CPP_Review

Classes and Objects

- Classes & Objects
- *Classes define abstract characteristics of a type*
- *Members can be*
  - Data variables
  - Functions & Constructors
- *Object*
  - Instance of a class
  - Classes and Objects II
- *Information Hiding Labels*
  - Private
  - Public
  - Protected
- *Default Parameters*
  - Parameter to the constructor is optional
- *Explicit Constructor*
  - Avoids automatic conversion
  - Functions & Constructors
  - Multi-File Compilation
  - Composition and aggregation
- *Constant Member functions*
  - Examines but does not change the state of the object
  - Called accessors
- *Interface is defined through the .h files*
  - `#include` in the cpp file
  - Also referred to as declaration
- *Preprocessor commands*
  - Guards against multiple inclusions of .h files

```
// Preprocessor Statements
#ifndef _NAME_OF_FILE_
#define _NAME_OF_FILE_
```

```
/*
        Code Goes Here
*/

#endif
```

- *Scope-resolution operators*
    - symbolized by the `::`
    - To identify the corresponding class to each function
    - Function signatures must match in the definition and the implementation file

```
// dog.h *Implementation File*

class dog{
public:
        dog();
        void setName(string n);
private:
        string name;
};
```

```
// dog.cpp *Definition File*

dog::dog(){
        name = "Jerry"; // Default Name
}
void dog::setName(string n){
        name = n;
}
```

- *Objects are declared like primitive data types*
- *Standard Vector Class*
    - Gives `size()` function
    - Can be assigned using =
- *Standard String Class*
    - Compares with `==, <, etc`
- *Keyword auto*
    - You do not need to specify the type

```
auto i = 20
auto itr = vect1.begin();
```

- auto may not be used in some cases
- *Pointer variable*
  - stores the address of another obj in memory
  - before the variable name indicates a pointer declaration
  - `type * variableName;`
  - Pointers are uninitialized when declared, this may result in bugs

#cpp #review #classes #objects #code