Zero to Hero: C++ Classes & Structs

[aps]

meeting[14] = '05/02/2019'

Updates

Society updates 📅

- Discord link in the email
- AGM date TBD
    - Will have 2 weeks notice

Google Kick Start 2019 💻

- A global online coding competition
- Aimed at students
- Algorithmic challenges designed by Google engineers
- Top participants may be invited to interview at Google
- https://codingcompetitions.withgoogle.com/kickstart

Feedback & Meetings □💬□💬

- Feedback
  - Good
  - Bad
  - General comments
  - Improvements
- Meetings
  - Talks
  - Workshops
  - Anything else ???

- Where to send:
  - In person
  - Discord
  - Email
    - (abertaysa.com)
  - Airtable

💻 Workshop 💻

Structs

```
/* A predefined object structure that is made up of
other types. E.g: */

struct ApsStruct{
   int integer;
   char character;
   custom myType;
}
// Access defaults to public.
```

Classes

```
/* A predefined object structure that is made up of
other types. E.g: */
class ApsClass{
   int integer;
   char character;
   custom myCustomType;
}
/* Essentially the same as a struct (in C++) except
everything defaults to private. */
```

What does that mean?

```
/* struct members can be accessed easily*/
ApsStruct exampleStruct;
exampleStruct.integer = 5;
std::cout << exampleStruct.integer << std::endl;

/* unless public, class members cannot be accessed
like this ^

Two options available; the public/private labels, or
special functions called getters/setters*/
```

What does that mean? (to clarify)

```
class ApsClass{
public:
   int integer;
   char character;
   custom myCustomType;
} /* In this scenario ApsClass could be used similarly
to ApsStruct, however the getter/setter build pattern
is more than just an annoyance - it saves you from
yourself */
```

Saves me from myself?

```
ApsClass exampleClass;
exampleClass.integer = 1337;
if (exampleClass.integer = 1234){
   // will always do thing (notice above)
}
/* Among other things, Getters and setters can allow
different access levels -the get may be public, but
the set could be protected, and/or lazy loading (only
storing resource when it is needed (file handling?).*/
```

Inheritance

```
// Structs and classes are 2 sides of the same coin
// Which allows the following (I’m sorry!)
struct InheritanceStruct : ApsClass {
   // Default Inheritance = private
}
// And also this
class InheritanceClass : ApsStruct {
   // Default Inheritance = public
}
```

Inheritance

```
// More examples
struct animal {
   bool isPet = false;
   std::string likes = "food";
};



struct cat : animal {
   int lives = 9;
   bool isPet = true;
};
cat miaow;
miaow.likes = "playing with food till it dead";
std::cout << "Miaow is a pet: " << miaow.isPet << ", and likes "<< miaow.likes << std::endl;
```

Inheritance

```
// You can extend any class/struct that allows it…
// (example has been trimmed for demo purposes)
// RSPEC-112: Generic exceptions should never be thrown
class DetailedException : public std::runtime_error{
    Private:
        std::string default_message = "An unknown error occurred!";
    Public:
        // Allows us to throw a 'DetailedException' with a custom_message argument
        DetailedException(std::string &custom_message) throw() : std::runtime_error(custom_message.c_str()) {}
        // Allows us to throw a 'DetailedException' with no message argument (uses default_message)
        DetailedException() throw() : std::runtime_error(default_message.c_str()) {}
}


class FileNotFound : DetailedException{
    std::string default_message = "Could not open file: FileNotFound";
}
```

Tip for constructors

```
class DeviceManager{
public:
        DeviceManager(const std::string &ip_or_devname, bool verbose){
        // Do thing
        }
        // Delegated constructor (requires c++ 11) - Effectively lets you use default parameters
        DeviceManager(const std::string &ip_or_devname) : DeviceManager(ip_or_devname, false){
        // Maybe do thing, maybe not
        }
}
```

When to use

```
// students example
All students have:
    names
    modules (name, desc, grade)
    start year, end year
student functions:
    getGPA(), getName(), setName(), getTerm(), setEnd()
module functions:
    getName(), getDesc(), getGrade(), setGrade()
```

How to use

```
// it depends
```

Top tips

- Class = noun
- Function = verb
- Attribute = adjective

Extra problem

A local library is cataloguing their collection of books. Each book has an author, title, and year associated with it. Different books might have the same author, title, or year.