Literate Object Oriented Library.

Brandon Koepke

March 24, 2016

Contents

Abstract

The following is a simple object library for C. An object has a class name, a to_string method, and a free method. We also included default implementations of to_string and free that will be suitable for most implementations.

1 Class Definition

We start by defining a basic class structure. We are forward declaring our *Class* to ensure that the implementation details can be changed at a later date.

1a ⟨*Class Declaration* 1a⟩≡
```
typedef struct _Class Class;
```

The class vtable contains a single operation, the ability to get the class name. The vtable should also be considered opaque to other objects, manually manipulating this from outside a class (including invoking it) is considered undefined behavior. In addition, by defining the name as a function as opposed to including a string reference enables us to change the implementation in child classes.

1b ⟨*Default Class Functions* 1b⟩≡
```
typedef struct {
  const char * const (*name)(const Class * const);
} class_vtable;
```

To get the class name safely, use the following method. The class name implementation invokes the name method from the vtable on the class object itself.

1c ⟨*Class Name Declaration* 1c⟩≡
```
const char * const class_name(const Class * const);
```

1d ⟨*Class Name Implementation* 1d⟩≡
```
const char * const class_name(const Class * const c) { return c->vtable->name(c); }
```

The class implementation only contains the class vtable, everything else is deferred to implementers.

1e ⟨*Class Implementation* 1e⟩≡
```
struct _Class {
  class_vtable *vtable;
};
```

Finally we have the completed class declaration and implementation.

1f ⟨*class.h* 1f⟩≡
```
#pragma once
```
⟨*Class Declaration* 1a⟩
⟨*Default Class Functions* 1b⟩
⟨*Class Name Declaration* 1c⟩

2 ⟨*class.c* 2⟩≡
```
#include "class.h"
⟨Class Implementation 1e⟩
⟨Class Name Implementation 1d⟩
```

2 Object Definition

An object inherits all the abilities of a class. In addition, an object can also have it's memory freed and can be converted to a string representation. Here we are hiding the implementation of the object entirely, but for implementers of this object library yuou must include a pointer to the vtable as the first part of your object structure.

3a ⟨*Object Declaration* 3a⟩≡
```
typedef struct _Object Object;
typedef struct {
  class_vtable class;
  const char * const (*to_string)(const Object * const);
  void (*free)(Object *);
} object_vtable;
void object_free(Object *o);
const char * const object_to_string(const Object * const o);
```

3b ⟨*Object Implementation* 3b⟩≡
```
struct _Object {
  object_vtable *vtable;
};
```

We also have default implementations of *free* and *to_string*. These are not intended to be called directly, rather these can be set as your function implementations on your subclass.

3c ⟨*Default Object Method Declarations* 3c⟩≡
```
void _object_free(Object *o);
const char * const _object_to_string(const Object * const o);
```

We now define the interface method for *object_free* where we invoke the vtable implementation and also the default implementation which runs the stdlib free.

3d ⟨*Default Object Free Implementation* 3d⟩≡
```
void object_free(Object *o) { o->vtable->free(o); }
void _object_free(Object *o) { free(o); }
```

Next we define the *to_string* method for objects. Make sure you free the string once it goes out of scope.

3e ⟨*Default Object ToString Implementation* 3e⟩≡
```
const char * const object_to_string(const Object * const o) {
  return o->vtable->to_string(o);
}
```

The default *to_string* implementation is the same as the default Java implementation. We get the name of the class and the hex representation of the pointer. From here we print '*class_name*@pointer'.

3f ⟨*Java Style Object ToString* 3f⟩≡
```
const char * const _object_to_string(const Object * const o) {
  const char * const name = class_name((Class *)o);
  size_t size = sizeof(void *) + sizeof('@') + strlen(name) + sizeof('\n');
  char *buffer = malloc(size);
  snprintf(buffer, size, "%s@%x", name, (unsigned int)o);
  return buffer;
}
```

Finally we have the object header and implementation files all together.

3g ⟨*object.h* 3g⟩≡
```
#pragma once
#include "class.h"
```
⟨*Object Declaration* 3a⟩
⟨*Default Object Method Declarations* 3c⟩

4 ⟨*object.c* 4⟩≡

```
#include "object.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
⟨Object Implementation 3b⟩
⟨Default Object Free Implementation 3d⟩
⟨Default Object ToString Implementation 3e⟩
⟨Java Style Object ToString 3f⟩
```