CS 485
Advanced Object Oriented Design

Dynamic Memory

Spring 2017

pointers! new/delete

- Dynamic memory
  - What does that mean?
  - Why do we want it?
  - Where does it come from?
  - How do we get it?

Don’t use malloc/free in C++

- Design Techniques
  - RAII
    - resource acquisition is initialization
      - constructor/destructor
  - CopyAndSwap

const

```
const int * pData;

int const * pData;

int * const pData;

const int * const pData;

int const * const *pData;
```

const applies to the type to the immediate left, however you can put const as the left most token and then it applies to the immediate right.

http://c-faq.com/decl/spiral.anderson.html

NULL vs nullptr

- NULL is really an int
- nullptr is its own type (std::nullptr_t)

```
void foo(int *ptr);
void foo(int data);

foo(NULL); // who gets called?
foo(nullptr); // who gets called?


auto ptr = NULL;
auto ptr2 = nullptr;

cout << typeid(ptr).name();
cout << typeid(ptr2).name();
```

various new C++11 std:: functions take only nullptr

legacy C code may not work with nullptr!

Classes that contain dynamic memory

- Or any dynamic resource (file, network, …)

```
class bigData
{
  public:

    bigData();
    ~bigData();
    bigData(const bigData &rcData);

    // bigData& operator=(const bigData &rcData);

    // you cannot declare both operator= at the same time!
    bigData& operator=(bigData cData); //force copy constructor to be called


  private:
    int *mpHugeData = nullptr;
}
```

copy-and-swap

```
bigData::bigData(const bigData &rcData)
{
 // let’s assume this is written and works correctly
}


bigData& bigData::operator=(bigData cData)
{
  // what happens when cBigData1 = cBigData2; is executed?









  // what happens when operator= terminates?

}
```

Smart Pointers

- C++11 wrapper classes to manage pointers
  - RAII for pointers
  - <memory>
- unique_ptr
- shared_ptr
- weak_ptr

https://msdn.microsoft.com/en-us/library/hh279674.aspx

http://umich.edu/~eecs381/handouts/C++11_smart_ptrs.pdf

Lab - use raw pointers

- You must write your own string class, backed by a dynamic char array.
- The string class must implement the interface provided.
  - PacString.h
- Use the provided test driver. main.cpp
  - make sure there are no memory leaks
  - determine how many times each of the following is called by the test driver:
    - default constructor
    - constructor (const char *)
    - copy constructor
    - destructor