



**AGH – UNIVERSITY OF SCIENCE AND  
TECHNOLOGY**

Project documentation for

## **ToDo List App**

### **Object-Oriented Programming Language**

Computer Science and Intelligent Systems, second-cycle, II year

*Szymon Lepianka*

lecturer: Rafał Frączek

Kraków 2023

# 1. Project description

The project is a simple command-line Todo list application. The user can add new Todo items, mark Todo items as complete, and delete existing Todo items. This application provides a simple and easy-to-use interface for managing a list of Todo items and is a great tool for keeping track of tasks and activities. It can be used for personal or professional purposes, and can be easily modified and customized to suit the user's specific needs. The Todo items are stored in a list, and the list is saved to a file so that the Todo items persist even after the program is closed. The user can also view all the Todo items in the list, and the Todo items are displayed with their ID, description, and completion status.

The program uses C++ programming language and the standard library. The program uses a class `TodolItem` to represent a Todo item, with the Todo item's ID, description, and completion status being represented as private member variables. The class has getters and setters for these member variables, as well as a `save()` method that writes the Todo item's data to a file. The program uses a list container to store the Todo items, which allows for easy manipulation of the Todo items. The program uses file I/O to read and write the Todo items to a file.

The program also uses error handling to handle various error scenarios, such as when the file is missing or unreadable, and when the user inputs invalid options or ids. The program uses more meaningful variable names.

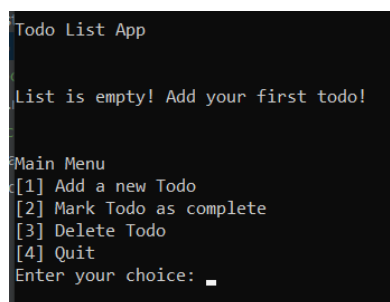
## 2. User's manual

This is the Todo List App. This program is designed to help keeping track of tasks and ensure that user don't miss anything important.

When you first run the program, it will automatically read any existing Todo items from a file called "todolist.txt" in the same directory. If the file is missing or unreadable, the program will continue with an empty list.

The main menu of the program displays a list of all the `TodolItems` in your list, along with their ID, description, and completion status. The list is displayed in the order in which the `TodolItems` were added. The main menu will also display the following options:

- Add a new Todo.
- Mark Todo as complete.
- Remove a Todo.
- Exit the program.

A screenshot of a terminal window showing the 'Todo List App' interface. The text displayed is: 'Todo List App', 'List is empty! Add your first todo!', 'Main Menu', '[1] Add a new Todo', '[2] Mark Todo as complete', '[3] Delete Todo', '[4] Quit', and 'Enter your choice: ' followed by a cursor. The background is dark, and the text is light-colored.

```
Todo List App

List is empty! Add your first todo!

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: _
```

To make a selection, enter the corresponding number and press Enter.

To add a new `TodolItem`, select option 1 from the main menu. You will be prompted to enter a description for the `TodolItem`. The program will check if the description doesn't include commas. If it's invalid, the program will display an error message. The new Todo will be added to the list and displayed on the screen.

```

Todo List App

List is empty! Add your first todo!

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: 1
Enter todo item description: example todo 1_

```

```

Todo List App

1 , example todo 1 , not done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: _

```

To mark a TodoItem as complete, select option 2 from the main menu. You will be prompted to enter the ID of the TodoItem you wish to mark as complete. If the ID is invalid, the program will display an error message and return to the main menu. The selected Todo will be marked as complete and displayed on the screen with the label "done".

```

Todo List App

1 , example todo 1 , not done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: 2
Enter Todo ID (to mark as completed): 1

```

```

Todo List App

1 , example todo 1 , done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: _

```

To delete an existing TodoItem, select option 3 from the main menu. You will be prompted to enter the ID of the TodoItem you wish to delete. If the ID is invalid, the program will display an error message and return to the main menu. The selected Todo will be removed from the list and the updated list will be displayed on the screen.

```

Todo List App

1 , example todo 1 , done
2 , example todo 2 , not done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: 3
Enter Todo ID (to delete): 2

```

```

Todo List App

1 , example todo 1 , done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: _

```

To exit the program, select option 4 from the main menu. The program will automatically save any changes you have made to the TodoItem list to the "todolist.txt" file before exiting. The program will then exit.

```
Todo List App

1 , example todo 1 , done

Main Menu
[1] Add a new Todo
[2] Mark Todo as complete
[3] Delete Todo
[4] Quit
Enter your choice: 4
Quitting...
Press any key to continue . . .
```

### 3. Compilation

Here provide information on how to build the project. Do not forget to mention whether the standard build is enough or whether a custom build is necessary. If the project works only in one operating system (eg Linux), please provide information about this fact.

This project was developed and tested using CLion on Windows, but it should be able to build on other platforms as well as long as the necessary dependencies are met. In order to build the project, you will need to have CLion and MinGW installed on your system. To build the project, open the project in CLion, and select the build configuration (debug or release) you want to use. Then, click the "Build" button in the top toolbar. The executable file will be generated in the corresponding cmake-build-debug or cmake-build-release folder. You can also run the program directly from the IDE by clicking the "Run" button.

If you wish to build the project from the command line, you can use the following commands:

```
cd path/to/project
cmake .
make
```

This will build the project in debug mode and create the executable file in the cmake-build-debug folder.

It should be noted that this project was developed and tested on Windows using CLion, and may not work correctly on other operating systems without modification. This project does not require any non-standard or additional actions to run. However, it is important to note that the program expects the file "todolist.txt" to be in the same folder as the executable. If this file is not present or if the program is unable to read or write to it, the program will start with empty list and try save "todolist.txt" on quitting.

### 4. Source files

The project consists of the following source files:

- *TodoItem.h*, *TodoItem.cpp* – declaration and implementation of the `TodoItem` class,
- *Utils.h*, *Utils.cpp* – declaration and implementation of miscellaneous functions used in the project.
- *Globals.h*, *Globals.cpp* – declaration and implementation of public functions used in the project.
- *main.cpp* – file containing function named `main`, which is the designated start of the program.

### 5. Dependencies

none

### 6. Class description

In the project the following classes were created:

- `TodoItem` – represents a todo item.

- `int getTodoItemId(void)` – returns the todo item id.
- `string getDescription(void)` – returns the todo item description.
- `bool isCompleted(void)` – returns the todo item status.
- `void setCompleted(bool completed)` – sets the todo item status.
- `void save(ofstream &file)` – writes todo item details to file.
- **Globals** – a class that contains public functions.
  - `list <TodoItem> readTodoItemsFromFile(const string &fileName)` – reads all todo items from file,
  - `void saveTodoItemsToFile(list <TodoItem> &todoItems, const string &fileName)` – saves the current program state,
  - `void printTodoItems(list <TodoItem> &todoItems)` – prints all todo items to standard output,
- **Utils** – a class that contains miscellaneous functions.
  - `void handleAddingNewTodo(list<TodoItem> &todoItems, ostream &error_message)` – handles interacting with user to add new todo item.
  - `void handleCompletingTodo(list<TodoItem> &todoItems, ostream &error_message)` – handles interacting with user to set todo item as completed.
  - `void handleDeletingTodo(list<TodoItem> &todoItems, ostream &error_message)` – handles interacting with user to delete todo item.

## 7. Resources

In the project the following resources are used:

- `todolist.txt` – a file with all todo items. Each line contains:
  - ID of todo item – positive integer,
  - description of todo item,
  - completion status – 0 or 1.

The data is comma separated.

## 8. Future development

Future development of this Todo List application could include the following features:

- Adding the ability to assign a due date to each TodoItem, and display a reminder when the due date approaches.
- Implementing a sorting function to sort TodoItems by due date, priority or alphabetically.
- Implementing a search function that allows users to search for TodoItems by keywords or tags.
- Adding the ability to collaborate on TodoItems with other users, either by sharing a list or by allowing multiple users to access and edit the same list.
- Implementing a user-friendly and customizable UI, to make the application more visually appealing and easier to use.

These are just a few examples of potential features that could be added to the application in the future, depending on user feedback and the development team's priorities. The goal is to make the application more useful and efficient for users by providing them with more ways to manage their tasks.

## **9. Other**

none