Informatica B
2017-2018

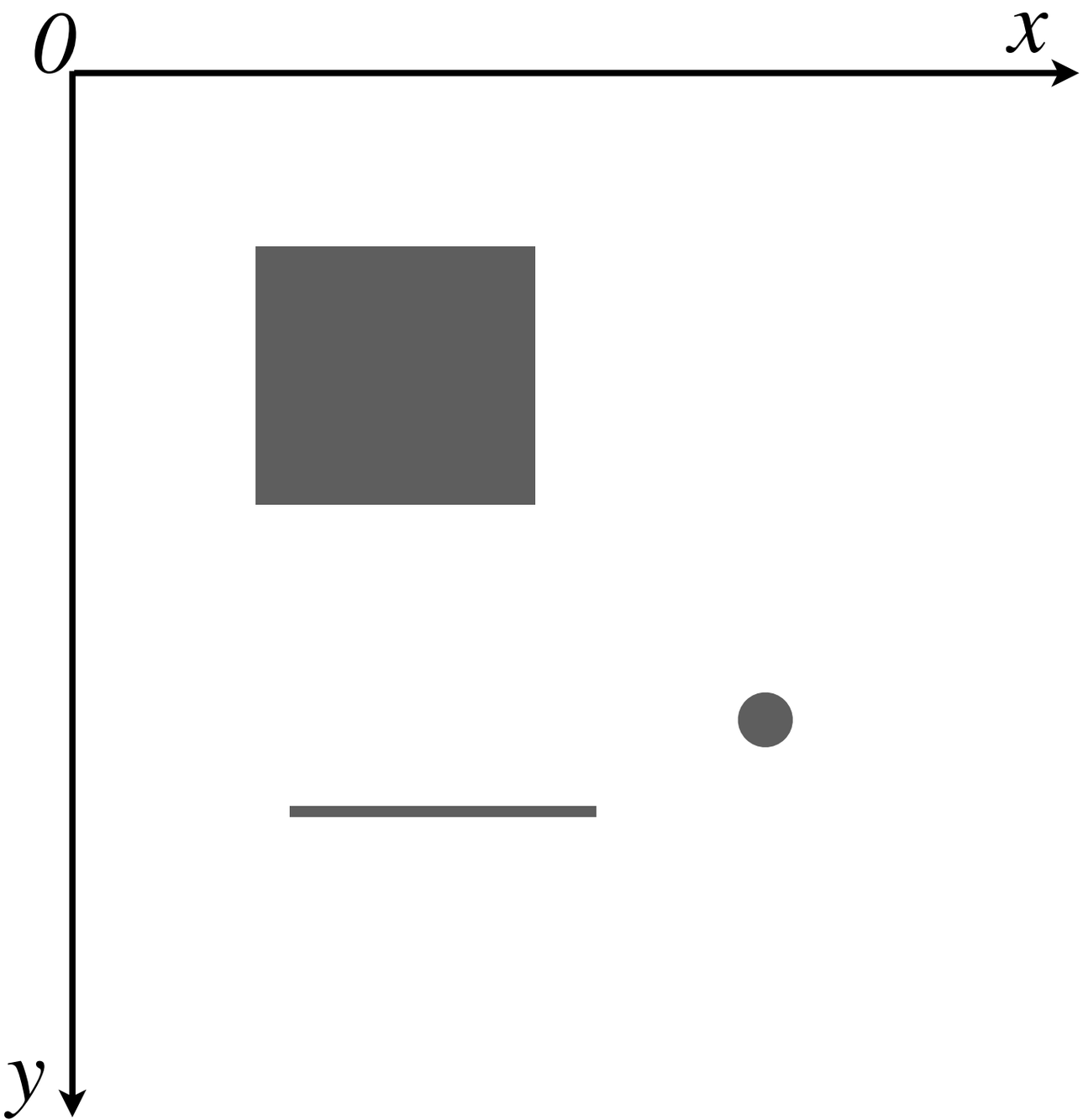
Esercitazione 7

Ripasso sul C - Matrici e Struct in un caso "reale"
Il piano cartesiano

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

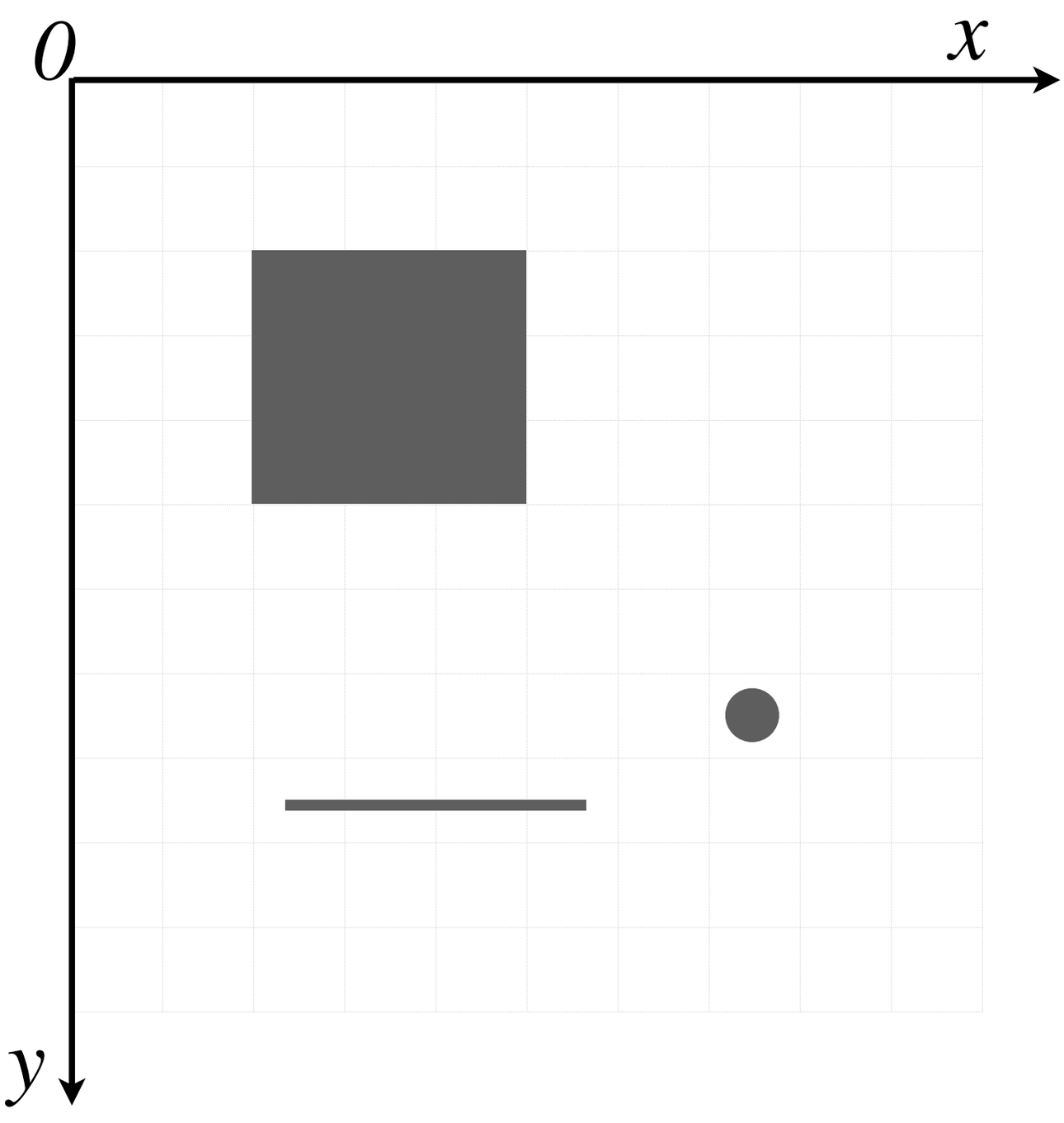
Specifiche dell'esercizio

- 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..



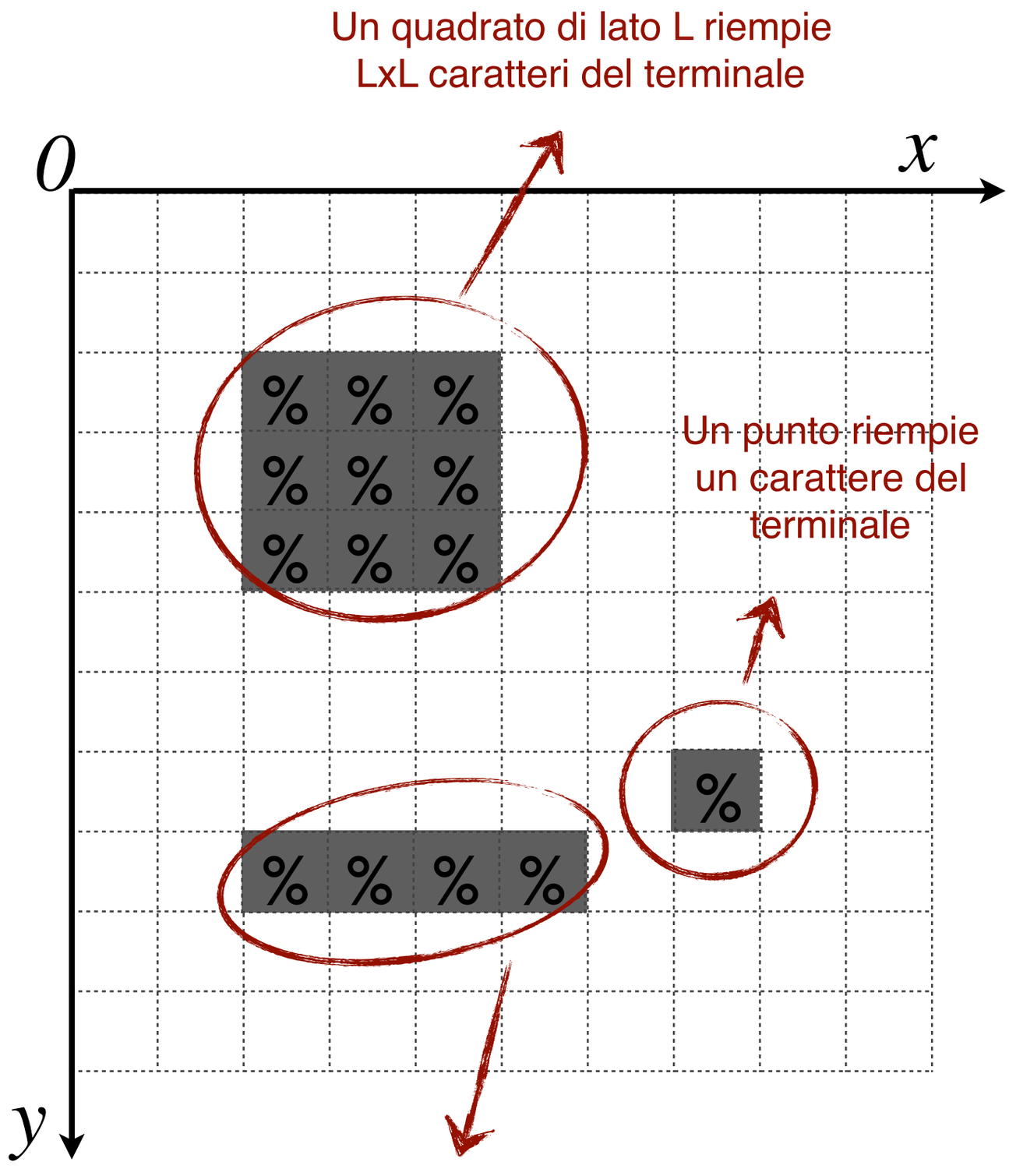
La rappresentazione

- 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



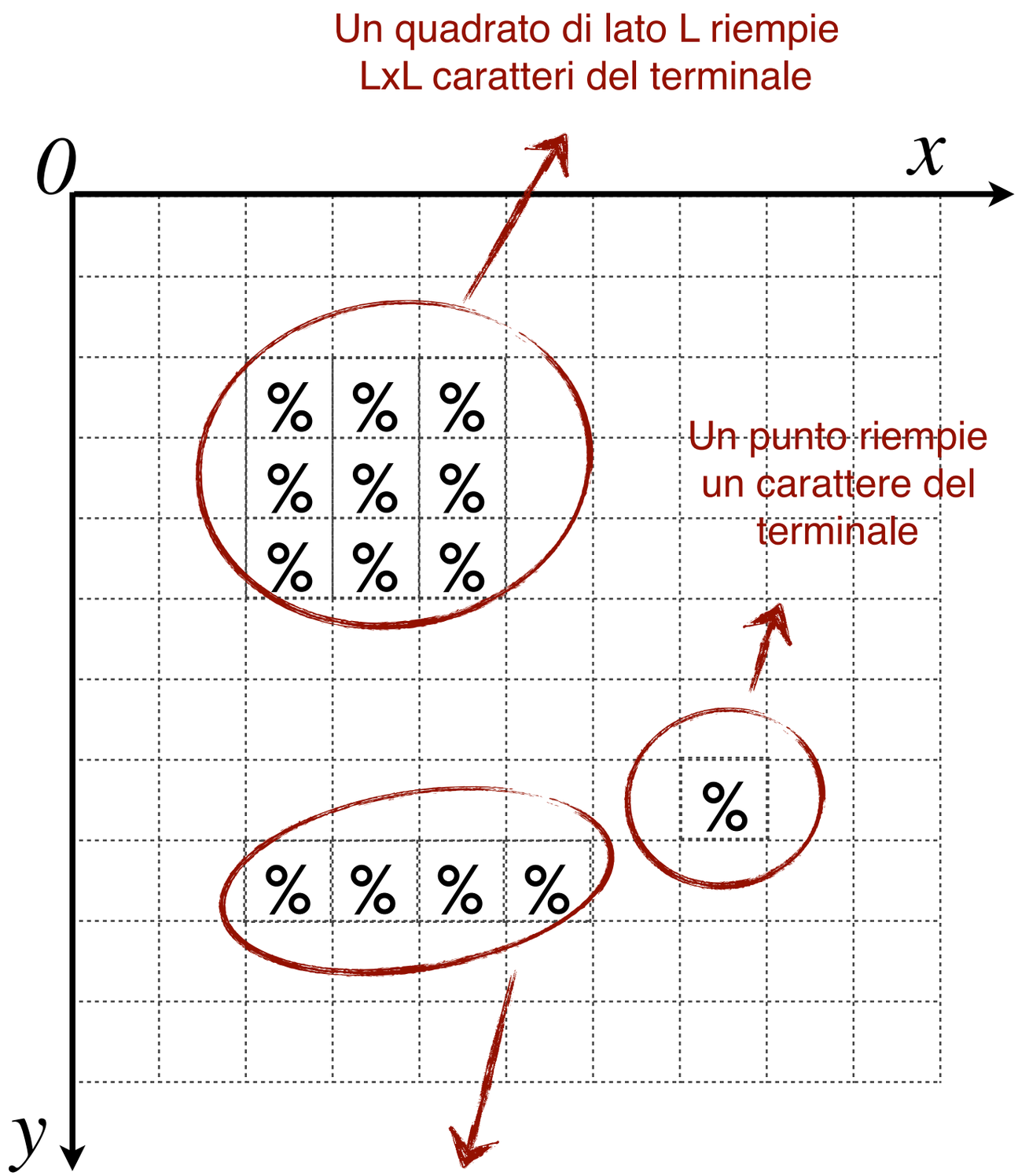
La rappresentazione

- 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



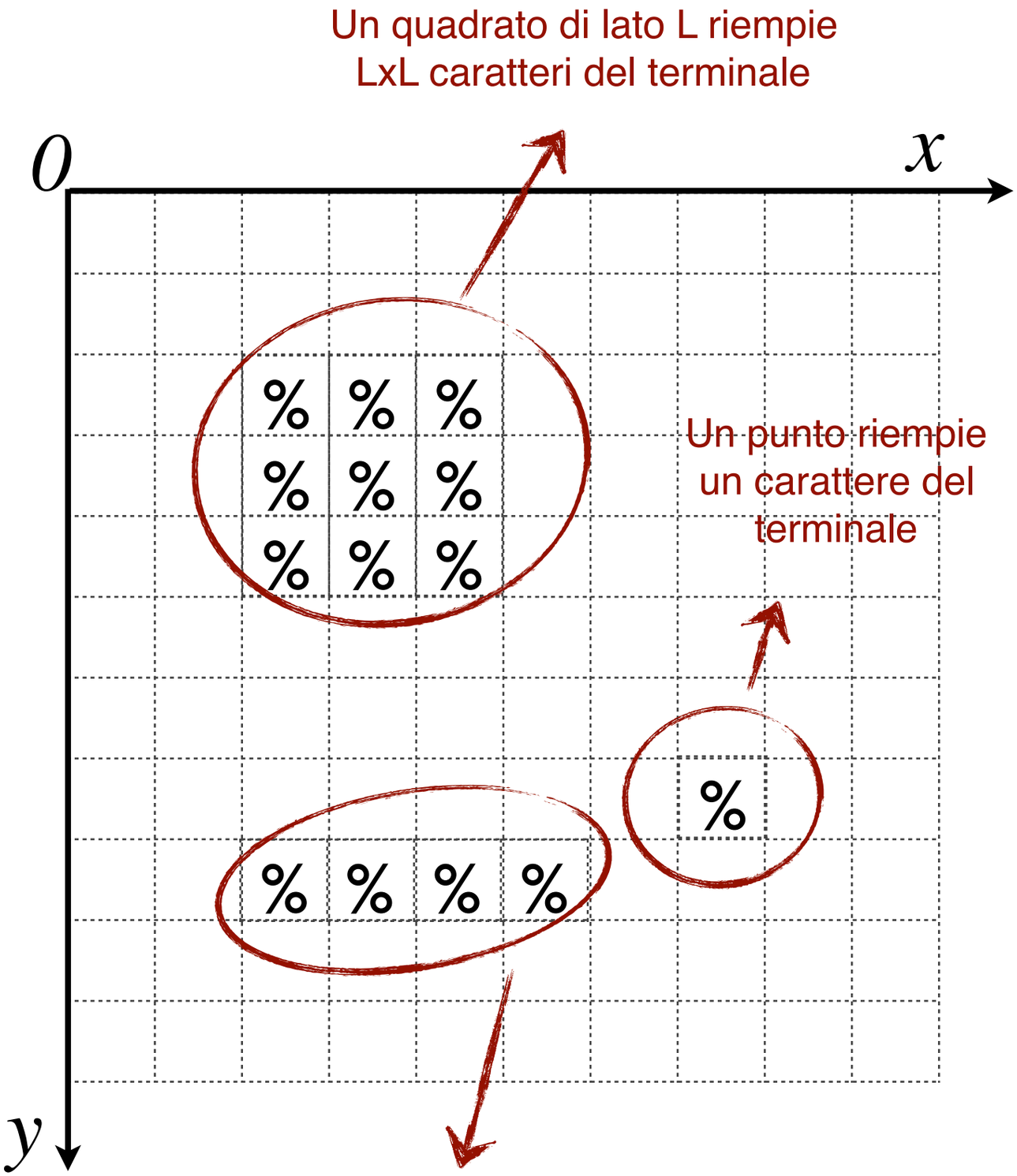
Cosa ci serve?

- 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;
```

Inizializza schermo

Scrivere una porzione di codice che inizializzi lo schermo
char schermo[SCREEN_W][SCREEN_H]

Riempie di caratteri vuoti la matrice in ingresso 'schermo'

```
int x,y;

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

Inserisci bordi

Scrivere una porzione di codice che metta i bordi allo schermo
char schermo[SCREEN_W][SCREEN_H]

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

```
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

Scrivere una porzione che stampi a schermo la variable schermo
char schermo[SCREEN_W][SCREEN_H]

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

Disegna Forma

Scrivere una porzione che stampi scriva nello schermo una data forma f

```
forma f;
punto_schermo p;
char schermo[SCREEN_W][SCREEN_H];

//init all vars here
//……

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

Scrivere una porzione di codice che crei un quadrato dati in input i dati **int dim** e **char carattere**

```
int dim = 3;
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;
```

Genera linea

Scrivere una porzione di codice che crei una linea di dimensione dim, con direzione direzione_linea e con carattere di rappresentazione carattere.

```
int dim = 10;
direzione = D_VERTICALE;
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;
```

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it