Higher National Unit specification: general information

Unit title: Software Development: Data Structures

Unit code: H16Y 35

Superclass: CB

Publication date: November 2012

Source: Scottish Qualifications Authority

Version: 02

Unit purpose

This Unit is designed to enable candidates to become familiar with the data structures and collection classes in common use within current software development environments. This knowledge will be supplemented by the coding of collection and/or aggregation associations using appropriate standard generic collection classes.

The Unit is a mandatory Unit for the HND Computing: Software Development and has been designed to enhance candidates' programming and algorithm design skills. These skills should help prepare candidates for employment and/or further study in the field of software development.

On completion of the Unit the candidate should be able to:

1 Describe data representation and storage in computer systems.
2 Describe and use data structures.
3 Describe, develop and use abstract data types.
4 Use Standard Collection classes to implement object oriented designs.

Recommended prior knowledge and skills

Access to this Unit will be at the discretion of the centre, however it is recommended that candidates should have prior experience of appropriate high-level languages and systems development. This may be demonstrated by possession of the core HNC Computing Units along with the software development feeder HN Unit *Software Development: Developing Small Scale Standalone Applications*. Alternatively, candidates may have considerable practical work experience and some appreciation of the role of data structures in program design and program implementation. Ideally the Unit should be delivered alongside the HN Unit *Software Development: Object Oriented Programming*.

General information (cont)

Credit points and level

2 Higher National Unit credits at SCQF level 8: (16 SCQF credit points at SCQF level 8*)

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

Core Skills

Opportunities to develop aspects of Core Skills are highlighted in the Support Notes of this Unit specification.

There is no automatic certification of Core Skills or Core Skill components in this Unit.

Context for delivery

If this Unit is delivered as part of a Group Award, it is recommended that it should be taught and assessed within the subject area of the Group Award to which it contributes.

Higher National Unit specification: statement of standards

Unit title: Software Development: Data Structures

Unit code: H16Y 35

The sections of the Unit stating the Outcomes, Knowledge and/or Skills and Evidence Requirements are mandatory.

Outcome 1

Describe data representation and storage in computer systems.

Knowledge and/or Skills

- Representation of simple and structured data types
- Static and dynamic memory allocation
- Standard File types including compression, images, sound and video
- XML data files

Evidence Requirements

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills by showing that they can describe:

- the representation of at least three simple data types (eg character, integer, floating point, Boolean, etc.) relative to the programming language used.
- the representation of structured data types including string, record, table, one-dimensional array and two-dimensional array.
- static and dynamic memory allocation
- At least two Standard File types for images, sounds, video or compression
- XML data file structure

Outcome 1 should be assessed using a closed-book knowledge based assessment such as an objective test that covers all of the requirements outlined in the Evidence Requirements for the Outcome. Suggested approaches are outlined in the Support Notes, Guidance on the assessment of this Unit.

Higher National Unit specification: statement of standards (cont)

Unit title: Software Development: Data Structures

Outcome 2

Describe and use abstract data structures.

Knowledge and/or Skills

- Array data structures
- Linked List data structures
- Binary Tree data structure
- Hash Table data structure
- Searching and sorting algorithms
- Insertion and deletion algorithms

Evidence Requirements

Most of the evidence for this Outcome will be obtained from the practical assessment described in the Evidence Requirements for the Unit.

In addition candidates will need to provide evidence to demonstrate their Knowledge and Skills by showing that they can:

- step through three data structure algorithms (these must include at least one sort and one search algorithm).

This assessment must be conducted under supervised open-book conditions and the questions presented must change on each assessment occasion. The assessment may be conducted as three separate half hour exercises or as a single assessment with three questions. Possible approaches are described in the assessment guidelines.

Outcome 3

Describe, develop and use abstract data types.

Knowledge and/or Skills

- Concept of an Abstract Data Type (ADT)
- Concept of a range of ADTs including Stack, Queue, Set, List and Map
- Develop Interfaces (method signatures) for a range of ADTs including Stack, Queue, Set, List and Map
- Implement a Stack, Queue and List using a given array data structure
- Implement a Stack, Queue and List using a given linked list data structure

Evidence Requirements

See Evidence Requirements for the Unit.

Higher National Unit specification: statement of standards (cont)

Unit title: Software Development: Data Structures

Outcome 4

Use Standard Collection classes to implement object oriented designs.

Knowledge and/or Skills

- Concept of Generics
- Inserting and deleting from collections
- Iterating through Collections
- Implementing associations using standard collection classes
- Implementing a Map using standard collection classes
- Implementing a Set using standard collection classes
- Testing implemented code.

Evidence Requirements

See Evidence Requirements for the Unit.

Evidence Requirements for the Unit

As an alternative to traditional assessment methods (eg paper-based), Candidates can provide a digital record of evidence to demonstrate Knowledge and/or Skills. Suggested approaches are outlined in the Support Notes, Guidance on the assessment of this Unit.

Outcome 1

This Outcome should be assessed using a closed-book knowledge based assessment such as an objective test that covers all of the requirements outlined in the Evidence Requirements for the Outcome.

Outcome 2 (partial)

The candidate's ability to follow algorithms should be assessed using a supervised open-book assessment as described in the Evidence Requirements for the Outcome.

Outcomes 2 (partial), 3 and 4

These Outcomes should be assessed using the practical based assessment for the Unit described below.

Higher National Unit specification: statement of standards (cont)

Unit title: Software Development: Data Structures

Practical Based Assessment

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills by showing that they can:

- develop interfaces (method signatures) for two ADTs selected from Stack, Queue, Set, List and Map.
- use an Array data structure to implement an ADT selected from Stack, Queue or List.
- use a Linked List data structure to implement an ADT selected from Stack, Queue or List.
- use standard collection classes to implement a Map or a Set.
- use standard collection classes to implement collection and/or aggregation associations.
- implement code that iterates through data stored in standard collection classes.
- test implemented code using given test plans.

This assessment should be conducted under open-book conditions and may well consist of a series of implementation completion exercises conducted during the delivery of the Unit. Candidates should be given the designs and partial classes to allow them to concentrate on implementing the associations and corresponding methods.

Higher National Unit specification: support notes

Unit title: Software Development: Data Structures

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

While the exact time allocated to this Unit is at the discretion of the centre, the notional design length is 80 hours.

Guidance on the content and context for this Unit

This Unit is designed to enable Candidates to become familiar with the data structures and collection classes in common use within current software development environments. This knowledge will be supplemented by the coding of collection and/or aggregation associations using appropriate standard generic collection classes. The Unit is a mandatory Unit for the HND Computing: Software Development and ideally should be delivered in tandem with the HN Unit *Software Development: Object Oriented Programming*. The Unit has evolved from the prior Units *Software Development: Array Data Structures* (DM31 35), *Software Development: Object Oriented Collections* (DM33 35) and *Software Development: Linked Data Structures* (DM32 35).

The Unit may be used as a stand-alone Unit within other frameworks but should be restricted to courses at SCQF level 8 or equivalent within the disciplines of Computer Science or Information Systems.

Outcome 1

The first Outcome provides an opportunity to study how various data types are represented and stored in computer systems. The range of primitives should cover integers, floating point numbers, characters and Boolean as appropriate to the implementation language used. The representation of structured data types should include strings, records, tables, one-dimensional arrays and two-dimensional arrays. Candidates should be familiarised to common standards such as IEEE 754 for floating points as well as ASCII and Unicode standards for characters. The difference between static and dynamic memory allocation should be described. Candidates should be introduced to the concept of file storage including text and binary files. They should be aware of the purpose of common image, audio, video and compression file standards.

Unit title: Software Development: Data Structures

Outcome 2

The second Outcome concentrates on a range of data structures commonly used within software development environments. Candidates should be introduced to both linear and circular array data structures, single and double linked list data structures, binary tree data structures and hash table data structures. The Outcome also looks at some of the common algorithms used with data structures to add, remove and sort items within a data structure as well as how to search for data. The searching algorithms covered should cover linear, binary and hash table searching techniques to illustrate how the efficiency of searches can be dependent on the underlying data structure. The sorting algorithms should cover an appropriate range (eg contrast bubble and selection sorts with the merge and quick sorts) to illustrate how the time efficiency can be improved at the expense of increased space complexity. Candidates should be introduced to the concept of recursive algorithms to enable them to understand their advantages and disadvantages as compared to iterative algorithms.

Outcome 3

The third Outcome introduces the Candidate to the concept of an abstract data type and how an ADT defines the method signatures rather than the internal workings, reinforcing the concept of encapsulation. Candidates should be given the opportunity to implement a range of common collection ADTs using more than one data structure.

Outcome 4

The final Outcome introduces Candidates to the concept of generics in order to allow them to understand how standard generic collection classes can be used to implement collection and aggregation associations. They should be taught how to iterate through collections and how to select appropriate collection classes to implement associations dependent on the scenario. The scenarios selected should include scenarios that would relate to the ADTs introduced in Outcome 3.

This Unit covers some of the skills described for a pre-entry/junior technician role in the National Occupational Standards — IT and Telecoms (2009). The main areas covered correspond to disciplines 4.7 Systems Design, 5.1 Systems Development, 5.2 Software Development and 5.3 IT/Technology Solution Testing. There are also ample opportunities within the Unit to address a range of skills at both foundation and intermediate level that are described in the National Occupational Standards for IT Users v3. The most likely area to be covered would be IT Software Fundamentals and Using the Internet.

Higher National Unit specification: support notes (cont)

Unit title: Software Development: Data Structures

Guidance on the delivery of this Unit

Although this Unit has been designed to be programming language independent, it will require the use of an object oriented programming environment that has an appropriate generic collection class library. At the time of writing Java or any of the.Net languages that support the.net generic collection class library would be appropriate.

Outcome 1

This Outcome should be delivered first as it provides candidates with the fundamental concepts of how data is stored within computer systems. This knowledge will help prepare Candidates for the later Outcomes.

Candidates should be introduced to both signed and unsigned integer data types and be made aware of the potential problems of type conversion between the two. The use of standards in defining how data is stored should be emphasised to encourage candidates to think about cross platform compatibility. Candidates should be aware of how floating point numbers are stored but should not be expected to memorise the conversion methodology. Candidates should be aware of how the storage of user defined data types is dependent on the simpler types of the record fields. Strings should be introduced by illustrating how characters can be stored using ASCII or Unicode (specifically UTF-16) and then introducing the concept of character arrays. String classes could then be covered by utilising the debug facility of the development environments to allow candidates to view the component attributes. The additional methods provided can be highlighted using the environment's intellisense options (if available). One possible method of introducing 2d arrays is by looking at bitmap images — this could then also lead on to standard bitmap file types such as JPEG. The differences between text and binary files can be highlighted by illustrating how text files are readily readable by text editors and/or from memory dumps as compared to how binary files need to be opened by an application that 'knows' how the data is stored. The range and purpose of standard file types should include standard bitmap image, vector image, sound, video and compression formats. The basic format of xml data documents should be introduced to illustrate how this file format allows data to be transferred in a compatible manner.

Outcomes 2 and 3

It would make sense to deliver these two Outcomes in tandem. A possible approach would be to start by looking at linear array data structures and using implementation exercises to allow candidates to build a linear array data structure for a simple data type (integer or character). The linear and binary search techniques could then be illustrated by using both paper based tracing exercises and implementation exercises. This also allows you to introduce the big O notation for analysing the efficiency for algorithms. Graphs could be used to contrast the time taken for the worst case search for both the linear (directly proportional to size O(n)) and the binary search (logarithmically proportional to size O(log n)). It might be sensible to introduce the concept of a hash table here to let candidates see how closed hash tables allow O(1) searching algorithms. The concept of a circular array data structure could then be introduced and an implementation exercise given out to those who are ahead, allowing consolidation time for any candidates who are struggling.

This could be followed by looking at the concept of an ADT and using implementation exercises to build a simple ADT such as a Date or Weight class. If C# is used as the implementation exercise, this could be built using a struct. This option would also allow you to emphasise the difference between by reference and by value parameter passing. The concept of a Stack and a Queue could then be introduced. Candidates could use implementation exercises to develop interfaces for a Stack and Queue of integers (or characters) and then produce classes that implement the interfaces and use the linear array data structure developed earlier. This would allow candidates to understand the concept of interfaces preparing them for using standard collection class libraries. It would also be possible to cover some of the assessment criteria for the practical based assessment.

The concept of a sorted list could then be introduced, allowing you to cover the sorting algorithms. The simpler in place O(n) sorting algorithms (eg Bubble and Selection) could be demonstrated using a mixture of animations and implementation exercises. The more complex O(log n) algorithms (eg merge and quick sort) could be demonstrated by using animations to illustrate the vastly improved time efficiency. This is also an opportunity to introduce the concept of recursive algorithms to help explain the increased space complexity. One method of doing this would be to look at an iterative and a recursive factoral algorithm. Both could be coded and then stepped through using the debug facilities in the chosen development environment allowing the candidates to observe the recursive methods being pushed onto the dynamic memory stack. The assessment criteria for part b of the knowledge based assessment could be covered here.

The concept of a linked list data structure could be introduced by looking at how they work (reinforcing the concept of dynamic memory allocation) and then using implementation exercises to build a single linked list (SLL) data structure for a simple data type (integer or character). This data structure could then be compared to the linear array data structure to illustrate the advantages and disadvantages of both data structures. It could then be used to replace the linear array data structure used earlier to implement a Stack. This will reinforce the concept of an ADT by illustrating how the method signatures remain the same although a different data structure has been used in the implementation. You could then discuss how the SLL would not be suitable for a implementing a Queue and then introduce the concept of a double linked list (DLL). This again gives an opportunity for consolidation by allowing those that are ahead to implement a DLL data structure whilst allowing consolidation time for any candidates who are struggling. Candidates could then implement a Queue using a given DLL data structure that implements the Queue interface developed earlier. Again there are opportunities here to cover some of the assessment criteria for the practical based assessment.

Perhaps the best way to introduce the concept of a tree data structure is to use an implementation exercise where the candidates develop a binary search tree for holding a simple ordinal data type (eg integer or character). This could be reinforced by using animations to illustrate how they work.

The concept of a Set and a Map ADT could be introduced at this point by using animations. These could then be implemented later once the candidates are familiar with the generic standard class libraries.

There are many resources available on the web that could be used to help illustrate the concepts covered in these Outcomes. Perhaps one of the most useful starting points is the NIST Dictionary of Algorithms and Data Structures.

Outcome 4

The final Outcome concentrates on providing candidates with the skills and knowledge to use the standard collection classes provided with development environments. Candidates should be introduced to the concept of generics before starting to use the libraries. This also allows you to illustrate how generics can be used to develop more powerful ADTs that can be used for any data types in a type safe manner. You could illustrate this by demonstrating how the standard class libraries could be used to implement a Set and a Map.

Perhaps the most sensible manner to approach this Outcome is to work through some examples starting from simple class diagrams showing a single collection or aggregation association. Candidates could then undertake completion exercises where they are given the classes and then implement the associations and related methods.

There are opportunities to use some of these exercises as part of the assessment criteria for the practical based assessment.

Guidance on the assessment of this Unit

All of the assessments for this Unit could be undertaken using e-assessment. The closed-book assessment for Outcome 1 lends itself to a standard online objective assessment. The assessment for Outcome 2 (under supervised conditions) could make use of e-documents. The practical assessment for the Unit could be undertaken using an e-portfolio with links to the completed implementation exercises and test data.

Assessment Guidelines

Outcome 1 — closed-book Knowledge Based Assessment

This closed-book assessment covers some the concepts studied in Outcomes 1 and could be conducted using a knowledge based assessment such as a closed-book objective test consisting of 30 questions with a one hour maximum time duration. The assessment should include a range of questions covering each of the topics below:

- Minimum of six questions covering unsigned integers, signed integers, ASCII, Unicode, floating point numbers and Boolean.
- Minimum of five questions covering the representation of structured data types including string, record, table, one-dimensional array and two-dimensional array.
- Minimum of two questions covering static and dynamic memory allocation
- Minimum of four questions covering standard file types for images, sounds, video and compression
- Minimum of two questions covering XML data file structure
- Minimum of four questions covering data structures including Arrays, Linked Lists, Binary Tree and Hash Table

The questions presented should change on each assessment occasion and care should be taken to ensure that there is an appropriate mix of question cognitive levels for an SCQF level 8 Unit. A range of objective question types could be use although it is unlikely that simple true/false questions would be appropriate.

The assessment should be undertaken in supervised conditions and is closed book. A Candidate should complete this assessment within one hour. Candidates may not bring to the assessment event any notes, textbooks, handouts or other material.

Candidates should answer at least 18 of the questions correctly to pass the assessment.

An alternative approach to this assessment would be to use a closed-book short response style assessment abiding by the Evidence Requirements.

Outcome 2 Assessment under Supervised Conditions

One approach to this assessment would to use a series of three desk checking exercises covering at least one searching algorithm and at least one sorting algorithm. Candidates should be given the algorithm and data set. Each desk checking exercise should have a maximum duration of 30 minutes.

An alternative approach would be to use the debug facilities within the chosen development environment to allow the candidates to step through implementations of the algorithms. They could use the trace to count the number of comparisons and/or swaps as appropriate to the algorithms. Some candidates may well realise that they could edit the code to do this and this approach should be encouraged. The algorithms used must include at least one searching and at least one sorting algorithm. Candidates could provide recordings of the traces as evidence. It is envisaged that each trace would take no longer than 30 minutes.

This assessment must be undertaken in supervised conditions.

Practical Based Assessment

The practical based assessment for the Unit could be undertaken by using a series of development exercises that the candidates build into a portfolio of evidence. The guidance on delivery section illustrates where these exercises could be interspersed with the learning process. A possible approach would be to split the assessment into four exercises as detailed below.

- Develop interfaces (method signatures) for two ADTs selected from Stack, Queue, Set, List and Map

  During the delivery, candidates could be shown how to implement interfaces for one or more of the ADTs and then be given an exercise where they have to develop interfaces for two ADTs given the static model. The method signatures developed for the interfaces should be kept to the minimum required to identify the specific ADT, eg for a Stack this should include pop and push. This exercise should take no more than one hour.

- Use an Array data structure to implement an ADT selected from Stack, Queue or List

  This exercise should be given after the candidates have developed an array data structure during class based activities. They could then develop a class that uses the array data structure and implements one of the ADTs developed earlier. Candidates should be given the static model. This exercise should take no more than one hour.

- Use a Linked List data structure to implement an ADT selected from Stack, Queue or List

  This exercise should be given after the candidates have developed a linked list data structure during class based activities. They could then develop a class that uses the linked list data structure and implements one of the ADTs developed earlier. Candidates should be given the static model and the exercise should take no more than one hour.

- Use standard collection classes to implement a Map or a Set.
- Use standard collection classes to implement collection and/or aggregation associations.
- Implement code that iterates through data stored in standard collection classes.
- Test implemented code using given test plans.

  This exercise should be given at the end of the teaching process, allowing the candidate to consolidate their understanding and skills. The exercise could take the form of a completion exercise where the candidates are given a starter project that might well include the GUI. The candidates should be given an appropriate UML model that will allow them to use standard collection classes to implement a relatively simple problem. Two possible scenarios that could be used would to develop a candidate database or address book.

Online and Distance Learning

It would be perfectly feasible to develop a range of blended learning material to support distance learners. Online technology such as e-learning objects and links from virtual learning environments could be used support this type of delivery. Support for distance learners could be provided by both synchronous and asynchronous communication technologies such as the use of virtual classrooms and forums.

Part A of the knowledge based assessment could be delivered using an online objective assessment and part B could make use of video to record stepping through the algorithms. The practical based assessment could make use of an E-portfolio. Care would need to be taken to ensure the authenticity of assessments undertaken by distance learners.

Higher National Unit specification: support notes (cont)

Unit title: Software Development: Data Structures

Opportunities for the use of e-assessment

E-assessment may be appropriate for some assessments in this Unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all candidate evidence and that conditions of assessment as specified in the Evidence Requirements are met, regardless of the mode of gathering evidence. Further advice is available in *SQA Guidelines on Online Assessment for Further Education (AA1641, March 2003).*

Opportunities for developing Core Skills

Although there is no automatic certification of Core Skills or Core Skill components in this Unit, there are opportunities for developing *Problem Solving* skills at SCQF level 6.

Disabled candidates and/or those with additional support needs

The additional support needs of individual Candidates should be taken into account when planning learning experiences, selecting assessment instruments, or considering whether any reasonable adjustments may be required. Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**

History of changes to Unit

Version	Description of change	Date
02	Amendment to support notes for Outcome 1 — 70% threshold reduced to 60%.	13/11/12

General information for candidates

Unit title: Software Development: Data Structures

Data structures are used in all but the simplest developments and hence are a topic that you will need to understand if you want to develop software applications. The Unit starts by allowing you to explore how different types of data are stored in computer systems and how the use of standards allows data to be transferred from one system to another system.

The Unit then introduces you to a range of data structures that can be used to store collections of data and illustrates how these can be used to implement some common abstract data types such as Queues and Lists. You will also be shown how the efficiency of searching for and sorting data can be improved using a range of algorithms. This should help you enhance your own skills in using and developing algorithms.

The final Outcome of the Unit will allow you to further enhance your programming skills by teaching you how to use the generic collection class libraries provided in development environments. These skills are essential if you want to be able to program object oriented designs.

The Unit is assessed using three assessments. The first assessment is a knowledge based assessment that helps to ensure that you have the background knowledge required to develop skills in using data structures. The second assessment looks at your ability to follow an algorithm. The final assessment consists of a number of implementation exercises that enable you to develop your implementation skills.