Computer Programming: Introduction

Asst. Prof. Dr. Yalçın İşler
Izmir Katip Celebi University

Contact Information

- My office
- My e-mail: islerya@yahoo.com
- Note that you are forbidden to contact me via
  - My phone numbers
  - My Facebook account
  - Any Messenger programs

  about any subject related to this course ☺

Course website

- http://me.islerya.com

  using the Main Menu on the left side, click

  → for Students,

  → Lecture Notes,

  → Spring Semester ,

  → Introduction to Programming.

- Or simply, go http://me.islerya.com/matlab.php

Office hours

- By appointment via:
  - E-mail
  - Speak to me in class
  - Information also will be contained on website
    - (Not currently up, expect mostly finalized soon)

Coursework

- Processing of the course
  - Presentations (2 hours/session)
  - Practical laboratory exercises (2 hours/session)
  - 4-hour session/week in total ☺
- Evaluation
  - Mid-term exam (%40)
  - Homeworks (%20)
  - Final (or Make-Up) Exam (%40)

References

- "MATLAB: An Introduction With Applications," 3rd edition from Amos Gilat TEXTBOOK
- "Her Yonuyle Matlab" from Mehmet Uzunoglu, Ali Kizil, Omer Caglar Onar
- "Matlab: A Practical Introduction to Programming and Problem Solving," 2nd edition from Stormy Attaway
- "Matlab 7.04" from Ugur Arifoglu
- Internet
- Help documents of Matlab

Outline

Week	Subjects
#1	Introduction (this presentation)
#2	Algorithm, Flow Chart
#3	Starting with MATLAB
#4	Creating Arrays
#5	Math Operations
#6	Script Files
#7	Two-Dimensional Plots
#8	User-Defined Functions
#9	Programming in MATLAB
#10	Mid-Term Exam
#11	Polynomials, Curve Fitting, Interpolation
#12	Applications in Numerical Analysis, Symbolic Math
#13	Graphical User Interface
#14	Simulink

Basic Concepts

- **Hardware**

We can define as “Everything related to computer technology you are able to see and to touch physically.” Screen, Mouse, Keyboard, Hard disk, USB Memory, Printer, Internet, etc.

- **Software**

Simply, “you cannot touch this ☺.” Programs, Data, Documents, Video, Audio, Internet, etc.

Computer Program

A computer program (or just a program) is a sequence of instructions written to perform a specified task with a computer. A computer requires programs to function, typically executing the program's instructions in a central processor. The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled), enables a programmer to study and develop its algorithms.

Computer Programming

- **Computer programming** (often shortened to **programming** or **coding**) is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs.
- The source code is written in one or more programming languages. The purpose of programming is to create a set of instructions that computers use to perform specific operations or to exhibit desired behaviors.
- The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.
- Within software engineering, programming (the *implementation*) is regarded as one phase in a software development process.

Programming languages

Different programming languages support different styles of programming (called programming paradigms). The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Ideally, the programming language best suited for the task at hand will be selected. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. It is usually easier to code in "high-level" languages than in "low-level" ones.

Functional Categorization

- System software
  - Operating system
  - Utility programs
- Application software
  - Other programs than system software

Single- or Multi-Tasking

- Two or more computer programs may run simultaneously on one computer, a process known as multitasking.
- Grid-computing

Compiler & Interpreter

- A *computer program* in the form of a **human-readable**, computer programming language is called **source code**. Source code may be converted into an **executable image** by a **compiler** or executed immediately with the aid of an **interpreter**.

Quality requirements

Whatever the approach to software development may be, the final program must satisfy some fundamental properties. The following properties are among the most relevant:

- Reliability
- Robustness
- Usability
- Portability
- Maintainability
- Efficiency / Performance

Readability

Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Some of these factors include:

- Different indentation styles (whitespace)
- Comments
- Decomposition
- Naming conventions for objects (such as variables, classes, procedures, etc.)

Algorithmic complexity

The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problem. For this purpose, algorithms are classified into orders using so-called Big O notation, O(n), which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.

Methodologies

- Software Development Processes:
  - Use Case
  - Agile Software Development
- Modelling:
  - Object-Oriented Analysis and Design
  - Model-Driven Architecture
  - Entity-Relationship Modeling
- Implementation Techniques:
  - Imperative Languages
  - Functional Languages
  - Logic Languages

Debugging

- Debugging is a very important task in the software development process, because an incorrect program can have significant consequences for its users. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Use of a static code analysis tool can help detect some possible problems.
- Debugging is often done with IDEs like Eclipse, Kdevelop, NetBeans, Code::Blocks, and Visual Studio. Standalone debuggers like gdb are also used, and these often provide less of a visual environment, usually using a command line.

How To Think Like A Computer Scientist

The details look different in different languages, but a few basic instructions appear in just about every language:

- **input**: Get data from the keyboard, a file, or some other device.
- **output**: Display data on the screen or send data to a file or other device.
- **arithmetic**: Perform basic arithmetical operations like addition and multiplication.
- **conditional execution**: Check for certain conditions and execute the appropriate sequence of statements.
- **repetition**: Perform some action repeatedly, usually with some variation.

Many computer languages provide a mechanism to call functions provided by libraries. Provided the functions in a library follow the appropriate run time conventions (e.g., method of passing arguments), then these functions may be written in any other language.

Software Engineering at Glance

What is programming?

- Programming is breaking a task down into small steps.
- Programmers think in an unnatural way.

Who can be a programmer?

- To be a good programmer, you would be:
  - Logical
  - Patient
  - Perceptive
  - At least moderately intelligent
  - Enjoys intellectual challenges
  - Female

Programming steps

- Write a program.
- Compile the program.
- Run the program.
- Debug the program.
- Repeat the whole process until the program is finished.

Programming is programming

- Scripting languages
  - JavaScript
  - AppleScript
  - ActionScript
  - Tcl
  - HTML
- Where to start
  - Not so important, just start suffering
  - Take your time, and don’t expect to go fast.
  - Do the excersises.

Homework #1: What are these?

Not later than the next week:

Explain the followings by up to 3 sentences for each terms of «C, C++, Java, FORTRAN, BASIC, COBOL, ADA, PROLOG, LISP, API, GUI (Graphical User Interface), ASCII, Binary, Bit, Byte, KB, MB, GB, TB, Cross-platform, Linux, Windows, MacOS, SDK (software development kit), Virtual Machine, Virtual Memory, Open Source, GPL (general public licence), Object Oriented Programming»