HIT2080 – Introduction to Programming

Learning Summary Report

Reuben Wilson (9988289)

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment (please tick)				X

Self-assessment Statement

	Included (please tick)
Learning Summary Report	X
Final versions of Test 1, Test 2, and Test 3	X
Completed Glossary	X
Pascal programs that demonstrate coverage of core concepts	X
C programs that demonstrate coverage of core concepts	X

Minimum Pass Checklist

	Included (please tick)
Extension and Core exercises that demonstrate good coverage of all core concepts	X
High quality glossary, with all required items of a good standard	X
All Core Tasks signed off in Doubtfire	X

Minimum Credit Checklist, in addition to Pass Checklist

	Included (please tick)
Code for non-trivial program(s) of own design	X
Description of program's structure (functions, procedures, types)	X
Screenshots of your program in action	X

Minimum Distinction Checklist, in addition to Credit Checklist

	Included (please tick)
Research report, and associated pieces	X

Minimum High Distinction Checklist, in addition to Distinction Checklist

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature:

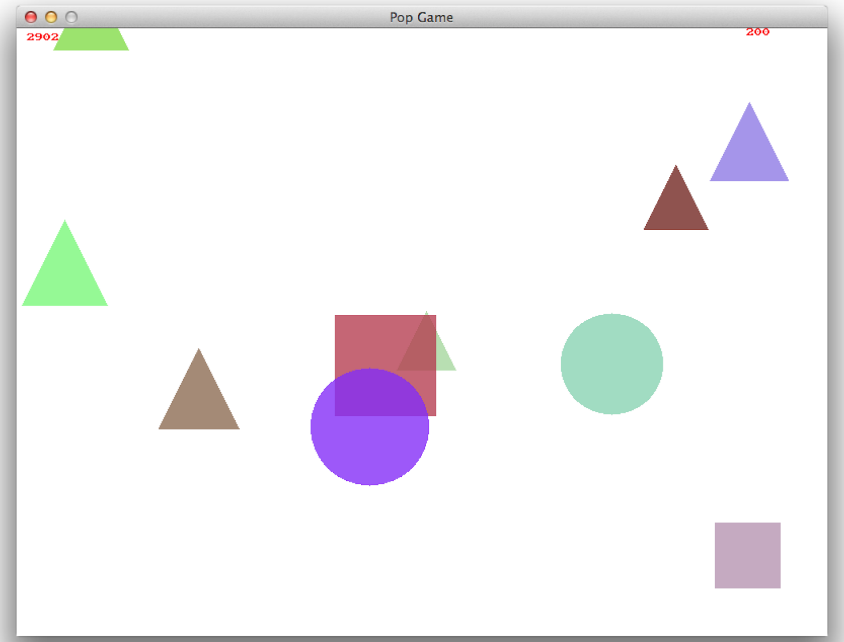
Introduction

This report summarises what I learnt in HIT2080 Introduction to Programming. It includes a self-assessment against the criteria described in the unit outline, a justification of the pieces included, details of the coverage of the unit's intended learning outcomes, and a reflection on my learning.

Overview of Pieces Included

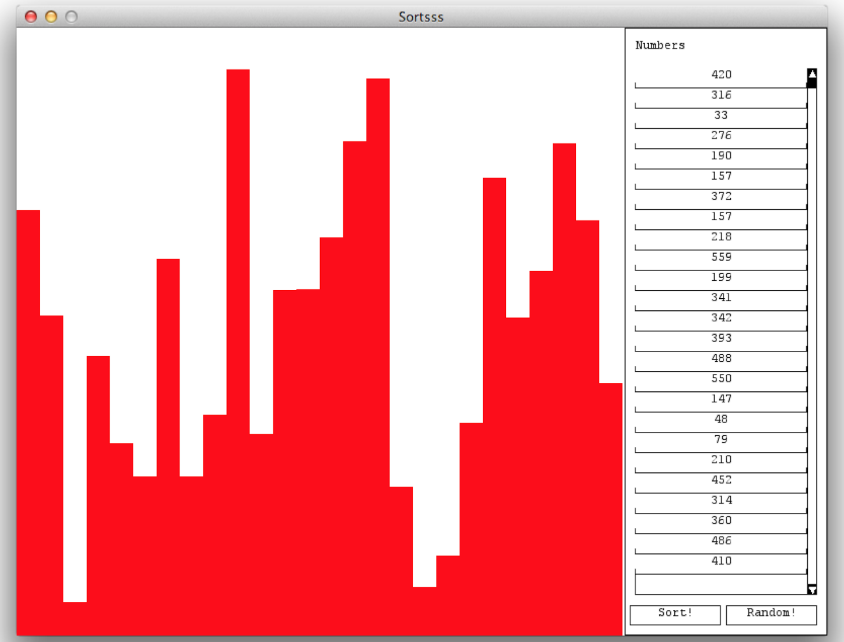
This section outlines the pieces that I have included in my portfolio…

1. **Glossary**: this contains descriptions of the terms, actions, and artefacts covered in the unit. My glossary contains descriptions of the programming statements, and artefacts for both C and Pascal.
2. **Test 1, 2 & 3**: these three tests demonstrate the application of theoretical concepts related to the course and the material covered.
3. **PopGame**: this small game draws a random shape to the screen, which the user clicks to 'pop'. The shape is a square, triangle or a circle; each shape has a random colour and location on the screen.



PopGame draws some random shapes to the screen, waiting to be popped!

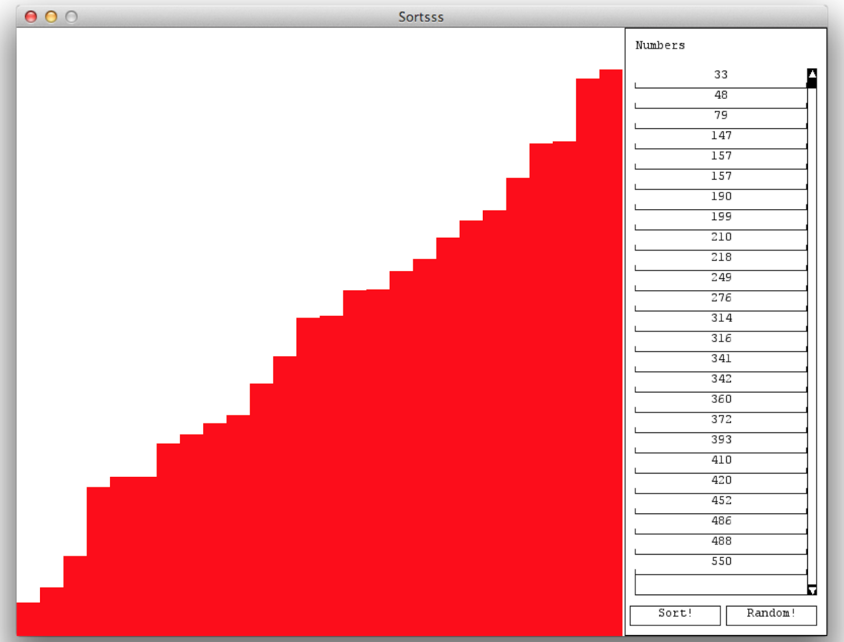
4. **Sort Visualiser**: is a small program, which makes use of SwinGames graphical user interface features. It generates a list of random numbers, and plots them on a bar chart. A sort button can be clicked in order to arrange the numbers and corresponding graphs in order from smallest to largest.



Numbers

```
420
316
33
276
190
157
372
157
218
559
199
341
342
393
488
550
147
48
79
210
452
314
360
486
410
```

Sort Visualiser running before any sorting has been done.

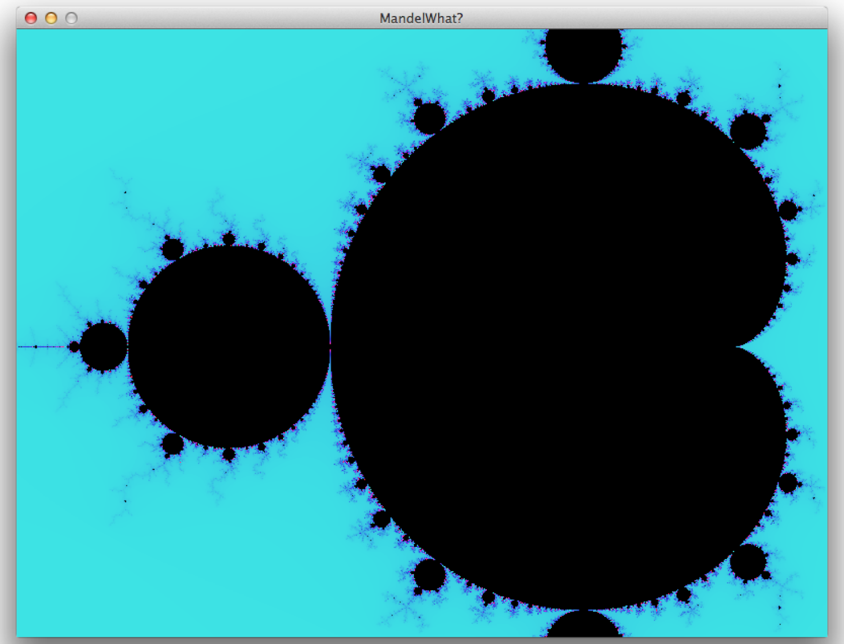


Numbers

```
33
48
79
147
157
157
190
199
210
218
249
276
314
316
341
342
360
372
393
410
420
452
486
488
550
```

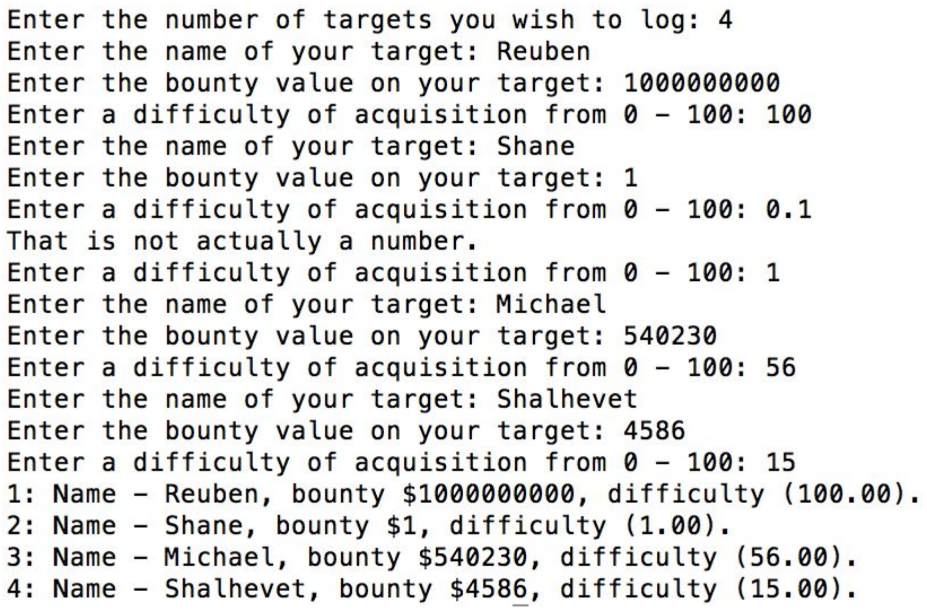
Sort Visualiser when sorting has been executed!

5. **Mandelbrot:** is a program, which draws an image of a Mandelbrot to the screen. The user can zoom in and out on the Mandelbrot in order to visualize the full extent of its mathematical beauty.



Mathematics represented graphically!

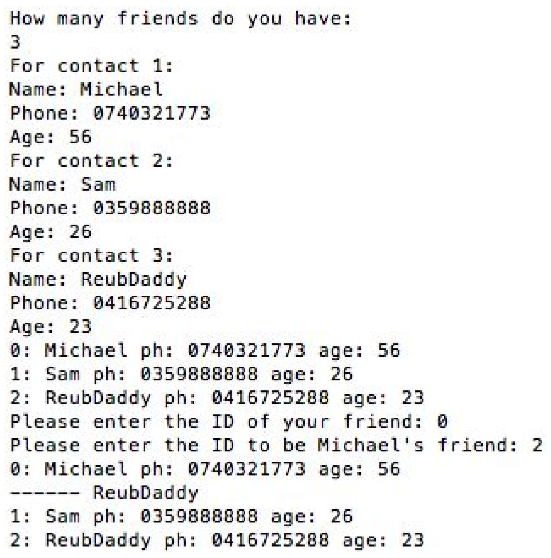
6. **Hit List:** is a small program designed to log targets for a bounty hunter. This program has been provided in both Pascal and C. It makes use of custom data types and arrays to track values specified within the program. The C version I have included even exports the data the user inputs to a text file for later reference.



```
Enter the number of targets you wish to log: 4
Enter the name of your target: Reuben
Enter the bounty value on your target: 1000000000
Enter a difficulty of acquisition from 0 - 100: 100
Enter the name of your target: Shane
Enter the bounty value on your target: 1
Enter a difficulty of acquisition from 0 - 100: 0.1
That is not actually a number.
Enter a difficulty of acquisition from 0 - 100: 1
Enter the name of your target: Michael
Enter the bounty value on your target: 540230
Enter a difficulty of acquisition from 0 - 100: 56
Enter the name of your target: Shalhevet
Enter the bounty value on your target: 4586
Enter a difficulty of acquisition from 0 - 100: 15
1: Name - Reuben, bounty $1000000000, difficulty (100.00).
2: Name - Shane, bounty $1, difficulty (1.00).
3: Name - Michael, bounty $540230, difficulty (56.00).
4: Name - Shalhevet, bounty $4586, difficulty (15.00).
```

Hit List is a terminal application. This is a demonstration of execution.

7. **Address Book:** is a small program designed to log information as contacts. It makes use of custom data types, arrays to track values specified within the program and pointers.



```
How many friends do you have:
3
For contact 1:
Name: Michael
Phone: 0740321773
Age: 56
For contact 2:
Name: Sam
Phone: 0359888888
Age: 26
For contact 3:
Name: ReubDaddy
Phone: 0416725288
Age: 23
0: Michael ph: 0740321773 age: 56
1: Sam ph: 0359888888 age: 26
2: ReubDaddy ph: 0416725288 age: 23
Please enter the ID of your friend: 0
Please enter the ID to be Michael's friend: 2
0: Michael ph: 0740321773 age: 56
------ ReubDaddy
1: Sam ph: 0359888888 age: 26
2: ReubDaddy ph: 0416725288 age: 23
```

Address Book is a terminal application. This is a demonstration of execution.

Coverage of the Intended Learning Outcomes

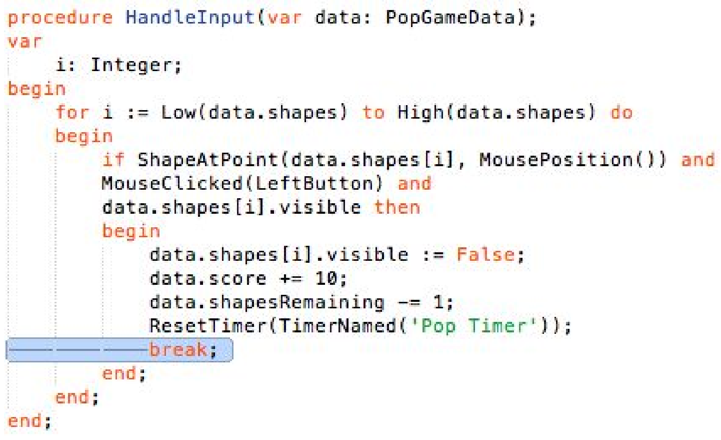
This section outlines how the pieces I have included demonstrate the depth of my understanding in relation to each of the unit's intended learning outcomes.

ILO 1: Reading and Interpreting

Read, interpret, and describe the purpose of supplied code, and identify (a) errors in syntax/logic, and (b) poor programming practices.

The following pieces demonstrate my ability in relation to this ILO:

- Test 1, 2 and 3: All three tests have hand execution exercises in them which required me as a programmer in learning to be able to read and interpret the intention of the coded exercises.
  Test 2 included a question, which demonstrated a poorly written function; poorly written due to horrible programming practices. I was required to re-write the function using conventional syntax and naming practices. This demonstrates the ability to identify poor programming practices as well as erroneous code and to apply practical solutions.
- PopGame, Sort Visualiser and the Mandelbrot programs were all interpreted and constructed by reading pseudo code and pieced together using iterative steps. The ability to read and translate pseudo code directly reflects an understanding of what the program requires and how it works.
- As is the case during development of any program, I encountered bugs and errors in syntax and logic, which I had to be able to identify and solve. One simple logic problem I faced with PopGame was the fact that if two shapes overlapped and the user clicked an overlapping area, both shapes would disappear. Using a break statement in the procedure, which controls user input, solved this simple issue. I feel this demonstrates being able to identify logic issues and apply practical solutions. Refer to the code below:



```
procedure HandleInput(var data: PopGameData);
var
    i: Integer;
begin
    for i := Low(data.shapes) to High(data.shapes) do
    begin
        if ShapeAtPoint(data.shapes[i], MousePosition()) and
        MouseClicked(LeftButton) and
        data.shapes[i].visible then
        begin
            data.shapes[i].visible := False;
            data.score += 10;
            data.shapesRemaining -= 1;
            ResetTimer(TimerNamed('Pop Timer'));
            break;
        end;
    end;
end;
```

ILO 2: Language Syntax

Describe the principles of structured programming, and relate these to the syntactical elements of the programming language used, and the way programs are developed using this language.

When Andrew Cain first introduced the learning group to the subject, HIT2080, correct usage of programming syntax and other programming conventions such as indentation and naming were stressed upon us due to the importance of being able to communicate your ideas clearly, making it easy for others to follow.

Throughout the semester, I followed some simple rules while developing all the small programs that I made in both Pascal and C. I made a zealot like effort to conform to using decent conventions in all my code. This is demonstrated in all three of my tests, which required me to handwrite small programs, as well as the other programs I am submitting in this learning summary report.

I believe that one may pick up my code and read it from start to finish and develop a clear understanding of what it is supposed to do due to naming of variables, functions and procedures as well as indentation, which makes obvious the intended flow of my code.

These simple skills, which were imposed upon me during the semester, are skills that I will continue to develop and utilise throughout the progression of my programming career.

ILO 3: Writing Programs

Write small programs making use of pointers, records, functions and procedures, and different parameter passing techniques.

All the programs I have submitted with this learning summary report include the implementation of functions and procedures as well as records, and in PopGame, an enumeration has been used.

Different parameter passing techniques have also been implemented, those being passing by value, by reference and by constant reference. I have a clear understanding as to how the different parameter passing methods work as this topic is what my research report is based on.

Pointers are something that are used heavily in C programs I have written, especially by my partially complete C version of my distinction project. I have used pointers in the Pascal version of Address Book included in this report.

I feel I can confidently implement and understand the programming artefacts that I have learnt and practiced all through the semester.

ILO 4: Functional Decomposition

Use modular and functional decomposition to break a problem down functionally, represent the resulting structure diagrammatically, and implement the structure in code as functions and procedures.

Functional decomposition is a programming convention I really adapted to well and practiced religiously. I cover this in my reflective statements below.

The programs that I have submitted with this report demonstrate the clear, practical use of functional decomposition. With most of the extension work, structure charts were included and all of these gave me a clear understanding as to how each function and procedure tied together to produce an intended outcome. PopGame is an excellent example as it included many iterations to produce the final, complete result.

My distinction program was the largest task I had to tackle, breakdown and understand. Refer to my custom project design report for design charts in relation to how it works.

Reflection

The most important things I learnt:

Prior to commencing this subject, I had very little experience with programming. I had a brief encounter with Python in high school, but anything learnt there was lost well before I began this semester.

Personally, I think I have come along way since week 1. I feel as if I grasped the concepts really well and one thing that I'm very proud of in particular is the fact that I've been able to learn how to think like a programmer. This was achieved by thinking about how the computer executes code rather than thinking about how it **should** execute code. I found that when I was assuming the computer should behave a certain way, the outcomes were far from my expectations. Instead, I began to imagine that I was the computer and how I would deal with certain instructions being asked of me.

The most prolific learning point in the development of my programming skill set was deciding to go through with a custom program project. The commitment required by me in order to complete this project played an integral part in becoming more confident and expressive with my code. My custom project enabled me to put to use and practice everything that we'd covered in the unit as well as exploring a vast number of internet communities for external ideas and inspirations. I want to be able to have something to offer, I don't want my work to be mediocre and I want to be recognised as an individual with potential and these motivators helped me to push through the extensive workload required of this subject.

One thing worth noting was my decision to change courses due to my positive experience in this subject. After assessing the course that I had chosen, I spoke with Willie and Andrew and expressed that I wanted to do more programming which was one thing the future of my initial course preference offered very little of. They both helped me realise what course I should be studying and assisted me with the transition.

The things that helped me most were:

I found my tutor, David Willie, and a volunteer at the help desk by the name of Simon to be extremely helpful in terms of providing constructive criticism. They offered me ideas and pointed me in the right direction on several occasions regarding my work, and this really helped to motivate me to want to do the best I possibly can. I think it's really important to have exceptional people on board who sacrifice their time for the greater good and benefit of others.

My interactions and collaborative work with other students also helped me to develop a keen programmers perspective. To be able to group together and discuss ideas and programming problems is hugely beneficial to me because it enables me to be able to communicate my ideas with others, as well as learn more about what they have to offer.

Although I didn't use the discussion board as much as I should have, I found that the few times I did, I got quick, reliable responses from either Shannon or Andrew. I can appreciate that is an extremely valuable resource and to be honest, not using it taught me to be able to understand that I should have used it more for any problems that I during the semester.

I found the following topics particularly challenging:

To be honest, I did not find one topic more challenging than the other. After each lecture and each lab, I took it upon myself to dissect all the code and concepts that we'd covered and written ourselves in order to gather a comprehensive understanding of how it all worked. I was sure to ask questions outside of class and at the help desk if anything came to my attention in regards to being misunderstood. Another reason why I feel I didn't face any overwhelming obstacles was because I practiced programming in my spare time and made small applications using all the topics that we'd covered.

I found the following topics particularly interesting:

The one topic, which really got me excited, was the implementation of custom data types. Prior to learning about records and structures, I was quite amazed with where I was at with using variables and arrays to store data. However, when we were all introduced to this concept of being able to store multiple values of different data types in the one entity, it really opened my eyes to the possibilities in terms of functionality and dynamics that could now be implemented into any program of my making.

I feel I learnt these topics, concepts, and/or tools really well:

The programming concept that I learnt really well and practised religiously is **functional decomposition**. I was always actively trying to reduce excessive amounts of code native to the functions and procedures I'd write in my programs. Andrew mentioned in a lecture once that if a function or procedure is more than twelve lines long, it is probably longer than it needs to be. This is most evident in my custom program project as well as in the extension pieces I have included in this document.

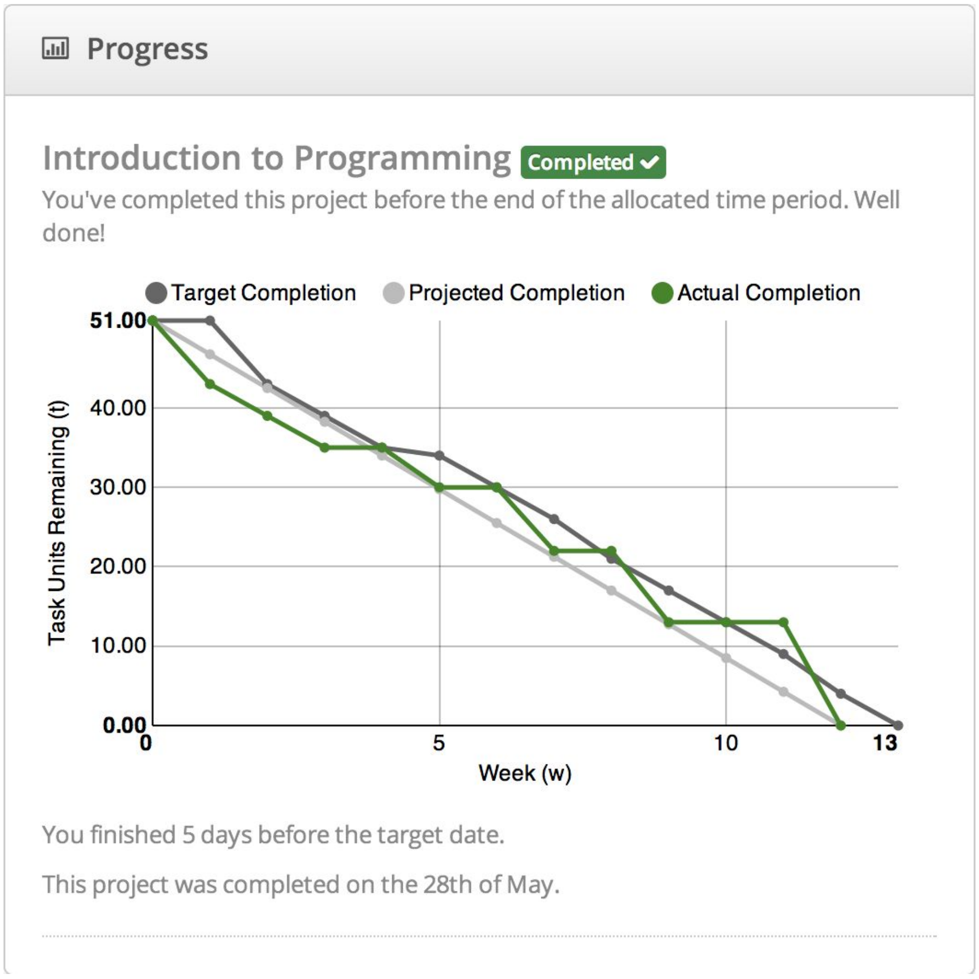
I also made sure that all my syntax followed the chosen language's conventions and indentation was always implemented in order to be able to follow the flow of the code when reading it.

I still need to work on the following areas:

The one thing I really want to be able to work on and master in the future is file input and output. I want to be able to make my programs much more dynamic by implementing file input output so that their use far extends the single terminal execution, after which, all data is lost.

One of the ideas I had for the future was making a simple car tracker, where I could store all appropriate details for my vehicle so that I could track fuel consumption averages, when to service the car based on mileage and average cost of repair. This kind of program would only be useful if all the data it relied upon could be written and read from a file efficiently.

My progress in this unit was ...:



Progress

Introduction to Programming Completed ✓

You've completed this project before the end of the allocated time period. Well done!

You finished 5 days before the target date.

This project was completed on the 28th of May.

This unit will help me in the future:

This unit is going to help me in the future in terms of applying acceptable programming practices where ever I work and what ever I work on. It's the simple fundamentals that I'm going to hold close which will really provide to be valuable assets in future career prospects as well as any kind of project that I establish with other people. These fundamentals include the decent programming conventions, which include the use of correct syntax and naming principles, proper indentation to ensure maximum communicability of code etc.

Asides from decent programming conventions, I'm going to actively expand the use of the programming concepts that we've covered this semester. I want to become completely adept in any programming language that I may have to use and I want to be able to learn how to make computers do exactly what I want them to do. This unit has laid the foundations that my motivations are built upon; knowing those foundations exist, all I have to do is continue to expand on my knowledge and make practical use of my desire to learn more.

If I did this unit again I would do the following things differently:

The main thing I'd like to change about my method of study in the future is having a more responsive, immediate approach to my workload. This semester, I found my ability to procrastinate a problem. Procrastination did not have a negative bearing on my overall work ethic; I still ended up producing quality work in a short amount of time however, leaving tasks late induced a certain degree of stress upon me which is something I do not need.

What really worked well for me was my ability to research in my own time. If I didn't understand something, or I wanted to learn more about a certain topic, I'd take the time to find out as much as possible about it and practice the theory obtained. An example of where I had done this during the semester was file input output. Before we touched on the topic in the lecture and lab material, I used it in one of my Pascal programs I made in my own time to store simple data in a text file.

Concluding Statement

In summary, I believe that I have clearly demonstrated that my portfolio is sufficient to be awarded a high distinction.

I feel that I have approached this unit with enthusiasm and a commitment to learn as much as I possibly could have during the semester. I was proactive about finding help when I needed it and constantly referring to the programming help desk to further my knowledge and discuss topics of interest to me with others who share similar outlooks.