## y\

## 

## Thanks to all of you who pointed out typos in the notes, and gave formatting suggestions:)

## But the google doc seems auto-reject comments from time to time - please lmk if you ran into this!

### **Variables in C++**

#### **name, type, location (in memory), and value**

### **Type**

#### **Primitive type variables**

* + - Similar to what you have seen in Java
    - int/char/double/boolean/float/pointer (See more in Wikipedia)

#### **User-defined variable/complex variable/object**

#### We use classes to define new variable types

### **Encapsulation**

* + The point of encapsulation is to separate the interfacefrom **implementation** but still keep them as a cohesive unit
    - **Interface/the API** - **what** is the class supposed to do
    - **Implementation** - **how** is the class supposed to do it
  + With such separation between interface and implementation, we could update our coding implementation of function without update our interface
  + In C++ convention, normally we put the interface in a file with “.h” extension and the implementation in a file with “.cpp” extension.

|  |  |  |  |
| --- | --- | --- | --- |
| cube.h | | cube.cpp | |
| 1  2  3 | Interface/API | 1  2  3 | Implementation |

* + Classes are like containers of holding variables and methods. Therefore, to **define** class is to specify its components - all its members including variables and methods.
    - This is within the .h file. Technically, defining a class is creating its API.
  + On the other side, the member variables and methods are said to be **declared** within the “.cpp” file. This means that we have determined their components - their return type, name, and parameters.
    - A class is **defined** by specifying its members, however, methods and variables are **declared** when we specify their components.

### **Inclusion guards**

* + **“#pragma once”** is like sending a message to the compiler such that this particular file will be included only once within this single compilation.
  + Either way works. So pick one and be consistent. However, it seems that now we are moving more towards “#pragma once” style.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cube.h | |  | cube.h | |
| 1  2  3 | #ifndef CUBE\_H\_  #define CUBE\_H\_  class Cube {  public:  private:    };  #endif | Equivalent | 1  2  3 | #pragma once  class Cube {  public:  private:    }; |

A class is **defined** by specifying its members. However, methods and variables are **declared** when we specify their components. We implement methods and initialize variables within the .cpp file.

For example:

|  |  |  |  |
| --- | --- | --- | --- |
| cube.h | | cube.cpp | |
| 1  2  3 | #pragma once  class Cube {  public:  double getVolume();  ...  }; | 1  2  3 | #include “Cube.h”  double Cube::getVolume() {  return length\_ \* length\_ \* length\_;  }  ... |

#### **Scope resolution**

* + **Example**

Cube::getVolume() means the getVolume() method of class Cube

To be continued..