Worksheet 4

COMP110: Principles of Computing

Ed Powley

January 2016

Introduction

In this assignment, you will create three small C++ programs:

A. A console application implementing the word guessing game Hangman;

B. A console application implementing the 2-player strategy game Connect 4;

C. A graphical application which generates and displays the Mandelbrot fractal.

This worksheet tests your ability to translate various program notations (pseudocode, flowcharts, mathematics, narrative descriptions) into C++ code.

Submission instructions

The GitHub repository at the following URL contains skeleton projects for the three parts of this worksheet.

`https://github.com/Falmouth-Games-Academy/comp110-worksheet-4`

Fork this repository into your own GitHub account. To submit, create a GitHub pull request.

For the most part, this worksheet requires you to edit C++ code in the skeleton projects provided. **Please do not rename or delete the projects or files provided, and please do not create new projects.** For those questions which require you to produce artefacts other than C++ code, e.g. pseudocode or flowcharts, this material should be formatted in Markdown (possibly with embedded images) and added to your GitHub repository.

Information on using GitHub is available on LearningSpace, and elsewhere online.

Marking

Timely submission: 40%

To obtain the marks for timely submission, you must submit Part A by **6pm, Monday 18th January**, **and** Parts B and C by **6pm, Monday 8th February**. As with other worksheets, you may resubmit after these deadlines in order to address any correctness issues or satisfy further quality criteria. This 40% is awarded as long as you submit *something* by the deadline.

Correctness: 30%

To obtain the marks for correctness, you must submit working solutions for the following:

- Part A, Sections 1–3;
- Part B, Sections **??**–**??**;
- Part C, Sections **??**–**??**.

Note that this is a threshold: the full 30% is awarded for work which is *complete* and contains *no clear errors*. In particular you will not be penalised for trivial errors which do not affect the overall functioning of your programs, nor will you receive extra credit for highly polished solutions.

Quality: 30%

The extra quality criteria for this worksheet are as follows:

1. **Part A stretch goal.** You have submitted a working solution for Part A, Section 4.
2. **Part B stretch goal.** You have submitted a working solution for Part B, Section **??**.
3. **Part C stretch goal.** You have submitted a working solution for Part C, Section **??**.
4. **Presentation.** Your solutions are appropriately presented in GitHub, with descriptive commit messages and appropriate documentation in `readme.md` files. You have edited the provided skeleton projects, and refrained from renaming the provided files or creating new projects.
5. **Sophistication.** Your solution for **any one** of the stretch goals is particularly sophisticated, demonstrating mastery of C++ programming concepts appropriate to the task at hand.

Part A.
Hangman

In this part, you will implement a basic Hangman[1] game. The provided skeleton project contains code for reading a list of words from a text file, and choosing one at random. You will implement the rest of the game.

1.

Implement a function `char getLetter()` which does the following:

1. Prompt the user to enter a letter.
2. Get a line of input from the console, and take the first character.
3. If the character is a letter, convert it to upper case and return it.
4. Otherwise, display an error and go to step 1.

You may find the following functions useful: `isalpha`, `toupper`, and `std::getline`. Look online for details of how to use them.

2.

Implement the following algorithm as a C++ function. The algorithm takes the current partially revealed word, the secret word, and a guessed letter. It returns a new partially revealed word, in which the guessed letter has been filled in where it appears in the string.

```
procedure FillInLetter(partialWord, secretWord, letter)
    result ← empty string
    for i = 0, 1, ..., secretWord.length − 1 do
        if secretWord[i] = letter then
            append letter to result
        else
            append partialWord[i] to result
        end if
    end for
    return result
end procedure
```

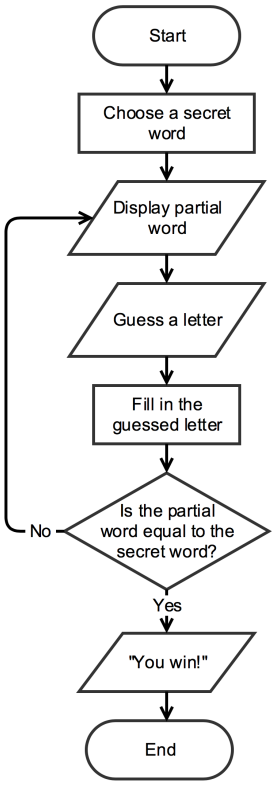
The following table gives some examples of possible input and output:

[1] `https://en.wikipedia.org/wiki/Hangman_(game)`

partialWord	secretWord	letter	result
"B-----"	"BANANA"	'A'	"BA-A-A"
"B-----"	"BANANA"	'E'	"B-----"
"-----"	"APPLE"	'L'	"---L-"
"-----"	"APPLE"	'B'	"-----"

3.

Implement a playable Hangman game, using the functions implemented above and structured according to the following flowchart:



4. Stretch goal

The game would be more interesting if the player had a limited number of lives in which to guess the word, and lost a life for guessing a letter which was not present in the word.

Modify the flowchart above to make this change. **Modify** your C++ program to reflect your revised flowchart.

Part B.
Connect 4

1.

Do a thing

2.

Do another thing

Part C.
Mandelbrot

1.

Implement the following algorithm in C++. Use a `switch` statement, taking advantage of the fact that casting a `double` to an `int` causes it to be rounded down. The `abs` and `fmod` functions are defined in `<math.h>`.

---

procedure HSVtoRGB(h, s, v)
  $c \leftarrow s \times v$
  $h' = 6 \times h$
  $x \leftarrow c \times (1 - \text{abs}\,(\text{fmod}\,(h', 2) - 1))$
  if $0 \leq h' < 1$ **then**
    $r = c;\ g = x;\ b = 0$
  else if $1 \leq h' < 2$ **then**
    $r = x;\ g = c;\ b = 0$
  else if $2 \leq h' < 3$ **then**
    $r = 0;\ g = c;\ b = x$
  else if $3 \leq h' < 4$ **then**
    $r = 0;\ g = x;\ b = c$
  else if $4 \leq h' < 5$ **then**
    $r = x;\ g = 0;\ b = c$
  else if $5 \leq h' < 6$ **then**
    $r = c;\ g = 0;\ b = x$
  else
    $r = 0;\ g = 0;\ b = 0$
  end if
  return $(r + v - c,\ g + v - c,\ b + v - c)$
end procedure

This algorithm converts a colour in HSV space to the equivalent colour in RGB space; all of r, g, b, h, s, v are assumed to be between 0 and 1.