SUBJECT OUTLINE

31251 Data Structures and Algorithms

Course area	UTS: Information Technology
Delivery	Autumn 2020; City
Credit points	6cp
Requisite(s)	48024 Applications Programming
Result type	Grade and marks

Recommended studies: basic programming concepts: variables, loops and decisions; basic file manipulation in UNIX: directories and files, editing files, re-direction; basic understanding of the standard Von Neumann computer model: the fetch-execute cycle, single memory with byte addressing, input and output with disks, keyboard and screen; understanding of character sets and internal data representations, including ASCII, signed integers, floating point

Subject coordinator

Dr Xianzhi Wang
Office: CB11.07.110
Telephone: (02) 9514 5386
Email: xianzhi.wang@uts.edu.au

Consultation Hours: Thursday, 2-4pm

Consultations will be held in the FEIT Learning Precinct (CB11.05.300). Appointments outside the given consultation hours may be arranged upon request. To do so, please contact the subject coordinator by email.

Questions regarding assessment or content within the subject are welcome in lectures or tutorials (i.e., Computer Labs) or alternatively post them to the discussion board in UTS Online. This helps ensure that all students get the benefit of the answers given.

The Subject Coordinator may be contacted by email if you have matters of a personal nature to discuss, e.g., illness, study problems, and for issues to do with extensions or other matters of importance.

All emails sent to subject coordinators, tutors or lecturers must have a clear subject line with the subject code and the purpose of the email, [e.g., 31251 - Missed a class], and must be sent from your UTS email addresses.

Teaching staff

Lecturer: Dr Xianzhi Wang (see subject coordinator for details).

Tutors: to be advised (see Staff Contacts via UTSOnline)

Subject description

This subject teaches students how to design, develop and evaluate data structures and algorithms to meet predefined quality characteristics of functionality (suitability) and usability (understandability, learnability, operability, compliance). Software solutions are implemented using C++. Concepts, theories and technologies underlying the methods and techniques are introduced and explained as required.

Subject learning objectives (SLOs)

Upon successful completion of this subject students should be able to:

1. Explain the basic data structures and algorithms for manipulating them.
2. Implement these data structures and algorithms in the C++ language.
3. Integrate these data structures and algorithms in larger programs.
4. Code and test well-structured programs of moderate size using the C++ language.

5. Apply principles of good program design to the C++ language.
6. Demonstrate advanced programming skills.

Course intended learning outcomes (CILOs)

This subject also contributes specifically to the development of the following Course Intended Learning Outcomes (CILOs):

- Design Oriented: FEIT graduates apply problem solving, design and decision-making methodologies to develop components, systems and processes to meet specified requirements. (C.1)

Teaching and learning strategies

Lecture topics and supporting material will be available online, along with readings to be completed before the tutorial. The lectures provide the fundamental background material and guidance to support the students' learning as they implement, evaluate, compare and critique the data structures and algorithms presented throughout the course.

The tutorials provide a point for feedback and to collaboratively experiment with implementations. In tutorial discussion about the material will provide the students a forum to reflect on their learning and consider the input of both their peers and their tutor.

The knowledge that the students build each week around specific data structures and algorithms leads into and supports the completion of their assessment tasks throughout the session.

Feedback for assessments will be given within two weeks of the in class demonstration (for programming assignments) and from completion (for quizzes).

Content (topics)

1. C++ constructs including templates and the STL
2. Program design
3. Design, implementation, and evaluation of data structures
4. Design, implementation, and evaluation of algorithms
5. Recursion
6. Computability, NP-Completeness, Optimization

Program

Week/Session	Dates	Description
1	9 March	**Tasks** • Watch the introductory videos • Read about the basics of C++ (in your chosen reference material) • Have a go at making a basic "Hello World" program. **Topics** Preparation: • Understand basic concepts, e.g., data structure, algorithm. • Know the C++ compiling and development process • Useful tools for learning this subject • Overview of this subject Preliminary introduction to C++: • Data Types, • Loops, • Functions, • Basic compilation • Tools for C++ development • Similarities and differences with Java **Notes:**

It is suggested that before starting the formal tutorials, you take some time to familiarise yourself (and install) your options for C++ development on the computer that you normally work on.

If you're a Linux or Mac user, it's straight forward, g++ comes packaged with the OS.

For Windows users, the situation is slightly trickier, for some suggestions, take a look at this StackOverflow post:
https://stackoverflow.com/questions/1154517/c-compiler-for-windows

I have had positive experiences with the MinGW toolkit, but there's plenty of others to try.

You may also consider investigating IDEs, I have used Eclipse + CDT, which is okay, but can be a bit touchy. Explore!

---

2 16 March

Tasks

- Program, compile and demonstrate a small C++ program.
- Discuss in the tutorial issues you encountered in this process.

Topics

- Pointers
- Strings in C and C++
- Arrays
- Classes (what is the point of headers??)
- Linked Lists

---

3 23 March

Tasks

- Implement a simple Linked List class.
- Discuss in the tutorial what you found easy or difficult about the implemention.
- As class, share and brainstorm design solutions to problems you encountered.

Topics

- Basic I/O
- A little more about Classes
- Exceptions in C++
- Compiling and linking multiples files.
- Queues and Stacks

Assessment

- Quiz 1 to be completed by 5pm, Friday 27 March 2020.

Notes:

Quiz 1 due.

---

4 30 March

Tasks

- Building from your Linked List (or a new implementation), write implementations for a Queue and a Stack.
- In the tutorial, compare and contrast these data structures. If you were writing these for a library, how would you do it?

Topics

		• Vectors in the STL • Templates • Iterators • Simple Big O notation and comparing algorithms.
5	6 April	**Tasks** • Analyse the code you have implemented so far in terms of Big O analysis. ◦ Consider what you're using as an atomic step. ◦ What level do you analyse the code at? Functions? Classes? Programs? **Topics** • Recursion • Divide and Conquer • Recursion vs. Iteration
6	13 April	**Tasks** • Implement two versions of a function to compute the Fibonacci sequence, one recursive, one iterative. • Compare the implementations practically. How high of a Fibonacci number can you compute? • Attempt a Big O analysis of the two implementations. What is the reason they are different? • Compare in the tutorial your implementations with others, can they be improved? Practically? Asymptotically? • Discuss in the tutorial what you think makes recursion useful, why not just use iteration? **Topics** • Binary trees. • Binary tree traversals. • Uses of binary trees: Binary Search Trees, Heaps **Assessment** • Quiz 2 to be completed by 5pm, Friday 17 April 2020 . • Assignment 1 released. **Notes:** Quiz 2 due.
	20 April	Stu-Vac
7	27 April	**Tasks** • Implement a simple Binary Tree data structure. • Use this to implement a Binary Search Tree. • Analyse your BST methods' asymptotic running times. • Share your solutions in the tutorial and discuss possible improvements. **Topics**

- Hashing and Hashmaps.

---

8 4 May

Tasks

- Implement your own Hashmap. Choose a collision handling approach.
- Analyse the running time of your implementation. Is it in line with theoretical expectations?
- Discuss in class the success of your Hashmap and your choice of collision handling algorithm. Did it work well? Compare with other students and the std::undordered_map implemented in Standard C++.

Topics

- Sorting

Assessment

- Assignment 1 due by 5pm, Friday 8 May 2020.

Notes:

Assignment 1 due

---

9 11 May

Tasks

- Implement Bubble Sort, Merge Sort and Quick Sort.
- Design a test that compares the performance of these algorithms.
- Discuss in the tutorial their relative performance and what improvements might be made. Consider why you might use one over the other (you may want to do some research on which sorting algorithms are used where).

Topics

- String Searching

Assessment

- Assignment 1 demonstrated in your tutorial.
- Assignment 2 released.
- Quiz 3 to be completed by 5pm, Friday 15 May 2020.

Notes:

Quiz 3 due.

---

10 18 May

Tasks

- Implement the Rabin-Karp algorithm (you may use either C++ strings or C strings, but avoid using the built-in functions).
- Implement a naive string search algorithm.
- Compare the two algorithms practically and asymptotically. Is one always better? (Is this because of your implementation, or because of the algorithm itself?)
- Discuss in the tutorial how you implemented the algorithm, and consider how you might've implemented it if you were working in a different language (e.g. Java) - is it easier or harder in C++?

Topics

- Greedy Algorithms
- Dynamic Programming

11	25 May	**Tasks** • Implement a greedy Minimum Spanning Tree Algorithm • Consider why a greedy strategy works for this problem (can you prove it?). Think of a problem where this fails. Discuss in tutorials your thoughts. **Topics** • P, NP, NP-hardness and basic complexity theory. **Assessment** • Assignment 2 due by 5pm, Friday 29 May 2020. **Notes:** Assignment 2 due.
12	1 June	**Catch-up and Recap Week** This time can be used to review, recap and catch up on topics or tasks that you have missed, or want further coverage of. **Assessment** • Assignment 2 demonstrated in your tutorial.
	8 June	Final Stu-Vac

Additional information

U:PASS Program

This subject participates in the UTS Peer Assisted Study Success (U:PASS) program. Students are strongly encouraged to participate, regardless of ability level.

U:PASS is a voluntary "study session" where students study the subject in a group environment with other students. The session will be led by a student who has previously achieved a high mark in the subject and has a good academic record. The leader will typically prepare questions or exercises to work with, or you can bring your own questions. The environment is friendly, relaxed and informal. As the leader is still a student, they understand what it's like to study Data Structures and Algorithms and how to do well in the subject from a student perspective.

Students can sign up for U:PASS sessions via U:PASS website from week 2.

Students are strongly encouraged to consider giving U:PASS a go, even students who are already doing well can benefit from the program.

If you have any questions about U:PASS, please contact upass@uts.edu.au or check out the website.

Assessment

Assignment Demonstration:

As part of marking assessment tasks 1 & 2, you will be required to demonstrate the functionality of your program in your assigned laboratory.

Quizzes:

Weeks 3, 6 and 9 include quizzes that will assist in gauging your progress and understanding as you progress through the subject. The quiz will be a simple online quiz administered via UTSOnline. The quizzes are to be completed by

Friday, 5pm, of the week they are officially released. The answers may be discussed in the tutorial the following week.

Group Work:

All assessment is individual. There is NO group work.

Assessment task 1: Programming Assignment 1

Intent: This assessment task allows you to demonstrate an understanding of some of the data structures and algorithms covered in the course to this point, and their implementation in C++. It supports demonstration of the ability to combine individual data structures and algorithms to form a coherent solution to a computational problem.

Objective(s): This assessment task addresses the following subject learning objectives (SLOs):

2, 3, 4 and 5

This assessment task contributes to the development of the following Course Intended Learning Outcomes (CILOs):

C.1

Type: Exercises

Groupwork: Individual

Weight: 25%

Task: Given a specification, create a well designed and well tested command line program in C++.

Length: 200-300 lines of code.

Due: 5.00pm Friday 8 May 2020
See also Further information.

Further information: The assignment will be demonstrated in your assigned laboratory in Week 9.

The feedback from this assessment task will include both summative and formative feedback. Summative feedback will include your mark against the marking criteria. Formative feedback will, at minimum, be given by the tutors to the tutorial class as a whole.

Assignments are to be submitted electronically. Further details will be given in the assignment specification.

PLEASE NOTE. The Criteria Linkage weightings given in the Subject Outline are a rough approximation. Please see the Assignment Specification for a detailed explanation of how the criteria are combined.

Assessment task 2: Programming Assignment 2

Intent: This assessment task builds on the skills and knowledge you have developed through the subject and combines more complicated data structures and algorithms in a more sophisticated manner, allowing you to demonstrate ability in advanced programming and a higher level understanding of the main topics areas of the subject.

Objective(s):	This assessment task addresses the following subject learning objectives (SLOs): 2, 3, 4, 5 and 6 This assessment task contributes to the development of the following Course Intended Learning Outcomes (CILOs): C.1
Type:	Exercises
Groupwork:	Individual
Weight:	35%
Task:	Given a specification, create a well designed and well tested progran in C++.
Length:	300-400 lines of code.
Due:	5.00pm Friday 29 May 2020 See also Further information.
Further information:	The assignment will be demonstrated in your assigned laboratory in Week 12. The feedback from this assessment task will include both summative and formative feedback. Summative feedback will include your mark against the marking criteria. Formative feedback will, at minimum, be given by the tutors to the tutorial class as a whole. Assignments are to be submitted electronically. Further details will be given in the assignment specification. PLEASE NOTE. The criteria weightings given in the Subject Outline are a rough approximation. Please see the Assignment Specification for a detailed explanation of how the criteria are combined.

Assessment task 3: Final Examination

Intent:	The final exam assesses your grasp of the discipline knowledge component of the subject. It also allows you to demonstrate your ability to communicate algorithm and data structure design ideas in short form.
Objective(s):	This assessment task addresses the following subject learning objectives (SLOs): 1 and 5 This assessment task contributes to the development of the following Course Intended Learning Outcomes (CILOs): C.1
Type:	Examination
Groupwork:	Individual
Weight:	30%

Task:	Open book exam. The questions will include a mixture of multiple choice and short answer questions. The questions will cover the whole range of material presented in the lectures and the laboratories. The questions will delve in to the students understanding of the material rather than how much they remember.
Length:	2 Hour exam
Due:	UTS Exam period
Further information:	The exam forms the final, summative assessment of subject knowledge and skills. **Missing the Exam:** If student/s miss the exam through documented illness or misadventure, they should consult with the Subject Coordinator.

Assessment task 4: Quizzes

Intent:	The quizzes aim to provide regular feedback and basic assessment of your progress through the course by addressing key knowledge points. In conjunction with the weekly programming tasks and communication with your tutor and peers, these help form the primary routine feedback channel for the subject.
Objective(s):	This assessment task addresses the following subject learning objectives (SLOs): 1 and 5 This assessment task contributes to the development of the following Course Intended Learning Outcomes (CILOs): C.1
Type:	Quiz/test
Groupwork:	Individual
Weight:	10%
Task:	The quizzes will be short quizzes administered through UTSOnline.
Due:	Quizzes are to be completed throughout the session. See program for individual quiz completion deadlines.

Assessment feedback

Feedback will be in two parts.

1. A submission system will be used that gives automated feedback about many features of the student’s code. This is done with static analysis.
2. The marker will give individual feedback to the students.

Feedback will be given 2 weeks after submission.

Minimum requirements

In order to pass the subject, a student must achieve an overall mark of 50% or more.

Prize offered

WiseTech Global Computer Programming Prize

Recommended texts

Elliot B. Koffman & Paul A.T. Wolfgang
Objects, Abstraction, Data Structures and Design Using C++
John Wiley & Sons, Inc
ISBN 0-471-46755-3

References

Possibly Useful Books

S. Lippman, J. Lagoie & B. Moo
C++ Primer - 4th Edition.

N. Josuttis
The C++ Standard Library: A Tutorial and Reference
A very useful book that goes into the details of the STL.

D. S. Malik
C++ Programming: Program Design Including Data Structures
A more extensive equivalent of the recommended text.

T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein
Introduction to Algorithms
A classic reference text that goes well beyond this subject, but is useful to anyone who plans on being a programmer.

S. Skiena
The Algorithm Design Manual
A really useful algorithms book that includes numerous real-world considerations and anecdotes.

J. Erickson
Algorithms
An excellent, thorough text, licensed under a CCA 4.0 license, so you can get it and print it for free:
http://jeffe.cs.illinois.edu/teaching/algorithms/

Online References

References for C++ itself:
http://www.cppreference.com/
http://www.cplusplus.com/

The Unit Testing Suite Used in the Course:
http://cxxtest.com/

Other resources

Computer account

Students will need to have a student computer account with the Faculty of Engineering and Information Technology. If students are a faculty-student they will already have one. If students are a non-faculty student they will need to ensure they have one. If students are unsure or need to arrange an account, then they can contact the FEIT Technical Support Help Desk.

UTS Online

This subject makes use of UTS Online, for all course components including announcements, distribution of course materials, assessment submission and discussions boards. Students are strongly encouraged to make use of the discussion boards to ask questions to help with their learning. Students should check UTS Online regularly.

Graduate attribute development

For a full list of the faculty's graduate attributes refer to the FEIT Graduate Attributes webpage.

For the contribution of subjects taken in the Bachelor of Engineering (Honours) or Master of Professional Engineering to the Engineers Australia Stage 1 Competencies, see the faculty's Graduate Attributes and the Engineers Australia Stage 1 Competencies webpage.

Assessment: faculty procedures and advice

Marking criteria

Marking criteria for each assessment task will be available on the Learning Management System: UTS Online.

Extensions

When, due to extenuating circumstances, you are unable to submit or present an assessment task on time, please contact your subject coordinator before the assessment task is due to discuss an extension. Extensions may be granted up to a maximum of 5 days (120 hours). In all cases you should have extensions confirmed in writing.

Special consideration

If you believe your performance in an assessment item or exam has been adversely affected by circumstances beyond your control, such as a serious illness, loss or bereavement, hardship, trauma, or exceptional employment demands, you may be eligible to apply for Special Consideration.

Late penalty

Work submitted late without an approved extension is subject to a late penalty of 10 per cent of the total available marks deducted per calendar day that the assessment is overdue (e.g. if an assignment is out of 40 marks, and is submitted (up to) 24 hours after the deadline without an extension, the student will have four marks deducted from their awarded mark). Work submitted after five calendar days is not accepted and a mark of zero is awarded.

For some assessment tasks a late penalty may not be appropriate – these are clearly indicated in the subject outline. Such assessments receive a mark of zero if not completed by/on the specified date. Examples include:

a. weekly online tests or laboratory work worth a small proportion of the subject mark, or
b. online quizzes where answers are released to students on completion, or
c. professional assessment tasks, where the intention is to create an authentic assessment that has an absolute submission date, or
d. take-home papers that are assessed during a defined time period, or
e. pass/fail assessment tasks.

Querying results

If you wish to query the result of an assessment task or the final result for a subject:

- Assessment task: query the result with the Subject Coordinator within 5 working days of the date of release of the result
- Final subject result: submit an application for review within 5 working days of the official release of the final subject result.

Academic liaison officer

Academic liaison officers (ALOs) are academic staff in each faculty who assist students experiencing difficulties in their studies due to: disability and/or an ongoing health condition; carer responsibilities (e.g. being a primary carer for small children or a family member with a disability); and pregnancy.

ALOs are responsible for approving adjustments to assessment arrangements for students in these categories. Students who require adjustments due to disability and/or an ongoing health condition are requested to discuss their situation with an accessibility consultant at the Accessibility Service before speaking to the relevant ALO.

Statement about assessment procedures and advice

This subject outline must be read in conjunction with the Coursework Assessments policy and procedures.

Statement on copyright

Teaching materials and resources provided to you at UTS are protected by copyright. You are not permitted to re-use these for commercial purposes (including in kind benefit or gain) without permission of the copyright owner. Improper or illegal use of teaching materials may lead to prosecution for copyright infringement.

Statement on plagiarism

Plagiarism and academic integrity

At UTS, plagiarism is defined in Rule 16.2.1(4) as: 'taking and using someone else's ideas or manner of expressing them and passing them off as ... [their] own by failing to give appropriate acknowledgement of the source to seek to gain an advantage by unfair means'.

The definition infers that if a source is appropriately referenced, the student's work will meet the required academic standard. Plagiarism is a literary or an intellectual theft and is unacceptable both academically and professionally. It can take a number of forms including but not limited to:

- copying any section of text, no matter how brief, from a book, journal, article or other written source without duly acknowledging the source
- copying any map, diagram, table or figure without duly acknowledging the source
- paraphrasing or otherwise using the ideas of another author without duly acknowledging the source
- re-using sections of verbatim text without using quote marks to indicate the text was copied from the source (even if a reference is given).

Other breaches of academic integrity that constitute cheating include but are not limited to:

- submitting work that is not a student's own, copying from another student, recycling another student's work, recycling previously submitted work, and working with another student in the same cohort in a manner that exceeds the boundaries of legitimate cooperation
- purchasing an assignment from a website and submitting it as original work
- requesting or paying someone else to write original work, such as an assignment, essay or computer program, and submitting it as original work.

Students who condone plagiarism and other breaches of academic integrity by allowing their work to be copied are also subject to student misconduct Rules.

Where proven, plagiarism and other breaches of misconduct are penalised in accordance with UTS Student Rules Section 16 – Student misconduct and appeals.

Avoiding plagiarism is one of the main reasons why the Faculty of Engineering and IT is insistent on the thorough and appropriate referencing of all written work. Students may seek assistance regarding appropriate referencing through UTS: HELPS.

Work submitted electronically may be subject to similarity detection software. Student work must be submitted in a format able to be assessed by the software (e.g. doc, pdf (text files), rtf, html).

Further information about avoiding plagiarism at UTS is available.

Retention of student work

The University reserves the right to retain the original or one copy of any work executed and/or submitted by a student as part of the course including, but not limited to, drawings, models, designs, plans and specifications, essays, programs, reports and theses, for any of the purposes designated in Student Rule 3.9.2. Such retention is not to affect any copyright or other intellectual property right that may exist in the student's work. Copies of student work may be retained for a period of up to five years for course accreditation purposes. Students are advised to contact their subject coordinator if they do not consent to the University retaining a copy of their work.

Statement on UTS email account

Email from the University to a student will only be sent to the student's UTS email address. Email sent from a student to the University must be sent from the student's UTS email address. University staff will not respond to email from any other email accounts for currently enrolled students.