Pointers in General

- pointer references a location in memory
- get memory address with reference operator `&`
- access memory location with dereference operator `*`
- Pointer arithmetic depends on the pointer data type
- Precedence of the pointer operations depend on the location: `*p++ != *++p`
- data type `void` indicates the absence of type
- pointer of type `void*` has undetermined length and undetermined dereference properties

The Copy Assignment Operator

Copy assignment operator is called in object assignments (not in object initializations!)

```
Rectangle& operator=(const Rectangle& rect){
 _x = rect._x;
 _y = rect._y;
 return *this;
}
```

By deferencing the `this` pointer, we actually returning the object

Why do we have to specify the return value as `Rectangle&` ?

Call by Value/Reference

Call by Value

- The values of the variables are copied
- Changes are made to the copies

Call by Reference

- The references to the variables are passed
- Changes are made to the variables

Pointers and Arrays

Identifier of an array is equivalent to the address of its first element

```
void read(int numbers[]){
 // read data from the console
}

int numbers[2] = {0,0};
read(numbers);
```

Why is it possible write into the argument `number[]` , aren't we using a copy?