

```
class Point {
```

```
public:
```

```
int x;
```

```
int y;
```

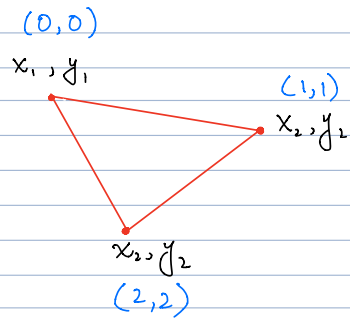
```
void print_point();
```

```
};
```

```
void Point::print_point() {
```

```
cout << "(" << x << ", " << y << ")" << endl;
```

```
}
```



Inheritance

```
class Shape {
```

```
public:
```

```
int
```

```
num_points;
```

```
// future
```

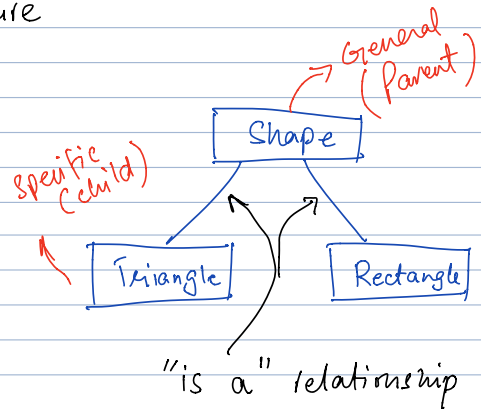
```
Point
```

```
* points;
```

```
float
```

```
get_area();
```

```
};
```



```
class Triangle : public Shape {
```

```
public:
```

```
float get_area();
```

```
// "overriding"
```

```
};
```

```
float Triangle::get_area() {
```

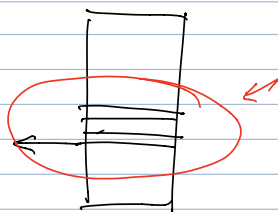
```
// computation ...
```

```
}
```

```

void Triangle::set-points ( Point *p ) {
    this->points = p;
}

```



```

void Triangle::print_shape() {
    Point *temp; temp = points;
    for (int i=0; i < num-points; i++) {
        temp->print-point();
        temp++;
    }
}

```

```

int main() {
    create_triangle();
}

```

```

void create_triangle() {

```

~~Triangle t;~~

```

Triangle *t; t = new Triangle;

```

```

// t->set-points(

```

```

Point p1, p2, p3;

```

```

p1.x = p1.y = 0;

```

```

p2.x = p2.y = 1;

```

```

p3.x = p3.y = 2;

```

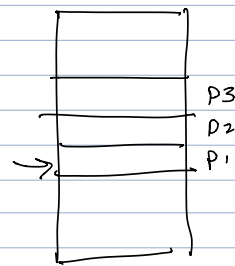
local variable

```

Point points[3];
points[0] = p1;
points[1] = p2;
points[2] = p3;

```

constant



points

Point *p ;
int size=3;

p = new

≡

p[0]
p[1]
p[2]

Point[size];

→ heap

variable on
because of
"dynamic allocation"