

# Copying Algorithms

## copy()

- `copy()` will copy elements from one container into another

```
vector<int> v {3, 1, 4, 1, 5, 9};
```

```
vector<int> v2(v.size());
```

```
copy(v.begin(), v.end(), v2.begin());
```

// Target container must be big enough!

- We can use a `back_insert_iterator` to populate an empty vector

```
vector<int> v3;
```

```
copy(v.begin(), v.end(), back_inserter(v3));
```

## copy\_if()

- `copy_if()` copies only the elements for which a predicate is true

```
copy_if(v.begin(), v.end(), back_inserter(v3),  
        [](int n) { return (n % 2 == 0); }           // Only copy even numbers  
);
```

## copy\_n()

- `copy_n()` will copy only the first n elements

```
vector<int> v {3, 1, 4, 1, 5, 9};
```

```
copy_n(v.begin(), 2, back_inserter(v3));
```

## transform()

- transform() will call a given function on every element in the range and store the result in a destination

```
// Double each element in v and store the results in v2
```

```
transform(v.begin(), v.end(), back_inserter(v2),
```

```
    [](int n) { return 2*n; }
```

```
);
```

```
// v2 has been assigned with elements which are double those in v
```

```
for (auto el: v2) {
```

```
    cout << el << endl;
```

```
}
```

## Overload of transform()

- There is an overload of transform() which takes an extra argument, a second source container
- This will call the function on every corresponding element from each source container
- The result is stored in the destination

```
// Add each element in v to the corresponding element in v2 and store the result in v3
transform(v.begin(), v.end(), v2.begin(), back_inserter(v3),
    [](int n1, int n2) { return n1 + n2; }
);
```