Task Manager

Ra’ed Alasmar

V2Y3X8

Introduction

In the modern world, efficient task management is crucial for both personal productivity and organizational efficiency. Personally, I like to start my morning with writing every single task I have for today or near deadlines that I should be aware of, this daily habit inspired me to implement a task management system in C++, my goal was to create a tool that assists me and other users to keep track of various important life aspects and achieving their goals.

The primary objective of this C++ Task Manager project is to provide a user-friendly platform that allows the users to add, delete, update, and display tasks. This system is designed to be easy and simple to use, ensuring that users can quickly adapt and integrate it into their daily routines. This documentation aims to provide a comprehensive overview of the Task Manager's functionality, including detailed description of its user interface, program execution, and the input/output mechanisms.

Program interface & execution

The Task Manager is a consol-based application developed in C++, users need to have a basic C++ development environment set up, such as Visual Studio for windows or any standard C++ compiler.

- ***Starting the program:***
  To start the Task Manager, users must compile the source code.

- ***Terminating the program:***
  the program can be terminated at any point by entering the appropriate menu option, which ensures that all the data is saved, and the application is closed.

The Task manager application offers a simple menu that allows the user to manage their tasks through a series of options. Here's how the program operates from the user's perspective:

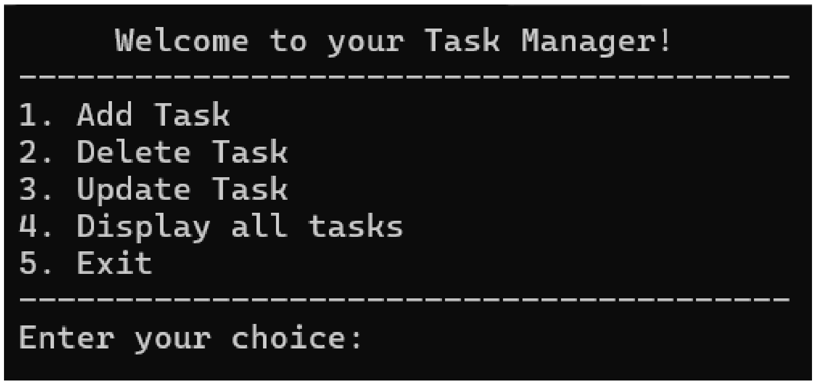
Main menu

Upon launching, the program will welcome the user and display the main menu with the following options:

1- Add task: allows the user to create a new task
2- Delete task: Enables the deletion of an existing task
3- Update task: allows the user to update the details of an existing task
4- Display all tasks: shows a list of all uncompleted tasks
5- Exit : closes the application

Users can select an option by entering the corresponding number.

Here is the visual representation upon executing the application:



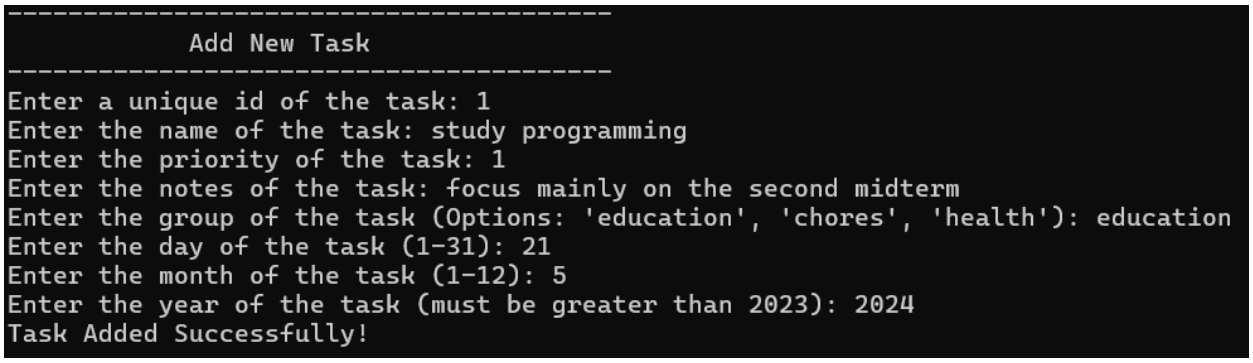
```
     Welcome to your Task Manager!
----------------------------------------
1. Add Task
2. Delete Task
3. Update Task
4. Display all tasks
5. Exit
----------------------------------------
Enter your choice:
```

Detailed Execution flow

- **Adding a Task:** Users are asked to input details about the task such as ID, name, priority, notes, group, and deadline. The system checks for the uniqueness of the ID before adding the task to ensure no duplicates.

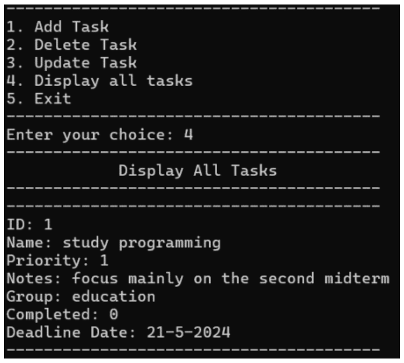
- **Deleting a Task:** Users provide the ID of the task they wish to delete. The system verifies its existence before proceeding with deletion.
- **Updating a Task:** Users enter the ID of the task to update and select the attribute to modify (e.g., name, priority, deadline). Each update involves specific validations relevant to the data type.
- **Displaying Tasks:** This option lists all tasks, showing details like name, priority, etc.
- **Exiting the Program:** Choosing this option terminates the application, with all changes saved to prevent data loss.

Here is an example of the output when the add task option is chosen:

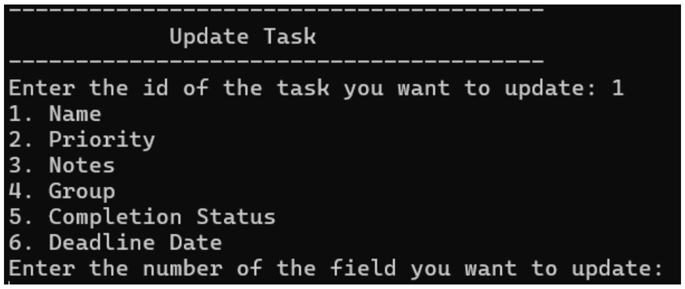


```
----------------------------------------
            Add New Task
----------------------------------------
Enter a unique id of the task: 1
Enter the name of the task: study programming
Enter the priority of the task: 1
Enter the notes of the task: focus mainly on the second midterm
Enter the group of the task (Options: 'education', 'chores', 'health'): education
Enter the day of the task (1-31): 21
Enter the month of the task (1-12): 5
Enter the year of the task (must be greater than 2023): 2024
Task Added Successfully!
```

And here is the output when the user decides to display this task or update it:



```
----------------------------------------
1. Add Task
2. Delete Task
3. Update Task
4. Display all tasks
5. Exit
----------------------------------------
Enter your choice: 4
----------------------------------------
            Display All Tasks
----------------------------------------
----------------------------------------
ID: 1
Name: study programming
Priority: 1
Notes: focus mainly on the second midterm
Group: education
Completed: 0
Deadline Date: 21-5-2024
----------------------------------------
```



```
----------------------------------------
            Update Task
----------------------------------------
Enter the id of the task you want to update: 1
1. Name
2. Priority
3. Notes
4. Group
5. Completion Status
6. Deadline Date
Enter the number of the field you want to update:
```

Input & output

User input:

The Task Manager application interacts with users primarily through the console, where they input data using the keyboard. The inputs required by the application include:

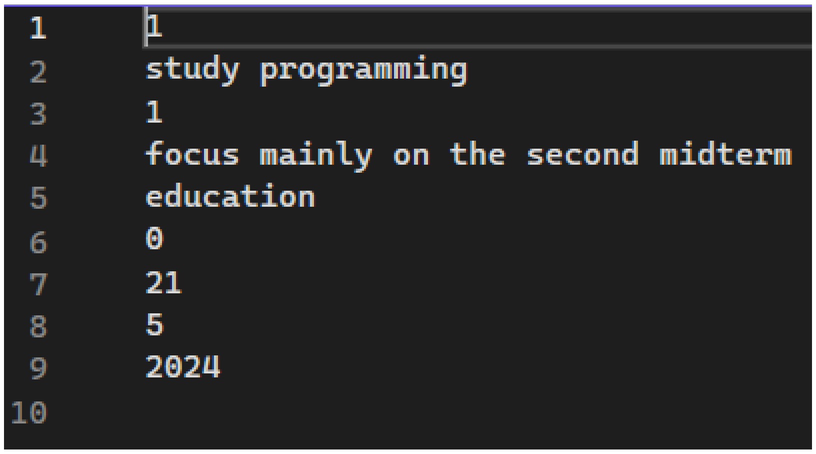
- **Task ID:** A unique integer to identify each task.
- **Name:** A string representing the task name.
- **Priority:** An integer indicating the task's priority.
- **Notes:** A string containing any additional details or notes about the task.
- **Group:** A string specifying the category of the task ('education', 'chores', 'health').
- **Completion status:** An integer where it represents if the task is complete or not, when creating the task, its automatically set to zero, and when updating the task and setting it to one, the program will congratulate the user for completing the task and delete it automatically.
- **Deadline:** Three integers representing the day, month, and year of the task's deadline.
- **Menu Selections:** Integers used to navigate the application's menu system.

Each input is validated for type and format. For example, the Task ID must be a unique positive integer, and the deadline must be a valid date after year 2023. If invalid inputs are detected, the application asks the user to re-enter the correct data before proceeding.

File input:

The application also read tasks from a file named ‘tasks.txt’, where each task is stored with its attributes on separate lines.

Using the same task that we specified earlier, here is the format of the ***tasks.txt:***



```
 1     1
 2     study programming
 3     1
 4     focus mainly on the second midterm
 5     education
 6     0
 7     21
 8     5
 9     2024
10
```

Where line 6 in here is the completion status.

Output specifications:

The application provides feedback and information through the console. Outputs include:

- **Task Details:** Displayed if the user asked for it.
- **Success Messages:** Confirmations of successful operations (e.g., "Task Added Successfully").
- **Error Messages:** Notifications of errors or invalid operations (e.g., "Invalid Input. Please try again.").
- **Menu Options:** The main menu and subsequent options are displayed for user navigation.

Outputs are designed to be clear and informative, ensuring that the user understands the result of their actions and the current state of their tasks.

File output:

When the application exits, it saves the current list of tasks back to ‘tasks.txt’ using the same format as the input file. This ensures that all the changes made are preserved.

Program structure

The Task Manager system is structured around two primary classes: ‘***Task’***, ‘***TaskGroup’***, and the Main program. These classes handle the logic for task manipulation and storage.

Task Class

Description:

The ***‘Task’*** class encapsulates all the properties of a task and provides methods for accessing and modifying these properties.

Attributes:

- int id - Unique identifier for the task.
- string name - Name or title of the task.
- int priority - Numerical priority level.
- string notes - Additional details or notes for the task.
- string group - Categorization of the task (Education, chores, Health).
- bool completed - Status of the task, whether it is completed or not.
- int day, month, year - Due date represented by day, month, and year.

Methods:

- Constructors for initialization.
- Getters and setters for all attributes to manage task data.
- increaseCount() - A static method to track the total number of tasks.

TaskGroup Class

Description:

The ‘TaskGroup’ class manages a collection of ‘Task’ objects.it uses dynamic memory allocation to store and manipulate tasks. This class is responsible for adding, deleting, updating, and displaying task.

Attributes:

- Task* data - Pointer to an array of Task objects.
- int totalNumberOfTasks - Keeps track of the current number of tasks.

Methods:

- Constructors and destructors to manage dynamic memory.

- ### addTask(Task task)

  #### Parameters:

  - “Task task” – the task object is added to the task collection.

  #### Functionality:

  - This method dynamically allocates a new array of tasks that is one element larger than the current array, copies existing tasks to the new array, adds the new task at the end, and finally deletes the old array to manage memory efficiently. It Also increments the totalNumberOfTasks.

- ### deleteTask(int id, bool updateMode)

  #### Parameters:

- int id - The unique identifier of the task to be deleted.
- bool updateMode – A flag that indicates if the task is deleted as part of an update operation (such as marking it as completed).

Functionality:

- This method first checks if a task with the specified ID exists. If it does, it creates a new array smaller by one element, copies over all tasks except the one to be deleted, and then deletes the old array. The updateMode indicates if the task is being deleted due to an update (marking as completed).

❖ printTasks(string filename)

Parameters:

- String filename – the name of the file where the tasks will be saved.

Functionality:

- Opens the specified file and write the details of each task In a structured format.

❖ updateTask(int id, int fieldNumber)

Parameters:

- Int id – the ID of the task to be updated.
- Int fieldNumber – the number indicating which attribute of the task needs to be updated, such as name, priority, etc.

Functionality:

- Finds the task by its ID and then updates the specific attribute based on the 'FieldNumber'. Then, the user will be input the new value.

❖ isTask(int id)

Parameters:

- Int id – the ID of the task to check if its within the collection of tasks.

Functionality:

- Iterates through the array of tasks to check if any task has the given ID. Returns true if the task exists and false if it does not exist. This method is crucial for validating user inputs.

❖ displayAllTasks()

Functionality:

- Iterates through the entire list of tasks and print each one's details onto the consol.

Main program:

Description:

The main driver of the program, main.cpp, utilizes the Task and TaskGroup classes to provide a menu interface for the task management system. It handles user inputs, displays outputs, and manages the flow of the program.

Functionality:

- Load tasks from a file upon startup.
- Provide a menu with options to add, delete, update, and display tasks.
- Handle user inputs and validate them.
- Save tasks back to a file upon exiting the program

Conclusion

The task Manager project taught me new skills and developed the skills that I have already learned in the “Basics of Programming 2” course, deepening my understanding of C++ and object-oriented programming. This experience motivated me to build new projects in the near future and develop my skills even more in this area.