C++ PROGRAMMING

DR. YIHSIANG LIOW (JULY 10, 2025)

Contents

68. Array of Objects

OBJECTIVES

- Review Array and pointer
- Use Arrays to store objects
- Build static sized Array Objects
- Build static sized Array Object Pointers
- Build dynamic sized Array Object
- Build dynamic sized Array Object Pointers

Review

As you know, the size of the array is fixed:

```
const int SIZE = 5;
int x[2 * SIZE];         // OK
int y[42];               // OK
int i = 5;
int z[i];                // WRONG!!!
```

For a "dynamic" array, i.e., an array whose size can be variable, you can use a pointer.

```
int * p = new int[5];          // OK
const int SIZE = 5;
int * q = new int [2 * SIZE]; // OK
int i = 5;
int * r = new int[i]; // OK!!!
```

Memory allocated to a pointer must be de-allocated by the programmer:

```
int i = 5;
int * r = new int[i];
// do something with r[0],...,r[i -- 1]
delete [] r;
```

Memory can be allocated to a pointer more than once. The sizes can be different.

```
int * r = new int[5];
// ... do something with r[0],...,r[4]
delete [] r;
...
r = new int;
// ... do something with *r;
delete r;
...
r = new int[10];
// ... do something with r[0],...,r[9]
delete [] r;
```

Like a pointer to a structure variable, if p points to an object with instance variable `x`, then

`(*p).x` is the same as `p-> x`

Similarly if the object has method `m()`, then invoking

`(*p).m()` is the same as `p-> m()`

Use `->` instead of `*.`

Array of Objects

So far objects are created individually.

What if you want to create an array of objects

```
C obj[10];
```

where `C` is the class?

Try the following example:

```
// Int.h
#include <iostream>
class Int
{
public:
        Int(int x=0)
           : x_(x)
        {
             std::cout << "Int ... x_: " << x_
                       << std::endl;
        }
private:
        int x_;
};
```

```
#include "Int.h"
int main()
{
    Int a[10];
    return 0;
}
```

The Point: Each `a[i] (i=0,...,9)` is constructed using the default constructor. When you declare an array of objects each object in the array calls the **default constructor**.

You **must** have a default constructor if you want to have an array of objects. (Don' t forget that if you don' t specify any constructor, C++ will give you a default constructor.)

Of course you can still call the constructor one at a time like this if you wish:

```
Int a[3] = Int(3), Int(-1), Int(42);
```

Now try to run this (why does it not work):

```
// vec2d.h
#include < iostream>
class vec2d
{
public:
        vec2d(double x, double y)
                : x_(x), y_(y)
                {}
private:
        double x_, y_;
};
```

```
#include "vec2d.h"

int main()
{
    vec2d v[10];
    return 0;
}
```

The point: Each of the `v[0],...,v[9]` calls `vec2d()` (the default constructor) to initialize itself. But there is no default constructor in `vec2d`. There' s only one constructor and it must receive two doubles.

Remember that! If you want to declare an array of objects without an initializer list, you **MUST** have the default constructor used to construct the objects in the array.

Exercise -1.0.1. Fix the following class so that an array of `WeatherControl` objects can be created with all instance variables initialized to 10.