IEIM 2015-2016

Esercitazione VI
"A cosa servono le funzioni, la ricorsione e l'enum"

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

Cosa facciamo oggi?

- A COSA SERVONO LE FUNZIONI
  - Esercizio sul piano cartesiano
- A COSA SERVE LA RICORSIONE
  - Esercizio sull'albero genealogico
- A COSA SERVE L'ENUM
  - Esercizio sul parco auto


READY...
SET...
GO

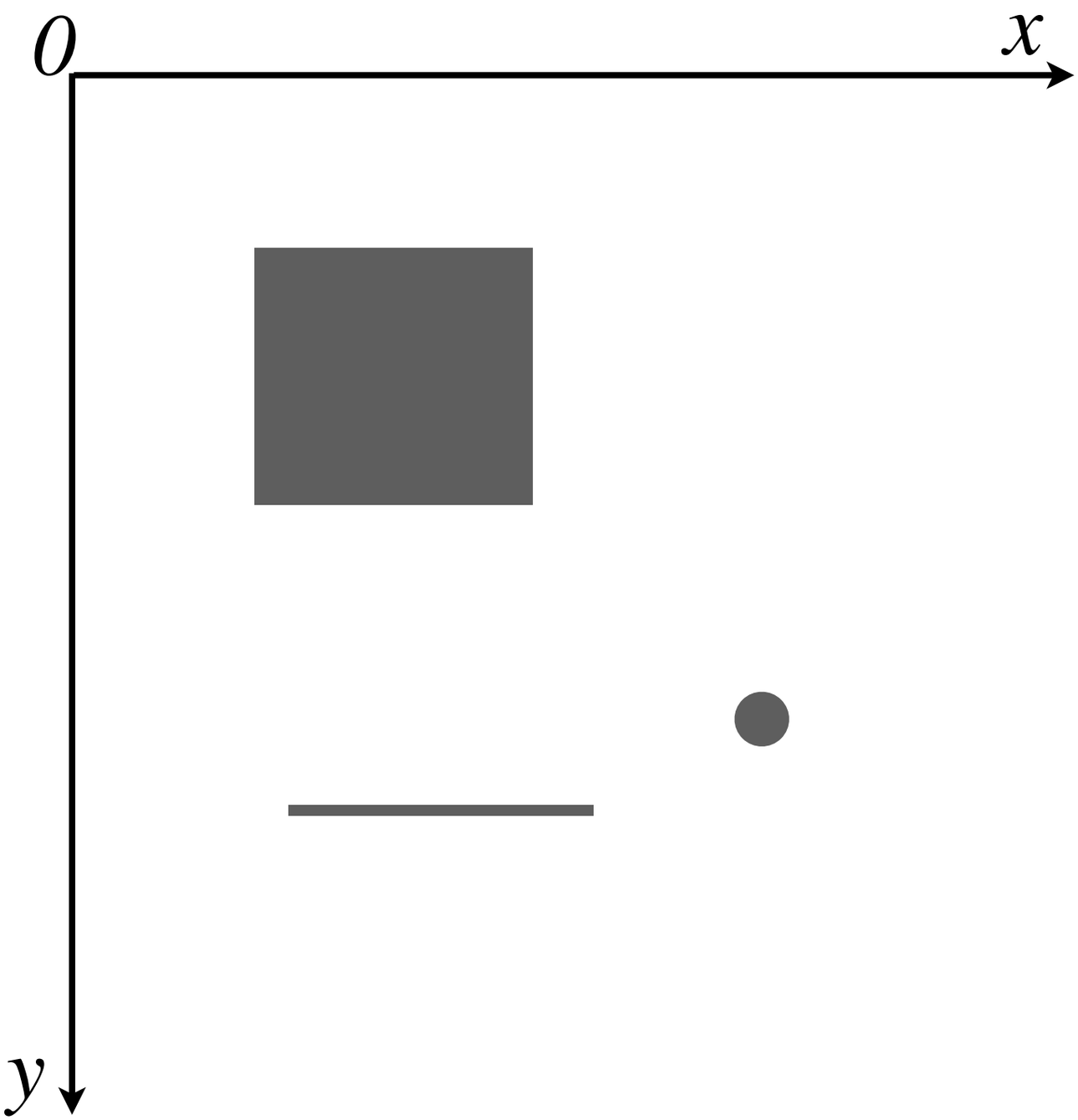
Il piano cartesiano

Esercizio 1

Specifiche dell'esercizio

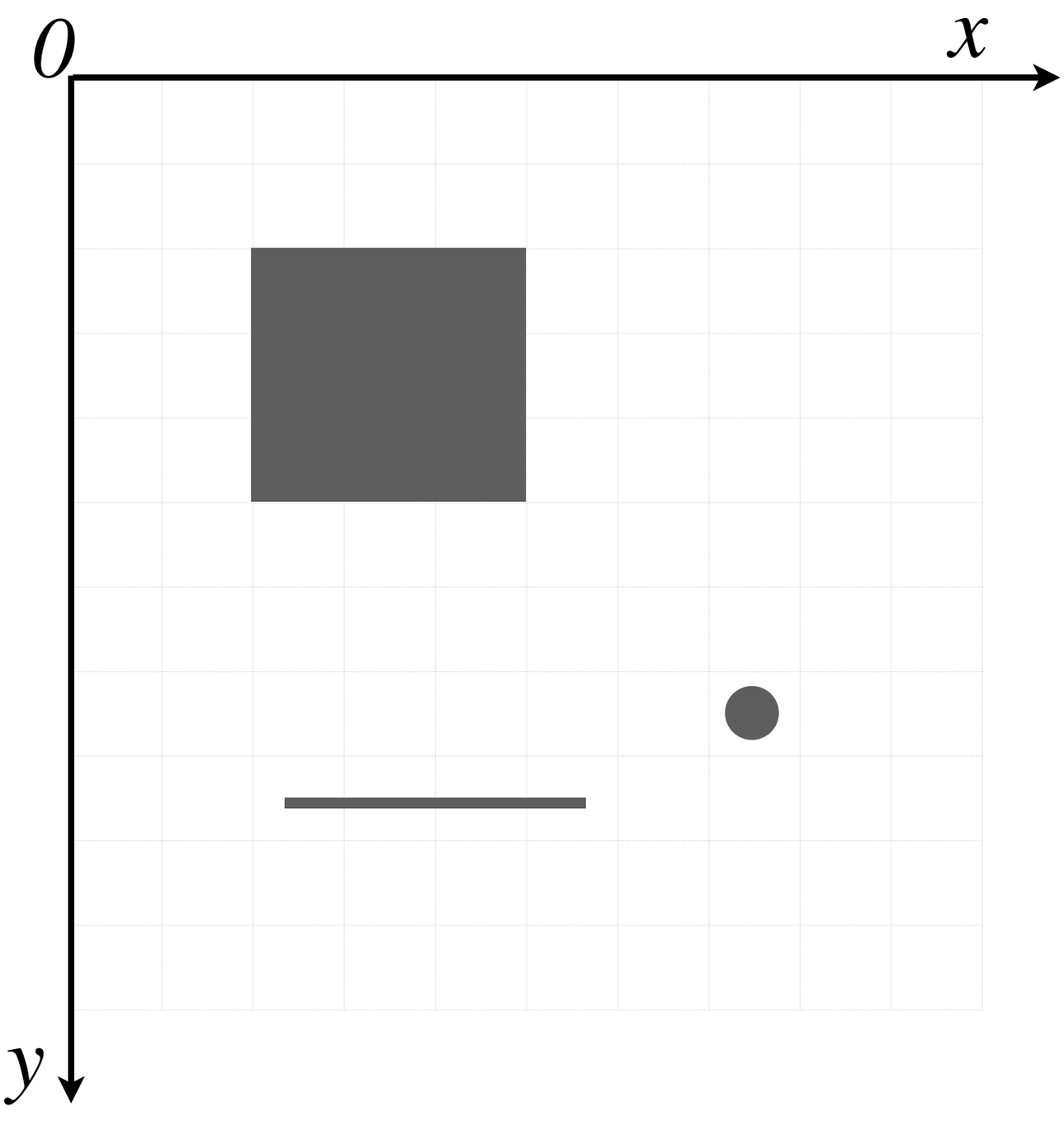
- Scrivere in C un programma che rappresenti un piano cartesiano
- Il programma deve poter rappresentare e visualizzare a schermo PUNTI, LINEE e QUADRATI
- Deve essere inoltre possibile manipolare le forme create (spostarle, cancellarle, ingrandirle, etc..)
- Il programma deve poter rappresentare la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

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



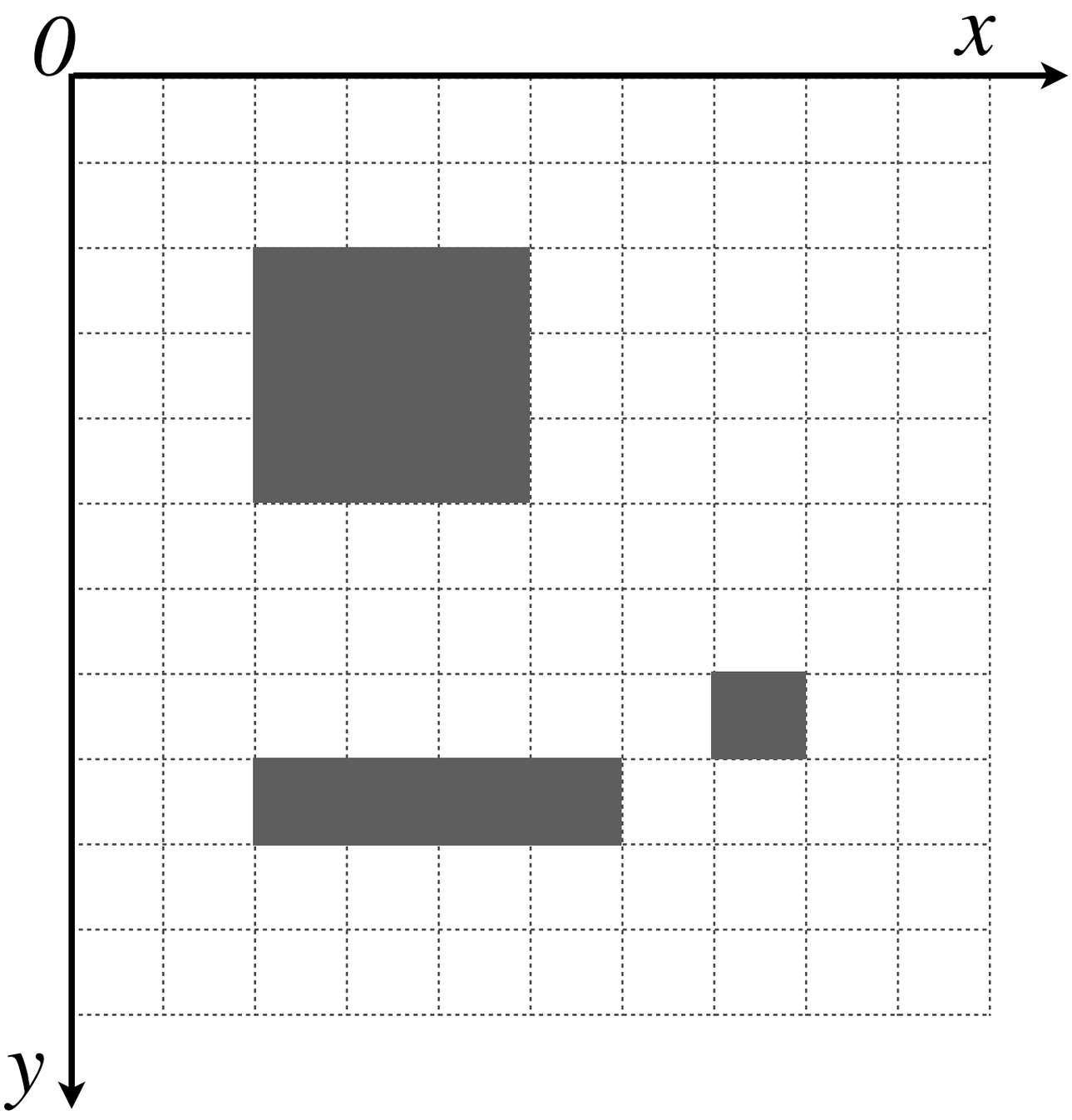
La rappresentazione

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



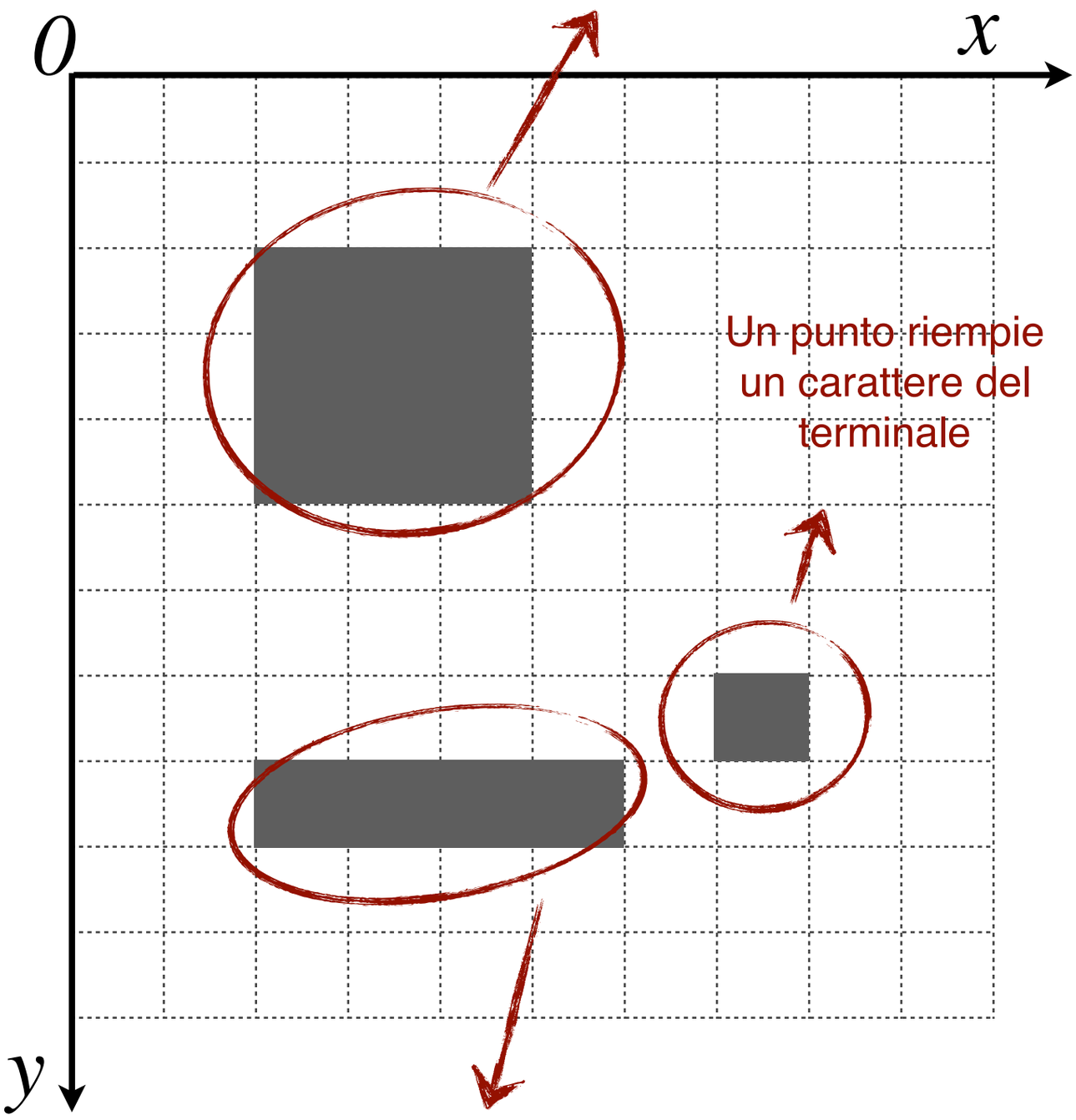
La rappresentazione

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



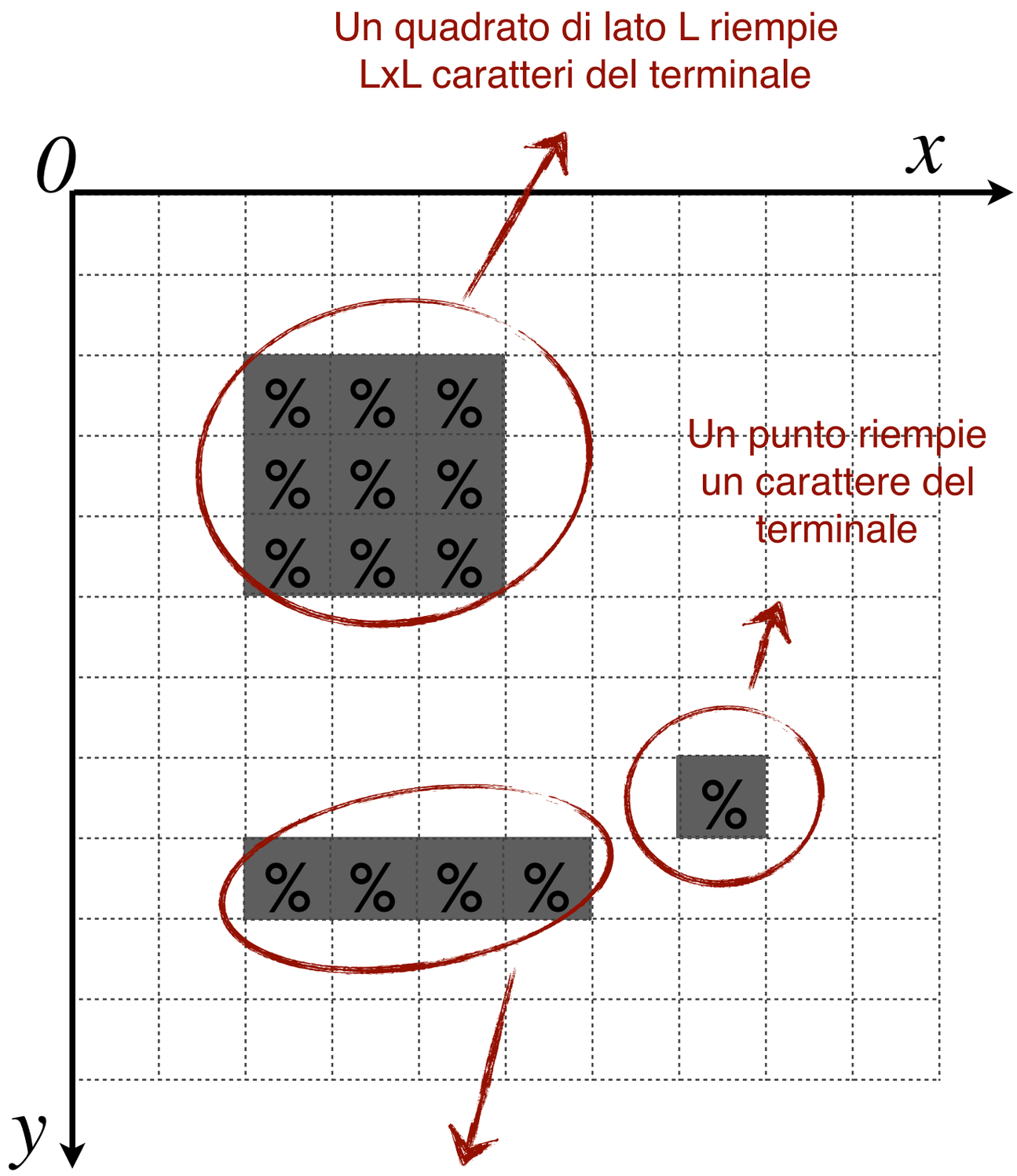
La rappresentazione

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



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



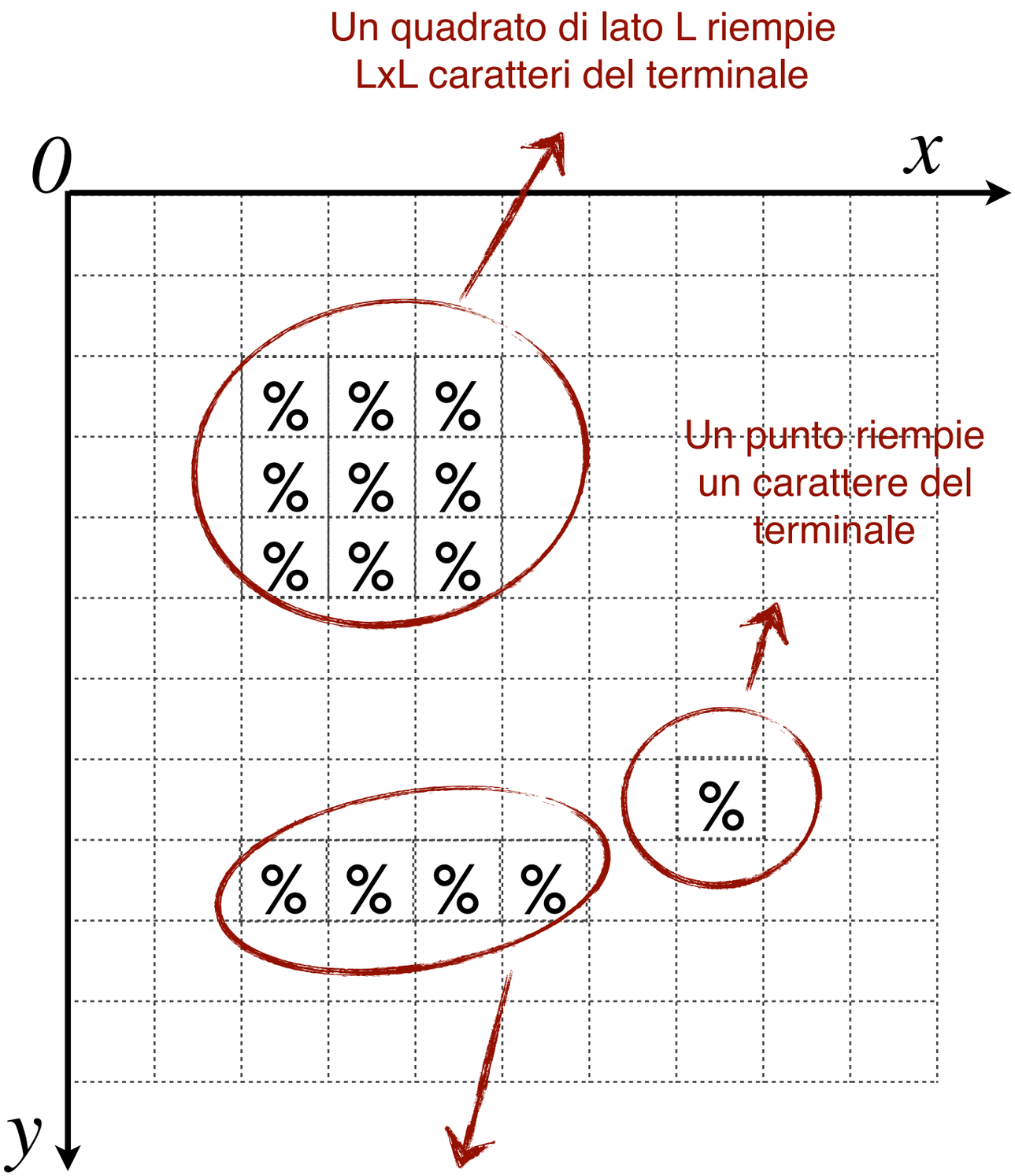
La rappresentazione

- 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

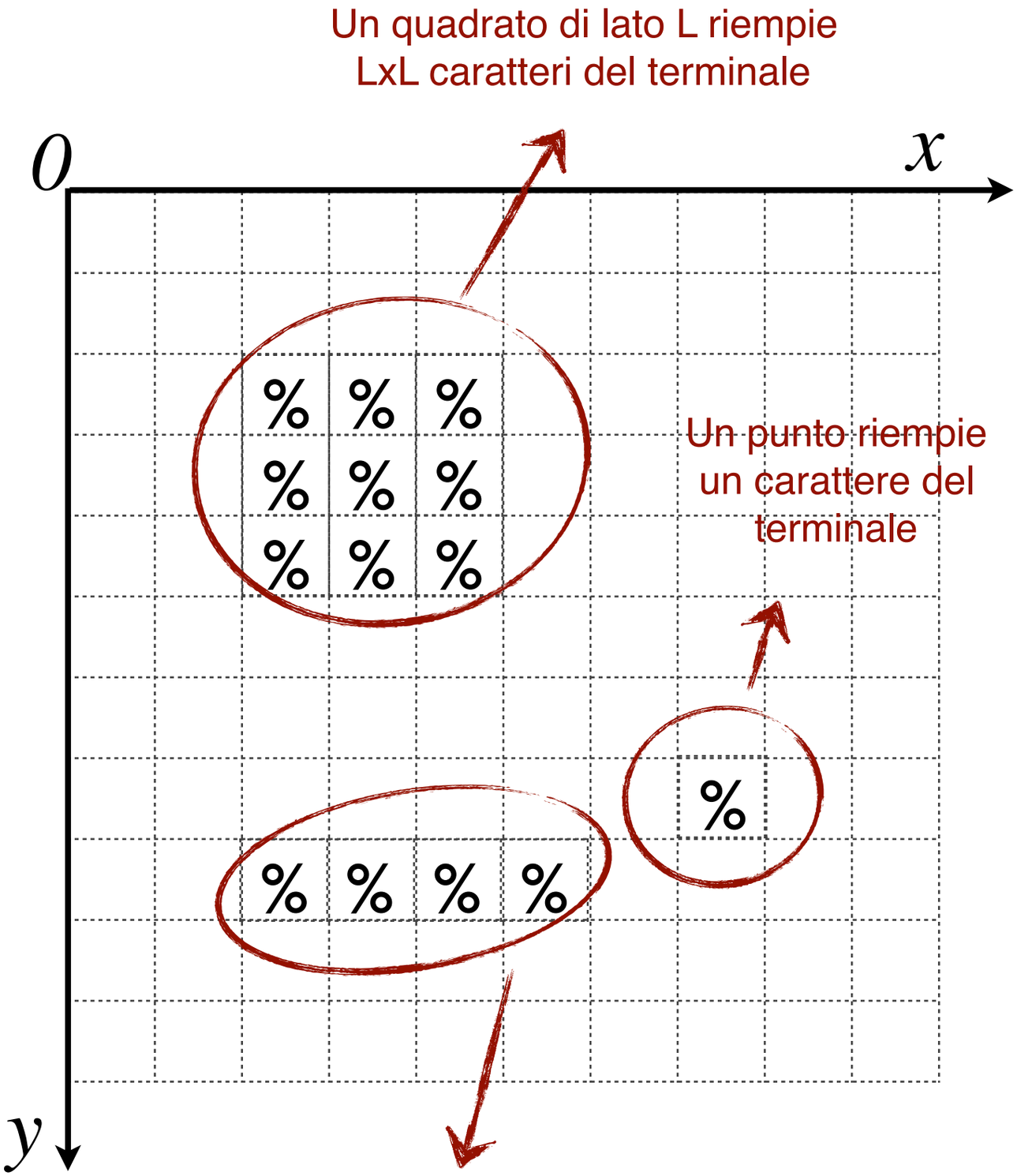


Cosa ci serve?

Cosa ci serve?

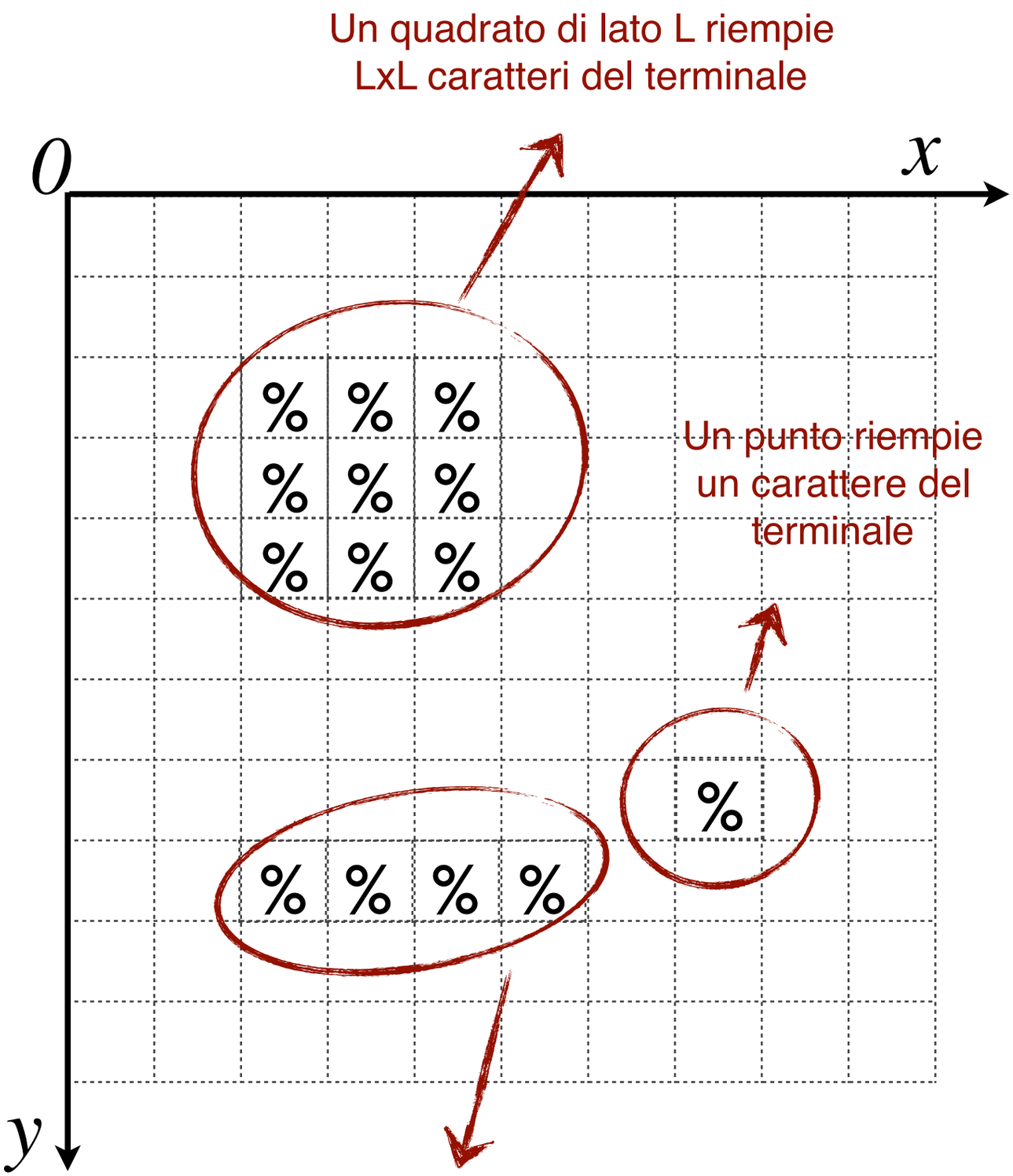
- Come sempre, prima di scrivere un programma, dobbiamo:
  1. definire i tipi di dato
  2. pensare di quali *costanti* avremo bisogno
  3. pensare di quali variabili avremo bisogno
  4. scegliere quali funzioni implementare
  5. implementare

Definizione dei tipi di dato



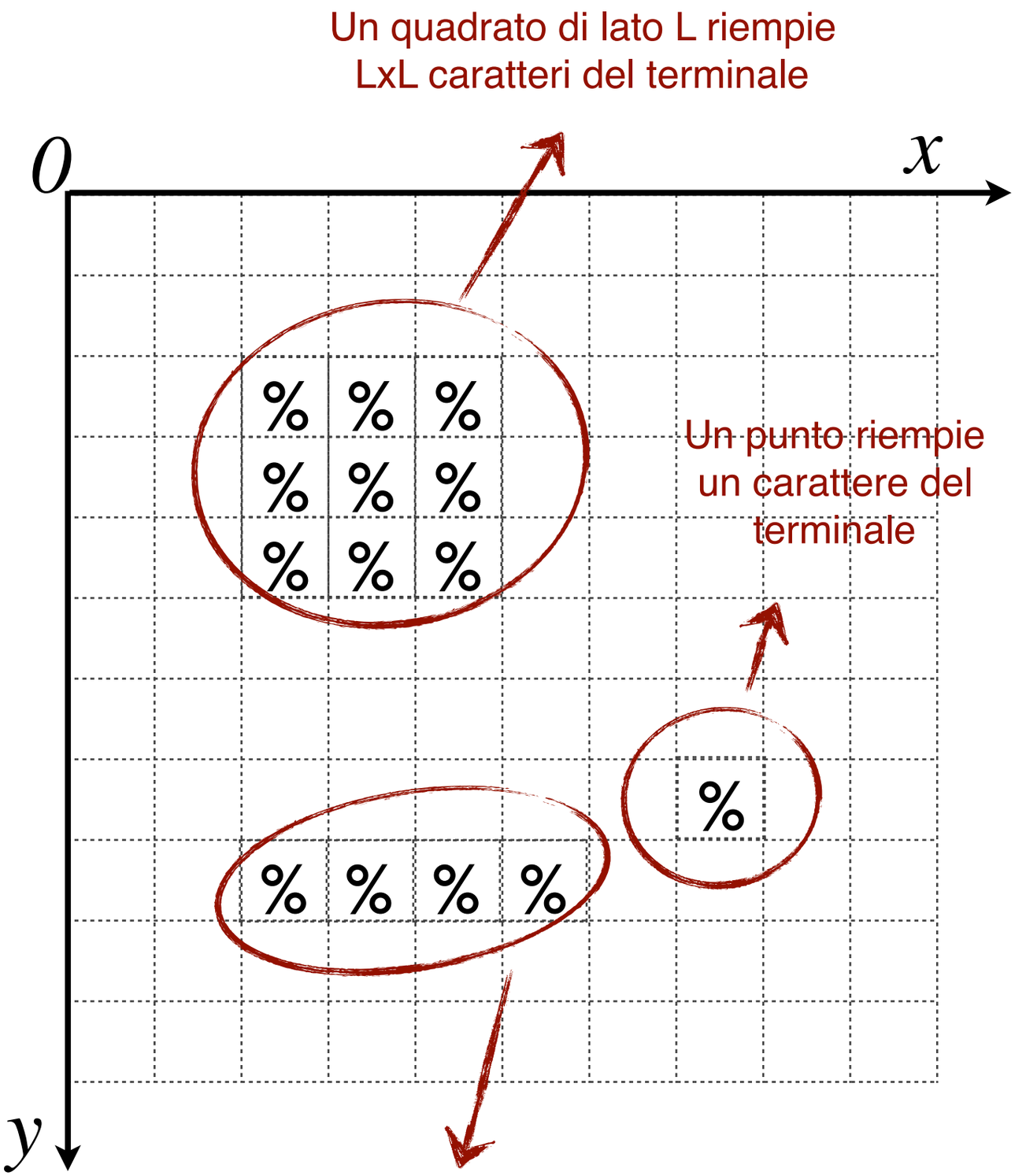
Definizione dei tipi di dato

- I tipi di dato che possono essere utili sono:



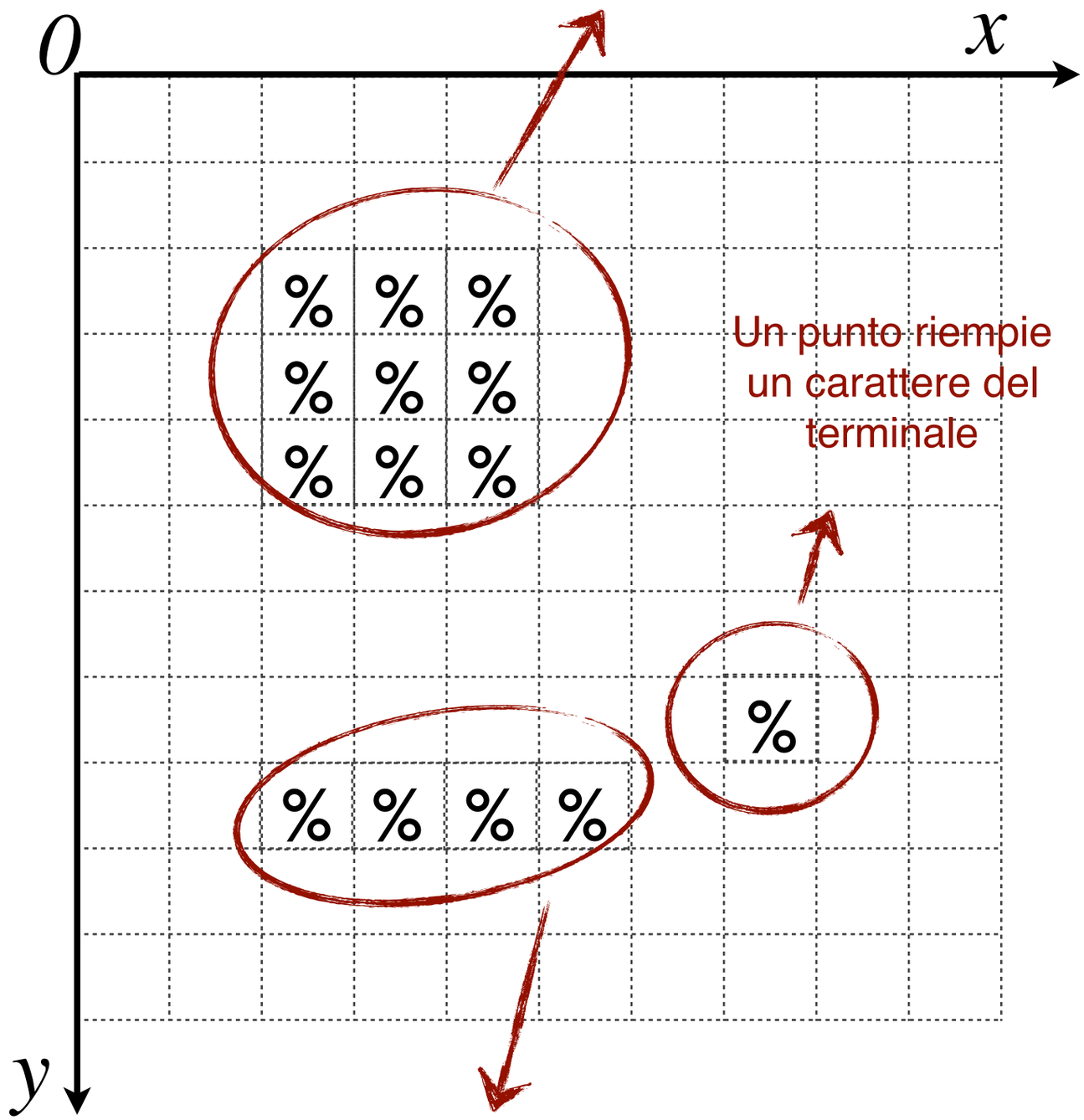
Definizione dei tipi di dato

- I tipi di dato che possono essere utili sono:
  - Punto dello schermo



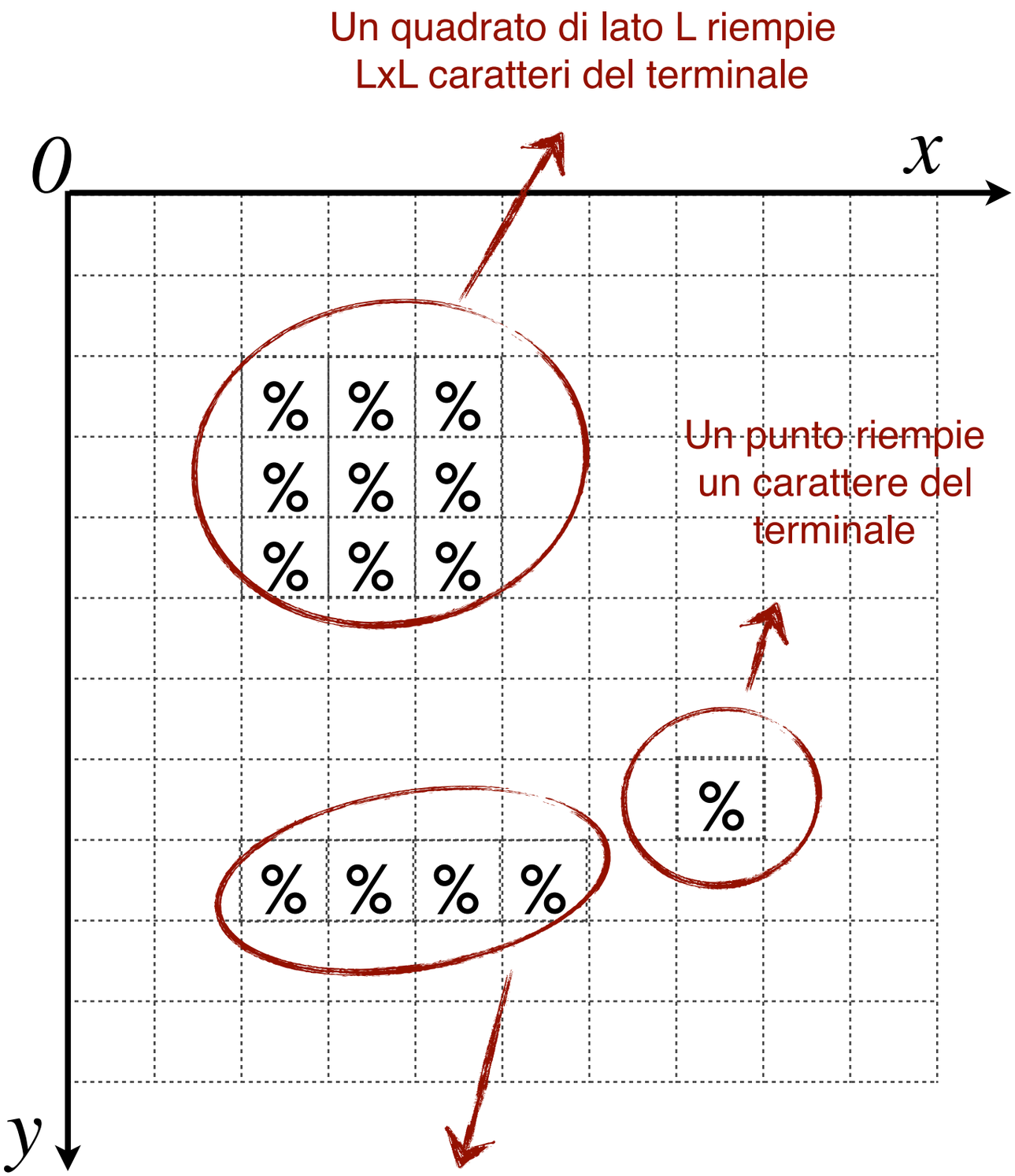
Definizione dei tipi di dato

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



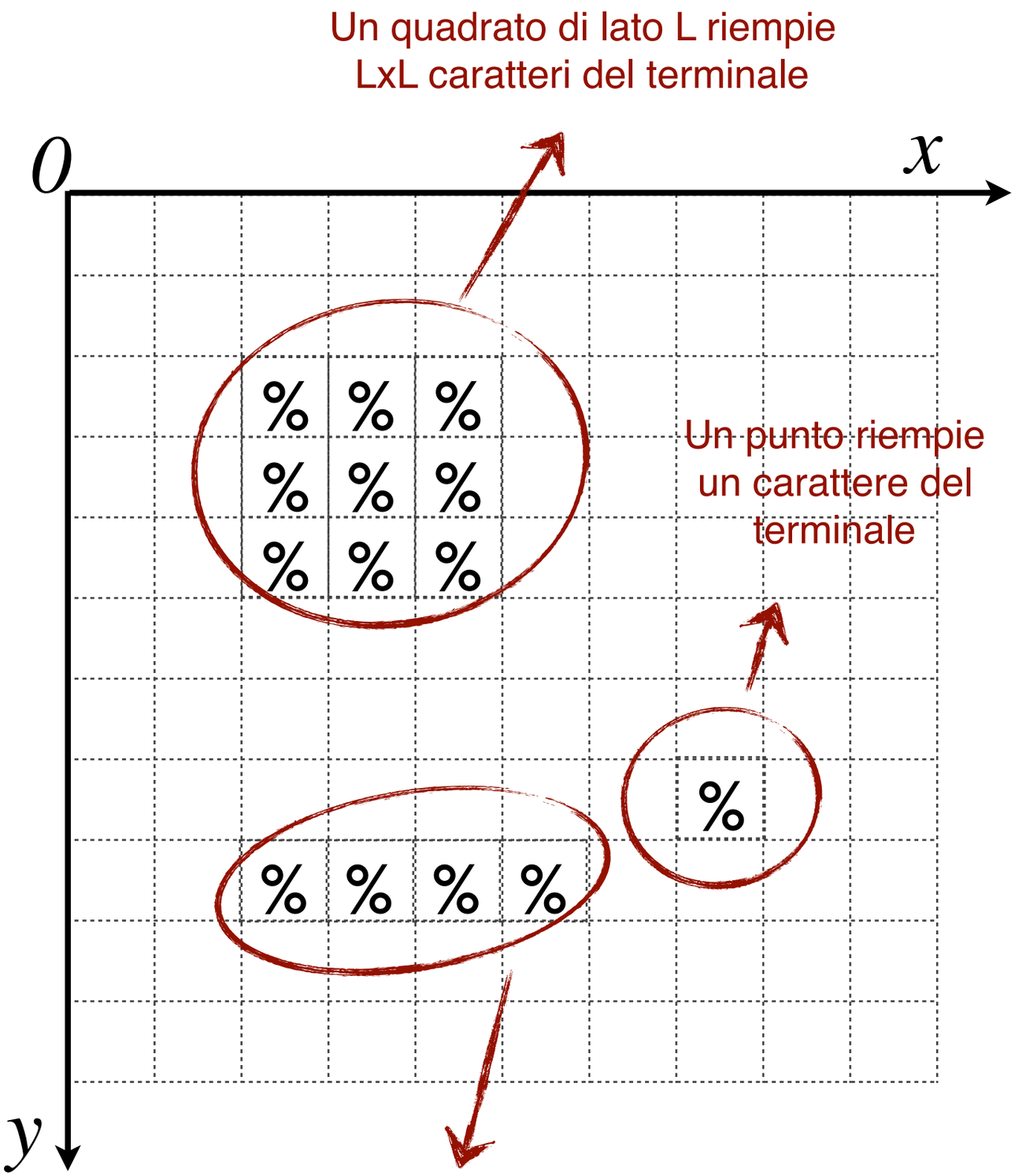
Definizione dei tipi di dato

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



Definizione dei tipi di dato

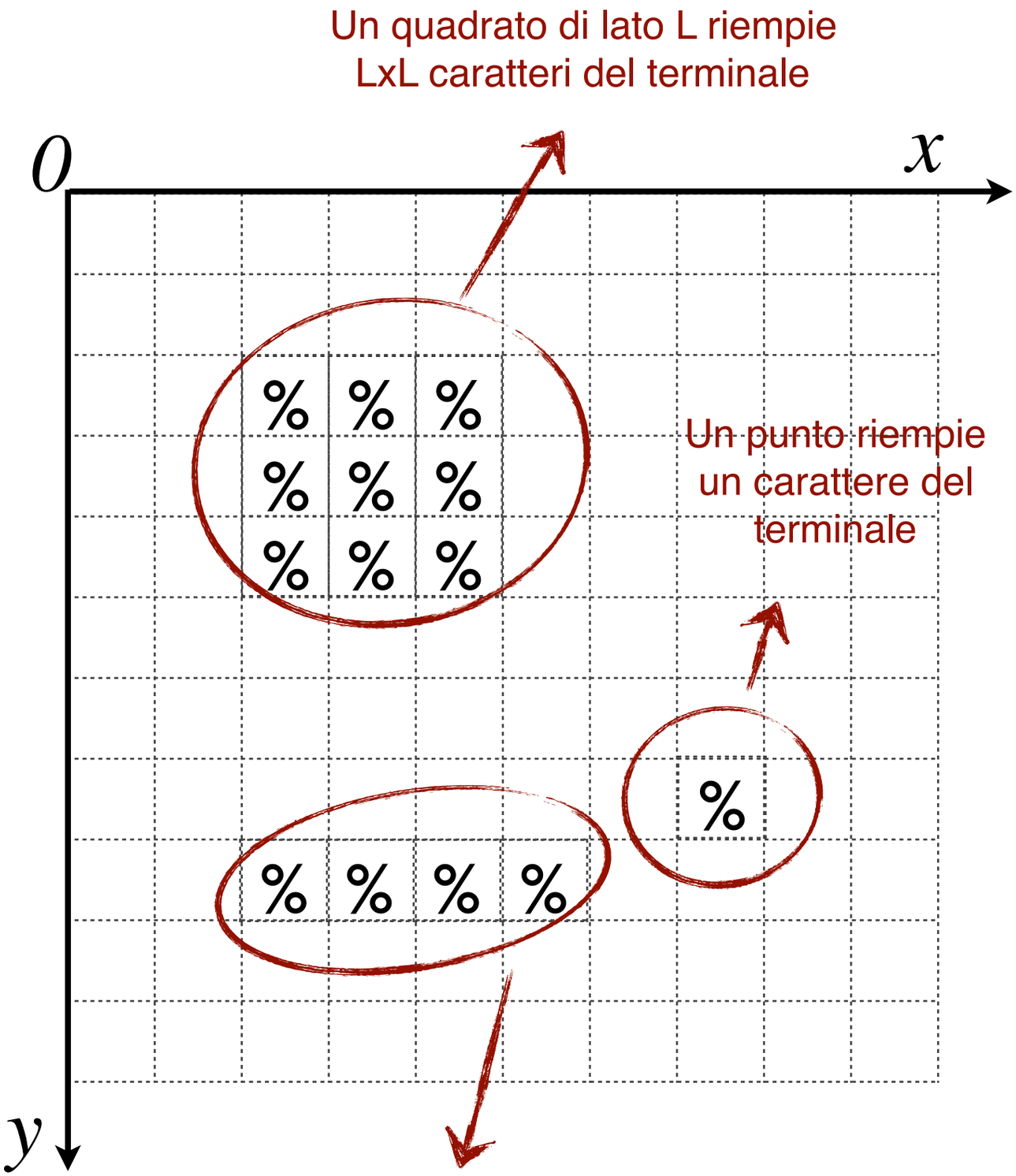
- I tipi di dato che possono essere utili sono:
  - Punto dello schermo
  - Una generica forma
  - Quali forme sono disponibili
- Direzioni



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
```

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
```

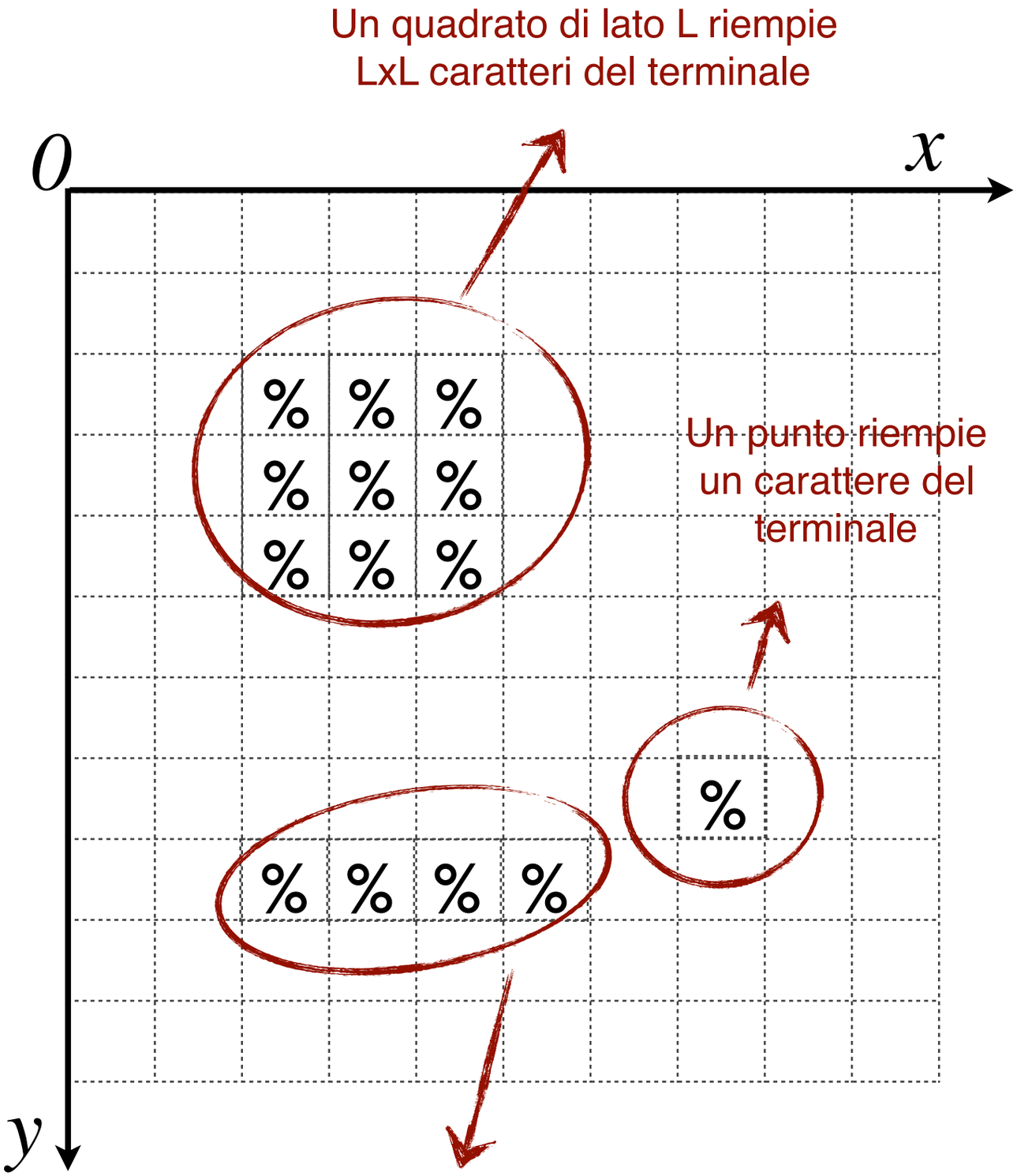
Possiamo usare le #define per definire valori costanti durante l'esecuzione del programma

E le variabili?

- Tendenzialmente, avremo bisogno di qualcosa tipo…

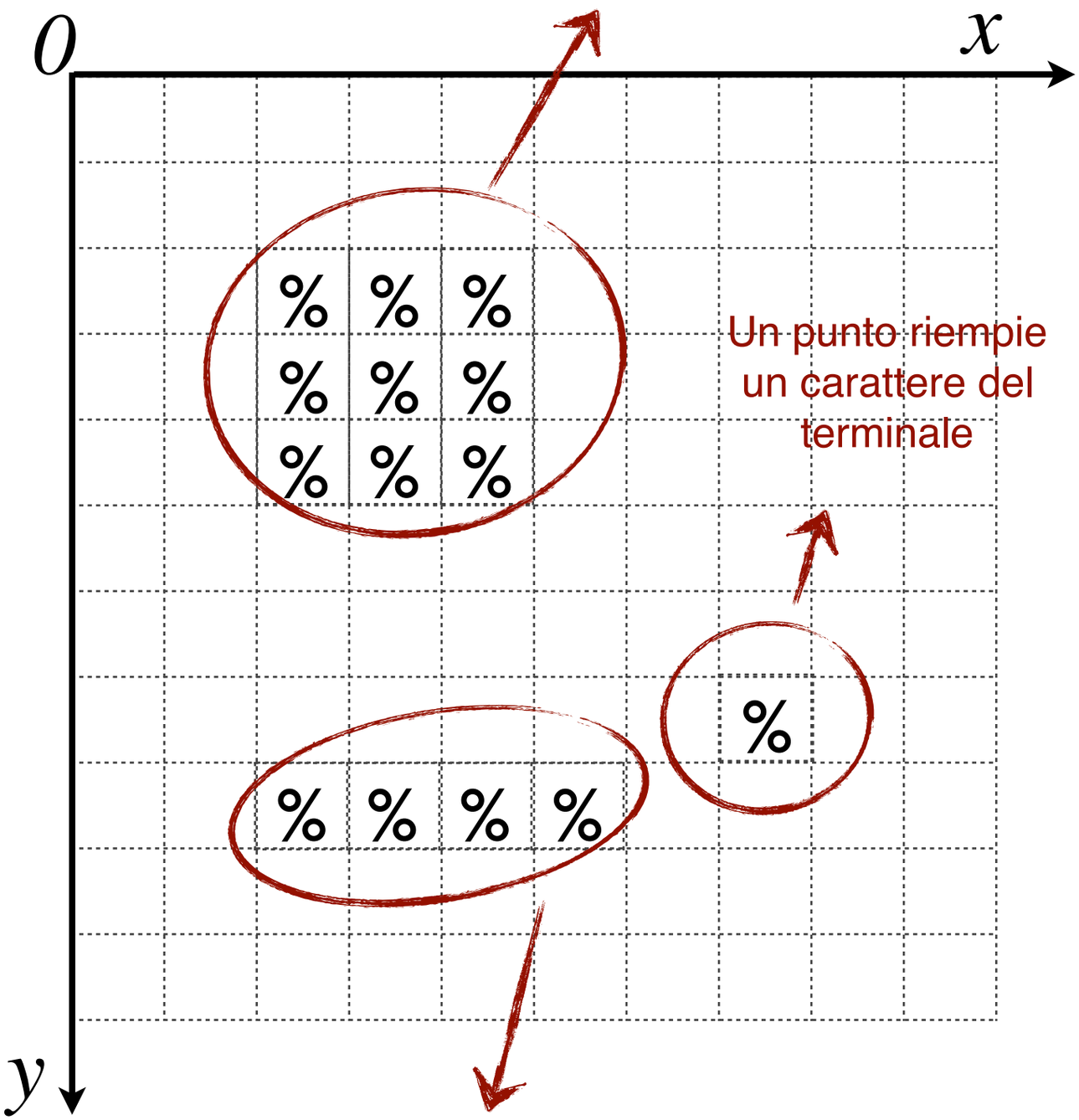
```
int main(){
char schermo[SCREEN_W][SCREEN_H];
forma quadrato;
punto_schermo p;
forma linea_or;
forma linea_vr;
forma punto;
```

Le funzioni


Un quadrato di lato L riempie
LxL caratteri del terminale
0
x
y
Un punto riempie
un carattere del
terminale
Una linea di lunghezza L riempie
Lx1 caratteri del terminale

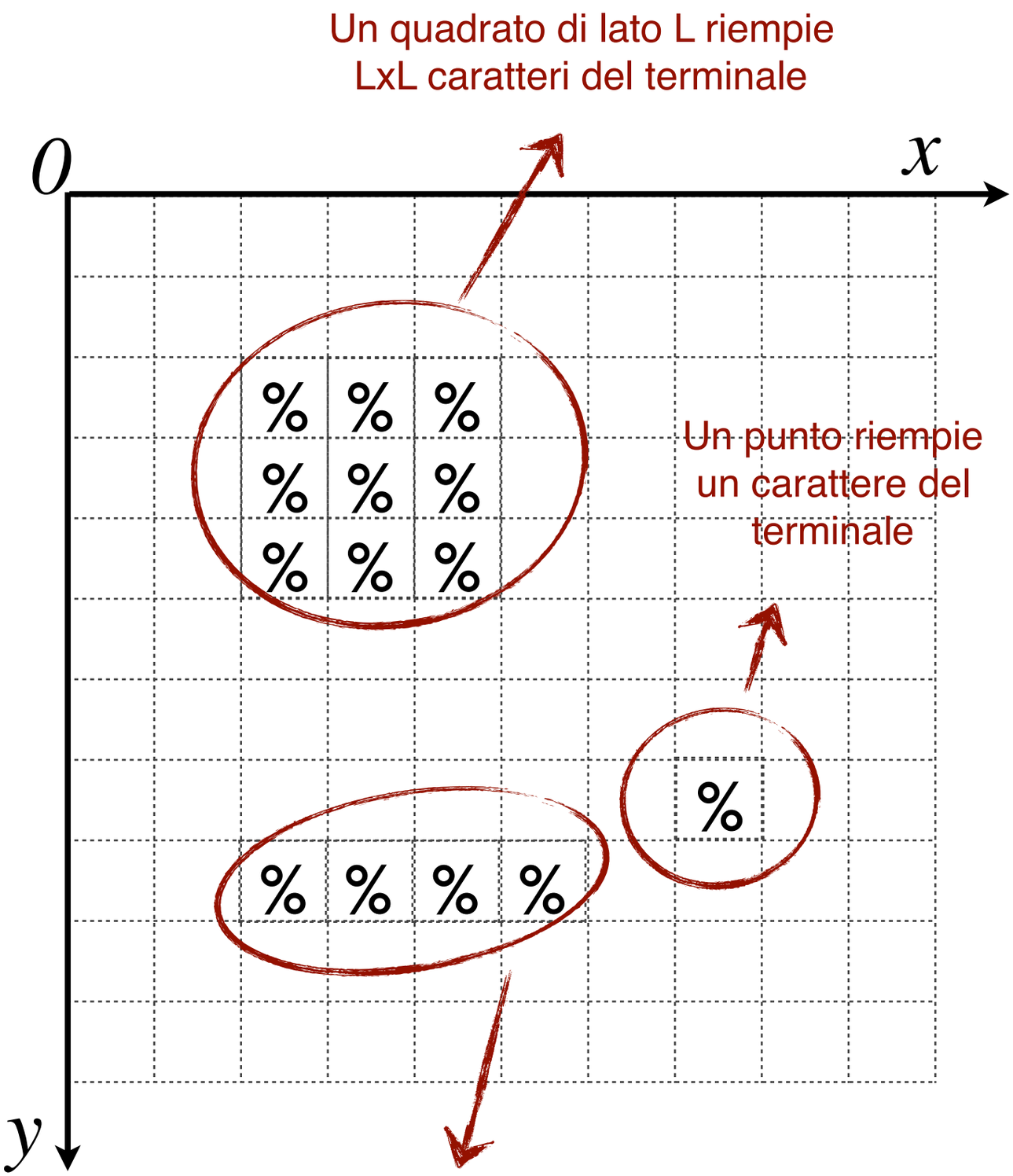
Le funzioni

- Le funzioni che possono essere utili sono:



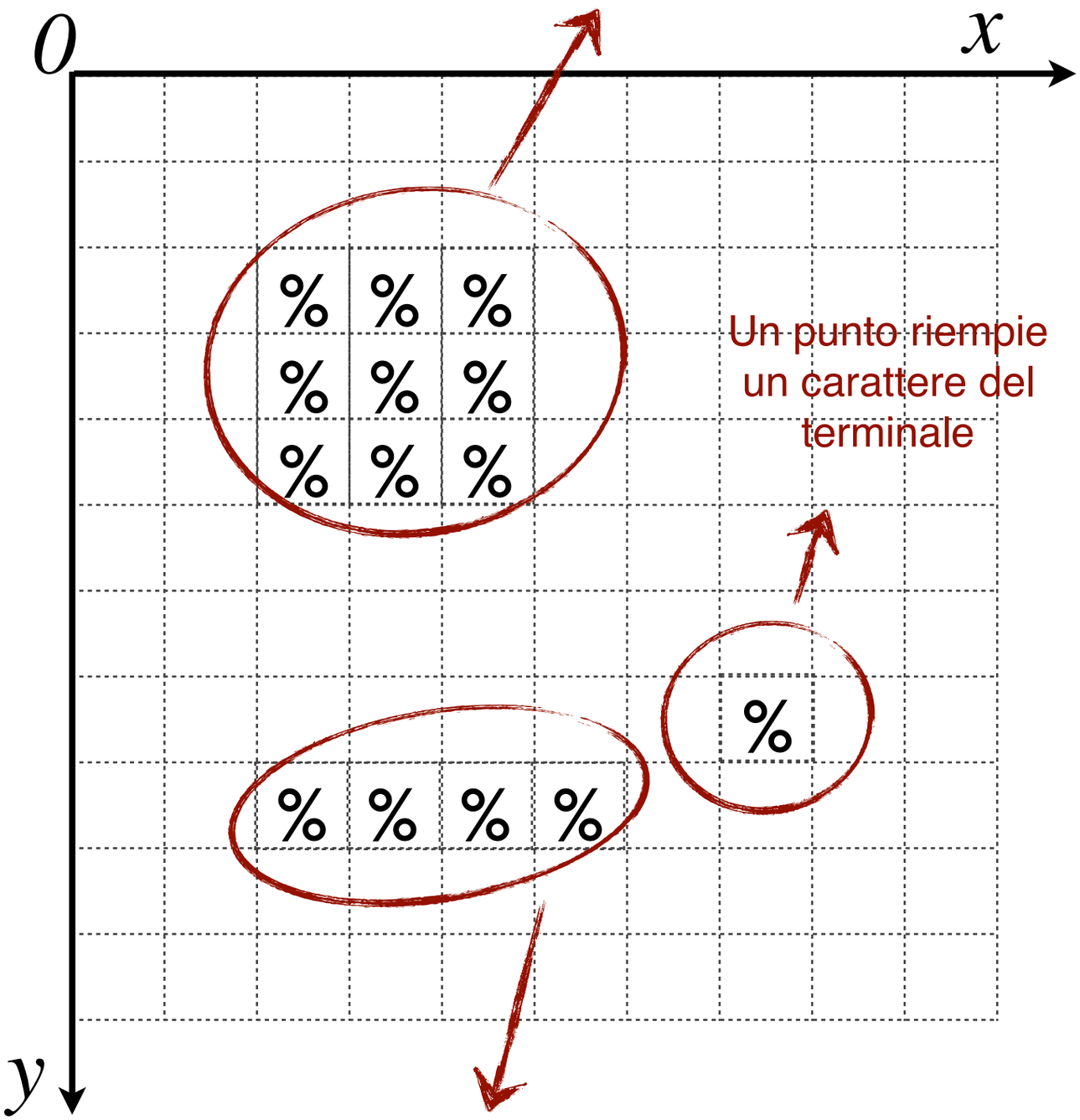
Le funzioni

- Le funzioni che possono essere utili sono:
  - Funzioni per la visualizzazione a video



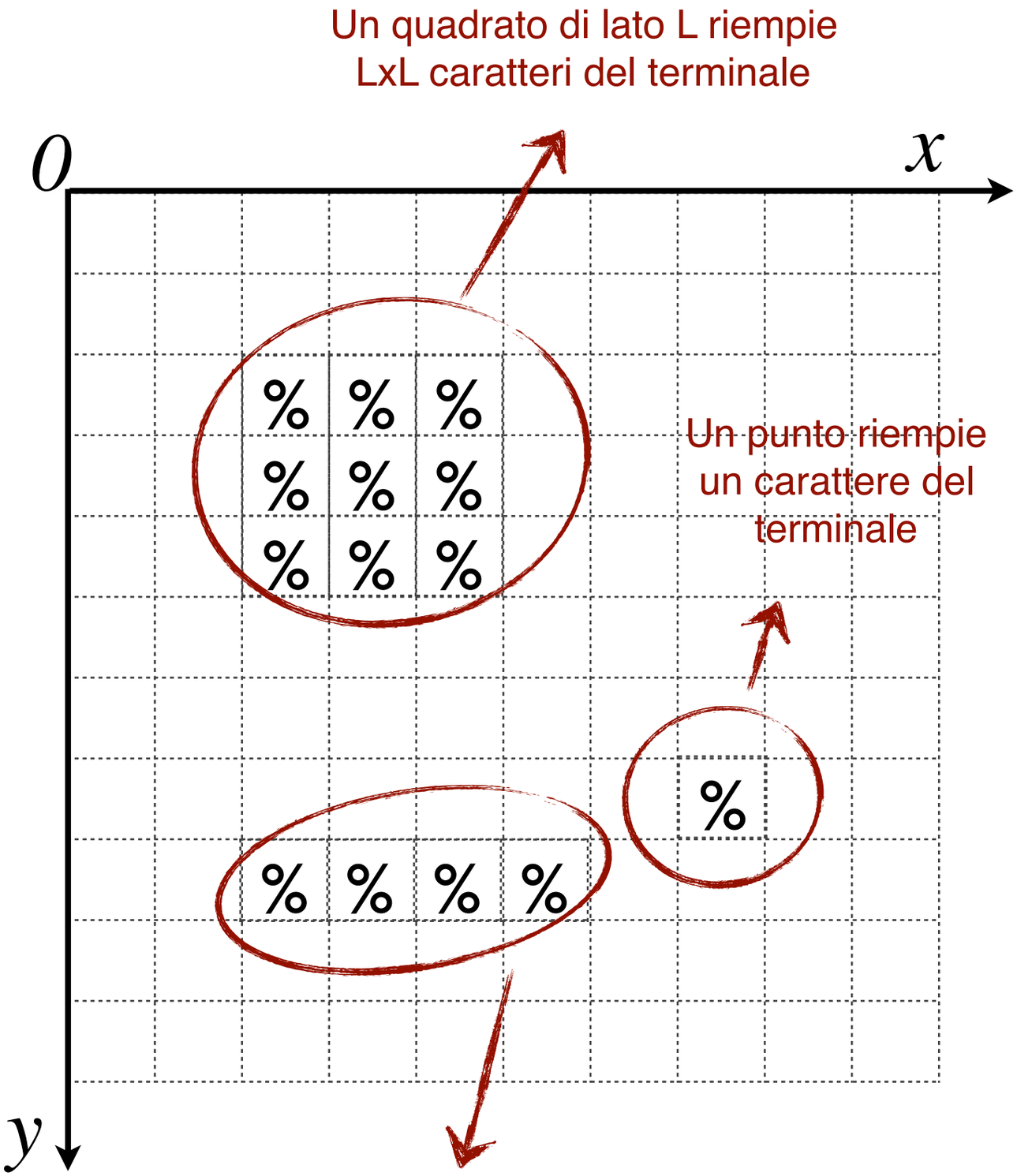
Le funzioni

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



Le funzioni

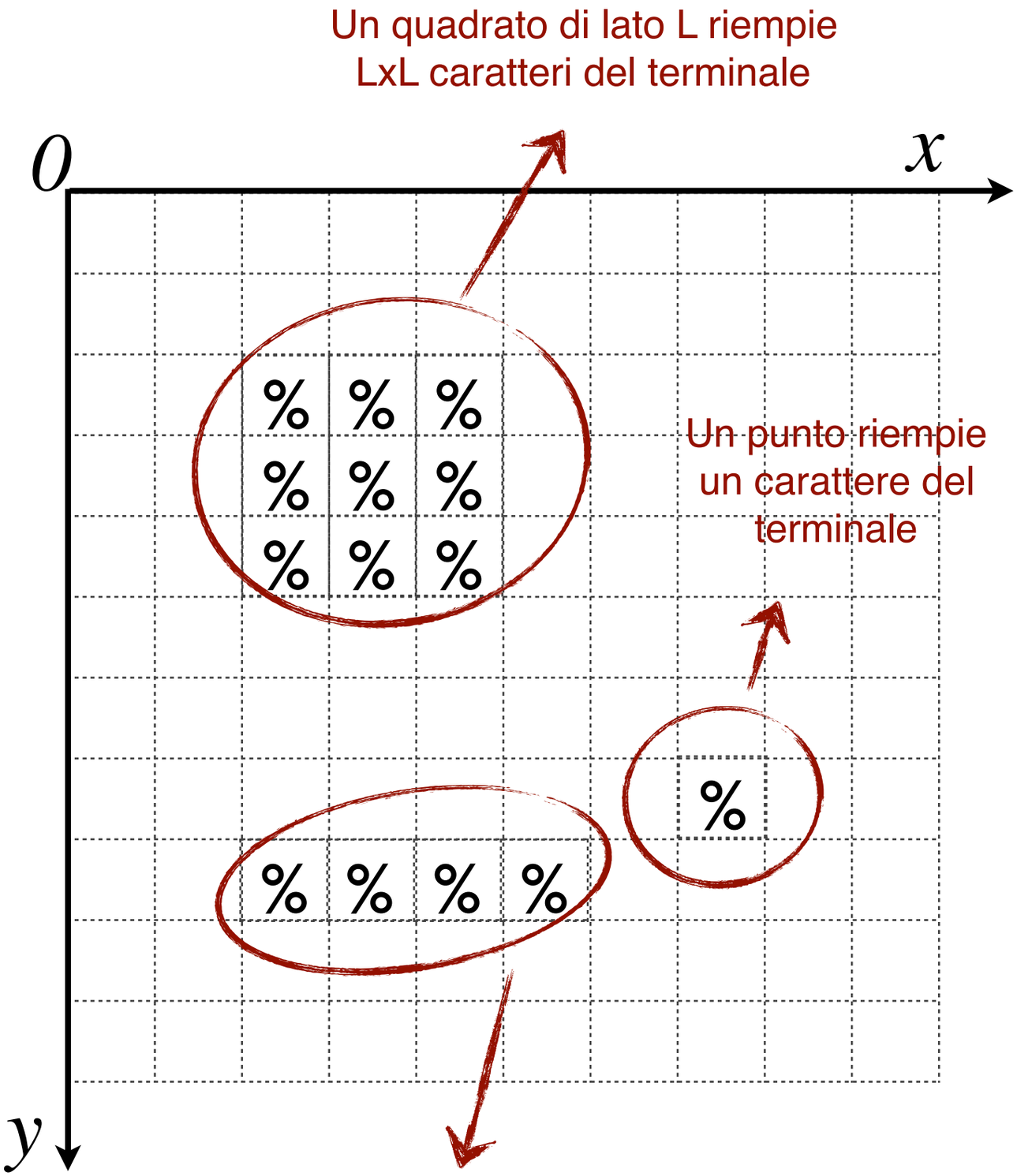
- Le funzioni che possono essere utili sono:
  - Funzioni per la visualizzazione a video
  - Modifica delle matrice che rappresenta lo schermo
  - Generazione delle forme



Le funzioni

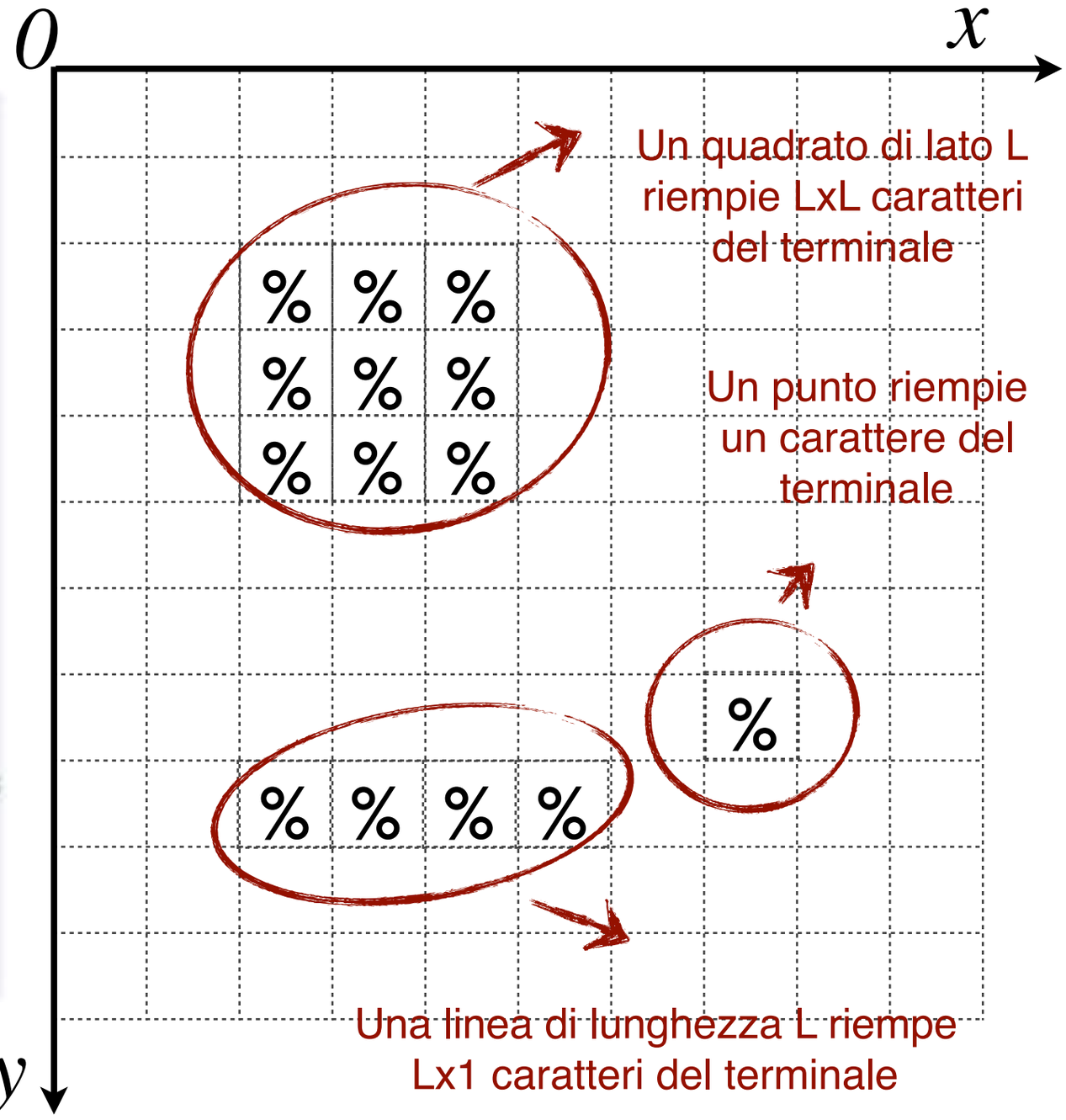
- Le funzioni che possono essere utili sono:
  - Funzioni per la visualizzazione a video
  - Modifica delle matrice che rappresenta lo schermo
  - Generazione delle forme

IMPLEMENTIAMO ALLA LAVAGNA!



Inizializza schermo

```
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();

// Funzioni sulle matrici che rappresentano 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);
```




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

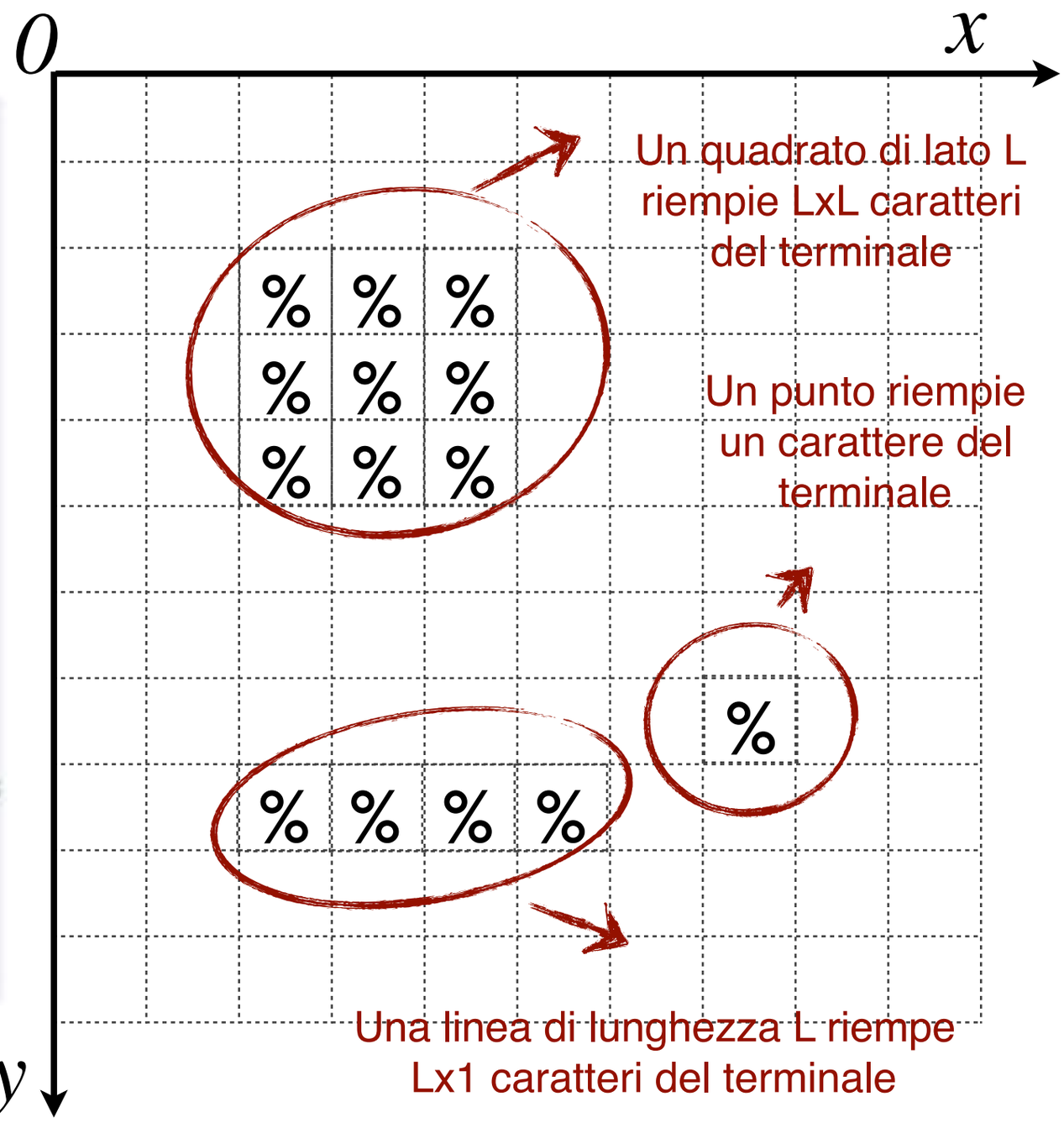
Riempie di caratteri vuoti la matrice in ingresso 'schermo'

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();

// Funzioni della matrice 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 inserisci_bordi(char schermo[SCREEN_W][SCREEN_H]);
```

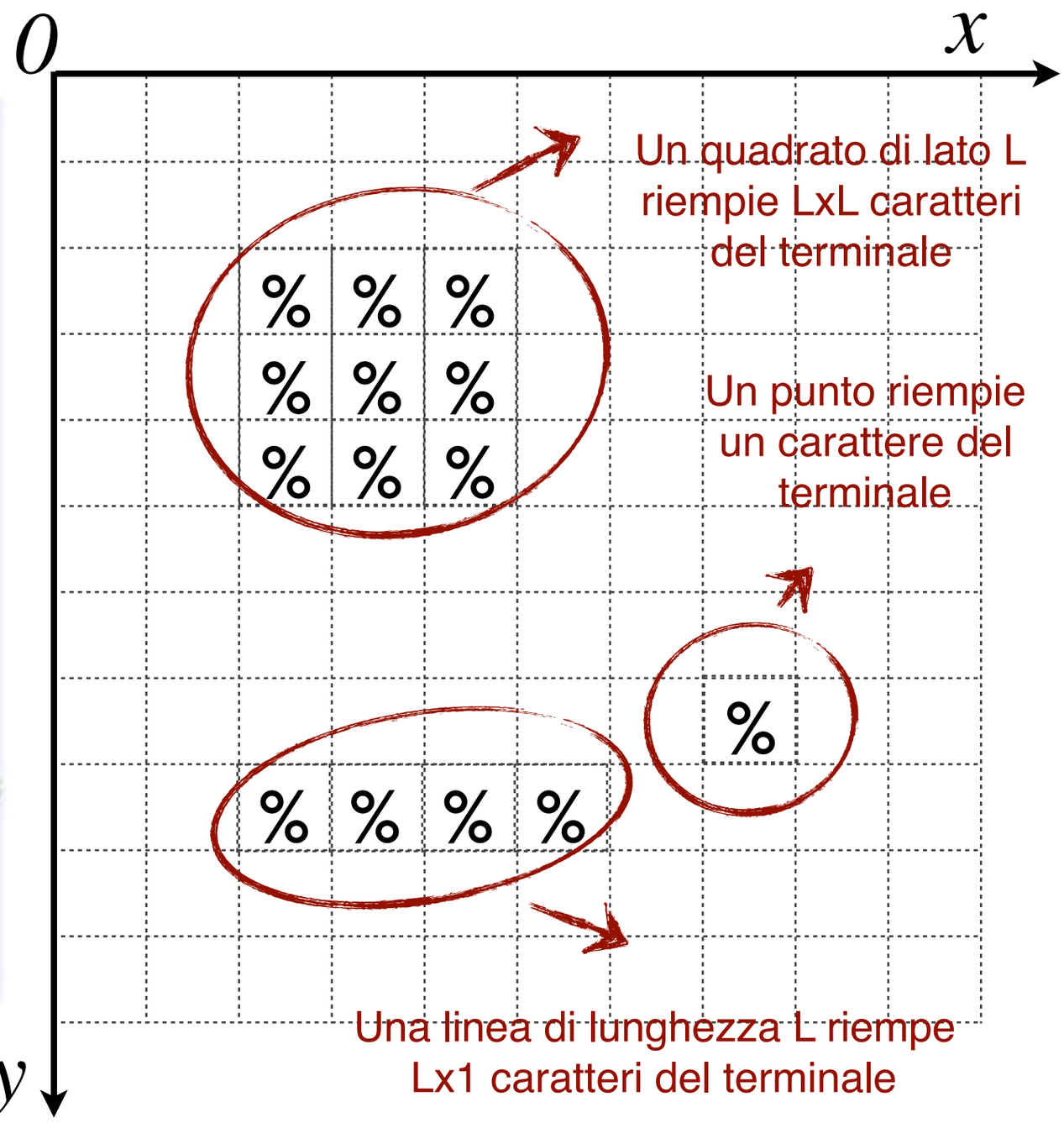
Inserisce il carattere '%' sui bordi della matrice 'schermo'

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

```
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();

// Funzioni sulle matrici che rappresentano 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);
```




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

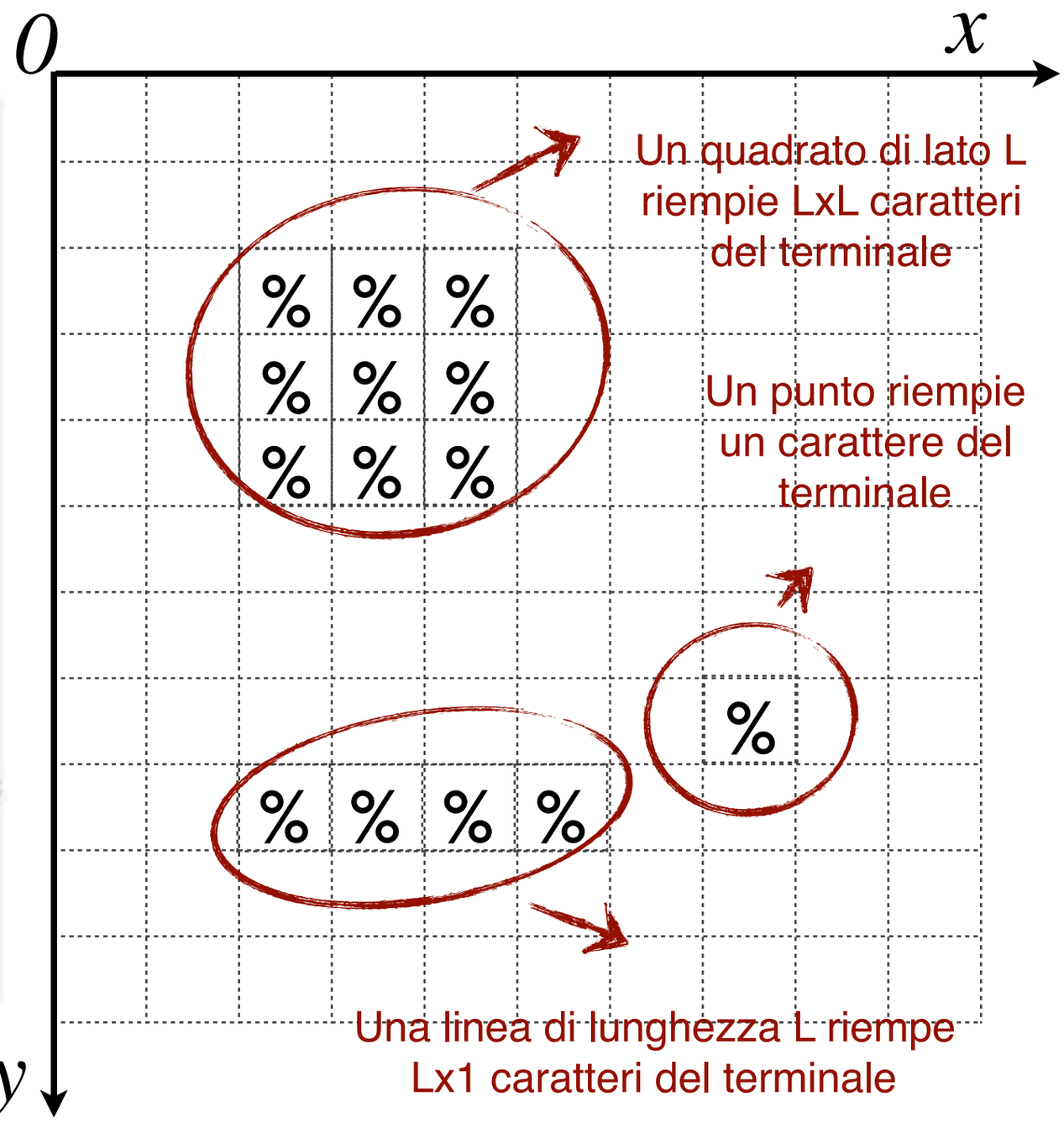
Disegna a schermo la matrice in ingresso 'schermo'

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();

// Funzioni sulle strutture che rappresentano le forme
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();
```

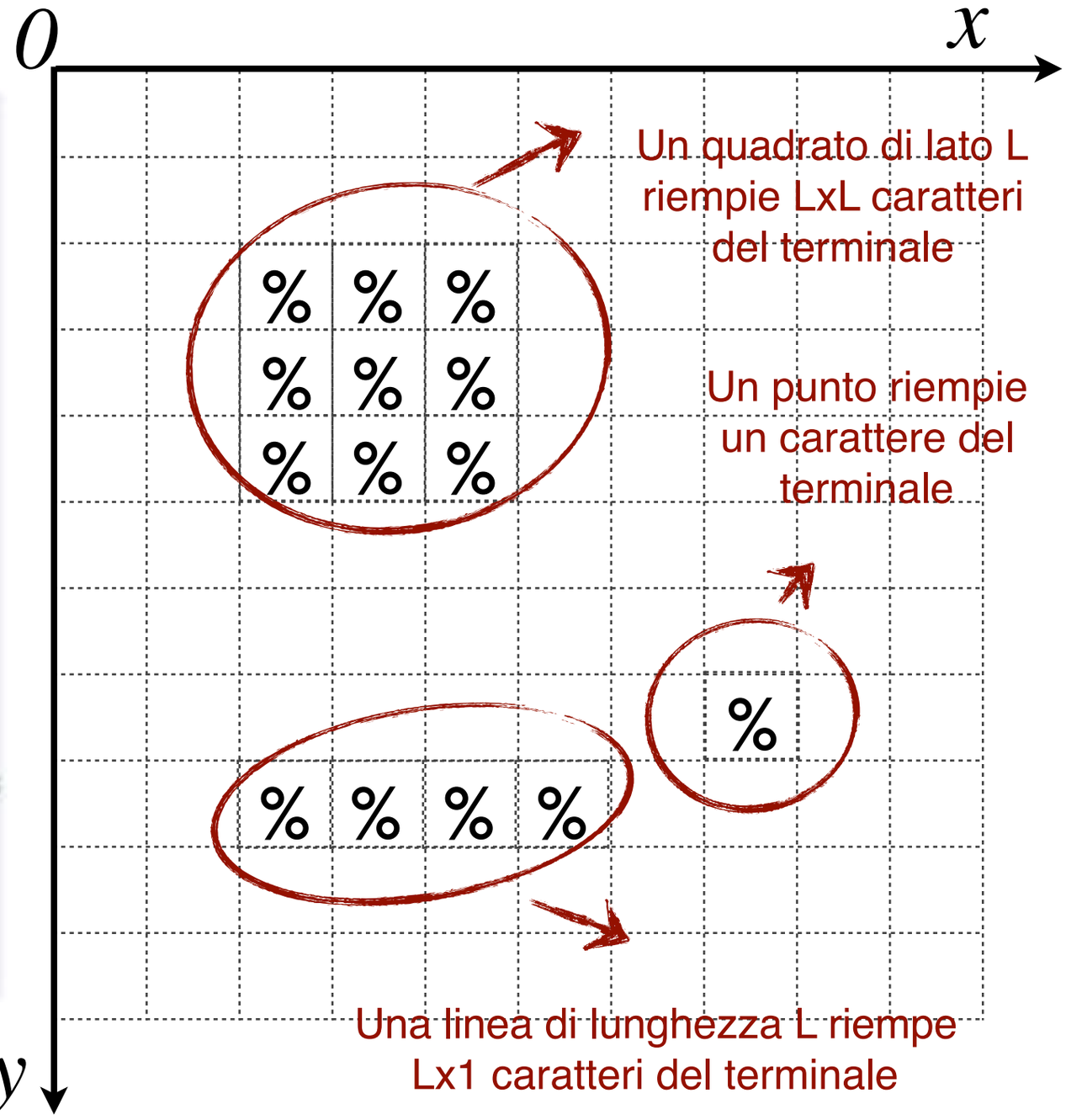
Quando lanciata, mette il programma in attesa di un INVIO da parte dell'utente

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();

// Funzioni sulle strutture che rappresentano le forme
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();
```

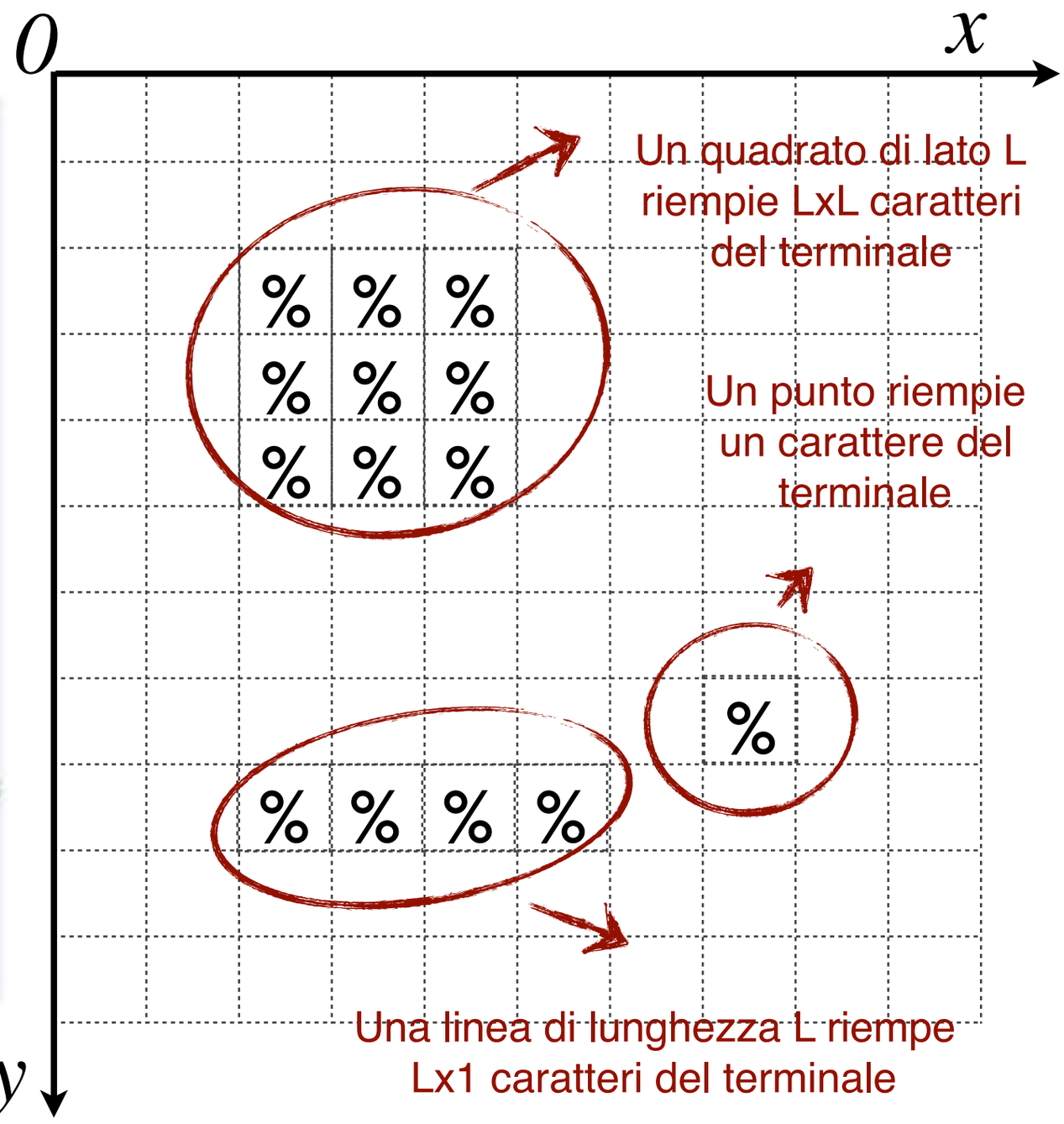
Cancella il contenuto del termianale su cui viene stampato l'output del programma

Pulisci terminale

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

Disegna forma

```
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();

// Funzioni per la costruzione di oggetti
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);

// Costruzione 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]);
```

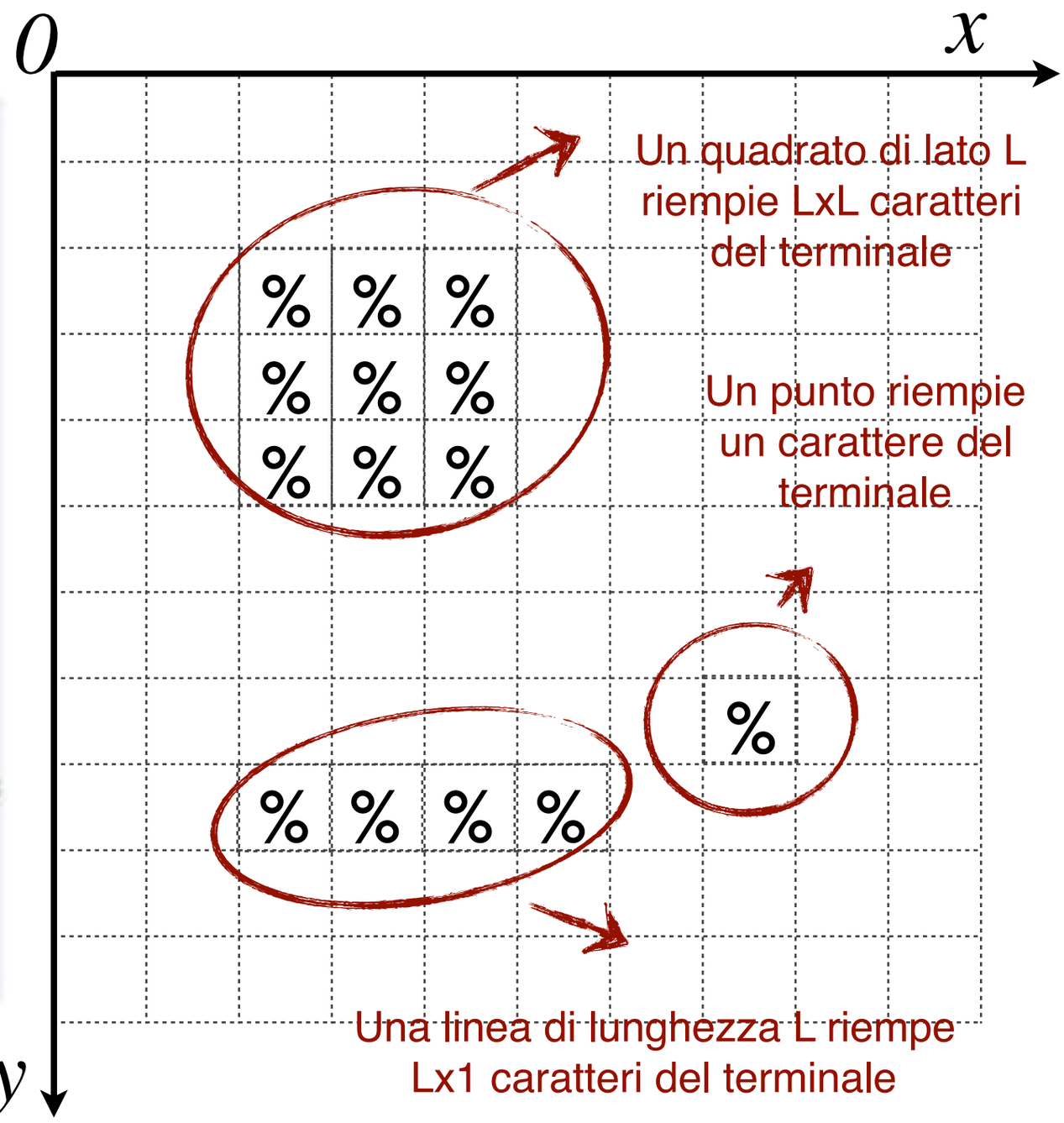
Disegna una data un generica forma 'f' nella posizione 'p' dello schermo 'schermo'

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

```
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();

// Funzioni sulle strutture che rappresentano la 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);
```




```
forma genera_quadrato(int dim, char carattere);
```

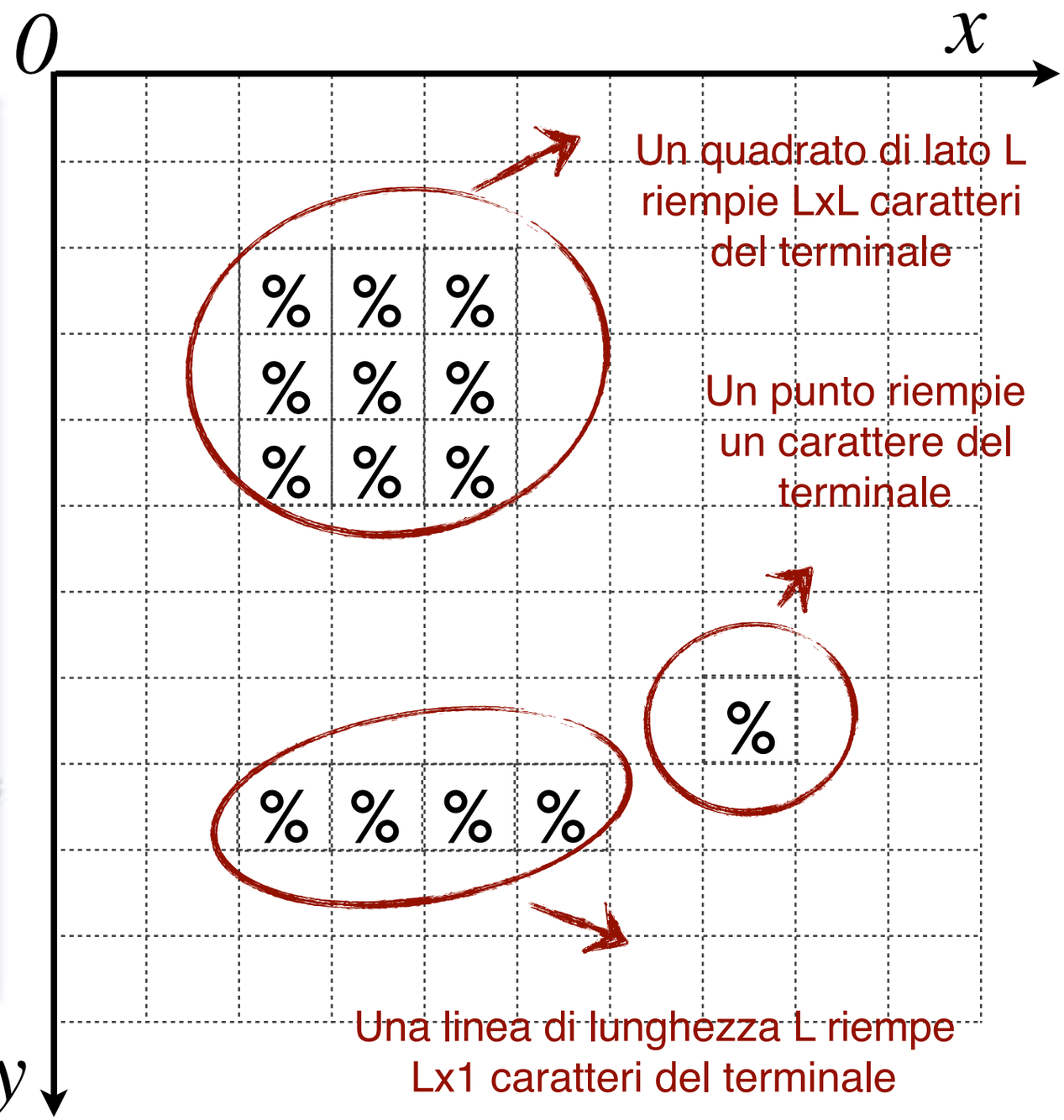
Restituisce una forma quadrata di lato 'dim'

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

```
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();

// Funzioni sulle forme che compongono 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);
```



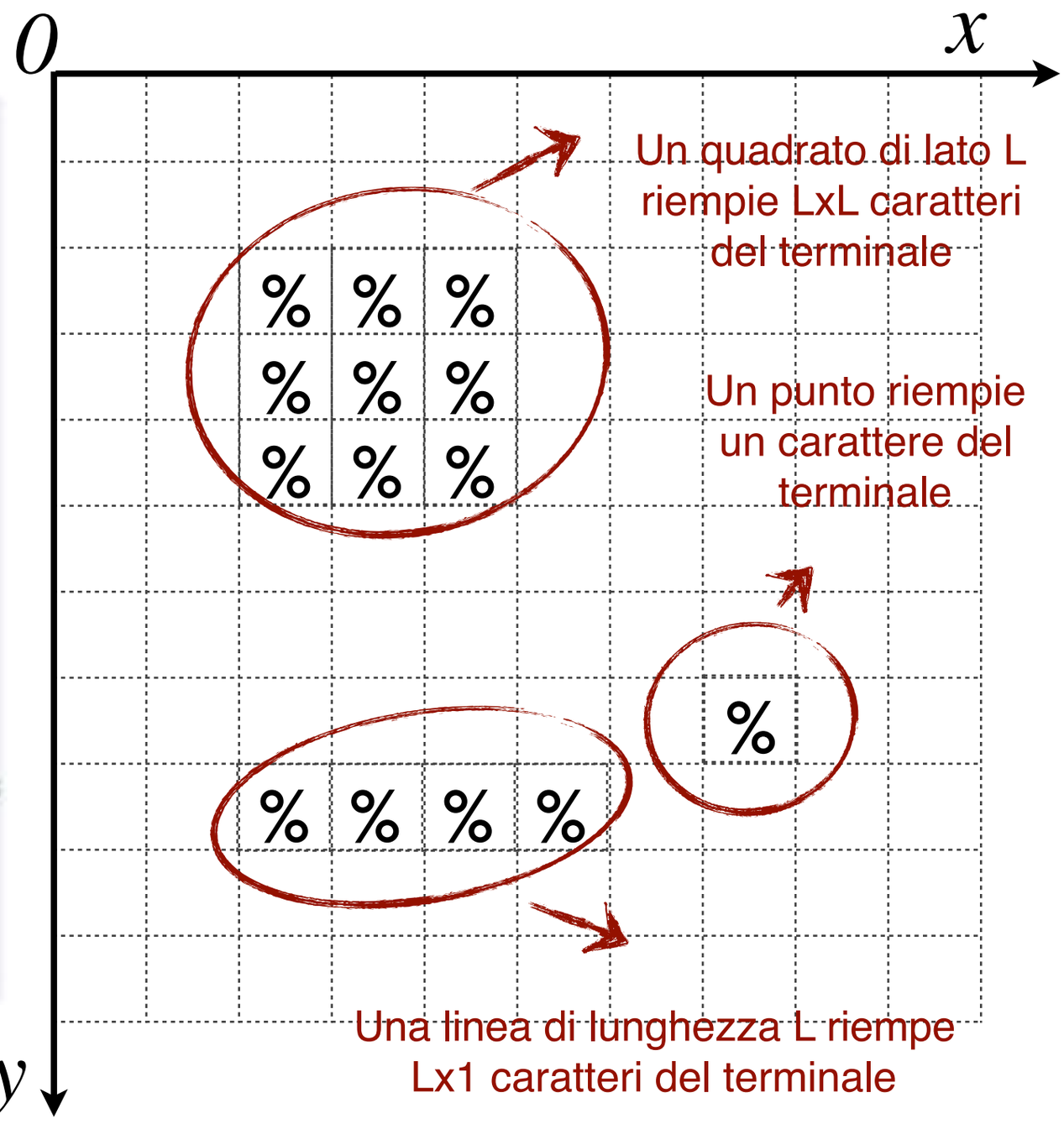

```
forma genera_linea(int dim,
direzione direzione_linea, char carattere);
```

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

```
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();

// Funzioni sulle strutture che rappresentano le forme
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);
```

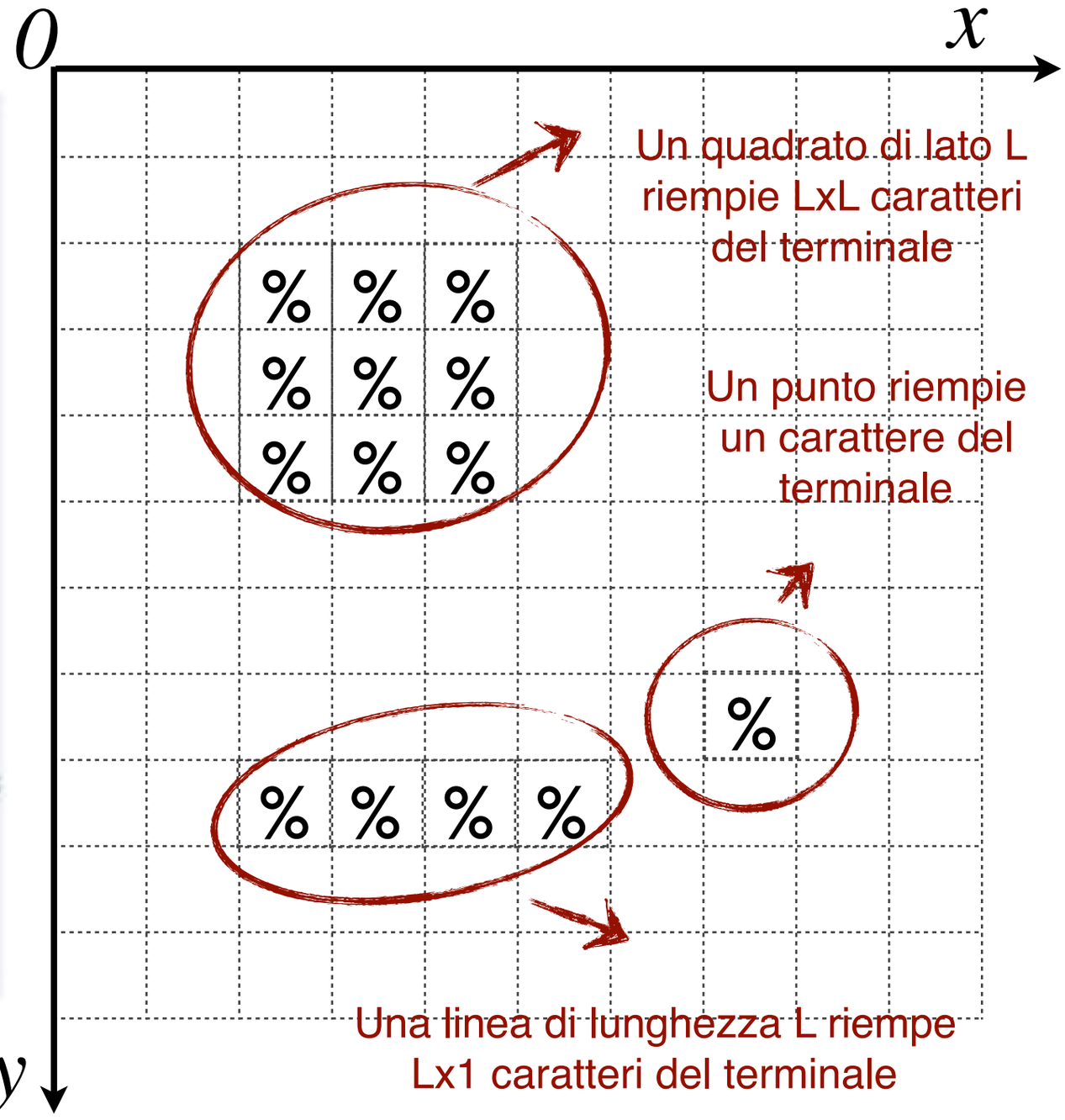
Restituisce una forma punto (linea di lunghezza 1)

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();

// Funzioni sulle strutture che rappresentano la 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);
```




```
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;
```

Implementiamo il main()

```c
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();
```

Implementiamo il main()

```c
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();
```

Implementiamo il main()

```c
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();
```

```c
// 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();
```

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();


}
```

Pause

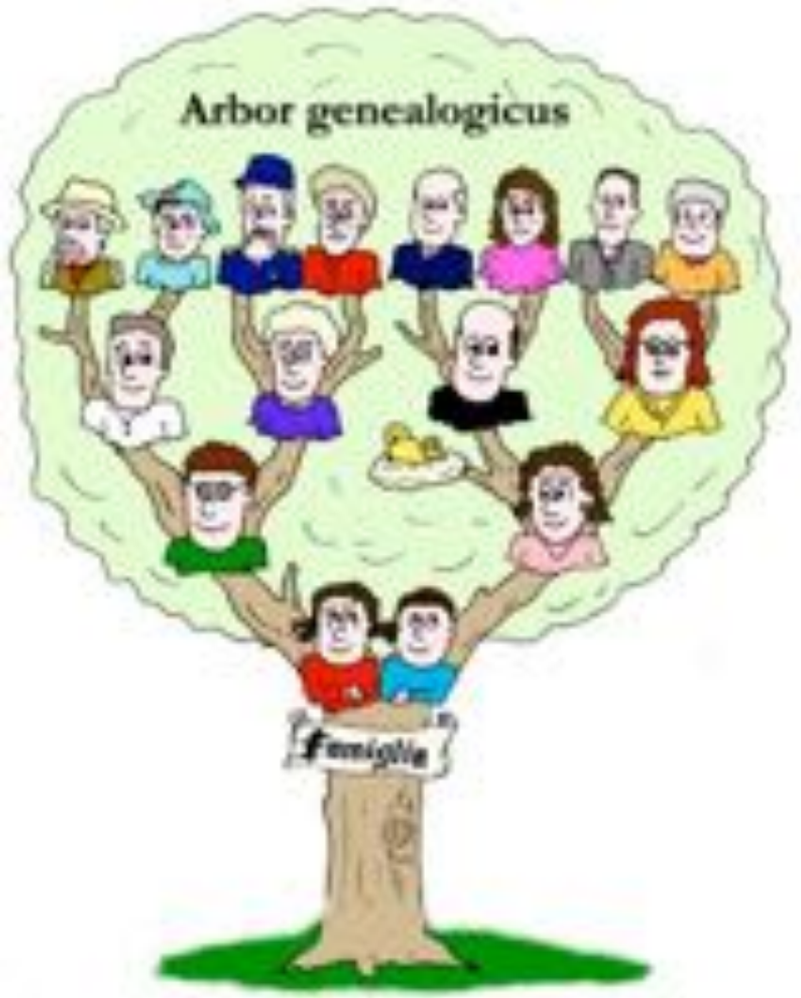
L’albero genealogico

Esercizio 2

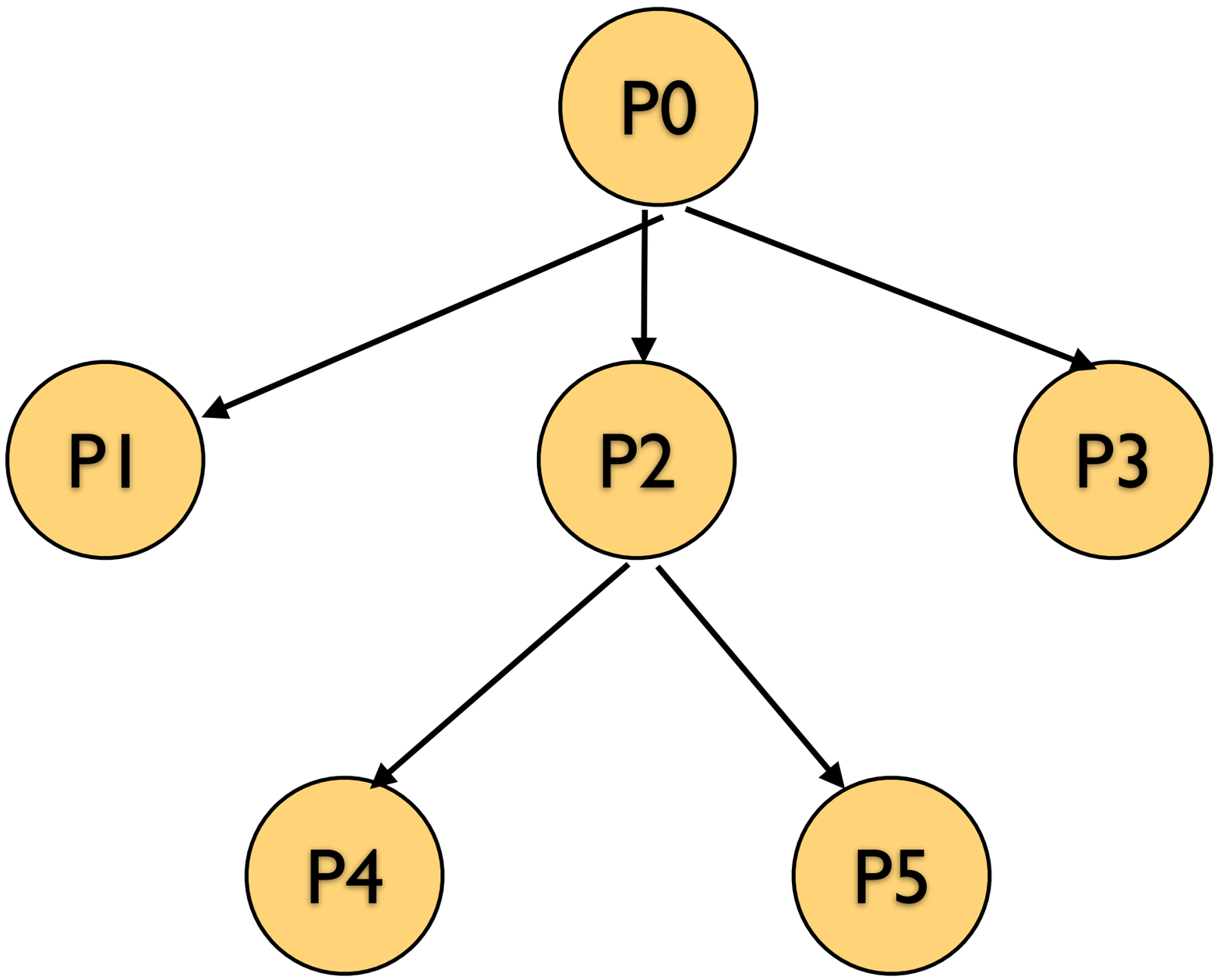
L'albero genealogico

- Scrivere un programma C che sia in grado di rappresentare e gestire un albero genealogico

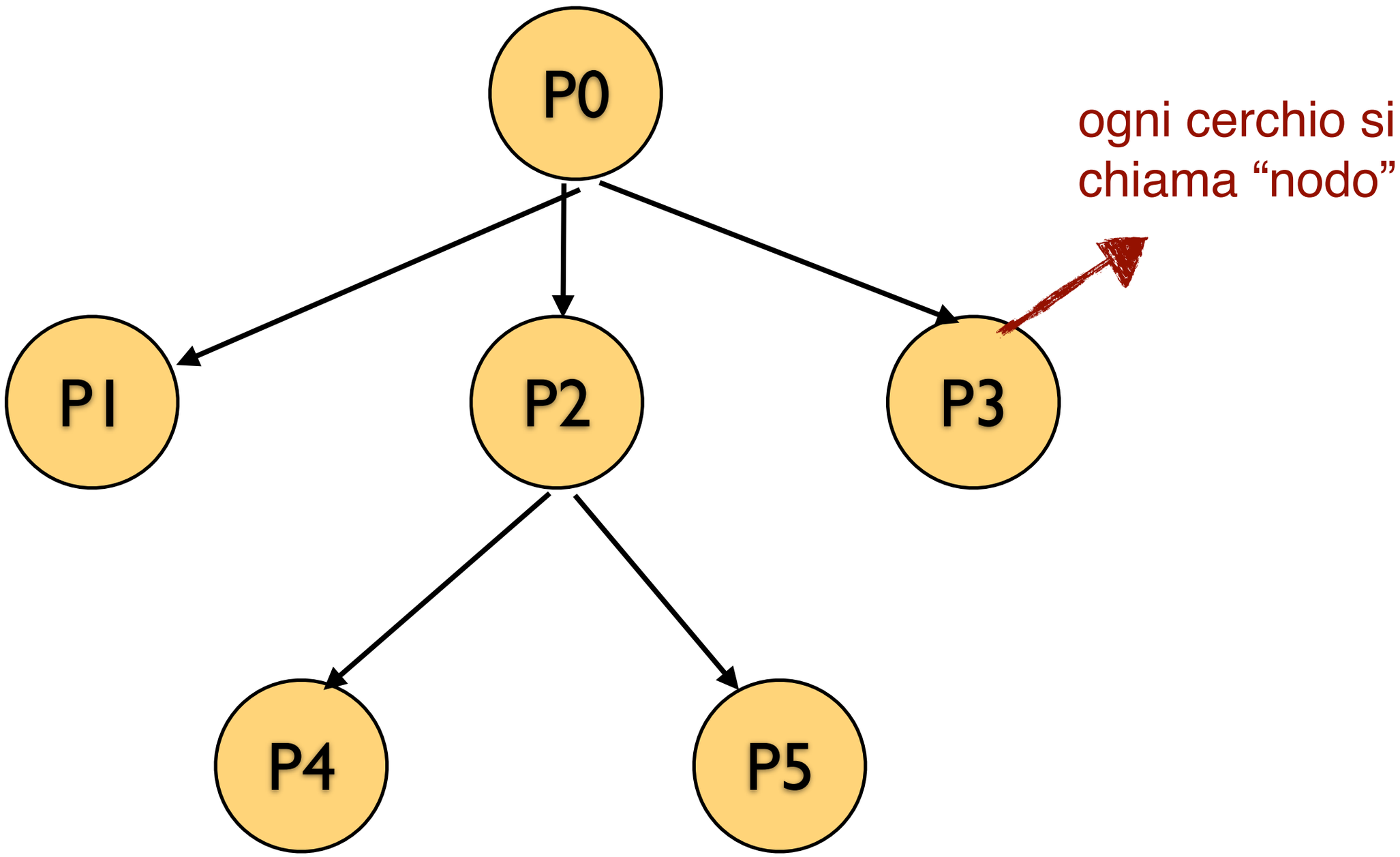
- In particolare, vogliamo poter fare:
  - Creare una persona
  - Rappresentare di una popolazione
  - Aggiungere figli ad una persona
  - Elencare i figli e i nipoti dato un antenato



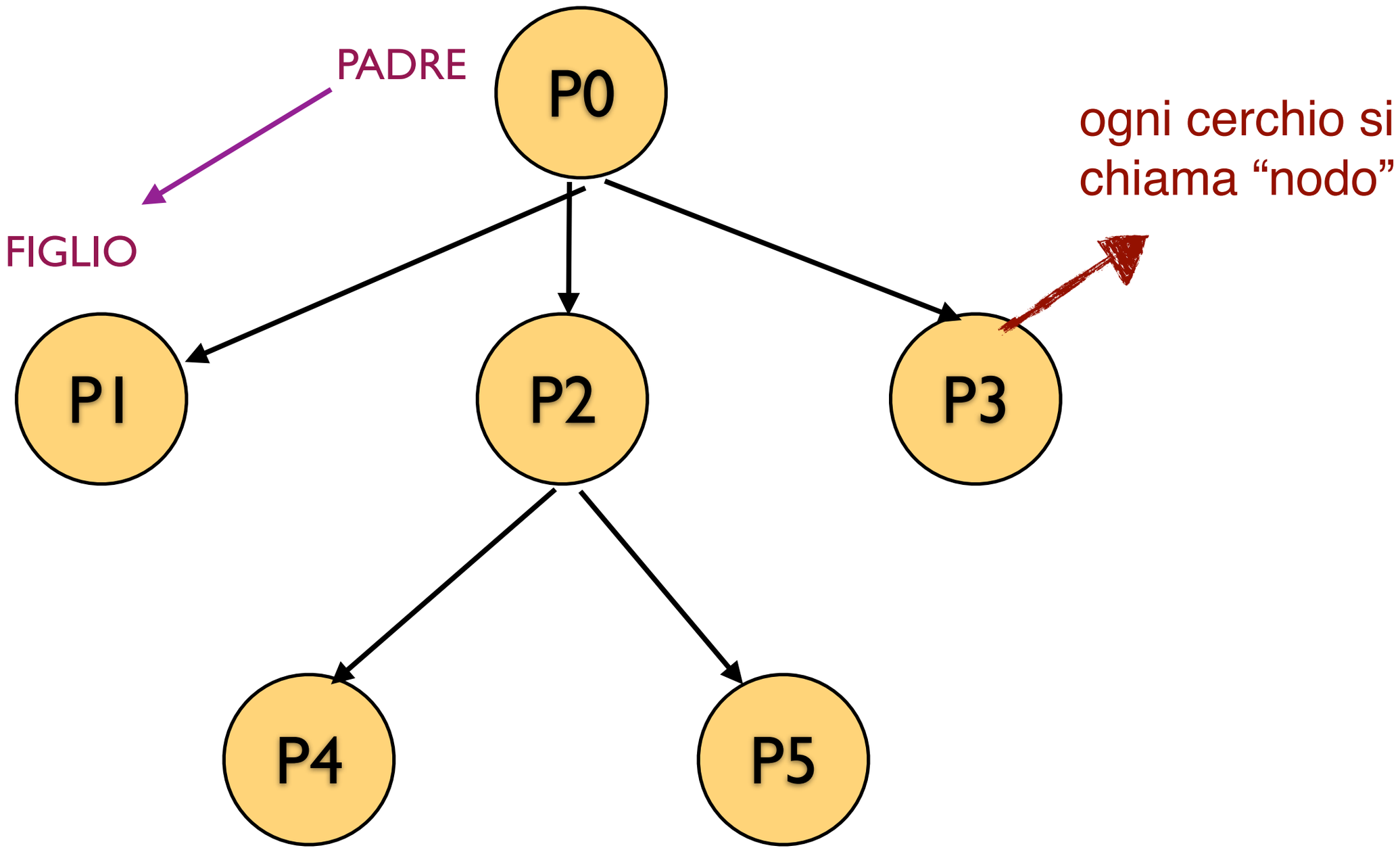
Una famosa struttura dati: l'albero



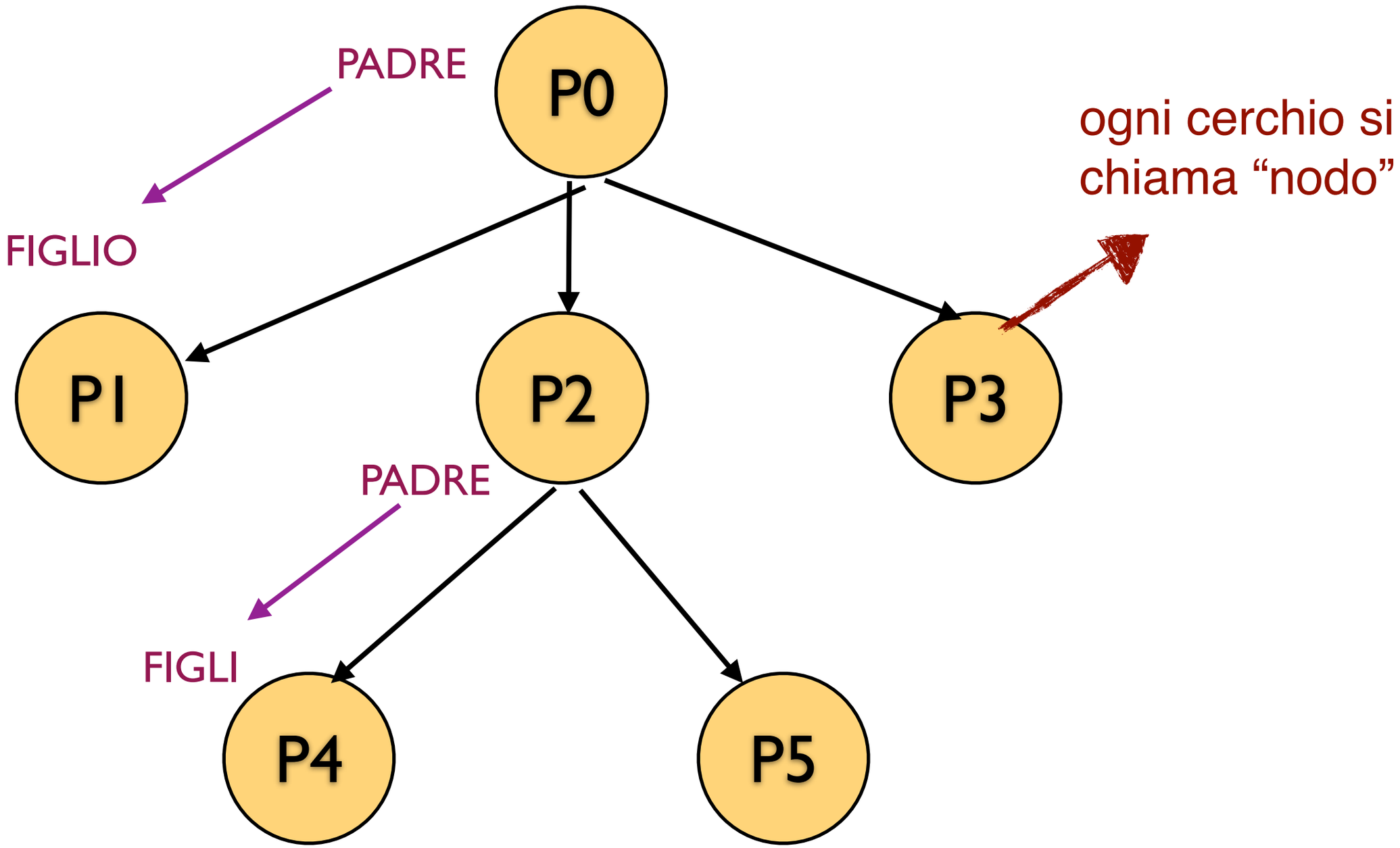
Una famosa struttura dati: l'albero



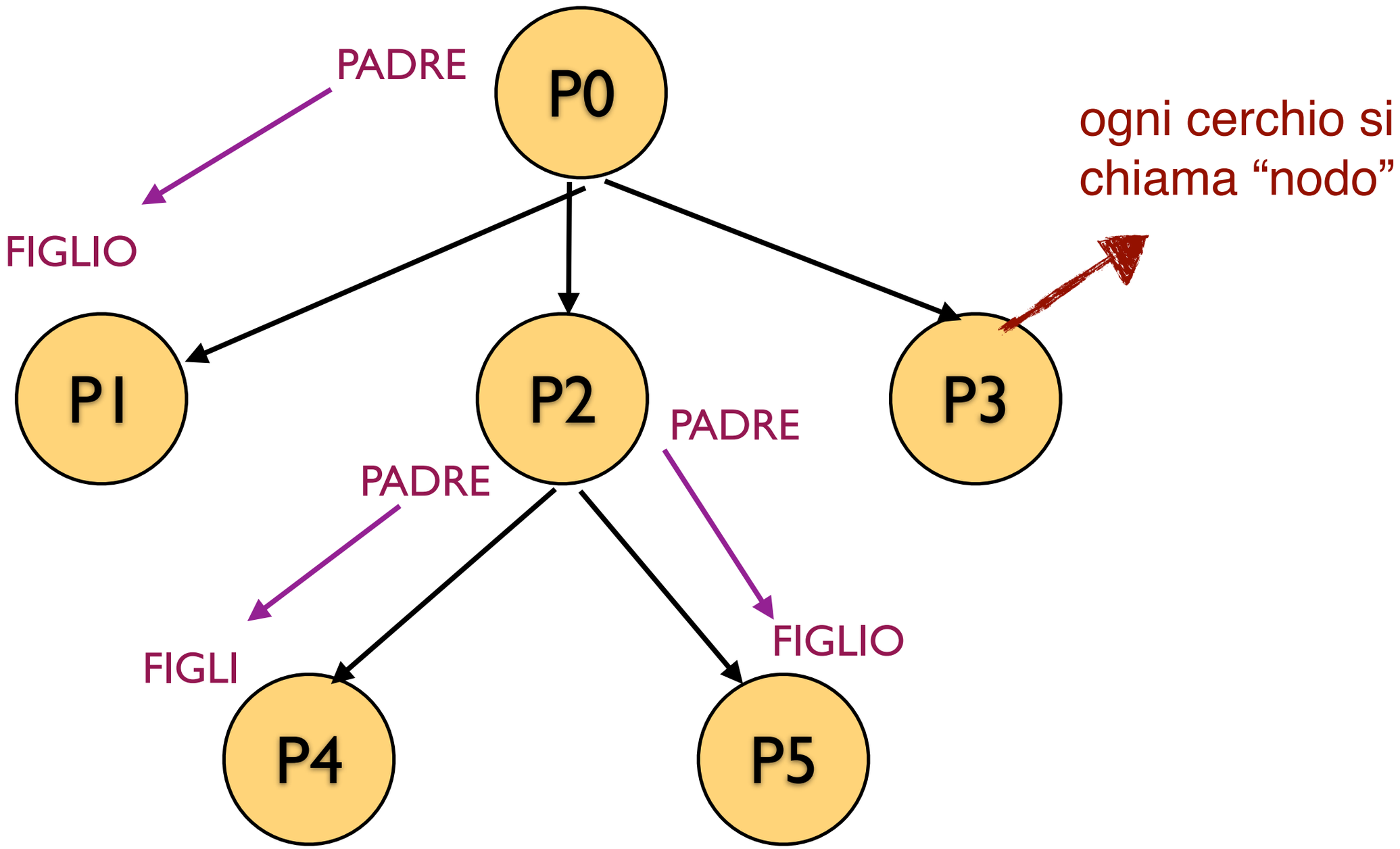
Una famosa struttura dati: l'albero



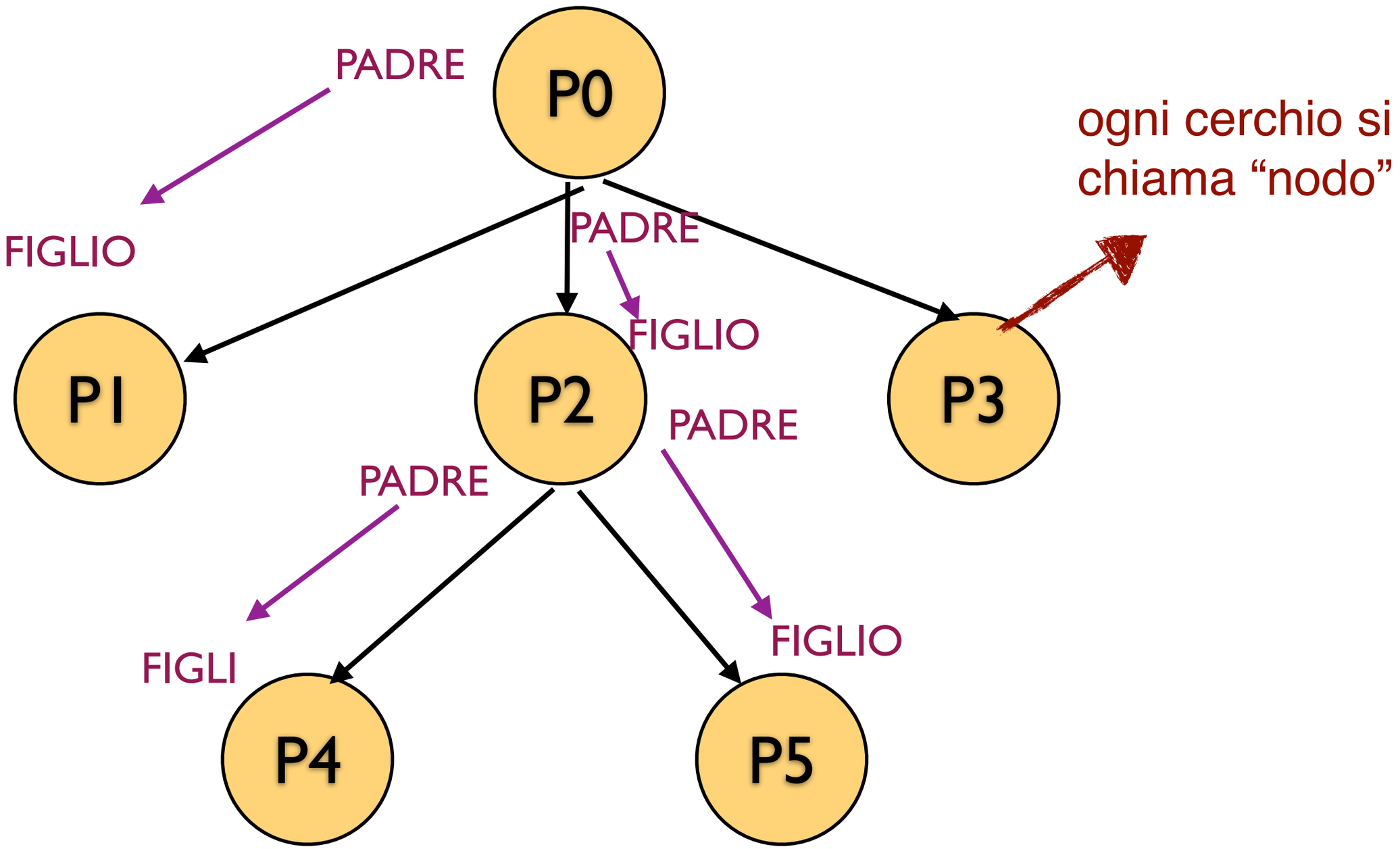
Una famosa struttura dati: l'albero



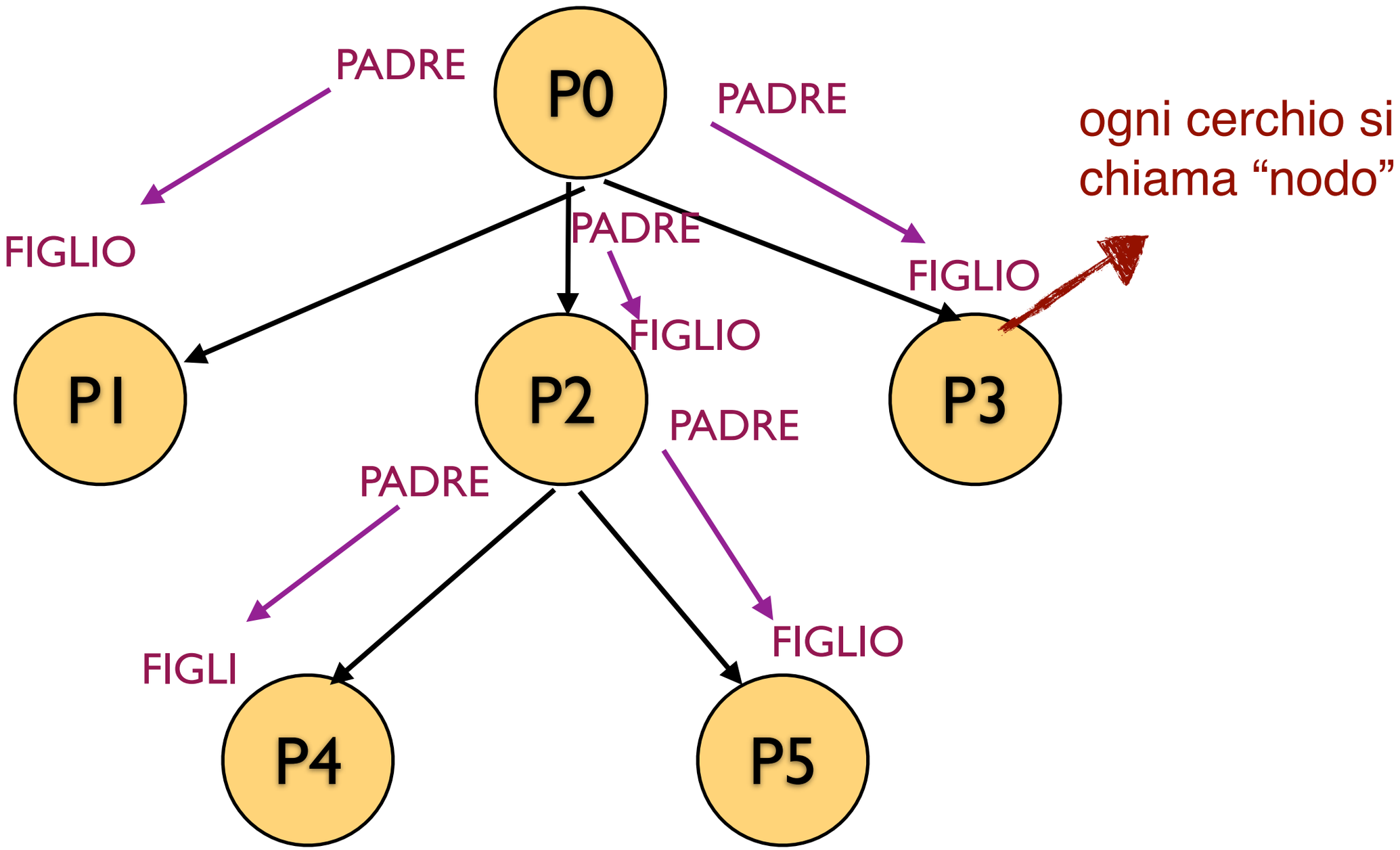
Una famosa struttura dati: l'albero



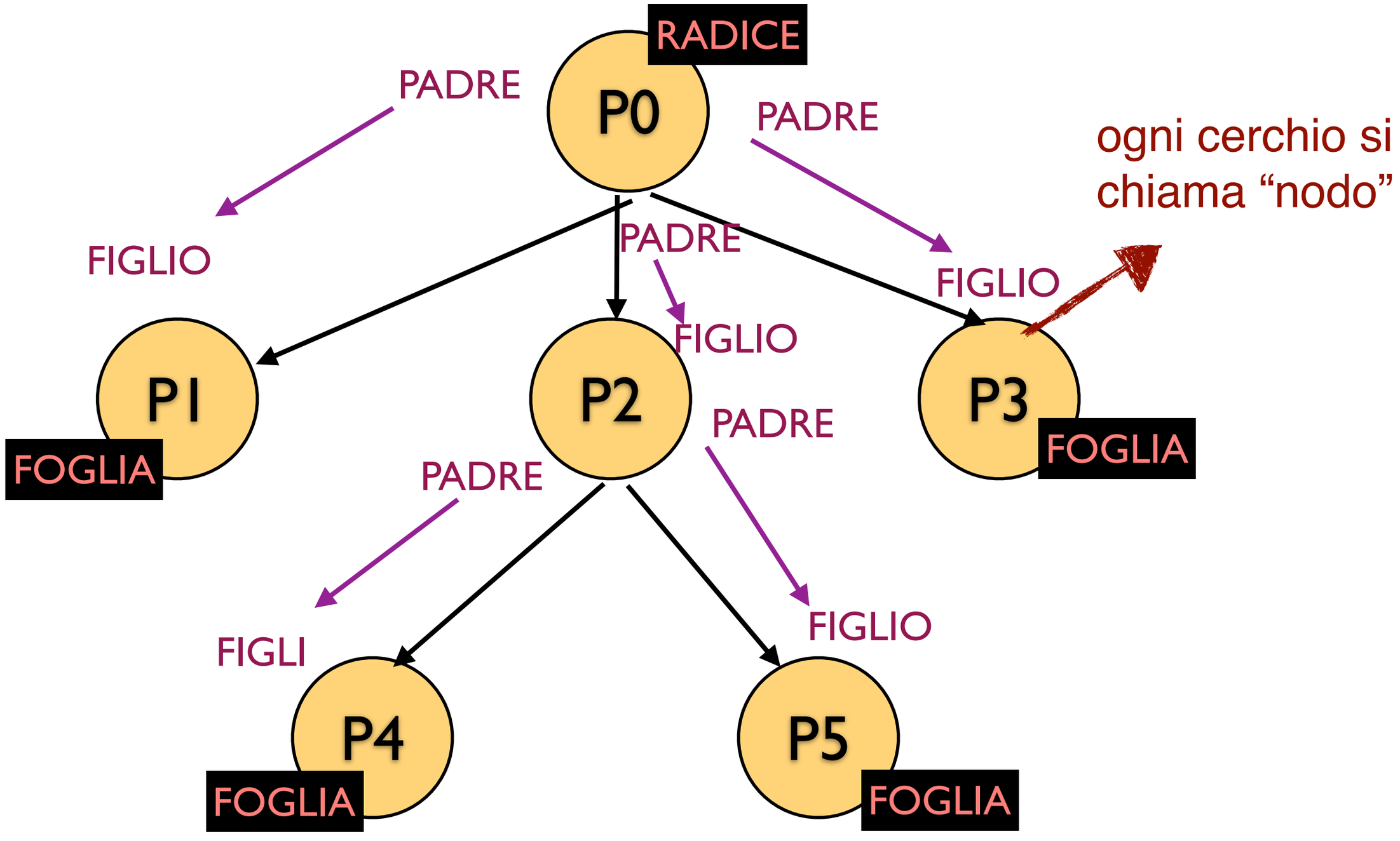
Una famosa struttura dati: l'albero



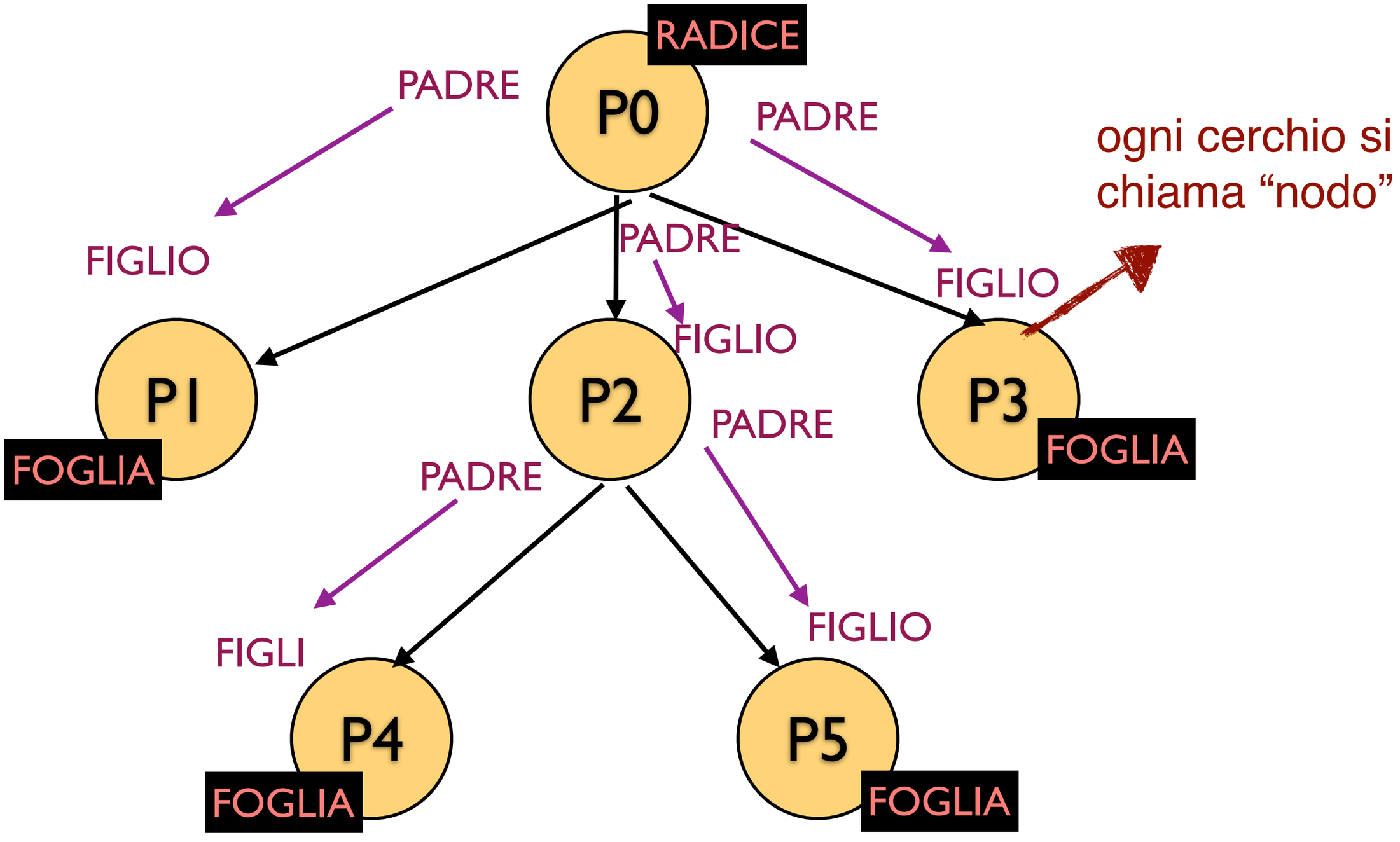
Una famosa struttura dati: l'albero



Una famosa struttura dati: l'albero

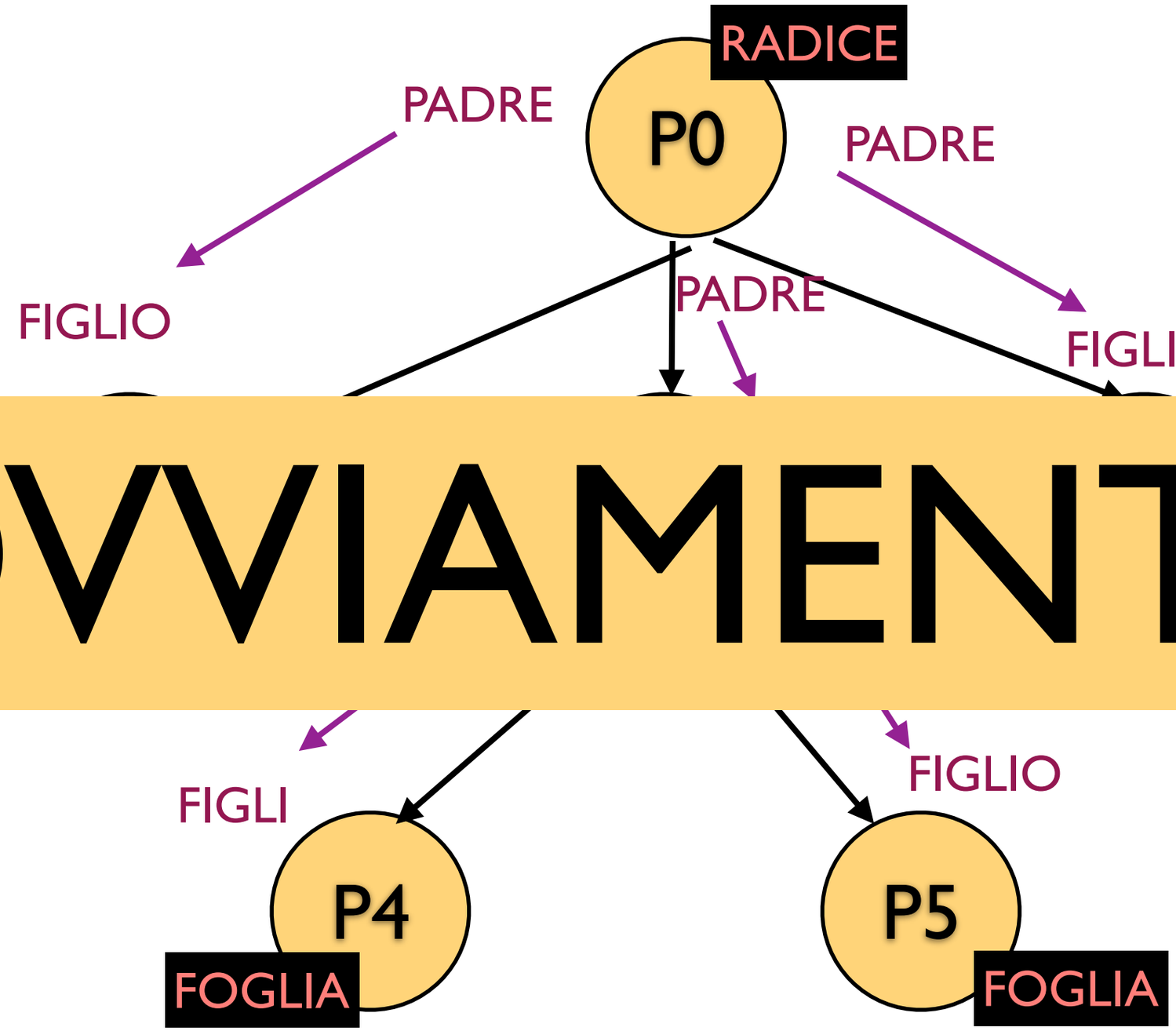


Una famosa struttura dati: l'albero



Può essere utile per rappresentare un albero genealogico?

Una famosa struttura dati: l'albero



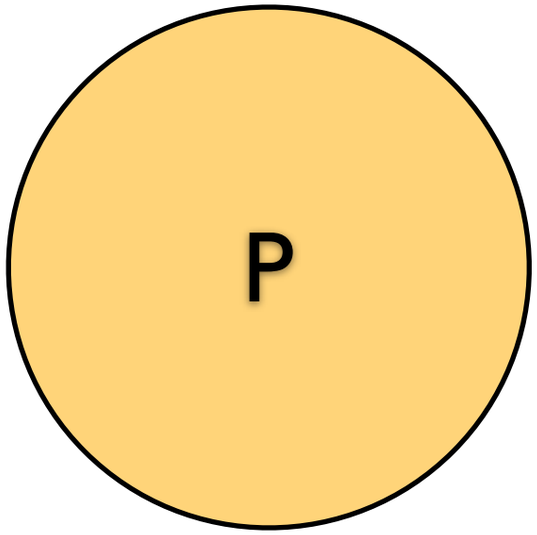
ogni cerchio si chiama "nodo"

OVVIAMENTE SI

Può essere utile per rappresentare un albero genealogico?

- OGNI NODO DELL'ALBERO SARA' PER NOI UNA PERSONA

==

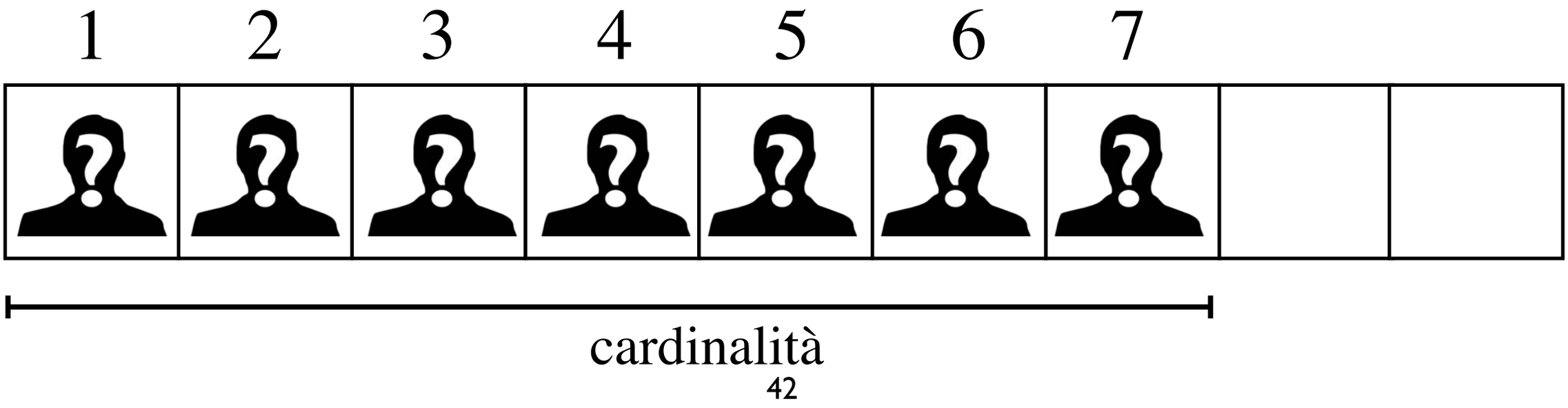


Una Persona

- SESSO
- NOME
- ETA?
- CHI SONO I GENITORI?
- CHI SONO I FIGLI?
- QUANTI FIGLI?

Una Popolazione

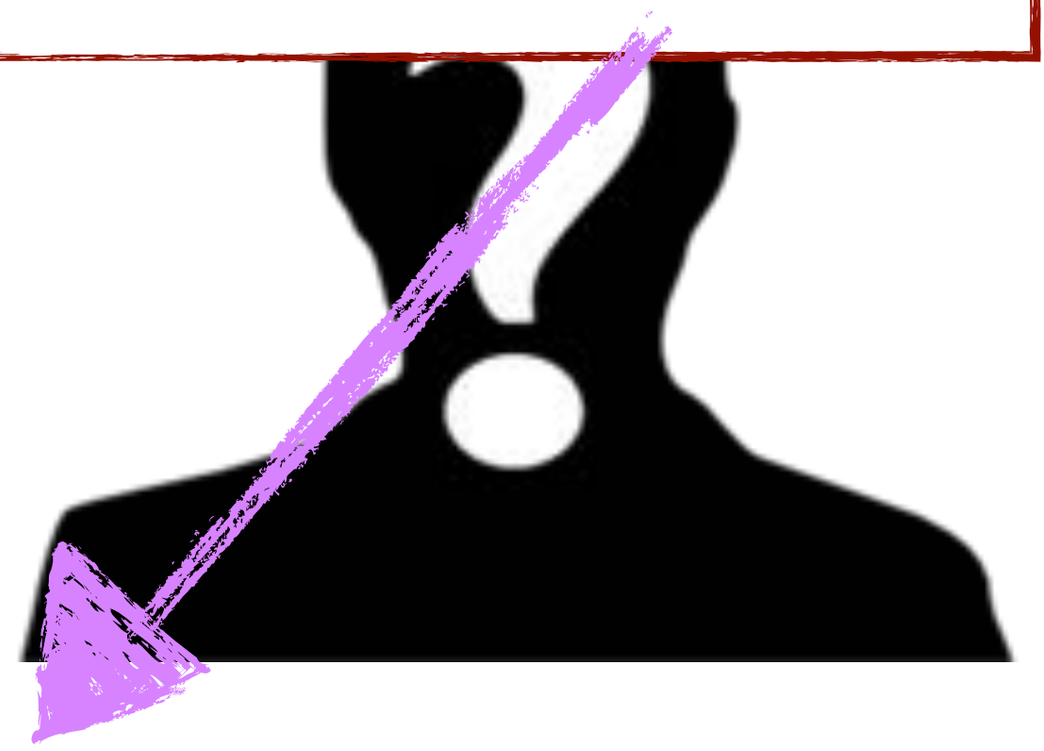
- Una popolazione è rappresentata da un insieme di persone
- Ogni persona ha un suo indice (numero univoco di identificazione)
- Esiste un numero di persone della

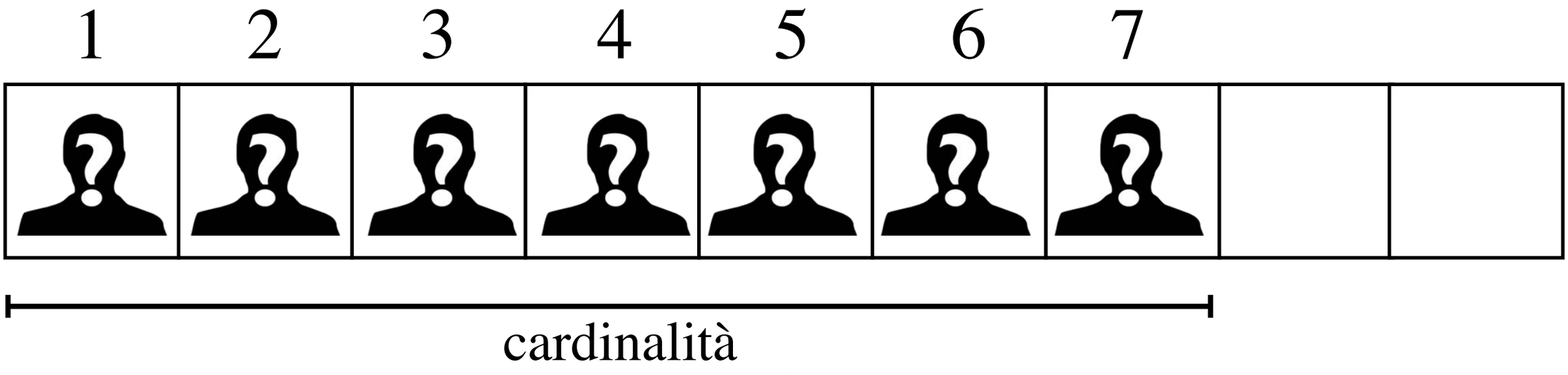


Una Persona nella popolazione

- SESSO
- NOME
- ETA?
- CHI SONO I GENITORI?
- CHI SONO I FIGLI?
- QUANTI FIGLI?

Li rappresentiamo con l'indice della persona nella popolazione





Persona e Popolazione (codice C)

```c
typedef struct {
} persona;

int main () {
    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;
```

Persona e Popolazione (codice C)

```c
typedef struct {
    genere sesso;
    char nome[STR_LEN];
    int i_persona;
    int i_genitore1;
    int i_genitore2;
    int i_figli[MAX_FIGLI];
    int numero_figli;
    int eta;
} persona;
```

```c
int main () |
{

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;
```

Persona e Popolazione (codice C)

```c
typedef enum {MASCHIO, FEMMINA} genere;

typedef struct {
    genere sesso;
    char nome[STR_LEN];
    int i_persona;
    int i_genitore1;
    int i_genitore2;
    int i_figli[MAX_FIGLI];
    int numero_figli;
    int eta;
} persona;
```

```c
int main () {

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;
```

Creazione di una persona

```
persona crea_persona(genere sesso, char nome[STR_LEN], int eta)
{

}
```

Creazione di una persona

```
persona crea_persona(genere sesso, char nome[STR_LEN], int eta)
{
    persona p;
    p.sesso = sesso;
    strcpy(p.nome,nome);
    p.i_genitore1 = -1;
    p.i_genitore2 = -1;
    p.numero_figli = 0;
    p.eta = eta;
    return p;
}
```

Aggiunta persona alla popolazione

```
int aggiungi_a_popolazione(persona* popolazione[MAX_PERSONE],
    persona* p, int* cardinalita_popolazione)
{

}
```

Aggiunta persona alla popolazione

```
int aggiungi_a_popolazione(persona* popolazione[MAX_PERSONE],
    persona* p, int* cardinalita_popolazione)
{
    (*cardinalita_popolazione)++;
    p->i_persona = *cardinalita_popolazione;
    popolazione[*cardinalita_popolazione] = p;
}
```

Aggiunta di un figlio

```
void aggiungi_figlio(persona* genitore, persona* figlio)
{

}
```

Aggiunta di un figlio

```
void aggiungi_figlio(persona* genitore, persona* figlio)
{
    genitore->i_figli[genitore->numero_figli] = figlio->i_persona;
    genitore->numero_figli++;

    if (figlio->i_genitore1 == -1)
        figlio->i_genitore1 = genitore->i_persona;
    else if (figlio->i_genitore2 == -1)
        figlio->i_genitore2 = genitore->i_persona;
    else
        printf("errore\n");

}
```

Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)
{



}

void stampa_persona(persona* p)
{


}
```

Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)
{
    if (sesso == MASCHIO) return "MASCHIO";
    if (sesso == FEMMINA) return "FEMMINA";
    return "?";
}

void stampa_persona(persona* p)
{


}
```

Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)
{
    if (sesso == MASCHIO) return "MASCHIO";
    if (sesso == FEMMINA) return "FEMMINA";
    return "?";
}

void stampa_persona(persona* p)
{
    printf("Nome: %s - Genere: %s - Eta':%d - Id:%d - #Figli: %d\n",
    p->nome, genere_to_str(p->sesso), p->eta, p->i_persona, p->numero_figli);
}
```

Elenco dei figli e dei nipoti

```
void elenca_figli_nipoti(persona* popolazione[MAX_PERSONE],
    persona* p, genere sesso, int eta_minima)
{
    ...
}
```

Elenco dei figli e dei nipoti

```
void elenca_figli_nipoti(persona* popolazione[MAX_PERSONE],
    persona* p, genere sesso, int eta_minima)
{

    int i;

    if (p->sesso == sesso && p->eta >= eta_minima)
        stampa_persona(p);


    for (i = 0; i < p->numero_figli; i++)
    {
        persona* f = popolazione[p->i_figli[i]];
        elenca_figli_nipoti(popolazione, f, sesso, eta_minima);
    }
}
```

La nostra popolazione

MARCO	STEFANIA	LUCA	PIPPO	LUCIA	ARIANNA	RINALDO	STEFANO
P0	P1	P2	P3	P4	P5	P6	P7

La nostra popolazione

MARCO P0 | STEFANIA P1 | LUCA P2 | PIPPO P3 | LUCIA P4 | ARIANNA P5 | RINALDO P6 | STEFANO P7

Marco e' padre di LUCA e di PIPPO

Stefania e' madre di LUCA e di PIPPO

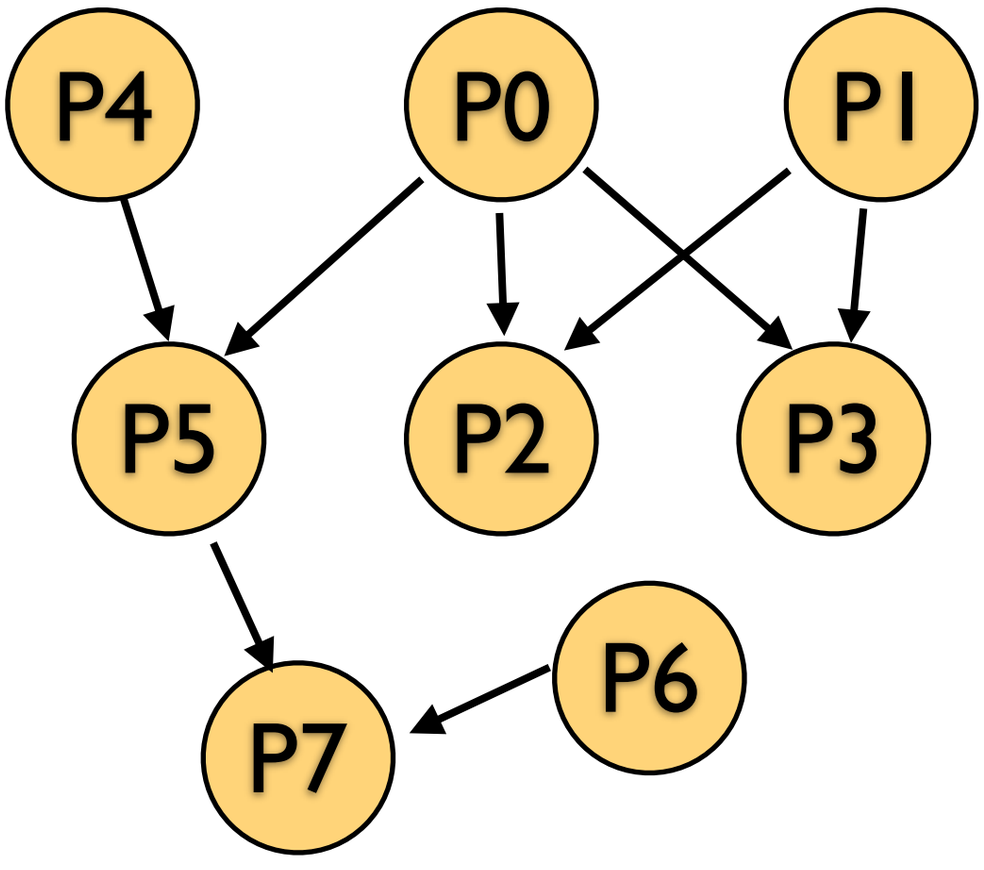
Arianna e' figlia di Marco e Lucia

Stefano e' figlio di Arianna e Rinaldo

La nostra popolazione

MARCO P0 | STEFANIA P1 | LUCA P2 | PIPPO P3 | LUCIA P4 | ARIANNA P5 | RINALDO P6 | STEFANO P7

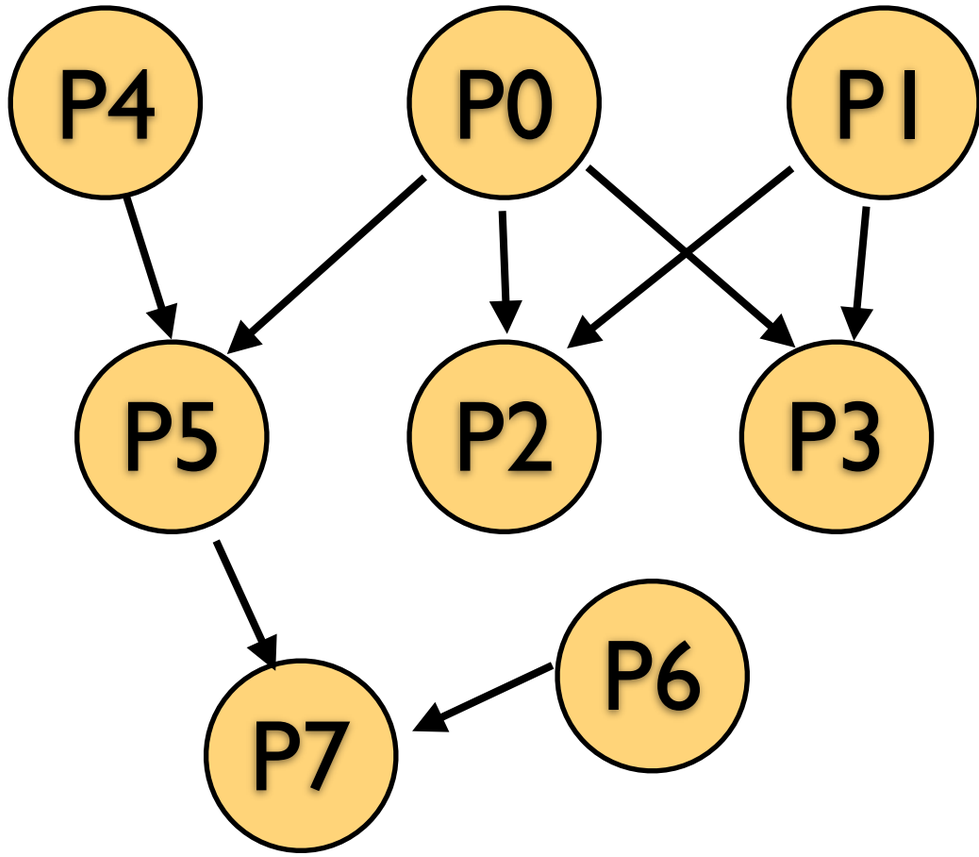
Marco e' padre di LUCA e di PIPPO
Stefania e' madre di LUCA e di PIPPO
Arianna e' figlia di Marco e Lucia
Stefano e' figlio di Arianna e Rinaldo



La nostra popolazione (codice C)

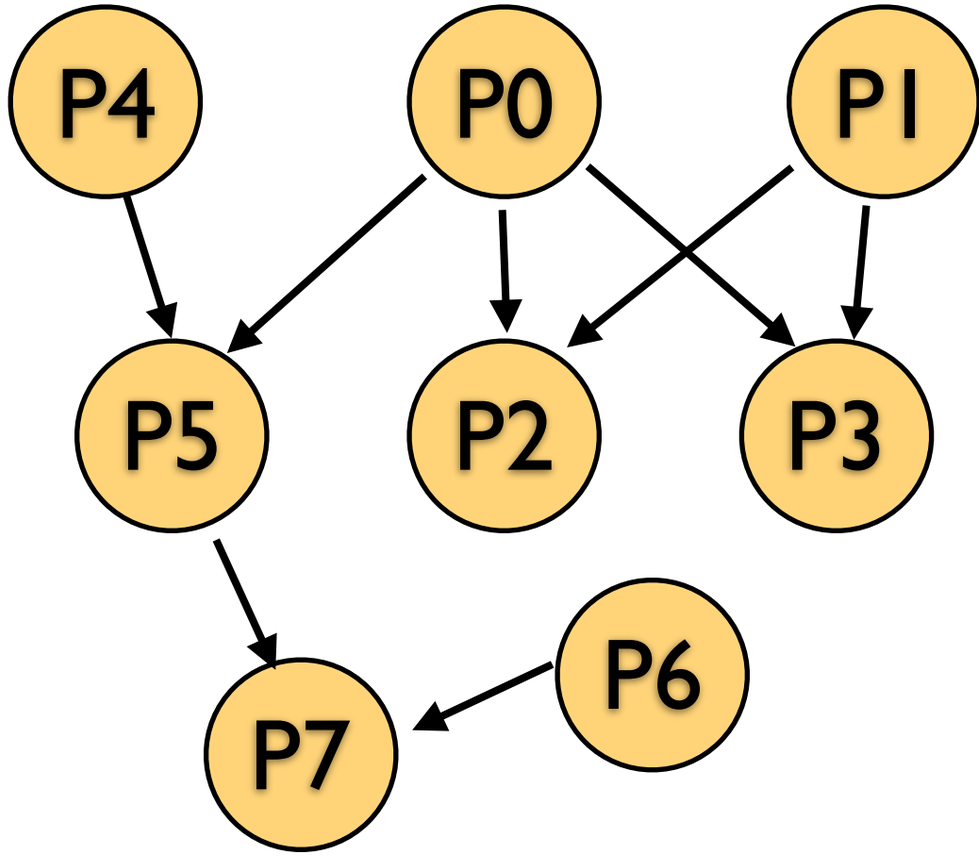
```c
int main ()
{

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;

    persona p0 = crea_persona(MASCHIO, "MARCO", 50);
    persona p1 = crea_persona(FEMMINA, "STEFANIA", 49);
    persona p2 = crea_persona(MASCHIO, "LUCA", 30);
    persona p3 = crea_persona(MASCHIO, "PIPPO", 26);
    persona p4 = crea_persona(FEMMINA, "LUCIA", 53);
    persona p5 = crea_persona(FEMMINA, "ARIANNA", 30);
    persona p6 = crea_persona(MASCHIO, "RINALDO", 32);
    persona p7 = crea_persona(MASCHIO, "STEFANO", 10);

    aggiungi_a_popolazione(popolazione, &p0, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p1, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p2, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p3, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p4, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p5, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p6, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p7, &cardinalita_popolazione);
```

Aggiungiamo le parentele



Aggiungiamo le parentele

```
// Marco e' padre di LUCA e di PIPPO
aggiungi_figlio(popolazione[0], popolazione[2]);
aggiungi_figlio(popolazione[0], popolazione[3]);

// Stefania e' madre di LUCA e di PIPPO
aggiungi_figlio(popolazione[1], popolazione[2]);
aggiungi_figlio(popolazione[1], popolazione[3]);

// Arianna e' figlia di Marco e Lucia
aggiungi_figlio(popolazione[0], popolazione[5]);
aggiungi_figlio(popolazione[4], popolazione[5]);

// Stefano e' figlio di Arianna e Rinaldo
aggiungi_figlio(popolazione[5], popolazione[7]);
aggiungi_figlio(popolazione[6], popolazione[7]);
```



Il main()

```c
int main ()
{

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;

    persona p0 = crea_persona(MASCHIO, "MARCO", 50);
    persona p1 = crea_persona(FEMMINA, "STEFANIA", 49);
    persona p2 = crea_persona(MASCHIO, "LUCA", 30);
    persona p3 = crea_persona(MASCHIO, "PIPPO", 26);
    persona p4 = crea_persona(FEMMINA, "LUCIA", 53);
    persona p5 = crea_persona(FEMMINA, "ARIANNA", 30);
    persona p6 = crea_persona(MASCHIO, "RINALDO", 32);
    persona p7 = crea_persona(MASCHIO, "STEFANO", 10);

    aggiungi_a_popolazione(popolazione, &p0, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p1, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p2, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p3, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p4, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p5, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p6, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p7, &cardinalita_popolazione);

    // Marco e' padre di LUCA e di PIPPO
    aggiungi_figlio(popolazione[0], popolazione[2]);
    aggiungi_figlio(popolazione[0], popolazione[3]);

    // Stefania e' madre di LUCA e di PIPPO
    aggiungi_figlio(popolazione[1], popolazione[2]);
    aggiungi_figlio(popolazione[1], popolazione[3]);

    // Arianna e' figlia di Marco e Lucia
    aggiungi_figlio(popolazione[0], popolazione[5]);
    aggiungi_figlio(popolazione[4], popolazione[5]);

    // Stefano e' figlio di Arianna e Rinaldo
    aggiungi_figlio(popolazione[5], popolazione[7]);
    aggiungi_figlio(popolazione[6], popolazione[7]);


    elenca_figli_nipoti(popolazione, popolazione[0], MASCHIO, 3);

    return 0;

}
```

Tutte il materiale sarà disponibile sul mio sito internet:

alessandronacci.it

L’enumerazione

Esercizio 3

Il parco macchine

- Quale è l'ouput del seguente codice?

```
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

Il parco macchine

- Quale è l'ouput del seguente codice?

```c
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

Il parco macchine

- Quale è l'ouput del seguente codice?

```c
#include <stdio.h>

   typedef enum{bianco,azzurro_chiaro,giallo,rosso,verde_scuro,rosa,
      azzurro_scuro,verde_chiaro,nero,marrone} colore;

int main(){



   int auto_codice[3];          //ATTENZIONE QUI!
   colore auto_colore[3];       // tipo_di_dato nome_varibile[]
   int i;

   auto_codice[0] = 1000;
   auto_colore[0] = giallo;

   auto_codice[1] = 4000;
   auto_colore[1] = rosa;

   auto_codice[2] = 8000;
   auto_colore[2] = nero;



   for (i = 0; i < 3; i++ )
      printf("L'auto con codice %d e' di colore %d\n",
         auto_codice[i], auto_colore[i]);

   return 0;
}
```

```
Terminal — bash — 80×24
> ./es3
> L'auto con codice 1000 e' di colore 2
> L'auto con codice 4000 e' di colore 5
> L'auto con codice 8000 e' di colore 8
```

MA QUINDI A COSA SERVE L'ENUM?

Solo a semplificare la vita al programmatore! Non ti devi ricordare la corrispondenza numerica!a


That's all Folks!

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it

- Sabato 21 maggio
- Ore 9.00 - 13.00
- smartbrainmnemonica@gmail.com