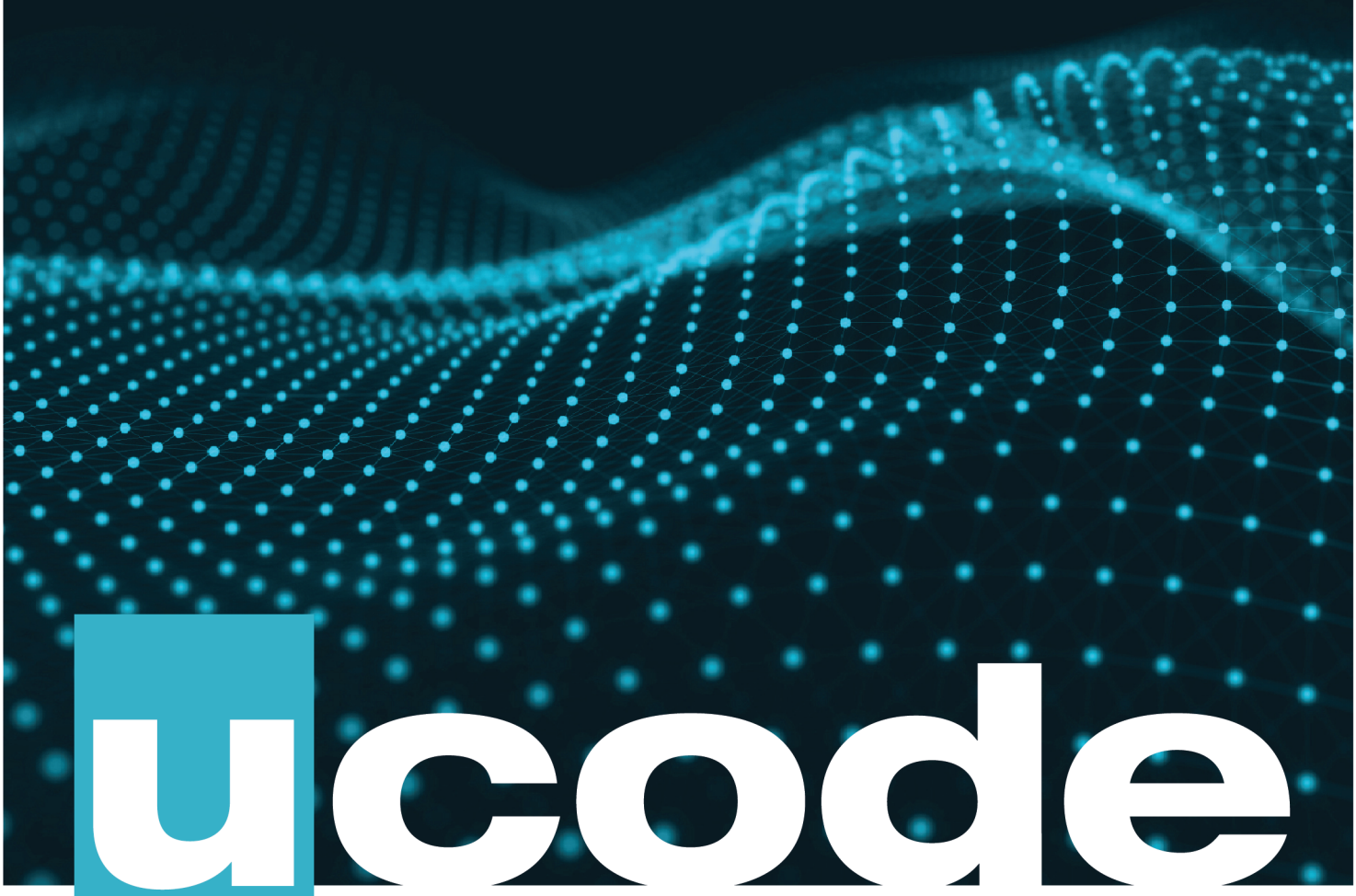
utag

Track C++

September 15, 2020



ucode

Contents

Engage 2

Investigate 3

Act: Basic 6

Act: Creative 8

Share 9

ucode

Engage

DESCRIPTION

Hi, friend!

How are your C++ skills? It's time to boost them in the first C++ challenge. You will see that C++ has more capabilities than you can imagine. Using your acquired knowledge, you will be able to solve applied problems relevant to the real world. It will also increase your C++ proficiency to a rather high level.

There are many ways to store needed information in the files. In this context, hidden information can be identified as metadata. There are many distinct types of metadata, including descriptive, structural, administrative, reference, and statistical. Metadata contain many essential details which can be used to summarize basic information about the specific file contents. That makes tracking and working with specific data easier.

The internal metadata is a part of the data they describe, so they could be edited only locally. In order to display metadata correctly, it is necessary to encode them right. Data encoding is an essential process which allows you to access hidden information about a large majority of files. Editing metadata is also a useful process that offers great encoding capabilities.

Metadata is used everywhere: images, videos, text documentation, web-pages, and so on. Imagine that even an audio file can store auxiliary information. So-called audio tags contain information about the artist, album, year, etc.

It's cool to have the ability to encode different types of files. It opens new horizons! The number of features and capabilities you can implement is too great to be calculated or estimated.

Don't delay, this is your first real C++ challenge.
We wish you success and inspiration!

BIG IDEA

Metadata.

ESSENTIAL QUESTION

How is metadata stored in files?

CHALLENGE

Create a tag editor.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is metadata? What types of metadata are there?
- What are metadata containers?
- Where can metadata be saved?
- What kind of information can be stored in files?
- What is an audio file?
- What audio file extensions do you know?
- What is an mp3 file?
- What are tags in mp3 files?
- What tag editors do you know?
- What libraries for reading and editing metadata exist in C++?
- What is the difference between `ID3v1` and `ID3v2`?
- How many bytes does the first `ID3` format have?
- What is an enhanced tag?
- What is the difference between `ID3`, `APE` and `Vorbis comments`?
- How are audio tags encoded?
- What audio tags have the longest length of the string in `ID3` container?
- What is the overall tag structure?
- How to create the best UI and UX?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read the story.
- Define the terms of the challenge (shortcut, toolbar, product, etc.).
- Discuss the challenge with the students that have already started.
- Brainstorm ways to make your program actually useful for the user.
- Analyze various audio file encodings.
- Analyze existing GUI tag editors (EasyTag, Ex Falso, Kid3, MusicBrainz Picard, puddletag, Mp3tag, Jaikoz, etc.).
- Investigate which features are a must-have in any tag editor.
- Research the problems that users have with tag editors.

- Think about extra features in advance.
- Make a plan for this challenge. You may use various project management tools such as `Trello`, which help to manage your work.
- Read about `README` and why it is needed.
- Clone your git repository that is issued on the challenge page.
- Start to develop the solution. Make improvements. Test your code.
- Explore new things.
- Think about making a specialized tag editor that has basic features as well as solving at least one real-world problem that other tag editors have not solved yet.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.
- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- You can proceed to `Act: Creative` only after you have completed all requirements in `Act: Basic`. But before you begin to complete the challenge, pay attention to the program's architecture. Take into account the fact that many features indicated in the `Act: Creative` require special architecture. And in order not to rewrite all the code somewhen, we recommend you initially determine what exactly you will do in the future. Note that the `Act: Basic` part gives the minimum points to validate the challenge.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Put the `README` file at the root of the repository.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- Complete tasks according to the rules specified in the Google C++ Style Guide. But there are few exceptions for the guide listed below:
  - you can use `#pragma once` directive instead of `#ifndef ... #define`
  - variables can be written in `mixed case`
  - class data members must begin with `m_` prefix (m for member)
  - indent 4 spaces at a time
  - each line of text in your code must be at most 120 characters long
- Compile files with commands `cmake . -Bbuild -Wdev -Werror=dev && cmake --build ./build` that will call `CMake` and build an app. There must be no errors or warnings.
- Your project must be built with `CMake 3.18.2`.

- Your project must be built with the following compile options: `-Wall -Wextra -Werror -Wpedantic`.
- Use a project structure consisting of logical parts: the main app, own libraries and 3dparty libraries. Each individual unit should know only about its resources and, if necessary, link other libraries to itself.

```
>tree project_dir --dirsfirst --charset=ascii
project_dir
|-- app
|   |-- src
|   |   |-- CMakeLists.txt
|   |   `-- ...
|   |-- resources
|   |   `-- ...
|   |-- CMakeLists.txt
|   `-- main.cpp
|-- lib1
|   |-- CMakeLists.txt
|   `-- ...
...
|-- libN
|   |-- CMakeLists.txt
|   `-- ...
|-- 3dparty
|   |-- lib1
|   |  |-- CMakeLists.txt
|   |  `-- ...
|   ...
|   |-- libN
|   |  |-- CMakeLists.txt
|   |  `-- ...
|-- CMakeLists.txt
`-- build.sh
```

- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- If you have any questions or don't understand something, ask other students or just Google it.

Act: Basic

SUBMIT

`according to the described structure in the ANALYSIS`

BINARY

`utag`

DESCRIPTION

Create an app to read and write tags of mp3 files.

Implement a GUI app with a user-friendly interface.
The exact layout of the program is up to you, but app GUI must look good and be intuitive.
Seriously, we mean it.
Make a perfect UI/UX, otherwise users will not use your product.

After you've read this story to the end, start your implementation by creating a README file. In order for people to want to use your product, their first introduction must be through the README on the project's git page. By writing this file, you get some benefits:

- you have an opportunity to think through implementation without the overhead of changing code every time you change your mind about how something should be organized. You will have very good documentation available for you to know what you need to implement
- if you work with a development team and they have access to this information before you complete the project, they can confidently start working on another part of projects that will interact with your code
- everyone can figure out how to launch your product. And also you can always help yourself to remember how your project works

Of course, for this, you should learn to write a nice-looking and helpful README file. There are several links that can help you:

- Make a README
- How to write a readme.md file?
- A Beginners Guide to writing a README

We recommend using this feature. As a developer, you need this. It will also be evaluated at the assessment.

So, the app must be able to:

- accept a directory as a command-line argument
- define audio file format by extension `.mp3`
- display a list of audio-file names in the requested directory
- display only audio-file formats supported by the app
- display/edit these audio file tags in the GUI window:
  - `Artist`

  - `Title`
  - `Album`
  - `Genre`
  - `Audio file path` (not editable)
- support audio files items sorting in alphabetical (ascending and descending) order by a certain tag

Error handling:

- if a command-line argument is invalid, then the respective error message must be displayed with the GUI
- if an mp3 file does not have read/write permissions, then the respective error message must be displayed with the GUI
- the app should catch any other errors and display them only with the GUI

Libraries:

- you are free to use any library to implement reading/writing audio file tags
- you are free to use any library/framework to implement the GUI part (`SDL`, `GTK`, `Qt`, etc.)

Act: Creative

DESCRIPTION

It is the place where your imagination and creativity plays a major role. Implement additional features to make the program better and more unique. Listed below are a few ideas you can add to your program. You can come up with everything you want to improve your program. Creative features:

- supports more audio file formats e.g. `.flac`, `.wav`, `.ogg`, etc.
- opens the directory with the audio files through the GUI as well as through the command-line argument
- allows to edit more tags of audio files (year, track number, etc.)
- supports the display and edit of the lyrics
- supports the tracking of editing history
  - displays activity log
  - allows undo/redo changes
- edits an album image for an audio file and displays it via the GUI
- supports `Audio file path` editing
- supports bulk editing of the selected tags.
  For example, change the `Artist` tag for all selected audio files
- allows browsing the opened directory and all subdirectories recursively
- searches audio files in the directory by tag
- fully supports Unicode. Correctly displays any character from any language
- supports various themes for customizing the look and feel of the editor's entire user interface
- other creative features

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.