IEIM 2016-2017

Esercitazione VI
"Puntatori e Struct"

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

Puntatori e memoria

Esercizio 2

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO
9	

ATTENZIONE!

Il comportamento della memoria mostrato in questo esercizio non è del tutto coerente con quanto avviene su un reale calcolatore. L'esempio mostrato è però funzionale alla spiegazione del comportamento dei puntatori in C.

```
}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

DATO UN PROGRAMMA C, COME SI COMPORTA LA MEMORIA?

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1		
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	~	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2		
3		
4		
5		
6		
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7		
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	~	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

11

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	~	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8		
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9		

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

Esercizio 2: Puntatori e memoria

Indicare nella tabella come il programma C mostrato modifica lo stato della memoria del calcolatore.

IND.	CONTENUTO	
0	3	a
1	0	b
2	3	mat
3	~	mat
4	13	mat
5	~	mat
6	~	mat
7	5	c
8	3	d
9	~	e

```
int main()
{
  int a = 3;
  printf("%d", &a);
  int* b;
  b = &a;

  int mat[2][2];
  int* c;
  mat[0][1] = 13;
  c = &(mat[1][0]);
  int d = *b;
  int e;

}
```

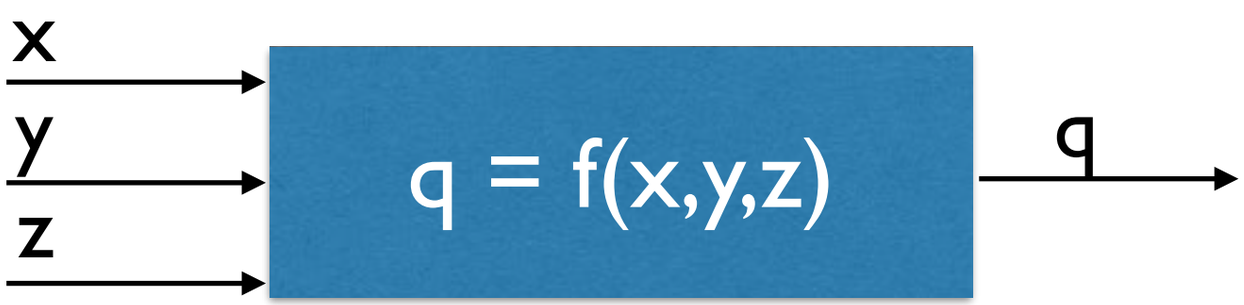
Disclaimer

- Fino ad ora **non** avete visto le **funzioni**.
- Per poter fare un esercizio “complesso”, ci vengono utili

- **Introduciamo alcuni aspetti delle funzioni utili alla comprensione del prossimo esercizio**
- **La trattazione seguente NON è esaustiva e serve SOLO alla comprensione del prossimo esercizio!**

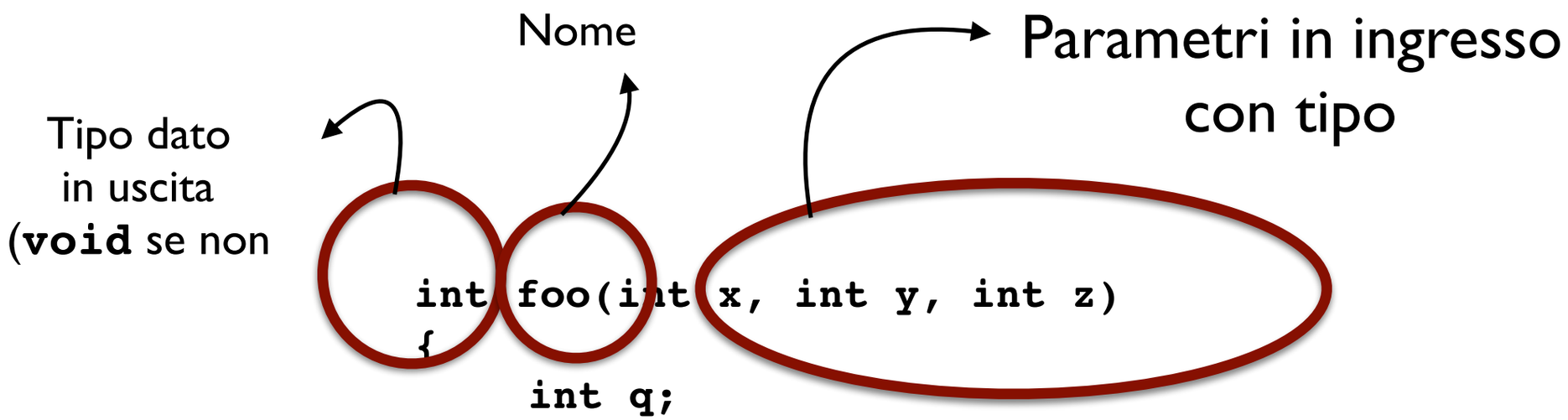
Le funzioni

- Le funzioni sono "blocchi" di codice che prendono in ingresso dei valori tramite i *parametri* e restituiscono, dopo della computazione, un risultato.




```
int foo(int x, int y, int z)
{
     int q;


     // codice vario
     // che effettua calcoli


     return q;

}
```

Le funzioni




```
    // codice vario
    // che effettua calcoli

    return q;
}
```

Singolo valore in uscita
(in C si può restituire un solo valore
alla volta)

Le funzioni con gli array

- Per alcuni motivazioni tecniche del C, con gli array ci sono alcuni "problemini"
- Semplificando, In C non è possibile fare **`return`** di un array
- **In C, quando si passa in ingresso un array come parametro, le modifiche fatte su quel parametro agiscono direttamente sull'array originale passato dal chiamante.**

Le funzioni con gli array

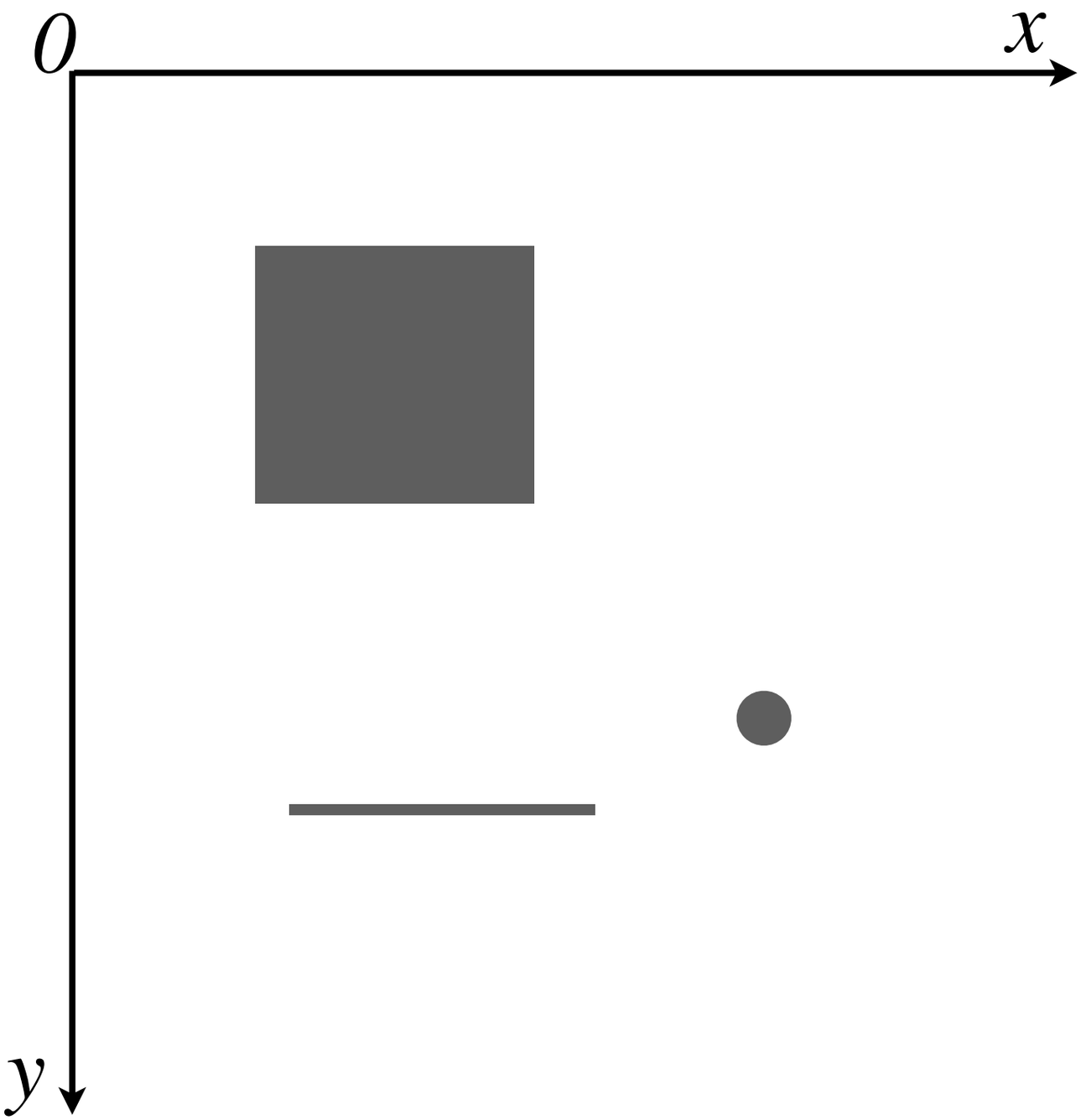
```
void foo(int x_arr[])
{
    x_arr[3] = 13;
    return;
}


int main()
{
  int x_arr[10];
  x_arr[3]= 2;

  printf(“%d”,x_arr[3]); // Stampa 2

  foo(x_arr);

  printf(“%d”,x_arr[3]); // Stampa 13
}
```

Il piano cartesiano

Specifiche dell'esercizio

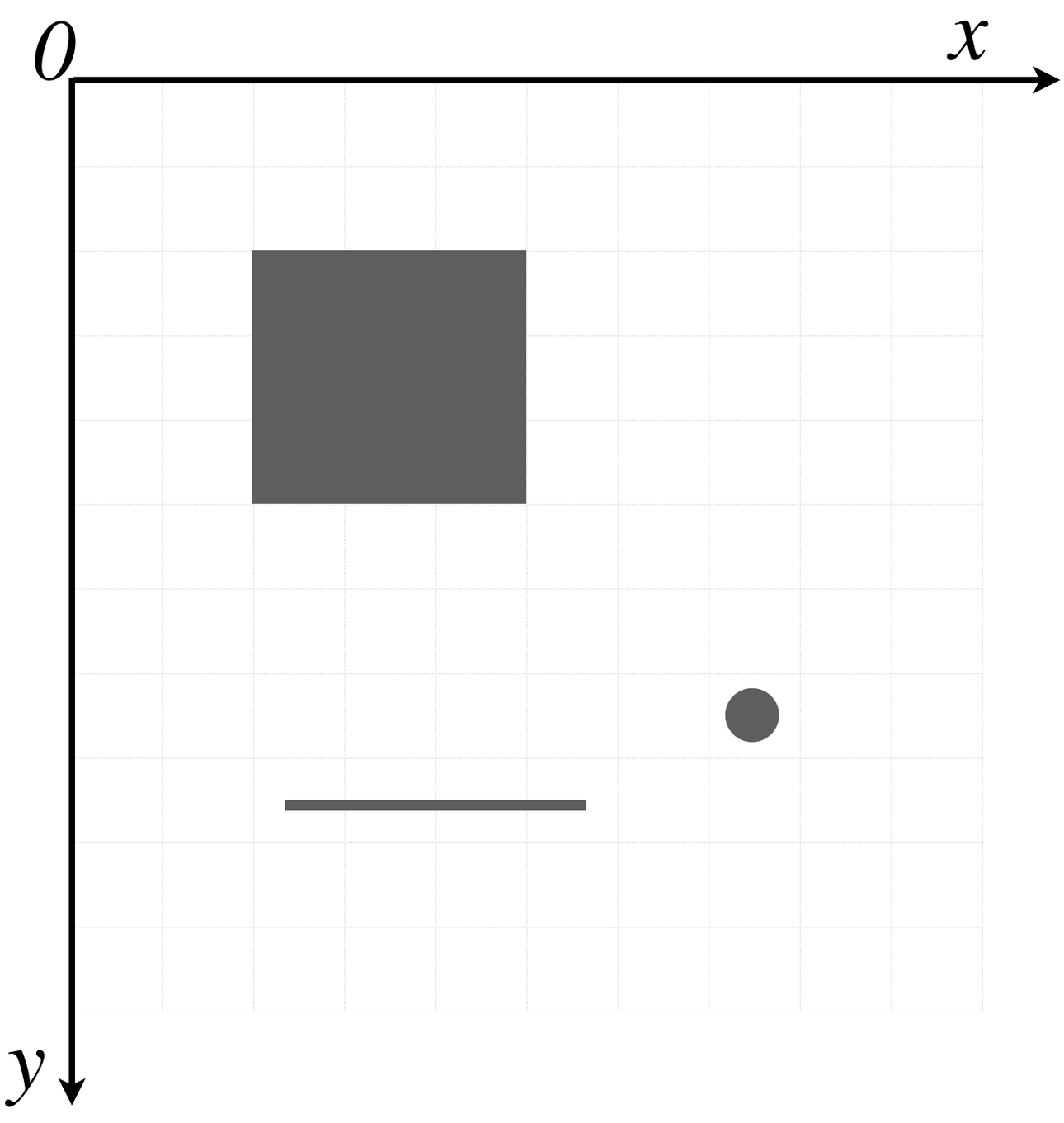
- Scrivere in C un programma che rappresenti un piano cartesiano
- Il programma deve poter rappresentare e visualizzare a schermo PUNTI, LINEE e QUADRATI
- Deve essere inoltre possibile manipolare le forme create (spostarle, cancellarle, ingrandirle, etc..)
- Il programma deve poter rappresentare la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

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



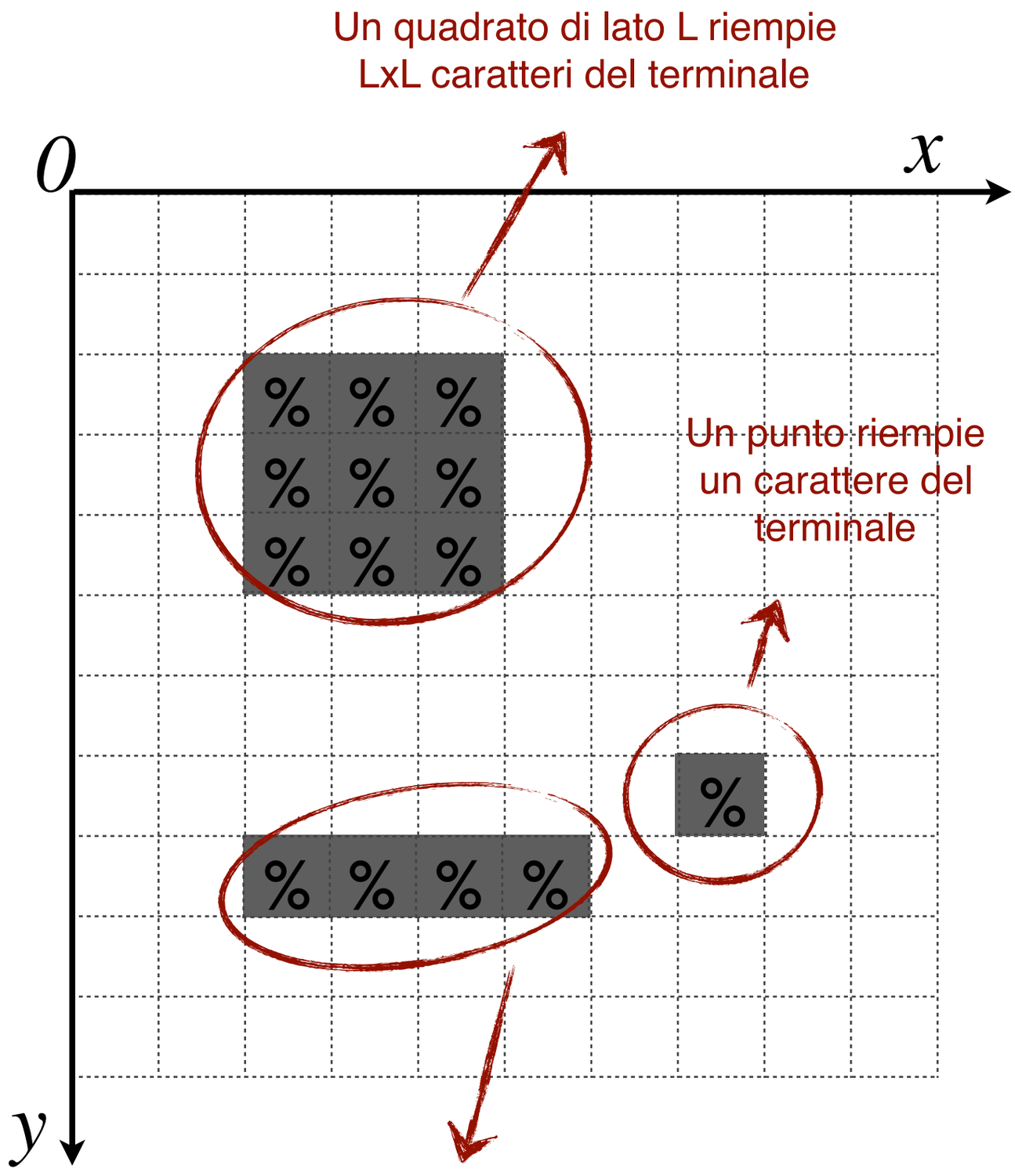
La rappresentazione

- 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



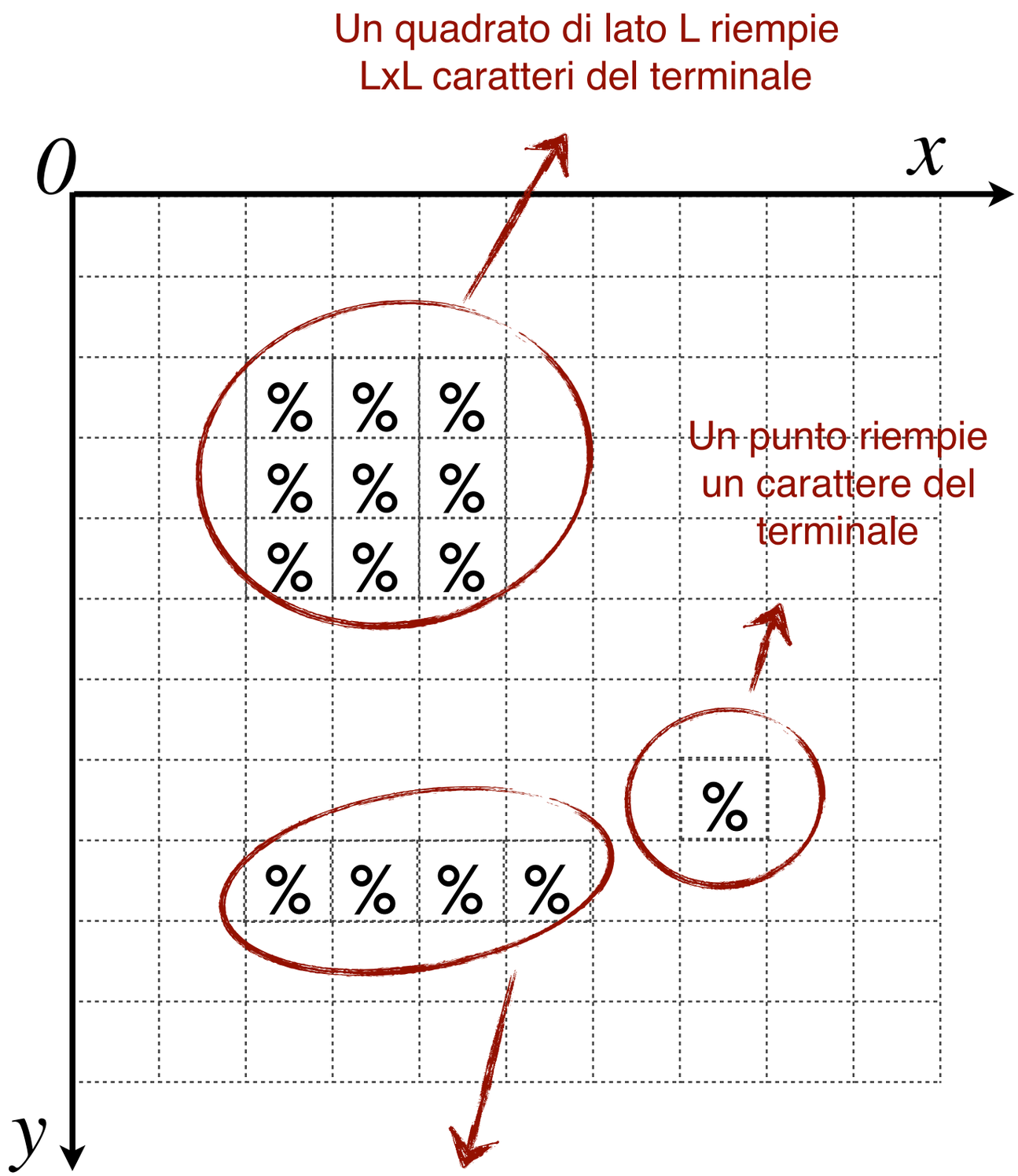
La rappresentazione

- 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



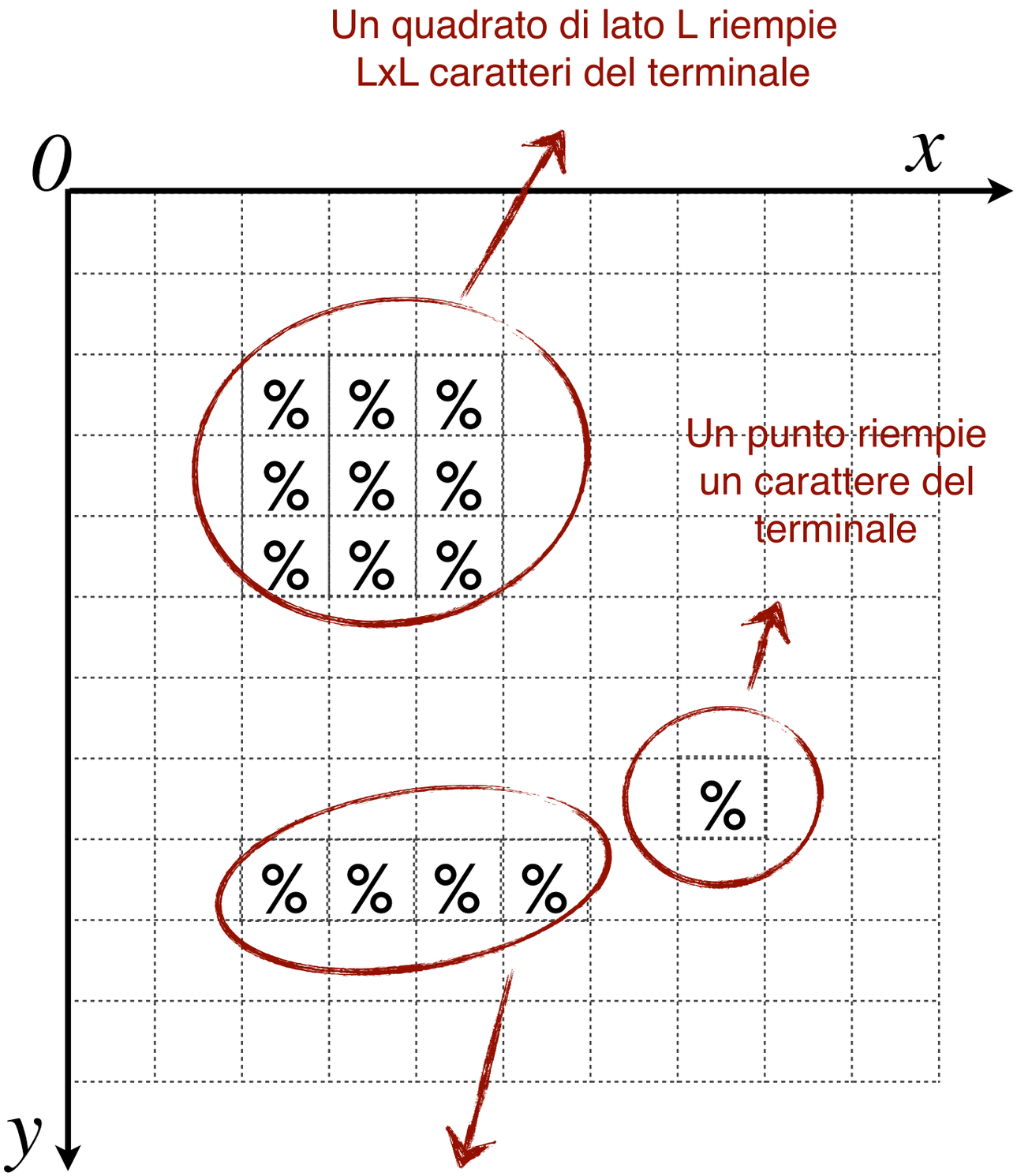
Cosa ci serve?

- 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?

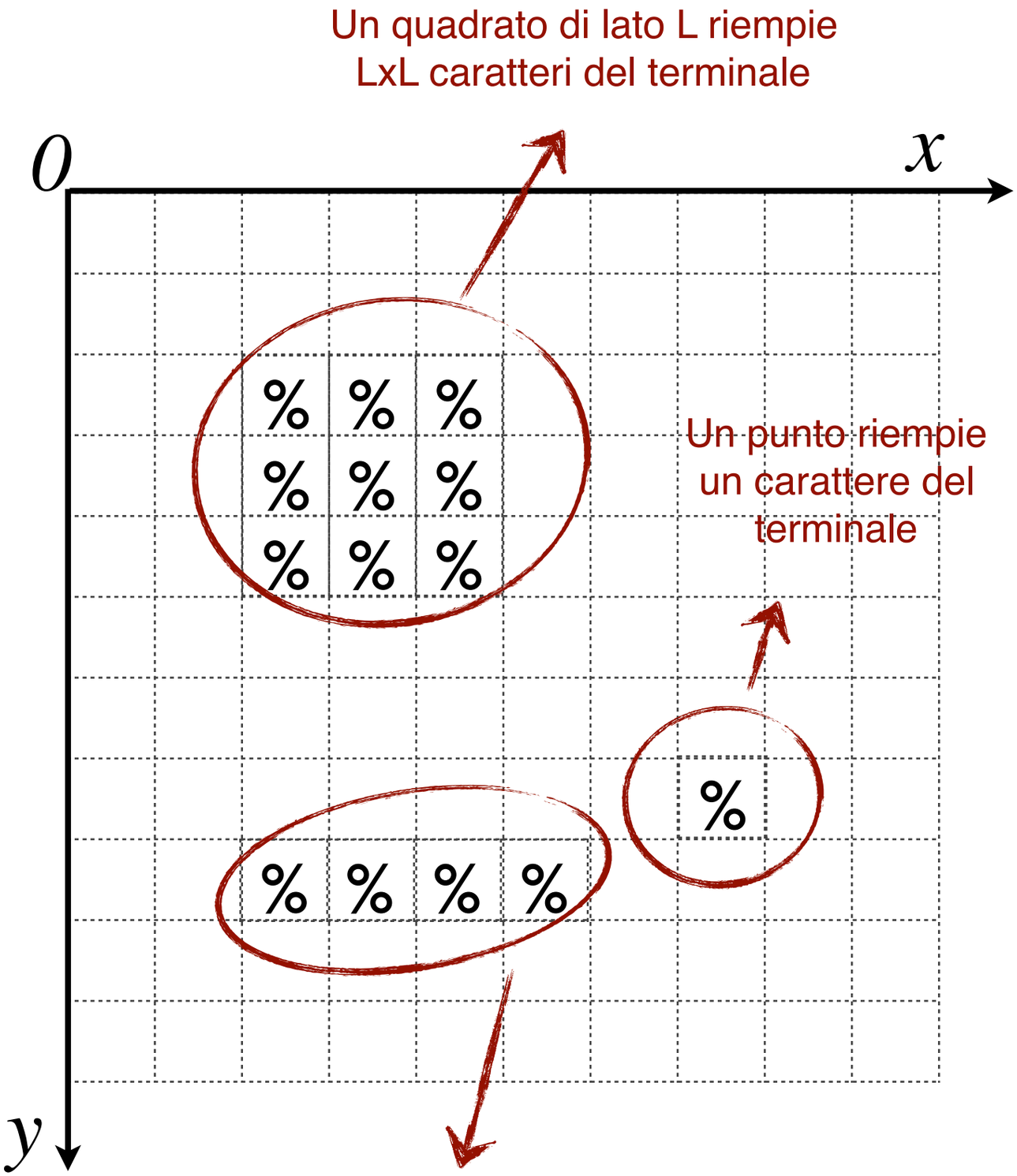
- Tendenzialmente, avremo bisogno di qualcosa tipo…

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

Le funzioni

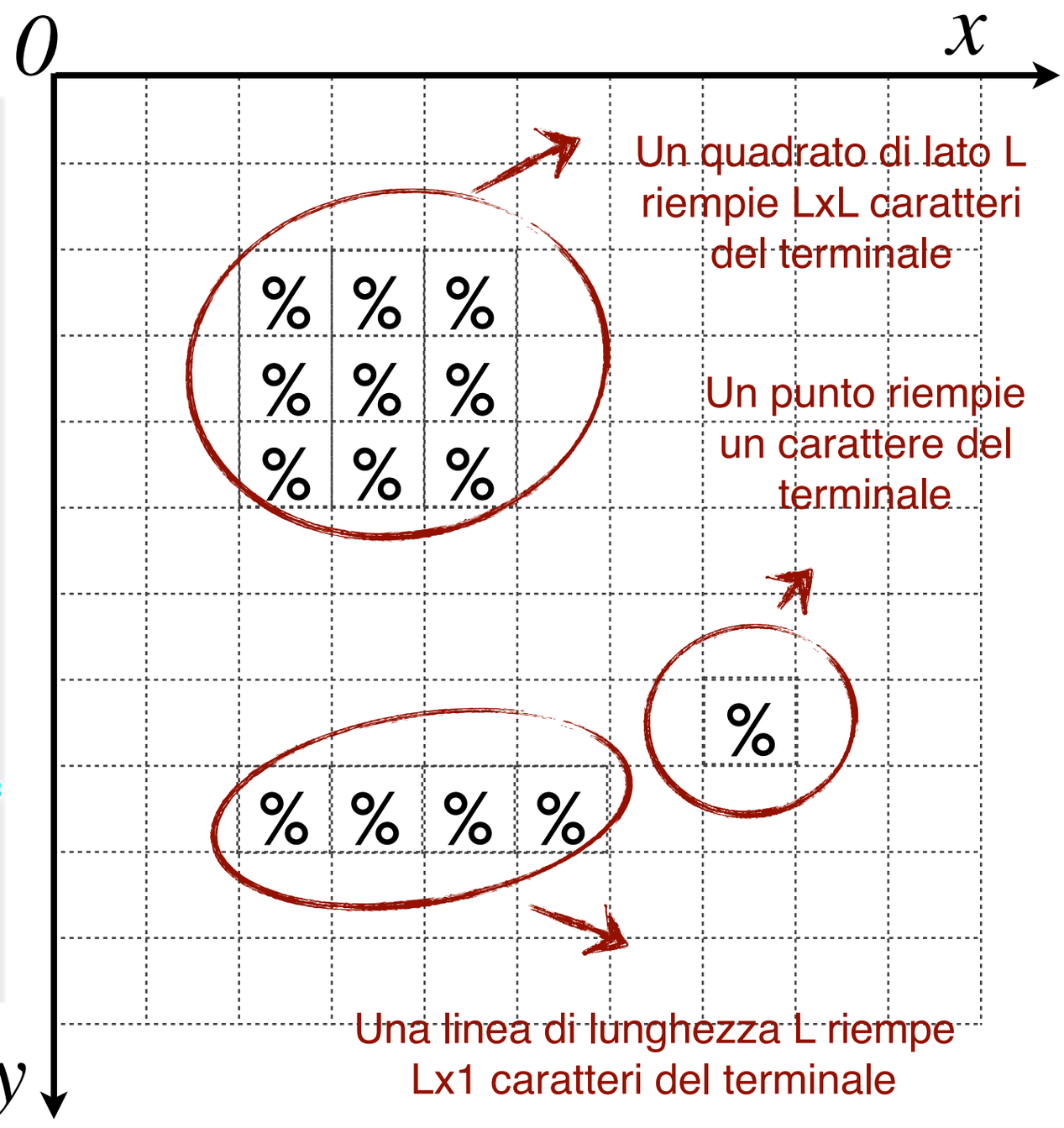
- Le funzioni che possono essere utili sono:
  - Funzioni per la visualizzazione a video
  - Modifica delle matrice che rappresenta lo schermo
  - Generazione delle forme

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]);
```

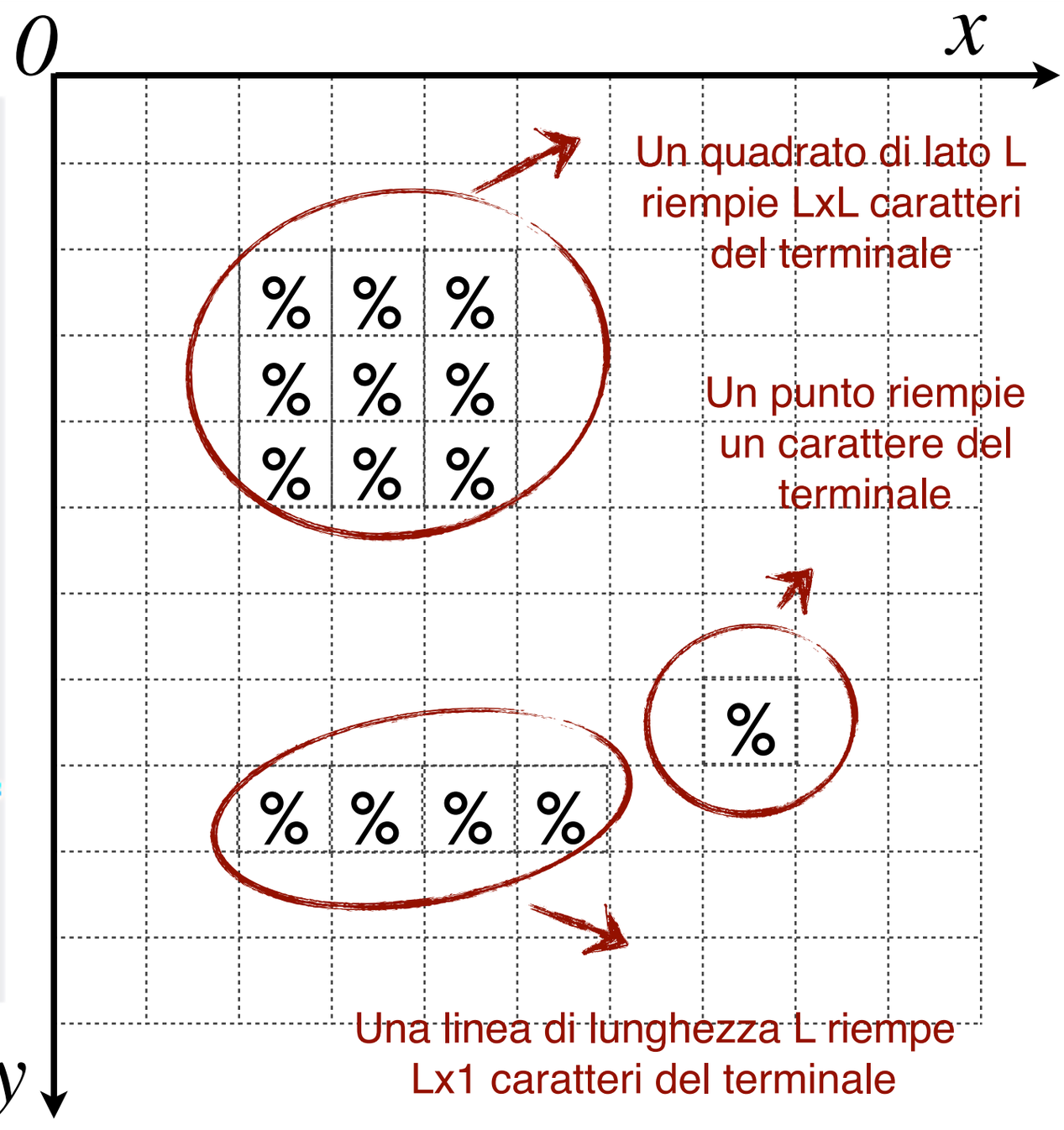
Riempie di caratteri vuoti la matrice in ingresso 'schermo'

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]);
```

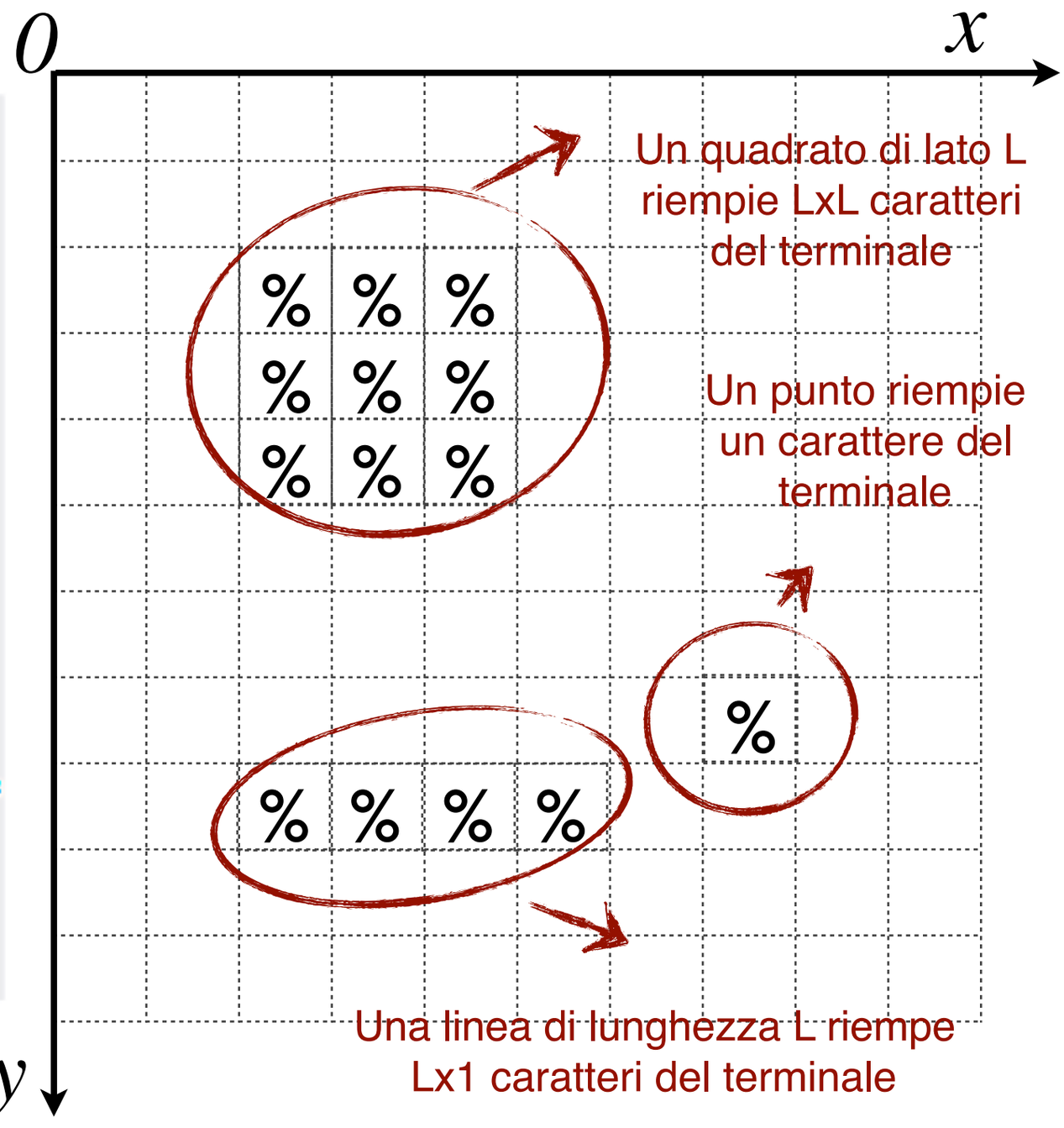
Inserisce il carattere '%' sui bordi della matrice 'schermo'

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 scheno
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]);
```

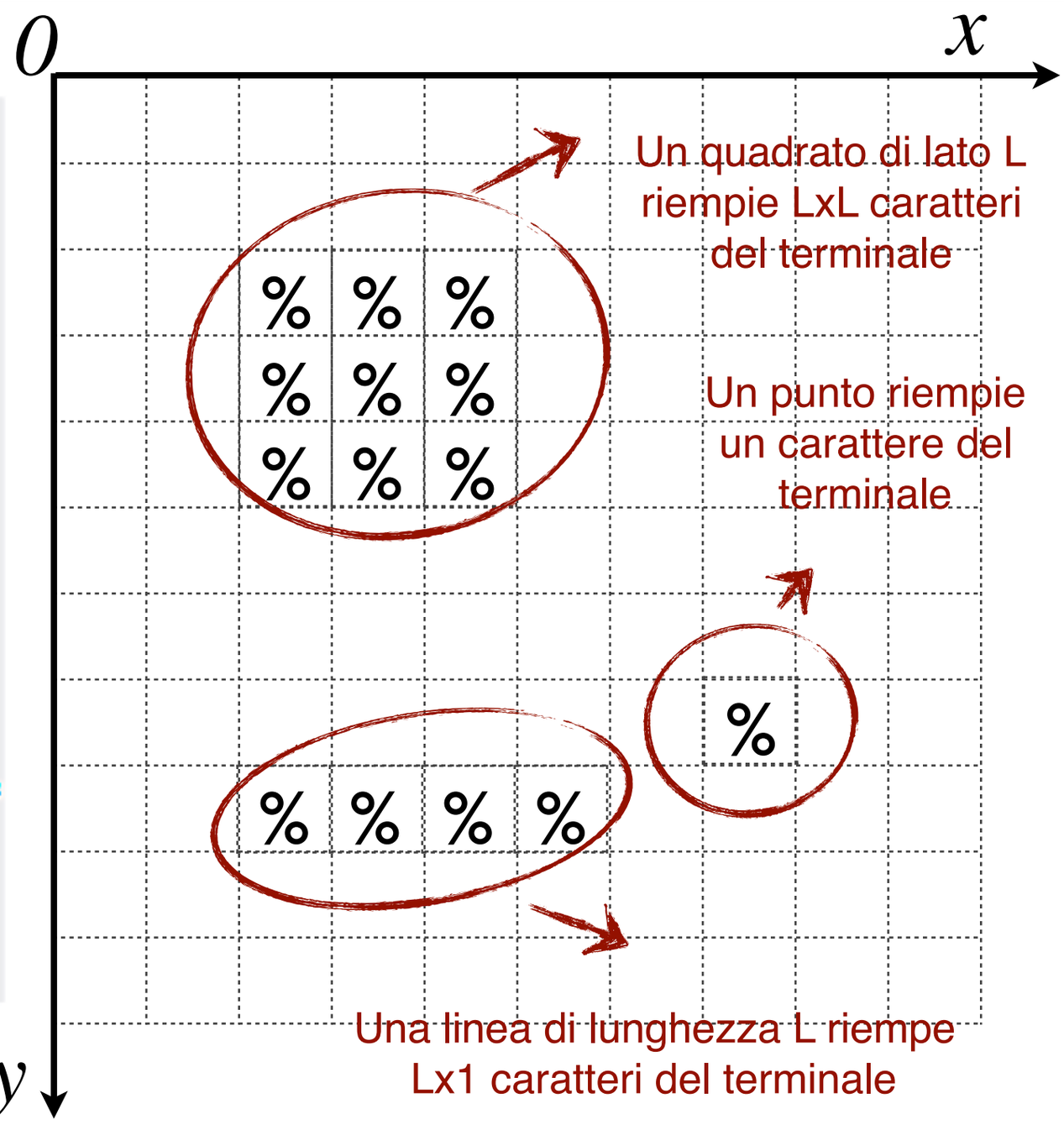
Disegna a schermo la matrice in ingresso 'schermo'

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();
```

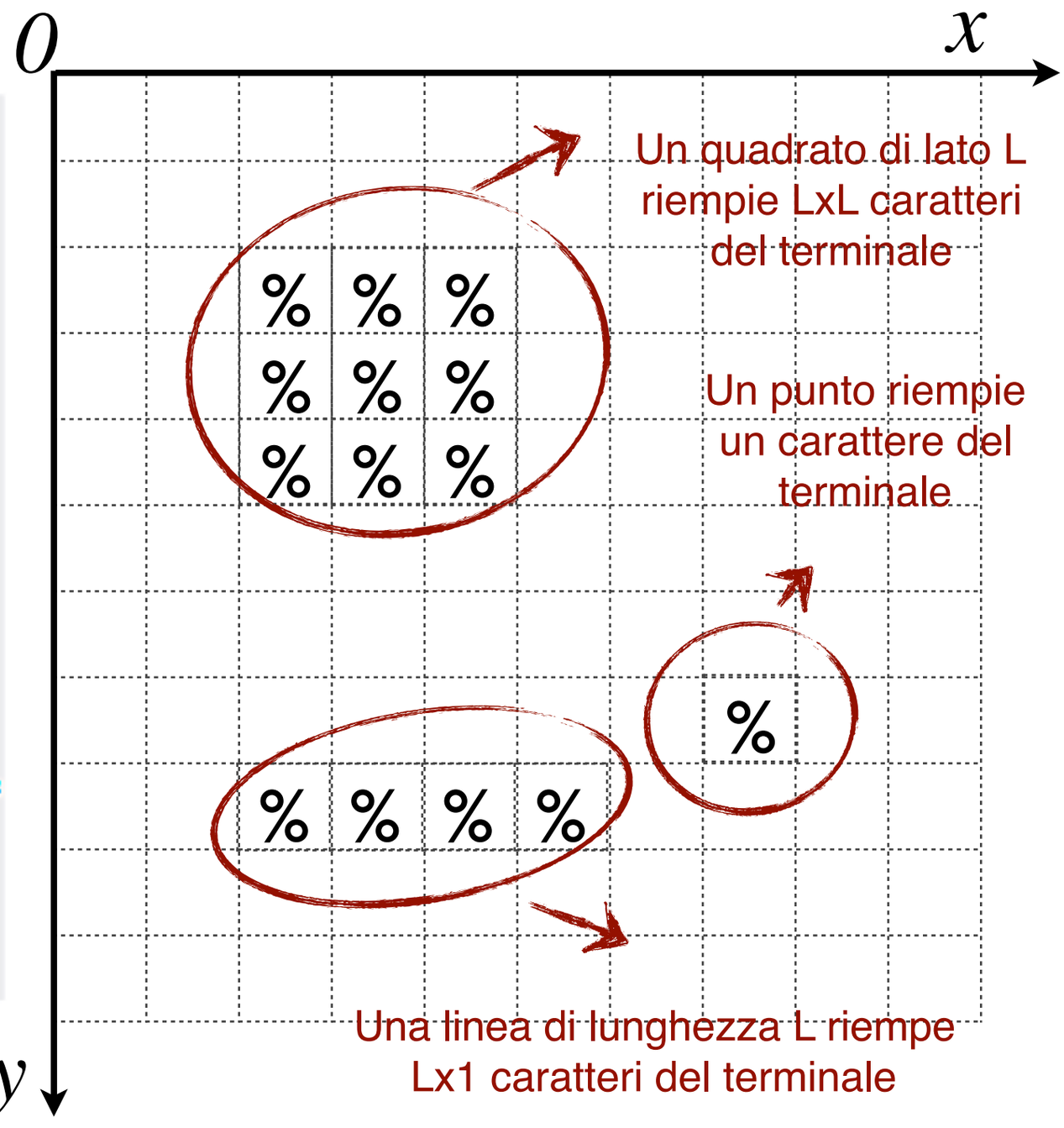
Quando lanciata, mette il programma in attesa di un INVIO da parte dell'utente

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);
```




```c
void pulisci_terminale();
```

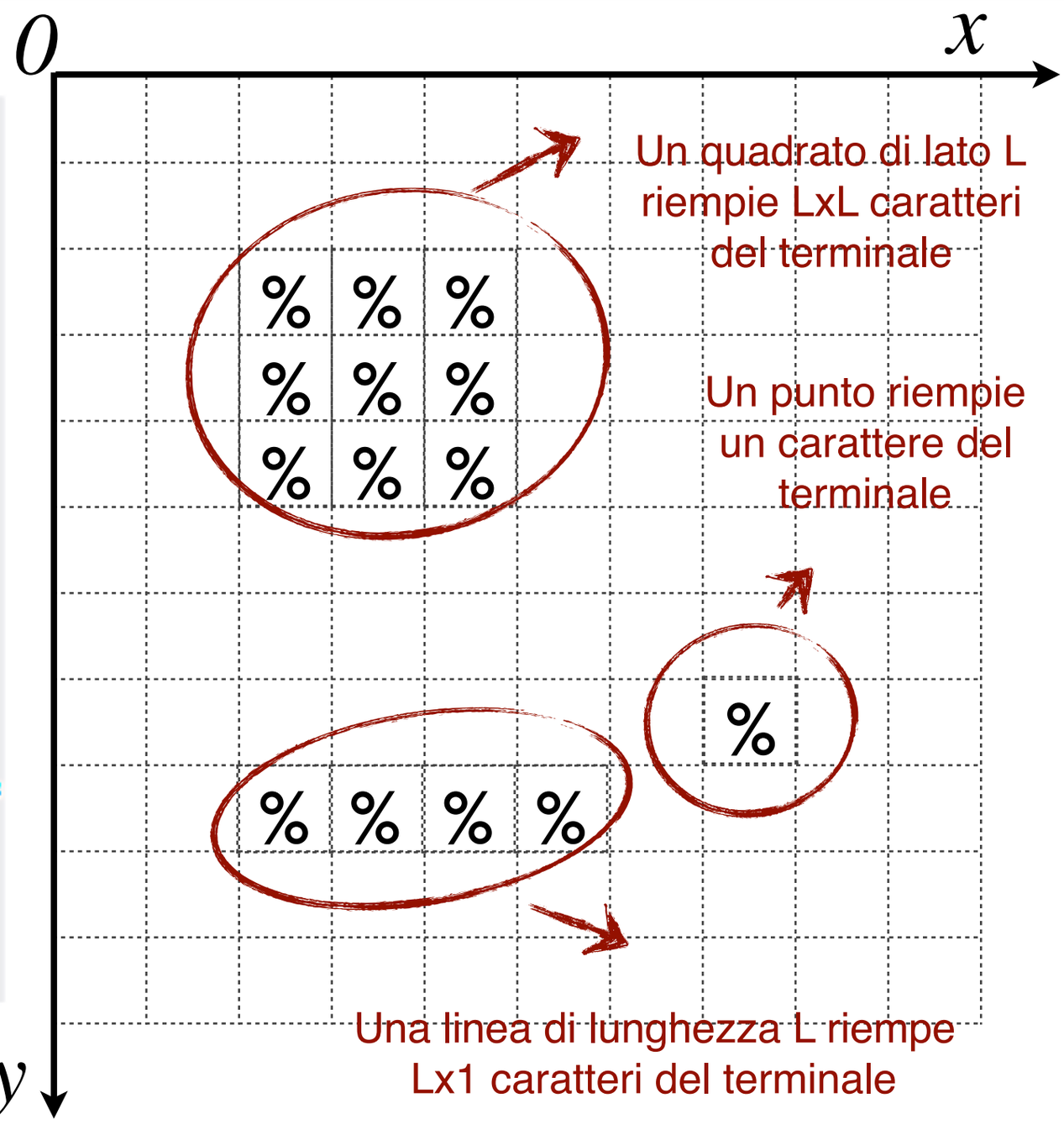
Cancella il contenuto del termianale su cui viene stampato l'output del programma

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

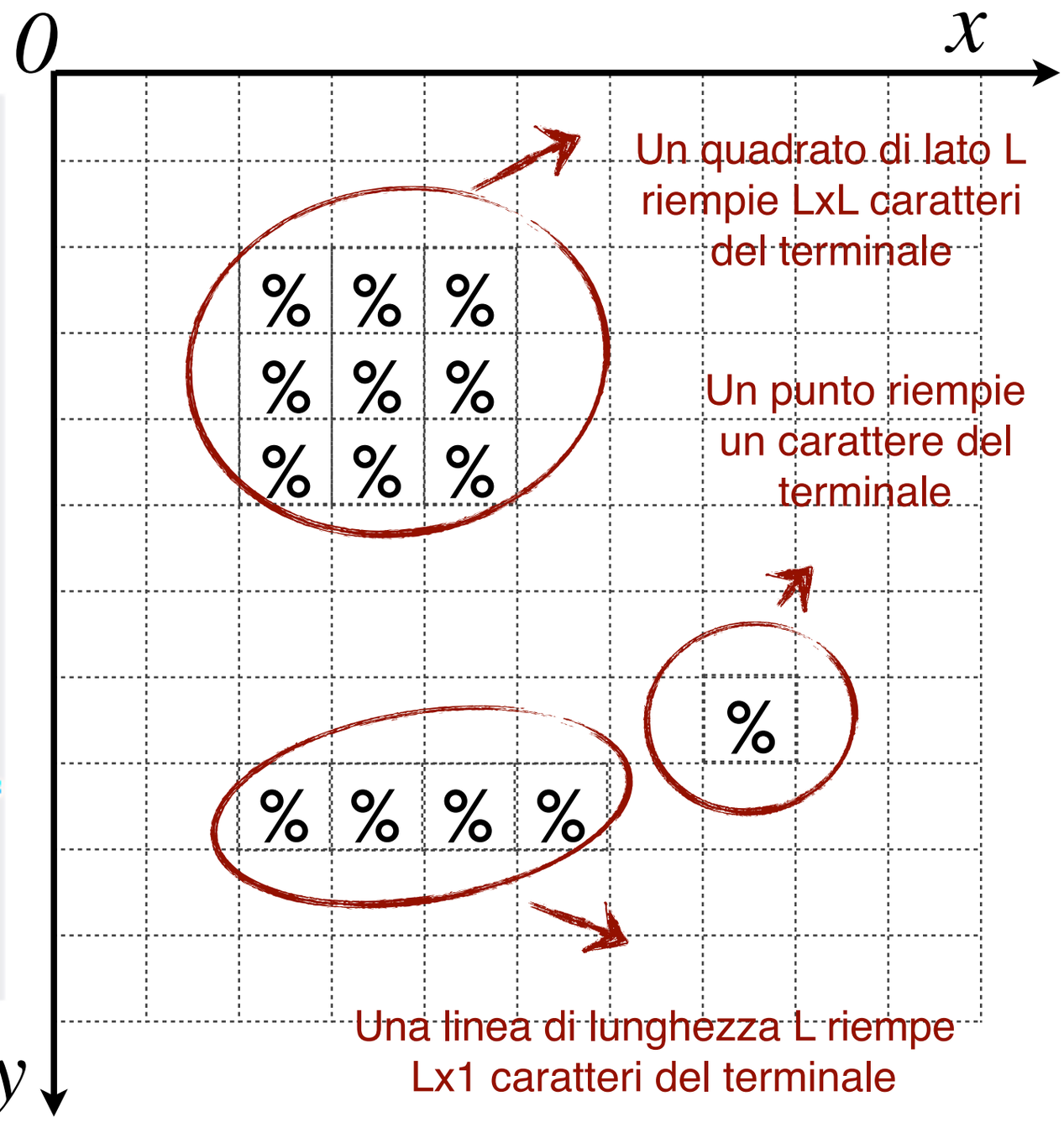
Disegna una data un generica forma 'f' nella posizione 'p' dello schermo 'schermo'

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);
```

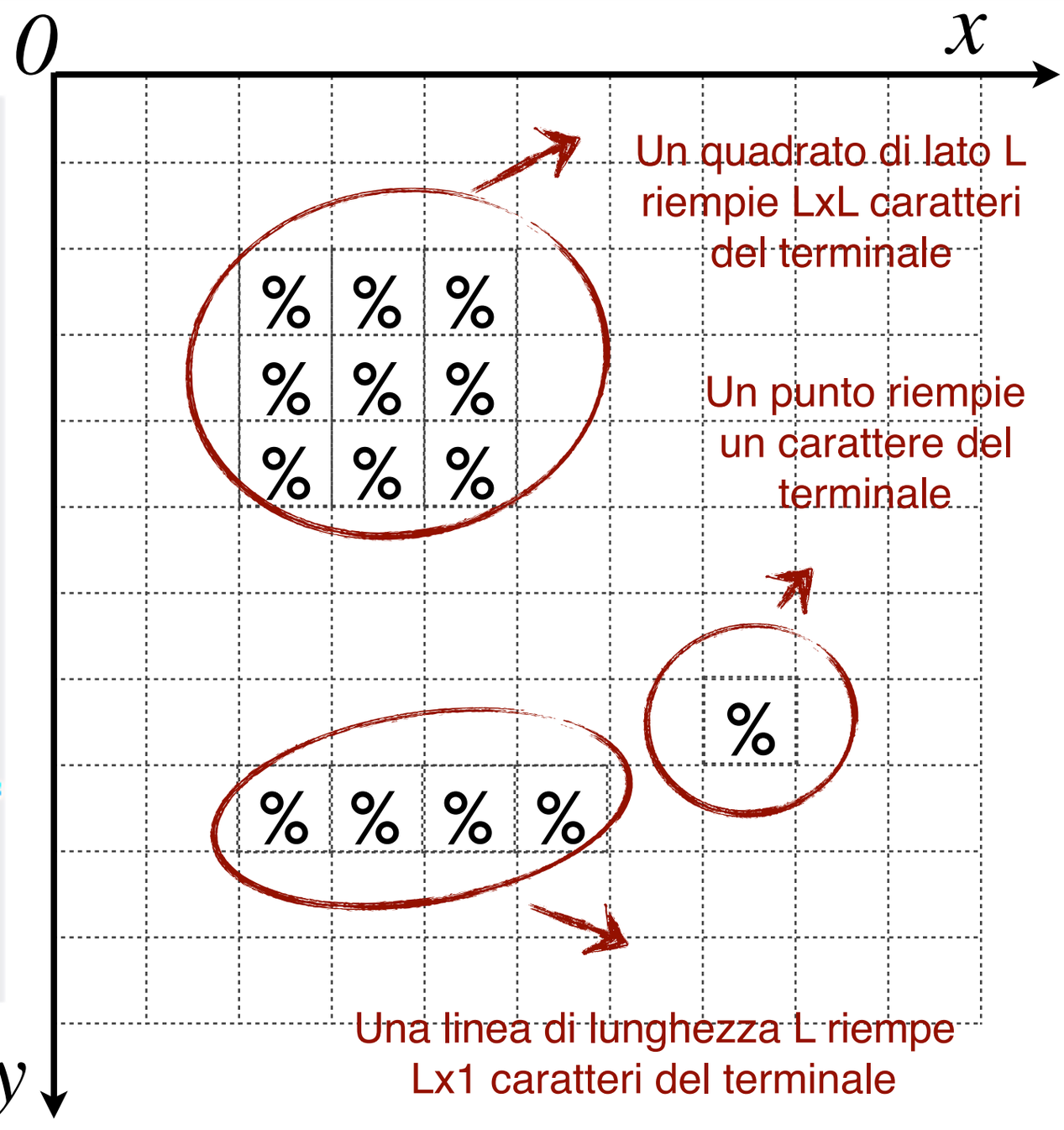
Restituisce una forma quadrata di lato 'dim'

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);
```

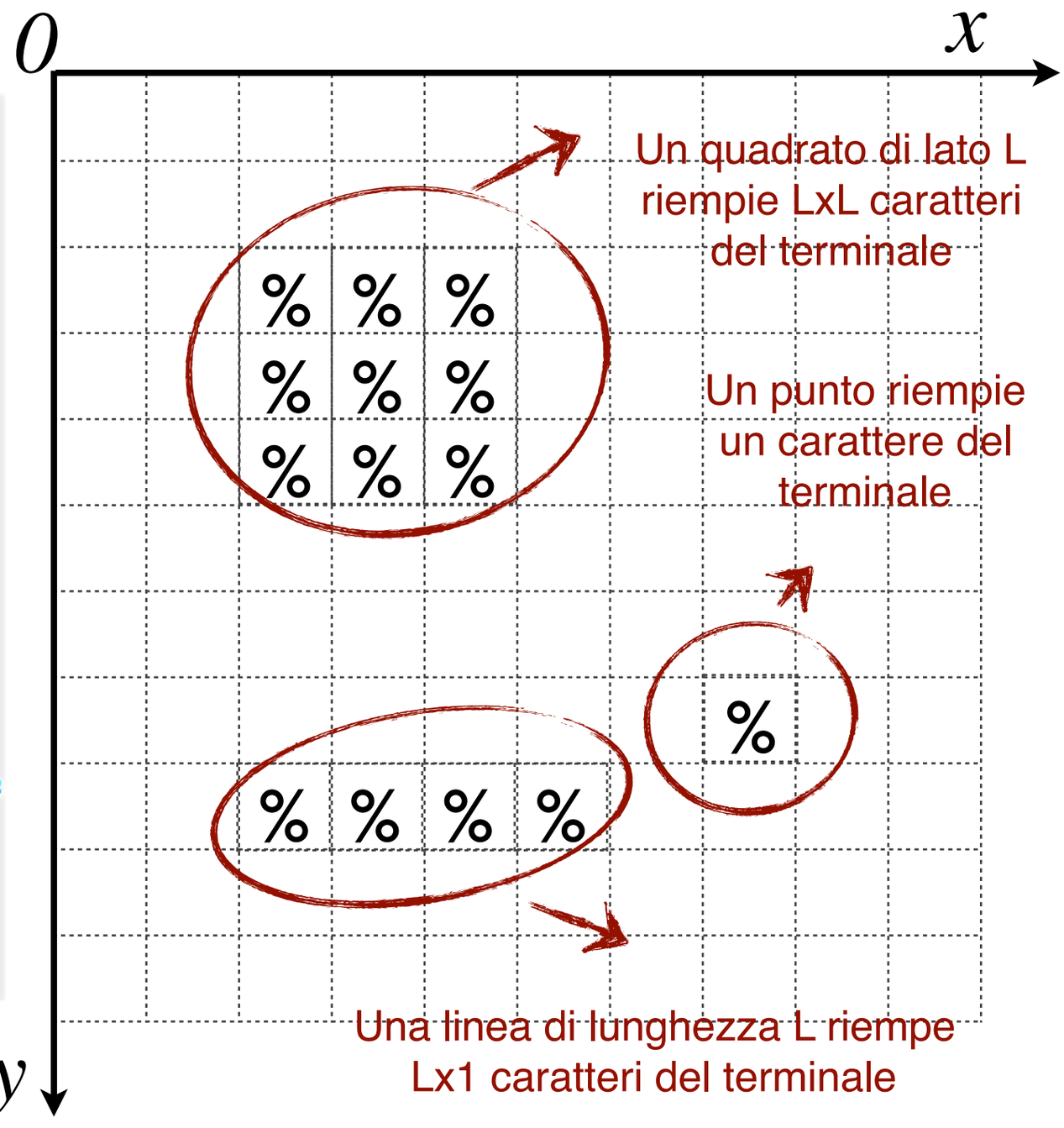
Restituisce una forma linea di lunghezza 'dim' con direzione ‘direzione_linea

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);
```

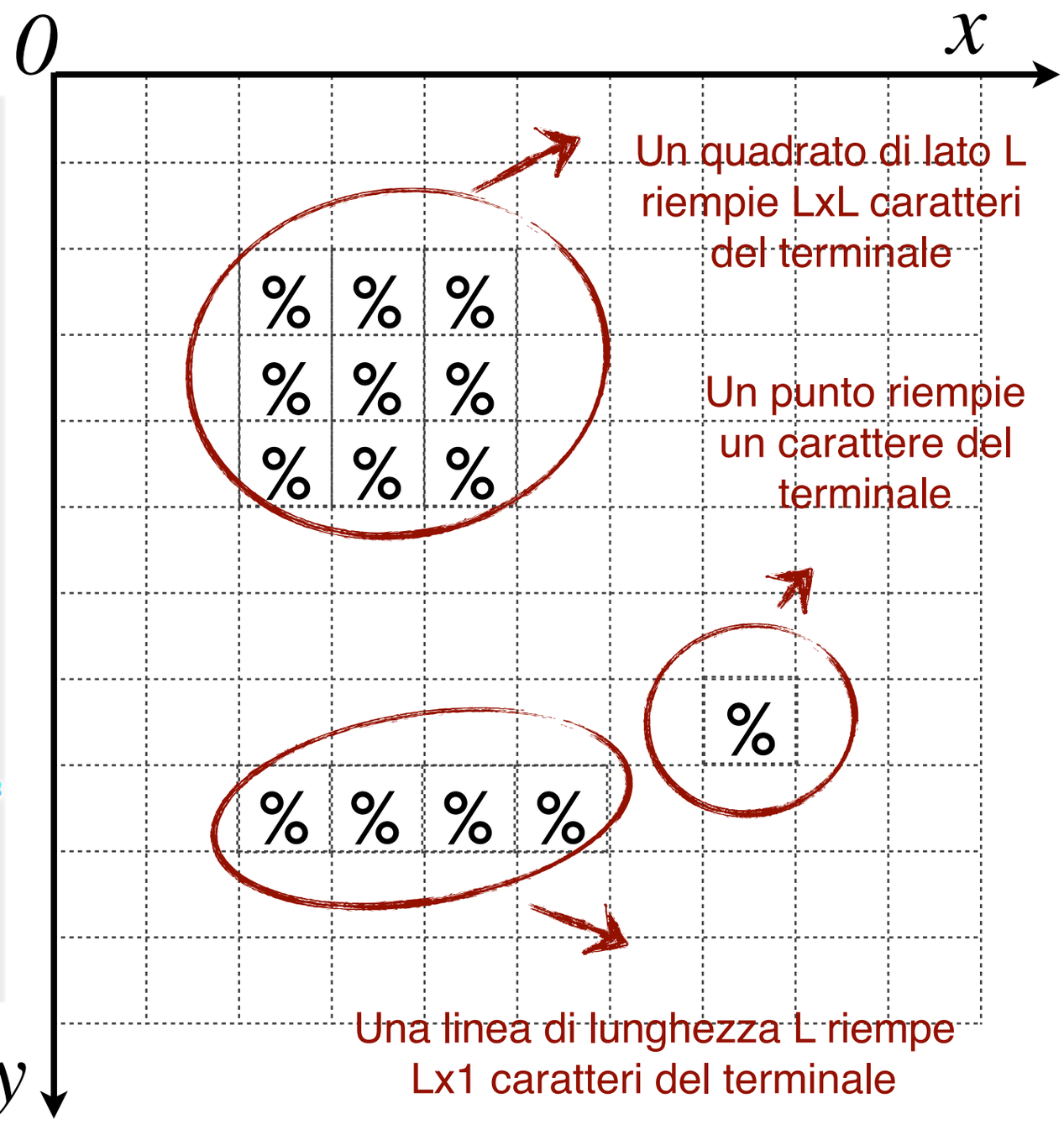
Restituisce una forma punto (linea di lunghezza 1)

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();


}
```


That's all Folks!

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it