COMPUTER PROGRAMMING 1

Grado en Computación e Inteligencia Artificial / Bachelor in Computer Science and Artificial Intelligence BCSAI SEP-2025 CP1-CSAI.2.M.A

Area Computer Science

Number of sessions: 30

Academic year: 25-26

Degree course: SECOND

Number of credits: 6.0

Semester: 1º

Category: BASIC

Language: English

Professor: **SUZAN T S AWINAT**

E-mail: suzant@faculty.ie.edu

Adjunct Professor | Women in Robotics & STEM

PhD candidate in computer engineering.

Research areas: Pattern recognition, ArabicNLP, genetic programming, evolutionary algorithms, Semantics and Robotics.

Office Hours

Office hours will be on request. Please contact at:

My Office: T-05.09 , IE Tower.

You can always find me there from 9:00am -18:00pm unless I have classes.

Or contact me on Teams: suzant@faculty.ie.edu

SUBJECT DESCRIPTION

This course strengthens students' C programming skills, introducing advanced memory management, multi-threading, and networking concepts. Students will build high-performance, networked C applications and explore systems programming, embedded development, and AI-based optimizations.

This course takes a deep dive through modern C programming, polishing your skills while learning?about best practices, helpful tools and full programming techniques!

Software stack:

CMake for building
GTest for testing
Conda for dependencies
VSCode for editing

LEARNING OBJECTIVES

- Master dynamic memory allocation and optimization techniques.
- Implement data structures using pointers efficiently.
- Develop multi-threaded and networked applications in C.
- Apply low-level systems programming concepts (process management, IPC).
- Debug and optimize C programs for performance and security.
- Write networked applications using socket programming in C.

TEACHING METHODOLOGY

IE University teaching method is defined by its collaborative, active, and applied nature. Students actively participate in the whole process to build their knowledge and sharpen their skills. Professor's main role is to lead and guide students to achieve the learning objectives of the course. This is done by engaging in a diverse range of teaching techniques and different types of learning activities such as the following:

1. **Lecture-based Teaching:** The course will mainly be taught through lectures that provide a conceptual understanding of C programming, including syntax, control structures, functions, arrays, pointers, and data structures. The lectures will be interactive and include examples and explanations of programming concepts.
2. **Hands-on Practice:** Students will have the opportunity to practice programming exercises and assignments during the class. This will help them gain hands-on experience in writing and debugging C code using proper programming techniques. Students will be encouraged to work in groups and share their knowledge and experience with their peers.
3. **Assignments:** Regular assignments will be given to students to reinforce the programming concepts learned in class. The assignments will be designed to challenge students to think creatively and solve problems related to computer science and AI. Feedback on the assignments will be provided to help students improve their programming skills.
4. **Online Resources:** Students will be given access to online resources such as tutorials, videos, and programming exercises to help them practice programming outside of class. These resources will also provide additional explanations and examples to supplement the lecture-based teaching.
5. **Project Based Learning:** Students will work on a programming project throughout the course, applying the concepts and techniques they learned in class to solve real-world problems related to computer science and AI. This project will be done in groups and will

involve identifying a problem, designing a solution, and implementing it using C programming language. The project will also require students to present their work to the class and receive feedback.

6. **Peer Learning:** Students will be encouraged to work in groups and share their knowledge and experience with their peers. This will foster a collaborative learning environment where students can learn from each other and develop teamwork skills.
7. **Assessments:** Regular assessments such as quizzes, tests, and exams will be conducted to evaluate students' understanding of programming concepts and techniques. These assessments will provide feedback to students and help them identify areas where they need to improve.

Overall, the teaching methodology for Computer Programming 1 will be a combination of lecture-based teaching, hands-on practice, assignments, online resources, project-based learning, peer learning, and assessments. This methodology will provide a comprehensive learning experience that will help students develop their programming skills and prepare them for advanced programming courses in computer science and AI.

Learning Activity	**Weighting**	**Estimated time a student should dedicate to prepare for and participate in**
Lectures	26.7 %	40.0 hours
Discussions	10.0 %	15.0 hours
Exercises in class, Asynchronous sessions, Field Work	36.7 %	55.0 hours
Group work	16.7 %	25.0 hours
Individual studying	10.0 %	15.0 hours
TOTAL	100.0 %	150.0 hours

AI POLICY

In today’s world, generative artificial intelligence (GenAI) is changing how we work, study and, in general, how we get things done. However, in the context of this course, the use of GenAI is not permitted, unless it is otherwise stated by the instructor. The use of GenAI tools would jeopardize the students’ ability to acquire fundamental knowledge or skills of this course.

If a student is found to have used AI-generated content for any form of assessment, it will be considered academic misconduct, and the student might fail the respective assignment or the course.

PROGRAM

SESSION 1 (LIVE IN-PERSON)

Module 0/6: C Language Basics refresh

Name Session: Course Overview and general C language review

Session Content:
In this session we'll presents the general contents of the course and also will do a fast refresh of C language basics and environment setup

VSCode refresher
GitHub Education /Copilot in VS Code

SESSION 2 (LIVE IN-PERSON)

Module 0/6: C Language Basics refresh

Name Session: Tooling

Session Content:
We will go over the tooling for the course, mainly:

The terminal
Conda for dependencies
CMake intro (just a simple add_executable and add_library perhaps) and compiling

The workflow intended for programming in the course will also be introduced (navigating via the terminal to somewhere, activating/creating a conda env, opening VSCode, running cmake…)

SESSION 3 (LIVE IN-PERSON)

Module 1/6: Error Handling and Debugging

Subtopic 1/3: Error Handling and Assertions

Name Session: Error Handling and Assertions

Session Content:
In this session, students will learn about error handling techniques and assertions in the C programming language, and at the same time refresh debugging techniques knowledge.

SESSION 4 (LIVE IN-PERSON)

Module 1/6: Error Handling and Debugging

Subtopic 2/3: Unit Testing

Name Session: Unit Testing

Session Content:

Unit testing
GTest
Setting up tests in CMake

SESSION 5 (LIVE IN-PERSON)

Module 2/6: Pointers and Structures

Subtopic 1/3: Pointers

Name Session: Pointers

Session Content:

We'll refresh the concept of pointers, their purpose, and their relationship with memory addresses.

We'll also talk about pointers declaration, initialization, referencing and dereferencing.

Finally we'll also work with pointer arithmetic, how to use and why this is useful.

SESSION 6 (LIVE IN-PERSON)

Module 2/6: Pointers and Structures

Subtopic 2/3: Dynamic Memory Allocation and Memory Management

Name Session: Dynamic Memory Allocation and Memory Management

Session Content:
In this session, students will dive into the concept of dynamic memory allocation and learn how to effectively manage memory in C programming. Dynamic memory allocation allows programs to allocate and deallocate memory at runtime, providing flexibility in memory usage and enabling efficient memory management.

SESSION 7 (LIVE IN-PERSON)

Module 2/6: Pointers and Structures

Subtopic 3/3: Structure and Unions

Name Session: Structure and Unions

Session Content:
In this session, students will dive into the concept of structures and unions in the C programming language. Structures and unions are powerful data constructs that allow the organization and manipulation of related data elements in a program.

Intro to Data-oriented programming

SESSION 8 (LIVE IN-PERSON)

Module 3/6: File Handling and Preprocessor Directives

Subtopic 1/3: File Input/Output

Name Session: File Input/Output

Session Content:

Input/Output
Dealing with files
Error handling and resource management with files

SESSION 9 (LIVE IN-PERSON)

First midterm

SESSION 10 (LIVE IN-PERSON)

Module 3/6: File Handling and Preprocessor Directives

Subtopic 3/3: Preprocessor Directives and Macros

Name Session: Preprocessor Directives and Macros

Session Content:
In this session, students will delve into the world of preprocessor directives and macros in the C programming language. Preprocessor directives are commands that instruct the C compiler to perform certain actions before the actual compilation process begins. Macros, on the other hand, are a way to define reusable code snippets or constants that can be expanded inline during the compilation process.

SESSION 11 (LIVE IN-PERSON)

Module 4/6: Advanced Concepts in C

Subtopic 1/4: Bit Manipulation

Name Session: Bit Manipulation

Session Content:
In this session on bit manipulation, students will dive into the world of binary operations and learn how to manipulate individual bits of data in C programming. Bit manipulation is a powerful technique that allows programmers to perform operations at the bit level, enabling efficient and optimized solutions to a variety of programming problems.

Bit manipulation (Carmack`s fast rsqrt, Morton hash for spatial hashing)
1 vs. 2 complement
Using enums as flags

SESSION 12 (LIVE IN-PERSON)

Module 4/6: Advanced Concepts in C

Subtopic 2/4: Enumerations and Typedef in C

Name Session: Enumerations and Typedef in C

Session Content:
In this session, we will explore the concepts of enumerations and typedef in the C programming language. Enumerations provide a way to define a set of named constants, while typedef allows us to create custom data type aliases. Understanding and effectively using enumerations and typedef can enhance code readability, maintainability, and efficiency.

SESSION 13 (LIVE IN-PERSON)

Module 4/6: Advanced Concepts in C

Subtopic 3/4: Linked Lists

Name Session: Linked Lists

Session Content:
In this session, students will delve into the world of linked lists, a fundamental data structure widely used in programming and particularly useful in C. Linked lists provide a dynamic way to store and organize data, allowing for efficient insertion, deletion, and traversal operations.

SESSION 14 (LIVE IN-PERSON)

Module 4/6: Advanced Concepts in C

Subtopic 4/4: Recursion and Backtracking

Name Session: Recursion and Backtracking

Session Content:
In this session, we will delve into the powerful concepts of recursion and backtracking in the context of the C programming language. Recursion involves solving a problem by breaking it down into smaller, simpler instances of the same problem. Backtracking, on the other hand, is a technique used to systematically explore all possible solutions to a problem by incrementally building and undoing partial solutions.

SESSION 15 (LIVE IN-PERSON)

Module 5/6: Concurrent programming

Subtopic 1/2: Thread Concepts & Lifecycle

Name Session: Concurrent Programming: Thread Concepts & Lifecycle

Session Content:
In this sections we'll learn how to take advantage of platforms with several processors or cores, where different parts of a program can be executed simultaneously. We'll explain the basics of concurrent programming using the multithread capabilities accessible via C language.

SESSION 16 (LIVE IN-PERSON)

Module 5/6: Concurrent programming

Subtopic 2/2: Advanced thread management

Name Session: Concurrent Programming: Advanced thread management

Session Content:
During this section we'll learn how to deal with common thread issues like race-conditions, shared data, critical data and critical sections, in order to build reliable and optimized programs.

POSIX threads, semaphores and barriers
Overview of higher level constructs (OpenMP, MPI)

SESSION 17 (LIVE IN-PERSON)

Module 6/6: Networking and sockets

Subtopic 1/4: Socket programming in C for network communication

Name Session: Socket programming in C for network communication

Session Content:
In this section we'll start introduction some networking concepts that will allow you building programs able to communication across network. We'll continue explaining what a socket is, and how to use Socket API to get able to enable network communication.

SESSION 18 (LIVE IN-PERSON)

Module 6/6: Networking and sockets

Subtopic 2/4: Client-server architectures using TCP and UPD protocols

Name Session: Client-server architectures using TCP and UPD protocols

Session Content:
In this section we'll start with an Introduction to Network Protocols, then we will dive into the client-Server architecture, and the use of different network protocols focusing on: TCP Protocol and Socket I/O Operations AND UDP Protocol and Socket I/O Operations.

SESSION 19 (LIVE IN-PERSON)

Module 6/6: Networking and sockets

Subtopic 3/4: Secure connections with sockets

Name Session: Secure connections with sockets

Session Content:
In this section we'll learn how to secure network connections, using standard protocols like HTTPS and TSL via the C programming language. We'll introduce encryption and certificate basics for a better understanding of how TLS protocol is able to secure our connections.

SESSION 20 (LIVE IN-PERSON)

Name Session: Mid Exam

SESSION 21 (LIVE IN-PERSON)

Module 6/6: Networking and sockets

Subtopic 4/4: Networked applications

Name Session: Networked applications

Session Content:
In this section we'll review different real world networked applications that can be implemented with the different concepts we've learnt during the course, like webservers.

SESSION 22 (LIVE IN-PERSON)

Name Session: GroupWork Session

Session Content:

Group Work is an essential component of the C programming course as it promotes collaboration, communication, and problem-solving skills. Through group work, students can work together, share ideas, and collectively tackle programming challenges. Group Work also simulates real-world scenarios where software development is often a collaborative effort.

Activity: Collaborative Program Development
Objective: The objective of this group work activity is to develop a C program collaboratively, utilizing the skills and concepts learned throughout the course. The activity will require students to work together, dividing tasks and combining their individual contributions to create a cohesive and functional program.

Description:

Group Formation:

Students will be divided into groups, typically comprising 3–5 members, depending on the class size.

The groups can be either self-selected or assigned by the instructor, considering factors such as diversity of skills and knowledge.

Program Selection:

Each group will choose a program topic or problem to solve using C programming.

The program should be of moderate complexity, challenging enough to utilize various programming concepts covered in the course.

Planning and Design:

The group will collectively brainstorm and plan the overall structure and design of the program.

This includes defining the program's objectives, identifying required input and output, and outlining the individual tasks to be assigned.

Task Allocation:

The group will assign specific programming tasks to individual members based on their strengths and interests.

Tasks can include module implementation, function development, debugging, testing, documentation, and overall program integration.

Collaboration and Code Integration:

Group members will work individually on their assigned tasks.

Regular communication and collaboration among group members are encouraged to ensure the seamless integration of individual code contributions.

Testing and Debugging:

Once the individual tasks are completed, the group will conduct comprehensive testing of the program to identify and resolve any issues or bugs.

Debugging efforts will involve systematic error identification and troubleshooting to ensure the program's functionality and correctness.

Documentation:

The group will collectively document the program, including a description of its functionality, input/output specifications, and any special considerations or limitations.

Documentation should also cover the program's design choices, algorithms used, and individual contributions from each group member.

SESSION 23 (LIVE IN-PERSON)

Name Session: GroupWork Session

Session Content:

Group Work is an essential component of the C programming course as it promotes collaboration, communication, and problem-solving skills. Through group work, students can work together, share ideas, and collectively tackle programming challenges. Group Work also simulates real-world scenarios where software development is often a collaborative effort.

Activity: Collaborative Program Development
Objective: The objective of this group work activity is to develop a C program collaboratively, utilizing the skills and concepts learned throughout the course. The activity will require students to work together, dividing tasks and combining their individual contributions to create a cohesive and functional program.

Description:

Group Formation:

Students will be divided into groups, typically comprising 3–5 members, depending on the class size.

The groups can be either self-selected or assigned by the instructor, considering factors such as diversity of skills and knowledge.

Program Selection:

Each group will choose a program topic or problem to solve using C programming.

The program should be of moderate complexity, challenging enough to utilize various programming concepts covered in the course.

Planning and Design:

The group will collectively brainstorm and plan the overall structure and design of the program.

This includes defining the program's objectives, identifying required input and output, and outlining the individual tasks to be assigned.

Task Allocation:

The group will assign specific programming tasks to individual members based on their strengths and interests.

Tasks can include module implementation, function development, debugging, testing, documentation, and overall program integration.

Collaboration and Code Integration:

Group members will work individually on their assigned tasks.

Regular communication and collaboration among group members are encouraged to ensure the seamless integration of individual code contributions.

Testing and Debugging:

Once the individual tasks are completed, the group will conduct comprehensive testing of the program to identify and resolve any issues or bugs.

Debugging efforts will involve systematic error identification and troubleshooting to ensure the program's functionality and correctness.

Documentation:

The group will collectively document the program, including a description of its functionality, input/output specifications, and any special considerations or limitations.

Documentation should also cover the program's design choices, algorithms used, and individual contributions from each group member.

SESSION 24 (LIVE IN-PERSON)

Programming paradigms (mainly OOP)

SESSION 25 (LIVE IN-PERSON)

Programming paradigms (mainly OOP)

SESSION 26 (LIVE IN-PERSON)

Advanced concepts (LAB) depending on the progress of the course

SESSION 27 (LIVE IN-PERSON)

Name Session: Revision ..... :)

SESSION 28 (LIVE IN-PERSON)

Name Session: Final Project Presentation and Evaluation

Session Content:
Students will present their final projects to the instructor and peers. The projects will be evaluated based on functionality, code quality, design principles, and adherence to best practices learned throughout the course.

SESSION 29 (LIVE IN-PERSON)

Name Session: Final Project Presentation and Evaluation

Session Content:
Students will present their final projects to the instructor and peers. The projects will be evaluated based on functionality, code quality, design principles, and adherence to best practices learned throughout the course.

SESSION 30 (LIVE IN-PERSON)

Name Session: Final Exam

Session Content:
Final written exam, consisting of multiple-choice and coding sections.

EVALUATION CRITERIA

criteria	percentage	Learning Objectives	Comments
Final Exam	50 %		
Class Participation	10 %		
Intermediate tests	20 %		
Group Project	20 %		

RE-SIT / RE-TAKE POLICY

BIBLIOGRAPHY

Compulsory

- Brian W. Kernighan and Dennis M. Ritchie. (1988). *The C programming Language.* 2nd. Prentice-Hall. ISBN 0131103628 (Digital)

"The C Programming Language" is a timeless classic that has been a staple resource for programmers since its first publication in 1978. The second edition, authored by Brian W. Kernighan and Dennis M. Ritchie, the creators of the C programming language itself, further solidifies its reputation as the definitive guide for learning and mastering C.
https://venkivasamsetti.github.io/ebookworm.github.io/Books/cse/C%20Programming%20Language%20(2nd%20Edition).pdf

- Steve Oualline. (1997). *Practical C Programming,.* 3rd Edition. O'REILLY. ISBN 1565923065 (Digital)

Practical C Programming, written by Steve Oualline, is a comprehensive and practical guide to the C programming language. This highly regarded book has been updated to its 3rd edition, providing readers with valuable insights into writing efficient, reliable, and portable C code. Designed for both beginner and experienced programmers, Practical C Programming starts with the basics and gradually progresses to more advanced concepts. The book offers a hands-on approach, emphasizing practical exam

https://repo.zenk-security.com/Programmation/O%20Reilly%20-%20Practical%20C%20Programming,%203rd%20Edition.pdf

- Lewis Van Winkle. (2019). *Hands-On Network Programming with C.* Packt Publishing. ISBN 9781789349863 (Digital)

https://github.com/bilalmohib/ProgrammingBooks/blob/crystal/hands-on-network-programming-with-c-learn-socket-programming-in-c-and-write-secure-and-optimized-network-code-true-pdf-1nbsped-9781789349863_compress.pdf

BEHAVIOR RULES

Please, check the University's Code of Conduct here. The Program Director may provide further indications.

ATTENDANCE POLICY

Please, check the University's Attendance Policy here. The Program Director may provide further indications.

Attendance Policy:Hola, solo necesitamos que añadas el resit/retake policy, por favor. Si lo necesitas, puedes usar este texto:

Each student has four chances to pass any given course distributed over two consecutive academic years: ordinary call exams and extraordinary call exams (re-sits) in June/July.

Students who do not comply with the 80% attendance rule during the semester will fail both calls for this Academic Year (ordinary and extraordinary) and have to re-take the course (i.e., re-enroll) in the next Academic Year.

Evaluation criteria:

Students failing the course in the ordinary call (during the semester) will have to re-sit the exam in June / July (except those not complying with the attendance rule, who will not have that opportunity and must directly re-enroll in the course on the next Academic Year).

The extraordinary call exams in June / July (re-sits) require your physical presence at the campus you are enrolled in (Segovia or Madrid). There is no possibility to change the date, location or format of any exam, under any circumstances. Dates and location of the June / July re-sit exams will be posted in advance. Please take this into consideration when planning your summer.

The June / July re-sit exam will consist of a comprehensive exam. Your final grade for the course will depend on the performance in this exam only; continuous evaluation over the semester will not be taken into consideration. Students will have to achieve the minimum passing grade of 5 and can obtain a maximum grade of 8.0 (out of 10.0) – i.e., “notable” in the in the re-sit exam.

Retakers: Students who failed the subject on a previous Academic Year and are now re-enrolled as re-takers in a course will be needed to check the syllabus of the assigned professor, as well as contact the professor individually, regarding the specific evaluation criteria for them as retakers in the course during that semester (ordinary call of that Academic Year). The maximum grade that may be obtained in the retake exam (3rd call) is 10.0.

After ordinary and extraordinary call exams are graded by the professor, you will have a possibility to attend a review session for that exam and course grade. Please be available to attend the session in order to clarify any concerns you might have regarding your exam. Your professor will inform you about the time and place of the review session. Any grade appeals require that the student attended the review session prior to appealing.

Students failing more than 18 ECTS (single degree) or 21 ECTS (dual degree) in the academic year after the June-July re-sits will be asked to leave the Program. Please, make sure to prepare yourself well for the exams in order to pass your failed subjects.

In case you decide to skip the opportunity to re-sit for an exam during the June / July extraordinary call, you will need to enroll in that course again for the next Academic Year as a re-taker and pay the corresponding extra cost. As you know, students have a total of four allowed calls to pass a given subject or course, in order to remain in the program.

ETHICAL POLICY

Please, check the University's Ethics Code here. The Program Director may provide further indications.