Leibniz Universität Hannover

Programmieren 1: Übungstest C-Teil

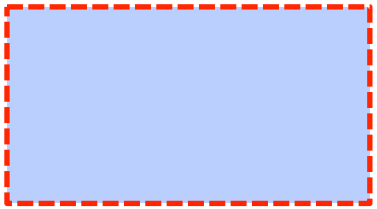
Human-Computer Interaction Group

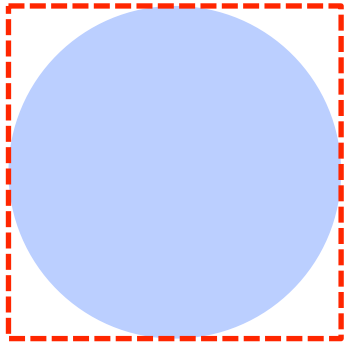
Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

Bounding Box Shape

Implementieren Sie die Funktion
`Rectangle bounding_box_shape(Shape *s)`.

Diese bekommt einen Zeiger auf eine Shape übergeben und soll das kleinste achsenparallele Rechteck berechnen, das die Shape enthält.





Bounding Box Shape

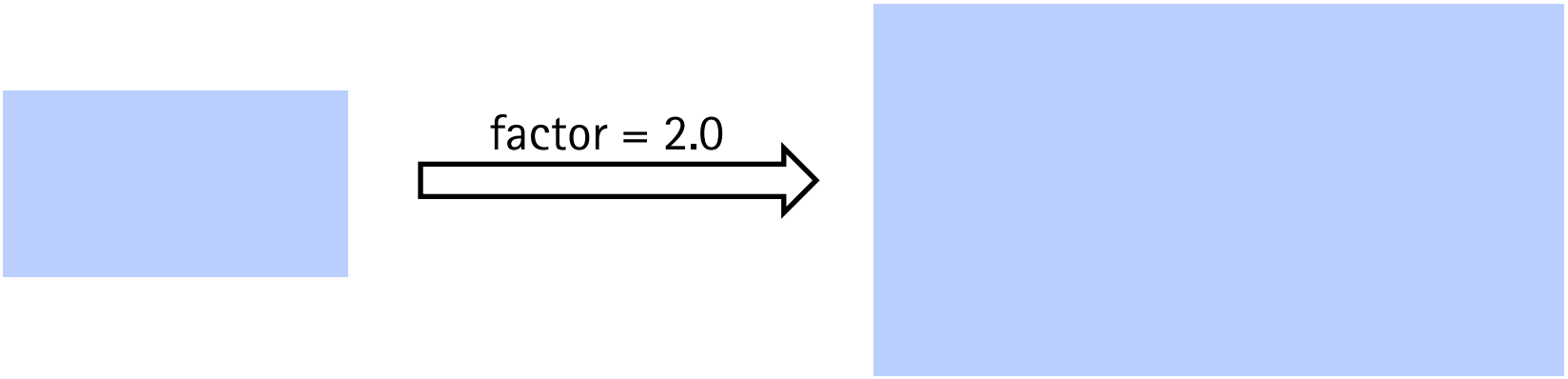
```
Rectangle bounding_box_shape(Shape *s) {
    int w;
    switch (s->tag) {
        case RECTANGLE:
            return s->r;
        case CIRCLE:
            w = 2 * s->c.r;
            return make_rectangle(s->c.x - s->c.r, s->c.y - s->c.r, w, w);
    }
}
```

Scale Shape

Implementieren Sie die Funktion

`void scale_shape(Shape *s, double factor).`

Diese bekommt einen Zeiger auf eine Shape übergeben. Die Funktion skaliert die Shape mit factor.



Scale Shape

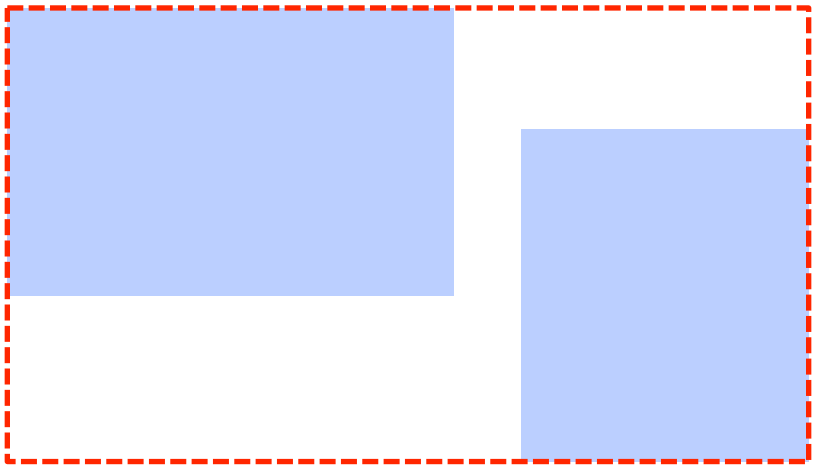
```
void scale_shape(Shape *s, double factor) {
    switch (s->tag) {
        case RECTANGLE:
            s->r.w *= factor;
            s->r.h *= factor;
            break;
        case CIRCLE :
            s->c.r *= factor;
            break;
    }
}
```

Bounding Box Rectangles

Implementieren Sie die Funktion

```
Rectangle bounding_box_rectangles(
                        Rectangle a, Rectangle b).
```

Diese bekommt zwei Rectangles übergeben und soll das kleinste achsenparallele Rechteck berechnen, das beide Rectangles enthält.



Bounding Box Rectangles

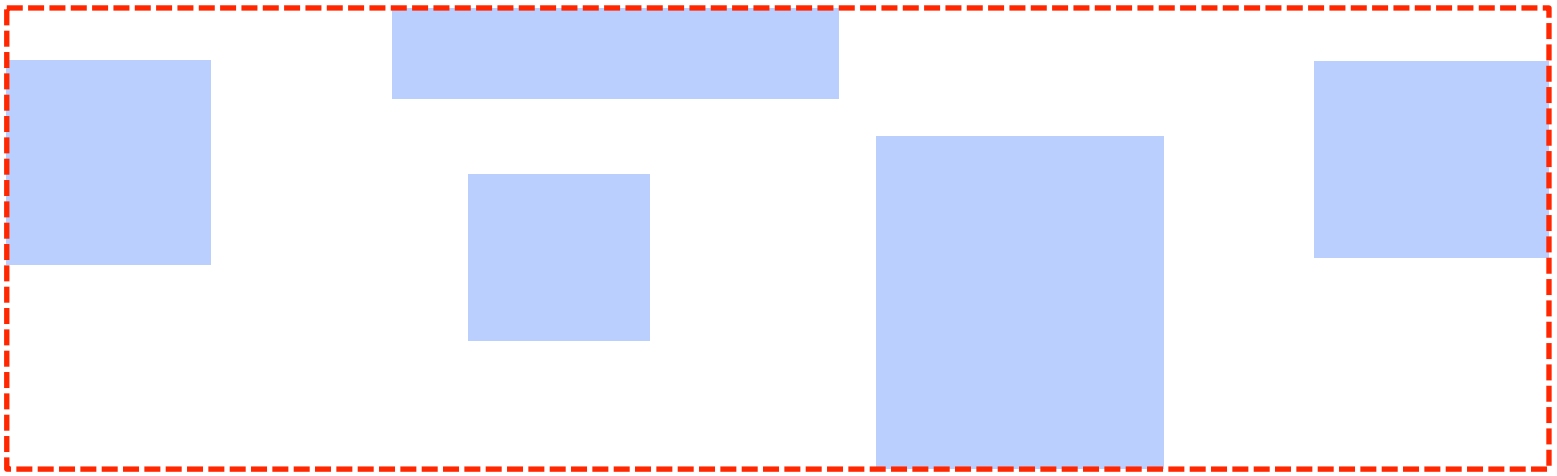
```
Rectangle bounding_box_rectangles(Rectangle a, Rectangle b) {
    int x = min(a.x, b.x);
    int y = min(a.y, b.y);
    int w = max(a.x + a.w, b.x + b.w) - x;
    int h = max(a.y + a.h, b.y + b.h) - y;
    return make_rectangle(x, y, w, h);
}
```

Bounding Box List

Implementieren Sie die Funktion

`Rectangle bounding box list(List list).`

Diese bekommt eine Liste übergeben, die als Elemente Zeiger auf Shapes enthält. Die Funktion soll das kleinste achsenparallele Rechteck berechnen, das alle Shapes der Liste enthält.



Bounding Box List

```
Rectangle bounding_box_list(List list) {
    ListIterator iter = l_iterator(list);
    if (!l_has_next(iter)) exit(EXIT_FAILURE); // empty list
    Shape *s = pl_next(&iter);
    Rectangle bb = bounding_box_shape(s);
    if (!l_has_next(iter)) return bb; // single-element list
    while (l_has_next(iter)) {
        Shape *s = pl_next(&iter);
        bb = bounding_box_rectangles(bb, bounding_box_shape(s));
    };
    return bb;
}
```