Memory management

Gianluca Campanella

Arrays

Arrays

Arrays...

- Are fixed-size lists of elements
- Can only store one type
- Are indexed starting at 0

```
1 int[5] x = {0, 1, 2, 3, 5};
2 x[0] = 1;
```

Arrays

- The size of arrays must be known at compile time
- Arrays (and other types) can also be allocated dynamically

```
1  int n = 5;
2  int* x = new int[n];
3
4  // Don't forget to free the memory!
5  delete[] x;
```

Memory leaks

Memory leaks…

- Occur when dynamically allocated memory is not released
- Can slow down the entire system significantly!
- Are particularly problematic for long-running applications

C++11 introduced smart pointers:

- `unique_ptr` cannot be copied
- `shared_ptr` can be copied

Pointers

Pointers

Pointers…

- Point to some memory location (static or dynamic)
- Are declared using the type of the ‘pointee’ followed by `*`
- Can point to anything using `void*`

```
1 int* x = new int;
```

Pointers

Dereference operator *

- ‘Follows’ a pointer to the pointee
- Make sure you know what’s on the other side!

```
1  int* x = new int;
2  *x = 5;
3
4  int* y;
5  *y = 5; // Boom!
```

Pointers

Address-of operator &

- Creates a new pointer to an existing variable
- Careful with static memory!

```
1  int x = 3;
2  int x_ptr = &x;
3
4  *x_ptr = 5;
```