MONASH University
Information Technology

FIT1040
Programming fundamentals

Unit Guide

Semester 2, 2015

Last updated: 12 Aug 2015

Table of Contents

FIT1040 Programming fundamentals - Semester 2, 2015

This unit will provide students with an overview of the fundamentals required to create programs. Students will learn to develop descriptions of algorithms and program logic using pseudocode which will be implemented as working software programs using a visual procedural programming language. The unit will explore a variety of application domains including: computer games, business and science applications, computer generated arts, computer-based simulations and the control of simple robots. The topics covered will include the fundamental concepts: data types and structures, basic types of input and output, program control structures, and modular design along with the basics of event-driven programming and objects. These topics will be covered while placing an emphasis on the need to design program code that is easy to maintain, readable, tested, and well documented.

At the end of the unit students will be expected to be able to apply the knowledge and skills learned in further units that cover software development using industry strength programming languages.

Mode of Delivery

- Clayton (Day)
- Clayton (Online)

Workload Requirements

Minimum total expected workload equals 12 hours per week comprising:

(a.) Contact hours for on-campus students:

- 2 hours of lectures
- One 2-hour laboratory
- One 1-hour tutorial

(b.) Study schedule for off-campus students:

- Off-campus students generally do not attend lecture and tutorial sessions, however should plan to spend equivalent time working through the relevant resources and participating in discussion groups each week.

(c.) Additional requirements (all students):

- A minimum of 7 hours independent study per week for completing lab and project work, private study and revision.

See also Unit timetable information

Unit Relationships

Prohibitions

FIT1002

Chief Examiner

Mr Stephen Huxford

Campus Lecturer

Clayton

Cheryl Howard

Tutors

Clayton

To Be Annouced

Your feedback to Us

Monash is committed to excellence in education and regularly seeks feedback from students, employers and staff. One of the key formal ways students have to provide feedback is through the Student Evaluation of Teaching and Units (SETU) survey. The University's student evaluation policy requires that every unit is evaluated each year. Students are strongly encouraged to complete the surveys. The feedback is anonymous and provides the Faculty with evidence of aspects that students are satisfied and areas for improvement.

For more information on Monash's educational strategy, see:

www.monash.edu.au/about/monash-directions/ and on student evaluations, see:
www.policy.monash.edu/policy-bank/academic/education/quality/student-evaluation-policy.html

Previous Student Evaluations of this Unit

Changes made in S2/2014:

- Most of the tutes and labs were overhauled from a "walkthrough" approach to a problem-solving approach.
- An introduction to a real-world ("industrial") language - Java is introduced gradually, with examples gradually introduced alongside Scribble code...
- ... this takes the form of optional Advanced Lab questions throughout semester, before their formal assessment in Lab 11.

- The last two lectures were overhauled, to handle transition issues with later subjects. Overview of lecture content introduced:
-
    1. how pseudo-code translates to other coding languages
    2. Integrated Development Environments (IDEs) and the syntax and debugging support they provide.
    3. Understanding how testing and good programming practice - can be applied in transitioning to real-world ('industrial') programming (*with credits to Robert Merkel*)

4. Common issues faced in transitioning to a real-world language (e.g. syntax, debugging fears, etc)
5. Get started with industrial programming using Java through a beginner-friendly IDE - BlueJ.
6. Write simple programs using Java (using a 'template' based approach).
7. Faithful translation from Scribble to Java (and vice-versa).
8. Programming languages and paradigms
9. Intro to class-based OO programming in Java, and how it differs from Scribble "cloning"
10. Tools and techniques used in programming - design, source control, debugging, testing, profiling..

- The last assessable Lab and Tute was on translation between Java and Scribble (and vice versa)
- Expanded introductions to Scribble's primitive cloning (prototype-based approach)
- Included FIT1002 historical Java notes as extra reading, as it has been requested by students who want to try Java programming over the summer holidays. The intro to Java and real-world coding in Lectures 11 and 12 should be a good launchpad for this.
- The **MADAM (Marking Admin and Distribution At Monash) grading system** was used for marking A1 and A2 (thanks to Robyn McNamara; and implementation thanks to Phil Abramson). The rich feedback generated by MADAM was informative to help the students in improving their work for A2.

Comments from 2013-2014:

Student feedback about this unit - which is new - has been very positive. There will be minor adjustments to the presentation of the content. One irritation that students have noted is that the tutorial exercises took longer than the hour allocated. That will be adjusted in this offering.

Staff watched the progress of students who completed this unit closely. The unit provided to be an excellent introduction to programming for students undertaking further study in units with an "industrial" language in semester 2. Staff in subsequent units noted that students in those units were a little thrown when they first had to code by typing their code in an exact syntax.

The unit content will be adjusted to help students transition a more smoothly to a real-world language.

If you wish to view how previous students rated this unit, please go to https://emuapps.monash.edu.au/unitevaluations/index.jsp

Academic Overview

Learning Outcomes

At the completion of this unit students should be able to:

1. recognise the relationship between a problem description and program design;
2. implement problem solving strategies;
3. construct and test simple computer programs;
4. analyse and debug existing programs;
5. recognise the importance of programming and documentation;
6. apply good programming practices in accordance with industry standards and professional ethics.

Unit Schedule

Week	Activities	Assessment
0	Students should read the unit guide and become familiar with the assessment requirements of the unit	No formal assessment or activities are undertaken in week 0
1	Introduction to programming with Scribble	
2	Finding errors in programs: testing and debugging	Laboratory work and short tutorial quizzes are assessed weekly between Weeks 2 to 11 (inclusive)
3	Using variables in programs	
4	Making decisions in programs	
5	Using loops	
6	Using loops (continued)	Assignment 1 due Friday 11:59pm
7	Using lists in loops	
8	Using abstraction to represent game play	
9	Objects and Functions	
10	Cloning Multiple Objects	
11	Software development and programming environments I	Assignment 2 due Friday 11:59pm. Laboratory Work and Short Tutorial Tests end
12	Software development and programming environments II	Assignment 2 interviews held
	SWOT VAC	No formal assessment is undertaken in SWOT VAC
	Examination period	LINK to Assessment Policy: http://policy.monash.edu.au/policy-bank/academic/education/assessment/assessment-in-coursework-policy.html

*Unit Schedule details will be maintained and communicated to you via your learning system.

Teaching Approach

Lecture and tutorials or problem classes

This teaching and learning approach helps students to initially encounter information at lectures, discuss and explore the information during tutorials, and practice in a hands-on lab environment.

Assessment Summary

Examination (3 hours): 60%; In-semester assessment: 40%

Assessment Task	Value	Due Date
Assignment 1: Programming Basics	10%	Friday 11:59pm Week 6
Assignment 2: Advanced Programming Application	10%	Friday 11:59pm Week 11. Interviews held in Week 12

Laboratory work and short tutorial quizzes	20%	The end of the tutorial or laboratory session in Weeks 2 to 11 in which the work is performed.
Examination 1	60%	To be advised

Assessment Requirements

Assessment Policy

Faculty Policy - Unit Assessment Hurdles
(http://intranet.monash.edu.au/infotech/resources/staff/edgov/policies/assessment-examinations/assessment-hurd

Academic Integrity - Please see resources and tutorials at
http://www.monash.edu/library/skills/resources/tutorials/academic-integrity/

Assessment Tasks

Participation

- **Assessment task 1**

Title:
Assignment 1: Programming Basics
Description:
This assignment will require the development of a software application with Scribble that reacts to user interface events, taking input from a user and then performing a complex calculation. The task will require creating a programming solution to a calculation-based problem, creating software that reacts to an event, makes decisions with a IF-THEN-ELSE logic, stores user input in variables, and makes calculations using variables. Students are expected to abide by good programming practice (including testing and documentation).
Weighting:
10%
Criteria for assessment:

1. The application must run correctly. Evidence of testing is required.
2. The application must meet the problem specification.
3. The application logic should be documented with pseudocode/flowchart.
4. The application sprites and scripts should be constructed in a way that makes them easy to understand and maintain.
5. Students should be able to answer questions about their own work.

Detailed assessment criteria will be provided on the unit web site along with full details of the assignment task.
Due date:
Friday 11:59pm Week 6

- **Assessment task 2**

Title:
Assignment 2: Advanced Programming Application
Description:
This assignment will require the development of a software application or game with Scribble. The task will require creating an abstraction of its internal state, and changing that state as the user interacts with the application. This will require the creation of appropriate data structures to store the internal state, the user's interaction path (e.g. moves for a game, or commands issued to an app), and scripts that represent the program logic (e.g. game rules or app constraints). The software will make decisions with

IF-THEN-ELSE logic, process data using loop-based logic, and display appropriate UI/status updates (e.g. for a game - state of play, wins, losses; or for an app - output, errors, prompts).

Students will be asked to answer questions about their own work during an interview scheduled outside the lab class in Week 12. Students are expected to abide by good programming practice (including testing and documentation).

Weighting:

10%

Criteria for assessment:

1. The application must run correctly. Evidence of testing is required.
2. The application must meet the problem specification.
3. The application logic should be documented with pseudocode/flowchart.
4. The application sprites and scripts should be constructed in a way that makes them easy to understand and maintain.
5. Students should be able to answer questions about their own work during an interview scheduled outside the lab class.

Detailed assessment criteria will be provided on the unit web site along with full details of the assignment task.

Due date:

Friday 11:59pm Week 11. Interviews held in Week 12

- **Assessment task 3**

Title:

Laboratory work and short tutorial quizzes

Description:

In Weeks 2 to 11 students will be expected to write and execute code to perform tasks specified at the start of their Laboratory session. The specified coding tasks will come from a Laboratory task specification sheet released prior to each Laboratory session allowing for preparation.

Students will be expected to complete a very short quiz at the end of each Tutorial session. They will assess student knowledge of the lecture material for the week the Tutorial is based on.

Weighting:

20%

Criteria for assessment:

Laboratory work will be assessed during the Laboratory session. Full marks will require both working code and good coding style with the latter carrying more weight.

Tutors will mark tutorial short quizzes after the tutorial. The questions will examine both conceptual and practical working knowledge covered in the lecture slides relevant to the tutorial.

Due date:

The end of the tutorial or laboratory session in Weeks 2 to 11 in which the work is performed.

Examinations

- **Examination 1**

  Weighting:
  60%
  Length:
  3 hours
  Type (open/closed book):
  Closed book
  Electronic devices allowed in the exam:
  None

Learning resources

Reading list

Armoni, M. and M. Ben-Ari (2013) *Computer Science Concepts in Scratch.* Weizmann Institute of Science. [on-line] http://stwww.weizmann.ac.il/g-cs/scratch/scratch_en.html.

Gaddis, T. (2012) *Starting out with programming logic and design*. 3rd Edition. Addison Wesley.

Lane, A., B. Meyer and J. Mullins (2012) *Generative Art with Scribble: A Project Based Introduction to Programming.* Monash BlockBooks Series. [on-line] via Apple iTunes Store and http://monash-blockbooks.appspot.com

Lane, A., B. Meyer and J. Mullins (2012) *Simulation with Cellular: A Project Based Introduction to Programming.* Monash BlockBooks Series. [on-line] via Apple iTunes Store and http://monash-blockbooks.appspot.com

Lane, A., B. Meyer and J. Mullins (2012) *Robotics with Enchanting and LEGO® NXT: A Project Based Introduction to Programming.* Monash BlockBooks Series. [on-line] via Apple iTunes Store and http://monash-blockbooks.appspot.com

Sprankle, M & J. Hubbard (2012) *Problem solving and programming concepts*. 9th Edition. Prentice Hall.

Venit, S. & E. Drake (2011) *Prelude to programming: Concepts and design*. 5th Edition. Addison Wesley.

Monash Library Unit Reading List (if applicable to the unit)
http://readinglists.lib.monash.edu/index.html

Feedback to you

Types of feedback you can expect to receive in this unit are:

- Informal feedback on progress in labs/tutes
- Graded assignments with comments
- Interviews
- Test results and feedback
- Quiz results
- Solutions to tutes, labs and assignments

Extensions and penalties

Submission must be made by the due date otherwise penalties will be enforced.

You must negotiate any extensions formally with your campus unit leader via the in-semester special consideration process: http://www.monash.edu.au/exams/special-consideration.html

Returning assignments

Students can expect assignments to be returned within two weeks of the submission date or after receipt, whichever is later.

Assignment submission

It is a University requirement
(http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-managing-pla
for students to submit an assignment coversheet for each assessment item. Faculty Assignment coversheets can be found at http://www.infotech.monash.edu.au/resources/student/forms/. Please check with your Lecturer on the submission method for your assignment coversheet (e.g. attach a file to the online assignment submission, hand-in a hard copy, or use an electronic submission). Please note that it is your responsibility to retain copies of your assessments.

Online submission

If Electronic Submission has been approved for your unit, please submit your work via the learning system for this unit, which you can access via links in the my.monash portal.

Recommended Resources

BYOB (Build Your Own Blocks), “Scribble” Edition. This free software development tool is available for MacOS, Windows. It is available for download at: http://monofonik.github.com/scribble/

Other Information

Policies

Monash has educational policies, procedures and guidelines, which are designed to ensure that staff and students are aware of the University's academic standards, and to provide advice on how they might uphold them. You can find Monash's Education Policies at:
www.policy.monash.edu.au/policy-bank/academic/education/index.html

Faculty resources and policies

Important student resources including Faculty policies are located at
http://intranet.monash.edu.au/infotech/resources/students/

Graduate Attributes Policy

http://www.policy.monash.edu/policy-bank/academic/education/management/monash-graduate-attributes-policy.h

Student Charter

www.opq.monash.edu.au/ep/student-charter/monash-university-student-charter.html

Student services

The University provides many different kinds of support services for you. Contact your tutor if you need advice and see the range of services available at http://www.monash.edu.au/students. For Malaysia see http://www.monash.edu.my/Student-services, and for South Africa see http://www.monash.ac.za/current/.

Monash University Library

The Monash University Library provides a range of services, resources and programs that enable you to save time and be more effective in your learning and research. Go to www.lib.monash.edu.au or the library tab in my.monash portal for more information. At Malaysia, visit the Library and Learning Commons at http://www.lib.monash.edu.my/. At South Africa visit http://www.lib.monash.ac.za/.

Disability Liaison Unit

Students who have a disability or medical condition are welcome to contact the Disability Liaison Unit to discuss academic support services. Disability Liaison Officers (DLOs) visit all Victorian campuses on a regular basis.

- Website: http://www.monash.edu/equity-diversity/disability/index.html
- Telephone: 03 9905 5704 to book an appointment with a DLO; or contact the Student Advisor, Student Commuity Services at 03 55146018 at Malaysia
- Email: dlu@monash.edu
- Drop In: Equity and Diversity Centre, Level 1, Building 55, Clayton Campus, or Student Community Services Department, Level 2, Building 2, Monash University, Malaysia Campus

Other

Recognition of Prior Learning

Prior to the start of semester, students who have advanced programming skills are invited to attempt an on-line based assessment of their existing skills, knowledge and ability. Students who obtain a pass grade may choose to enroll in a more advanced programming unit in place of FIT1040 and receive an exemption (but not credit) for FIT1040.

The on-line test can be found at http://dsslab.infotech.monash.edu.au:8080/moodle/

Engineers Australia Stage 1 competencies

This unit is a core unit in the Bachelor of Software Engineering accredited by Engineers Australia. Engineers Australia Accreditation Policy of Professional Engineering Programs requires that programs demonstrate how engineering graduates are prepared for entry to the profession and achieve Stage 1 competencies. The following information describes how this unit contributes to the development of these competencies for the Bachelor of Software Engineering. (Note: not all competencies may be emphasised in this unit).

Stage 1 competency	**How the compency is developed in this unit**
1. Knowledge and Skills base	
1.1. **Comprehension, theory based understanding** of the underpinning natural and physical sciences and the engineering fundamentals applicable to the engineering discipline.	The unit covers computing foundations which underpins the programming knowledge required in software engineering. The required knowledge is covered by theoretical lecture materials, and recommended reading, tutorials and laboratory tasks.
1.2. **Conceptual understanding** of the mathematics, numerical analysis, statistics, and computer and information sciences, which underpin the engineering discipline.	Lecture materials and exercises provide conceptual understanding of computer science, in particular programming solving techniques, which underpins software engineering.
1.3. **In-depth understanding** of specialist bodies of knowledge within the engineering discipline.	Not covered in this unit.
1.4. **Discernment** of knowledge development and research directions within th engineering discipline.	Not covered in this unit.
1.5. **Knowledge** of engineering design practice and contextual factors impacting the engineering discipline.	Not covered in this unit.
1.6. **Understanding** of the scope, principles, norms, accountabilities and bounds of sustainable engineering practice in the specific discipline.	Not covered in this unit.
2. Engineering application ability	
2.1. **Application** of established engineering methods to complex engineering problem solving.	The unit addresses programming issues and problems and how to solve these problems. It is covered by programming exercises and assessments.
2.2 **Fluent application** of engineering techniques, tools and resources.	The fundamental aspects of programming are covered and applied in this unit.
	Not covered in this unit.

2.3. **Application** of systematic engineering synthesis and design processes.	
2.4. **Application** of systematic approaches to the conduct and management of engineering projects.	Not covered in this unit.
3. Professional and personal attributes	
3.1. **Ethical** conduct and professional accountability.	Some aspects are covered in this unit, in relation to good programming practice and ethics.
3.2. **Effective** oral and written communication in professional and lay domains.	This is covered in the lab exercises and assignments.
3.3. **Creative**, innovative and proactive demeanour.	Using problem solving techniques to develop programs is inherently a creative endeavour.
3.4. **Professional** use and management of information.	Not covered in this unit.
3.5. **Orderly** management of self, and professional conduct.	This is covered in the lab exercises and assignments, while students manage and conduct themselves during assessment.
3.6. **Effective** team membership and team leadership.	Not covered in this unit.

Relationship between Unit Learning Outcomes and BSE Course Outcomes

No.	CO 1	CO 2	CO 3	CO 4	CO 5	CO 6	CO 7	C0 8	CO 9	CO 10	CO 11	CO 12	CO 13
1	X				X	X		X			X		
2	X				X	X		X			X		
3	X				X	X		X			X		
4	X				X	X		X			X		
5	X				X	X		X			X		
6	X				X	X		X			X		

Relationship between Unit Learning Outcomes and Assessments

No.	Assignments	Tests	Practical Exercises	Exam
1	X	X	X	X
2	X	X	X	X
3	X		X	X
4	X	X	X	X
5	X		X	X
6	X		X	X