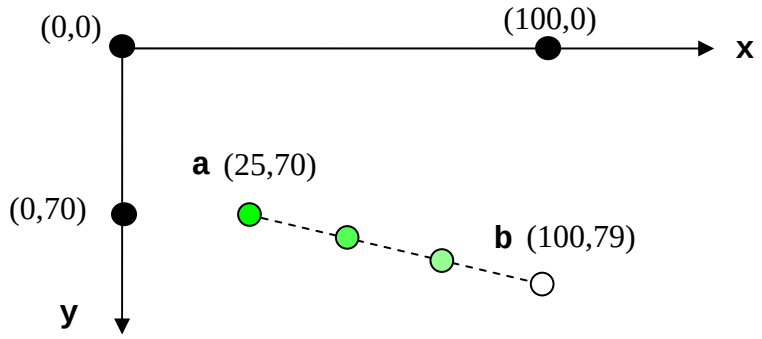
Paolo Torroni, *Fondamenti di Informatica L-A* (*Elettronica e Telecomunicazioni*) **14-15/11/2006**

1. Definire un nuovo tipo di dato **`pixel`**, di tipo record, composto dai seguenti campi: due campi **`int x`**, **`y`**, e un campo **`color`** di tipo record, composto da tre **`char`**: **`R`**, **`G`**, e **`B`**; i campi **`x`** e **`y`** rappresentano le coordinate di un pixel in una griglia 1024x768, mentre **`color`** viene utilizzato per rappresentarne il colore in termini di rosso, verde e blu.
2. Definire tre variabili, **`p`**, **`q`** ed **`r`**, di tipo **`pixel`**.
3. Inizializzare **`p`** e **`q`** 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).
4. Inizializzare **`r`** utilizzando dati forniti da input.
5. Fare in modo che **`r`** e **`q`** cambino di valore: in particolare, **`q`** deve assumere il colore di **`r`** e viceversa.
6. Definire tre variabili, **`a`**, **`b`** e **`v`**, le prime di tipo puntatore a vettore di **`pixel`**, la terza di tipo puntatore a vettore di puntatori a **`pixel`**; inizializzare tutti i puntatori al valore NULL.
7. Definire una variabile dinamica di tipo pixel, e fare in modo che **`a`** punti ad essa.
8. Modificare il valore dei puntatori: far sì che **`b`** punti alla variabile dinamica definita al punto precedente, e che **`a`** punti alla variabile **`p`**.
9. Definire una variabile dinamica di tipo *vettore di puntatori* a pixel, e fare in modo che **`v`** punti ad essa. La dimensione **`N`** del vettore deve essere un valore letto da input.
10. Scrivere un programma che crei **`N`** elementi di tipo pixel, che siano accessibili tramite **`v`**, e costruiti in modo che rappresentino punti compresi tra ***`a`** e ***`b`** e che siano a coppie equidistanti tra di loro: sia in termini di coordinate, sia in termini di colore. Ad esempio (vedi figura), se **`N`** = 2 ed **`a`**, **`b`** rappresentano i punti di coordinate rispettivamente (25,70) e (100, 79) e hanno come colori rispettivamente verde (0,255,0) e bianco (255,255,255), allora **`v[0]`** e **`v[1]`** devono puntare rispettivamente a un pixel di coordinate (50,73) e di colore (85,255,85) e a uno di coordinate (50,76) e di colore (170,255,170).



11. Definire un ciclo che modifica tutti i pixel di **`v`** (si assume che ce ne siano **`N`**), in modo che il colore di ciascun pixel sia diminuito dell'80%. Ad esempio, se un pixel ha come colore (25,10,200), il suo colore deve diventare (5,2,40).
12. Deallocare tutte le variabili dinamiche definite nei punti precedenti.

13. Si considerino le seguenti definizioni e istruzioni:

```
typedef int *type1;
typedef type1 *type2;
int i=200, j=300, k=400;
type1 x, y = &j, z = &k, *v;
type2 a = &x, *b = &a;
v = ( type2 )malloc( 10*sizeof( type1 ) );
x = ( type1 )malloc( 7*sizeof( int ) );
for(i=0;i<8;i++) {
     v[i]=&x[i];
     x[i]=100+i;
}
v[i++]=&i;
v[i++]=&j;
v[i]=&k;
```

    13.1. Quali sono le variabili in *aliasing*?
    13.2. Che valore ha **x[0]**?
    13.3. Che valore ha **x[7]**?
    13.4. Che valore ha **x[9]**?
    13.5. Che valore ha ***a[0]**?
    13.6. Di che tipo è ****b**?
    13.7. Di che tipo è *****b**?
    13.8. Dopo l'istruzione ***v[8]= *****b**, quanto valgono **x**, **y** e **z**?
14. L'eventuale esecuzione di una istruzione **free(x);**
    14.1. Crea delle *dangling reference*?
    14.2. Fa sì che nello *heap* rimangano delle aree inutilizzabili?
15. L'eventuale esecuzione di una istruzione **x=NULL;**
    15.1. Crea delle *dangling reference*?
    15.2. Fa sì che nello *heap* rimangano delle aree inutilizzabili?
16. L'eventuale esecuzione di una istruzione **a=NULL;**
    16.1. Crea delle *dangling reference*?
    16.2. Fa sì che nello *heap* rimangano delle aree inutilizzabili?
17. Date le definizioni del punto 13, qual è l'output del seguente programma?

```
while(--i) {
     printf( " %d", *v[i] );
}
```

18. Si liberino tutte le aree allocate dinamicamente al punto 13, in modo da non lasciare aree inutilizzabili nello *heap*.
19. Si considerino le seguenti definizionidi tipi:

```
typedef struct { int a,b,c; } type3;
typedef type3 *pt3;
```

    19.1. Si definisca una variabile dinamica di tipo vettore di 10 elementi di tipo **type3**, e la si renda accessibile tramite una variabile **w** di tipo **pt3**.
    19.2. Si inizializzi la struttura in modo che abbia tutti elementi nulli
    19.3. Si scriva un programma che, per ciascun elemento del vettore **w**, inizializzi il campo **c** alla media tra i campi **a** e **b**.
    19.4. Si liberi l'area allocata dinamicamente, in modo da non lasciare aree di memoria inutilizzabili nello *heap*.