Blekinge Institute of Technology
School of Computing, COM

Course Descriptor

C Programming For Engineers

15 högskolepoäng (15 ECTS credit points)

Course Code :	DV1445	Subject Area :	Computer Science
Educational Level :	Basic	Version :	1
Course Level :	G1N	Valid from :	2012-01-16
Field of Education :	Technology	Approved :	2011-12-05
Subject Group :	Computer Technology		

1 Course title and credit points

The course is titled C Programming For Engineers and awards 15 credit points. One credit point (högskolepoäng) corresponds to one point in the European Credit Transfer System (ECTS).

2 Decision and approval

This course is was established by the School of Computing 2012-01-16. The course descriptor was revised by School of Computing and applies from 2011-12-05.

3 Objectives

Programming is the basic foundational skill within both Computer Science and applied IT. Computerisation in many industries means that programming is now a basic skill for engineers in many fields. From low-level firmware and control systems that require knowlege of procedural programming in the C language, to larger-scale systems that use the object oriented principles and syntax of C++ for structure and organisation. This course teaches the student how to interpret specifications, fill in missing details and convert the result into working programs that solve given problems. Upon completion of the course the student will understand procedural programming and be familiar with the basics of object oriented design.

4 Content

The course runs over two semesters and covers the following material:

Semester I covers an introduction to basic programming techniques. Data-types from C++ are used to teach the representation of problems in a suitable form for processing. The syntax of C++ is introduced and basic forms of control-flow are used to manipulate data. This part of the course focuses on the relationship between problem specifications and the representation of data within those problems. The connection is used to design programs that manipulate data within those representations in order to solve the problems posed.

Semester II covers medium-scale Software Engineering. Building upon the experience from the first part student's will learn how to combine software components into larger projects. External libraries will be supplied and the work will focus on integrating pre-existing code. This part reinforces understanding of programming by considering larger scale projects than the student could write individually.

5 Learning outcomes

On completion of C Programming For Engineers the student will be able to:

know the syntax and uses of basic language constructs such as datatypes and control-flow.

identify parts of a specification that lack important details, and discover those details by expressing appropriate questions.

translate example techniques onto new problems by recognising similarities and arranging the constructs to fit the unseen problem.

split problems into smaller pieces, based on knowledge of the kind of problem that can be solved, and experimentation with new techniques.

analyse the solutions to sub-problems in order to decide how to combine them into larger pieces and what functionality must be tested to decide if they work or not.

6 Generic skills

The following generic skills are trained in the course:

- Problem solving
- Estimation and planning of work
- Generalisation and Adaption

7 Teaching and learning

Programming is taught directly through structured practical experience. In order for the student to gain this experience they must practice applying knowledge of how a program is constructed to inventing solutions for unfamiliar problems . The new knowledge is obtained primarily from the course text, supported by short

lectures on key topics. Performing the background reading is required to succeed on this course. The application of knowledge occurs in the laboratories and during the student's own periods of self-study. A selection of programming problems provides a structured challenge to guide the student through the course. The workload is designed to ensure that the student must attend all of the *mandatory* laboratory sessions and supply enough hours of self-study to match the pace set by exercises. The skills developed on the course are learned incrementally with each week's material depending on successful completion and understanding of the previous weeks. Lab attendence is mandatory. Progress in the programming exercises is evaluated with three assignments each semester. All assignments have a strict deadline and a single retry attempt within the term; grading will be by viva within five working days of the deadline to ensure that students have rapid feedback on their work. All assessments (assignments and projects) must be passed successfully to continue the course. At the end of each semester a computer based exam is conducted to determine the grades for the course.

8 Assessment and Grading

Code	Module	Credit	Grading
0810	Assignment IA	1cp	U/G
0820	Assignment IB	2cp	U/3/4/5
0830	Part I Exam	4.5cp	U/3/4/5
0850	Project	3cp	U/3/4/5
0860	Part II Exam	4.5cp	U/3/4/5

The final grade for the course will be a U/3/4/5, computed as a weighted average of the component grades rounded to the nearest grade.

9 Course Evaluation

The course responsible will gather feedback from the students. The results of these evaluations will be used to develop and improve the course.

10 Prerequisites

The special prerequisites for this course, besides basic eligibility for university studies, are a pass mark from the following upper secondary school courses: Mathematics C.

11 Field of education and subject area

The course is part of the field of education Technology and is included in the subject area Computer Science.

12 Restrictions regarding degree

The course cannot form part of a degree with another course, the content of which completely or partly corresponds with the contents of this course.

Course Literature.

[1] Walter Savitch *Problem Solving with C++* Addison Wesley, 2012, ISBN13: 978-0-273-75218-9