Course guide
270005 - PRO2 - Programming II

Last modified: 30/01/2024

Unit in charge:	Barcelona School of Informatics
Teaching unit:	723 - CS - Department of Computer Science.
Degree:	BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Compulsory subject).

Academic year: 2023 **ECTS Credits:** 7.5 **Languages:** Catalan, Spanish

LECTURER

Coordinating lecturer: GUILLERMO GODOY BALIL - BORJA VALLES FUENTE - JUAN LUIS ESTEBAN ÁNGELES

Others:

Primer quadrimestre:
M. LUISA BONET CARBONELL - 41
JUAN LUIS ESTEBAN ÁNGELES - 13
GUILLERMO GODOY BALIL - 41, 42, 43
XAVIER MESSEGUER PEYPOCH - 11, 43
BORJA VALLES FUENTE - 11, 12, 13, 42

Segon quadrimestre:
M. LUISA BONET CARBONELL - 11, 51, 52, 53
ALBERT CALVO IBAÑEZ - 51
JOSE CARMONA VARGAS - 42, 61
JORGE CASTRO RABAL - 31
JUAN LUIS ESTEBAN ÁNGELES - 21
PABLO FERNANDEZ DURAN - 13
GUILLERMO GODOY BALIL - 21, 22, 23, 41, 42, 43
SANTIAGO MARCO SOLA - 11, 12, 13, 31, 32, 33
XAVIER MESSEGUER PEYPOCH - 23, 33, 43
ALEJANDRO IVÁN PAZ ORTIZ - 53
MARIA JOSEFINA SIERRA SANTIBAÑEZ - 32, 41, 61, 62, 63
ALFONSO VALVERDE RUIZ - 22, 62
BORJA VALLES FUENTE - 12, 52, 63

PRIOR SKILLS

Students are expected to have been trained in imperative programming techniques based on:
- basic instructions (assignment, alternative and iteration)
- actions and functions, parameter passing and recurrence
- vectors, tuples and sequences
- sequential search and traversal schemes
- basic algorithms (binary search, vector sorting, matrix arithmetics).

They are also expected to know how to use at least one imperative language, preferably C++, and they should have some experience in implementing of C++ programs in the Linux environment.

Furthermore, they should be able to assimilate information from a statement, discuss the correctness of an algorithm and compare algorithmic solutions.

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
CT1.1A. To demonstrate knowledge and comprehension about the fundamentals of computer usage and programming, about operating systems, databases and, in general, about computer programs applicable to the engineering.
CT1.1B. To demonstrate knowledge and comprehension about the fundamentals of computer usage and programming. Knowledge about the structure, operation and interconnection of computer systems, and about the fundamentals of its programming.
CT1.2B. To interpret, select and value concepts, theories, uses and technological developments related to computer science and its application derived from the needed fundamentals of mathematics, statistics and physics. Capacity to understand and dominate the physical and technological fundamentals of computer science: electromagnetism, waves, circuit theory, electronics and photonics and its application to solve engineering problems.
CT3.6. To demonstrate knowledge about the ethical dimension of the company: in general, the social and corporative responsibility and, concretely, the civil and professional responsibilities of the informatics engineer.
CT4.1. To identify the most adequate algorithmic solutions to solve medium difficulty problems.
CT4.2. To reason about the correction and efficiency of an algorithmic solution.
CT5.1. To choose, combine and exploit different programming paradigms, at the moment of building software, taking into account criteria like ease of development, efficiency, portability and maintainability.
CT5.2. To know, design and use efficiently the most adequate data types and data structures to solve a problem.
CT5.3. To design, write, test, refine, document and maintain code in an high level programming language to solve programming problems applying algorithmic schemas and using data structures.
CT5.4. To design the programs¿ architecture using techniques of object orientation, modularization and specification and implementation of abstract data types.
CT8.6. To demonstrate the comprehension of the importance of the negotiation, effective working habits, leadership and communication skills in all the software development environments.

Generical:
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

TEACHING METHODOLOGY

Topics will be explained in a practical way by using many examples.

Theory classes introduce knowledge, techniques and concepts that will be used in laboratory sessions. They also include the presentation and discussion of the solutions of a set of problems.

The two-hour theory classes and the three-hour laboratory sessions will take place weekly.

The programming project integrates knowledge and skills of the entire course, except maybe for the topic (recursive data types) which will be assessed in the final theory exam.

LEARNING OBJECTIVES OF THE SUBJECT

1.To design a class of data with a clear independence between specification and implementation. To justify why an object of the class can only be created, consulted or modified using the operations in the class specification.
2.To solve any exercise which requires the application of a simple algorithm to a vector of objects of a class of data in C++.
3.Given an implementation for a simple class of data, make improvements in its representation and its operations.
4.To explain the main stages of modular design.
5.To identify in a textual statement of a problem data abstractions that may be represented by classes of data which could be used to solve the problem. To check whether any of the abstractions identified has been previously detected, in order to reuse the corresponding class of data.
6.To individually design a modular program in C++ from the data abstractions identified by analysing the statement of a problem. To modify or add some functionality to a given modular program written in C++.
7.To implement a modular program in C++ elegantly and in such a way that other programmers can understand what it does and modify it. To write documentation that facilitates the use of a modular program written in C++ by other programmers.
8.Prepare a C++ program which uses simple data types and C++ classes (some of the predefined and others defined by the student) to be executed. Thw student should be able to do this in two ways: 1) compiling and linking the program using the g++ command; and 2) writing a makefile file, and using it to compile and link the program.
9.To design in teams a modular program in C++ and/or a set of cases (i.e. a set of examples of correct input and corresponding output) to test the full functionality of the program. To debug this program systematically, so that small implementation errors are removed in a reasonable period of time.
10.To know the data types typically used to represent and manage linear data structures and their specification. To design iterative and recursive algorithms for solving search and traversal problems on stacks, queues and lists, using the operations of the corresponding data type and iterators (when appropriated).
11.To know data types used to represent and manage tree data structures and their specification. To design recursive algorithms for solving search and traversal problems about binary, n-ary and general trees, using the operations of the corresponding data type.
12.To describe the main steps in the design of iterative algorithms. To justify the correctness of relatively simple iterative algorithms.
13.To describe the main steps in the design of recursive algorithms. To justify the correctness of relatively simple recursive algorithms.
14.To know what a generalization of a function is, and to be able to explain the difference between specification gneralisations and efficiency generalisations. To know the different types of specification generalizations and the different types of efiiciency generalisations.
15.Given a simple recursive algorithm, to determine whether there is a straightforward way to obtain an equivalent iterative algorithm, and write it if there exists such a straightforward transformation.
16.To distinguish whether the cost of a given iterative or recursive simple algorithm which works on vectors, stacks, queues, lists or trees is linear or if it is quadratic (assuming that the cost is of one of these two types).
17.To determine if the efficiency of a given simple recursive algorithm can be improved and, if it is possible, to design a more efficient recursive algorithm that solves the same problem using efficiency generalizations.
18.To determine if the efficiency of a given simple iterative algorithm can be improved and, if it is possible, to design a more efficient iterative algorithm that solves the same problem.
19.To implement a data structure with specific requirements for its operations and/or the efficiency of such operations, using recursive

STUDY LOAD

Type	Hours	Percentage
Hours large group	30,0	16.26
Guided activities	7,5	4.07
Hours small group	45,0	24.39
Self study	102,0	55.28

Total learning time: 184.5 h

CONTENTS

Linear data structures

Description:
Stacks, queues, lists, maps and sets: specification and use (search and traversal operations). Iterators: definition and use.

Tree data structures

Description:
Binary trees.

Iterative program correctness

Description:
Loop invariants. Justification of the correctness of iterative algorithms.

Recursive programming and correctness of recursive algorithms

Description:
Inductive design of recursive algorithms. Justification of the correctness of recursive algorithms. Generalisation of a function . Specification immersions by weakening the postcondition and strengthening the precondition. Relationship between tail-recursive algorithms and iterative algorithms.

Efficiency enhancements for recursive and iterative programs

Description:
Detection of repeated calculations in recursive and iterative programs. Efficiency generalisations: new data (input parameters) and/or results (return values or output parameters) in recursive operations to improve efficiency. New local variables that use their previous values in iterative operations to improve efficiency.

Modular design and object-oriented design

Description:
Abstraction and the need for abstraction. Functional and data decomposition. Modules. Information hiding. Encapsulation. Modular design phases: difference between specification and implementation. Types of modules and their use. Libraries.

Basic principles of object-oriented design: classes and objects; fields and methods.

Implementing modular designs in C++. Separate compilation and linking. Debugging, testing and documentation of modular programs.

Recursive data types

Description:
Introduction to the use of recursive data types. Pointer type constructor and dynamic memory management. Implementation of linked data structures by means of recursive data types. Iterative and recursive algorithms for solving search and traversal problems in linked data structures by directly accessing the representation based on nodes and node pointers.

ACTIVITIES

C++ review exercises.

Description:
Link to PRO1 contents.

Specific objectives:
2, 8

Full-or-part-time: 6h
Laboratory classes: 3h
Self study: 3h

Linear data structures.

Description:
Development of the corresponding topic and laboratory exercises.

Specific objectives:
10

Full-or-part-time: 17h
Theory classes: 2h
Laboratory classes: 6h
Self study: 9h

Recursive programming

Description:
Development of the corresponding topic.

Specific objectives:
13, 14, 15

Full-or-part-time: 6h
Theory classes: 3h
Self study: 3h

Tree data structures.

Description:
Development of the corresponding topic and laboratory exercises.

Specific objectives:
11

Full-or-part-time: 27h
Theory classes: 2h
Laboratory classes: 9h
Self study: 16h

Iterative program correctness

Description:
Development of the corresponding topic.

Specific objectives:
12

Full-or-part-time: 4h
Theory classes: 2h
Self study: 2h

Efficiency enhancements for recursive and iterative programs.

Description:
Development of the corresponding topic.

Specific objectives:
12, 13, 14, 15, 16, 17, 18

Full-or-part-time: 24h
Theory classes: 3h
Laboratory classes: 9h
Self study: 12h

Introduction to modular design and object-oriented design.

Description:
Development of the corresponding topic.

Specific objectives:
1, 2, 4, 21

Full-or-part-time: 8h
Theory classes: 4h
Self study: 4h

Specification and use of classes of objects in C++.

Description:
Exercises from the corresponding topic.

Specific objectives:
1, 2, 21

Full-or-part-time: 9h
Laboratory classes: 3h
Self study: 6h

Implementing classes of objects in C++.

Description:
Exercises from the corresponding topic.

Specific objectives:
1, 3, 21

Full-or-part-time: 6h
Laboratory classes: 3h
Self study: 3h

Supervision of practical

Description:
Supervision of the design and implementation of the practical.

Specific objectives:
1, 3, 5, 6, 7, 8, 10, 11, 16, 17, 21

Full-or-part-time: 9h
Laboratory classes: 3h
Self study: 6h

Recursive data types.

Description:
Development of the corresponding topic.

Specific objectives:
1, 3, 16, 17, 18, 19, 20

Full-or-part-time: 32h 30m
Theory classes: 7h 30m
Laboratory classes: 9h
Self study: 16h

Review of theory and exam problems.

Description:
Questions can be asked about the topics covered in theory classes.

Full-or-part-time: 4h
Theory classes: 2h
Self study: 2h

EX_SIMUL1

Description:
Mock exam for midterm exam 1

Specific objectives:
2, 3, 5, 6, 7, 8, 10, 11, 16, 17, 18, 19, 20

Full-or-part-time: 4h
Guided activities: 2h
Self study: 2h

EX_PAR1_TP

Description:
Mid-term theory/problems exam.

Specific objectives:
2, 3, 5, 6, 7, 8, 10, 11, 16, 17, 18, 19, 20

Full-or-part-time: 7h
Guided activities: 2h
Self study: 5h

CODI_DOC_PRAC1

Description:
Delivery of the programming project code and documentation.

Specific objectives:
1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 21

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 10h
Self study: 10h

EX_SIMUL2

Description:
Mock exam for midterm exam 2

Specific objectives:
2, 3, 5, 6, 7, 8, 10, 11, 16, 17, 18, 19, 20

Full-or-part-time: 5h
Guided activities: 2h 30m
Self study: 2h 30m

EX_PAR2_TP

Description:
Final theory/problems exam.

Specific objectives:
2, 3, 5, 6, 7, 8, 10, 11, 16, 17, 18, 19, 20

Full-or-part-time: 9h
Guided activities: 2h 30m
Self study: 6h 30m

GRADING SYSTEM

The technical competence mark (NCTEC) is calculated as follows:

NCTEC = 0.3*EXAM1 + 0.3*EXAM2 + 0.25*PRAC + 0.15*DELIVERY

where

* EXAM1 and EXAM2 are the marks of the mid-term and final theory exams

* PRAC is the mark of the programming project; it may have an automatic part, derived from code testing, and a manual part.

* DELIVERY is the mark arising from the deliveries from students of solutions to exercises requested over the course.

Nevertheless, NCTEC will be NP if the weight of the assessment activities with an NP mark is greater than or equal to 70%.

The assessment of the generic competence Teamwork is obtained from some of the deliveries carried out along the course (DELIVERY) which will be done in teams.

BIBLIOGRAPHY

Basic:
- Alquezar, René; Valles, Borja; [et al.]. Apunts de teoria de PRO2.
- Valles, Borja; [et al.]. Sessions de Laboratori de PRO2.
- Cortadella, Jordi; Rubio, Albert; Valentin, Luis. Iniciació a la Programació (notes de curs). 2001.
- Weiss, M.A. Data structures and problem solving using C++. 2nd ed. Pearson Education International, 2003. ISBN 0321205006.
- Musser, D.R.; Derge, G.J.; Saini, A. STL tutorial and reference guide: C++ programming with the standard template library. 2nd ed. Addison-Wesley publishing company, 2000. ISBN 0201379236.

Complementary:
- Castro, J. [et al.]. Curs de programació. McGraw-Hill, 1992. ISBN 844810031X.
- Balcazar, J.L. Programación metódica. McGraw-Hill, 1993. ISBN 8448119576.
- Savitch, Walter J; Mock, Kenrick. Problem solving with C++. Tenth edition. Pearson, [2018]. ISBN 9780134448282.

RESOURCES

Hyperlink:
- http://www.cplusplus.com/reference/stl/- http://c.conclase.net/curso/index.php- http://www.cs.upc.edu/~pro2