LOYOLA UNIVERSITY CHICAGO | CENTER FOR TEXTUAL STUDIES AND DIGITAL HUMANITIES

DIGH 401 - Introduction to Computing

Fall Semester 2014

Week 11

Today's Class

- Week 9 Exercise
- Conceptual Design Specification
- Project Presentation
- Manipulating data (continued)
- Data Structures
    - array consideration
    - stacks
    - queues

DIGH 401 - Introduction to Computing

Week 9 Exercise - Mathematics and PHP

Write and test some PHP code to calculate the following:

- the square root of 12 plus 8 divided by 2, and the square root of 15 minus 9 multiplied by 4 - Uses sqrt() function
- using the results of the above calculations, round each number to the nearest whole integer - Use round() function to round a floating point number or use ceil() to round up, and floor() to round down
- using these two whole integers, determine the larger of the two numbers - Uses max() function

Conceptual Design Specification

- use your outline for a conceptual project (DIGH 400) as the basis
- this project focuses on the conceptual implementation rather than using a proposal model
- think about how to plan the construction of the programming or output of the project
- choose a 'software engineering model' to help plan and outline your project

eg: you might choose to base the process on the waterfall model using the steps for analysis, design, implementation, and testing

- detail each step of your model for your project
- implementation and testing phases can be conceptual
- explain in an introduction or overview why you chose your particular software model
- models can also be used, such as flowcharts or UML EXAMPLE

Project Presentation 2

25th November 2014 - ~10 mins per paper & questions from the class

- conference style short paper
- outline your design specification etc
- use template for guidance
- UML or flowchart outlines

Manipulating Data - Comparison Operators

- comparison of two data values to see which is bigger than the other
- comparison operator returns either true or false
- often used in conditional expressions

==	equal to	x == y
===	identical and of same type	5 === 5
<> or !=	not equal	x != y
!==	not identical	x !== y
<	less than	x < y
<=	less than or equal to	x <= y
>	greater than	x > y
>=	greater than or equal to	x >= y

Manipulating Data - Comparison Operators

String comparison

- comparing strings using this method can be slightly confusing
- computers use a number to represent a letter
- ASCII table contains numbers for characters

A > a

- compare multiple characters in a string

aA > aa

aA > a

Aa > a

- as soon as the computer finds values that differ, it will cease the comparison and return a boolean result

Manipulating Data - Using Boolean Operators

- comparison operators return Boolean values, either True or False
- as with numbers and strings, Boolean values can also be manipulated
- boolean operators also return either True or False results
    - Not, And, Or, Xor
- PHP known as 'Logical Operators'
    - And, Or, Xor, Not

```
x and y
x or y
x xor y
```

Manipulating Data - Using Boolean Operators

Java

!	NOT operator
&	AND operator
|	OR operator
^	XOR operator
||	short-circuit OR operator
&&	short-circuit AND operator
==	EQUAL TO operator
!=	NOT EQUAL TO operator

```
12 < 6 && 18 > 1
12 < 6 || 18 > 1
```

Manipulating Data - Using Boolean Operators

NOT operator

- takes a boolean value and converts it to the opposite
- True to False, and vice-versa

Not(3 > 2)

DIGH 401 - Introduction to Computing

Manipulating Data - Using Boolean Operators

AND operator

- takes two boolean values and converts them into a single boolean value
- if boolean values are true, the And operator will return true
- if a False boolean value exists, the And operator will always return false

1.

True	True	
True	False	
False	True	
False	False	

2. (3 > 2) AND (3 >= 18)

Manipulating Data - Using Boolean Operators

OR operator

- takes two boolean values and converts them into a single boolean value
- both boolean values are false, it will return a false value
- otherwise it will return a true value

1.

True	True	
True	False	
False	True	
False	False	

2. (3 > 2) OR (3 >= 18)

DIGH 401 - Introduction to Computing

Manipulating Data - Using Boolean Operators

XOR operator

- XOR is an exclusive OR operator
- converts two boolean values into a single boolean value
- both values True or False, XOR returns a False boolean value
- if one value is True and the other is false, XOR returns a True boolean value

1.

True	True	
True	False	
False	True	
False	False	

2. (3 > 2) XOR (3 >= 18)

DIGH 401 - Introduction to Computing

Data

- every computer program needs to store data
- dump data into a variable
- need a separate variable for each chunk of data
- we can also store data together, for example
    - arrays
    - structures

Data Structures

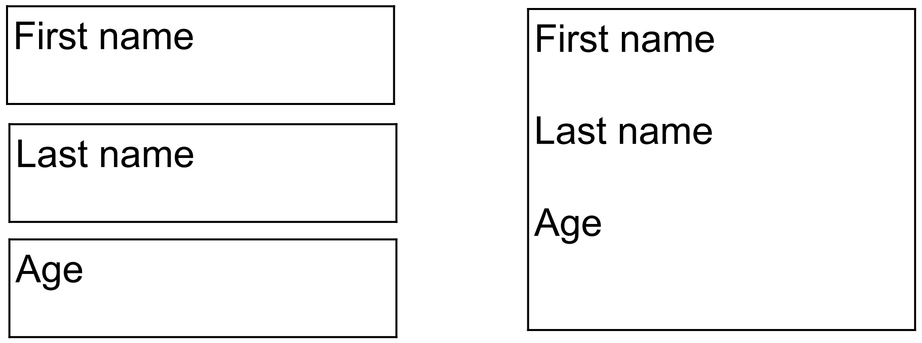
- group separate variables together
- store multiple variables within another variable
- user-defined data type
- cannot use a structure until you declare a variable to represent the structure

```
int[] anArray;
```

```
$students = array();
```

DIGH 401 - Introduction to Computing

Data Structures



DIGH 401 - Introduction to Computing

Data Structures - storing & retrieving data

- identify the variable that represents that structure
- identify the specific variable inside the structure to use

Create array: $students = array();

Insert a value: $students[0] = "Emma"

Output a value: echo $students[0];

Example

Data Structures - storing & retrieving data

```
<?php

$students = array();

$students[0] = 'Emma';

echo $students[0];

?>
```

Data Structures - Stacks

- data structure providing two basic methods
    - push
    - pop
- buffering a stream of objects
- last in is first out (LIFO)
- data items for processing, instructions for processing, instructions pending execution...

A stack of plates...

```
array_push($students, 'Catherine');

array_pop($students);      NB: deletes last element of the array
```

Example

DIGH 401 - Introduction to Computing

Data Structures - Stacks

```
<?php

$students = array();

$students[0] = 'Emma';

echo $students[0].'<br />';

array_push($students, 'Catherine');

echo $students[1];

?>
```

Data Structures - Stack implementation

plates again...

1. first plate begins the pile
2. next plate is placed on top and so on
3. a plate may only be removed from the top
4. order of pushing or popping plates is arbitrary
5. there will always be a certain number of plates on the pile
6. pushing a plate increase the pile by one, popping a plate decreases the pile by one
7. you cannot pop a plate off an empty pile
8. you cannot push a plate onto a full pile, assuming there is a maximum number for the pile

7 & 8 will need to be monitored.

Data Structures - Stack implementation

plates using a stack

1. We can create an array to hold the plates.

2. we then maintain a pointer to the top of the array, which acts as a marker to tell us the index in the array where the next plate is pushed and stored

3. we'd probably also set a maximum for the stack, and set the first top pointer to zero.

4. then we would need to be able to monitor the status of the stack to check total number of plates, and the current top pointer index.

5. so, we now know that the stack is empty when the pointer is zero, and it will be full when the top pointer is equal to the size of the stack array, as defined by the user, for example

6. when we push a new plate on to the pile we simply check the current number of plates, the current index for the top pointer, and then increment the top pointer by 1 if there is space.

7. to pop a plate from the pile we need to check that a plate exists, and then decrease the index of the top pointer by one.

Data Structures - Queues

- similar to a stack except you join the queue at one end and leave at the other
- first in is first out (FIFO)
- buffering a stream of objects
- we can use an array for storing items in the queue
- two pointers in the queue
    - one to indicate where the next item should be stored
    - another to indicate the location of the next item to be retrieved

```
array_shift($students);  NB: removes first array element & outputs
```

Example

DIGH 401 - Introduction to Computing

Data Structures - Queues

```
<?php

$students = array();

$students[0] = 'Emma';

echo $students[0].'<br />';

array_push($students, 'Catherine');

echo $students[1].'<br />';

echo array_shift($students);

?>
```