
CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

# DIGH 401 - Introduction to Computing

Fall Semester 2013

Week 10

# Today's Class

- Conceptual Design Specification

- Decision making with branching options
    - conditional statements and options

- Looping

- Arrays, a gentle introduction

## 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

4th December 2013 - ~10 mins per paper & questions from the class

- conference style short paper
- outline your design specification etc

## Conditional statements - IF-THEN

- simplest conditional statement is IF-THEN

IF (condition is true or false) THEN Command

- in many programming languages THEN is represented as {

```
if ( true or false ) {
...command
}
```

## Conditional statements - IF-THEN-ELSE

- IF-THEN only performs a command for one result
- ELSE allows additional commands to be used within the conditional statement

```
<?php
    $score = 65;

    if ($score >= 90) {
    echo 'grade = "A"';
    }
    else {
    echo 'grade = "F"';
    }
?>
```

Example

## Conditional statements - IF-THEN-ELSEIF

- ELSEIF used to add multiple possible choices to conditional statement
- two advantages over IF-THEN-ELSE
    - check a condition for each set of commands
    - define three or more separate sets of commands
- additional choices created by adding additional ELSEIF statements
- add ELSE to end of conditional statement to ensure a command is executed

```
<?php
  $score = 82;
   if ($score >= 90) {
    echo 'grade = "A"';
    }
    else if ($score >= 80 ) {
    echo 'grade = "B"';
    }
    else {
    echo 'grade = "F"';
    }
?>
```

Example 1  Example 2

## Conditional statements - IF-THEN-ELSEIF

multiple boolean expressions

- multiple boolean expressions evaluate to a single true or false value
- these expressions can also be used within IF-THEN statements
- no limit to the number of boolean expressions we can combine with boolean operators

```
(age >= 20) && (age <= 65) && (name = 'Jack') || (age = 27) || (name = 'Emma')
```

## Conditional statements - SELECT CASE or SWITCH

- an alternative to IF-THEN-ELSEIF
- examines a variable and if it matches specific values a command or block of commands will be performed

```
<?php
$score = 70;
   switch ($score) {
    case 90: $scoreString = "A";
    break;
    case 80: $scoreString = "B";
    break;
    case 70: $scoreString = "C";
    break;
    default: $scoreString = "Fail";
    break;
    }

    echo $scoreString;
?>
```

Example

## Conditional statements - SWITCH

```
switch ($month) {
case 1: $monthString = "January"; break;
case 2: $monthString = "February"; break;
case 3: $monthString = "March"; break;
case 4: $monthString = "April"; break;
case 5: $monthString = "May"; break;
case 6: $monthString = "June"; break;
case 7: $monthString = "July"; break;
case 8: $monthString = "August"; break;
case 9: $monthString = "September"; break;
case 10: $monthString = "October"; break;
case 11: $monthString = "November"; break;
case 12: $monthString = "December"; break;
default: $monthString = "Invalid month"; break;
}
```

## Conditional statements - SWITCH

```
switch ($month) {
     case 1:
     case 3:
     case 5:
     case 7:
     case 8:
     case 10:
     case 12:
     $numDays = 31;
     break;
     case 4:
     case 6:
     case 9:
     case 11:
     $numDays = 30;
     break;
    default: $monthString = "No idea...";
    break;
    }
```

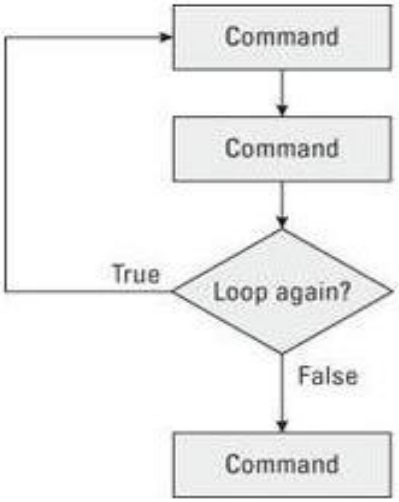
# Conditional statements - SWITCH

```
$month = 4;

echo 'Month = '.$month.'<br />';

switch ($month) {
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
     $numDays = 31;
     break;
case 4:
case 6:
case 9:
case 11:
     $numDays = 30;
     break;
case 2:
     if ( ((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0) ) $numDays = 29;
    else $numDays = 28;
    break;
default:
    echo 'Invalid month.';
    break;
 }

 echo 'Days = '.$numDays;
```

## Looping

Conceptual considerations

- we could write out a command the number of times it is needed, but this is often a complete waste of code
- a 'loop' is a shortcut for making a computer run one or more commands multiple times
- write out once, and then specify the number of times to repeat



## Looping - FOR loop

- simplest loop is a basic FOR loop, which runs one or more commands a fixed number of times

```
for ($i=1; $i<11; $i++) {
      echo 'position ='.$i.'<br />';
  }
```

Example 1  Example 2

## A quick detour into Arrays

- a type of data structure
- stores multiple values in one single special variable
- an array can hold all of your variable values under a single name
- access the multiple values by referring to the array name
- each element in the array has its own index
- in most languages there are multiple forms of arrays

PHP = numeric, associative & multi-dimensional

## A quick detour into Arrays - Numeric or One dimensional Arrays

- stores each array element with a numeric index
- simple to iterate over the array with a loop
- reference each element in the array by the name of the array and the index

```
$cars=array("Volvo","Saab","Audi","Renault");

$cars[0]="Volvo";
$cars[1]="Saab";
$cars[2]="Audi";
$cars[3]="Renault";

echo 'My car is a '.$cars[0];
```

Example

## A quick detour into Arrays - multi-dimensional Arrays

- simply arrays of an array
- each element can also be an array itself, and each sub-array can be an array....
- in Java they are often referenced by the depth of arrays, eg: 2D, 3D...
- in PHP we can auto-assign array keys or manually specify

2 4 7 15

3 1 8 10

6 0 9 12

```
$cars = array
  (
  "Swedish"=>array
  (
  "Volvo",
  "Saab",
  ),
  "German"=>array
  (
  "Audi"
  ),
  "French"=>array
  (
  "Renault",
  "Citroen",
  "Peugeot"
  )
  );

  echo 'Car chosen = '.$cars['French'][1];
```

```
$cars = array
  (
  "Swedish" => array
  (
  $cars['Swedish'][0] = "Volvo",
  $cars['Swedish'][1] => "Saab",
  ),
  "German" => array
  (
  $cars['German'][0] = "Audi"
  ),
  "French" => array
  (
  $cars['French'][0] = "Renault",
  $cars['French'][1] = "Citroen",
  $cars['French'][2] = "Peugeot"
  )
  );

  echo 'Car chosen = '.$cars['French'][2];
```

Example 1  Example 2

## Looping - Enhanced FOR loop

- FOREACH loop in many other languages, including PHP
- mainly used to loop through arrays

```
<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>
```

## Looping - WHILE loop

- useful if you're not sure how many times a loop needs to run
- four main parts to a WHILE loop
    - beginning, which checks a Boolean expression for true or false
    - one or more commands to run
    - one or more commands that can change the Boolean expression
    - the end of the WHILE loop
- before the WHILE loop starts it checks a boolean expression for true or false
    - if true the WHILE loop runs, if false it won't run
- if the WHILE loop doesn't include at least one command that can change its boolean expression, the WHILE loop runs indefinitely

## Looping - WHILE loop

while (true){
 perform this or that command
}

```
<?php
$i = 0;

while ($i <=10) {
echo 'value = '.$i.'<br />';
$i++;
}

?>
```

Example

## Looping - DO-WHILE loop

- runs the loop at least once
- acts like an upside down WHILE loop
- first it will run the loop once, then it checks the boolean expression
- conceptual DO-WHILE loop contains four main parts
    - DO
    - Command
    - Command to change the boolean expression
    - Loop WHILE (true or false boolean expression)

## Looping - DO-WHILE loop

```
<?php
$i=0;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=9);
?>
```

Example

## Looping - Nested loops

- every loop can also run one or more commands multiple times
- it's also possible for a loop to run another loop & another...multiple times
- eg: a FOR loop nested inside another FOR loop

## Looping - Exit a loop early or BREAK

- loops normally run for a fixed number of times (FOR loops...)
- or until a Boolean expression changes (While or Do-While loops...)
- it's also possible to exit a loop prematurely using EXIT or BREAK

```
for ($i=0; $i<10; $i++) {

 if ($i == 4) {
 echo 'end found at '.$i;
 break;
 }
 else {
      echo 'position ='.$i.'<br />';
      }
  }
```

## Looping - A few suggestions

- looping for a known number of times is often best with FOR loops
- use a WHILE loop for looping zero or more times
- loop at least once with a DO-WHILE loop
- WHILE and DO-WHILE loops usually need a variable to check the boolean expression to determine when the loop ends
- WHILE and DO-WHILE loops always need a command that changes the boolean expression to determine when the loop will finish
- beware infinite or endless loops
- use BREAK to prematurely stop a loop

Example - February