NCEA Level 2

DIGITAL TECHNOLOGIES & HANGARAU MATIHIKO

Teaching and learning programme

Nutrition calculator – Programming

MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA

Developed by: Kath Langman, Hillcrest High School

Cover image © Crown

Published 2019 by the Ministry of Education
PO Box 1666, Wellington 6011, New Zealand

www.education.govt.nz
All rights reserved
Copyright © Crown 2019

Summary of the teaching and learning programme

Students will learn C# programming by completing a range of teaching and learning activities. They will then apply their knowledge to create a nutrition calculator. Teachers can guide students to use an alternative programming language, but this programme focuses specifically on C#.

By the end of this teaching and learning programme, students will be able to:

- gain an understanding of programming techniques (appropriate for NCEA Level 2)
- apply their knowledge to create their own programs
- plan and develop a nutrition calculator (assessment task).

Duration

8–10 weeks (1 term).

The big ideas

Learn a programming language and be able to apply this to a specified task.

Alignment to the New Zealand Curriculum

DTHM – Computational Thinking for Digital Technologies: Progress outcome 7

Students will use an iterative process to design, develop, document and test advanced computer programs.

NZC – Technology: Technological Practice, level 7

Outcome development and evaluation: Students should also be encouraged to use good planning for practice knowledge.

Links to other learning areas

Food technology, maths

Teaching and learning pedagogy

Students should be encouraged to manage their own learning as they move through the activities. The more they practise, the easier they will find completing the assessment task and other programs on their own.

Teachers can use a mixture of short practical tasks, readings and videos to learn the programming language.

Students should be making connections between their prior knowledge of programming concepts and other programming languages they may have learnt. It is expected that they will further develop their problem-solving skills and take their understanding to the next level.

Prior knowledge and place in the learning journey

Students will be learning C# from the beginning, but prior knowledge of programming languages and conventions is an advantage.

Resources required

- Resources provided for this programme
- Computer with Microsoft Visual Studio
- Tool for recording pseudocode planning, such as MS Word, Google Docs, paper
- Planning tools – only to assist students in achieving the task within the time frame. This could include a checklist in a document or a more advanced tool, such as Gantter or Trello.
- Ability to share resources with students (eg, online platform, Google Drive)

How might you adapt this in your classroom?

Use the first part of this resource to teach C#, then use a different assessment task or context that suits your students and environment, for example, an event that is happening annually in your area or school.

Use the ideas in the activities or tasks and adapt them for another programming language

Assessment

AS91896 Use advanced programming techniques to develop a computer program.

TERM OUTLINE

Specific learning outcomes (may include what will be covered)	Duration	Learning activities	Resources provided
Gain an understanding of programming techniques	5 weeks	Cover the following and ask students to practise: • Variables • Getting input, formatting output • Selection and repetition • Arrays.	Teaching notes and model answers: Powerpoint 1 - Introduction Powerpoint 2 - Getting input Powerpoint 3 - Arrays Powerpoint 4 - Conditions and repetition
Apply knowledge into a practice task	1–2 weeks	Give students a practice task, such as analysing given data. Work through the planning and breaking down of the task as a class or in small groups. Guide students with coding of task.	Practice task Teaching notes and model answers
Plan and develop a nutrition calculator assessment task	2–3 weeks	Inform students of the assessment task – work through the requirements: • Planning • Programming • Testing	Assessment task Planning template Testing log Nutrient file

ASSESSMENT TASK: NUTRITION CALCULATOR

Curriculum key concepts	*DTHM – Computational Thinking PO7:* use an iterative process to design, develop, document and test advanced computer programs. *Technology – Technological Practice, level 7: Outcome development and evaluation.*
Achievement standard(s)	91896: Use advanced programming techniques to develop a computer program
NCEA Level	2
Credits	6
Learning time guidance	Prior learning 6–8 weeks Assessment time 2–4 weeks
Length guidance if appropriate	Program code should be efficient: longer programs are not necessarily better. Testing log should show testing evidence against each of the specifications.
Due date	Teacher to insert

Achievement criteria

Achieved	Merit	Excellence
Use advanced programming techniques to develop a computer program.	Use advanced programming techniques to develop an informed computer program.	Use advanced programming techniques to develop a refined computer program.

ASSESSMENT TASK: NUTRITION CALCULATOR

Your task:

Aunty Kath's Bakery needs some help!

Kath has come up with new products to add to her already successful bakery. The law requires products to have nutritional information displayed as well as any allergens.

Kath already has the recipes and quantities, now she needs to be able provide information to the printers to meet the Food Standards Authority. This information includes: energy, protein, carbohydrate, sugars, fats and sodium.

She also needs to provide the number of servings, quantity of servings and total weight.

What you need to think about before you begin this assessment:

Ensure that you know how to:

- select **appropriate data types** for variables (eg, using an integer instead of a double if decimal values are not required);
- accurately **scope** variables (ie, variables should be defined as global, within a module, or within a block as required);
- select an appropriate indexed data structure (ie, **array** or list) and plan how to access data within that structure;
- use **derived values** instead of hard-coded values (eg, accessing the length of the array instead of hard-coding its value);
- write **comments** that explain how the program works
- plan the structure of a program with **self-contained modules** so as to avoid redundancy (idleness).
- Use file management to display **directories** and read **files**, as well as write to a file.

Remember:

- An advanced computer program:
  - uses variables storing at least two types of data (e.g. numeric, text, Boolean)
  - uses sequence, selection and iteration control structures
  - takes input from a user, sensors, or other external source
  - produces output
  - uses two or more advanced programming techniques.
- Examples of *advanced programming techniques* include:
  - modifying data stored in collections (e.g. lists, arrays, dictionaries)
  - storing multidimensional data in collections
  - creating methods, functions, or procedures that use parameters and/or return values
  - responding to events generated by a graphical user interface (GUI)
  - using non-trivial string manipulation
  - using additional non-core libraries.
- Before attempting this task, ensure that you have practised using the programming language that you will use for this purpose. You need to have practised writing programs, solving programming problems, testing and debugging to ensure that the program is correct on all inputs – expected, boundary and invalid.

ASSESSMENT TASK: NUTRITION CALCULATOR

What you need to do (follow these steps):

- Plan your program (planning is not assessed, but is recommended)
- Write pseudocode to do the following:
  - Open 'Nutrientfile.txt' and read the data into an array.
  - Ask the user for the ingredient to search for, eg, 'butter', and display a listing of all foods which contain 'butter' in their name.
  - Print the recipe to the screen: loop through your ingredients array and show the name and quantity.
  - Calculate the nutritional information for the recipe (collective ingredients):
    - Calculate and display the total grams of all ingredients
    - Ask the user the serving size
    - Calculate the number of servings
    - Create two arrays to store (a) Nutritional information for 100g, and (b) Nutritional information for the average serving size
- Print the nutritional information on the screen, formatted to be in three columns. See example below.
- Use console programming to code the above. Ensure your main method calls the other methods.
- Add user input checking. You can use ReadDouble, ReadInt and ReadString methods to check all user input.

What you will hand in:

1) Your program – ensure that your name and the date is at the top in the comments.

2) Your testing log – showing screenshots of your program working.

Extra Information:

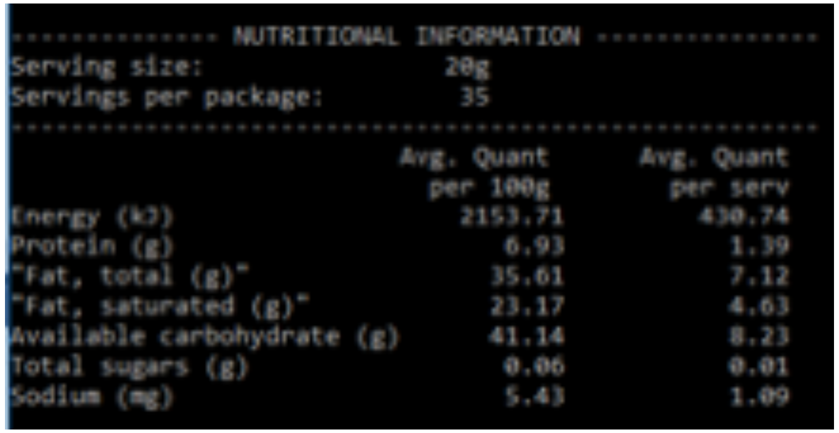
- Format of nutritional file:
  - Data is 'Tab' delimited
  - The first row contains a header
  - The data is as follows:

0	Food ID
1	Food name
2	Energy (kJ)
3	Protein (g)
4	Fat, total (g)
5	Fat, saturated (g)
6	Available carbohydrate (g)
7	Total sugars (g)
8	Sodium (mg)

ASSESSMENT TASK: NUTRITION CALCULATOR

- Example of nutritional information:

  04A10060 'Butter, salted' 3036 1.1 81.5 53.8 0 0 776



```
--------------- NUTRITIONAL INFORMATION ---------------
Serving size:                   20g
Servings per package:           35
--------------------------------------------------------
                             Avg. Quant      Avg. Quant
                               per 100g        per serv
Energy (kJ)                     2153.71          430.74
Protein (g)                        6.93            1.39
"Fat, total (g)"                  35.61            7.12
"Fat, saturated (g)"              23.17            4.63
Available carbohydrate (g)        41.14            8.23
Total sugars (g)                   0.06            0.01
Sodium (mg)                        5.43            1.09
```

Assumptions you can make:

- Liquid items will be given to you as a weight (grams).
- You do not need to save a recipe or recall it.
- You can simply search for an item using the 'contains' function.
- You can 'hard-code' the labels or read them from the first line.

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 5: Nutrition Calculator

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	EVIDENCE
Create program		
write code for a program that performs a specified task, using a suitable programming language	The student: creates a program using console programming that meets the specifications of the task uses C# or similar language	
create an advanced computer program	The student: uses variables storing at least two types of data (eg, numeric, text, Boolean) uses sequence, selection and iterative control structures (eg, for and while loops, IF statements) takes input from a user, sensors, or other external source (eg, console input, file input) produces output, eg: nicely formatted results onto the screen, writing to CSV file uses two or more advanced programming techniques, eg: using array to store values, creates extra methods that pass and return values such as searching the array for the given search item and returning selected item number	
follow conventions for the chosen programming language	The student: uses good coding layout: declaring global variables and constants at the top, main function is concise and only give instructions to call other methods, methods are broken down to do one task. Formats output appropriately. Files are opened and closed. No excess variables or loops are present. Names variables appropriately. Lower case is used. Names functions and methods using a verb and capital letter.	

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 5: Nutrition Calculator

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS. THE STUDENT:	EVIDENCE
Create program		
ensure that the program is a well-structured, logical response to the task use methods, functions, procedures, actions, conditions and control structures effectively.	The student: creates a solution that may contain methods to organise code, no redundant code is present. The code is clean, concise, and easily readable. used event procedures to perform an action.	
Layout and comments		
set out the program code clearly	The student: has used methods well, no excess space used	
document the program with comments	The student: explained the code with comments	
document the program with appropriate names and comments that describe code function and behaviour	The student: used frequent comments used appropriate variable names that describe the function of the variable, method names describe the function of the method (are verbs)	
use constants, variables and derived values in place of literals.	The student: used constants, variables and does not contain any hard-coded values.	

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 5: Nutrition Calculator

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS. THE STUDENT:	EVIDENCE
Testing		
test and debug the program to ensure that it works on a sample of expected cases	The student: creates a program that works without error for expected values tested and debugged the program to work with expected input, eg: data types, range of input	
test and debug the program in an organised way to ensure that it works on expected and relevant boundary cases	The student: tested and debugged the program to work with boundary input, eg: at of just beyond maximum or minimum limits	
comprehensively test and debug the program	The student: explained decisions made during testing	
correctly handle expected, boundary and invalid values.	The student: ensured program works for all input including invalid, eg: out of range, wrong datatype.	