Heap Memory – Allocating Arrays

```cpp
int *x;
int size = 3;
x = new int[size];
for (int i = 0; i < size; i++) {
x[i] = i + 3;
}
delete[] x;
```

*: `new[]` and `delete[]` are identical to `new` and `delete`, except the constructor/destructor are called on each object in the array.

---

Memory and Function Calls

Suppose we want to join two Cubes together:

```cpp
/*
 * Creates a new Cube that contains the exact volume
 * of the volume of the two input Cubes.
 */
Cube joinCubes(Cube c1, Cube c2) {
double totalVolume = c1.getVolume() + c2.getVolume();
double newLength = std::pow( totalVolume, 1.0/3.0 );
Cube result(newLength);
return result;
}
```

By default, arguments are “passed by value” to a function. This means that:

- *

---

Contrasting the three methods:

<table>
<thead>
<tr>
<th></th>
<th>By Value</th>
<th>By Pointer</th>
<th>By Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exactly what is copied when the function is invoked?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Does modification of the passed in object modify the caller’s object?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is there always a valid object passed in to the function?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Using the const keyword
1. Using const in function parameters:

Best Practice: “All parameters passed by reference must be labeled const.” – Google C++ Style Guide

2. Using const as part of a member functions’ declaration:

Returning from a function
Identical to passing into a function, we also have three choices on how memory is used when returning from a function:

Return by value:

Return by reference:

...remember: never return a reference to stack memory!

Return by pointer:

...remember: never return a reference to stack memory!

Copy Constructor
When a non-primitive variable is passed/returned by value, a copy must be made. As with a constructor, an automatic copy constructor is provided for you if you choose not to define one:

All copy constructors will:

The automatic copy constructor:

1.

2.

To define a custom copy constructor:

CS 225 – Things To Be Doing:

1. Exam 0 is ongoing
2. lab_debug due Sunday (11:59pm)
3. MP1 due Monday (11:59pm)
4. Daily POTDs every weekday