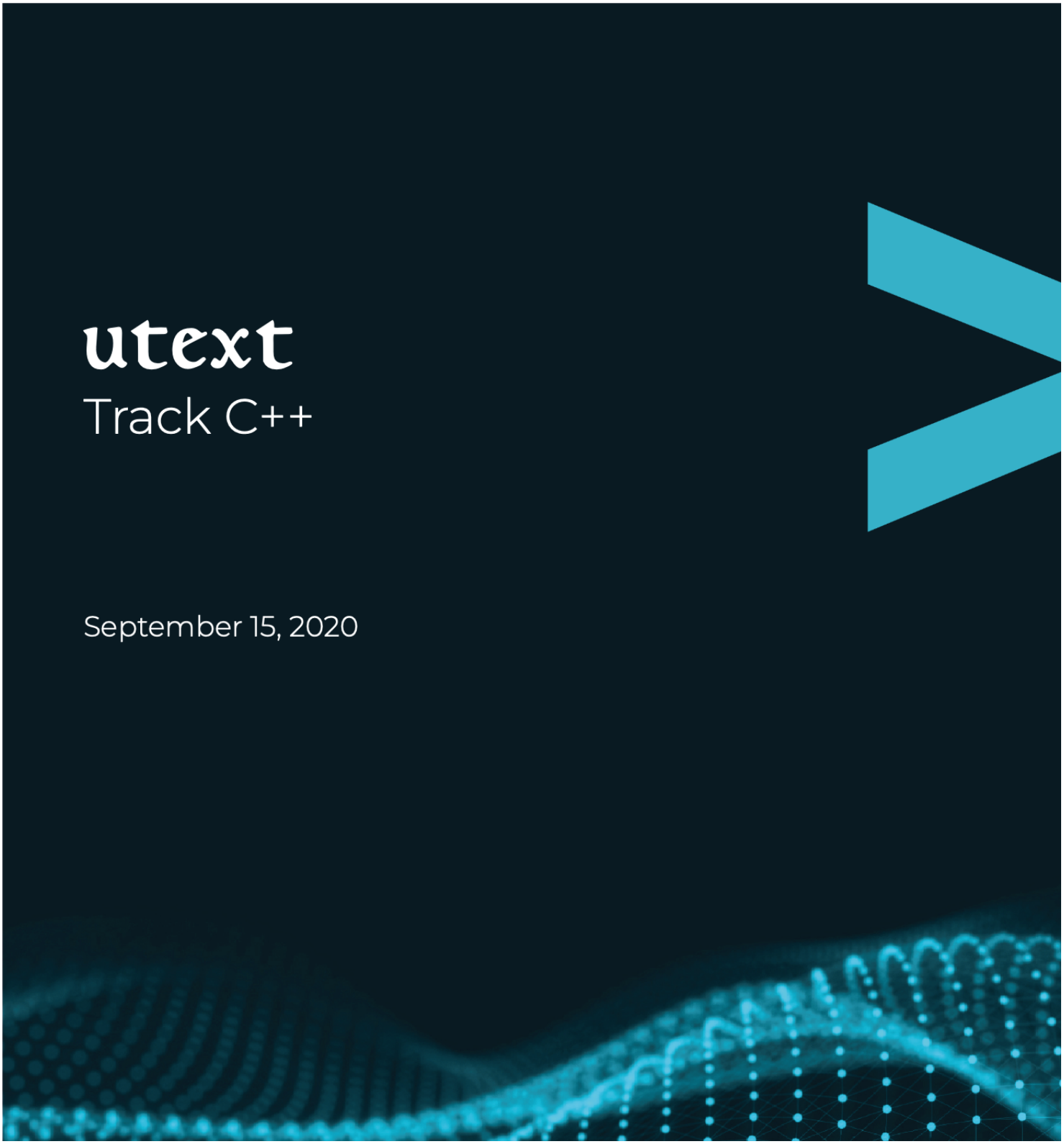
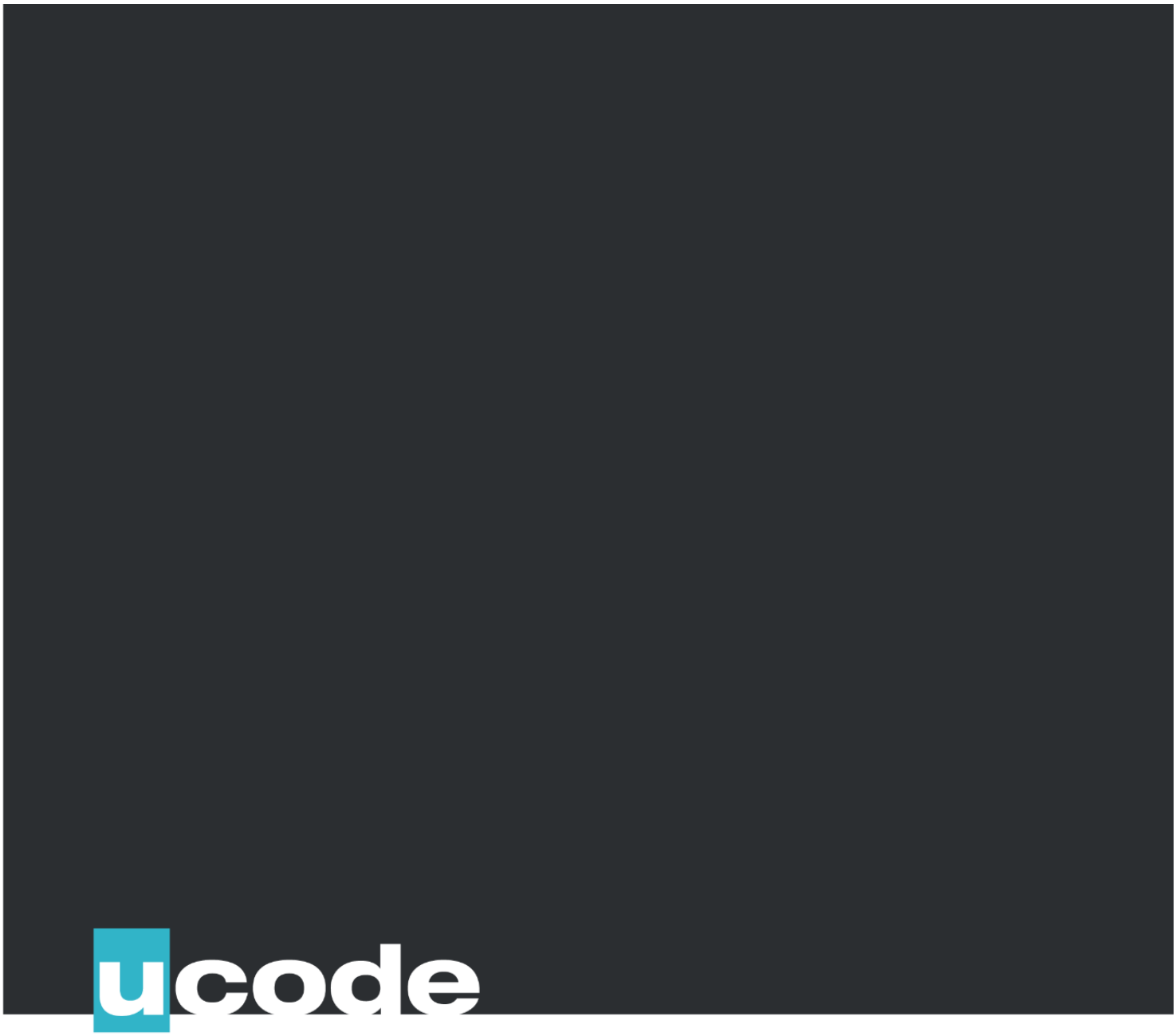

utext
Track C++
September 15, 2020

Contents


ucode


Engage
DESCRIPTION

Hi there!

How is your success in C++? Hope you are enjoying the learning process.

Working with digital information is a major component of people's activities. With the advent of the computer, the process of creating documents has become much easier. The first text editors appeared at the same time as the first serial PCs. These editors only allowed you to enter text information, edit it, and save it to the computer's memory. As well as improving the hardware of computers, text editors are also being upgraded. Now, we cannot imagine our lives without them, they are used everywhere: at work, in everyday life, in teaching and in programming.

People can do whatever they want using text editors: format, paste, replace, copy data and much more. These days, remote editors are particularly trendy, their functionality is especially usable. Modern editors include data conversion, text highlighting, content filtering, even plugin extensions. Well, it sounds like a fantasy, but it can be implemented with C++.

There are so many different features, so it is impossible even to predict their relevance in the
appropriate situations. It is difficult to find a multipurpose text editor, which would fully satisfy your needs. Just consider how many flaws there are in existing editors. Everything is in your hands! We offer you a great opportunity to create such a cool and useful product right now! Imagine your own editor with all the useful features you want. It's too cool to be real, but it's actually possible.
Good luck!

BIG IDEA

Digital content.

ESSENTIAL QUESTION

How to manage digital information effectively?

CHALLENGE

Develop a competitive text 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 the difference between various text editors?
- What features would you like to have in a text editor?
- In what way can the user interact with text?
- What common features do all text editors share?
- What keys do we combine when using shortcuts?
- What skills do you want to get?
- What is the best way to present directory structure?
- What resources do you need to start learning?
- What are regex and how can they improve your search?
- Vim or emacs?
- How to make a product unique and useful?
- How can `rope`, `gap buffer` and `piece table` data structures help to deal with editable text challenges?
- What algorithms of text processing do popular text editors use?
- How do collaborative text editors (e.g. Google Docs) work from the inside?
- What is the difference between plain text and rich text?
- How to deal with character encoding?
- How to deal with large files (log files) or huge lines (minimised JS)?
- Can the Memento design pattern be applied to create clean architecture?
- Can the Composite design pattern be applied to create clean architecture?

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.

- Meet the team.
- Define the terms of the challenge (text editor, shortcut, toolbar, product, etc.).
- Ask students who have already started this challenge what you need to figure out. Prepare your mind for an explosion.
- Analyze existing GUI text editors (MS Word, Pages, Google Docs, Sublime Text, Notepad, Notepad++, CLion, etc.).
- Investigate which features are must-have in any text editor.
- Research user problems with text editors.

utext | Track C++ **> 3**

- Brainstorm to find the solutions for users' pains.
- Make a plan where to start.
- Think about extra features in advance.
- Think about making a specialized editor that has basic features as well as solving at least one real-world problem that other text editors have not solved yet.
- Read the story, taking everyone's ideas on board.
- Clone your git repository that is issued on the challenge page.
- Try to choose the best solutions.
- Distribute tasks between all team members.
- Start to develop the solution. Offer improvements. Test your code.
- Communicate with the team on a daily basis, share information on time.

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.

- Challenge has to be carried out by the entire team.
- Each team member must understand the challenge and realization, and be able to reproduce it individually.
- It is your responsibility to assemble the whole team. Phone calls, SMS, messengers are good ways to stay in touch.
- Be attentive to all statements of the story.
- Analyze all information you have collected during the preparation stages.
- 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`

utext | Track C++ **> 4**

- 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.

utext | Track C++ **> 5**

Act: Basic

SUBMIT

`according to the described structure in the ANALYSIS`

BINARY

`utext`

DESCRIPTION

Create a GUI text editor. The program's GUI must have a good looking user interface and offer intuitive user experience. The UI/UX can be inspired by apps like Sublime Text, Notepad++, MS Word, Google Docs, etc., but don't stop your research here. Make your program unique.

The program window must consist of at least three parts:

- `Toolbar` shows control elements
- `Text area` displays file content
- `Tree area` shows directory and file structure

General features:

- open directories and single files
- list opened directory files and subdirectories in the Tree area. Subdirectories must be collapsible
- open files from the Tree area for reading/editing in the Text area
- save files after editing
- copy/cut/paste options in the Toolbar
- find and replace options in the Toolbar
- undo and redo options in the Toolbar
- keyboard shortcuts for all options from Toolbar. Key combinations must be user-friendly
- keyboard and mouse management of carriage position. There must not be any lags, delays, latencies or slowdowns during text typing/deleting and carriage movements
- ability to select a piece of text with the keyboard as well as the mouse

  - the selected text is highlighted
  - copy/cut/paste options are applied only for the selected piece of text
  - the selected portion of text is taken for search when find/replace option is applied
- horizontal and vertical scroll if the Text area is smaller than the content available

Error handling:

- if the file/directory has no 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/framework to implement the GUI part (`SDL`, `GTK`, `Qt`, etc.)
- text editor features must be implemented without third-party libraries
- some features from `Act: Creative` part may be implemented with third-party libraries, but their usage must be justified, e.g. source code analyzers for syntax highlighting or network operations

ucode

utext | Track C++ > 7

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:

- syntax highlighting for source code, markup languages, config files and other text that appears in an organized or predictable format
- regex-based find and replace options
- find and replace in multiple files in the opened directory and recursively all the subdirectories in it
- support of multiple carriages which allow printing at once in several places
- support for multiple windows. One instance of the program can launch multiple full-featured windows (e.g. as it is implemented in Sublime Text)
- basic text formatting (font weight, font style, font size, indentation, alignment, bullet list, etc.)
- text folding. Pieces of text that can be collapsed and something else can be shown there instead
- find matching parenthesis or bracket, taking nesting into the account
- line wrapping. Fit long lines into display area without horizontal scrolling
- autocompletion and auto-indentation
- split view of the Text area for multiple file editing, remembering the current-line cursor of each file
- ability to create/delete/rename files or directories from Tree area
- handle at least UTF-8 character encoding as well as support of other encoding formats
- a programmable editor where commands or macros can be input with the command line interface
- themes to change 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 understandi 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.


ucode
utext | Track C++ > 9