Philadelphia University
Faculty of Information Technology
Department of Computer Science
First semester, 2008/2009

Course Syllabus

Course Title: **Programming Fundamentals**	**Course code: 750112**
Course Level: 1	**Course prerequisite (s) and/or corequisite(s):** none
Lecture Time:	**Credit hours: 3**

Academic Staff Specifics

Name	Rank	Office Number and Location	Office Hours	E-mail Address

Course/Module Description:
This module focuses on problem solving strategies and the use of algorithmic language to describe such problem solving. It introduces the principles of procedural programming, data types, control structures, data structures and functions, data representation on the machine level. Various problems are considered to be solved using C-like procedural programming language.

Course/Module Objectives:
This module aims to introduce computer programming and emphasis in problem solving on the fundamentals of structured design using the principles of Top Down problem solving strategy (divide and conquer). This includes development, testing, implementation, documentation.
The module also aims to explore the logic of programming via the algorithm concepts and implement them in programming structures including functions, arrays, strings, and pointers.

Course/ module components

- **Textbook:**

  D.S. Malik , Thomson, C++ Programming: From Problem Analysis to Program Design, Third Edition, Course Technology, 2007

- **Supporting material(s):** Lectures handouts

Teaching methods:
Duration: 16 weeks, 80 hours in total
Lectures: 32 hours (2 hours per week),
Tutorials: 16 hours (1 per week),
Laboratories: 32 hours, 2 per week

Learning outcomes:
- Knowledge and understanding
  1- Understand the fundamental programming constructs.
  3- Understand and write searching and sorting techniques.
  4- Understand a typical C-like program environment.

- Cognitive skills (thinking and analysis).
  1- Be able to understand and analysis any problem and derive its solution.
  2- Be able to develop algorithms.

- Communication skills (personal and academic).
  1- Be able to work as a team

- Practical and subject specific skills (Transferable Skills).
  1- Be able to write C-like programs including searching and sorting techniques.

Assessment instruments

Allocation of Marks	
Assessment Instruments	**Mark**
First examination	**15%**
Second examination	**15%**
Final examination	**40%**
Lab works, Quizzes, and tutorial contributions	**30%**
Total	**100%**

Course/Module Academic Calendar

Week	Basic and support material to be covered	Homework/reports and their due dates
(1)	***Problem Solving***: process, Analyze (requirement, Design algorithm, Tracing algorithm, Example, Design problems) **Tutorial 1**	**Lab work #1** (Get started with C language environment program editing, compiling, executing, debugging with PW 1)
(2)	***Problem Analysis***: Algorithm discovery, Algorithm design strategies**,** Stepwise refinement**,** Control requirements, **Tutorial 2**	**Lab work #1** (PW 2)
(3)	Implementing algorithm, Conclusion, **Tutorial 3**	**Lab work #3** (PW 3)
(4)	***Data Definition Structures***: Types, constants, variables, Expressions: Arithmetic, Logical; Precedence rules; **Tutorial 3** ***Control Structures***: Sequencing; Input and output statements; Assignment statement; **Tutorial 4**	**Lab work #4** (Data types, Sequencing, Assignment operations)
(5)	***Control Structures***: Selection: one-way (if .. then), two-way (if .. then .. else), multiple (switch); **Tutorial 5**	**Lab work #5** (if and switch statements)
(6)	***Control Structures***: Repetition (while structure); **Tutorial 6**	**Lab work #6** (While statement)
(7) First examination	***Control Structures***: Repetition (do while for); **Tutorial 7**	**Lab work #7** (while and for statements)
(8)	***Control Structures***: Combination; **Tutorial 8**	**Lab work #8** (combination of selection & repetition programming)
(9)	***Functions:*** Parameters definition and passing (functions depth look); prototypes; **Tutorial 9**	**Lab work #9** (function definition in C)
(10)	***Functions:*** Parameters definition and passing (Scope: local and global variables); **Tutorial 10**	**Lab work #10** (programs with functions)
(11)	***Data Structures:*** One and two dimensional arrays; **Tutorial 11**	**Lab work #11** (programs with arrays)
(12) Second examination	***Abstract data type:*** Records (**struct** definition statement); Strings (use of main operations: Concatenate, string copy, compare, etc.); **Tutorial 12**	**Lab work #12** (programs with struct)
(13)	***Strings; Files*** (use of main operations of a sequential file: open, reset, rewrite, read, write, eof); **Tutorial 13**	**Lab work #13** (programs with strings)
(14)	Files; Pointers; **Tutorial 14**	**Lab work #14** (programs with pointers & files)
(15)	Pointers; **Tutorial 15;**	**Lab work #15** (Comprehensive assignment covers all mentioned topics)
(16) Final Examination	Review and final Exam	**Lab work #16** (Revision)

Expected workload:
On average students need to spend 3 hours of study and preparation for each 50-minute lecture/tutorial.

Attendance policy:
Absence from lectures and/or tutorials shall not exceed 15%. Students who exceed the 15% limit without a medical or emergency excuse acceptable to and approved by the Dean of the relevant college/faculty shall not be allowed to take the final examination and shall receive a mark of zero for the course. If the excuse is approved by the Dean, the student shall be considered to have withdrawn from the course.

Module references

Students will be expected to give the same attention to these references as given to the Module textbook(s)

1. Friedman Frank and Koffman Elliot B., "*Problem Solving, Abstraction and Design using C++*", Addison Wesley, Fourth Edition. 2004
2. Jeri R. Hanly and Elliot B. Koffman, Problem Solving and Program Design in C, Pearson Education, Inc., ISBN: 0-321-21055-7,
3. Deitel & Deitel, C++ How to Program, Prentice-Hall, 2001.
4. A. Lambert Kenneth and Nance Douglas W., "*Understanding Programming and Problem Solving With C++*", PWS Publishing Compny, Fourth Edition. 1996
5. Forouzan, B. A. & R. F. Gilberg. *"Computer Science: A Structured Programming Approach using C",* Second Edition, Pacific Grove, CA: Brooks/Cole, 2001
6. Bruce Eckel, *"Thinking in C++*", Second Edition, Prentice Hall, 2000.
7. Herbert Schildt, *"Teach Yourself C++*", Third Edition, McGraw-Hill. 1998

Website(s):

- www.cee.hw.zc.uk/~pjbk/pathways/cpp1/cpp1.html
- www.edm2.com/0507/introcpp1.html
- www.doc.ic.ac.uk/~wjk/C++intro
- www.cprogramming.com/tutorial.html
- www.cs.umd.edu/users/cml/cstyle/ellemtel-rules.html
- www.deakin.edu.au/~agoodman/Ctutorial.html
- www.tldp.org/howto/c++programming.howto.html
- www.vb-bookmark.com/cpptutorial.html

DOCUMENTATION FOR PROGRAMS:
(All programming assignments must include at least the following comment lines)

```
/*TASK:          Identify what the program will accomplish              */
/*               */
/*WRITTEN BY:                                                            */
/*               */
/*DATE:          List creation & modification dates                      */
/*               */
/*VARIABLES:     List and give what each represents                      */
/*                                  */
/*INPUT:         Identify the input parameters: Give examples            */
/*               */
```

```
/*OUTPUT:          Identify the expected output: Give examples             */
/*                 */


/*ALGORITHM:       Briefly describe the algorithm used*/

#include <stdio.h>
main ( )
{    …  }


(If your program includes any function modules, each function needs to be documented)
/*TASK:            Identify what the function accomplishes                 */
/*                 */
/*DATE:            List creation and modification dates                    */
/*                 */
/*WRITTEN BY:                                                              */
/*                  */
/*VARIABLES:       List names and what each represents                     */
/*                 */
/*INPUT:           Identify the input parameters, if any. Give examples    */
/*                  */
/*OUTPUT:          Identify the output. Give examples                      */
/*                  */
/*ALGORITHM:       Briefly describe the algorithm used                     */

int function1( )
  {   …  }
```