Assignment

CSP2104 Object-oriented Programming with C++

Overview

Due date:

Part 1 (20 Marks): **14th April 2021 (9AM)**

Part 2 (30 Marks): **26th May 2021 (9AM)**

This assignment is to create an application that loads an English dictionary and then performs certain tasks using that dictionary. The assignment is worth 50 marks (or 50% of the total mark for the unit). The assignment contains tasks of varying complexity and students are expected to consult sources outside of the unit materials in order to accomplish them.

Related Learning Outcomes from Unit Outline

On completion of this assignment students should be able to:

- Formulate computer algorithms using the operations, control structures and classes provided in the C++ programming languages;
- Write, test and debug computer programs written in C++;
- Design and implement a class library as an abstraction, using the facilities of the C++ language and environment;

General

Whilst you are allowed to discuss aspects of the assignment with your colleagues, all material submitted for marking must be your own work. Plagiarism and other forms of academic misconduct will be dealt with as per the relevant ECU policy.

Your assignments will be processed with plagiarism detection software.

Submission Instructions

The assignment submission should be made via the Unit's Blackboard assignments site. Penalties will be applied for late submission.

Your submission must be a single zipped file containing your entire visual studio project. The file name should be in the format of <yoursurname_studentnumber.zip>. For example, smith_1001234.zip.

Your submission must be uploaded to BlackBoard before or on the due date unless you have an approved extension. Extensions must be applied for in writing before the due date as per University policy.

Also ensure that you keep a backup copy of all documentation and program files.
Submit your work as a visual studio project. Make sure it compiles.

Part 1 (10 Marks)

Write a program that when executed will parse the dictionary file, **dictionary2021.txt**, provided with this assignment. It will load each record from the file into a new instance of a **Word** class and add this instance to a list of Word objects contained in an instance of a **Dictionary** class. Your software will then display a menu of tasks it can perform. It will then prompt the user to enter a number corresponding to one of the following menu items:

Basic Tasks

1. Prompt the user to enter a word. If that exact word is in the dictionary, print the word's name, the word type using the following scheme: Noun (n.), Verb (v.), Adverb (adv.), Adjective(adj.), Preposition(prep) MiscWords(misc.),ProperNouns (pn.), NounAndVerb (n. v.), and the word's definition. If the word is not in the dictionary print 'word not found.'
    a. For words with multiple definitions, these should be separated with a line-break.
    b. Proper nouns should have their word name capitalised.
2. Find the word(s) with more than three 'z''s
3. List the words that have a 'q' without a following 'u' eg 'Iraqi'

Once a task has been executed, the program should return to the main menu.

Classes to Create (for Part 1)

This is to help you get started with your dictionary. Words in italics indicate what you MUST call your classes, methods or fields if you wish to receive marks for your effort. Each class should be implemented in its own file.

Create a *'Word'* class

- Fields:
    - *word* (string) The word in the dictionary
    - *type*
    - *definition* (string)
- Methods:
    - Getters for *word, definition and type*
    - *printDefinition – print the word's definition to the console in accordance with the requirement for basic task 1.*

Create a *'Dictionary_Part01'* class which

- maintains an appropriate STL container of Word objects
- loads the dictionary file into its array of Word objects
- Performs the tasks needed for this assignment.
- provides the methods:
    - *loadDictionary*() - (loads the dictionary file into its array of Word objects)
    - *Other appropriate methods so as to implement the tasks.*

Appendix 1 at the end of this document outlines the structure of the dictionary file.

Part 2 (30 Marks)

An extension of Part 1. Create a class Dictionary_Part02, which inherits from Dictionary_Part01. In Dictinoary_Part02, implement these additional tasks and add appropriate menu options for them to the menu of the program (the tasks from Part 1 should remain functional).

Basic Tasks:

1. List all words that are a noun and a verb e.g. “Phone”
2. List all words that are palindromes. e.g .”civic”

Intermediate tasks

3. Prompt the user for a word, and report all words that are anagrams of the word (e.g. “admirer” and “married”)
4. Guessing game – present the definition of a random noun and the length of that noun and ask the user to guess that noun, giving three tries. After the first incorrect guess, reveal the first letter of the word, after the second incorrect guess reveal the second letter.

Advanced tasks

5. Fun with Tri-Grams
    a. Using a new class, implemented in a separate file, create a class that encapsulates the following functionality: From any given text document (you can use the dictionary), calculate the probability of any character occurring after any combination of two characters (including where the first character is a space, i.e. the start of a word) and store these probabilities. Based on an input of two characters, return the three most likely characters to occur after the two characters. Use the class to generate 10 random words which sound like English words, but do not exist in the dictionary;

Marking Guide

Earn marks

Part 1	**Marks**	**Note**
Word class and Dictionary class correctly implemented.	4	
Dictionary file loaded and parsed correctly.	4	
3 Basic Tasks	12	
Part 2	**Marks**	**Note**
2 basic tasks	10	
2 intermediate tasks	10	
1 advanced task	10	Create classes and methods appropriate to the task.

Lose marks

Problem	**Marks (up to)**	**Note**
Poor programming practice.	5	Lack of commenting Magic numbers instead of constants Poor variable and function names

Input validation fails	5	I will try to break your program with dodgy input.
Program crashes	5	If your program crashes during execution then you will lose marks.

As an indication to the level of commenting needed – each module (class and method) should have a 'prologue' – a comment block summarising the function of the module, its input and output, who it was written by, and a date of creation. Within a module, comments should be included to explain what is happening in areas where its not obvious by looking at the code itself.

Appendix 1: Format of the dictionary file:

Notes about dictionary.txt

- 106,184 word definitions
- Text format (ascii)
- 3 lines per definition
    - Line 1: The word, followed by type in []
    - The definition (on one line) - multiple definitions of the same word separated by semicolon.
    - Blank line
- Word
    - Only uses characters A-Z a-z and the hyphen '-'
    - No punctuation or similar
    - Abbreviations are given without the '.' For example, "e.g." would be "eg"
    - No words are presented with spaces, the words are joined OR a hyphen is used. e.g. "bumble bee" is "bumblebee"
    - The language conversions are made eg: é=e æ=ae ö=o
    - No word has more than 45 characters
    - ALL words are in lower case, even proper nouns.
- Type, a single word (see table 1) enclosed in square brackets, appearing on the same line as the word, separated by a space from it;
- Definition
    - Multiple definitions are separated by a semicolon;
    - No Definition has more than 6014 characters
    - The language conversions are made eg: é=e æ=ae ö=o

• The definitions were not written by your lecturer or ECU. We take no responsibility for any inaccuracies or the content.

- The definitions are from the GCIDE project, made available under the terms of the GNU general Public License, GCIDE_XML is necessarily also published under those terms. See the file gpl.txt or <http://www.gnu.org/copyleft/gpl.txt>.

Table 1 Types present in the dictionary and the key Words used to denote them.

Word	Count	Meaning
v	9715	verb (“run”, “jump”)
n	59652	noun (“cat”, dog”)
adv	3413	adverb (“slowly”)
adj	28453	adjective (“big”, “glowing”, “inexpensive”)
prep	96	preposition (“beneath”, “against”)
pn	874	proper noun (“Perth”, “Edith Cowan”)
n_and_v	156	This word is a noun and a verb (“Rain”, “Phone”)
misc	3899	other words e.g. “shh”, “and”, but”, “arg”, “more” and prefixes