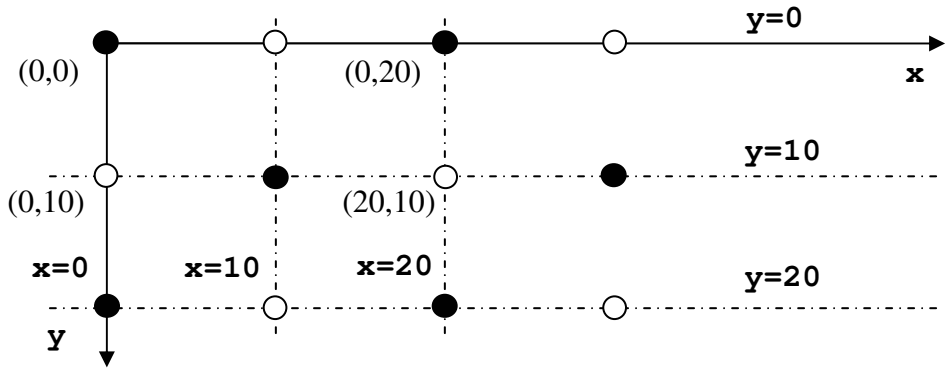
Paolo Torroni, *Fondamenti di Informatica L-A* (*Elettronica e Telecomunicazioni*)
14-15/11/2006 BIS

1. Dichiarare una funzione **`init_pixel()`** che restituisca un **`pixel`**, e che abbia come valori di input due **`int`** e tre **`char`**.
   NOTA: **`pixel`** è un tipo di dato definito nel modo seguente:
   ```
   typedef struct {
      char R,G,B; } RGB;
   typedef struct {
      int x,y; RGB color; } pixel;
   ```

2. Definire la funzione **`init_pixel()`** di cui sopra. La funzione deve inizializzare una variabile di tipo **`pixel`**, che verrà restituita in uscita, utilizzando i valori dei parametri di input.
3. Dichiarare una funzione **`init_rif_pixel()`** che restituisca un **`int`**, e che abbia come valori di input due **`int`**, tre **`char`**, e l'indirizzo di una variabile di tipo **`pixel`**.
4. Definire la funzione **`init_rif_pixel()`** di cui sopra. La funzione deve inizializzare la variabile di tipo **`pixel`**, il cui indirizzo è dato come parametro, e restituire in uscita un valore intero, corrispondente alla media dei tre **`char`**.
5. Scrivere un programma che faccia uso della funzione **`init_pixel()`** per inizializzare due variabili **`p`** e **`q`**, di tipo **`pixel`**, in modo che **`p`** si trovi all'origine degli assi e sia colorato di verde (colori **`R`**=0, **`G`**=255, **`B`**=0), mentre **`q`** si trovi nel centro della griglia, e sia colorato di bianco (**`R`**=255, **`G`**=255, **`B`**=255).
6. Scrivere un programma che faccia uso della funzione **`init_rif_pixel()`** per *modificare* la variabile **`p`**, che si assume già inizializzata (punto 5), in modo che il pixel sia traslato di 10 unità nella direzione delle x e di 20 unità nella direzione delle y.
7. Definire una funzione **`scambia_pixel()`** che scambi il valore di due variabili di tipo **`pixel`**.
8. Definire una funzione **`init_vet_pixel()`** da usare per l'inizializzazione di un vettore di pixel. La funzione deve far sì che tutti gli elementi siano posizionati all'origine degli assi e abbiano colore nero (0, 0, 0).
9. Definire una matrice quadrata di pixel 10x10.
10. Definire una funzione **`init_mat_pixel()`** da usare per l'inizializzazione di una matrice quadrata di pixel, tipo quella definita al punto sopra. La funzione deve far sì che tutti gli elementi siano posizionati a intervalli di 10 unità nelle due direzioni (x e y) a partire dall'origine degli assi, e abbiano colore alternativamente bianco (255, 255, 255) e nero (0, 0, 0). Ad esempio, se il primo elemento è di colore nero, i suoi "vicini" saranno due elementi di colore bianco, di coordinate rispettivamente (0, 10) e (10, 0). I "vicini" dell'elemento bianco (20, 10) saranno quattro pixel neri, di coordinate (20±10, 10±10), vale a dire (20, 0), (20, 20), (10, 0), e (10, 20). *Vedi disegno (pagina sucessiva).*
11. Definire una funzione **`crea_pixel()`** che crea una variabile dinamica di tipo pixel, la inizializza mediante una delle funzioni di cui sopra (vale a dire, usando **`init_pixel()`** o **`init_rif_pixel()`**) e restituisce al chiamante l'indirizzo della variabile creata.



12. Definire una funzione **`crea_N_pixel()`** che abbia un unico parametro di tipo int, e restituisca in uscita un vettore di pixel.
13. Definire una funzione **`crea_N_ptr_pixel()`** che abbia un unico parametro di tipo int, e restituisca in uscita un vettore di puntatori a pixel, ciascuno inizializzato all'indirizzo di una variabile dinamica pixel creata all'interno della funzione.
14. Definire una funzione **`free_N_ptr_pixel()`** che possa essere utilizzata per liberare tutte le aree di memoria (heap) allocate tramite la corrispondente funzione **`crea_N_ptr_pixel()`** del punto sopra.
15. Definire una funzione **`area2()`** che, dati in ingresso due pixel, restituisca in un **`int`** l'area del rettangolo i cui lati passano per le coordinate dei due pixel.
16. Definire una funzione **`triangle()`** che, dati in ingresso tre pixel, restituisca in un **`int`** pari a 1 nel caso in cui le coordinate dei tre pixel rappresentano i vertici di un triangolo proprio (cioè, non si trovano sulla stessa retta).
17. Definire una funzione **`area3()`** che, dati in ingresso tre pixel, restituisca in un **`float`** l'area del triangolo i cui vertici si trovano sulle coordinate dei pixel, 0 nel caso in cui non si tratti di un triangolo proprio. Allo scopo, si utilizzi la funzione **`triangle()`** definita al punto precedente, ed eventualmente un'altra funzione, definita per determinare la lunghezza di ciascun lato.

    ***NOTA**:* Formula di Erone per determinare l'area A di un triangolo di lati a, b, c e semiperimetro s = ½(a+b+c): **A = radice del prodotto (s(s-a) (s-b) (s-c))**.
    La radice quadrata si può calcolare in C usando la funzione **`sqrt`**, definita in *math.h*, la quale ha la seguente interfaccia: **`double sqrt( double );`**

18. Qual è la differenza tra procedura e funzione?
19. Che cos'è l'interfaccia di una funzione? Di cosa si compone?
20. Quali sono le modalità di comunicazione tra cliente e servitore?
21. Quali sono le caratteristiche di una funzione in quanto servitore?
22. Cosa sono i parametri effettivi e che cosa sono i parametri formali?
23. Che cosa si intende per binding? che cos'è l'environment?
24. Cosa succede all'atto della chiamata di una funzione?
25. Quali sono le due modalità di passaggio dei parametri in C? Spiegare.
26. Come si realizza il passaggio per riferimento in C? Fare un esempio.
27. Quali sono i possibili tipi di dato di uscita di una funzione in C?
28. Spiegare che cosa è l'*heap* e come viene gestito (e da chi).
29. Che cosa si intende per "ambiente locale"/"ambiente globale"?
30. Qual è la visibilità di una variabile globale? Qual è il suo tempo di vita?
31. Che cos'è una variabile "static"? Che tempo di vita ha? Che visibilità ha?