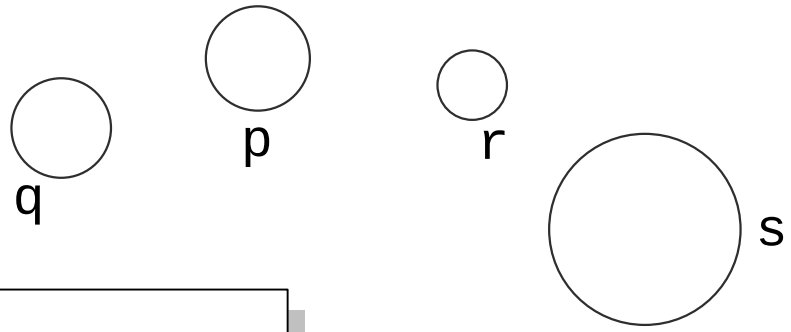
Structures




```
int     pX, pY, pR;
int     qX, qY, qR;
int     rX, rY, rR;
int     sX, sY, sR;
```

```
int             pX, qX, rX, sX;
int             pY, qY, rY, sY;
unsigned int    pR, qR, rR, sR;
```

Structures

- A collection of one or more variables grouped under a single name; each variable in a structure can be of different variable types.
- Accessing structure member is done by using its name with the structure member opeator (.) .

```
/* declare a structure named circle */
struct circle {
  int          cx, cy;
  unsigned int radius;
};
/* declare two instances of type circle */
struct circle  cir8, cir9;
...
cir8.cx = 2;
cir8.radius = cir9.radius * 1.5;
```

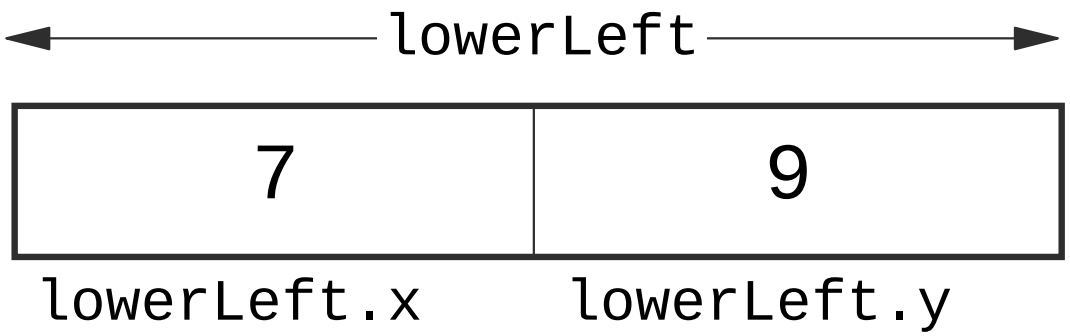
Structures

```
struct point {
  int  x;
  int  y;
} lowerLeft, upperRight;
```

```
struct point {
  int  x;
  int  y;
};
struct point lowerLeft, upperRight;
```

Structures

```
struct point {
  int  x;
  int  y;
};
struct point  lowerLeft;
...
lowerLeft.x = 7;
lowerLeft.y = 9;
...
```



Example : 2-D Distances

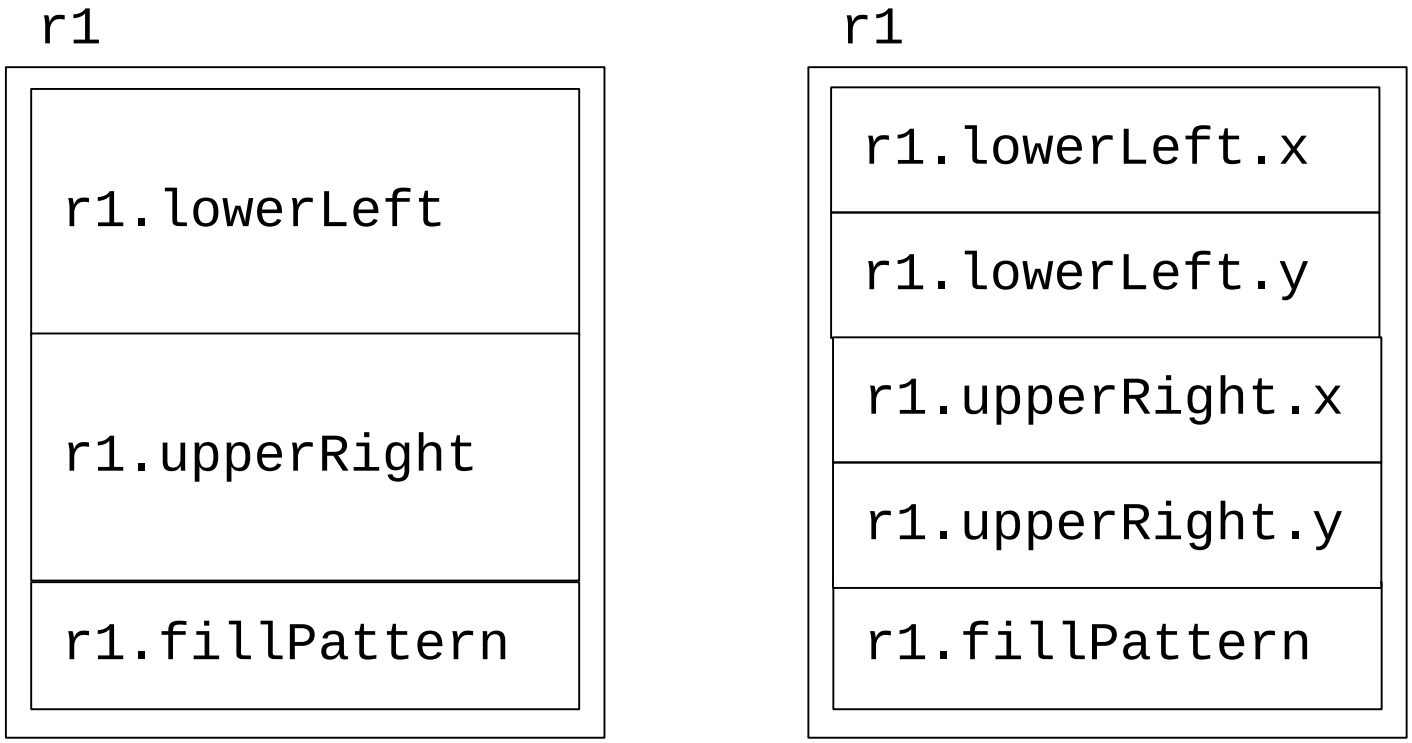
```
#include   <math.h>

struct point {
  int    x;
  int    y;
};

float EuclideanDistance( struct point p1,
                         struct point p2 )
{
  float     dx, dy;

  dx = p1.x - p2.x;
  dy = p1.y - p2.y;
  return ( sqrt( dx*dx + dy*dy ) );
}
```

$$d = \sqrt{\left(x_1 - x_2\right)^2 + \left(y_1 - y_2\right)^2}$$

Complex Structures

```
struct point {
  int  x;
  int  y;
};
struct rect {
  struct point lowerLeft;
  struct point upperRight;
  int          fillPattern;
};
```

Complex Structures



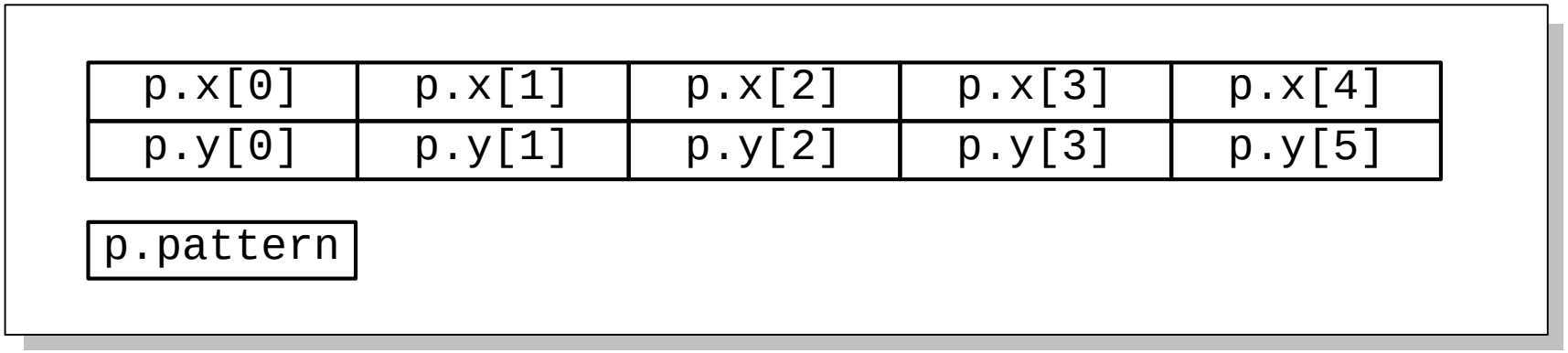

```
struct rect {
  struct point lowerLeft;
  struct point upperRight;
  int          fillPattern;
} r1;
```

Example : Calculating Area

```
#include <math.h>
#define  abs(x) ( (x) < 0 ? -(x) : (x) )

struct rect {
   struct point  lowerLeft;
   struct point  upperRight;
   int           fillPattern;
};

float RectArea( struct rect r )
{
  float     height, width;

  height = r.upperRight.y - r.lowerLeft.y;
  width  = r.upperRight.x - r.lowerLeft.x;
  return ( abs( height * width ) );
}
```

Structures that Contain Arrays

```
struct pentagon {
        int    x[5];
        int    y[5];
        int    pattern;
};
```

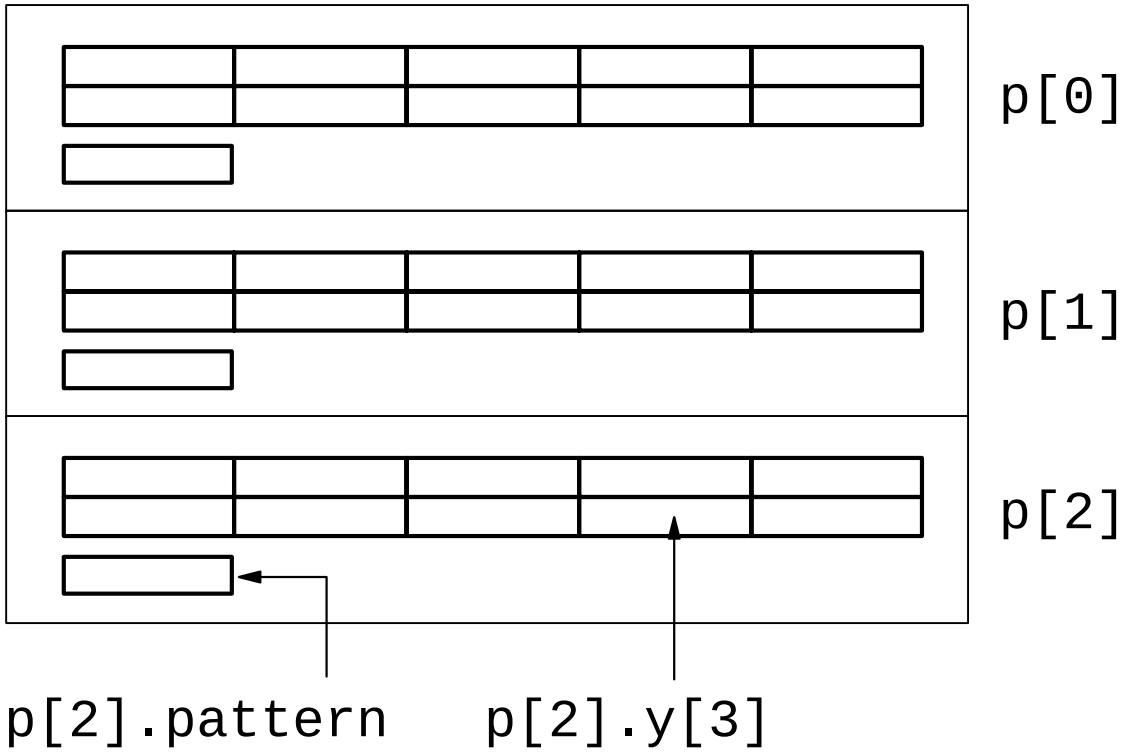



```
struct pentagon  p;
```

Array of Structures

```
struct pentagon {
        int   x[5];
        int   y[5];
        int   pattern;
};
```

```
struct pentagon  p[3];
```



Initializing Structures

```
struct sale {
        char   customer[20];
        char   item[20];
        float  amount;
};
struct sale mySale = { "Golden Hill",
                       "Chain saws",
                       1050.00
                     };
```

Initializing Structures

```
struct customer {
        char   fName[20];
        char   lName[30];
};
struct sale {
        struct customer buyer;
        char   item[20];
        float  amount;
};
struct sale mySale = { { "Vassana", "Heng" },
                       "Lip stick",
                       550.00
                     };
```

Can & Can't

```
struct customer {
        char   fName[20];
        char   lName[30];
} husband, wife;

...

husband = wife;         /*  OK   */

if ( husband == wife ) /* WRONG */
```

typedef

```
typedef struct point {
        int  x;
        int  y;
} PointType;

PointType vertex[20];
```

```
typedef  int   handle;
```

```
typedef struct rect {
        PointType lowerLeft;
        PointType upperRight;
        int       fillPattern;
} RectType;

RectType   rect1, box[25];
```

sizeof()

```
typedef struct rect {
        PointType lowerLeft;
        PointType upperRight;
        int       fillPattern;
} RectType;
```

```
sizeof( RectType ) == 2 * sizeof( PointType ) + sizeof( int )
```