IEIM

Esercizio: Il Piano Cartesiano

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

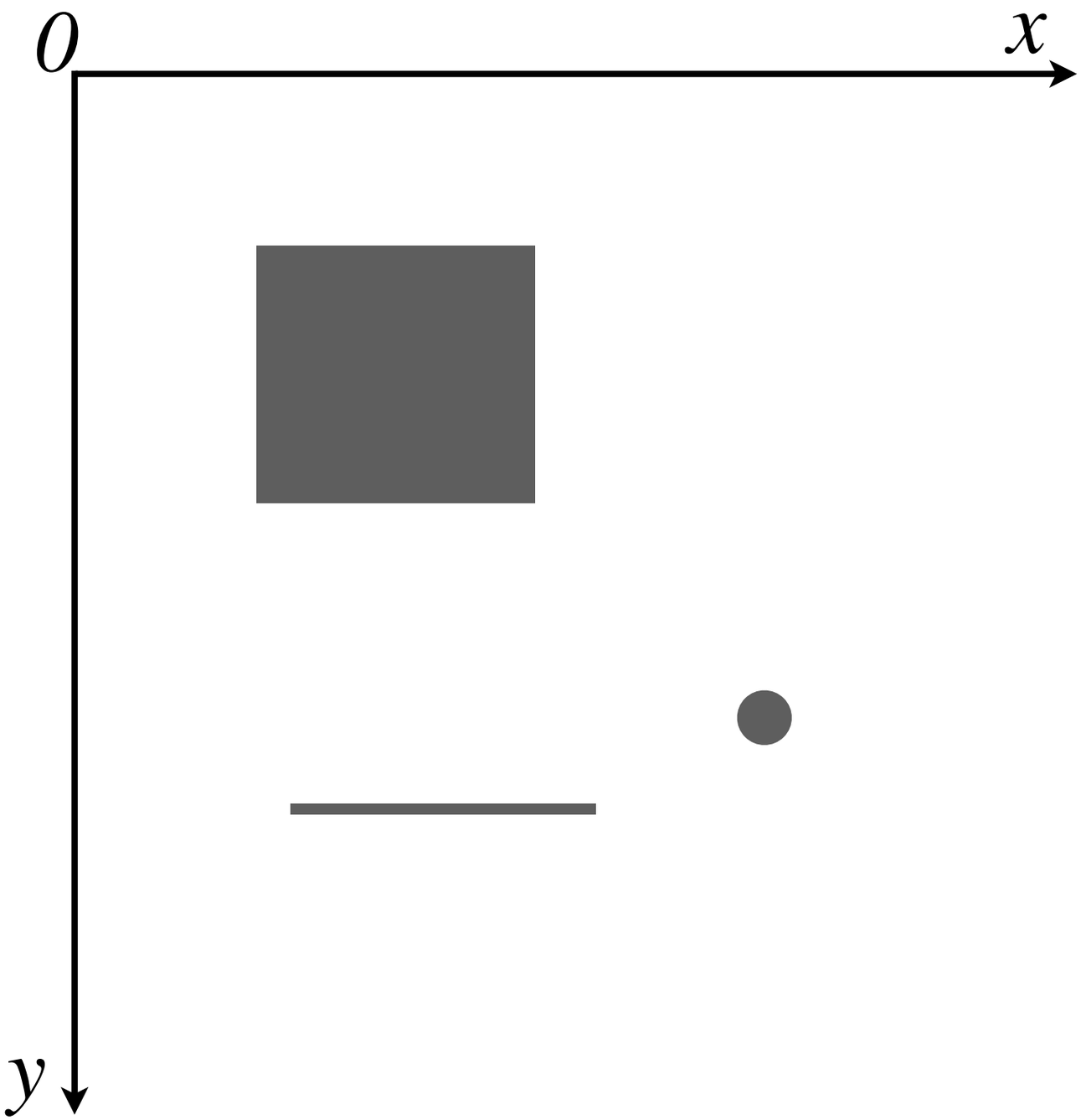
Il piano cartesiano

Esercizio 4

Es4: Il piano cartesiano

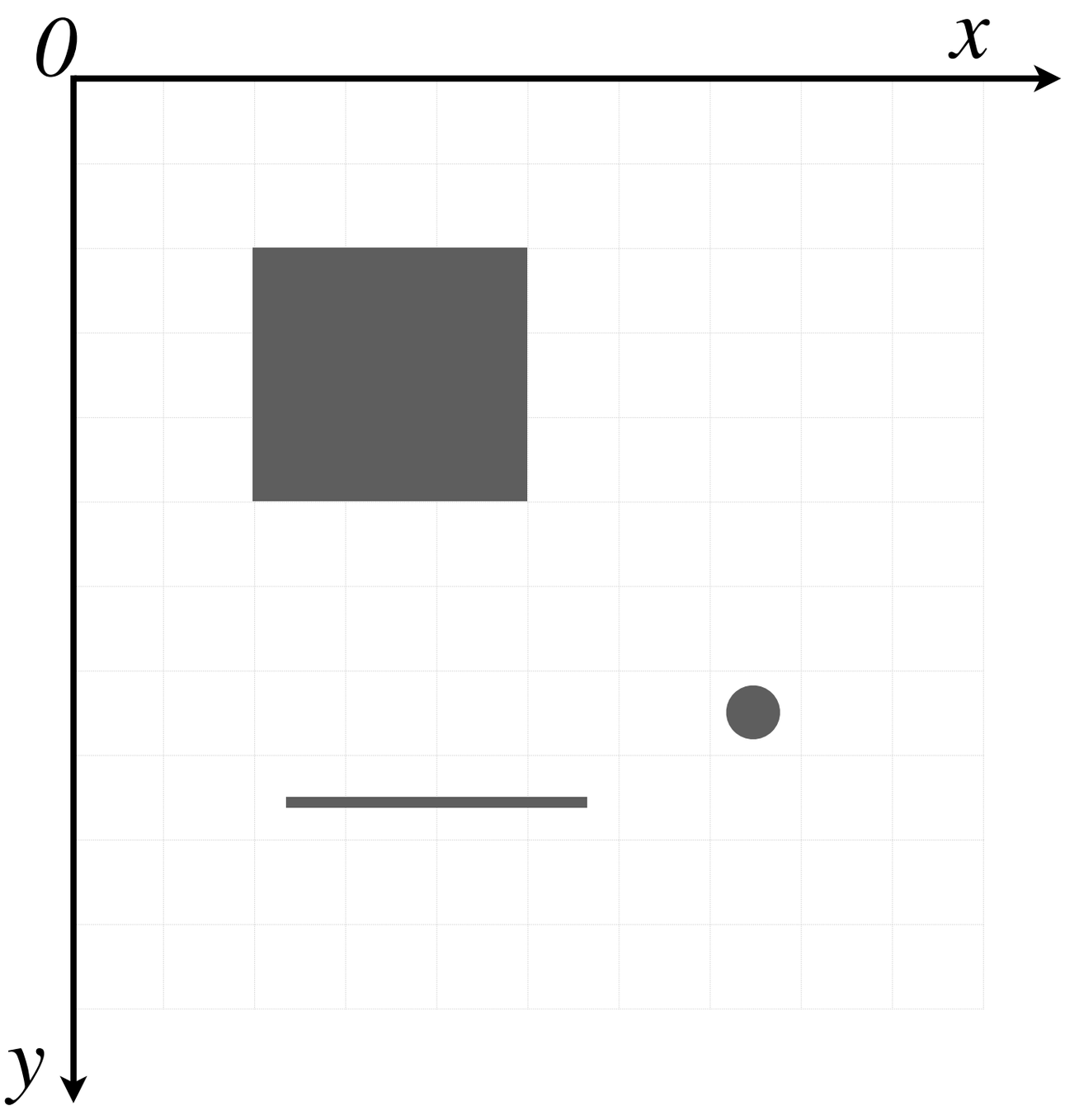
- Scrivere in C un programma che rappresenti un piano cartesiano
- Il programma deve poter rappresentare e visualizzare a schermo PUNTI, LINEE e QUADRATI
- Deve essere inoltre possibile manipolare le forme create (spostarle, cancellarle, ingrandirle, etc..)
- Il programma deve poter rappresentare la curva di una funzione di secondo grado:

$$y = a2 \cdot x^2 \cdot + a1 \cdot x + a0$$



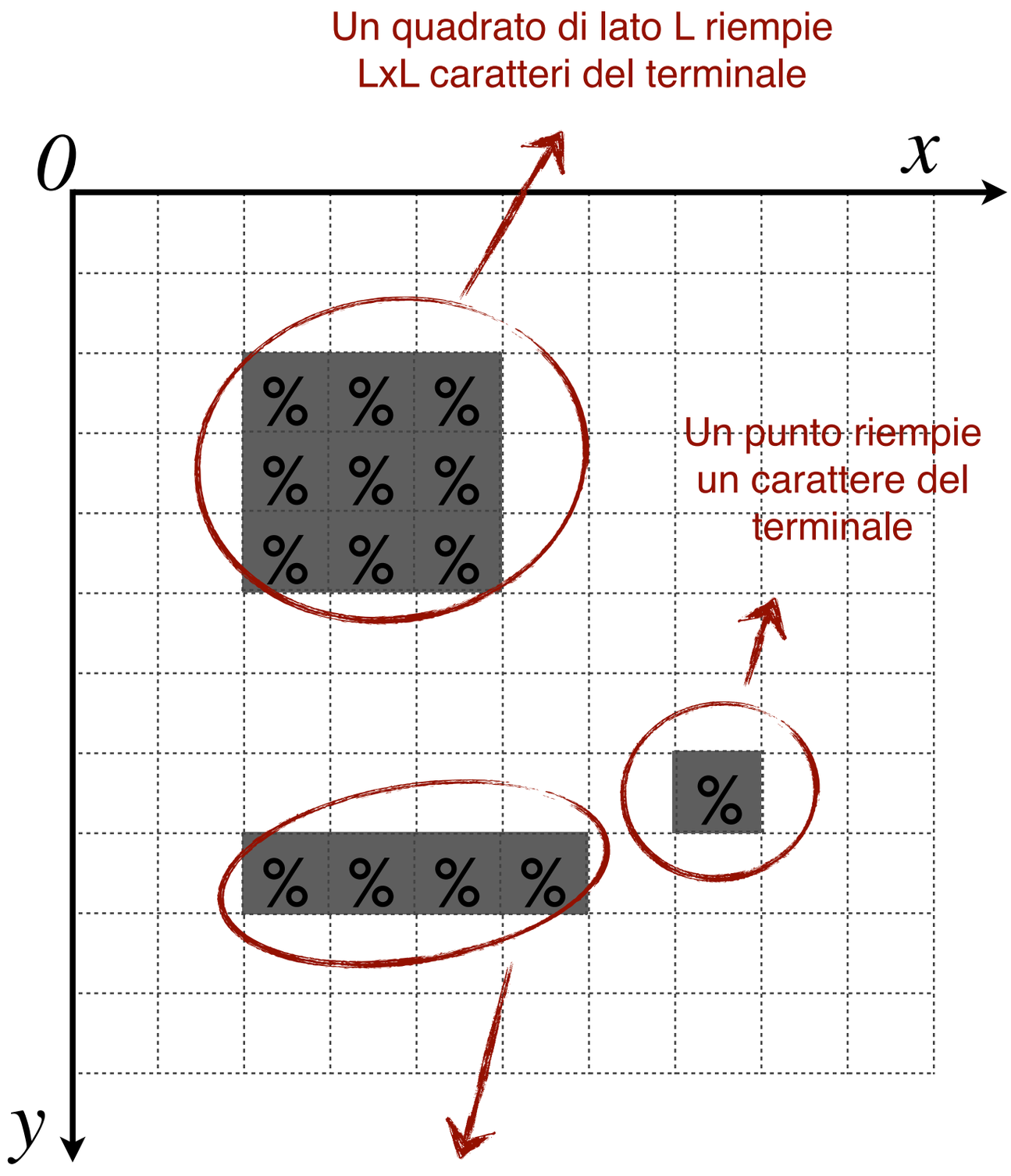
La rappresentazione

- Per visualizzare a schermo abbiamo a disposizione il solo terminale:
  - Formato da RIGHE e COLONNE di caratteri ASCII
- Ci *arrangiamo* con quello che abbiamo



La rappresentazione

- Per visualizzare a schermo abbiamo a disposizione il solo terminale:
  - Formato da RIGHE e COLONNE di caratteri ASCII
- Ci *arrangiamo* con quello che abbiamo



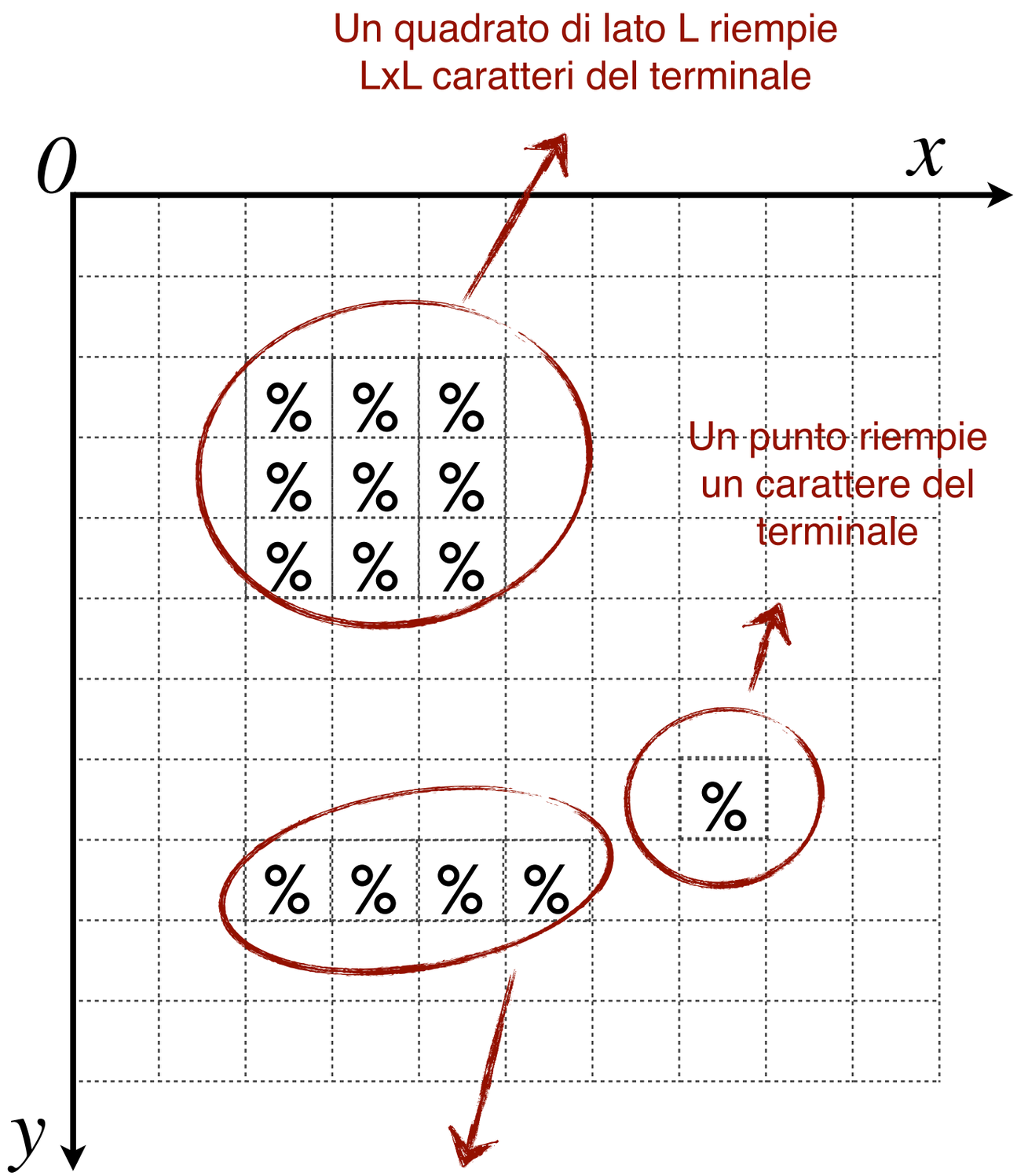
La rappresentazione

- Per visualizzare a schermo abbiamo a disposizione il solo terminale:
  - Formato da RIGHE e COLONNE di caratteri ASCII
- Ci *arrangiamo* con quello che abbiamo

ATTENZIONE!

NON POTREMO RAPPRESENTARE I BORDI DELLE FIGURE!

CI LIMITEREMO A RAPPRESENTARE IL CONTENUTO DELLE FIGURE CON DEI CARATTERI



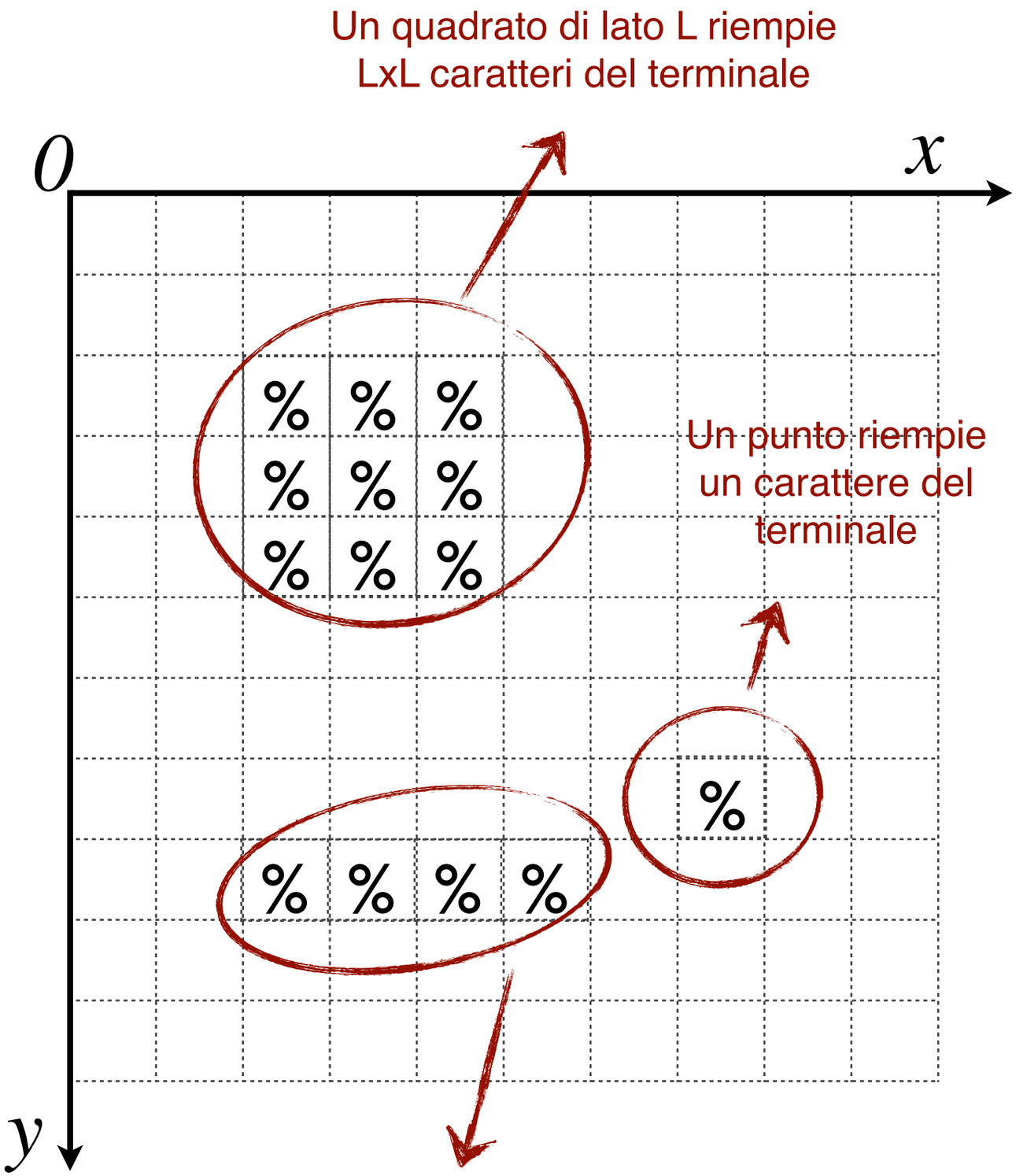
Cosa ci serve?

- Come sempre, prima di scrivere un programma, dobbiamo:
  1. definire i tipi di dato
  2. pensare di quali *costanti* avremo bisogno
  3. pensare di quali variabili avremo bisogno
  4. scegliere quali funzioni implementare
  5. implementare

Definizione dei tipi di dato

- I tipi di dato che possono essere utili sono:
  - Punto dello schermo
  - Una generica forma
  - Quali forme sono disponibili
  - Direzioni

IMPLEMENTIAMO ALLA LAVAGNA!



Definizione dei tipi di dato

```
// Elenco delle forme supportate dal programma
typedef enum {F_PUNTO, F_LINEA,
              F_POLIGONO_QUADRILATERO, F_GENERICA} categoria_forma;
             // Elenco delle forme supportate dal programma
typedef enum {D_VERTICALE, D_ORIZZONTALE} direzione;
                       // Elenco direzioni possibili per le linee

typedef struct {
     int x;
     int y;
     char valore;
} punto_schermo;
                                       // TIPO DI DATO per la creazione di variabili di tipo 'punto'

typedef struct {
     punto_schermo pixels[MAX_PUNTI_FORMA];
                                       // Elenco dei pixel che compongono la forma
     int numero_pixel;
                              // Numero dei pixel che compongono la forma
     categoria_forma categoria;
                              // Categoria della forma
} forma;
                              // TIPO DI DATO per la creazione di variabili di tipo 'forma'
```

Definizione delle costanti

```
// Impostazioni schermo
#define SCREEN_H 20
#define SCREEN_W 40
#define RISOLUZIONE 800      //SCREEN_H * SCREEN_W

// Impostazioni memorizzazione
#define MAX_PUNTI_FORMA 64
#define MAX_NUMERO_FORME 10

// Impostazioni sistema
#define LINEE_TERMINALE 25
```

Possiamo usare le #define per definire valori costanti durante l'esecuzione del programma

E le variabili?

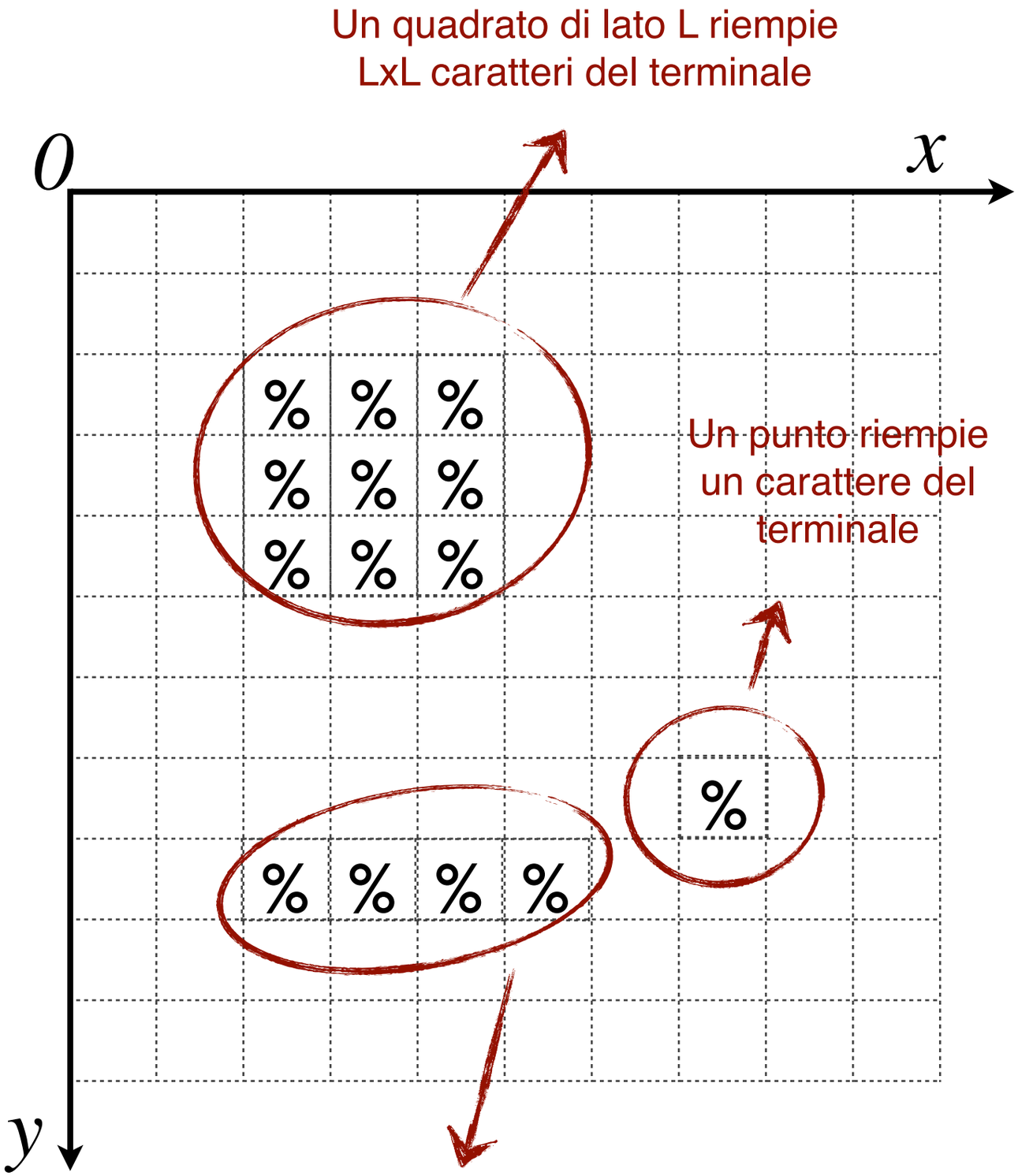
- Tendenzialmente, avremo bisogno di qualcosa tipo…

```
int main(){
char schermo[SCREEN_W][SCREEN_H];
forma quadrato;
punto_schermo p;
forma linea_or;
forma linea_vr;
forma punto;
```

Le funzioni

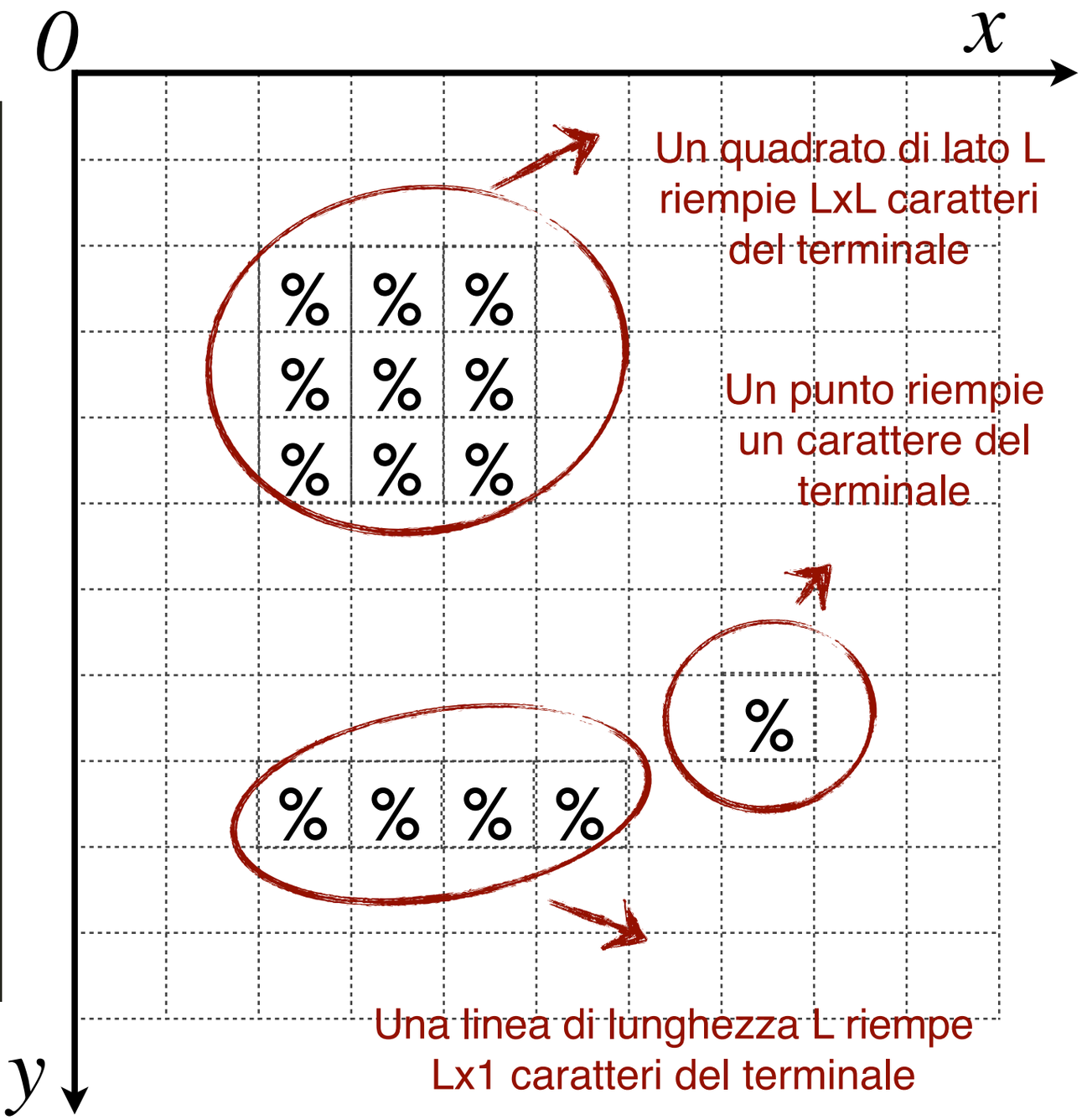
- Le funzioni che possono essere utili sono:
  - Funzioni per la visualizzazione a video
  - Modifica delle matrice che rappresenta lo schermo
  - Generazione delle forme

IMPLEMENTIAMO ALLA LAVAGNA!



Inizializza schermo

```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);
```




```c
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H]);
```

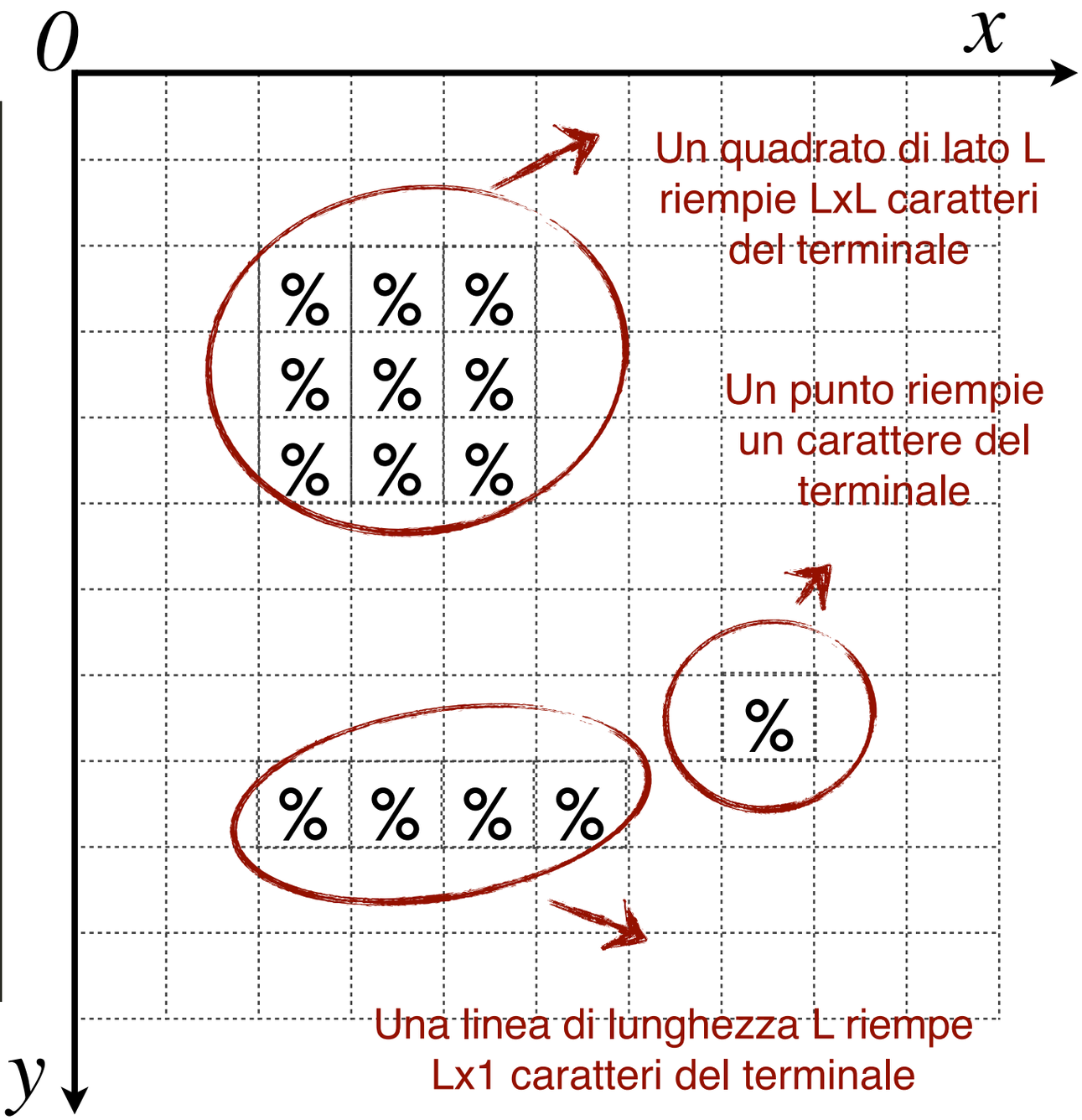
Riempie di caratteri vuoti la matrice in ingresso 'schermo'

Inizializza schermo

```
void inizializza_schermo(char schermo[SCREEN_W][SCREEN_H])
{

      int x,y;

      for (y = 0; y < SCREEN_H; y++){
            for (x = 0; x < SCREEN_W; x++){
                  schermo[x][y] = ' ';
            }
      }

}
```

Inserisci bordi

```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);
```




```c
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
```

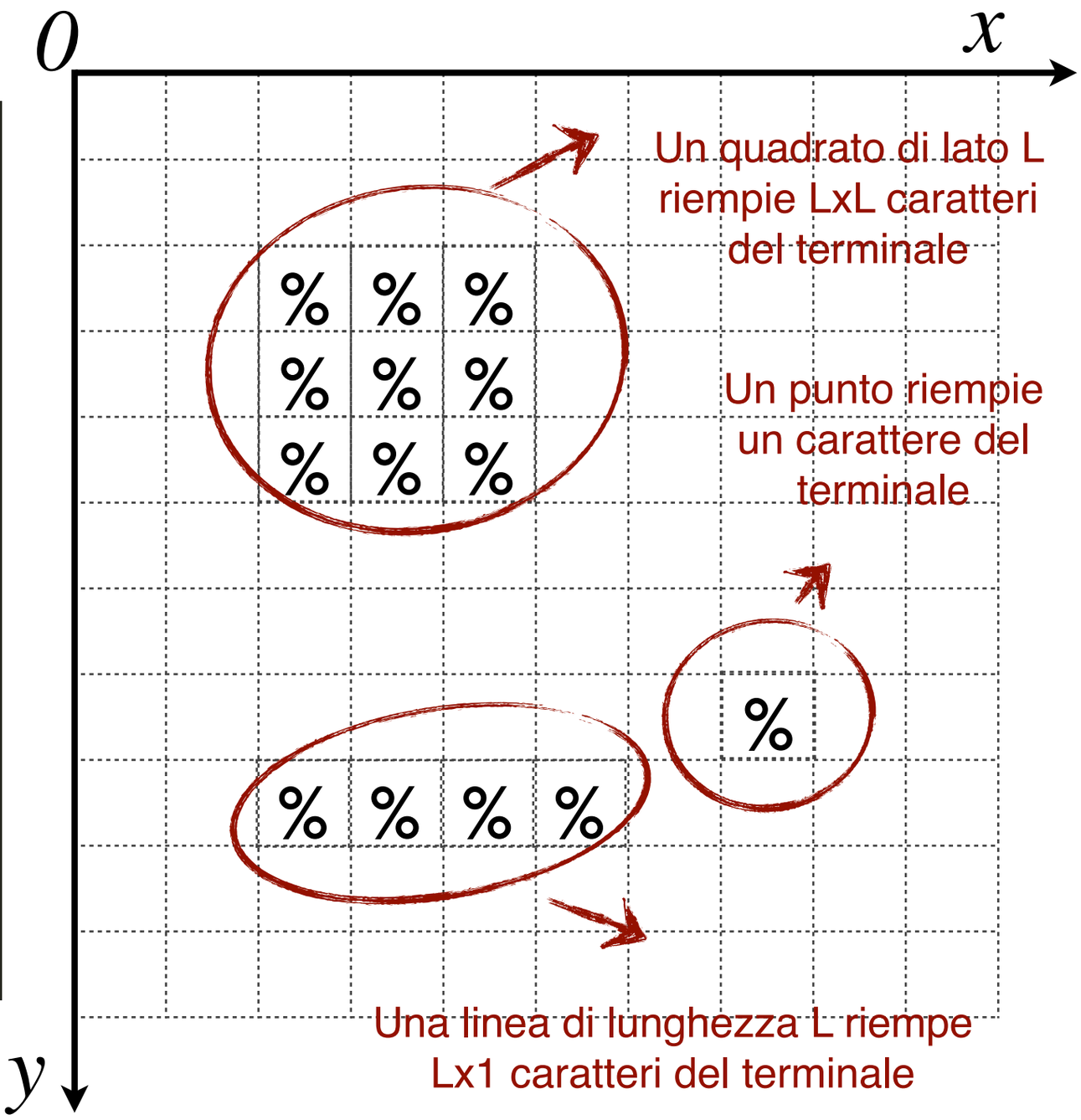
Inserisce il carattere '%' sui bordi della matrice 'schermo'

Inserisci bordi

```
void inserisci_bordi(char schermo[SCREEN_W][SCREEN_H])
{

     int x,y;
     char char_bordo = '%';

     for (y = 0; y < SCREEN_H; y++){
          for (x = 0; x < SCREEN_W; x++){
               if (x == 0 || x == SCREEN_W-1 || y == 0 || y == SCREEN_H-1)
                    schermo[x][y] = char_bordo;
          }
          printf("\n");
     }
}
```

Disegna schermo

```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);
```




```c
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H]);
```

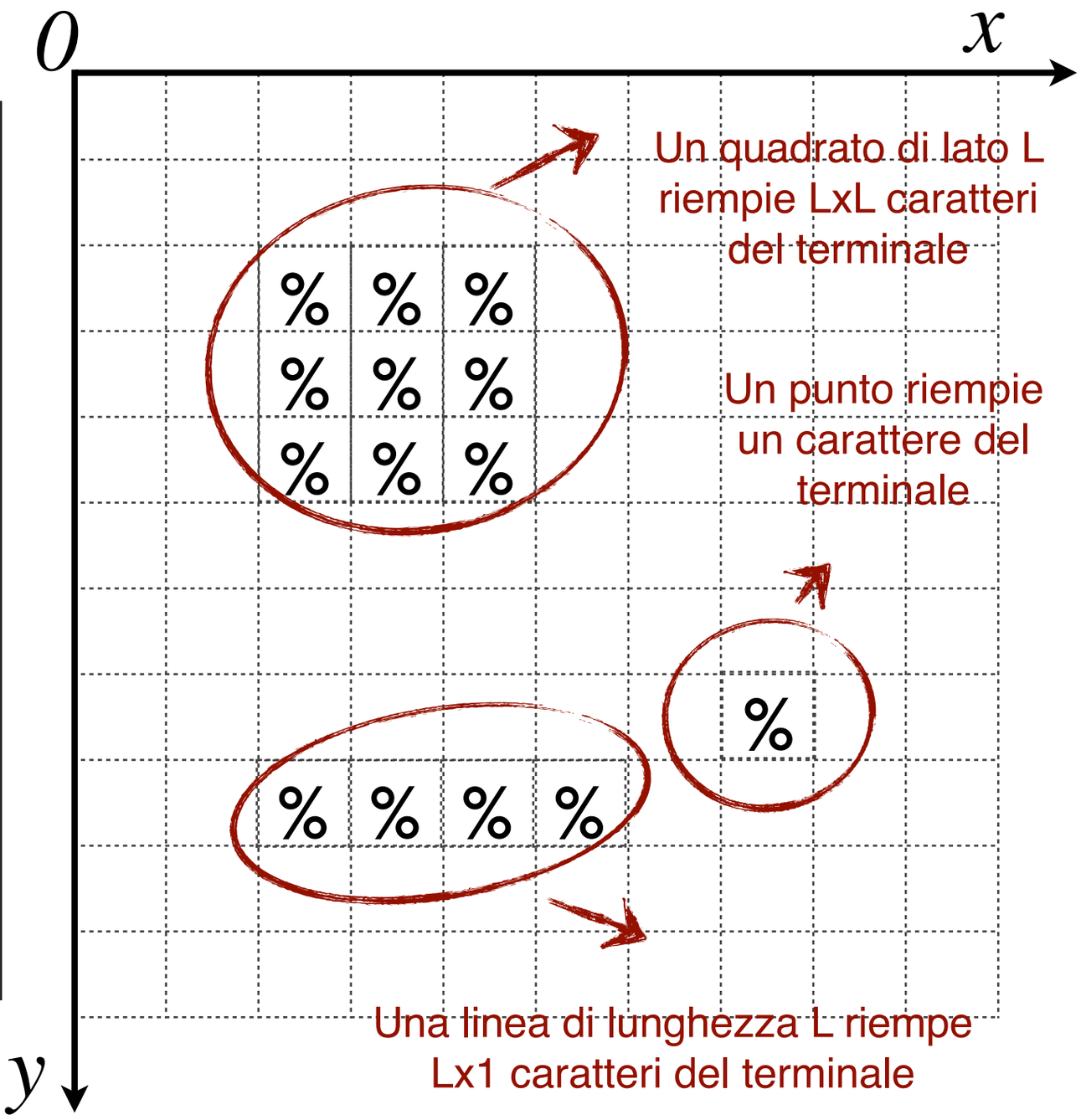
Disegna a schermo la matrice in ingresso 'schermo'

Disegna schermo

```
void disegna_schermo(char schermo[SCREEN_W][SCREEN_H])
{

     pulisci_terminale();


     inserisci_bordi(schermo);

     int x,y;

     for (y = 0; y < SCREEN_H; y++){
          for (x = 0; x < SCREEN_W; x++){
               printf("%c",schermo[x][y]);
          }
          printf("\n");
     }


}
```

Aspetta invio

```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);
```




```c
void aspetta_invio();
```

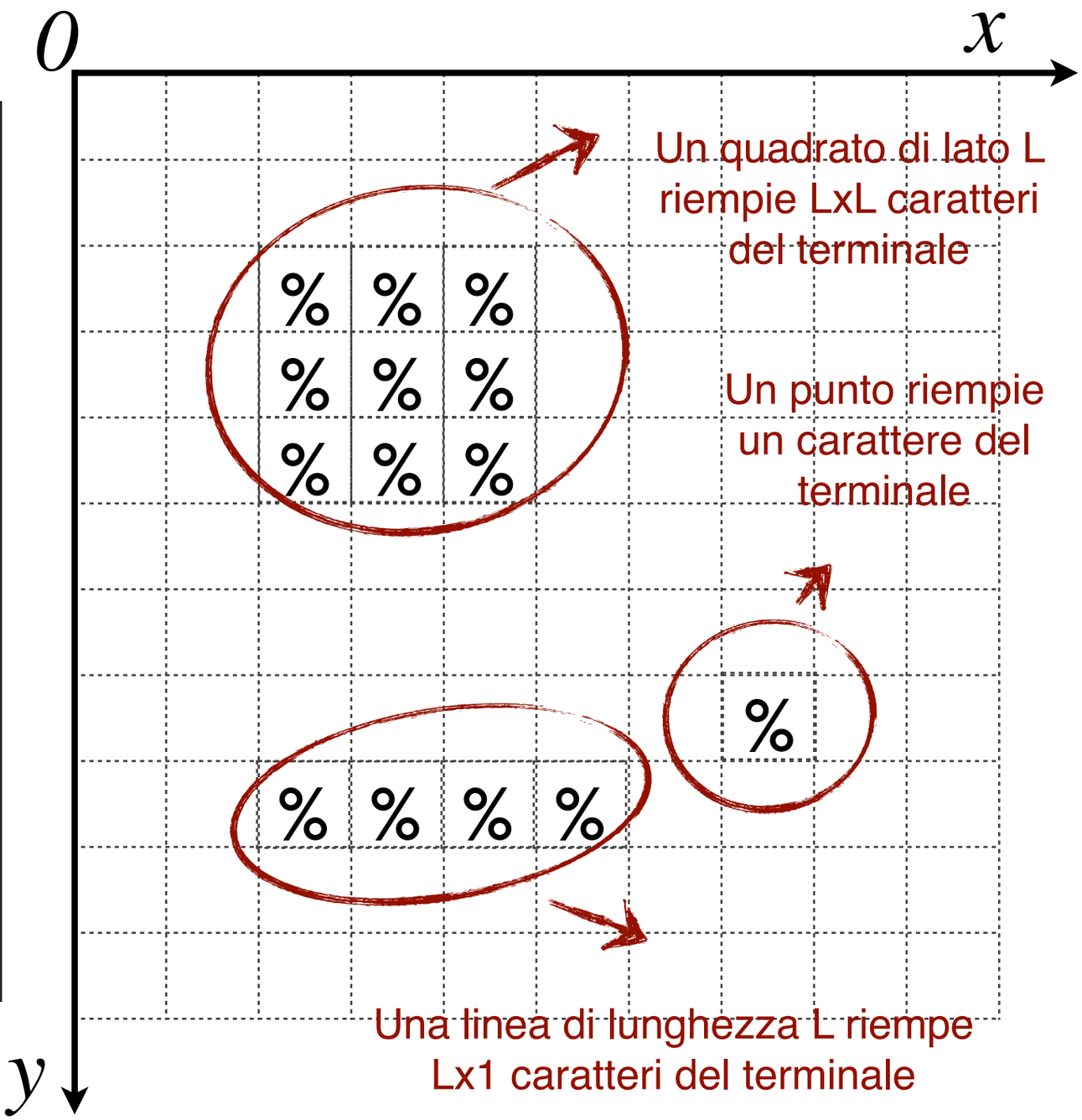
Quando lanciata, mette il programma in attesa di un INVIO da parte dell'utente

Aspetta invio

```
void aspetta_invio()
{
    printf("Press enter to continue\n");
    char enter = 0;
    while (enter != '\r' && enter != '\n') { enter = getchar(); }
}
```

Pulisci terminale

```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);
```




```
void pulisci_terminale();
```

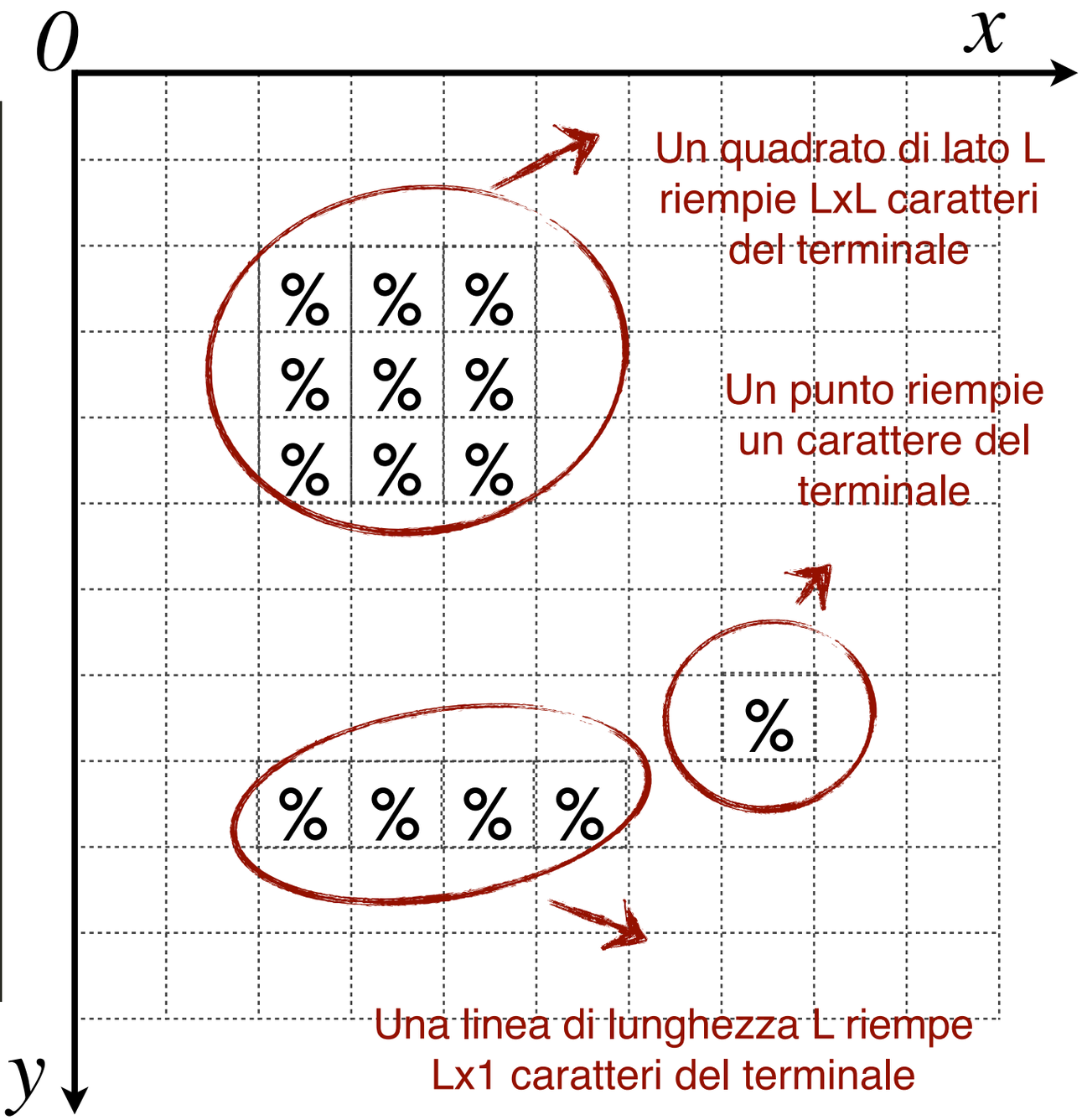
Cancella il contenuto del termianale su cui viene stampato l'output del programma

Pulisci terminale

```
void pulisci_terminale(){
    int i;
    for (i = 0; i < LINEE_TERMINALE; i++)
    {
        printf("\n");
    }
}
```

Disegna forma

```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);
```




```
void disegna_forma(forma f, punto_schermo p,
    char schermo[SCREEN_W][SCREEN_H]);
```

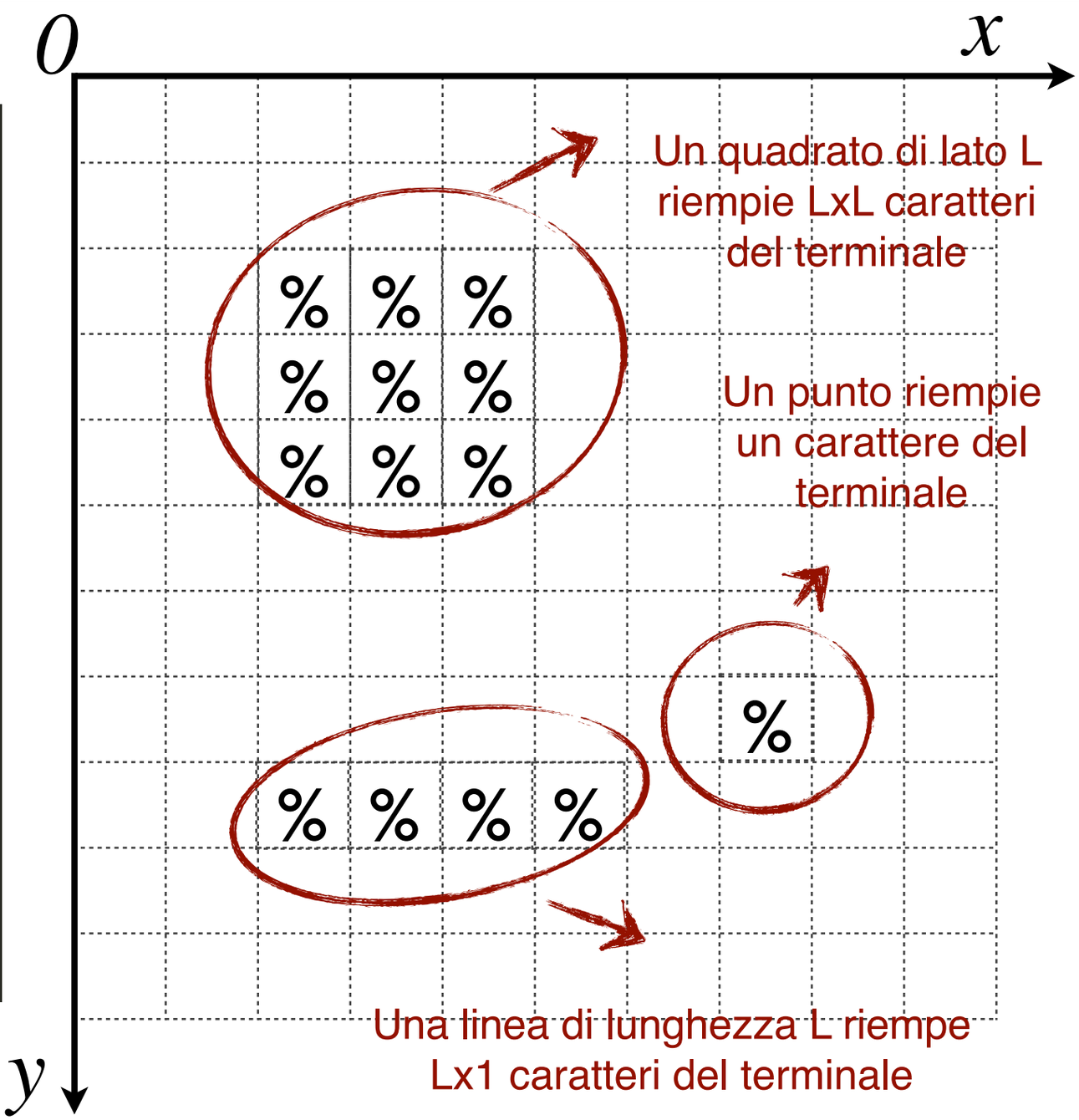
Disegna una data un generica forma 'f' nella posizione 'p' dello schermo 'schermo'

Disegna Forma

```
void disegna_forma(forma f, punto_schermo p, char schermo[SCREEN_W][SCREEN_H])
{

    int i;
    int x,y;

    for (i = 0; i < f.numero_pixel; i++)
    {
        x = f.pixels[i].x + p.x;
        y = f.pixels[i].y + p.y;

        if (x < SCREEN_W && y < SCREEN_H && x >= 0 && y >= 0)
            schermo[x][y] = f.pixels[i].valore;
    }

}
```

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);
```

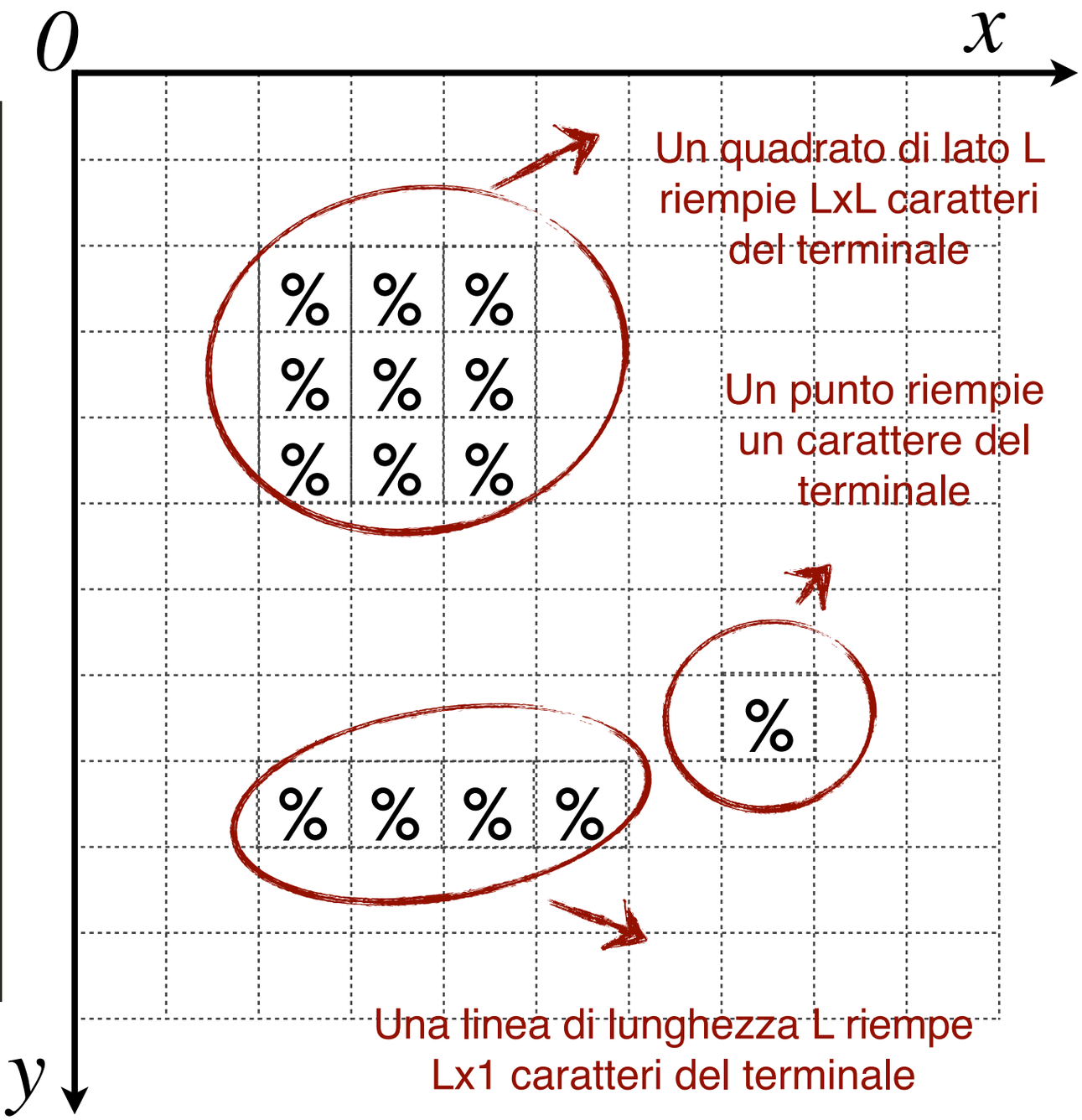
Restituisce una forma quadrata di lato 'dim'

Genera quadrato

```
forma genera_quadrato(int dim, char carattere)
{
    forma quadrato;
    int x,y,cont;

    cont = 0;

    for (y = 0; y < dim; y++){
        for (x = 0; x<dim; x++){
            quadrato.pixels[cont].x = x;
            quadrato.pixels[cont].y = y;
            quadrato.pixels[cont].valore = carattere;

            cont++;
        }
    }

    quadrato.numero_pixel = cont;
    quadrato.categoria = F_POLIGONO_QUADRILATERO;

    return quadrato;

}
```

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);
```

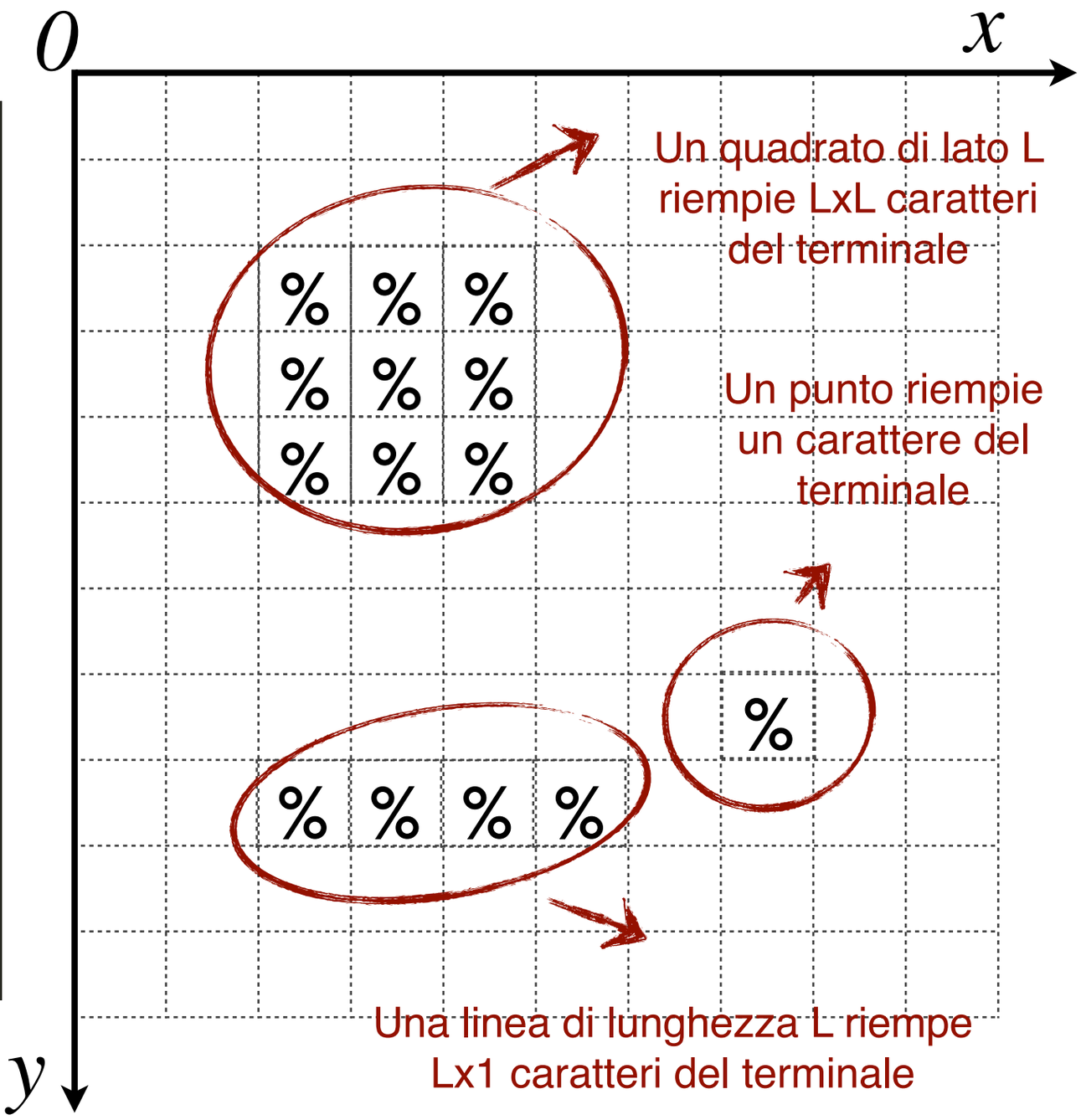
Restituisce una forma linea di lunghezza 'dim' con direzione ‘direzione_linea

Genera linea

```
forma genera_linea(int dim, direzione direzione_linea, char carattere)
{
    forma linea;
    int i,cont;

    cont = 0;

    for (i = 0; i<dim; i++){

        if (direzione_linea == D_VERTICALE)
        {
            linea.pixels[cont].x = 0;
            linea.pixels[cont].y = i;
        }
        else if (direzione_linea == D_ORIZZONTALE)
        {
            linea.pixels[cont].x = i;
            linea.pixels[cont].y = 0;
        }
        else
            printf("Errore direzione linea!\n");

        linea.pixels[cont].valore = carattere;

        cont++;
    }

    linea.numero_pixel = cont;
    linea.categoria = F_LINEA;

    return 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);
```

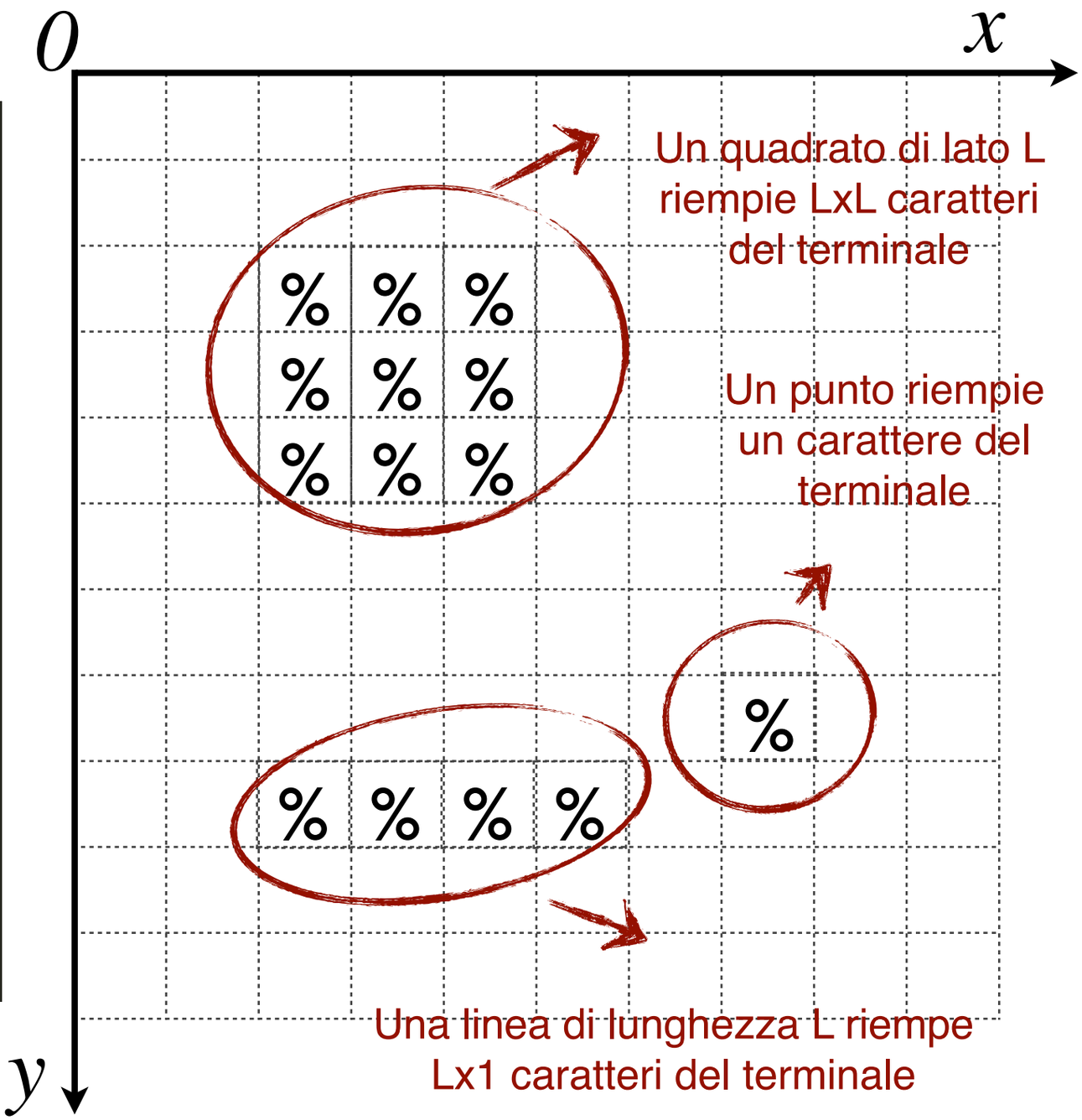
Restituisce una forma punto (linea di lunghezza 1)

Genera punto

```
forma genera_punto(char carattere)
{
    forma punto = genera_linea(1, D_VERTICALE, carattere);
    punto.categoria = F_PUNTO;

    return punto;
}
```

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

Genera polinomio

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


    forma polinomio;
    int x,y,cont = 0;

    for (x = 0; x < dim; x++){
        y = a2 * (x*x) + a1 * x + a0;
        polinomio.pixels[cont].x = x;
        polinomio.pixels[cont].y = y;
        polinomio.pixels[cont].valore = carattere;
        cont++;
    }

    polinomio.numero_pixel = cont;
    polinomio.categoria = F_GENERICA;

    return polinomio;

}
```

Implementiamo il main()

```
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();



// Spostiamo il quadrato
inizializza_schermo(schermo);
pulisci_terminale();

quadrato = genera_quadrato(4, '#');
p = crea_punto_schermo(10,10,0);

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

aspetta_invio();
```

```
    // Aggiungiamo una linea verticale, una orizzontale ed un punto

    linea_or = genera_linea(9, D_ORIZZONTALE, '#');
    p = crea_punto_schermo(7,1,0);

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


    linea_vr = genera_linea(5, D_VERTICALE, '#');
    p = crea_punto_schermo(7,3,0);

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


    punto = genera_punto('#');
    p = crea_punto_schermo(2,2,0);

    disegna_forma(punto, p, schermo);

    disegna_schermo(schermo);

    aspetta_invio();


    // Spostiamo il quadrato
    inizializza_schermo(schermo);
    pulisci_terminale();
    forma polinomio = genera_polinomio(0,-2,0,10,'@');

    p = crea_punto_schermo(0,20,0);

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

    aspetta_invio();


}
```

PIANO CARTESIANO

See You Next Time!

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it

IEIM

Esercizio: Forza Quattro

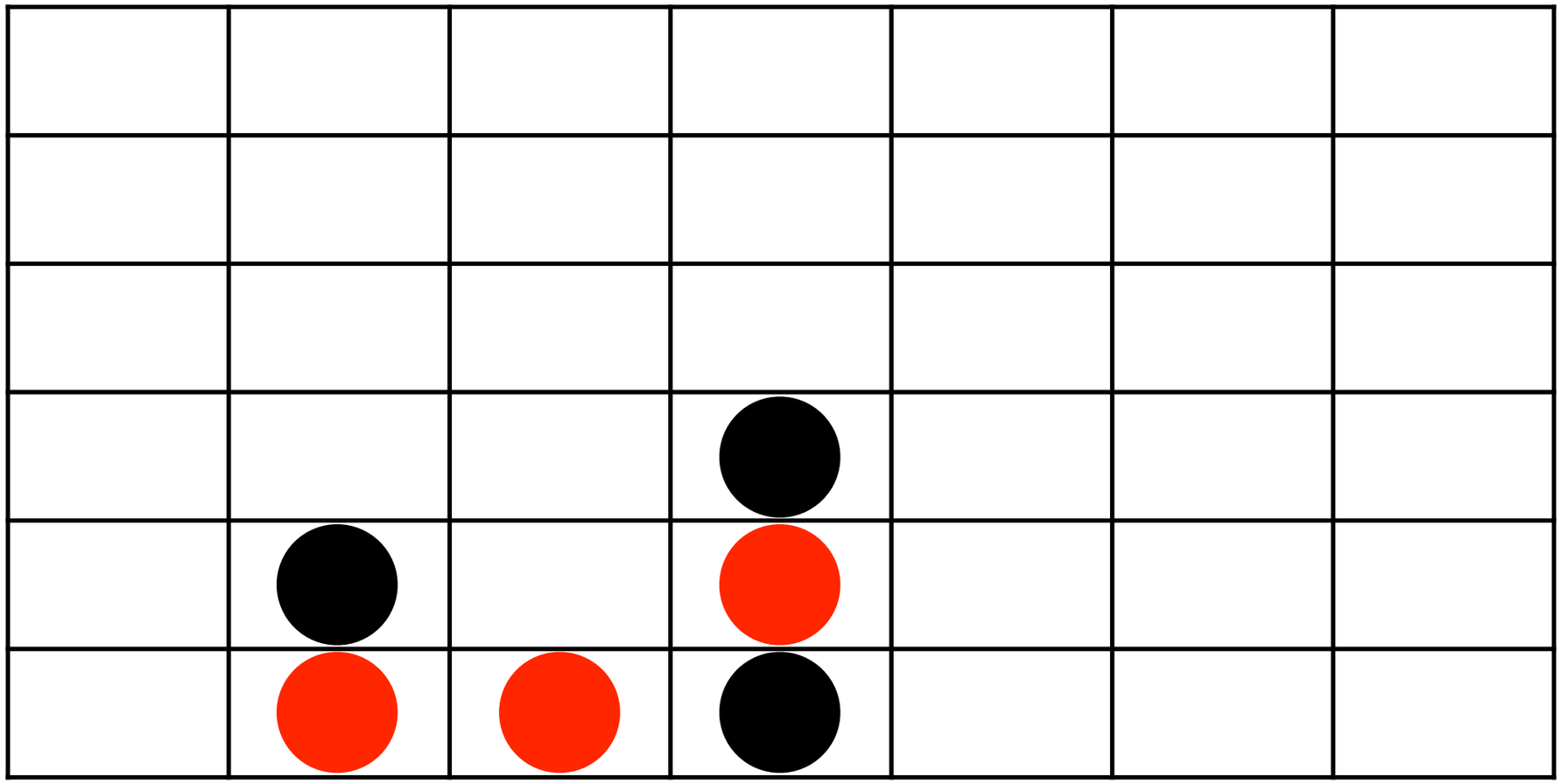
Alessandro A. Nacci
nacci@elet.polimi.it - alessandronacci.it


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

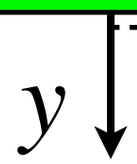


Come facciamo a rappresentare nel terminale, con quello che già conosciamo, delle pedine rosse e delle pedine nere?

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

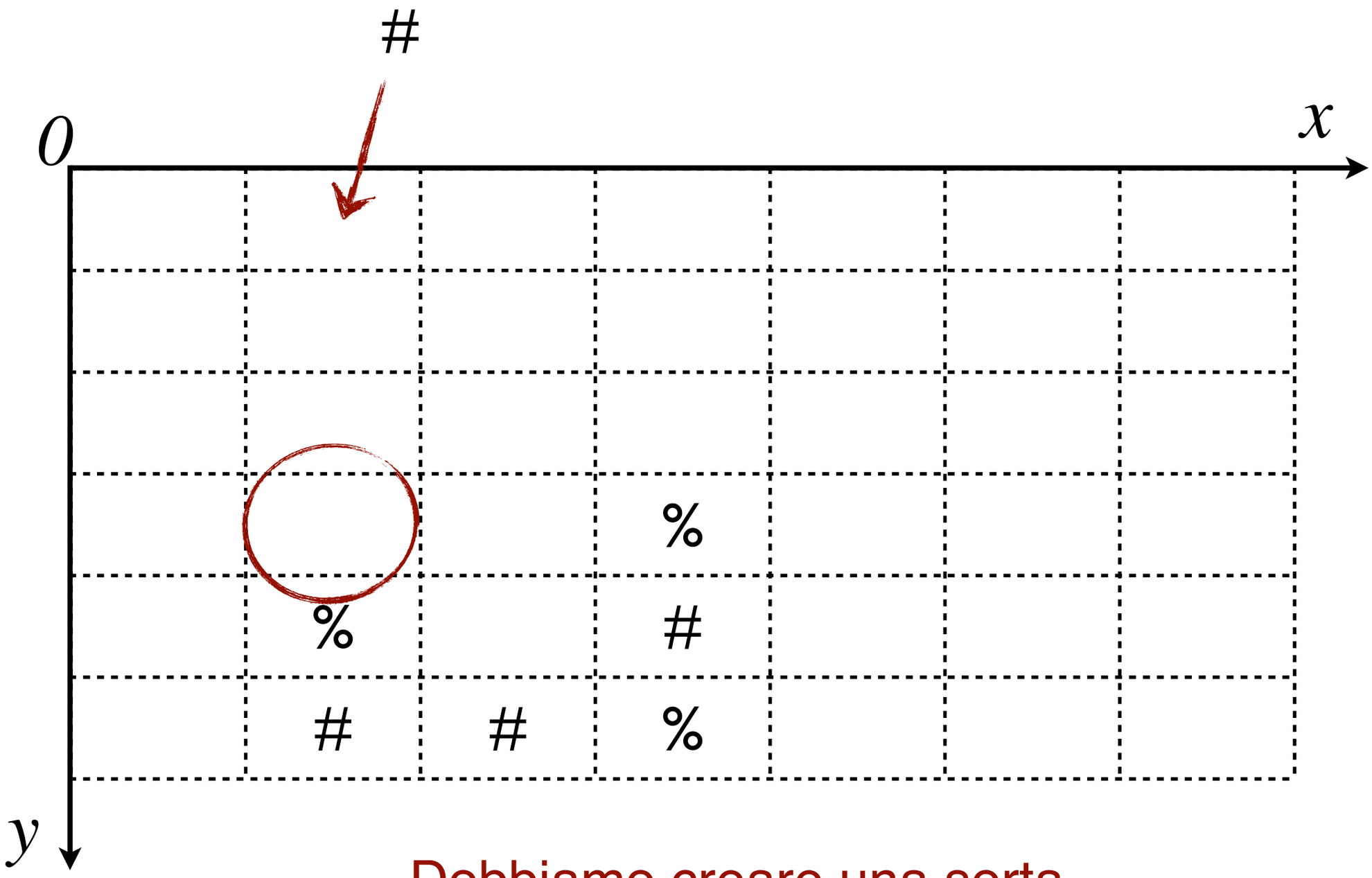
- Anche se possiamo utilizzare buona parte del codice già scritto per il "PIANO CARTESIANO", qualche concetto ancora ci manca:
  - Come rappresentiamo un GIOCATORE?
  - Come gestiamo l'inserimento di una pedina?
  - Come controlliamo la vincita?

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

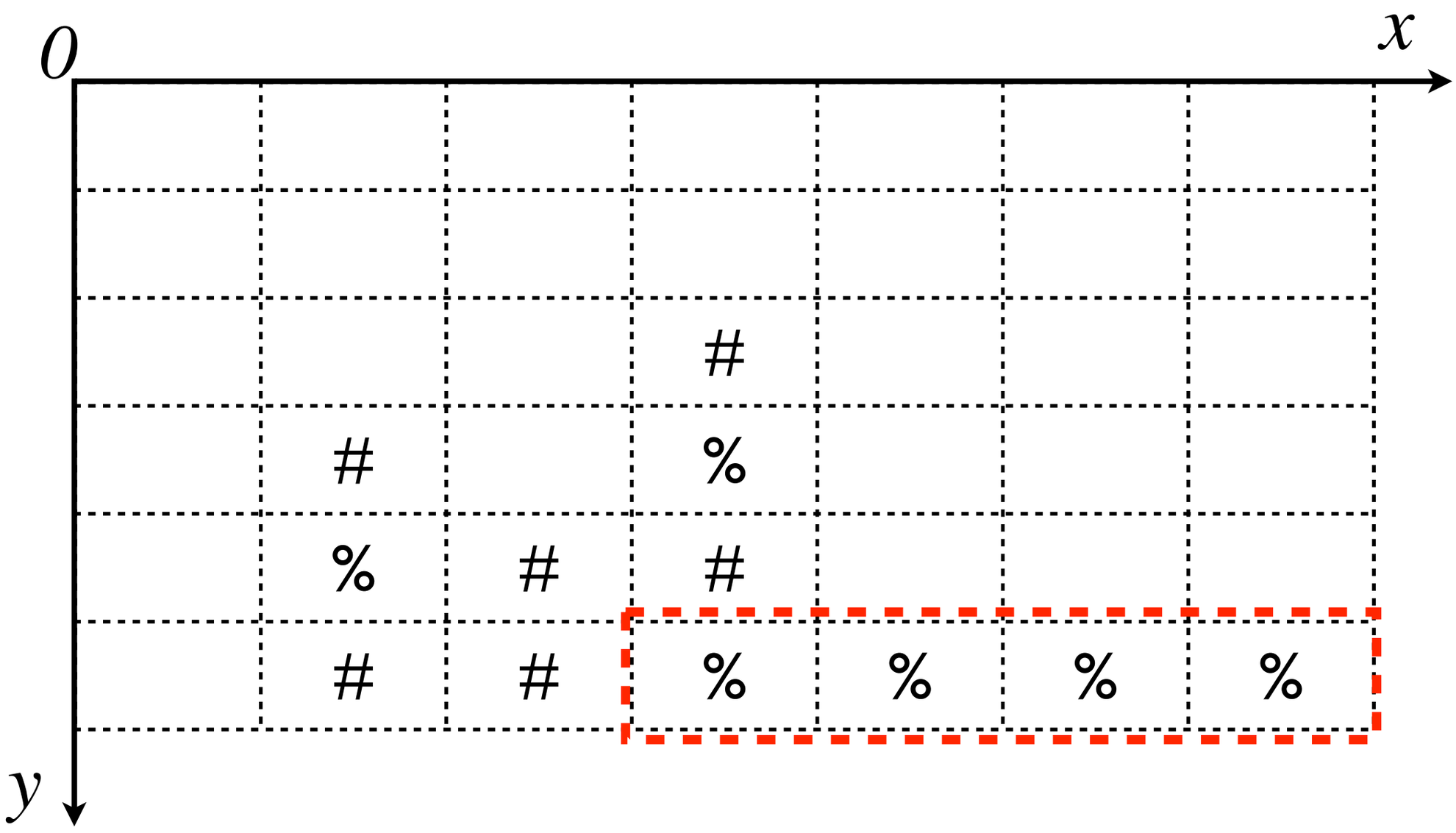


Dobbiamo creare una sorta
di effetto gravita!

L'inserimento di una pedina: codice C

```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



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;
}
```

Controllo vincita orizzontale

```c
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;

}
```

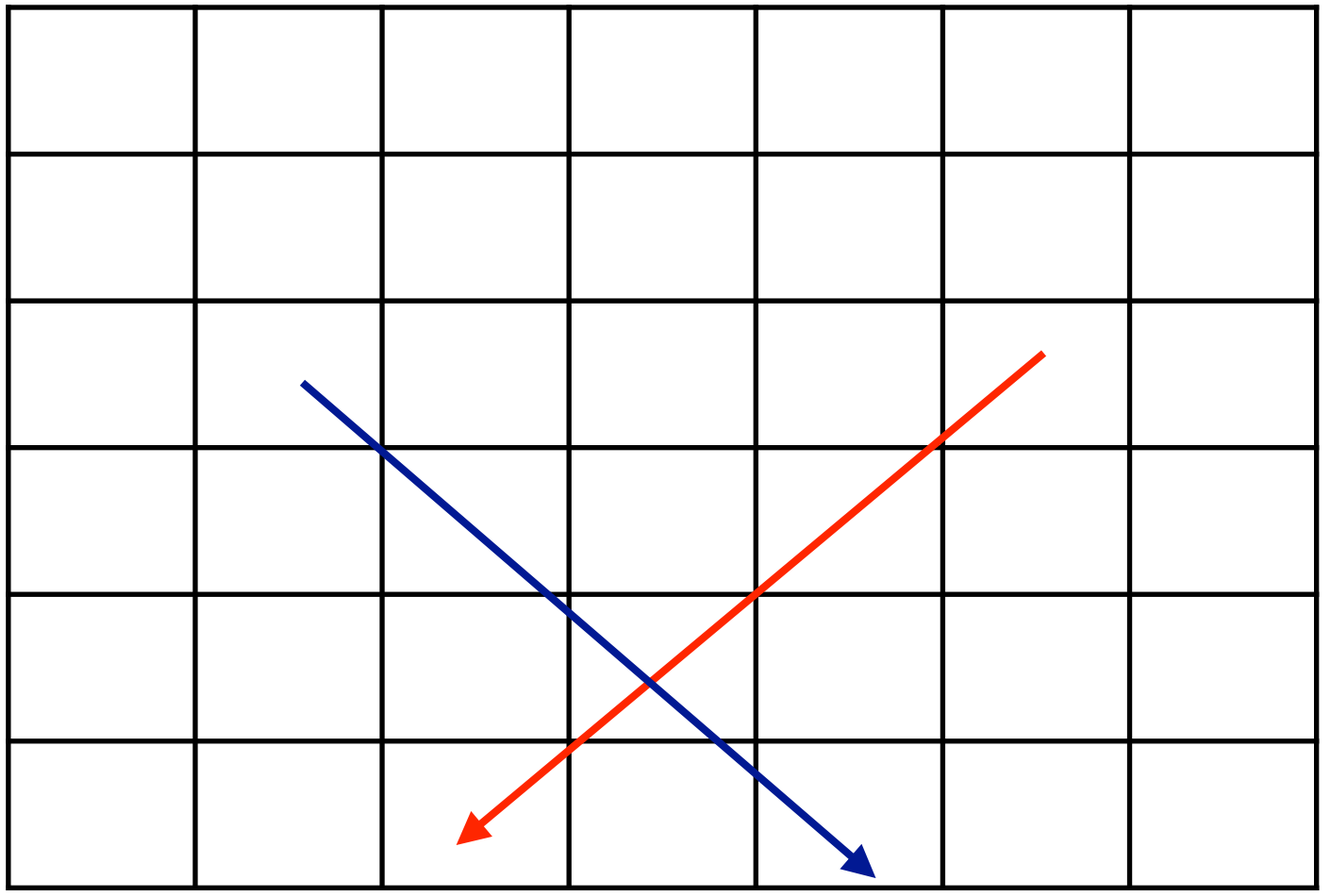
Controlla vincita verticale

```
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;

}
```

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)

```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);

}
```

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)
{

    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)
{




}
```

- 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)
{
    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

```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