Introduction and Overview

Joseph Spring

7COM1023 – Programming Paradigms
Areas for Discussion

• Introduction
  – 7COM1023 Module Guide
• Aims, Knowledge and Understanding
• Indicative Content
• Assessment
• Reading List
Areas for Discussion

• Background
• Paradigms
  – Imperative (2Weeks)
  – OOP (2Weeks)
  – Functional (2Weeks)
  – Logic (2Weeks)
  – Quantum Based (2Weeks + discussions)
Introduction

• This module explores:
  – the extent to which different programming paradigms can be applied to the implementation of elegant solutions to a given programming problem.

• To this end:
  – this module will evaluate different programming paradigms, such as imperative, functional, concurrent, object-oriented programming and quantum paradigms
Aims

The aims of this module are to enable students to:

• Gain experience in a range of different programming paradigms

• Critically evaluate how these can be effectively applied to the solution of computing problems.
Paradigm

• A pattern of thought that guides a collection of related activities

• A programming paradigm is a pattern of *problem solving thought* that underlies a particular genre of programs and languages

• Four distinct paradigms emerged in last three decades
  • Imperative Paradigm
  • Object Oriented Paradigm
  • Functional Paradigm
  • Logic Paradigm
Programming Paradigms

• Some paradigms support more than one paradigm:
  – C++ is designed to support both the imperative and OOP paradigm, said to be a hybrid language
  – The experimental programming language Leda (see Budd 1995) is designed to support the imperative, OOP, Functional and Logic paradigm
  – These (with the exception of C++) are said to be unsuccessful attempts to design general purpose languages supporting different paradigms
  – Eg: Algol 68, PL/I, Ada
Imperative Paradigm

• This is the oldest paradigm grounded in the classical von Neumann – Eckert model of computation

• In this model program and variables are stored together

• Program contains a series of commands that perform:
  – Calculations
  – Assign values to variables
  – Retrieve inputs
  – Produce outputs
  – Redirect control elsewhere in the program
Example 1

/* C program to display greeting on the screen. */
#include <stdio.h>
void main(void)
{
    printf("Welcome \n");
}

And in Ruby:
puts ‘Welcome’
Example 2

.....

int sum (int n)
{
    int i, s = 0;
    for (i=0; i < n; i++)
    {
        s += i;
    }
    return s;
}
Example 2

......

```ruby
def sum(n)
    sum = i = 0
    while i < n do
        sum += i
        i += 1
    end
    return sum
end
```

Adding the first n Natural numbers in Ruby
Imperative Paradigm

• Procedural Abstraction
  – An essential building block for the imperative paradigm

• As are:
  – Assignments
  – Loops
  – Conditional statements
  – Exception handling

• The predominant Imperative Programming Languages include:
  – Cobol
  – Fortran
  – C
  – Ada
  – Perl
  – Basic, Pascal, …
Object Oriented Paradigm

• Provides a model in which the program is
  – A model of the world in terms of objects, with state and which can react to methods
    • That interact with each other by passing messages that transform their state
  – In this sense message passing allows data objects to be active rather than passive
  – This characteristic helps to distinguish OOP from the imperative paradigm
  – Fundamental building blocks include:
    • Object classification
    • Inheritance
    • Message passing
Object Oriented Paradigm

• The predominant OOP languages include:
  – Simula (introduced OOP in mid 60’s)
  – Smalltalk
  – C++
  – Java
  – Ruby
  – C#
Functional Paradigm

• Functional programming
  – models a computational problem as a collection of mathematical functions
  – Each with an input (domain) and output (range) spaces
  – This sets functional programming apart from languages with assignment statement
    • E.g. the assignment statement \( x = x + 1 \) makes no sense in either mathematics or functional programming
  – Functions combine with each other via:
    • Functional composition
    • Conditionals
    • Recursion
Functional Paradigm

• The predominant Functional languages include:
  – Lisp
  – Scheme
  – Haskell
  – Erlang
  – ML
Logic Paradigm

• Logic (declarative) programming
  – models a computational problem by declaring what outcome the program should achieve
  – rather than how it should be achieved
  – These languages are often referred to as *rule based languages*
  – Program declarations look like a set of rules, or constraints on the problem rather than a sequence of commands to be carried out
  – Logic programming
    • A natural mechanism for expressing non determinism
    • Appropriate for problems whose specifications are incomplete

• A major logic programming language is Prolog
Example

• The first large Prolog Program was for Human-machine interactions such as the following:

Every psychiatrist is a person.
Every person the psychiatrist analyses is sick.
Jacques is a psychiatrist in Marseille.
Is Jacques a person?
Where is Jacques?
Is Jacques sick?
Quantum Paradigms

• Quantum information processing is a new paradigm offering significant improvements over classical paradigms for certain types of problem.

• Quantum protocols of interest include:
  – Superdense Coding Protocol
  – Deutsch–Josza Protocol
  – Teleportation Algorithm
  – Grover’s Algorithm
  – Shor’s Algorithm

• Simulations in classical paradigms have been successful.
Problems

• Various type of problem exist, some of which are solvable in polynomial time and others that are less likely to be solved

• As motivation for the Quantum Paradigm we begin with a particular problem that is solvable using quantum paradigms but not solvable (in general) using classical paradigms.
Areas for Discussion

• Introduction
  – 7COM1023 Module Guide
• Aims, Knowledge and Understanding
• Indicative Content
• Assessment
• Reading List
Areas for Discussion

• Background

• Paradigms
  – Imperative (2Weeks)
  – OOP (2Weeks)
  – Functional (2Weeks)
  – Logic (2Weeks)
  – Quantum Based (2Weeks + discussions)