development offices all over Silicon Valley, NYC, etc.
Pointer/reference/object diagrams are drawn on whiteboards in software
Computing the Cycles of a Permutation given by an integer array.
Data referring to data instead of real things.

Doing Project 1.

CSI 310: Lecture 6 (No lab assignment. Use Lab time for TA help with Proj. 1.)
Do Part 2 in similar fashion.

Perform (Build, Test, Debug, Develop, cycle) to finish Part 1.

test your Pile class.

Download sequence-test.cxx, rename to test-Pile.cxx and modify it to

    { 
      cout << "Pile::sort() not implemented yet" << endl;
      stub //  }
    }

void Pile::sort ( )

Code CData.cxx and Pile.cxx files... write stubs for methods first. Like

Copy our CData.h and Pile.h files

READ the assignment.

 Done Project 1:
LOCATOR strings to code a Web link, which locates ANOTHER Web page or
(1) Web pages in HTML are text files; Web authors type Uniform Resource
Examples:

BUT OTHER DATA
(not something real)

TARGET, IDENTIFY, DESCRIII 0r otherwise "get at"
Sometimes data is used to LOCATE, RETRIEVE, IDENTIFY, ACCESS,  
"disks", library books, etc.
Data is also often stored in persistent media, hard, optical or flash memory.
Data is often stored in variables/objects (while a program is running).

Subject:
most-recently-inserted \( \text{Coin} \) is located.

The value of \( \text{used} \) = \( \text{index value} \) is the index where the

\[
\begin{aligned}
\text{if not used:} \\
\text{\text{Coin[A[capacity]] = \_}} \\
\text{...} \\
\text{class \text{Pipe}}: \\
\text{(2) Current Project:}
\end{aligned}
\]
It is simple to store a permutation in an array:

The permutation's number in position 4 is 4.
The permutation's number in position 3 is 5.
The permutation's number in position 1 is 2.

Two line notation $n = 5$:

ORDER, with each number appearing exactly once.

A permutation of numbers $\{1, 2, \ldots, n\}$ is those numbers written in some

Another example of this theme...
Also, since the top line of the indexes are always 1, 2, 3, ... in the top line can be

```
[5] [4] [3] [2] [1] [0]
1 4 5 2 3 wasted
1 4 5 2 3
1 4 5
```

omitted:
Two-column (bipartite) graph

Graph on \(1\-n\) that shows cycles.
Cycle expression is \((1\ 2\ 3\ 5\ 4)\)

Let's think of a permutation as a function or mapping

University at Albany Computer Science Dept.
How to find this permutation's cycle:

<table>
<thead>
<tr>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>wasted</td>
</tr>
</tbody>
</table>

With any one of them, 5
Are any of the numbers 1, 2, 3, 4, 5 not in the cycle(s) written? If so, continue

```
\begin{align*}
\text{But, } p[5] &= 1, \text{ a number we already wrote. So finish the cycle } 1 \ 2 \ 3
\end{align*}
```

(2)

```
\begin{align*}
\text{Also, } p[3] &= 5, \text{ so continue } 1 \ 2 \ 3 \ 5
\end{align*}
```

(6)

```
\begin{align*}
\text{Similarly, } p[2] &= 3, \text{ so write } 1 \ 2 \ 3 \ 5 \ 2
\end{align*}
```

(5)

(3)

```
\begin{align*}
\text{If } p[1] \text{ equals } 1, \text{ you get back to the beginning of the cycle. Finish writing}
\end{align*}
```

(2)

```
\begin{align*}
\text{Find what I is mapped to—look up } p[i].
\end{align*}
```

(1)

```
\begin{align*}
\text{Start at } 1, \text{ write } 1.
\end{align*}
```

(1)
(11) Check and detect there are no more numbers not in cycles we have written.

(10) Detect $p_{4} = 4$, so finish the $I$-cycle $4$.

(9) In this example, continue with $4$. Write $4$.

$E$
instance FO which the function is called. It is a C++ pointer to that instance.

Advanced notice: "this is a reserved C++/Java keyword that refers to that
called... Here it is element 3 of the Passe array Pi.

The names "used", "used" data fields in that INSTANCE FO which put() was
"A" and "used" in the body of void Passe:put():... refer to the

pt [3] . put ( c ) ;
      c0n c ;
      ptte pt [6] ;
      Passe: p = ( ) ;
Thus:

For a particular variable (synonym: instance) whose type is that class. Thus:
An (instance) member function (synonym: method) of a class is always called
qualifier, as in Passe: kick ( ) ;

static method outside the scope of a class requires the class name as a scope
object/instance of the class; just like self-standing C/C++ functions. Calling a

In Java/C++, a method declared static is called independently of any

©S. Chakravorty 2006
University at Albany Computer Science Dept.
(changed.

features of the variable: its VALUE (synonym: state) can be
"cloned." Features of the variable, is VALUE (synonym: state) can be
person, autobiography, shoebox, a variable has a unique identity even after it is

The NAME can be copied. The VARIABLE is the OBJECT itself. Like a

VARIABLE

4. (This is new.) A NAME can be the VALUE of a

conceputal entities (3 different things).

3. The value, the name, and the variable itself are three DIFFERENT

2. Each variable holds a value (or state).

1. Each variable (synonym: object) has (some kind of) name.

The main points...

©S. Chalke/Ken2006
University at Albany Computer Science Dept.
We will use yellow to denote what has changed.

Brandon = 98

Adam

Brandon

Brandon

Charlie

Brandon

Charlie

Brandon

Adam

Brandon

Brandon
variables are comprised of different numbers of bytes. Don’t worry about this yet...

One big house might cover several adjacent lots. Technically, different sizes of

current VALUE.

of a memory location or byte IS VERY DIFFERENT FROM the variables’
address (called bytes). The byte numbers are called memory addresses. The address

a segment of computer memory consists of numbered memory locations

a house’s current OCCUPANTS.

called address. The address of a house IS very different from the

A block of real estate consists of numbered houses. The house numbers are

name that C/C++ code can copy, use, store, etc., at runtime.

A pointer is the memory address of a variable. A pointer is the kind of
The value of this variable is 57. A pointer value or points to a 4-byte integer variable.
Pointer values, like any other numbers, can be stored in variables, like any other variables. Real programmers write their addresses in hexadecimal (base sixteen) because hex to binary (base 2) conversion is very easy.

The pointer variable whose address is 987 is pointing to the above integer variable whose address is 992 and value is 57.

Here's the integer variable whose value is 57. Its address is 992.

Here's a pointer variable whose value is 992. Its address is 987.

I will denote pointer values by black dots because the numeric value is usually boring.

Real programmers write their addresses in hexadecimal (base sixteen) because hex to binary (base 2) conversion is very easy.
POINTER VALUES, and call them "references".

Unlike C/C++, Java and Perl (except debuggers) hide all numeric

systems. Sometimes within I/O devices (memory mapped ones).
The memory bus addresses usually locate data within cached RAM hardware

addresses and sometimes into page fault signals.

bus or physical memory management units sometimes into hardware memory

Hardware (binary) virtual addresses are quickly translated by

Numerical-pointer values are virtual addresses, and part of the

Numerical-pointer values, and part of the

PC/Workstation/Servers with Unix-like/Windows NT

operating systems:

a few background words...
$ the value of x is 98.6
the address of x is 0xbff9ba4
$a.out$
-------------------
{
  cout << "the value of x is " << x << end;
  cout << "the address of x is " << &x << end;
  float x = 98.6;
}
main()

using namespace std;
#include <iostream>

which must be a variable.
The C/C++ & operator provides the address of its operand,