Variables/objects/states.

http://www.cs.colorado.edu/~main/chapter1 guess.cxx

Analysis of id-numbers.

Exercises with arrays of throttles with titles and

Next week's labs: (1) Training in using DDD to observe copying of

Reading assignments.

CSI 310: Lecture 2 (Spr 2006)
It.
programming under Unix during the remainder of this if you need

don't have one, and get help from course staff about doing C++

Get an Academic Computing Unix cluster account ASAP if you

The training (next week) will be on add.

through sec. (2.3)

start reading them. Lab 2 exercise and training will use chapter 2

We will cover chapters 1, 2 and 3 during the first 3 lectures, so

postconditions.

documentation for function interfaces in the form of pre and

In this course, we will require and grade your writing of
Program documentation.

Invariants are one of the best ways to structure reliable and useful
Pre- and Post-conditions are special kinds of invariants.
the spot where the invariant is written. 
The invariant is TRUE whenever program execution passes through
code.

So, an invariant is written in a particular Spot in the Program
statement might be true sometimes but become false at other times.
Problem: Values of variables change as a program runs, so a

The best documentative statements express the TRUTH.
// Postcondition: The user has been asked to think.
// Precondition: n > 0.
//
void guessGame(int n)
{
    // this demonstration program
    // Prototype for the function used in

    // Library items to be used
    using namespace std;
    
    #include <cassert>
    #include <cstdlib>
    
    #include <iostream>
    #include <sstream>

    // as a time analysis example.
    // Demonstrates a guessing Game function that's used

    FILE *FILE; GuessGame xx

    // See the pre/post-condition documentation below.
(sic changed the tense to past).

series of questions, until the number was found. //
of a number between i and n. The function asked a //
{  
    return EXIT_SUCCESS;
    guess_game(100);
}

int main()

Test driver for guess_game()
```c
answer = 'N';

} else {

  cout << "Think of a whole number from 1 to 1000;"

  assert(n >= 1); // Library facilities used: assert, assertf

  void guess_game(int n)
  {
      Implementation:
  }
  ```
Flowcharts for `for` if and it statements contrasted.

Short-circuit evaluation of `TEST` expression howchart...

for-statement howchart.

The rest is omitted.

```cpp
{ 
  cin >> answer;
  cout >> "Please answer Y or N, and press return."
  cout >> "Is your number " "guess " "end?"

  if (answer == 'Y') {
    for (guess = 0; guess < n; guess = n); 
  }
  
  else { (answer = i = 1); 
  }
```
Usually, RAM: Random Access Memory

which costs money.

Each variable is implemented by a separate piece of hardware

such like: cell, box, storage facility

synonyms: object, instance, memory location, location.

**VARIABLE**

Data is more important than executable code.

**MEANING** of data is *required* and will be graded.

enough

Writing documentation of function interfaces and precise

CS1310: CS2 Major Subj: "Getting your program to work" is NOT
Inputs to a calculation. Calculation results can be stored in
Values in variables can be copied into variables or used as:

Stores.

A variable is a data structure. Its value is the data it currently

The variable is NOT THE SAME AS ITS VALUE.

The VALUE is in the variable.

Java variables are NEVER uninitialized.

```
{
    The value of N is 0.
    \N + N = N
    Now the value of N is 3.
    Assignment stmt. \N = 3;
    The value of N is unpredictable
    int N; // NO INITIALIZER!!
}
main()
such number, so we say this equation "has no solution." You add 5 to it, you get the same number. Of course, there is no is to solve the equation. That is, find, if any, a number so that it

\[ N + 5 = N \]

The routine mathematical problem about an equation like

Variables in programming are different in variable in mathematics.

(what it contains)

- Synonyms for "variable in a variable": The variable's state, its
data's value, etc.
- Synonyms for "variable": Object, instance, memory location.

variable; but storing changes the value it contains.

Copy into a variable's doesn't change the variable's identity or the
in the variable.

Copying or using a variable's value does not change the value
variables.
directional symbol for assignment, as in "\( n = n + 5 \)"

Some other programming languages (Algol and Pascal) used a

the old value." (not an equation). It means "replace the old value of \( n \) with \( 5 \) plus

In C/C++/Java, \( n = n + 5 \); is an ASSIGNMENT STATEMENT
\[ N = 2 \times N; \quad \text{if } N \text{'s value is now } 6 \quad \text{and } N \text{ value changed. NOT.} \]

\[ N = 3 \times N; \quad \text{if } N \text{'s value is now } 3 \quad \text{then the same } N. \]

\[ \text{int } N; \quad \text{if } N \text{ is created. We cannot predict its value.} \]

Value does change.

That identity is unchanged even though the variables created. Each variable retains the identity it gets (WHAT IT IS) when it is.
Each object has state together with behaviors.

The most important idea of Object Orientation Programming adds

Example of a concrete class, discussed fully in Ch. 2

```java
public class Point {
    double x; // x-coordinate holding part of the state
    double y; // y-coordinate holding part of the state

    public void print() {
        System.out.println("x = "+ x + " y = "+ y);
    }
}
```
states. OPEN, CLOSED, SHRIVELED.

Apologies to vegetarians, cat lovers, etc. Replace cat by horse with

States of a cat: AWAKE, SLEEPING, DEAD.

Example: Your cat, assuming you have exactly one cat.

same.

unique identity. It's state may change but its identity remains the

The NEXT important idea is that each object or variable has a
```
* // except in Java, don’t write the *
   // OK in C++ and Java
   
   point *ptrPoint = new point;
   
   // ONLY IN C++, NOT JAVA
   
   point MyPtrStPoint; // static or automatic extant,
   
   creating an object:
   
   {;
   
   this METHOD prints a report of this object’s state
   
   ( void print ( void )
   
   public:
   
   x-coordinate (another variable) holding part of the state.
   
   y-coordinate (another variable) holding part of the state.
   
   x;
   
   y;
   
   };
```
invoking a method in C++