- **Table of Content**

Abstract

Introducing a user-friendly To-Do List Application developed in C++. This application provides a hassle-free way for users to manage tasks, offering features to add, edit, and remove tasks effortlessly. Designed with simplicity in mind, the application serves as an excellent learning tool for beginners, demonstrating fundamental C++ concepts such as classes, arrays, and user input/output operations. Through its intuitive interface, this application enhances productivity by enabling users to organize tasks effectively, ensuring a streamlined approach to task management. This project not only aids in learning programming basics but also emphasizes the practical importance of efficient task handling in daily life.

- **Introduction**

Welcome to our project, in this project we create a to do list using OOP in C++

We use many features of OOP in this program like:

- Classes
- Objects
- Encapsulation.
- Abstraction.

We will be using loops, class, if statements and user defined functions in this program.

Our aim after making this program is to learn and practice more about classes and become faster and better at programming.

- **Hardware and software requirement**

Hardware:

- A computer or device capable of running C++ programs.
- Processor: intel i5
- Operating system: Windows 10

Software:

Turbo C++

- **Algorithm**

1. Define Task Class:
   - Create a class ‘Task’ with a public character array `description`.

2. Define ToDoList Class:
   - Create a class ‘ToDoList’.
   - Declare a private array of ‘Task’ objects named `tasks` with a size of `MAX_TASKS`.
   - Declare an integer variable ‘numTasks’ and initialize it to 0.

3. Define Methods in ToDoList Class:
   - Constructor ToDoList():
     - Initialize ‘numTasks’ to 0.

   - addTask() Method:
     - If ‘numTasks’ is less than ‘MAX_TASKS’:
       - Prompt user for task description.
       - Store description in the next available slot in the `tasks` array.
       - Increment ‘numTasks’.
     - If ‘numTasks’ equals `MAX_TASKS`, display "Task list is full!".

   - editTask() Method:
     - Prompt user for task number to edit.
     - If the input task number is valid (between 1 and ‘numTasks’):
       - Prompt user for a new task description and update the task.
     - If the input task number is invalid, display "Invalid task number!".

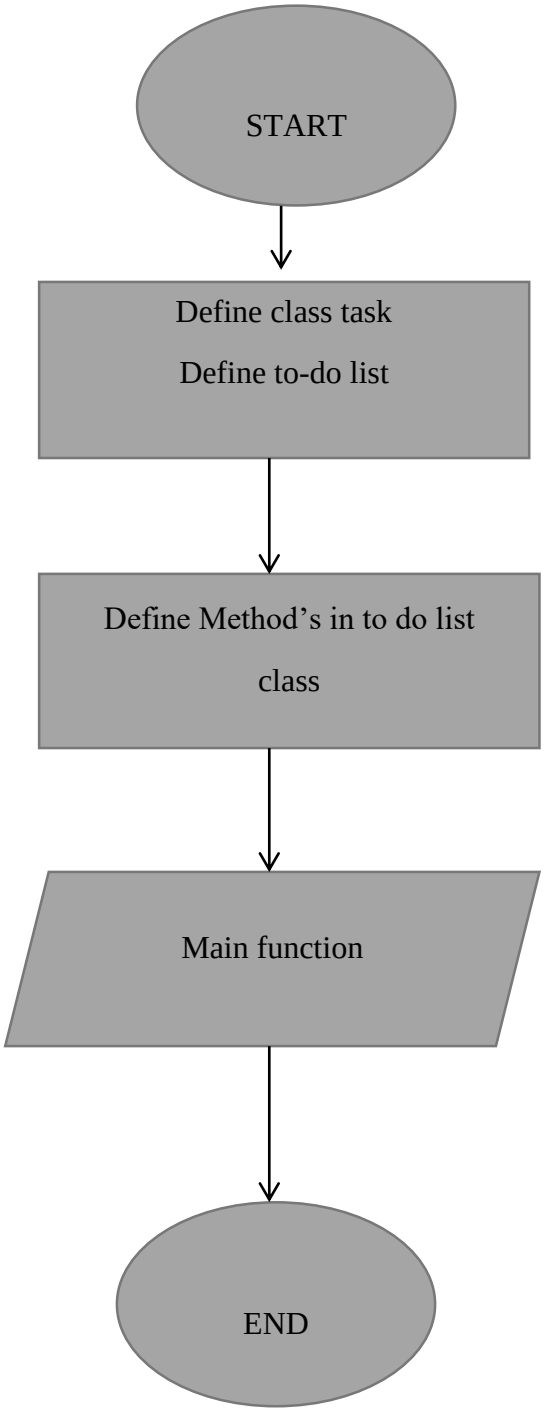
   - removeTask() Method:
     - Prompt user for task number to remove.
     - If the input task number is valid (between 1 and ‘numTasks’):
       - Shift tasks to remove the task at the specified index.
       - Decrement ‘numTasks’.
     - If the input task number is invalid, display "Invalid task number!".

   - listTasks() Method:
     - Display tasks with their index and description.

4. Main Function:
   - Create an instance of `ToDoList` named ‘todo’.
   - Enter a loop to display a menu and accept user choice:
     - Call ‘addTask()’ for choice 1.
     - Call ‘editTask()’ for choice 2.
     - Call ‘removeTask()’ for choice 3.
     - Call ‘listTasks()’ for choice 4.
     - Exit the loop and end the program for choice 5.

5. End of Algorithm.

- **Flowchart**


START
Define class task
Define to-do list
Define Method’s in to do list
class
Main function
END

- Source Code

Code:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

const int MAX_TASKS = 10;
int i;
class Task {
public:
   char description[100];
};

class ToDoList {
private:
   Task tasks[MAX_TASKS];
   int numTasks;

public:
   ToDoList() {
         numTasks = 0;
   }

   void addTask() {
      if (numTasks < MAX_TASKS) {
         Task newTask;
         cout << "Enter task description: ";
            cin.ignore();
            cin.getline(newTask.description, 100);
            tasks[numTasks] = newTask;
            numTasks++;
            cout << "Task added!\n";
         } else {
            cout << "Task list is full!\n";
         }
   }

   void editTask() {
         cout << "Enter task number to edit (1-" << numTasks << "): ";
         int taskNum;
         cin >> taskNum;
```

```
        if (taskNum >= 1 && taskNum <= numTasks) {
            cout << "Enter new task description: ";
            cin.ignore();
            cin.getline(tasks[taskNum - 1].description, 100);
            cout << "Task updated!\n";
        } else {
            cout << "Invalid task number!\n";
        }
    }

    void removeTask() {
        int taskNum;
        cout << "Enter task number to remove (1-" << numTasks << "): ";
        cin >> taskNum;

        if (taskNum >= 1 && taskNum <= numTasks) {
            for (i = taskNum - 1; i < numTasks - 1; ++i) {
                tasks[i] = tasks[i + 1];
            }
            numTasks--;
            cout << "Task removed!\n";
        } else {
            cout << "Invalid task number!\n";
        }
    }

    void listTasks() {
        cout << "Tasks:\n";
        for (i = 0; i < numTasks; ++i) {
            cout << i + 1 << ". ";
            cout << "[X] ";
            cout << tasks[i].description << "\n";
        }
    }
};

int main() {
    ToDoList todo;

    while (1) {
        clrscr();

        cout << "To-Do List Application\n";
        cout << "1. Add Task\n";
        cout << "2. Edit Task\n";
        cout << "3. Remove Task\n";
```

```
        cout << "4. List Tasks\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";

        int choice;
        cin >> choice;

        switch (choice) {
            case 1:
                todo.addTask();
                break;
            case 2:
                todo.editTask();
                break;
            case 3:
                todo.removeTask();
                break;
            case 4:
                todo.listTasks();
                break;
            case 5:
                cout << "Exiting...\n";
                return 0;
            default:
                cout << "Invalid choice! Please enter a valid option.\n";
        }

        cout << "\nPress enter to continue...";
        getch();
    }
    getch();
    return 0;
}
```

- **Output**

```
To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 1
Enter task description: Prepare for upcoming exams and quizzes.
Task added!

Press enter to continue..._
```

```
To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 1
Enter task description: Prepare for upcoming exams and quizzes.
Task added!

Press enter to continue..._
```

```
To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 1
Enter task description: Explore internship or job opportunities.
Task added!

Press enter to continue..._
```

```
To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 4
Tasks:
1. [X] Prepare for upcoming exams and quizzes.
2. [X] Explore internship or job opportunities.

Press enter to continue..._

To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 2
Enter task number to edit (1-2): 1
Enter new task description: Prepare for class test.
Task updated!

Press enter to continue..._
```

```
To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 3
Enter task number to remove (1-2): 1
Task removed!

Press enter to continue...

To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 4
Tasks:
1. [X] Explore internship or job opportunities.

Press enter to continue...


To-Do List Application
1. Add Task
2. Edit Task
3. Remove Task
4. List Tasks
5. Exit
Enter your choice: 4
Tasks:
1. [X] Explore internship or job opportunities.

Press enter to continue..._
```

- **Application**

  Task Management: The primary purpose of a to-do list app is to manage tasks. Users can create, organize, prioritize, and schedule tasks efficiently.

  Time Management: To-do list apps often come with features like deadlines, reminders, and calendars, helping users manage their time effectively and meet deadlines.

  Project Management: To-do lists can be used for managing projects, breaking them down into smaller tasks, setting deadlines, and tracking progress.

  Goal Setting: People use to-do lists to set and achieve personal and professional goals. Breaking down larger goals into smaller tasks makes them more manageable and achievable.

6. Conclusion

We created the to-do list application using basic features of C++ like classes and objects.

Reference:

www.youtube.com
https://chat.openai.com

Details of Activity	Name of responsible Team member
Code and Output	Arqam
Presentation	Zain
Research	Aman
Program research	Mudassir
Help with report	Arqam