# 601.220 Intermediate Programming

Dynamic memory allocation

# C++ dynamic memory allocation

new and delete are essentially the C++ versions of malloc and free

Big difference: new not only allocates the memory, it also calls the appropriate constructor if used on a class type (more on this later)

Small difference: new and delete are "keywords" rather than functions, so you don't use (...) when calling them

## new usage

```cpp
// dynamic1.cpp:
#include <iostream>

using std::cout;
using std::endl;

int main() {
    int *iptr = new int;
    *iptr = 10;
    cout << "value of iptr " << iptr << endl;
    cout << "value in *iptr " << *iptr << endl;
    return 0;
}


$ g++ -c dynamic1.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o dynamic1 dynamic1.o
$ ./dynamic1
value of iptr 0x13bde70
value in *iptr 10
```

## delete usage

delete deletes something allocated with new

```cpp
// dynamic2.cpp:
#include <iostream>

using std::cout;
using std::endl;

int main() {
    int *iptr = new int;
    *iptr = 10;
    // do more with iptr
    delete iptr;
    cout << "after delete" << endl;
    cout << "value in *iptr " << *iptr << endl;
    cout << "value of iptr " << iptr << endl;
    // note: new and delete don't use parentheses,
    // unlike malloc() / free()
    return 0;
}
```

# C++ delete usage

```
$ g++ -c dynamic2.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o dynamic2 dynamic2.o
$ ./dynamic2
after delete
value in *iptr 0
value of iptr 0x1604e70
```

## C++ dynamic array allocation

T * fresh = new T[n] allocates an array of n elements of type T

Use delete[] fresh to deallocate – always use delete[] (not delete) to deallocate a pointer returned by new T[n]

If T is a "built-in" type (int, float, char, etc), then the values are *not initialized*, like with malloc

If T is a class, then T's default constructor is called for *each* element allocated (more on this soon)

# C++ dynamic array allocation in action

```cpp
// dynamic3.cpp:
#include <iostream>

using std::cout;
using std::endl;

int main() {
    double *d_array = new double[10];
    for(int i = 0; i < 10; i++) {
        cout << (d_array[i] = i * 2) << " ";
    }
    cout << endl;
    delete[] d_array;
    return 0;
}
```

```
$ g++ -c dynamic3.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o dynamic3 dynamic3.o
$ ./dynamic3
0 2 4 6 8 10 12 14 16 18
```