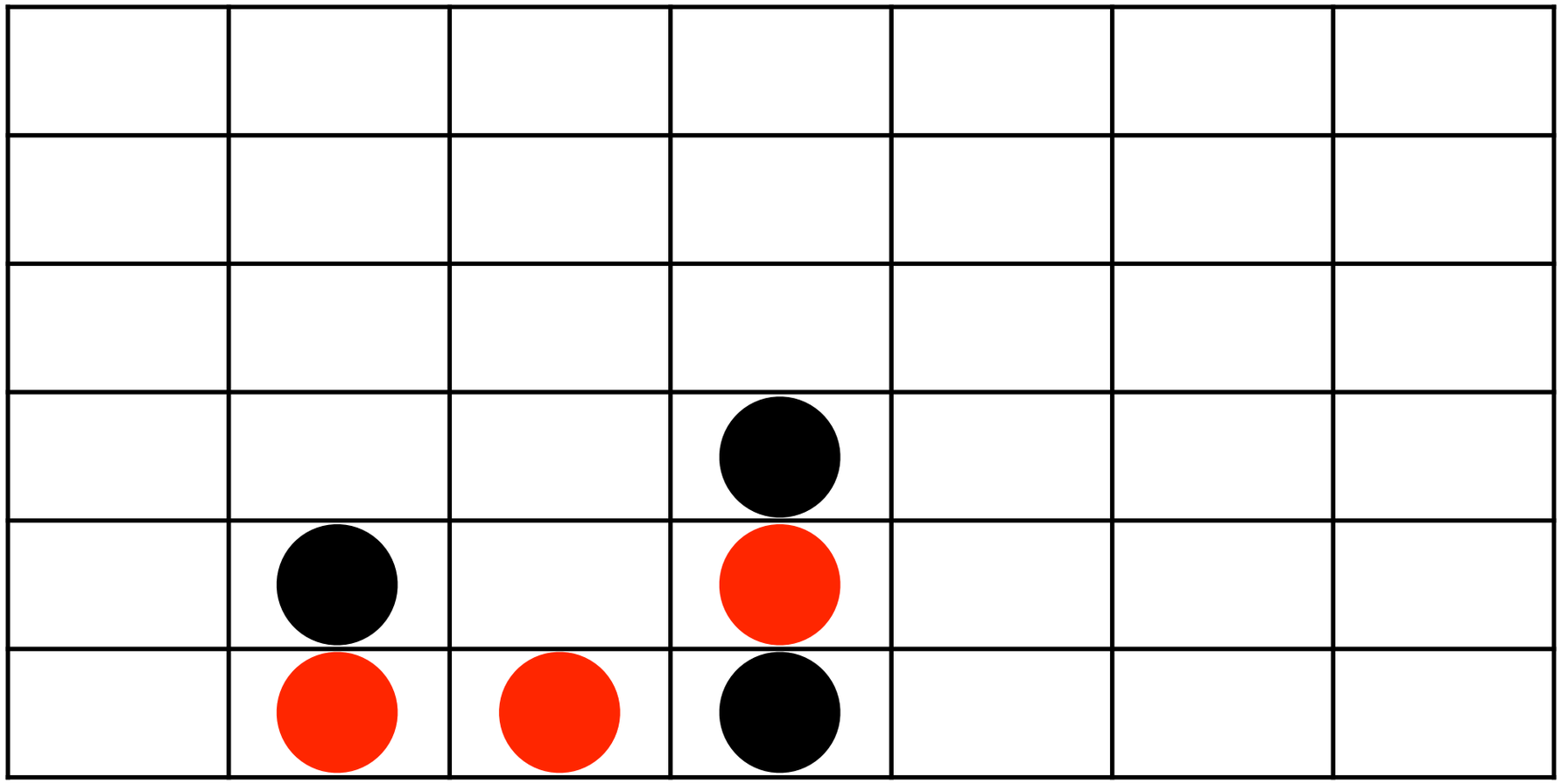
- Forza Quattro
- Il gioco dell'impiccato


READY...
SET...
GO

Un semplice videogioco

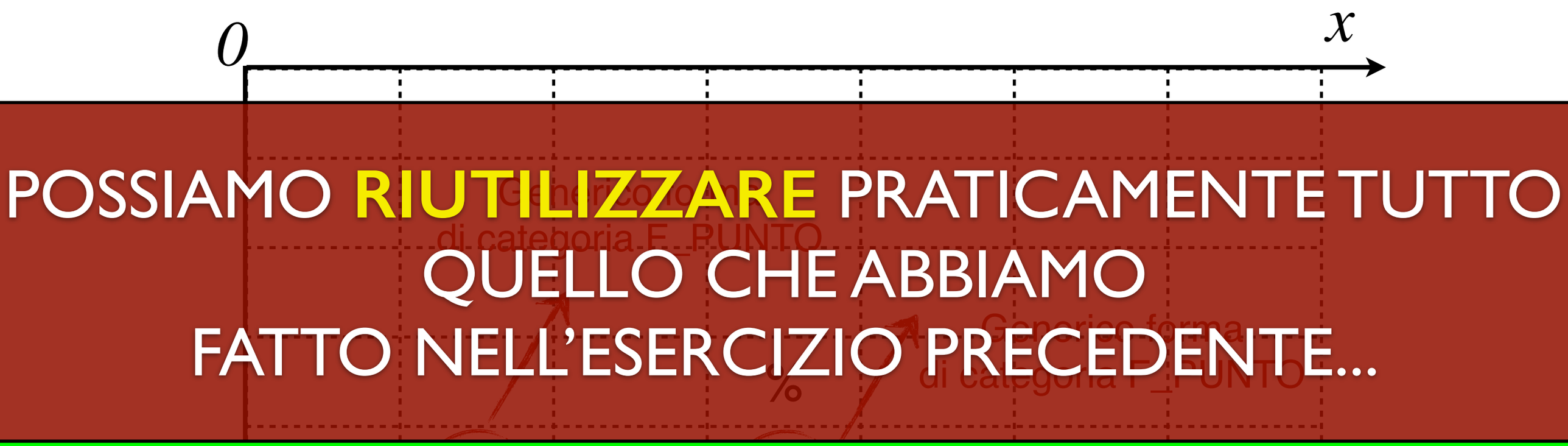
- Scriviamo un programma C che implementi il gioco del **Forza 4**
- Due giocatori, entrambi "reali"
- Il programma deve permettere di giocare
- Il programma deve annunciare il vincintore

Forza 4: rappresentazione



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

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

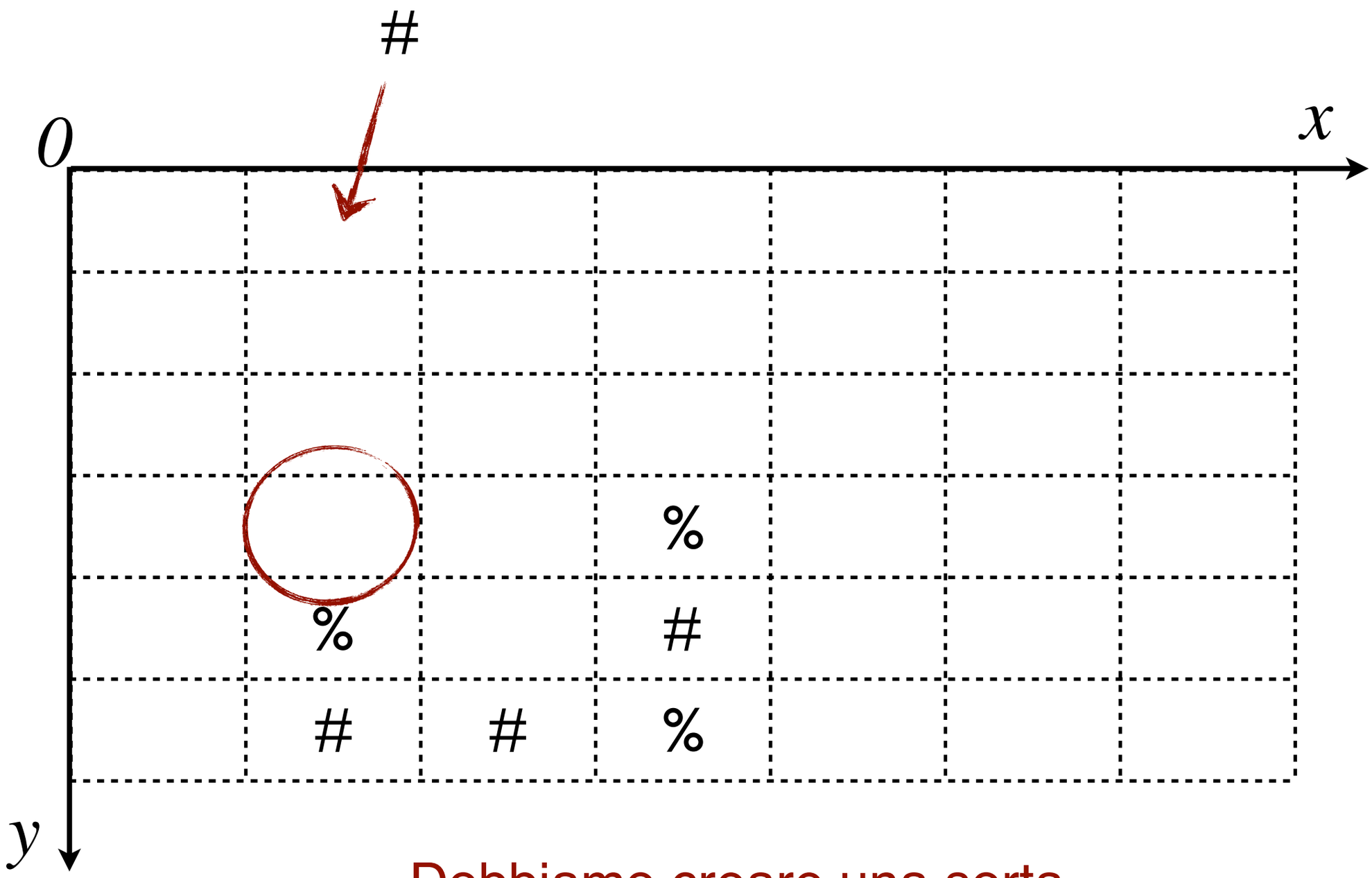
- Anche se possiamo utilizzare buona parte del codice già scritto per il "PIANO CARTESIANO", qualche concetto ancora ci manca:
  - Come rappresentiamo un GIOCATORE?
  - Come gestiamo l'inserimento di una pedina?
  - Come controlliamo la vincita?

Il giocatore

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

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

L’inserimento di una pedina

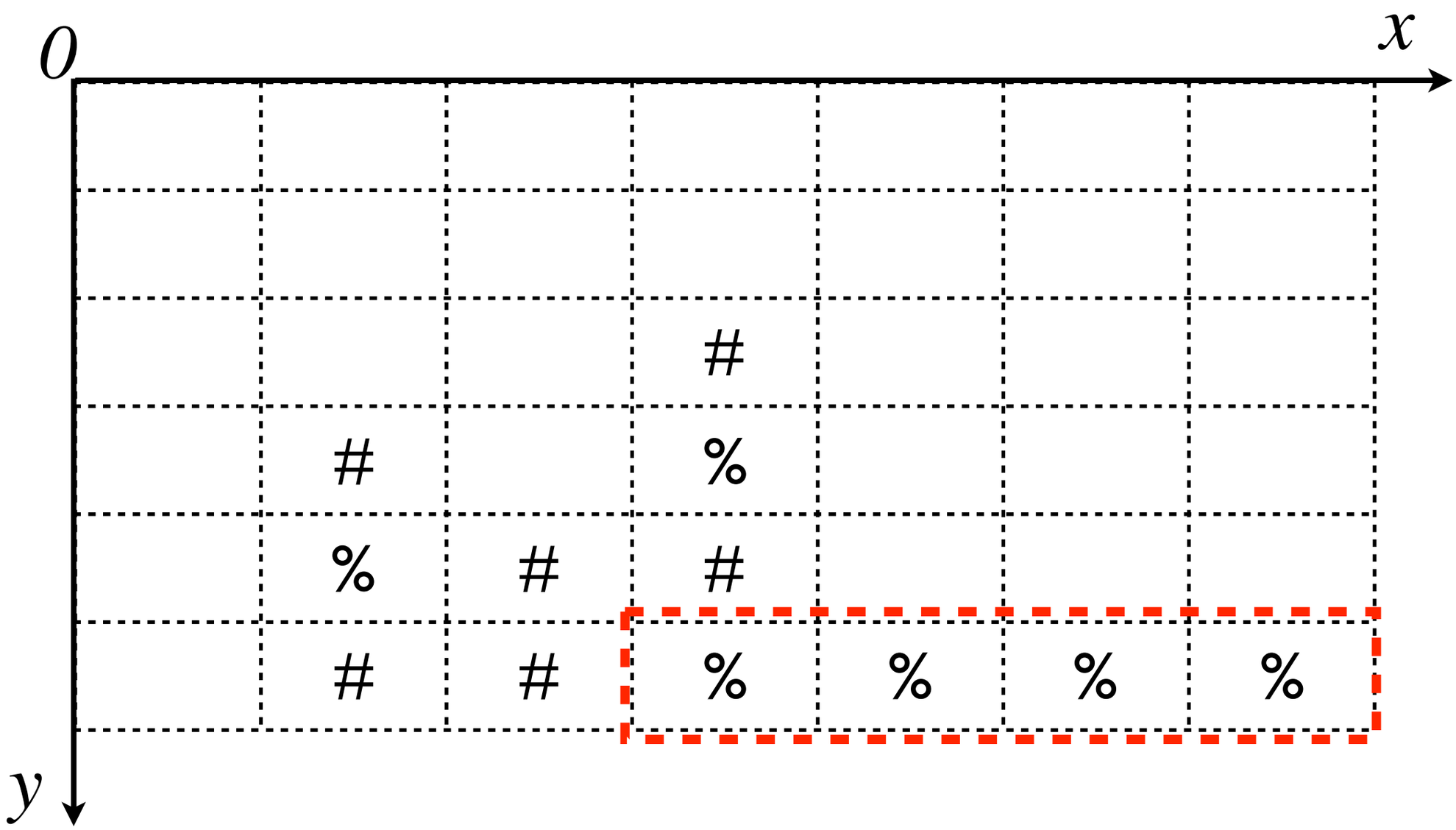


Dobbiamo creare una sorta
di effetto gravita!

L'inserimento di una pedina: codice C

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

}
```

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)
{

}
```

Controllo vincita orizzontale

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



}
```

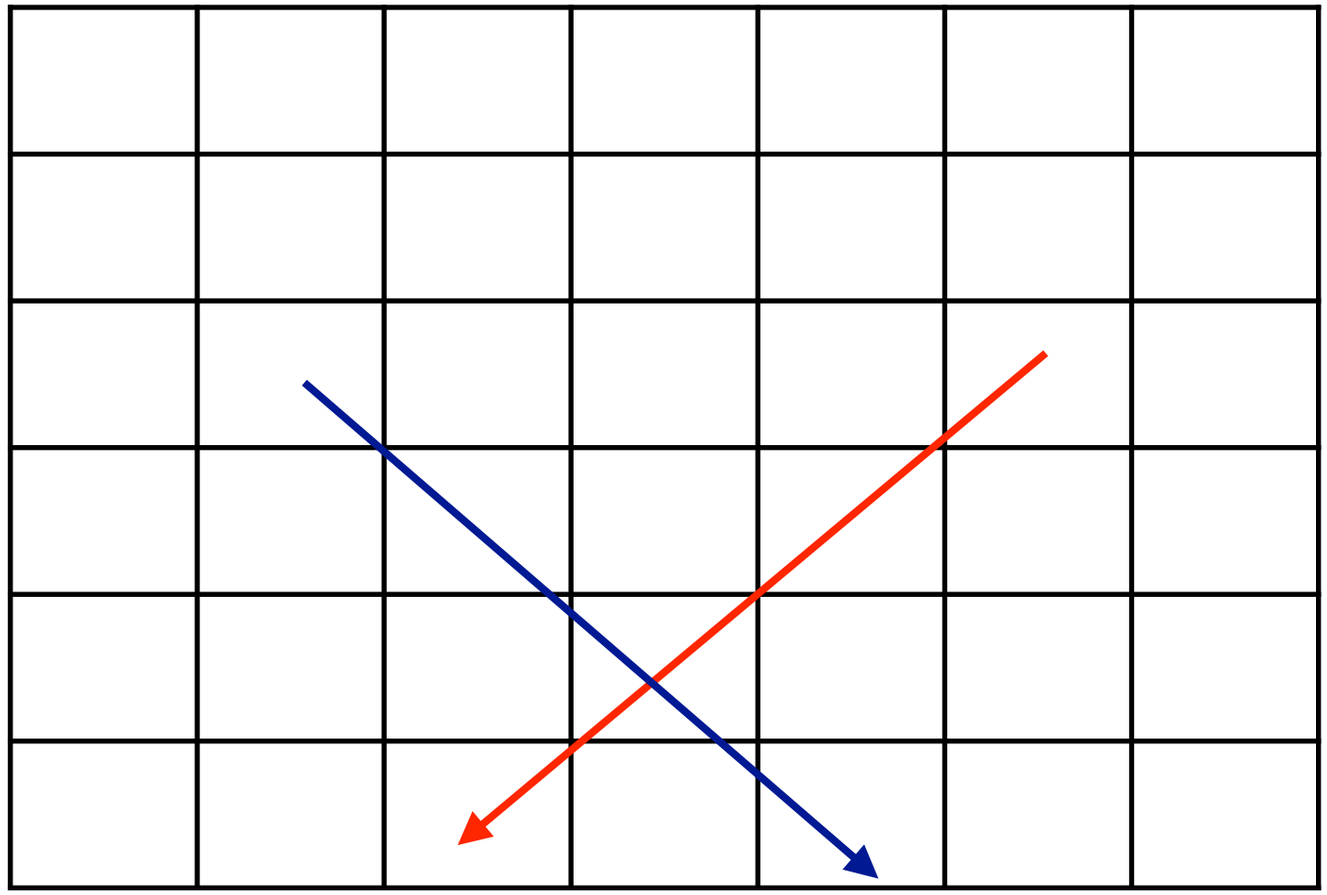
Controlla vincita verticale

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




}
```

Controlla vincita obliquo

- 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)
{



}
```

Controllo vincita in obliquo

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

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




}
```

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)
{



}
```

- 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. Restisce '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

```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;
}
```

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

Richiedi inserimento pedina: codice C

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

Inserisci pedina: codice C

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

}
```

- 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

```c
int main(){

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

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

    // Disegnamo 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!

alessandronacci.it