**PLANNER**

Data Structures and Algorithms

**Planner**

**Introduction:**

In DSA course, we created Planner as a semester project using doubly linked list. Classes, loops(for, do-while, while) , header files, switch statements, if-else statements, data types(string, char, int), main menu function, constructor(parametrized and non-parametrized) , destructor, Insertion at the start, insertion at the end, insertion by sorting, edit , deletion at the , display , save to the file and load from file functions has been used.

**Characteristics of planner:**

In this console application-based planner user can choose from the main menu and then add plan. It includes title of plan, priority of plan, time required to complete the plan and any other information that is important for user to remember or related to that plan can be written in this option. Similarly in edit plan user will be asked what they want to change either title, priority, time required or additional information. Then according to user need user can modify it. In delete user can delete any of the plan he has entered by plan’s name. In display function user can see all the plans he has entered. All the plans added by the user will be stored in file and user can view file in backend.

**Code:**

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

class task {

public:

string title, add;

char priority;

int time;

task\* next, \* prev;

task() : title("xx"), priority('x'), time(0), add("xx"), next(NULL), prev(NULL) {}

task(string t, char p, int tm, string a) : title(t), priority(p), time(tm), add(a), next(NULL), prev(NULL) {}

};

class plan {

task\* head;

void delete\_tasks() {

task\* current = head;

while (current) {

task\* next = current->next;

delete current;

current = next;

}

head = NULL;

}

public:

plan() : head(NULL) {}

bool is\_empty() {

return head == NULL;

}

void start(string t, char p, int tm, string a)

{

task\* newNode = new task(t, p, tm, a);

if (!is\_empty())

newNode->next = head;

head = newNode;

}

void end(string t, char p, int tm, string a)

{

task\* newNode = new task(t, p, tm, a);

if (is\_empty())

head = newNode;

else {

task\* current = head;

while (current->next != NULL)

{

current = current->next;

}

newNode->prev = current;

current->next = newNode;

}

}

//////////////\*\*\*\*ADD\_TASK\*\*\*\*\*\*\*\*/////////////////////////

void add\_task(string t, char p, int tm, string a)

{

task\* newNode = new task(t, p, tm, a);

if (is\_empty())

{

head = newNode;

}

else if (p == 'H')

{

start(t, p, tm, a);

}

else if (p == 'L')

{

end(t, p, tm, a);

}

else

{

task\* current = head;

while (current->next && current->next->priority == 'H')

{

current = current->next;

}

newNode->next = current->next;

newNode->prev = current;

if (current->next)

{

current->next->prev = newNode;

}

current->next = newNode;

}

}

//////////////\*\*\*\*DELETE TASK\*\*\*\*\*\*\*\*/////////////////////////

void delete\_task(string t) {

task\* current = head;

while (current && current->title != t) {

current = current->next;

}

if (current) {

if (current->prev == NULL)

head = current->next;

else if (current->next == NULL)

current->prev->next = NULL;

else {

current->prev->next = current->next;

current->next->prev = current->prev;

}

current->next = NULL;

current->prev = NULL;

delete current;

}

}

//////////////\*\*\*\*EDIT TASK\*\*\*\*\*\*\*\*/////////////////////////

void edit\_task(string t) {

task\* current = head;

char choice, cont;

string updatedt;

char updatedp;

int updatedtm;

while (current && current->title != t) {

current = current->next;

}

do {

cout << "Select what do you wish to change" << endl

<< "1. Title" << endl

<< "2. Priority" << endl

<< "3. Time" << endl

<< "4. Additional Information" << endl;

cout << "Choice please";

cin >> choice;

switch (choice)

{

case '1':

cout << "Enter the new title: ";

cin >> updatedt;

current->title = updatedt;

break;

case '2':

cout << "Enter the new priority: ";

cin >> updatedp;

current->priority = updatedp;

break;

case '3':

cout << "Enter the new required time: ";

cin >> updatedtm;

current->time = updatedtm;

break;

case '4':

cout << "Enter the new information: ";

cin >> updatedt;

current->add = updatedt;

break;

}

cout << "Any other changes? Y/N" << endl;

cin >> cont;

} while ((cont == 'Y') || (cont == 'y'));

}

//////////////\*\*\*\*DISPLAY TASK\*\*\*\*\*\*\*\*/////////////////////////

void display() {

task\* current = head;

int i = 1;

if (!is\_empty()) {

while (current) {

cout << "TASK 0" << i << endl;

cout << "Title: " << current->title << endl

<< "Priority: " << current->priority << endl

<< "Time Required: " << current->time << endl

<< "Additional Information: " << current->add << endl;

cout << "--------------------------------------------" << endl;

i++;

current = current->next;

}

}

else

cout << "NO TASKS" << endl;

}

//////////////\*\*\*\*SAVE TASK TO FILE\*\*\*\*\*\*\*\*/////////////////////////

void save\_to\_file(const string& filename)

{

ofstream file(filename);

if (!file.is\_open())

{

cerr << "Error opening file: " << filename << endl;

return;

}

task\* current = head;

while (current)

{

file << current->title << " " << current->priority << " "

<< current->time << " " << current->add << endl;

current = current->next;

}

file.close();

}

//////////////\*\*\*\*LOAD TASK FROM FILE \*\*\*\*\*\*\*\*/////////////////////////

void load\_from\_file(const string& filename)

{

ifstream file(filename);

if (!file.is\_open())

{

cerr << "Error opening file: " << filename << endl;

return;

}

delete\_tasks();

string t, a;

char p;

int tm;

while (file >> t >> p >> tm >> a)

{

add\_task(t, p, tm, a);

}

file.close();

}

//////////////\*\*\*\*PRNT MAIN MENU TASK \*\*\*\*\*\*\*\*/////////////////////////

void printMainMenu()

{

cout << endl << endl;

cout << "\t\t\t\t\t\*\*\*\*\*Main Menu:\*\*\*\*\*\*" << endl<<endl<<endl;

cout << "\t\t\t\t\t1. Add Plan" << endl;

cout << "\t\t\t\t\t2. Edit Plan" << endl;

cout << "\t\t\t\t\t3. Delete Plan" << endl;

cout << "\t\t\t\t\t4. Display Plan" << endl;

cout << "\t\t\t\t\t5. Exit" << endl;

cout << "\t\t\t\t\tChoose an option (1-5): ";

}

//////////////\*\*\*\*DESTRUNCTOR \*\*\*\*\*\*\*\*/////////////////////////

~plan() {

delete\_tasks();

}

};

int main()

{

plan p;

int option;

string taskTitle;

do

{

p. printMainMenu();

cin >> option;

switch (option)

{

case 1:

{

string title, add;

char priority;

int time;

cout << "Enter task details:" << endl;

cout << "Title: ";

cin >> title;

cout << "Priority (H/M/L): ";

cin >> priority;

cout << "Time required: ";

cin >> time;

cout << "Additional Information: ";

cin >> add;

p.add\_task(title, priority, time, add);

p.save\_to\_file("task\_plan.txt");

cout << "Task added successfully!" << endl;

break;

}

case 2:

cout << "Enter the title of the task to edit: ";

cin >> taskTitle;

p.edit\_task(taskTitle);

p.save\_to\_file("task\_plan.txt");

cout << "Task edited successfully!" << endl;

break;

case 3:

cout << "Enter title of task to delete: ";

cin >> taskTitle;

p.delete\_task(taskTitle);

p.save\_to\_file("task\_plan.txt");

cout << "Task deleted successfully!" << endl;

break;

case 4:

cout << "Current Plan:" << endl;

p.display();

cout << "Tasks saved in the file:" << endl;

p.load\_from\_file("task\_plan.txt");

p.display();

break;

case 5:

cout << "Exiting From Planner!!!" << endl;

break;

default:

cout << "Invalid option. Please choose a valid option (1-5)." << endl;

}

} while (option != 5);

return 0;

}