```c
#include <stdio.h>

typedef struct {
    int x, y;
} punto;

// una struttura può essere un campo di un'altra struttura
// per accedere ad un suo campo si usa ancora "."
typedef struct {
    punto p1, p2;
} rettangolo;

int area(rettangolo rett)
{
    return (rett.p1.x - rett.p2.x) * (rett.p1.y - rett.p2.y);
}

int perimetro(rettangolo *rett)
{
    return 2 * ((rett->p1.x - rett->p2.x) + (rett->p1.y -
    rett->p2.y));
}

```

```c
int main(int argc, char *argv[])
{
    rettangolo rett = { .p1 = {50, 30}, .p2 = { 20, 10} };
    printf("L'area è %d\n", area(rett));
    printf("Il perimetro è %d\n", perimetro(&rett));
    return 0;
}

```

```
31  #if 0
32  Inizializzazione dei campi di strutture
33
34   * i valori contenuti nei campi di una struttura si possono
35     inizializzare in tre modi diversi:
36      - elencando i campi separati da virgola (sintassi tradizionale)
37      - dichiarando i campi con i due-punti (sintassi specifica del
38        compilatore GNU gcc fin da prima della standardizzazione)
39      - usando l'assegnamento ai campi (sintassi standard C99)
40
41
42  punto pt1 = { 10, 20 };
43  punto pt2 = { x: 10, y: 20 };
44  punto pt3 = { .x = 10, .y = 20 };
45
46  le tre strutture sono tutte inizializzate con gli stessi valori
47
48   * la prima forma è poco leggibile
49   * l'uso di questa forma rende anche molto problematica la modifica
50     di programmi esistenti
51      - l'aggiunta, l'eliminazione o lo spostamento di un campo cambia
52        l'ordine con cui i campi devono essere inizializzati,
53        creando non pochi problemi
54
```

```
55   * La seconda forma e la terza forma di inizializzazione non
56     soffrono dei precedenti problemi
57      - il nome di ciascun campo viene specificato esplicitamente
58      - non è legato all'ordinamento dei valori di inizializzazione
59   * La seconda forma non è standard
60      - è sconsigliata per problemi di compatibilità tra
61        compilatori diversi
62
63   * In tutti e tre i casi, ogni campo non esplicitamente
64     inizializzato viene azzerato bit-per-bit dal compilatore
65
66  #endif
67
```