# C++ Smart Pointers

## C++ Managed Pointers

Pearson

## Traditional C++ Raw Pointers

```
#include <iostream>
#include <string>
#include "PlainBox.h"
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Risky to Use**
- **Can result in memory leaks when an object in the free store**
  - is not deleted after use
  - exists but no variable references it
- **Can result in dangling pointers**
  - an object is deleted, but a pointer still holds a reference to it
  - a pointer variable is created, but

# C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Managed Pointers**
  - Provides automatic memory management of objects
- **shared_ptr (Shared Pointer)**
  - Provides shared ownership of an object
- **unique_ptr (Unique Pointer)**
  - Provides unique ownership of an object
- **weak_ptr (Weak Pointer)**
  - Provides a “weak,” or non-owning, reference to an object that is already managed by a shared

# C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Managed Pointers**
  - Provides automatic memory management of objects
- **shared_ptr (Shared Pointer)**
  - Provides shared ownership of an object

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  std::shared_ptr< PlainBox<std::string> > giftBox(new PlainBox<std::string>("Ring"));
  std::cout << giftBox->getItem() << std::endl;
}
```

**std::shared_ptr< *sharedPointerObjectType* > *pointerName*(new *objectType*);**

Pearson

# C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Managed Pointers**
  - Provides automatic memory management of objects
- **shared_ptr (Shared Pointer)**
  - Provides shared ownership of an object

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  std::shared_ptr< PlainBox<std::string> > giftBox = std::make_shared< PlainBox<std::string> >("Ring");
  std::cout << giftBox->getItem() << std::endl;
}
```

```
std::shared_ptr< sharedPointerObjectType > pointerName = std::make_shared< sharedPointerObjectType >(ContructorParameterList);
```

Pearson

# C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Managed Pointers**
  - Provides automatic memory management of objects
- **shared_ptr (Shared Pointer)**
  - Provides shared ownership of an object

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  auto giftBox = std::make_shared< PlainBox<std::string> >("Ring");
  std::cout << giftBox->getItem() << std::endl;
}
```

```
auto pointerName = std::make_shared< sharedPointerObjectType >(ConstructorParameterList);
```

## C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  PlainBox<std::string>* giftBox = new PlainBox<std::string>("Ring");
  std::cout << giftBox->getItem() << std::endl;
  delete giftBox;
  giftBox = nullptr;
}
```

- **Managed Pointers**
  - Provides automatic memory management of objects
- **shared_ptr (Shared Pointer)**
  - Provides shared ownership of an object

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  auto giftBox = std::make_shared< PlainBox<std::string> >("Ring");
  std::cout << giftBox->getItem() << std::endl;

  auto otherPtr = giftBox;
  std::cout << otherPtr->getItem() << std::endl;
}
```

# C++ Smart Pointers

```
#include <iostream>
#include <string>
#include “PlainBox.h”
void someUsefulFunction()
{
  auto giftBox = std::make_shared< PlainBox<std::std::string> >("Ring");
  std::cout << giftBox->getItem() << std::endl;

  auto otherPtr = giftBox;
  std::cout << otherPtr->getItem() << std::endl;

  int howManyReferences = otherPtr.use_count();

  giftBox.reset();  // Same as giftBox = nullptr;

  PlainBox<std::std::string>* myRawPtr = otherPtr.get();  // Use only if legacy code needs raw pointer!!!

}
```