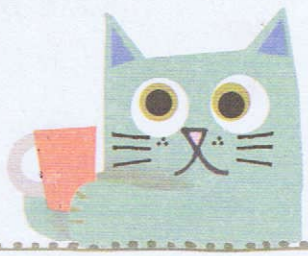
CS 109 - January 9 - 16:00 - 17:05

- Why object oriented?
  - software complexity
  - link together
  - the more flexibility, harder to build.
- Complexity
  - form of hierarchy
- Must be simple in object oriented design.
- Divide & Conquer
- * DECOMPOSITION (key term)
- Algorithmic Decomposition
  - structured design
  - no data in design, just modeling
- Object Oriented Decomposition
  - Class definitions
    - data
    - encapsulation, polymorphism, etc, security, inheritance
    - objects interact through messages.
- Algorithmic vs Decomposition
  - Choose both
  - start decomposing
    - start to breakdown



- Two Complementary Approaches

- Abstraction
  - diagnose, ~~diagnalyze~~, break-down.
  - decompose into chunks
  - eliminate un-needed details.

- Hierarchy
  - Class hierarchy
    - Parent, child class
    - structure, how classes are related to each other.
  - Object hierarchy
    - collaborations through pattern of interaction.

Designing

- start to plan logically
- must be a well constructed system.
- good performance and usage.
- restrictions (factors: length, time)
- simple internal structure, - Architecture.

Lecture 3- 1/13/17

Declaration & Definition

- Introducing some sort of scope (variable, name, function)
  - doesn't really define it.

-:: = global

- Scope: within curly brackets

Bitwise Operations

be able to print as binary

Basics

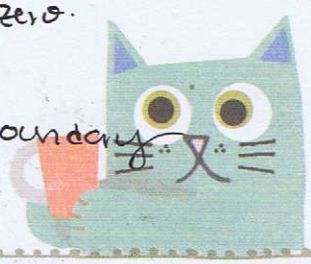
- auto — C++11, determines type
- if variable is defined, its type can't be changes throughout the double.

Static variables

- variable that is available to all scopes of the program.
- initialized once.

Allocation & Handling

- allocating in heap (mainly pointer)
- malloc - function takes size bytes and allocates in the heap, if can, returns value, if not, returns nothing.
- calloc - memset / initializes to zero.
- valloc - allgned on a page boundary



January 11 - 2017 - CMPS 109

Lecture 1:

- Meaning of Design
  - software will be running in consquent enviornment (adapt to limitations)
  - main ~~functionality~~ point functionality, only then performance
  - UML (design process)
    - key tasks: principals of decomposition, abstraction, hierarchy.
  - UML - describe problems through a different perspective.
  - 3 different design methods
  - notation: syntax / specification / tool to describe model.
  - process: develop / construct to build the model.
  - tools: configuring tools / reject errors/mistakes.

The Object Model

- Objects: ~~is~~ instance of a class - state of operations (contains data, properties of its class).
  - Classes compile time
  - objects are runtime

- Concentrating on OOP (Object Oriented Programming)
  - implementation of objects
- Object Oriented Analysis (OOA)
  - observing and analyzing classes ~~of objects~~.
- Object Oriented Design (OOD)
  - designing / building / decomposition / ~~[illegible]~~
  Choosing notation for a model.

Lecture 2

Compilation Process

g++ -E (generates pre-processed .cpp file)
-S generates assembly code.

- ~~[illegible]~~ number BSS - # of uninitialized global/static variables
- Makefile: framework / configuration file
  / ability to find out which file is ~~compiled~~ modified and compile only that.
  - commands must me processed by tab character.

g++ (files .cpp) - o (output file)
<target> : <dependencies>

[Command] make : looks for "GNUmake" "makefile" "Makefile"

tools: g++ - compiler
ld - linker
objdump - de-assemble, extract info from obj files.
nasm - assembly compiler
valgrind - memory leaks
gdb - debugger