



Introduction to Programming

Pass Task 3.2: Text Based Music Player with Menu

Overview

This task allows you to extend your text-interface music application to provide a menu for users to use to play music.

Purpose: To design code that integrates various modules.

Task: Demonstrate the use of:

- Functions
- Procedures
- File Handling
- Structured Design

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 8.

Resources:

- Chapter 6 of the Programming Arcana
- Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))
 - [Using arrays to work with multiple values](#)
 - [Dynamically changing the size of an array](#)
- Syntax Videos
 - [Pass by Reference \(Var Parameters\)](#), [Pass by Reference \(Const Parameters\)](#), [Pass by Reference \(Out Parameters\)](#), [Arrays](#), [For Loop](#), [Dynamic arrays](#)

Submission Details

You must submit the following files to Doubtfire:

- Code for the program, and a screenshot of it working at the terminal, also any text files you use.

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

- Code must follow the Pascal coding convention 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 extend the implementation of your Text Based Music Player.

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)

Your extended Text Based Music Application must add the following functionality:

1. Display a menu that offers the user the following options:
 1. Read in Albums
 2. Display Albums
 3. Select an Album to play
 4. Update an existing Album
 5. Exit the application

Menu option 1 should prompt the user to enter a filename of a file that contains the following information:

- The number of albums
- The first album name
- The first artist name
- The genre of the album
- The number of tracks (up to a maximum of 15)
- The name and file location (path) of each track.
- The album information for the remaining albums.

Menu option 2 should allow the user to either display all albums or all albums for a particular

genre. The albums should be listed with a unique album number which can be used in Option 3 to select an album to play. The album number should serve the role of a 'primary key' for locating an album. But it is allocated internally by your program, not by the user.

Menu option 3 should prompt the user to enter the primary key (or album number) for an album as listed using Menu option 2. If the album is found the program should list all the tracks for the album, along with track numbers. The user should then be prompted to enter a track number. If the track number exists, then the system should display the message "Playing track " then the track name, " from album " then the album name. You may or may not call an external program to play the track, but if not the system should delay for several seconds before returning to the main menu.

Menu option 4 should allow the user to enter a unique album number and change its title or genre. The updated album should then be displayed to the user and the user prompted to press enter to return to the main menu (you **do not** need to update the file at this level)..

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 and incorrect value for genre).

Note: See the BasicReadWrite.pas code in this task's resources for file handling examples.

Middle Pass Level Requirements (C)

This level requires all the features of the Basic Pass level (as above) but with the additional/changed functionality for the option shown below:

Menu option 6: Exit needs additional functionality. If the user has changed any of the album information since the application started, then on exit you need to display the message "Updating album file information" and you need to rewrite all the album and track information into the original file name such that the file can be loaded (with the new information) next time the application is run.

At this level more validation is required. Use functions like ReadIntegerRange() 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. Make sure your program does not crash if the file is not found.

Note: For reading in data you should use the TerminalUserInput.pas file which is in the resources for this task. You will need to implement any missing functionality. See also the BasicReadWrite.pas code in this task's resources for file handling examples.

High Pass Level Requirements (D)

This level requires all the functionality of the previous two levels above, but in addition you need to add the following functionality to the options as indicated below:

Menu Option 3: You need to add the ability to search for an album based on genre or title.

Menu Option 4: The ability to update track information, including title and filename.

At this level basic validation is required. Use functions like `ReadIntegerRange()` 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 `TerminalUserInput.pas` file which is in the resources for this task. You will need to implement the missing functions. See also the `BasicReadWrite.pas` code in this task's resources for file handling examples.

Top Pass Level Requirements (HD)

This level requires all the functionality of the previous levels, but in addition you need to be able to create a playlist. The play list should allow multiple songs to be added to the list, then using an external program your application should play each song in the list, one after the other.

At this level a higher of validation is required. Use functions like `ReadIntegerRange()` 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 (you may or may not choose to use Swingame audio at this point).

Note: For reading in data you should use the `TerminalUserInput.pas` file which is in the resources for this task. You will need to implement the missing functions. See also the `BasicReadWrite.pas` code in this task's resources for file handling examples.

End of Task