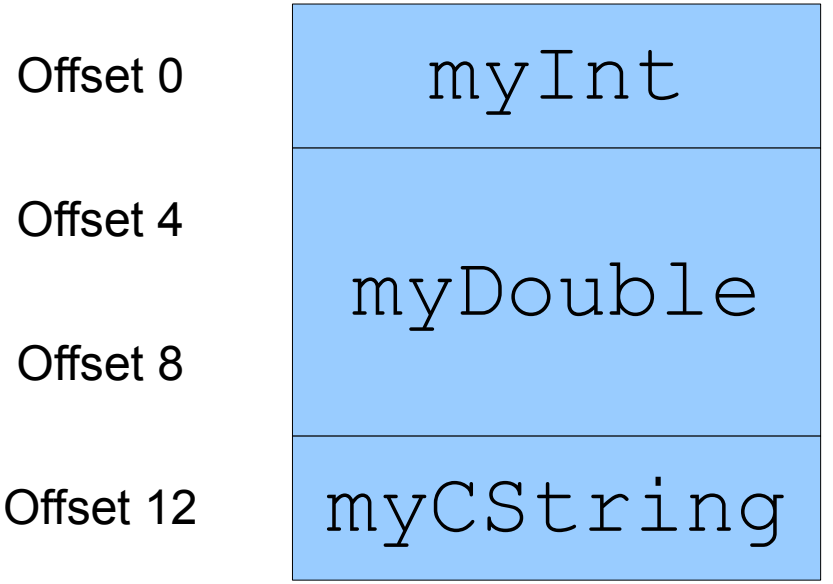
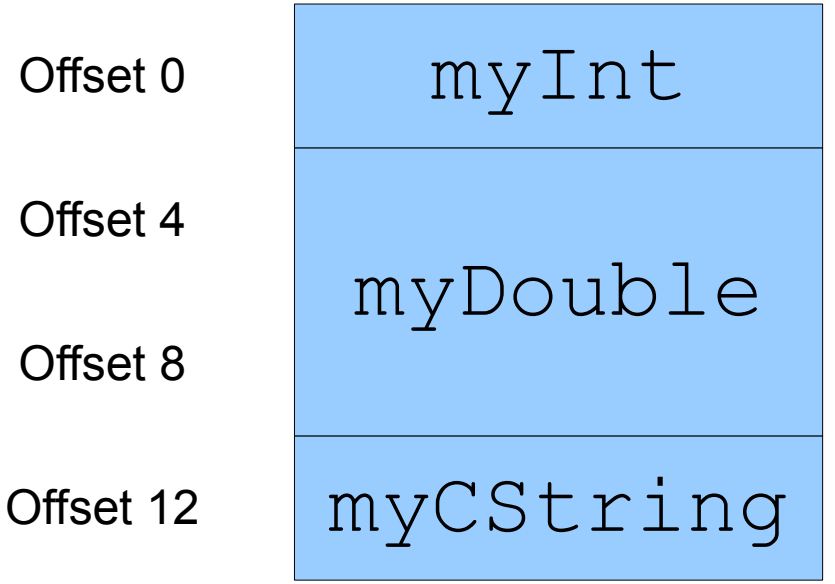
C++ Class Implementation

```
struct MyStruct
{
    int myInt;
    double myDouble;
    char *myCString
};
```

```
struct MyStruct
{
    int myInt;
    double myDouble;
    char *myCString
};
```

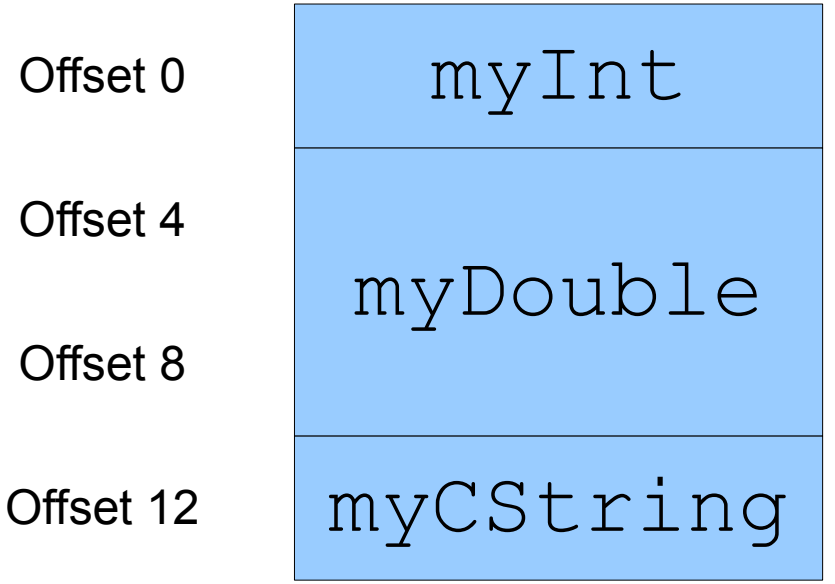


```
struct MyStruct
{
    int myInt;
    double myDouble;
    char *myCString
};
```




```
MyStruct* mc = new MyStruct;
mc->myCString = "This is a string!"
```

```
struct MyStruct
{
    int myInt;
    double myDouble;
    char *myCString
};
```



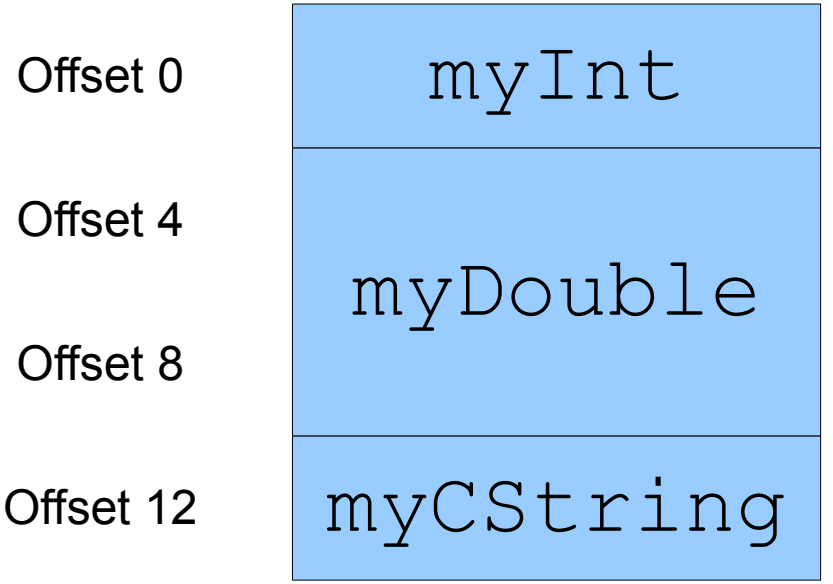

```
MyStruct* mc = new MyStruct;
mc->myCString = "This is a string!"
```

Implementation

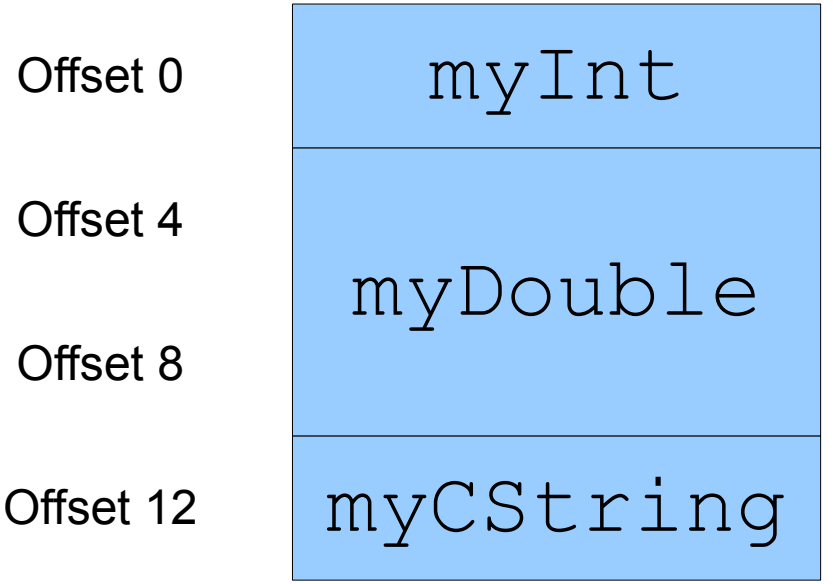
1. Go to the location pointed at by mc.
2. Go forward 12 bytes.
3. Store a pointer to "This is a string!"

```
class MyClass
{
public:
    int myInt;
    double myDouble;
    char *myCString
};
```

```
class MyClass
{
public:
    int myInt;
    double myDouble;
    char *myCString
};
```

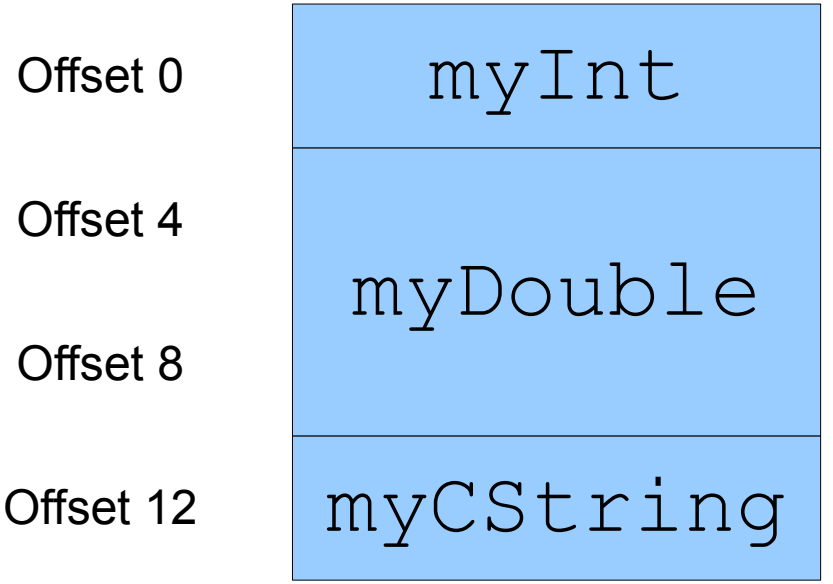


```
class MyClass
{
public:
    int myInt;
    double myDouble;
    char *myCString
};
```




```
MyClass* mc = new MyClass;
mc->myCString = "This is a string!"
```

```
class MyClass
{
public:
    int myInt;
    double myDouble;
    char *myCString
};
```



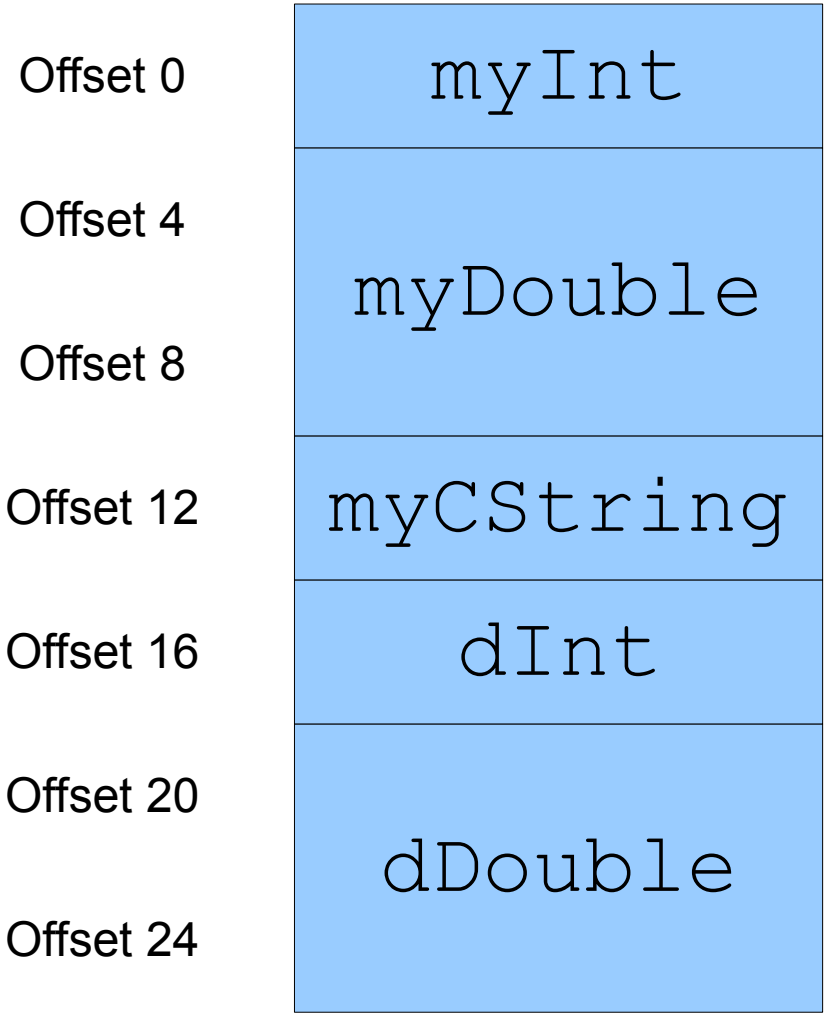

```
MyClass* mc = new MyClass;
mc->myCString = "This is a string!"
```

Implementation

1. Go to the location pointed at by mc.
2. Go forward 12 bytes.
3. Store a pointer to "This is a string!"

```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};
```

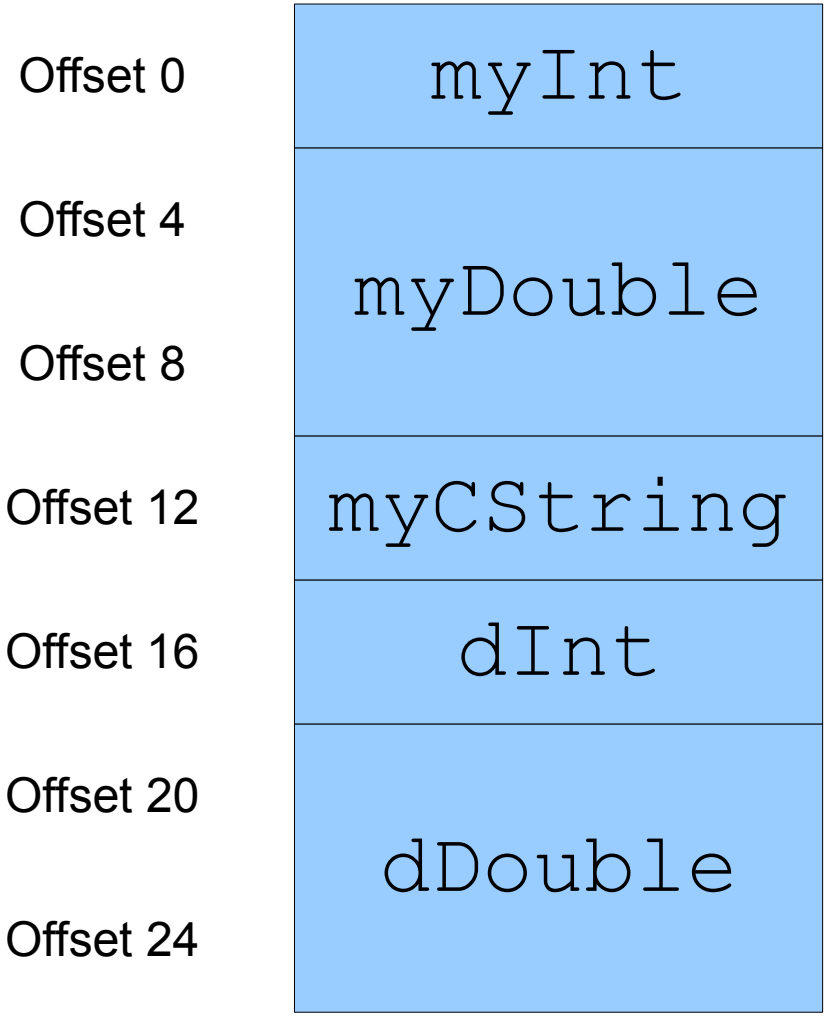
```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};
```



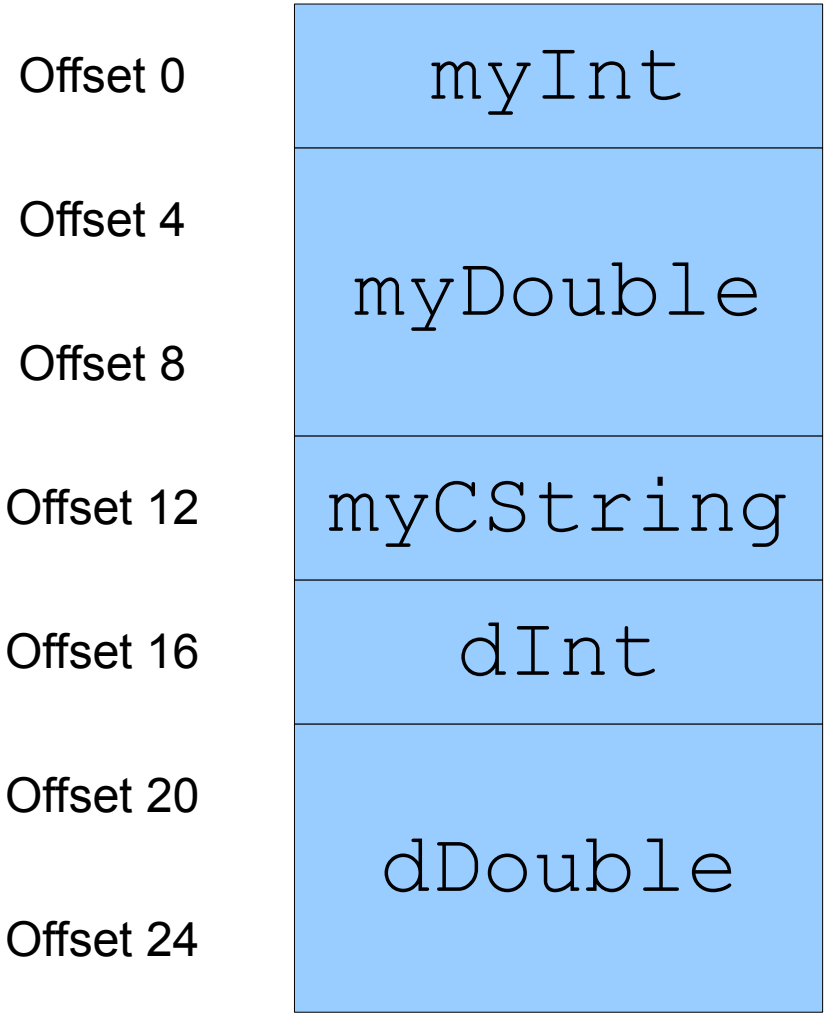
```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};
```

```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};
```

```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};
```

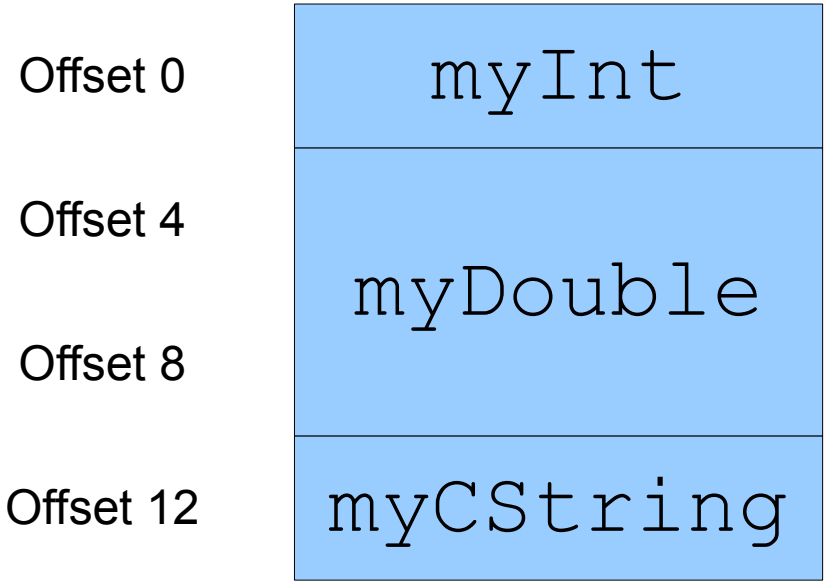


```
class Base
{
public:
    int myInt;
    double myDouble;
    char *myCString
};

class Derived: public Base
{
public:
    int dInt;
    double dDouble;
};

Base* mc = new Derived;
mc->myCString = "This is a string!"
```

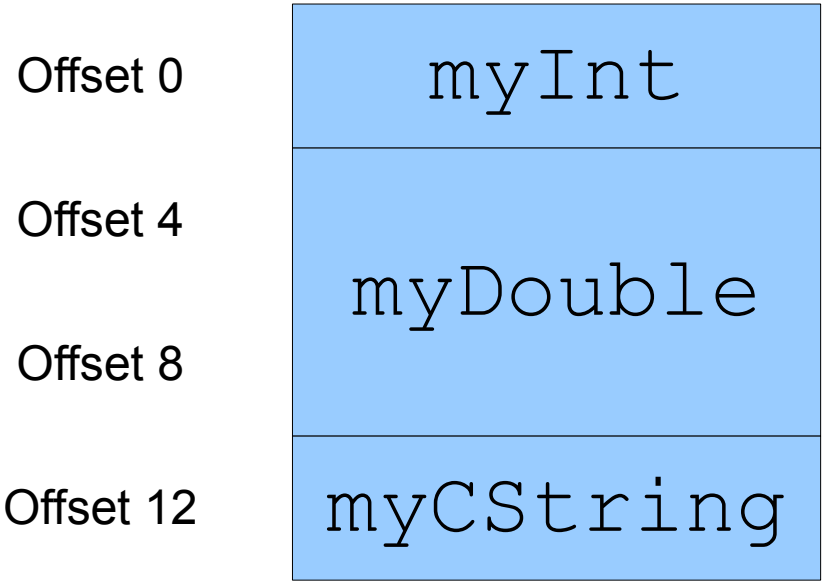


```
class MyClass
{
public:
    void myFn(int myInt);
private:
    int myInt;
    double myDouble;
    char *myCString
};
```

```
class MyClass
{
public:
    void myFn(int myInt);
private:
    int myInt;
    double myDouble;
    char *myCString
};
```



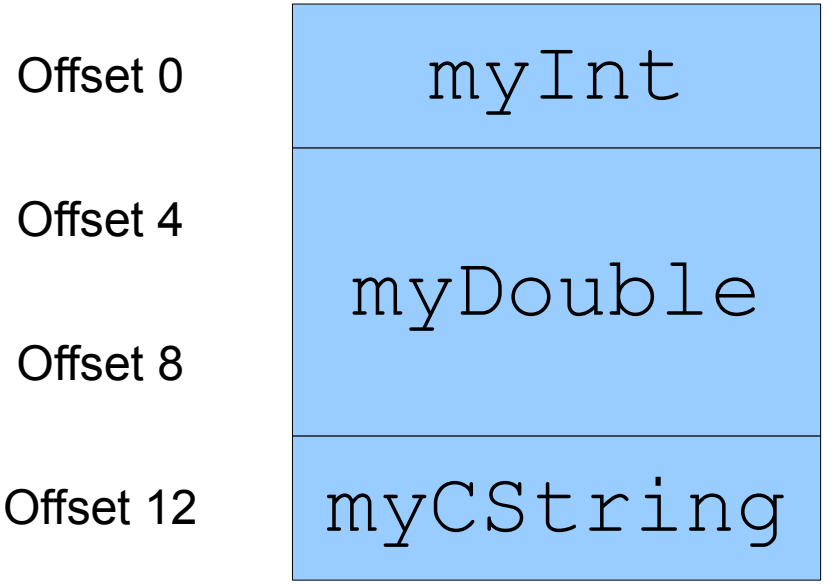
```
class MyClass
{
public:
    void myFn(int myInt);
private:
    int myInt;
    double myDouble;
    char *myCString
};

MyClass* mc = new MyClass;
mc->myFn(137);
```



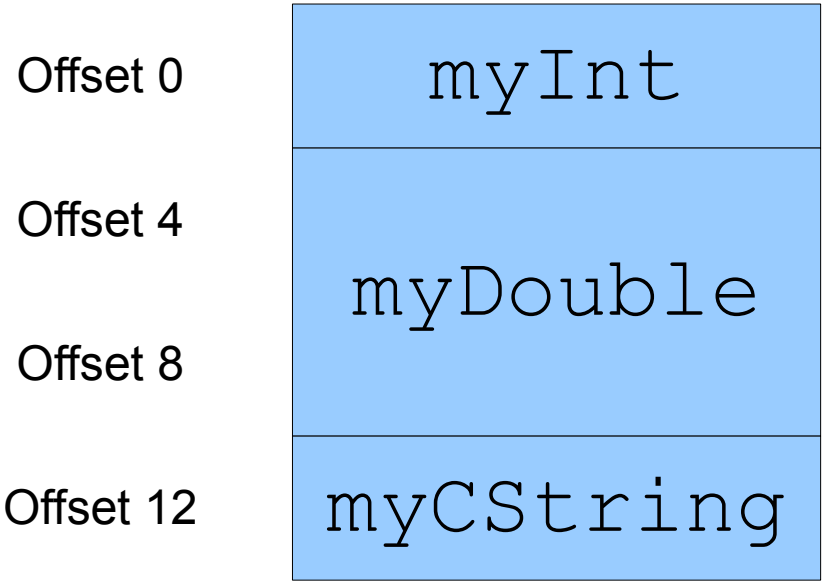
```
class MyClass
{
public:
    void myFn(int myInt);
private:
    int myInt;
    double myDouble;
    char *myCString
};

MyClass* mc = new MyClass;
mc->myFn(137);
```

Implementation

```
myFn(mc, 137);
```



```
class MyClass
{
public:
    void myFn(int myInt);
private:
    int myInt;
    double myDouble;
    char *myCString
};

MyClass* mc = new MyClass;
mc->myFn(137);
```

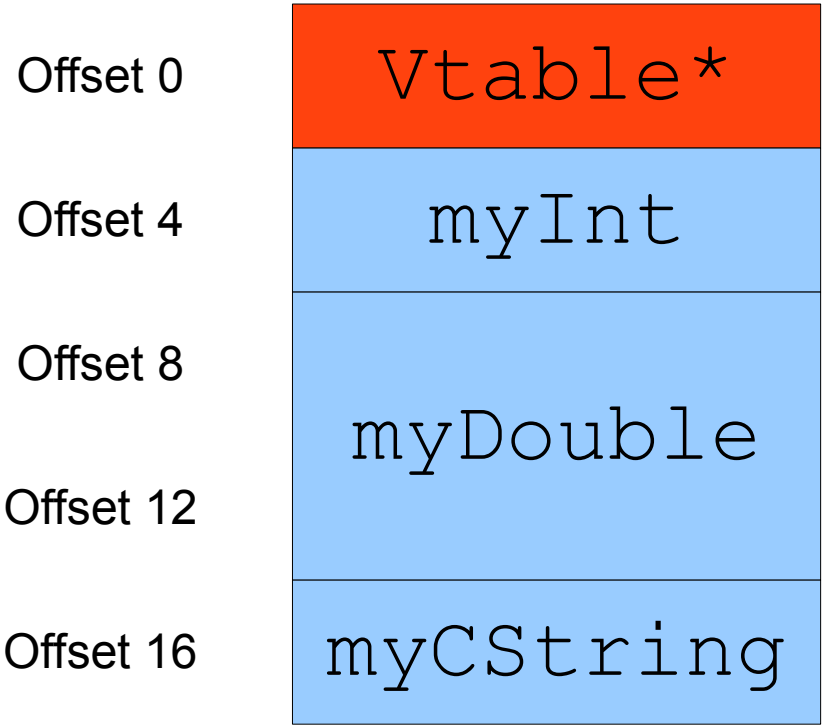


Implementation

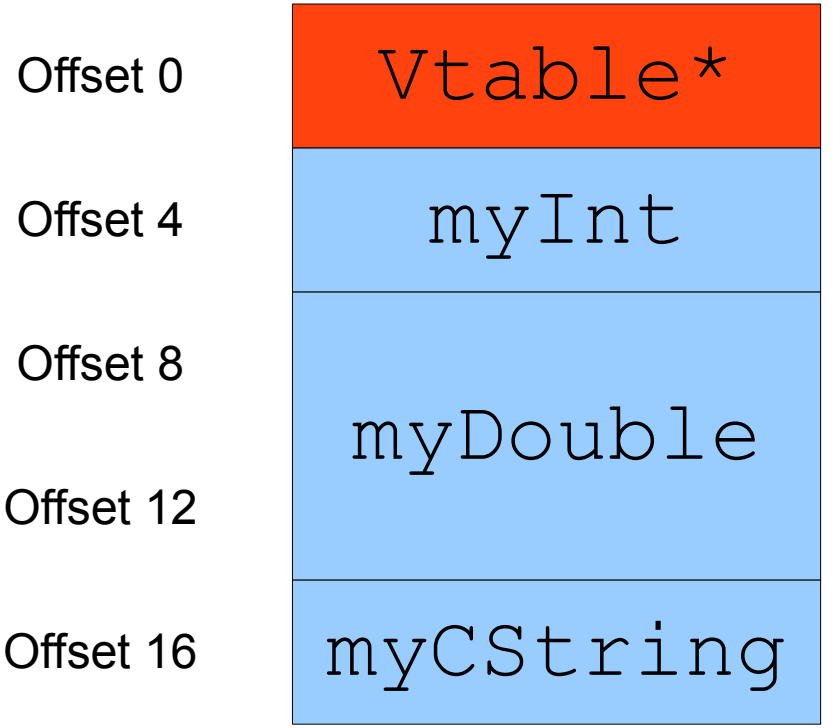
```
myFn(mc, 137); // Just a regular function call!
```

```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};
```

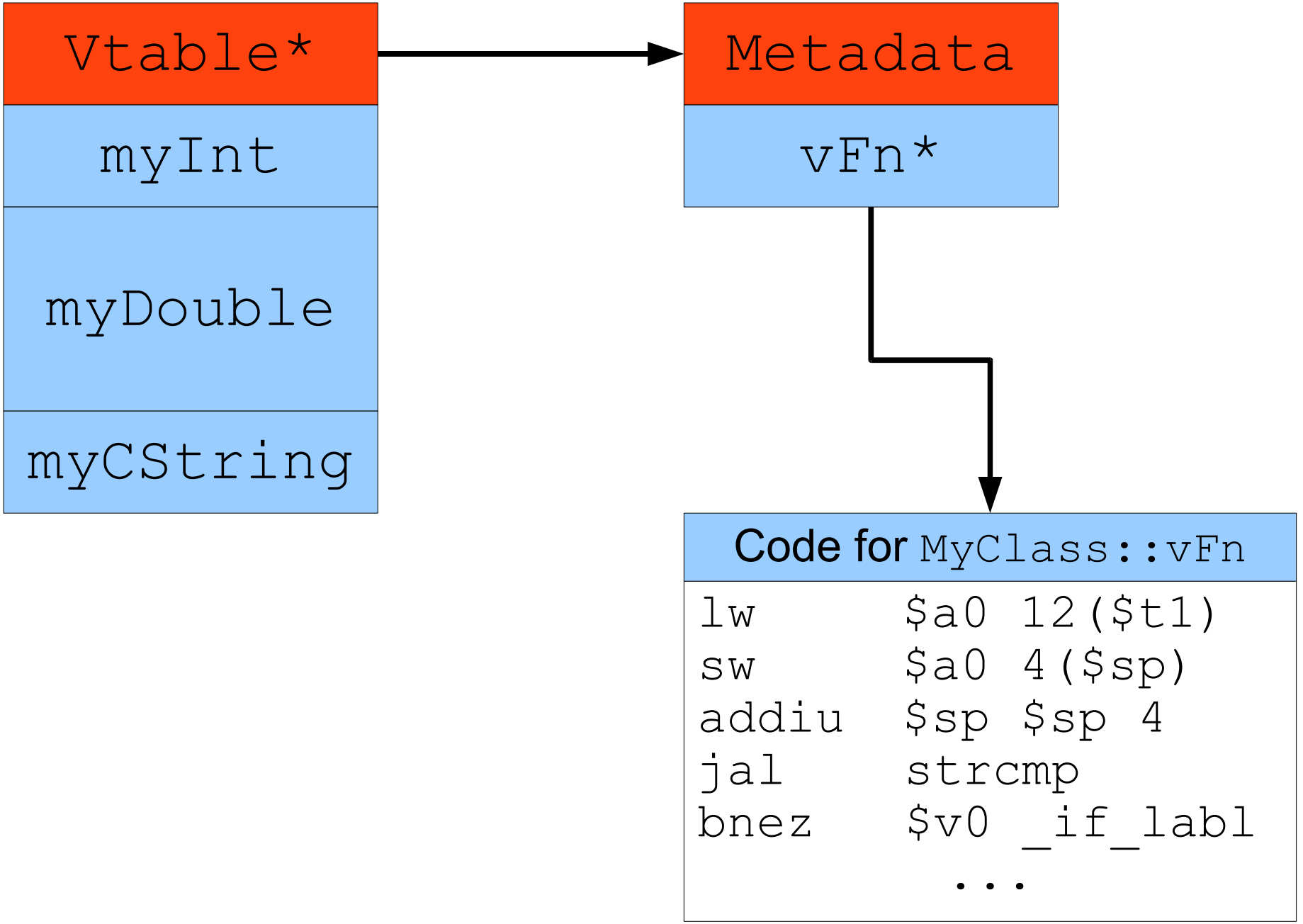
```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};
```

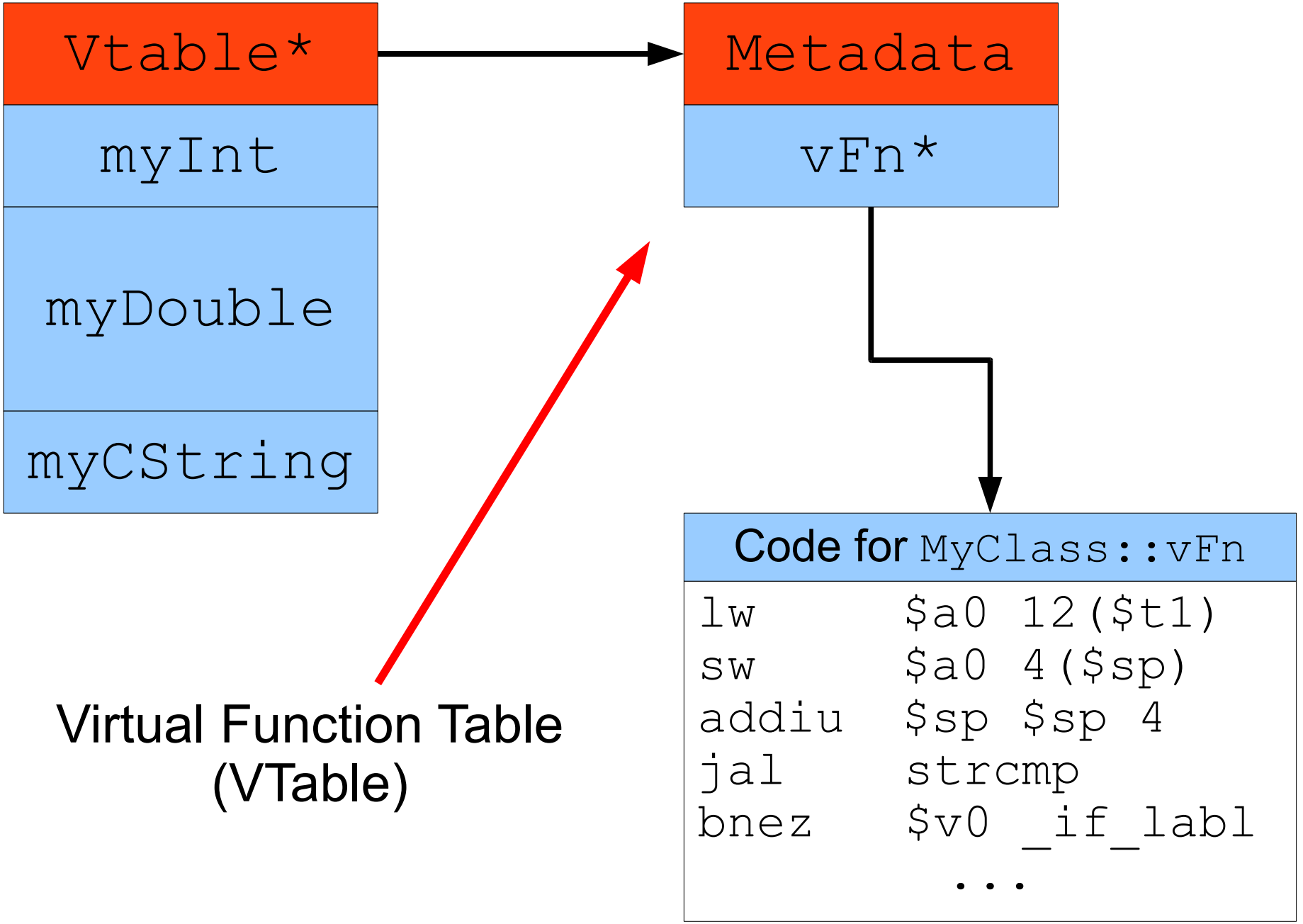


```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};
```

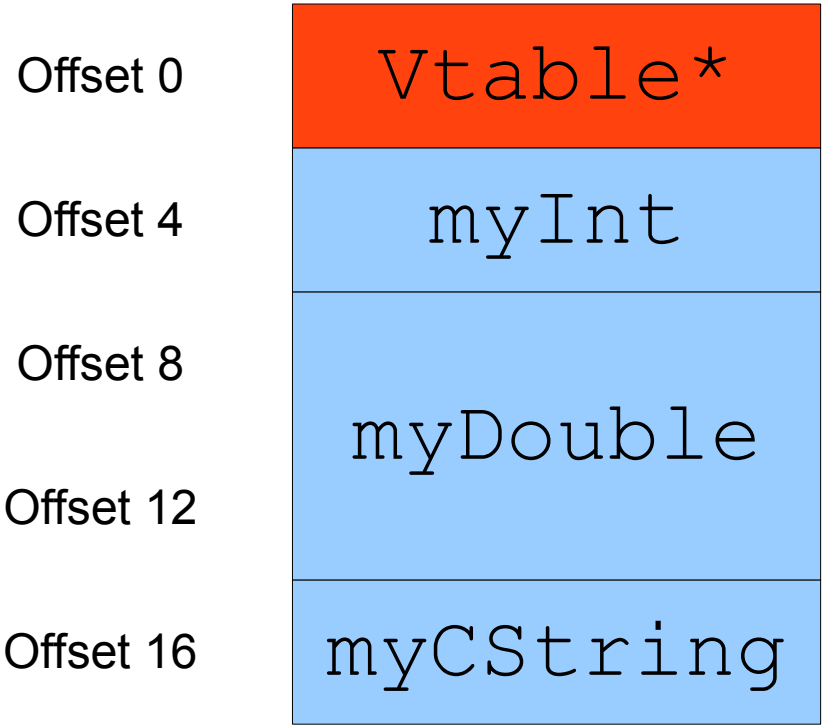


What's this Vtable* thing?

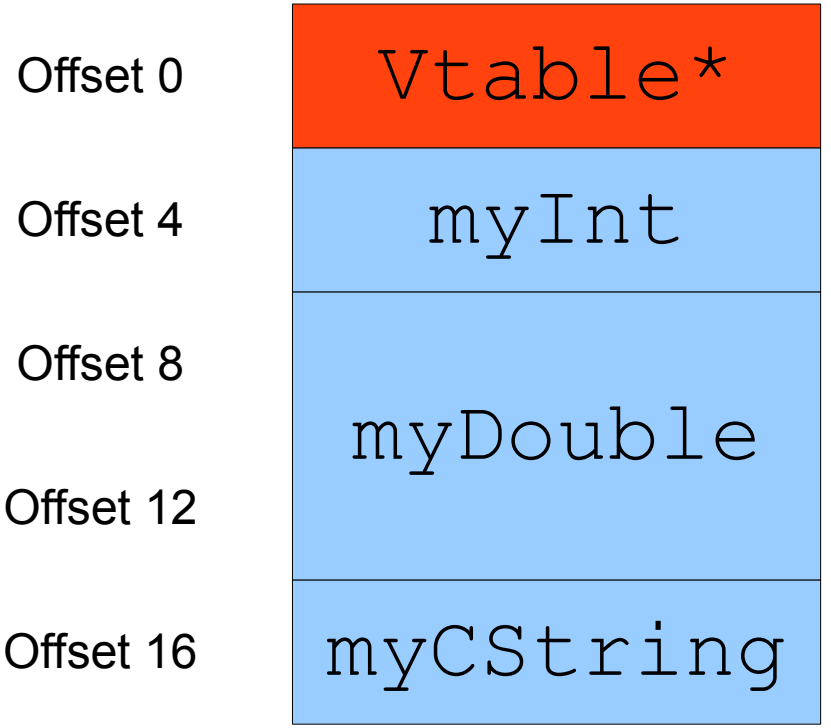

Vtable*
myInt
myDouble
myCString
Metadata
vFn*
Code for MyClass::vFn
lw $a0 12($t1)
sw $a0 4($sp)
addiu $sp $sp 4
jal strcmp
bnez $v0 _if_labl
...


Vtable*
myInt
myDouble
myCString
Metadata
vFn*
Code for MyClass::vFn
lw $a0 12($t1)
sw $a0 4($sp)
addiu $sp $sp 4
jal strcmp
bnez $v0 _if_labl
...
Virtual Function Table
(VTable)

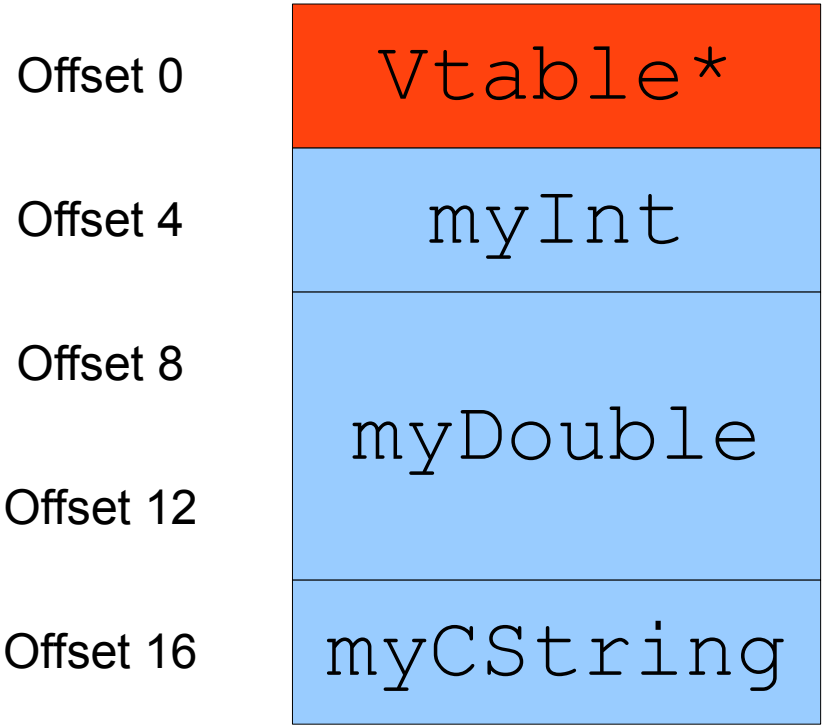
```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};
```



```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};

MyClass* mc = new MyClass;
mc->vFn();
```



```
class MyClass
{
public:
    void myFn(int myInt);
    virtual void vFn();
private:
    int myInt;
    double myDouble;
    char *myCString
};

MyClass* mc = new MyClass;
mc->vFn();
```



Implementation

1. Go to the object pointed at by mc.
2. Go to the vtable.
3. Look up which version of **vFn** to call.
4. Call **vFn(mc)**