**Project Overview of,**

**To-Do List Manager**

By, Didhiti Raj Chakraborty

**Project Description**

This project lets users manage a to-do list. It allows:

1. Adding new tasks.
2. Viewing all tasks.
3. Marking tasks as done.
4. Deleting tasks.

We’ll use **basic arrays** to store tasks, loops for user interaction, and functions to organize the code. This project gives hands-on experience in **loops, conditionals, functions, and basic data storage**.

**Program Code:**

#include <iostream>

#include <string>

using namespace std;

// Constants

const int MAX\_TASKS = 100; // Maximum number of tasks allowed

// Function prototypes

void showMenu();

void addTask(string tasks[], bool done[], int &taskCount);

void viewTasks(const string tasks[], const bool done[], int taskCount);

void markTaskDone(bool done[], int taskCount);

void deleteTask(string tasks[], bool done[], int &taskCount);

int main() {

string tasks[MAX\_TASKS]; // Array to store task descriptions

bool done[MAX\_TASKS] = {false}; // Array to track if a task is done

int taskCount = 0; // Number of tasks added

int choice; // User menu choice

do {

showMenu(); // Display menu options

cout << "Enter your choice: ";

cin >> choice;

// Perform actions based on user's choice

switch (choice) {

case 1: addTask(tasks, done, taskCount); break;

case 2: viewTasks(tasks, done, taskCount); break;

case 3: markTaskDone(done, taskCount); break;

case 4: deleteTask(tasks, done, taskCount); break;

case 5: cout << "Goodbye! Exiting program...\n"; break;

default: cout << "Invalid choice! Please try again.\n";

}

} while (choice != 5);

return 0;

}

// Function to display the menu

void showMenu() {

cout << "\n===== To-Do List Manager =====\n";

cout << "1. Add a new task\n";

cout << "2. View all tasks\n";

cout << "3. Mark a task as done\n";

cout << "4. Delete a task\n";

cout << "5. Exit\n";

}

// Function to add a new task

void addTask(string tasks[], bool done[], int &taskCount) {

if (taskCount >= MAX\_TASKS) {

cout << "Task list is full! Cannot add more tasks.\n";

return;

}

cout << "Enter the task description: ";

cin.ignore(); // Clear buffer

getline(cin, tasks[taskCount]); // Input full task description

done[taskCount] = false; // Mark task as not done

taskCount++;

cout << "Task added successfully!\n";

}

// Function to view all tasks

void viewTasks(const string tasks[], const bool done[], int taskCount) {

if (taskCount == 0) {

cout << "No tasks to display.\n";

return;

}

cout << "\nYour Tasks:\n";

for (int i = 0; i < taskCount; ++i) {

cout << i + 1 << ". " << tasks[i] << " [" << (done[i] ? "DONE" : "NOT DONE") << "]\n";

}

}

// Function to mark a task as done

void markTaskDone(bool done[], int taskCount) {

if (taskCount == 0) {

cout << "No tasks to mark as done.\n";

return;

}

int taskNumber;

cout << "Enter the task number to mark as done: ";

cin >> taskNumber;

if (taskNumber >= 1 && taskNumber <= taskCount) {

done[taskNumber - 1] = true;

cout << "Task marked as done!\n";

} else {

cout << "Invalid task number!\n";

}

}

// Function to delete a task

void deleteTask(string tasks[], bool done[], int &taskCount) {

if (taskCount == 0) {

cout << "No tasks to delete.\n";

return;

}

int taskNumber;

cout << "Enter the task number to delete: ";

cin >> taskNumber;

if (taskNumber >= 1 && taskNumber <= taskCount) {

// Shift tasks to fill the gap

for (int i = taskNumber - 1; i < taskCount - 1; ++i) {

tasks[i] = tasks[i + 1];

done[i] = done[i + 1];

}

taskCount--;

cout << "Task deleted successfully!\n";

} else {

cout << "Invalid task number!\n";

}

}

**Step-by-Step Explanation**

1. Includes and Setup:

#include <iostream>

#include <string>

using namespace std;

* Includes iostream for input/output and string for handling task descriptions.

1. Global Constants:

const int MAX\_TASKS = 100;

* Limits the number of tasks.

1. Main program:

* A menu is displayed repeatedly using a do-while loop.
* Based on the user’s input, the program calls functions to perform specific actions.

1. Functions:

* **showMenu()**: Displays the list of options.
* **addTask()**: Adds a task to the list and marks it as **not done**.
* **viewTasks()**: Displays all tasks with their status.
* **markTaskDone()**: Marks a selected task as **done**.
* **deleteTask()**: Deletes a selected task by shifting the remaining tasks.

## How to Use the Program

1. **Run the Program**: Compile and run the code in a C++ IDE.
2. **Choose an Option**:
   * Press 1 to add a task.
   * Press 2 to view all tasks.
   * Press 3 to mark a task as done.
   * Press 4 to delete a task.
   * Press 5 to exit.