

CORE COURSES

Computer Science Concepts

Programming in Java:

Variables, control structures, methods, classes, input/output; object orientation, interfaces, inheritance; introduction to graphical user interfaces. Introduction to computer systems, system software and basic Unix.

Data Structures & Algorithms

Program development techniques including basic ideas of correctness; representation of lists, stacks, queues, sets, trees and hash tables. Notions of complexity and analysis; notion of abstract data type; sets and sequences as examples; searching and information retrieval illustrated with a 'table' abstract data type; various representations of a 'table' abstract data type; recursion.

CORE COURSES (ADVANCED)

Computer Systems

Information storage representation, Memory organisation and hierarchy, Processor fundamentals, assembler programming, assembler operation, subroutine calling mechanisms, linking/loading, Input-output and device controllers requirements for supporting an operating system and device drivers.

CORE COURSES (LEVEL 3)

Software Engineering and Project

This course in software engineering provides an introduction to the production of high quality software solutions to large tasks. Among the topics covered in this course are the following: models of the software life-cycle, requirements analysis and specification, program design techniques and paradigms, software specification techniques, configuration management and version control, quality assurance, integration and testing, project management, risk analysis, case study of ethical considerations in Software Engineering.

ELECTIVES

Advanced Programming Paradigms

[A selection of topics from the following:](#)

Fundamental models of computation, illustrated by the lambda calculus.

[Different approaches to programming:](#)

Functional and logic paradigms. Fundamental concepts of programming languages, including abstraction, binding, parameter passing, scope, control abstractions.

[Programming models expressed via Scheme:](#)

Substitution model; map/reduce programming; environment model; object oriented model; a compositional programming model.

[Introduction to parallel computing:](#)

Data parallelism, Java threads, and relationship to distributed computing.

[Examples in application:](#)

Map/reduce programming in Google; flow-oriented programming for composition of web-services. Ontologies in the semantic web.

Computer Architecture

Fundamentals of computer design; quantifying cost and performance; instruction set architecture; program behaviour and measurement of instruction set use; processor datapaths and control; pipelining, handling pipeline hazards; memory hierarchies and performance; I/O devices, controllers and drivers; I/O and system performance.

Computer Networks and Applications

[Introduction to networks and digital communications with a focus on Internet protocols:](#)

Network layer model, Internet application architectures (client/server, peer-to-peer) and protocols (HTTP-web, SMTP-mail, etc), Transport protocols: UDP, TCP (reliable transport, congestion and flow control), IP (routing, addressing), Data Link layer operation (Ethernet, 802.11, PPP), selected current topics such as: security, multimedia protocols, Quality of Service, mobility, wireless networking, emerging protocols (IPv6).

Distributed Systems

[A selection of topics from the following: the challenges faced in constructing client/server software:](#)

Partial system failures, multiple address spaces, absence of a single clock, latency of communication, heterogeneity, absence of a trusted operating system, system management,

binding and naming. Techniques for meeting these challenges: RPC and middleware, naming and directory services, distributed transaction processing, 'thin' clients, data replication, cryptographic security, mobile code. Introduction to Java RMI.

Event Driven Computing

Event driven paradigm:

Finite State Automata, their behavior and implementation. Correspondence with regular expressions. Examples of embedded systems. Introduction to interconnected state machines, Petri Nets, and concurrency. Concepts of state-space and relationship to testing.

Building Graphical User Interfaces:

Model view controller paradigm. Introduction to design patterns for managing complexity in large systems. Building GUIs with the Java Swing library. Comparison/contrast with other GUI toolkits. Ease of use and human-computer interaction issues. Practical projects cover the use of FSAs for control logic and GUI design.

Operating Systems

OS purposes:

Resource management and the extended virtual computer; historical development. Processes: critical sections and mutual exclusion, semaphores, monitors, classical problems, deadlock; process scheduling. Input and Output: hardware and software control. Memory management: multi-programming; swapping; virtual memory, paging and symbolic segmentation;

File System:

Operations, implementation, performance.

Protection mechanisms:

Protection domains, access lists, capability systems, principle of minimum privilege.

Distributed systems:

Communication, RPC, synchronisation, distributed file systems, authentication.

Programming Techniques

Program development:

Methods of specification, design, implementations, testing and debugging, case studies, Graphs: construction, traversal, topological sorting, applications. Sorting and searching: internal and external algorithms, correctness and complexity analysis.