



# Introduction to Programming

## Pass Task 6.1: C Text based Music Player

### Overview

This task allows you to demonstrate a simple text-interface application that allows the user to read in an album with multiple tracks then displays the album and track information before allowing the user to select a track to play. Unlike Pass Task 3 – this program should be implemented using the C programming language.

**Purpose:** Learn how to declare and work with arrays of records and enumerations in the C programming language.

**Task:** Demonstrate the use in C of:

- Arrays
- Enumerations
- Records/Structs
- 

in the context of the requirements for the Music application described in this document.

**Time:** This task should be completed before the start of week 12.

**Resources:**

- Programming Arcana
- Google
- Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))
  - [Learning a new language](#)
  - [Introducing Objects](#)

### *Submission Details*

You must submit the following files to Doubtfire:

- Code for the program, written in C and a screenshot of it working at the terminal.

Make sure that your task has the following in your submission:

- Code must follow the C coding conventions used in the unit (layout, and use of case).
- You are storing and working with multiple values in an array.
- You are using records and enumeration to store the values.
- The code must compile and you must capture a screenshot that shows it working in accordance with the requirements as described here along with any clarifications provided by your tutor.
- Your tutor is your client for this application. The application must aim to meet the client's requirements as specified below along with any clarifications provided by your tutor.

## Instructions

In this task you will build on the skills developed in your other Pass and Tutorial tasks.

For higher grades additional requirements are required. These are identified below.

You will be given feedback on how well you design your code as well as how well you name your artifacts. There is a minimum requirement for naming and design before your code can be accepted at any of the following grade levels – regardless of how well the code is functioning.

The different grades achievable for this task (within the Pass range) are:

Basic Pass Level (P) - 50

Middle Pass Level (C) - 53

Higher Pass level (D) - 55

Top Pass Level (HD) - 57

## Basic Pass Level Requirements (P)

The program must read in an album and up to 15 tracks for the album as well as a genre for the album. You can have as many genres as you like, but these must be defined using an enumeration.

Your application must:

1. Prompt the user for an album name;
2. Display the available genres (that you defined) in a numbered list;
3. Prompt for the user to select a genre for the album;
4. Read in the genre number that user entered;
5. Prompt the user to enter the number of tracks (max 15 – use a fixed size array);
6. Read in the number of tracks;
7. Repeatedly ask the user to enter track names until all the tracks have been entered.
8. Read in each track name;
9. Print out the album name and genre, then all the track names in a numbered list;
10. Allow the user to select a track to play
11. Display "The track you selected " then the track name " from the Album: " then the album name then " is now playing...".
12. Wait for the user to press enter.

At this level minimum validation is required. Just make sure your program does not crash if

incorrect values are entered and that all fields have an expected value (eg: perhaps have a default genre of "unknown" in case the user enters an incorrect value for genre).

## Middle Pass Level Requirements (C)

The program must read in an album and a number of tracks for the album as well as a genre for the album. You can have as many genres as you like, but these must be defined using an enumeration.

Your application must:

1. Prompt the user for an album name;
2. Display the available genres (that you defined) in a numbered list;
3. Prompt for the user to select a genre for the album;
4. Read in the genre number that user entered;
5. Prompt the user to enter the number of tracks (create an array just large enough to hold all the tracks – but have a maximum of 15);
6. Read in the number of tracks;
7. Repeatedly ask the user to enter track names and track file locations until all the tracks have been entered.
8. Read in each track name, read in each file location;
9. Print out the album name and genre, then all the track names in a numbered list;
10. Allow the user to select a track to play.
11. Display "The track you selected " then the track name " from the Album: " then the album name then " is now playing..."

At this level more validation is required. Use functions like `read_integer_range()` that will only accept an integer input in the valid range for both the number of tracks and the genre, otherwise prompting the user to re-enter an acceptable value.

**Note:** For reading in data you should use the `terminal_user_input.c` file which is in the resources for this task. You will need to implement the missing functions.

## High Pass Level Requirements (D)

The program must read in multiple albums and a number of tracks for the album as well as a genre for the album. You can have as many genres as you like, but these must be defined using

an enumeration.

Your application must:

1. Prompt the user for an album name;
2. Display the available genres (that you defined) in a numbered list;
3. Prompt for the user to select a genre for the album;
4. Read in the genre number that user entered;
5. Prompt the user to enter the number of tracks (create an array just large enough to hold all the tracks – but have a maximum of 15);
6. Read in the number of tracks;
7. Repeatedly ask the user to enter track names and track file locations until all the tracks have been entered.
8. Read in each track name, read in each file location;
9. Read in the next album.
10. Print out all the album names and genre, in a numbered list;
11. Allow the user to select an album to play.
12. Display the album name and genre then all the track names in a numbered list.
13. Allow the user to select a track to play.
14. Display "The track you selected " then the track name " from the Album: " then the album name then " is now playing..." then " from file location: " then the file location..

At this level basic validation is required. Use functions like `read_integer_range()` that will only accept an integer input in the valid range for both the number of tracks and the genre.

**Note:** For reading in data you should use the `terminal_user_input.c` file which is in the resources for this task. You will need to implement the missing functions.

## Top Pass Level Requirements (HD)

The program must read in an album and a number of tracks for the album as well as a genre for the album. You can have as many genres as you like, but these must be defined using an enumeration. The user can then select an album which your program should play using a call to an external program.

Your application must:

1. Prompt the user for an album name;
2. Display the available genres (that you defined) in a numbered list;

3. Prompt for the user to select a genre for the album;
4. Read in the genre number that user entered;
5. Prompt the user to enter the number of tracks (create an array just large enough to hold all the tracks – but have a maximum of 15);
6. Read in the number of tracks;
7. Repeatedly ask the user to enter track names and track file locations until all the tracks have been entered.
8. Read in each track name, read in each file location;
9. Print out the album name and genre, then all the track names in a numbered list;
10. Allow the user to select a track to play.
11. Display "The track you selected " then the track name " from the Album: " then the album name then " is now playing..."

At this level a higher of validation is required. Use functions like `read_integer_range()` that will only accept an integer input in the valid range for both the number of tracks and the genre. Also when the user enters the location for each track's file you must check that the file exists and prompt again if it does not. When the user selects a track to play your program must call an external program to play the track.

**Note:** For reading in data you should use the `terminal_user_input.c` file which is in the resources for this task. You will need to implement the missing functions.

End of Task