
CS 241

Control Structures

Christopher A. Gantz

SPS Undergraduate Program
Regis University
cgantz@regis.edu

Lecture 1b

Overview of CS 241: Course Overview

Christopher A. Gantz
School of Professional Studies
Regis University
cgantz@regis.edu

Course Overview

- Intro to Problem Solving
- Program Format
- Simple Data Structures
- Basic Output
- Arithmetic Elements & Operations
- Basic Input
- Writing a Complete Program
- Selection Structure

Course Overview (Cont.)

- Nesting Control Structures (IF/Case)
- Debugging Techniques
- Repetition Structures
- Nesting (Loops)
- Procedures
- Functions
- Modular Programs
- Recursion

Computer Science Overview

- Reading
 - Nance textbook Pages 1 to 27
- Computer Science vs Computer Literacy
- Computer Architecture
- Computer Languages

Computer Science vs Computer Literacy

- Computer Science
 - Mathematics & Logic
 - Science
 - Engineering
 - Communication
 - Interdisciplinary application
- Computer Literacy
 - Utilization of various types of applications
 - Word processors
 - Databases
 - Internet browsers
 - Personal Finance
 - Knowledge of computer related technology terminology

Mathematics and Logic of CS

- Algorithm development to define a solution to a problem
 - An algorithm specifically defines a process/protocol/recipe that describes how to accomplish a task
 - An algorithm is composed of the following attributes
 - Finite ordered sequence of steps/activities
 - Input data that is consumed
 - Output data that is produced
 - Instructions or statements that manipulate the state of computation
 - An Algorithm is a formal mechanism utilized to begin the codification of a solution to a problem

Mathematics and Logic of CS

- Demonstrates that Computer Scientists are knowledge engineers (KE's)
- Where KE's must correctly and accurately comprehend the problem:
 - Domain or operating environment
 - Implicit internal & external requirements/constraints
- After the algorithm has been determined and developed it is then translated and expressed in a form that the computer can understand
- This is achieved by implementing the algorithm in some specific programming language like Pascal

Mathematics and Logic of CS

- Solution must be formulated, expressed and implemented in the appropriate programming language
 - Where a programming language is simply a formal language that implements an algorithm in terms of specific instructions that the computer system can both understand and process
 - E.g. Pascal, C, perl etc.
- These activities are synonymous with the development of a mathematical proof or formal argument in the field of mathematics of logic

Science of CS

- As the computer scientist formulates a solution he/she must
 - Formulate a hypothesis that maps to a solution
 - Carefully test the hypothesis to empirically prove correctness
 - Utilize mathematics within the solution and to test the solution
 - Understand the limitations of the solution
 - Acquire a deep understanding of the problem

Engineering of CS

- The Computer Scientist must:
 - Comprehend the scientific principles of computer science
 - Posses experiential knowledge of implementation techniques
- These primary skills are required to:
 - Develop innovative model's to solve a problem
 - Apply the engineering design process
 - Define and manipulate information structures

Communication within CS

- Why is communication critical
 - The computer scientist must be able to:
 - Define a problem clearly
 - Document a solution effectively for a variety of users
 - Communicate solutions to:
 - Colleagues
 - Professionals in other fields
 - General public
 - Maintain the solution throughout it's lifecycle in a manner that reduces costs

CS as an Interdisciplinary solution provider

- Computer Science is used to provide solutions in a variety of interdisciplinary fields such as:
 - Mathematics
 - Finance
 - Economics
 - Linguistics
 - Biology
- Therefore the computer scientist is required to be an ever changing quasi-expert of a variety of interdisciplinary fields

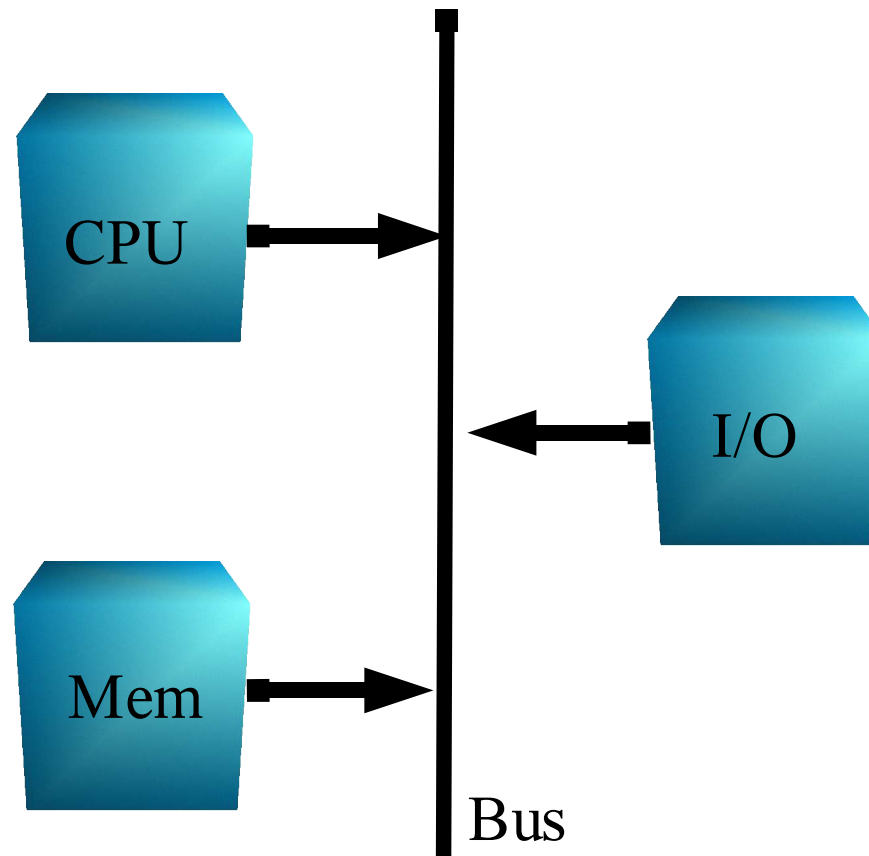
Computer Architecture

- Computer Architecture defines the components and interconnection topology that define a computation engine/machine
- The common components of a computer architecture are:
 - Processing Unit
 - ALU, Floating Point Unit
 - Memory, which include:
 - Onboard memory and internal registers
 - Input Output devices which include:
 - Disks, tapes, monitors, keyboards, microphones, speakers etc.

Computer Architecture (Cont.)

- The most common interconnection topology deployed within a computer architecture is that of a bus
- Where the bus typically utilizes one of two types of communication and access protocols:
 - Parallel
 - Serial

Computer Architecture (Cont.)



Computer Architecture (Cont.)

- Types/categories of computer architectures
 - Mainframe
 - Supercomputer
 - Minicomputer
 - Microcomputer/PC
 - Workstation
- A computer system/solution is composed of:
 - Hardware (CPU, Memory, Storage)
 - Software (Applications e.g. Netscape, StarOffice, MySQL)

Computer Languages

- The computer language is the formal mechanism utilized to translate an algorithm into a program
- A program is simply a set of related instructions that implement some algorithm
- The program defines what should be executed in order to implement a solution

Types of Computer Languages

- Machine Language
 - Streams of bits composed of 0's and 1's
- Assembly Language
 - Symbolic mapping of names to specific sized bit patterns
- High level Language
 - E.g. Pascal, C, C++ and Perl

Syntax vs Semantics

- All computer languages contain the following two primary abstract attributes:
 - Syntax
 - Rules for defining correct words and sentences of the language
 - Semantics
 - Defines the meaning of those words and sentence within it's immediate context

Processing of Computer Languages

- Computer languages are processed by translating one form into another until the original description is transformed into the language the computer system understands
- There are typically 3 forms of computer languages which correspond to the common types/categories of language
 - Source code (typically the high level language like Pascal)
 - Assembly code
 - Object code (Computer Architecture machine language)

Processing of Computer Languages

- Each level of translation is accomplished by a specific program or system software application
- There are two specific system software programs
 - The compiler
 - The assembler
- Translation details of the two system software programs
 - Compiler: Source code to Assembly code
 - Compiler: Source code to Object code
 - Assembler: Assembly code to Object code

Why learn Pascal?

- Pascal's strengths
 - It's program structure is a reasonable approximation of English
 - It supports the use of descriptive words for variable and data types
 - It facilitates good problem-solving habits
 - Comprehension and application of structured programming
 - Remember the purpose of this course is to teach the skill of developing solutions to problems by defining and implementing algorithms that are translated into a computer program or software application