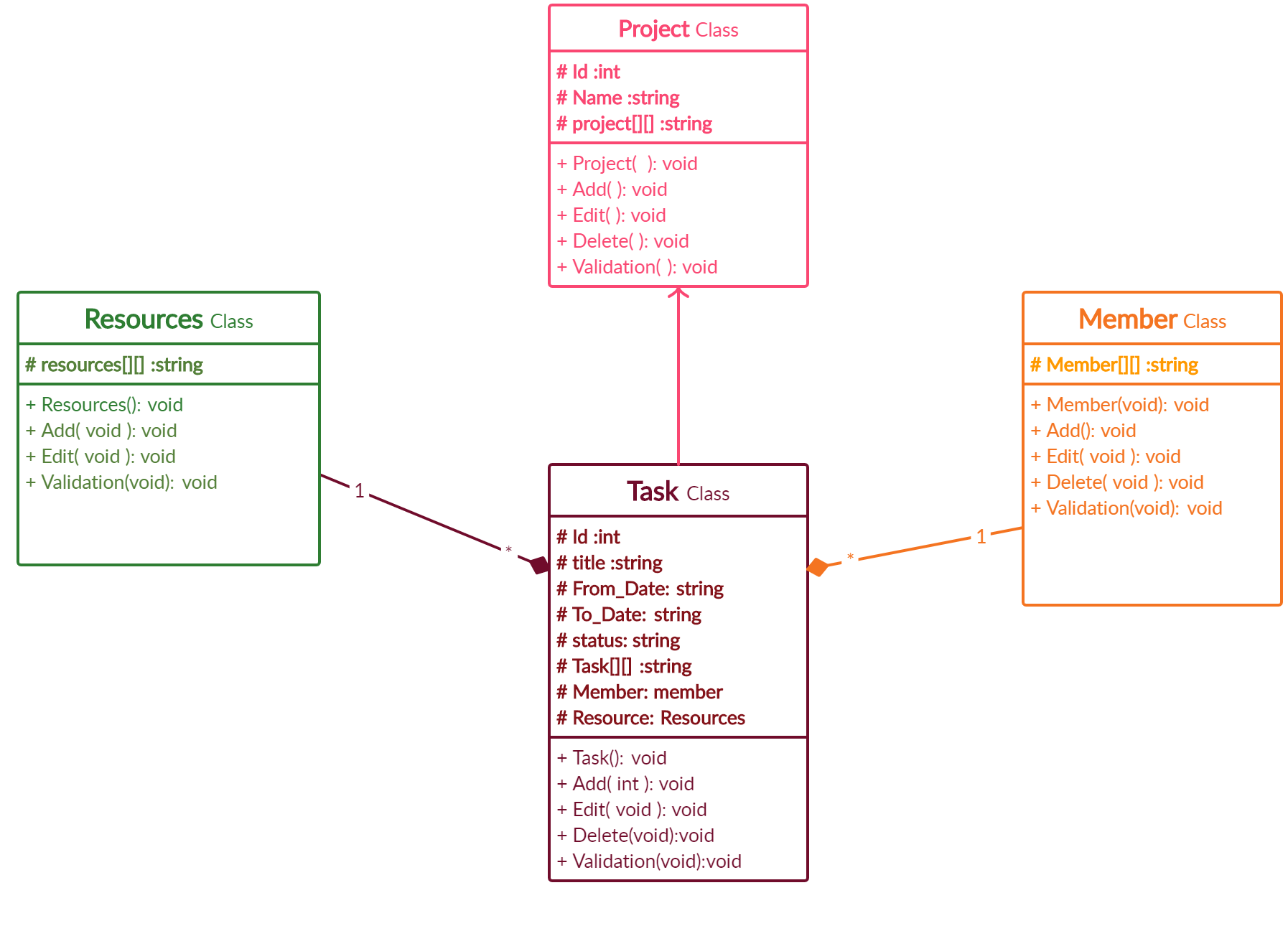


**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

* **2019-cs-5**
* **2019-cs-7**
* **2019-cs-10**
* **2019-cs11**

|  |  |
| --- | --- |
| **PROJECT MANAGEMENT SYSTEM** | object oriented programming project  **Team Members** |

**UML DIAGRAM**



**GUI**

****

USER NAME

**User1**

**User2**

**Password**

**Admin**

# **Description of the Methods:**

**The project is having the following basic member functions:**

* **Constructors:**

Every class is having *default constructor and parameterized* to initialize the data, so that compiler may not assign it the garbage value.

* **Add:**

This function allows the user to add in the field i.e. add the project, resources, tasks and members**.**

* **Edit:**

This function allows the user to edit in the field i.e. edit the project, resources, tasks and members.

* **Delete:**

This function allows the user to delete the field i.e. delete the project, resources, tasks and members**.**

* **Validation:**

This function assures that the user has entered correct values in the required fields or not.

* If yes, the program will continue
* If not, it will prompt the user to enter the correct value**.**
* **Print status:**

This function allows the user to print the delayed status.

## **User Manual:**

Dear user, as soon as you enter the program, you are required to make a selection (i.e. to tell us what exactly you want to do)

The program will first welcome you with the following menu:

1. Adding activities
2. Editing activities
3. Deleting activities
4. Status
5. Exit
6. **Adding Activities:**

This will allow you add:

1. Project
2. Tasks for a specified project
3. Members and Resources for a specified task
4. **Editing Activities**

This will allow you edit:

1. Project
2. Tasks for a specified project
3. Members and Resources for a specified task
4. **Deleting Activities**

This will allow you delete:

1. Project
2. Tasks for a specified project
3. Members and Resources for a specified task
4. **Status**

This will allow you to get the *delayed* tasks

1. **Exit**

You can exit by selecting this option.

## **Team Members Contribution:**

# Roll-No-05:

**Marriyam Nadeem** has worked on **task class, main function, GUI, and the part of designed document .** Moreover, has helped and guided other team members in their work.

# Roll-No-07:

**Mehwish Iqbal** has worked on **Resources class and the documentation part .**

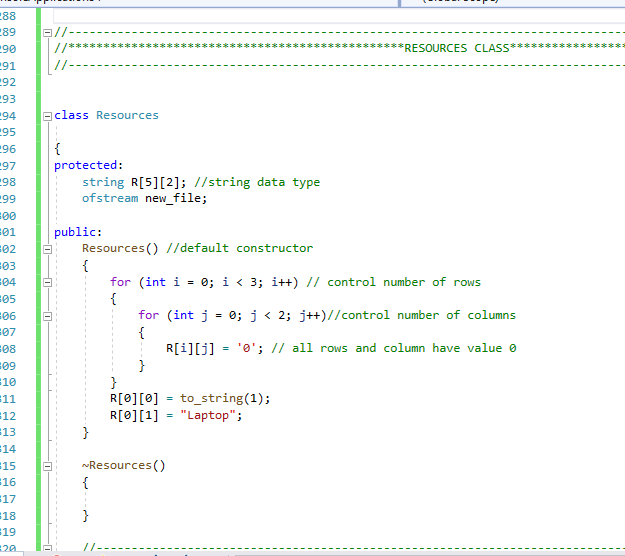
# **Roll-No-10:**

**Laraib Nawaz** has worked on **Project class and the documentation part.**

# Roll-No-11:

**Rabbiya Shoaib**has worked on **member class and GUI.** Moreover, has helped and guided other team members in their work.

**RESOURCES**

****

* **Header files used in this part of project**

# include<iostream>

# include<iomanip> //for setw

# include<string> // for string data

# include<fstream> // for fstream(input from file) and ofstream(output to file)(for file handling)

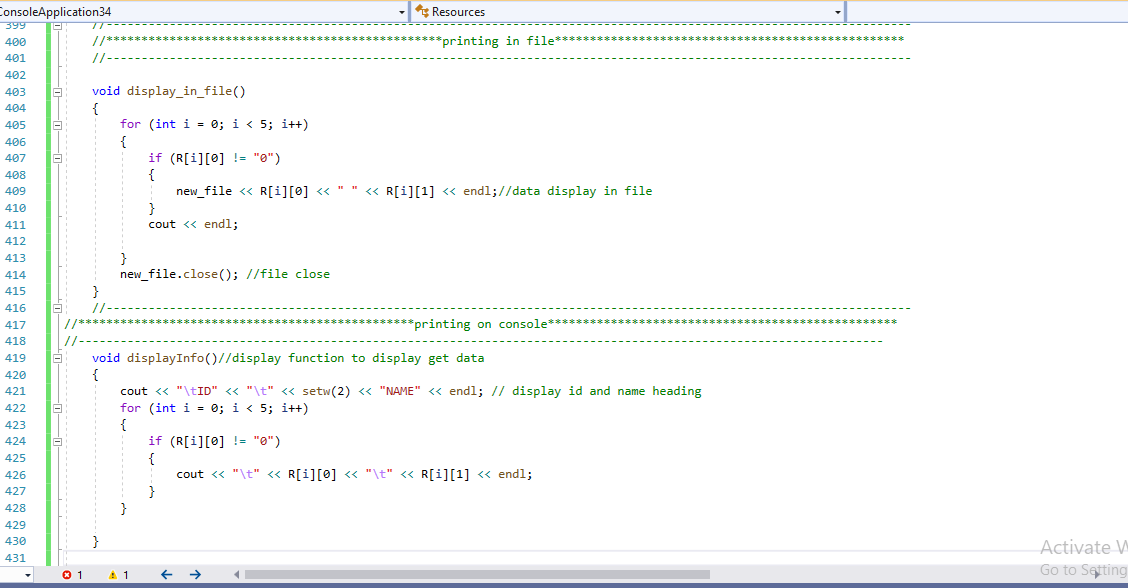
**CLASS NAME:** RESOURCES

**CONSTRUCTOR :** is use to initialize the array value by “0”

So no grabge value is come out .

**[0][0]=to\_string(i+1):**

Is use change the value of **Id** automatically or numerically change the changing data to constant



**getInfo input function to get resources input**

void getInput()

**{**

new\_file.open("resources.txt");//file open

for(int i=0;i<2;i++)// loop for rows

{

cout<<"RESOURCE: "<<i+1<<endl;

cout<<"Enter the Resource ID:"<<endl;

cin>>R[i][0];

cout<<"Enter the Resource Name:"<<endl;

cin.ignore(); //ignore the buffer values

getline(cin,R[i][1]);

**}**

**}**

Throught file handling input function data goes into file name resources.txt

2D array is decleare so first index get id and second index get name

Mean like table first row and column having a laptop always.

Input function show a message resources(i+1)//count number of resources

Then get resources id throught getline function.

**displayInfo function to display output**

void displayInfo()

{

cout<<"ID"<<" "<<setw(2)<<"NAME"<<endl; // display id and name heading

for(int i=0;i<2;i++)

{

cout<<R[i][0]<<" "<<R[i][1]<<endl;

}

for(int i=0;i<2;i++)

{

new\_file<<R[i][0]<<" "<<R[i][1]<<endl;//data display in file

}

new\_file.close(); //file close

}

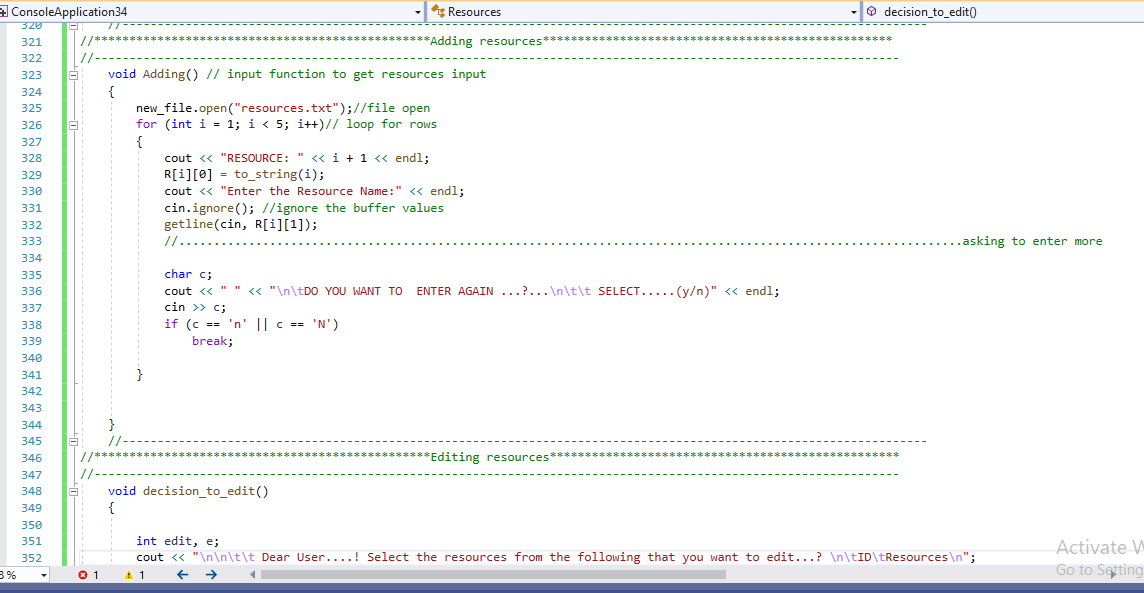
}

**Output is like that according to code**

|  |  |
| --- | --- |
| **Id** | **Name** |
| 0 | Laptop |
| 1 | Anything |

**Display\_infile :**  display data in file

**New\_file** close after getting or displaying data

**Void decision()**

This function provide dicision to the user about to add,editand delet data by calling their function

**Menu display in front of user to**

cout<<"Enter 1 to Add Resources"<<endl;

cout<<"Enter 2 to EDIT Resources"<<endl;

cout<<"Enter 3 to DELETE" <<endl;

**ADD FUNCTION:**

Get id from user which is to be add then add the data of that id the ask from use to add again if user press n then program return to menu option.

**EDIT FUNCTION:**

Get id from user which is to be edit then edit the data of that id the ask from use to edit again if user press n then program return to menu option.

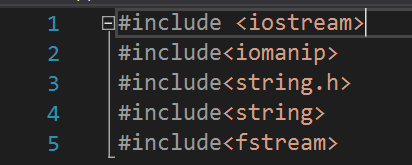
E=edit-1;

Use to edit the data which user want data from the memory sequence;

**DELETE FUNCTION:**

Get id from user which is to be delete then delete the data of that id the ask from use to delete again if user press n then program return to menu option.

**MEMBER**

1.Header files

Iomanip---🡪for setw

String.h----🡪for isalpha,isdisgit

String ---🡪for string data

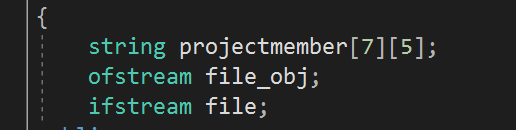
**Fstream** --🡪for file handling

2.Class teammember

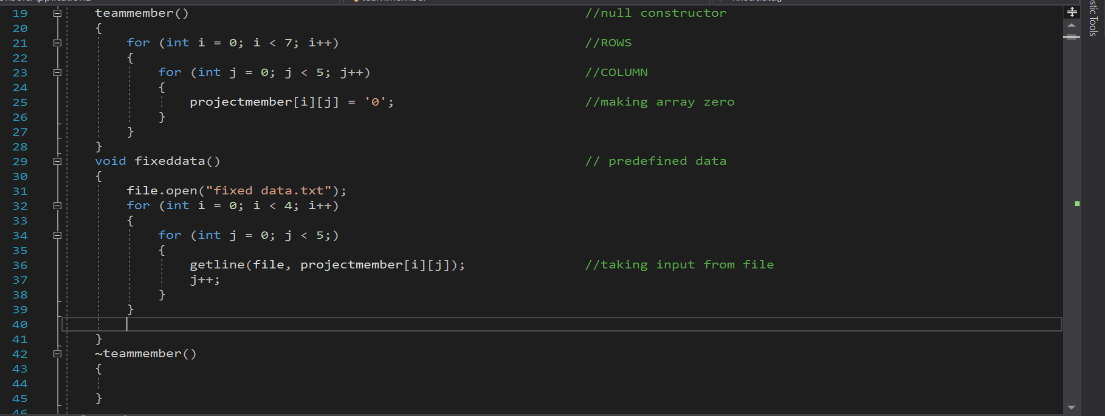
* **Initialization:**

**projectmember**[7][5]---🡪string array

**Ifstream** --🡪input from file

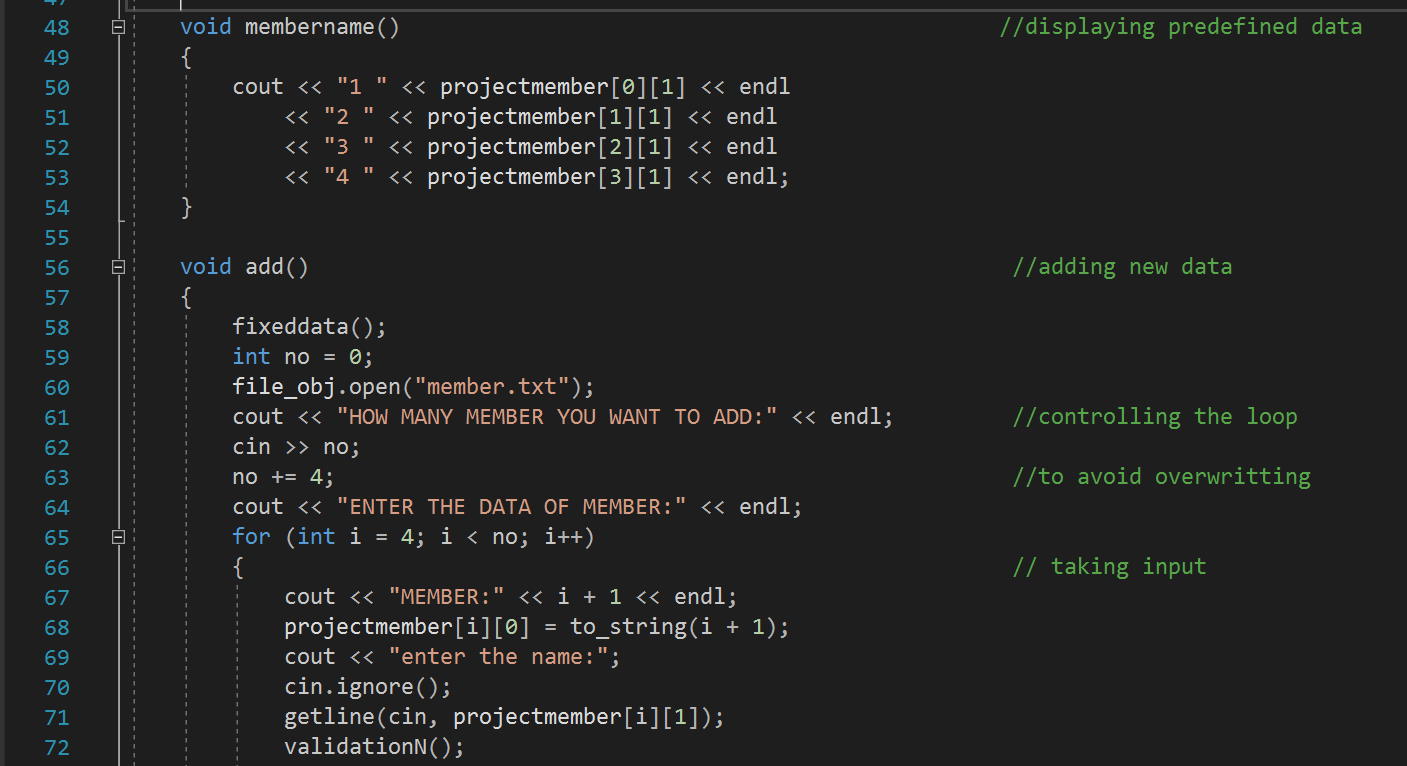
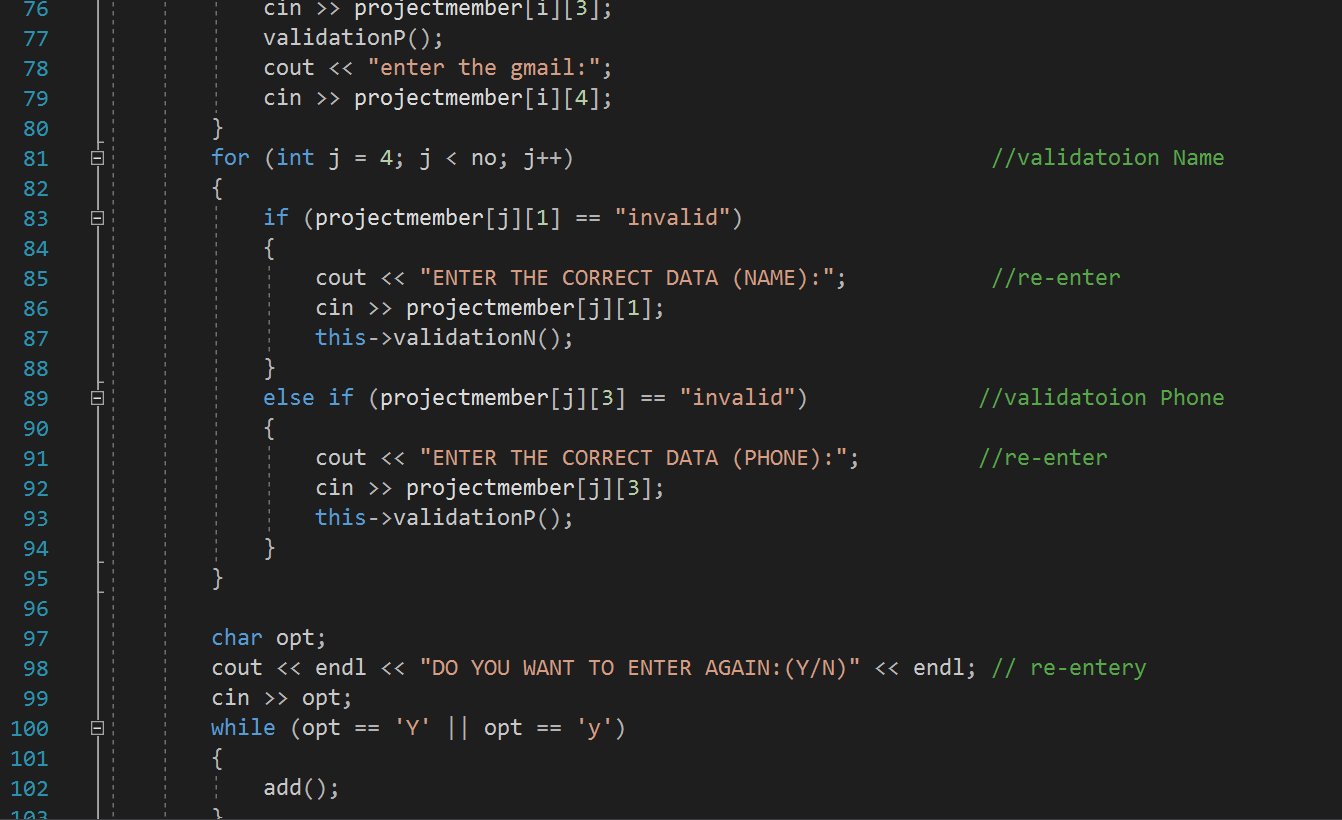
 **Ofstream** -🡪output to a file

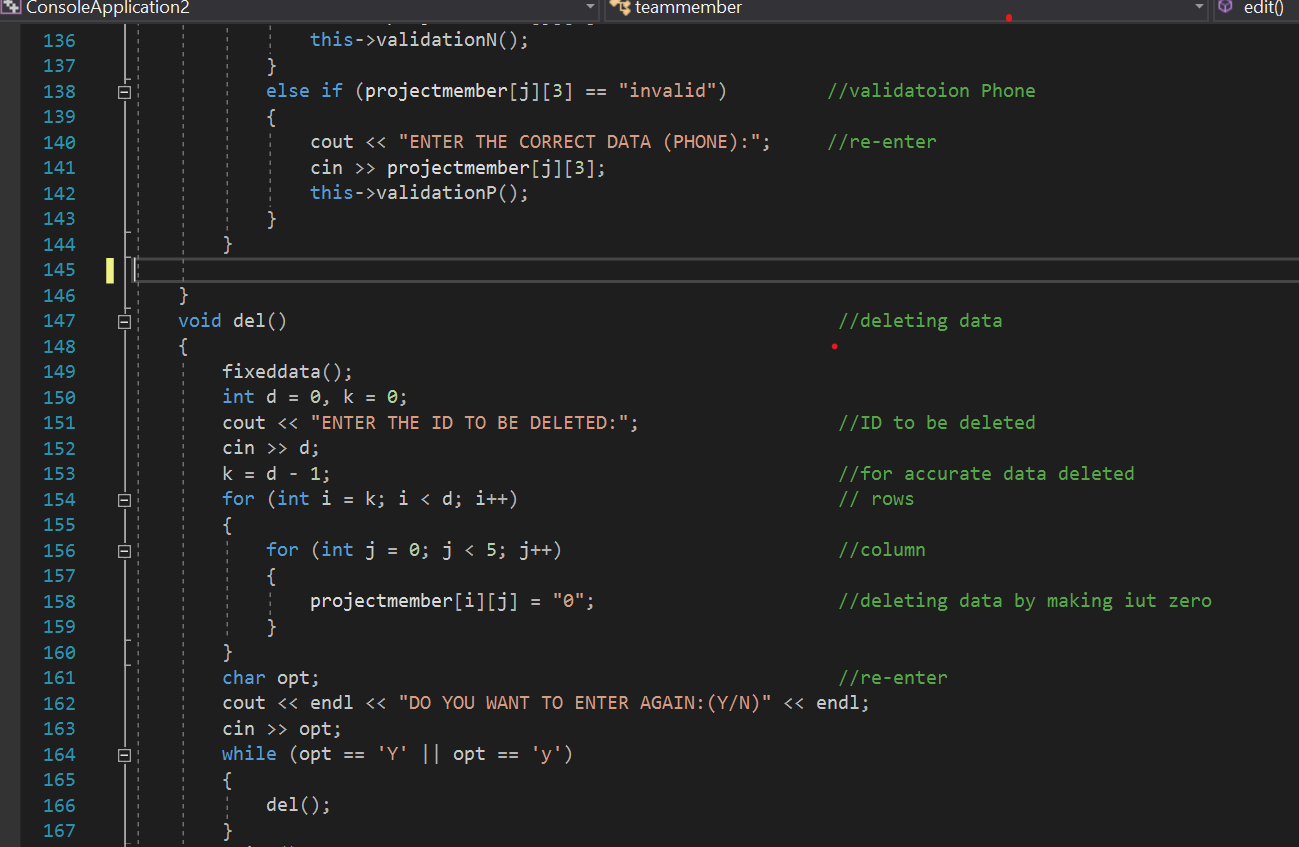
**teammember**  🡪 null constructorFixeddata 🡪taking data from file

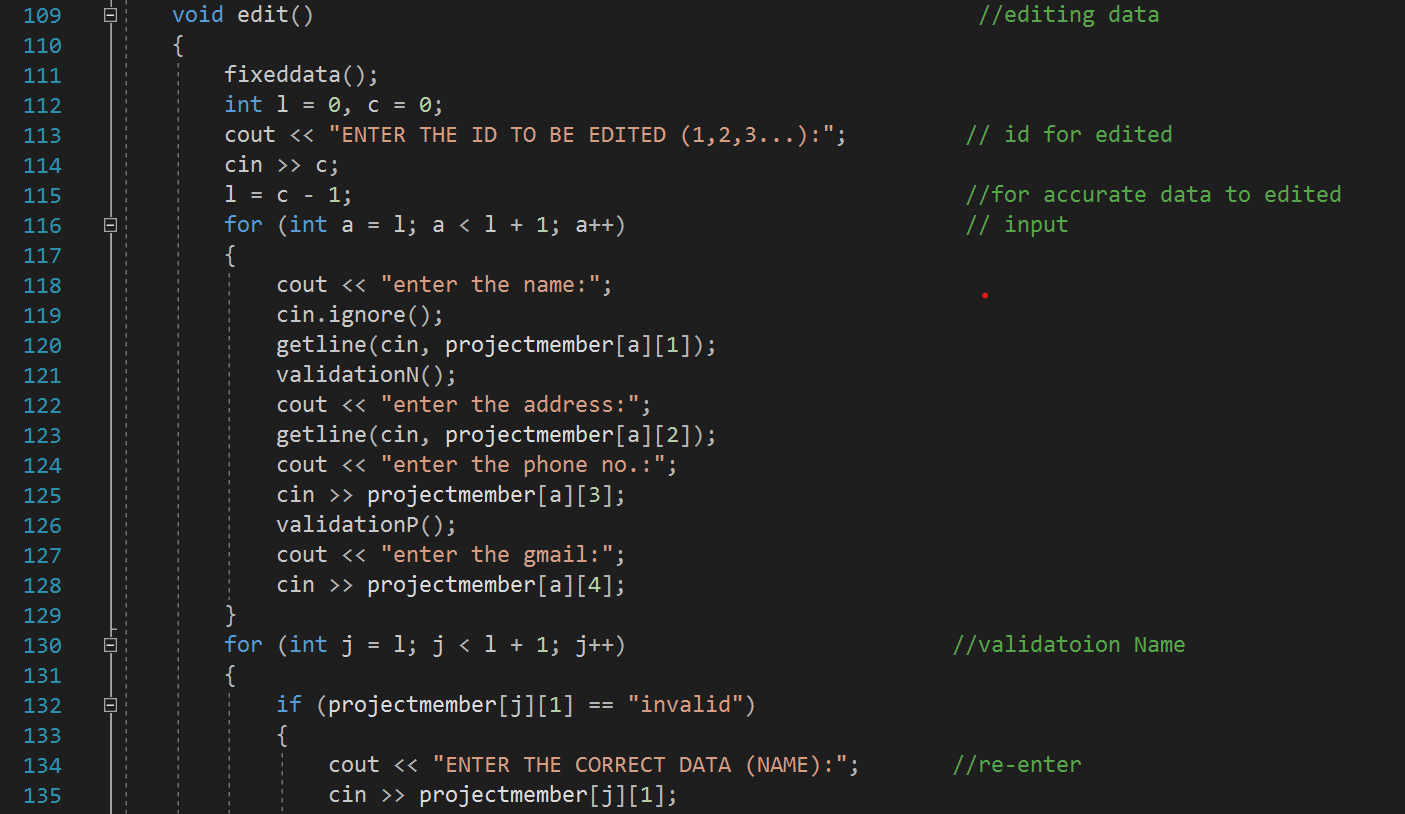
 **~teammember**  🡪destructor

**Membername-** 🡪for displaying predefined members

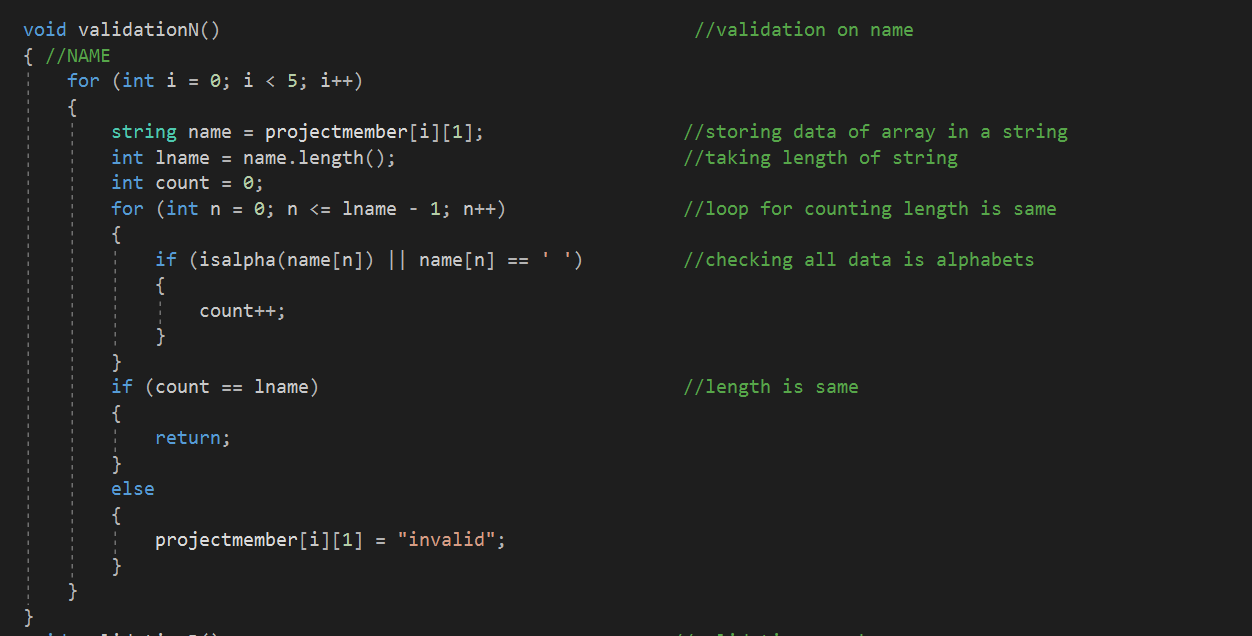
**Add**  🡪for adding new members and validating

****



**Edit** 🡪for editing the member and validating. We are taking id from user and editing all the data according to that id and in the end validation

**del**  🡪for deleting the member we are taking id from user and assigning that column s with null value (0)

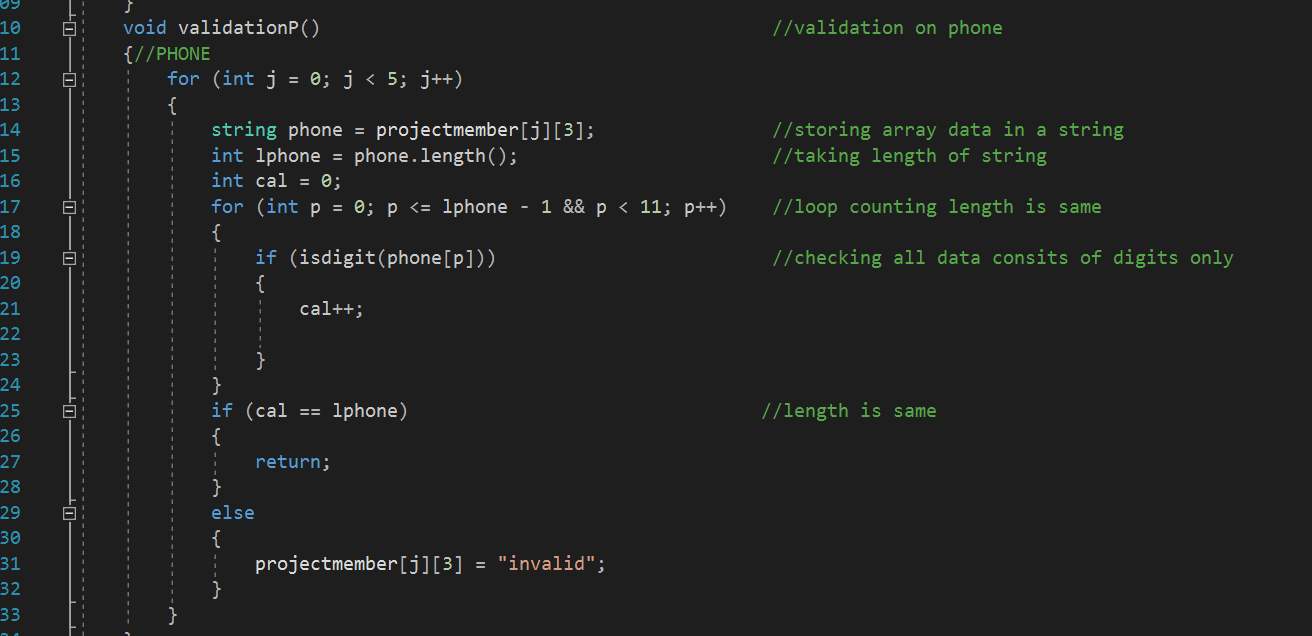


**validationN**  🡪for making sure user enter correct name

we are assigning the name from array to a sting variable

calculating its length, then running loop and checking whether

the data is alphabets. If the length is same then returning, he

value otherwise declaring invalid

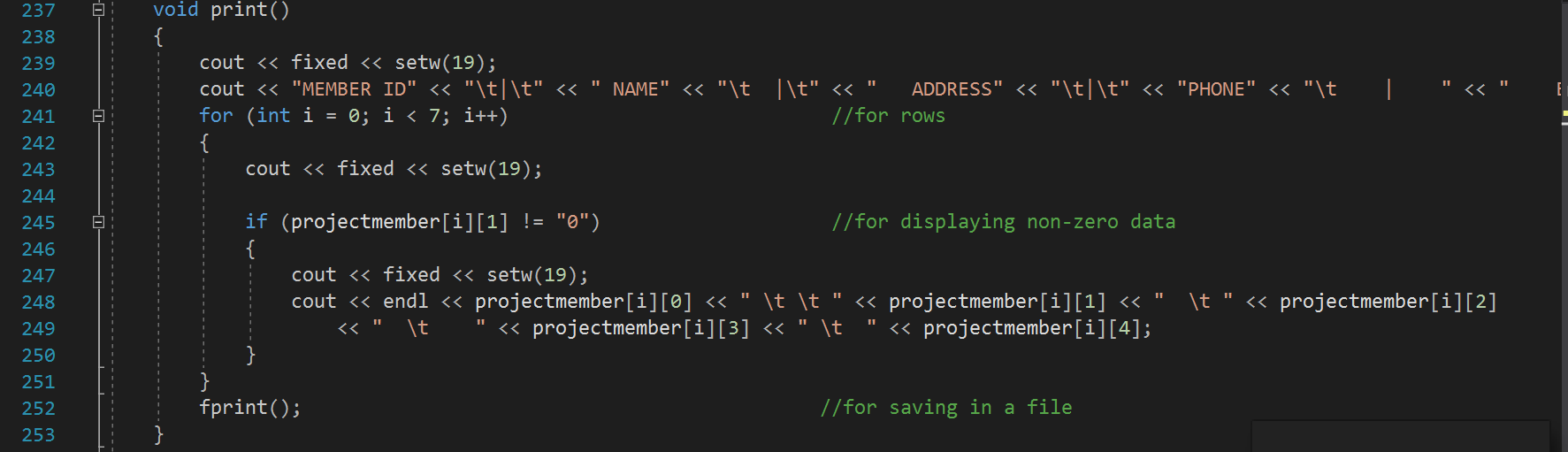
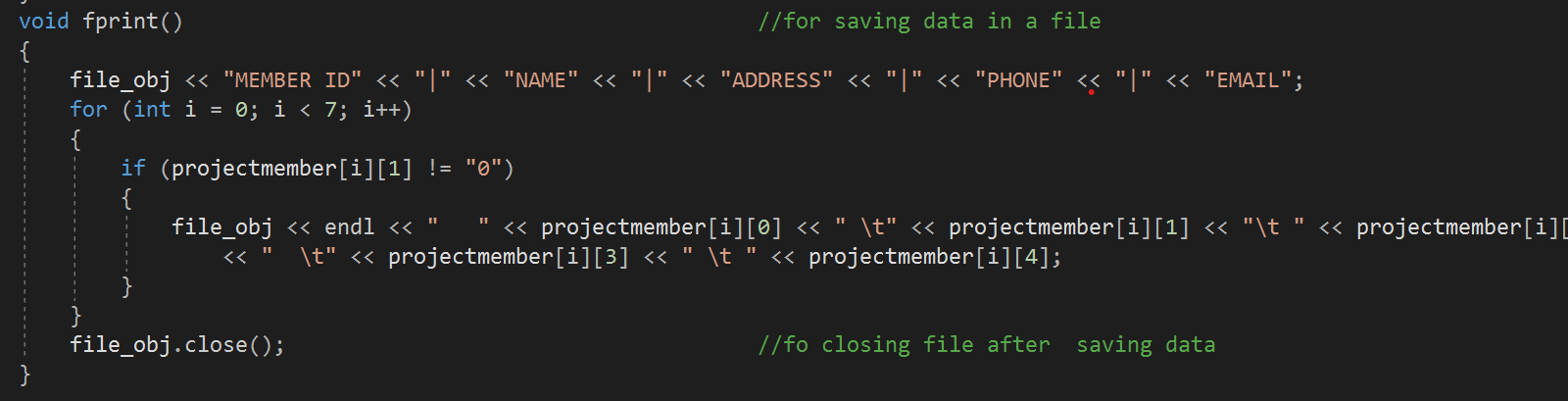
**validationP**  🡪for making sure user enter only digits in phone no

we are assigning the phone no from array to a sting variable

calculating its length, then running loop and checking whether

the data consists of digits. If the length is same then returning,

the value otherwise declaring invalid .

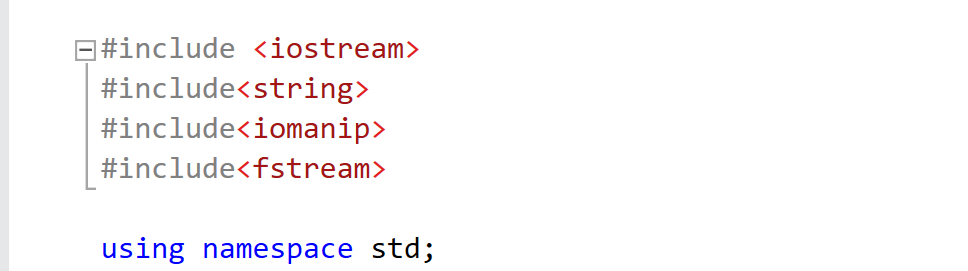


**print**  🡪for printing on console window

**fprint** 🡪for saving data in a file

**Project header files:**

**Project Class**



🡺**String:**

Is used for string data.

🡺**Iomanip:**

Is used to for manipulator functions as setfill(),setprecision() , and setw().

🡺**fstream:**

Is used for file handling.

Class Project:



**Default Constructor:**

Is used to initialize data to zero to prevent the garbage value.

**Parameterized constructor:**

Is used to copy the reference of an argument into the formal parameter.

This function:

Is used to refer the invoking object.



* **Add Function:**

Add function is used to get name and id from the user and after adding data user will receive a message that:

“Do you want to add more..?”

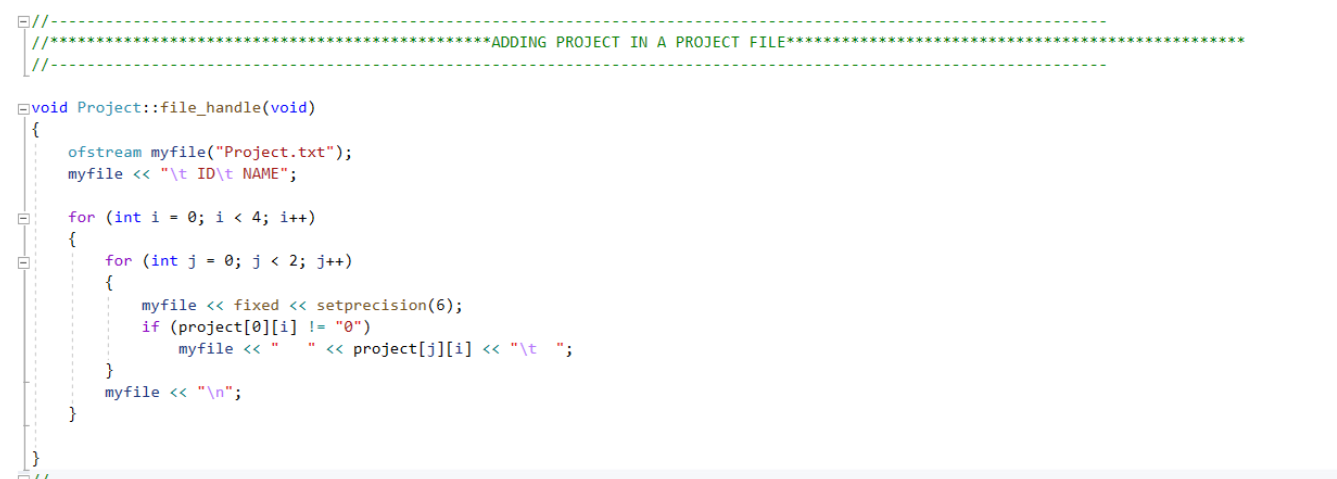
Then by entering ‘n’ you would be move to the menu and by entering ‘y’ user would be able to continue the process.

To\_string :

Is used in add function that converts the integer value into the string.

Due to file handling data will be store in a file.

* **File handling:**

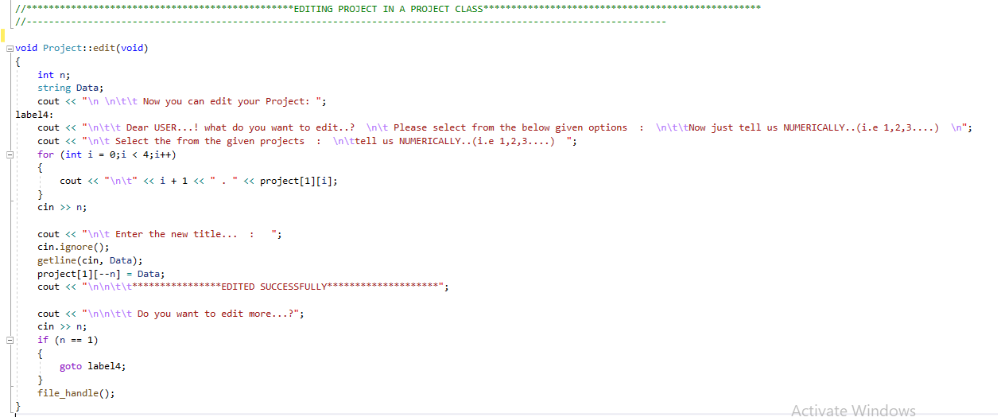


File handling is used to open data into a file “Project.txt”.

Due to ifstream the input would be in a file.

Due to ofstream the output would be in a file.

* **Edit Function:**



Edit function is used to edit the name of the project by getting the new name from the user. Then by entering ‘n’ you would be move to the menu and by entering ‘y’ user would be able to continue editing.

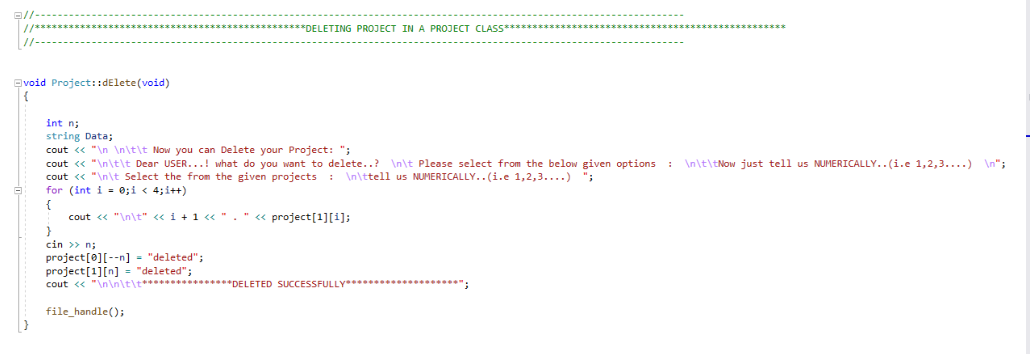
getline(cin,Data):

Is used to read a line or a string from the input stream.

Cin.ignore():

Is a function which is used to ignore or clear one or more characters from the input buffer.

* **Delete Function:**



In this function user will receive a message that “ Now you can Delete your Project:

Dear USER...! what do you want to delete..?

Please select from the below given options :

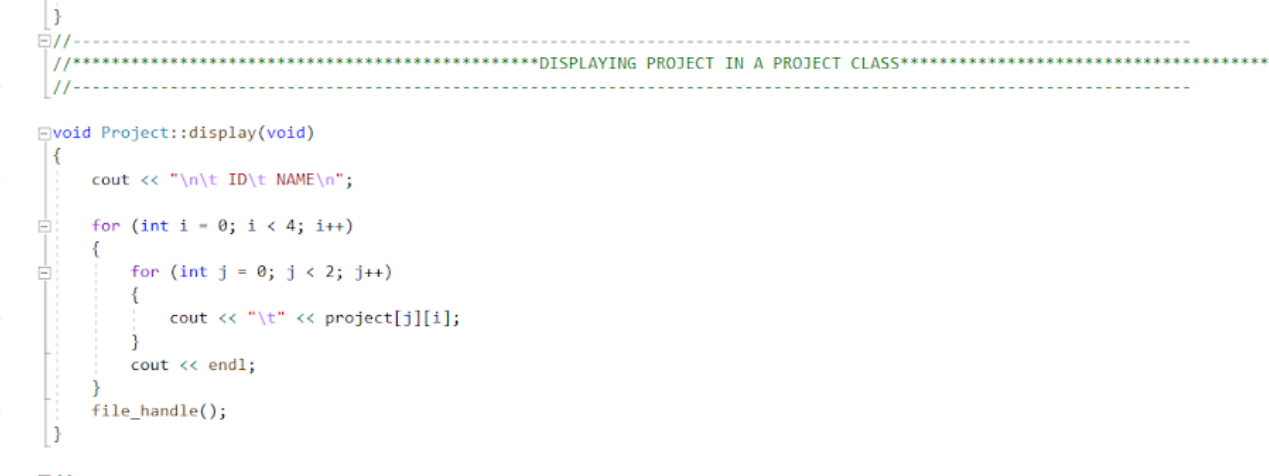
Now just tell us NUMERICALLY..(i.e 1,2,3....) ”

Then by selecting a number by the user (numbers show the data placed in the menu) then by

getting data from the user which user wants to delete and delete it then user will receive a message the ‘DELETED SUCCESSFULLY’.

Then by entering ‘n’ you would be move to the menu and by entering ‘y’ user would be able to continue the process.

* **Display Function:**



This function is used to display the name and id which is added by the user. After this user will again receive a message:

“ Do you want to enter more....(y/n)?”

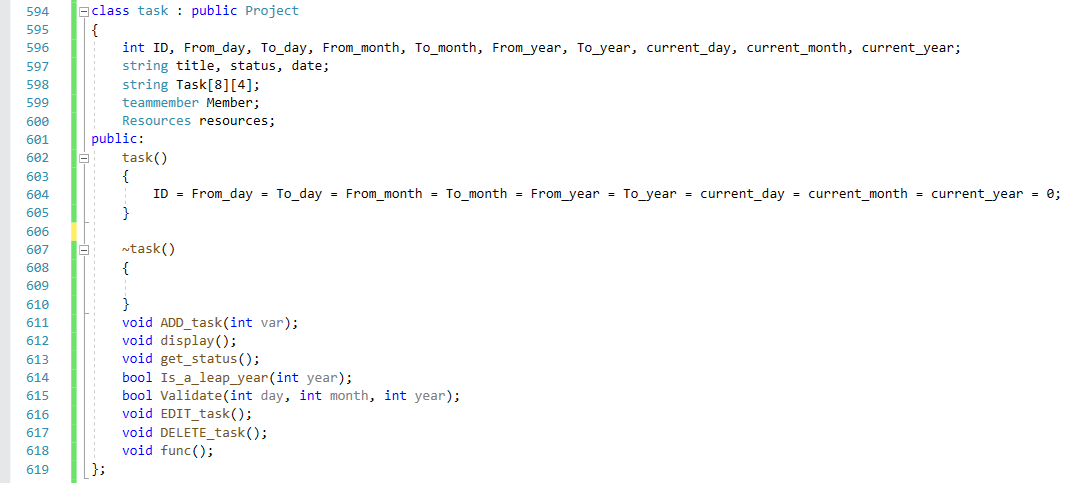
Then by entering ‘n’ you would be move to the menu and by entering ‘y’ user would be able to continue the process.

**TASK Class**

Task class is about performing actions on the tasks for a specified project. Task will get stored in the *txt* file automatically.

## **Description:**

Description of the parts of the task class is given bellow



The task class is having the following data members:

* **ID:**

It is used to store the task Id.

* **From\_Date (From\_day, From\_month, From\_year):**

It is used to get the start bindings of the date for the given task.

* **To\_Date (To\_day, To\_month, To\_year):**

It is used to get the end bindings of the date for the given task.

* **current\_Date (current\_day, current\_month, current\_year):**

It is used to current date.

* **Title:**

It is used to get the title of the task.

* **Status:**

It stores the calculated status, that results from the calculation of the current date and to date.

* **Task [8][4]:**

It is used to finally store data from all the data members into this array, so to print in the file.

* **Member(composition):**

It is used to get the data of *Member* class in this class.

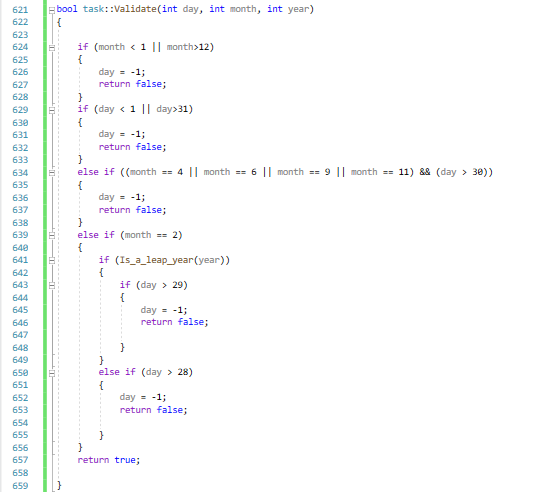
* **Resources (composition):**

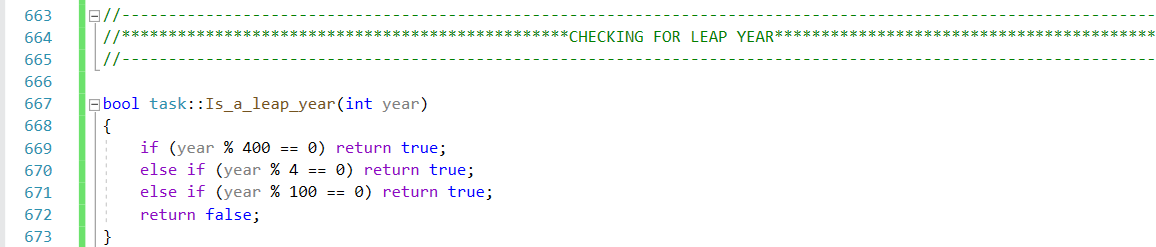
It is used to get the data of *Resources* class in this class.

It is having the following mentioned member functions:

* Validate
* Add
* Edit
* Delete
* Print on console
* Print in file

## **Validate Function:**





**Add Task** function allows the user to add the task for the selected project.

The user has to add the *title* first then the starting date is added in the following pattern:

* **To Date:**

To date take the starting date of the project

* **From Date:**

To date take the ending date of the project

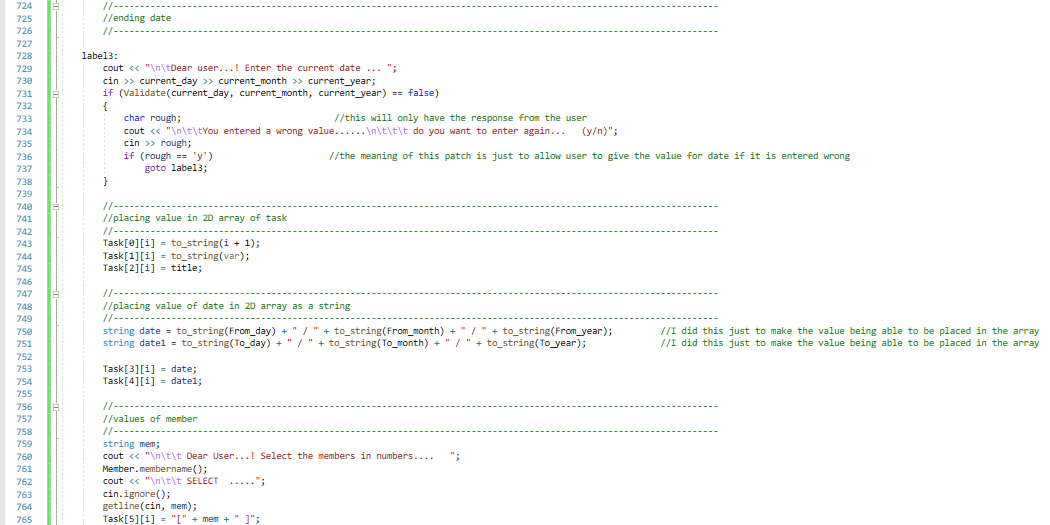
* **Current Date:**

To date take the current date of the project

At first the user is required to *add*the values of date in thepattern having date, month and year as integers

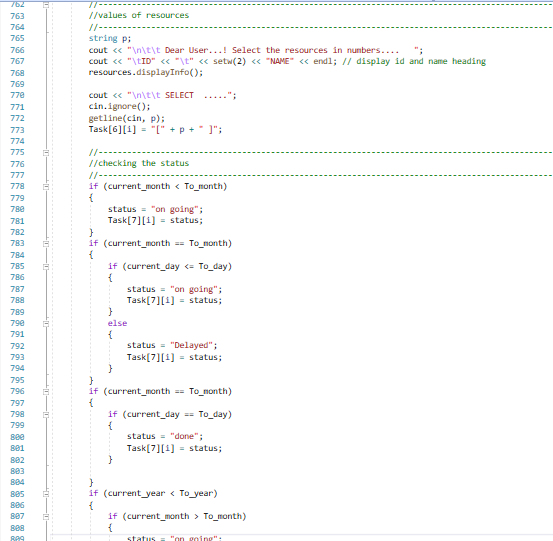
All the entered integers are then *tested* by using the validation function.

Here *static variable* is used, so that whenever this function is called multiple times, the addition of task gets started after the last addition, and does not start from the beginning.



* All the integers are converted into string using *to string* and then concatenated to store in the array, to match the data type of the array

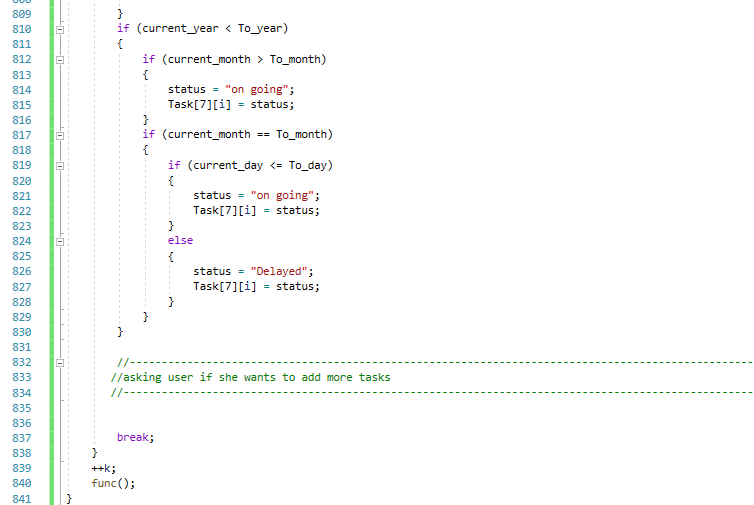
Then the user is required to select the values of members from the given list of the member. This input is also stored in the array in string format.



Here user is asked to select the resources from the given options, the input will be stored in the string data member and that is further placed at the desired location of the array.

The task array holds the values of different attributes of tasks in the given format:

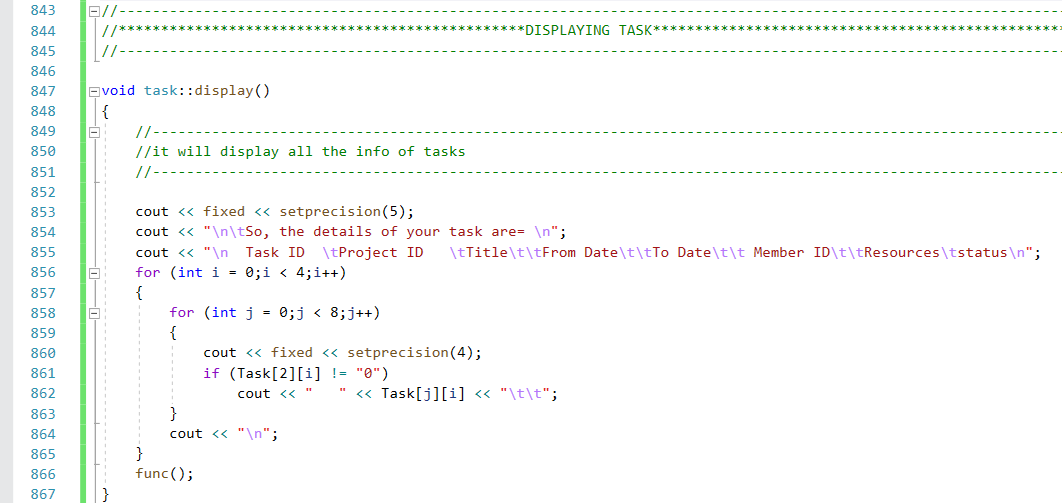
1. The **0th** location has task Id that is automatically generated by incrementing 1 in the counter variable.
2. The **1st** location holds the information about project Id.
3. The **2nd** location holds the title of the task
4. The **3rd**location has the starting date, **4th**location has the ending date and **5th**location the current date.
5. The **6th**location has the members selected for that project.
6. The **7th**location has the resources for the project.
7. The **8th**location has status of the task.



This part shows the status of the task that is generated automatically on the following basis:

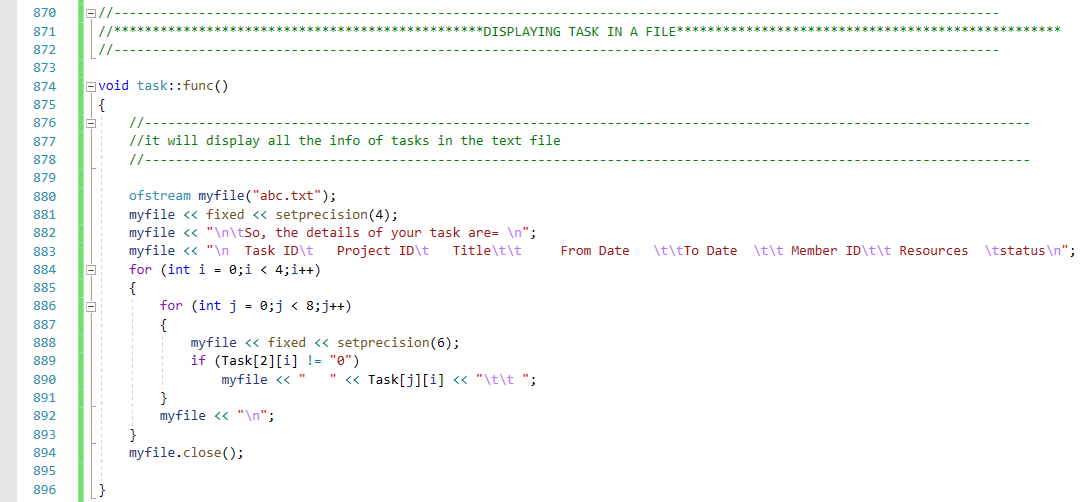
* If the current has exceeded the from date, then status of the task is **Delayed**
* If the current equals the from date, then status of the task is **Done**

If the current is less than the from date, then status of the task is **ongoing**

**Display Task Function:**

**Display Task** will display all the data on console. In this function, simply the array is printed on console because all the data has been added in the array beforehand.

## **Display Task in File Function:**

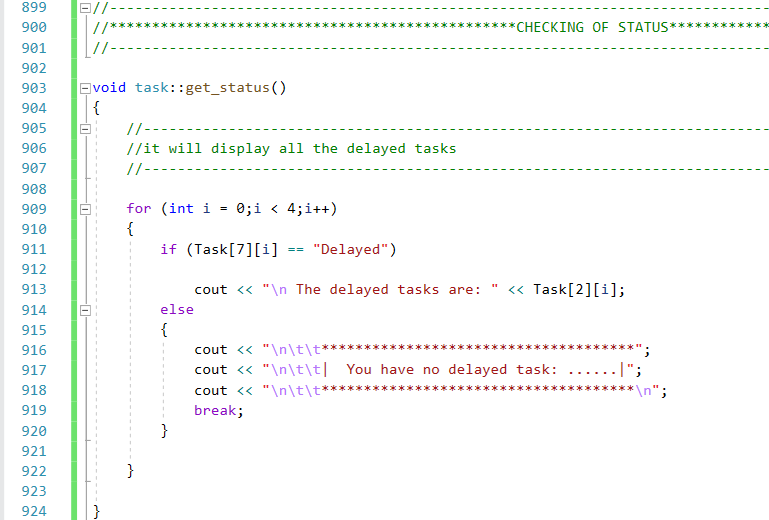


**Display\_Task\_in\_File function** has been created to print all the data in file. This function is added at the end of all the member function. So that all the changes are regularly placed in files.

## **Get Status Function:**

**Get status**

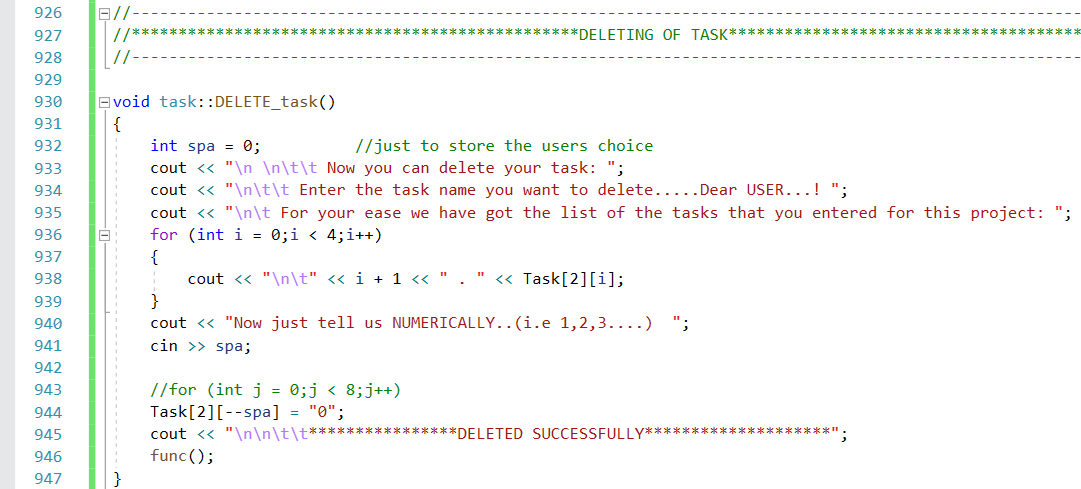
function allows the user to get info about the *delayed* tasks. The program has an array that will check up the array location that has delayed task, then it will display all the titles of required tasks



## **Delete Function:**

This function will simply place “**0** “in all the locations of title, to make it visible in the deleted task, so that when user asks to display the tasks, deleted tasks are not displayed.

The array will skip all those tasks that is having “0” in their title location.



## **Edit Function:**

Purposefully, this function allows the user to edit the tasks. This member function has the option to edit:

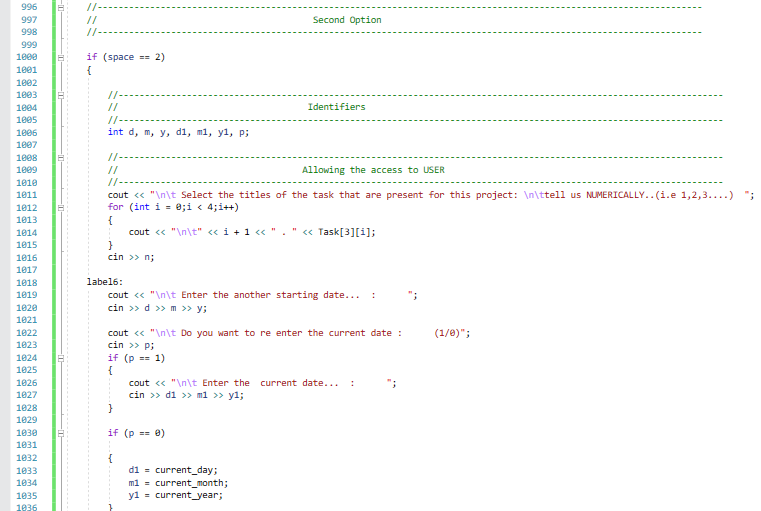
* Title
* To\_Date
* From\_Date
* Member Id
* Resources

Here the user is asked to make a selection that what he/she wants to edit. The menu is displayed to make a selection. The user’s input is placed in an integer variable namely *space*.

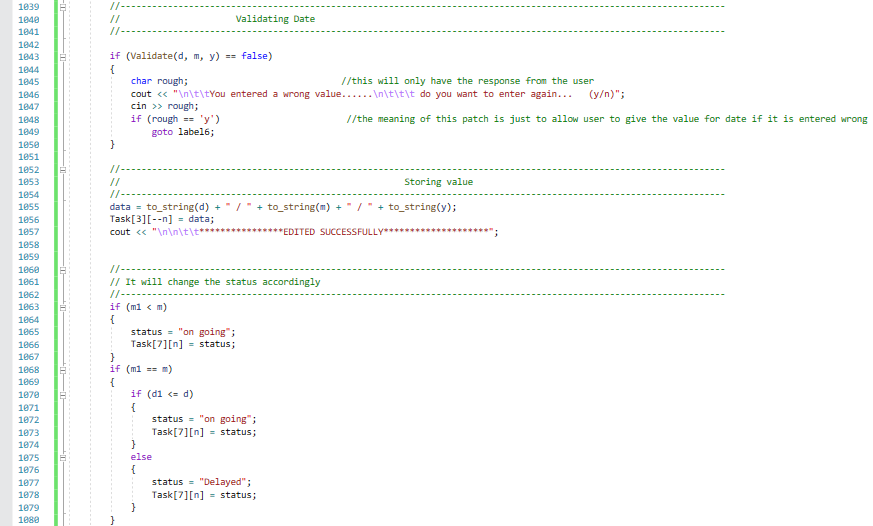
A sequence of the *if else* statement is placed in the edit part, as per the options of the menu.

Then, according to the user’s input a specified patch of the conditional sequence is selected.

If the user selects **1**, he is allowed to edit the title. After he is done with the edition of the title, a message *EDITED SUCCESSFULLY* is printed on console screen. Then he is asked, whether he want to edit more. Then operation is performed in accordance with the input, if **y** is selected *goto label* takes the user to the starting lines allowing him to select again that what he wants to enter

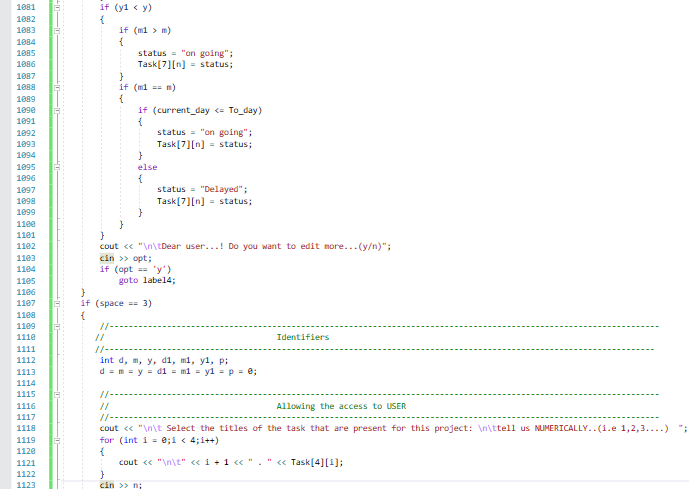


.



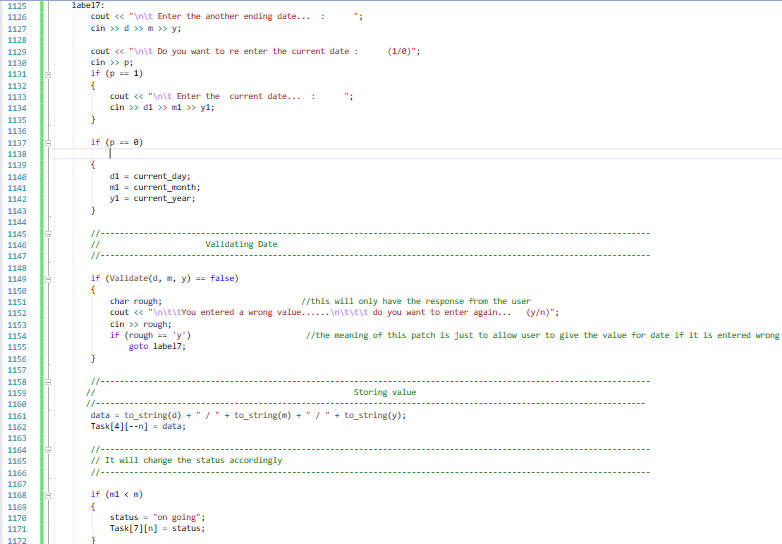
If the user enters **2,** he is allowed to enter starting date. Validation of the date is checked here

again, and same procedure goes on. The entrance of the date is carried in the same pattern as did in Add function. Similarly, the status is checked again.



Next, if user selects **3** he is allowed to edit *From date*, the end bindings of the task. Similarly, as the status depends on current date and from date.

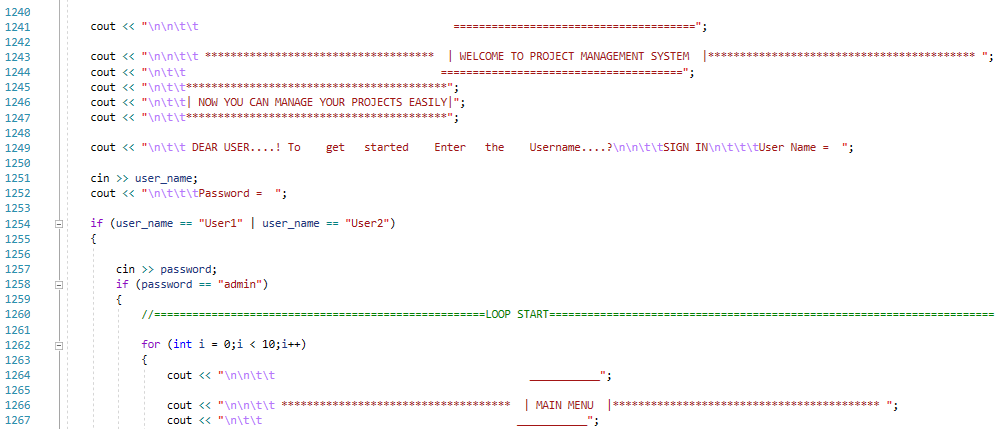




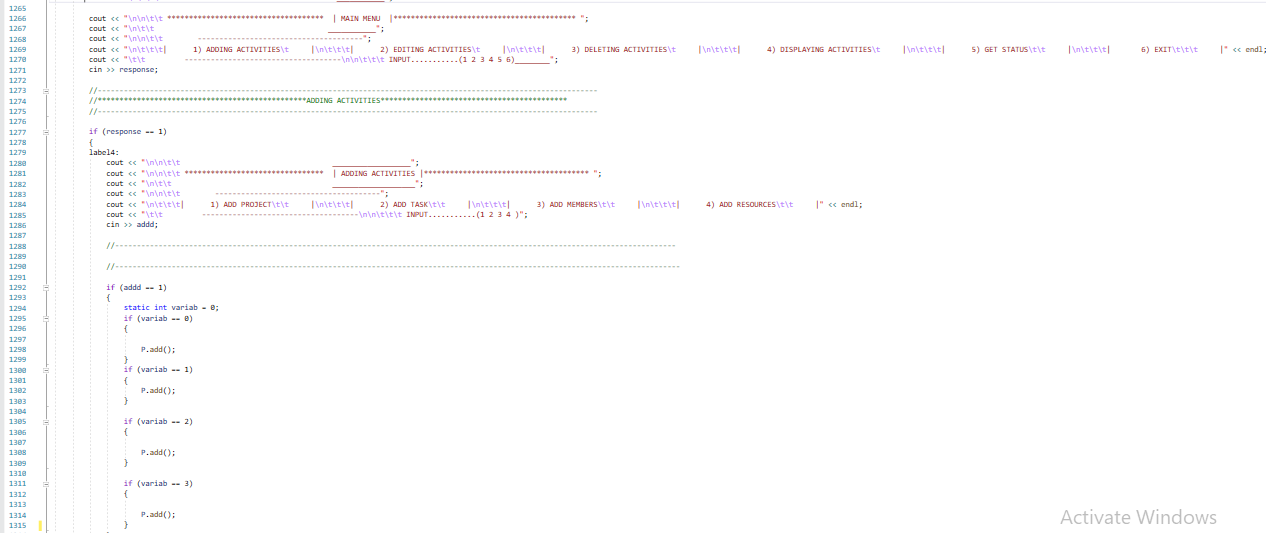
Next the date entered by the user is tested by calling ***Validation function*** if the format is not correct, he is asked to enter again. As, on changing the *from date* will put an effect on *status*, so it will again ask the user whether he wants to enter the current date, or go with the old one. Then the conditions of the status are provided to check it again.

**Main Function**

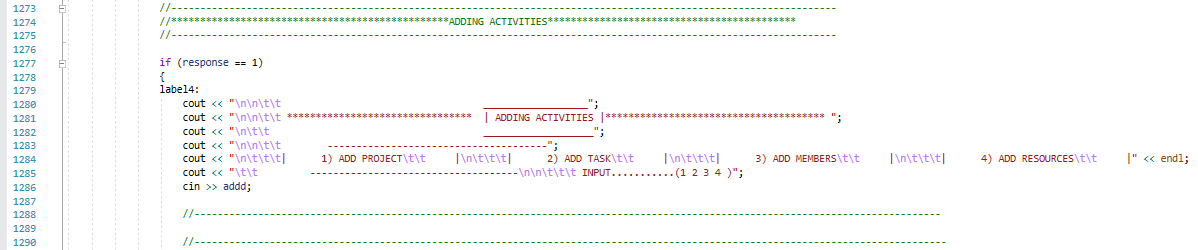
Main function is all about opportuning the user to access the PROJECT MANGEMENT SYSTEM in the best way.



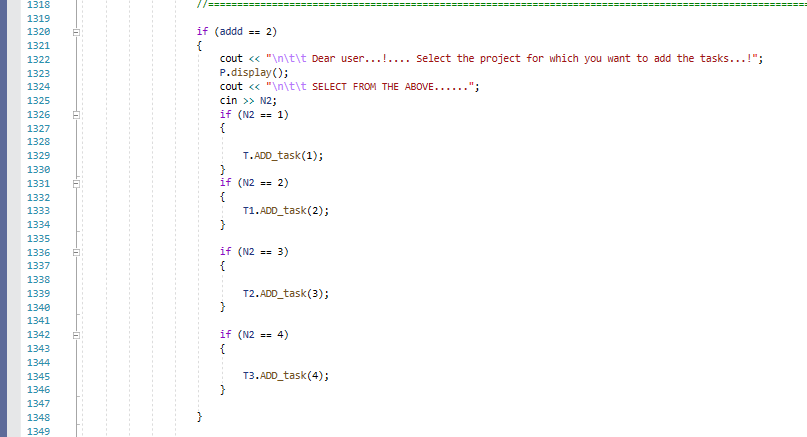
At first the user is asked to enter the **user name** and **password.** If correct, user will have further access. If not, user will get a message and program will terminate.



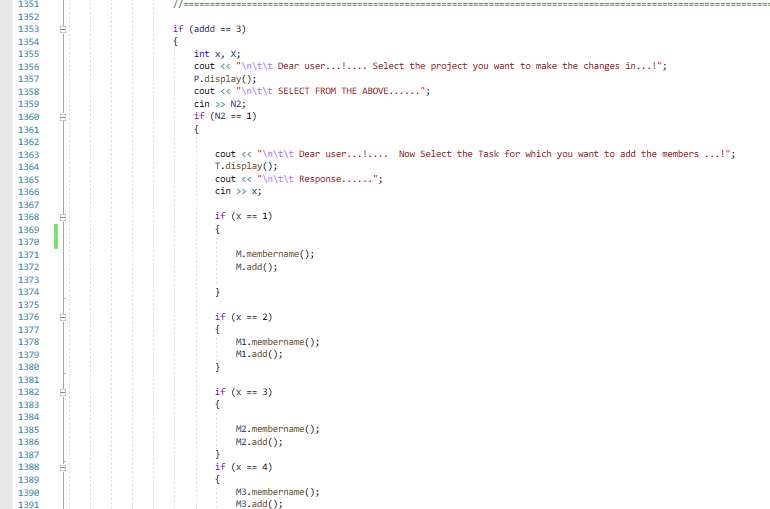
A menu is being displayed to the user displaying the following options:

* Adding activities
* Editing activities
* Deleting activities
* Display activities
* Get status
* Exit
* **Adding Activities**

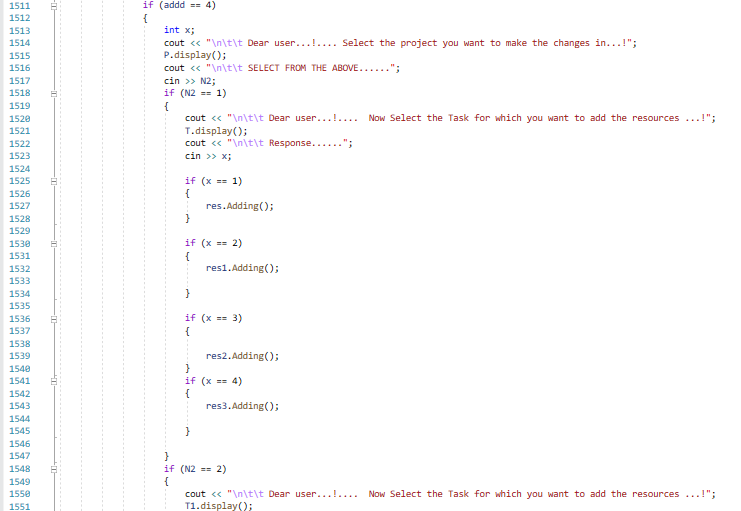
Here if user selects **1,** the user has got the access to add:

* Project
* Task
* Members
* Resources

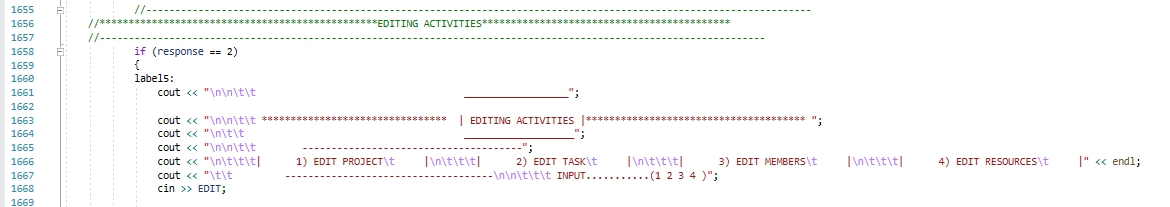
Next on the adding menu, if he selects **1** user can add projects, and if user selects **2**, a list of projects is displayed so that the user can select project to add the task in.



If user selects **3**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to add members for it.

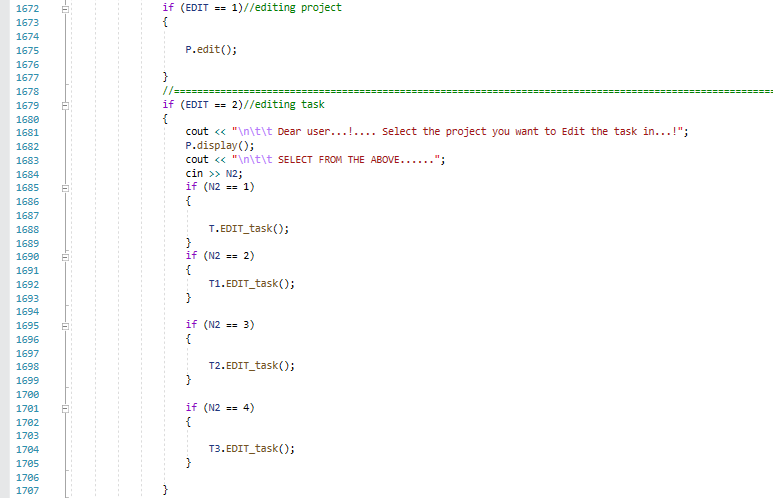


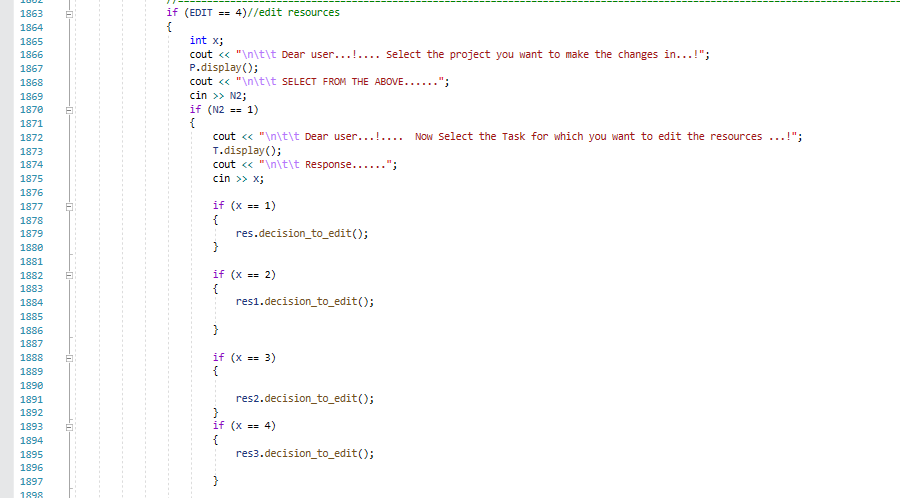
If user selects **4**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to add resources for it.

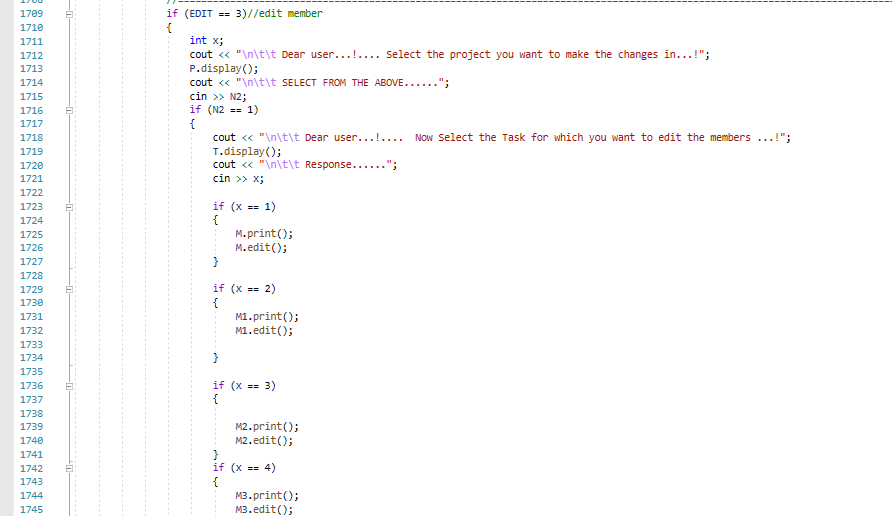
* **Editing Activities**

Then if user selects **1,** the user has got the access to edit:

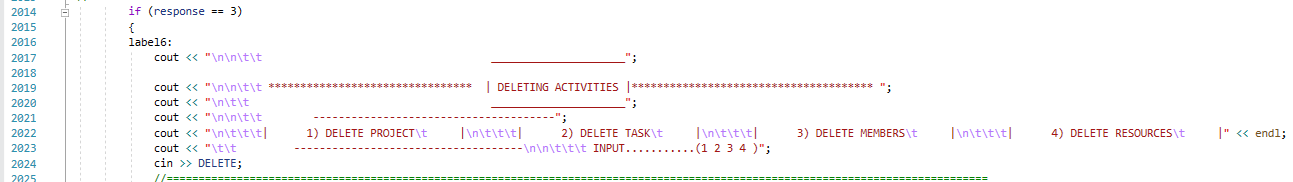
* Project
* Task
* Members
* Resources

****

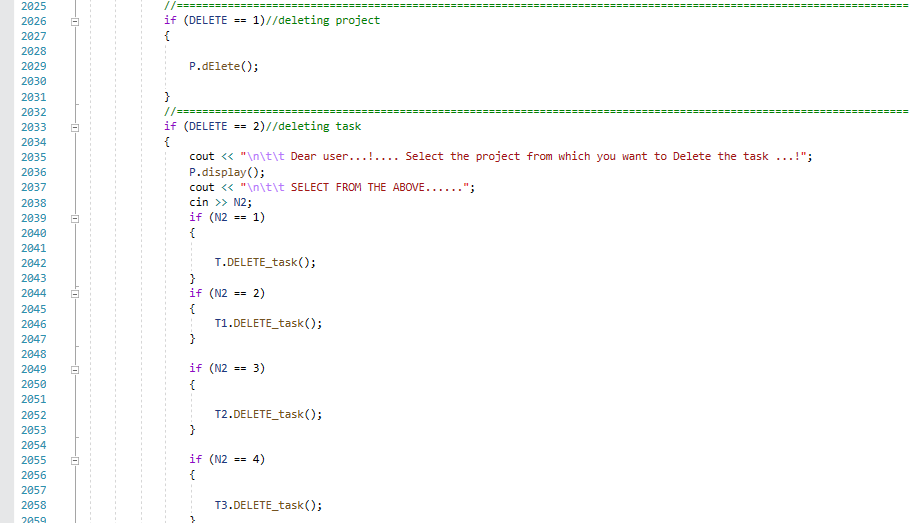
****Next on the editing menu, if he selects **1** user can edit projects, and if user selects **2**, a list of projects is displayed so that the user can select project to add the task in.

****If user selects **3**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to edit members for it.

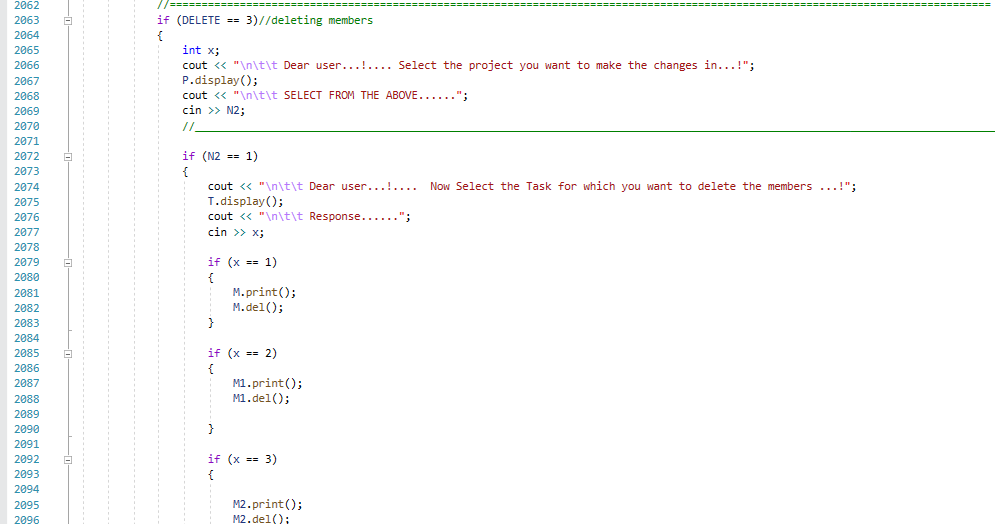
If user selects **3**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to edit resources for it.

* **Deleting Activities**

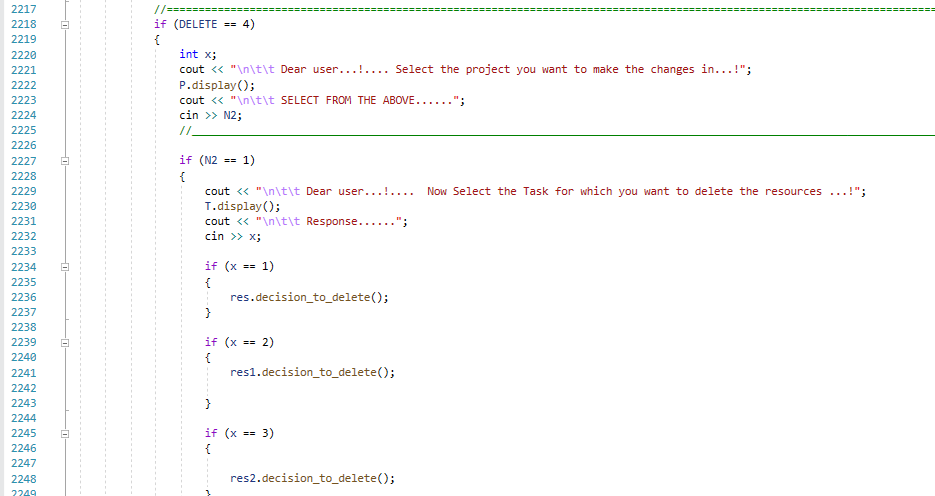
Then if user selects **1,** the user has got the access to delete:

* Project
* Task
* Members
* Resources

Next on the editing menu, if he selects **1** user can delete projects, and if user selects **2**, a list of projects is displayed so that the user can select project to delete the task in.

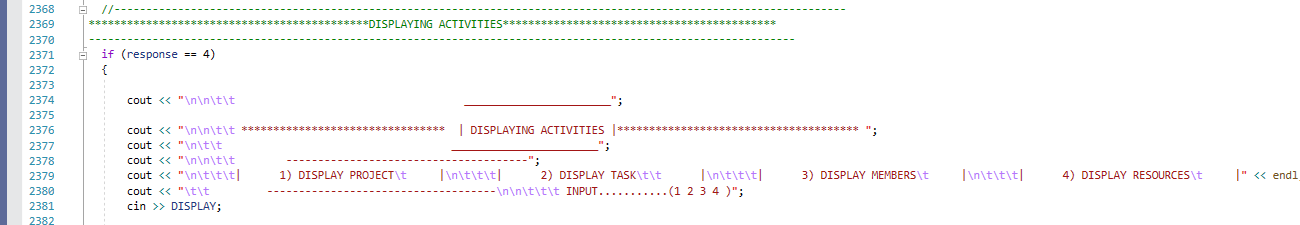


If user selects **3**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to delete members for it.

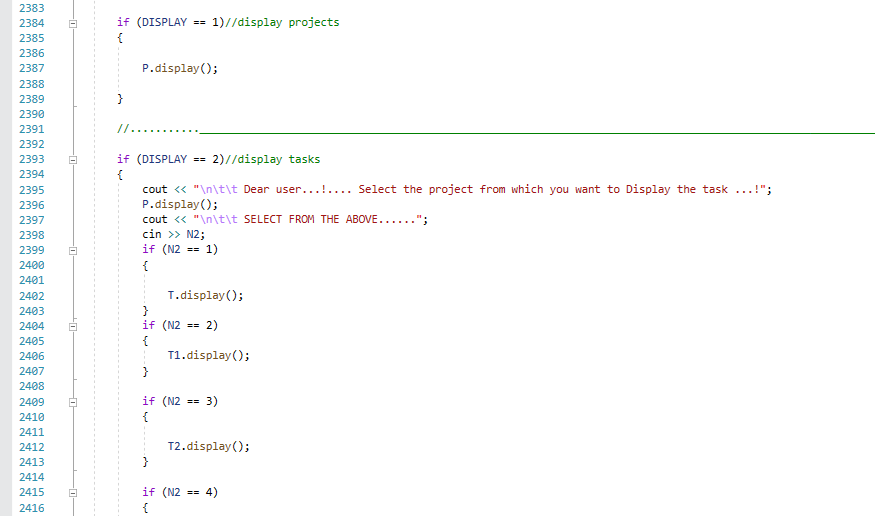
****

If user selects **4**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to delete resources for it.

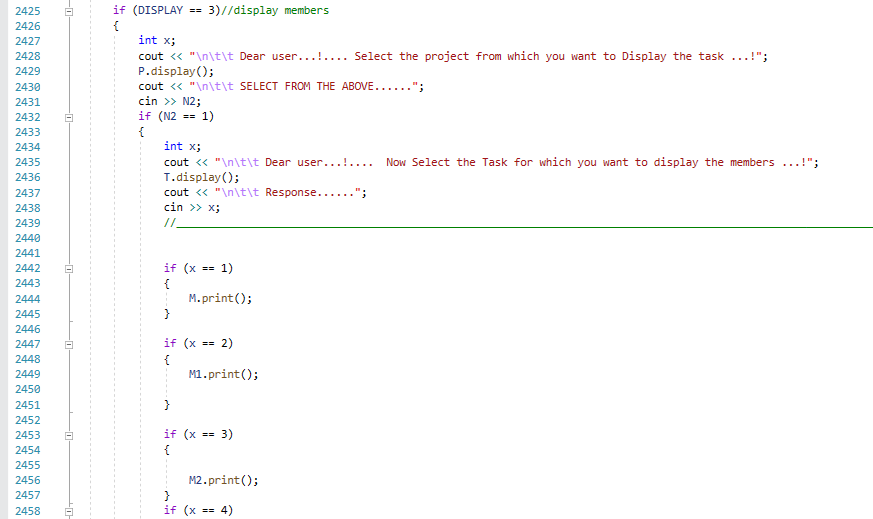
* **Displaying Activities**

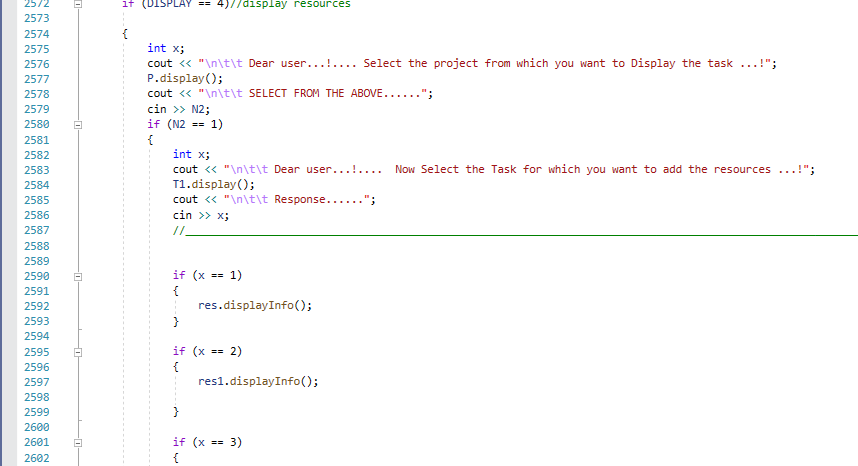
****

Then if user selects **1,** the user has got the access to display:

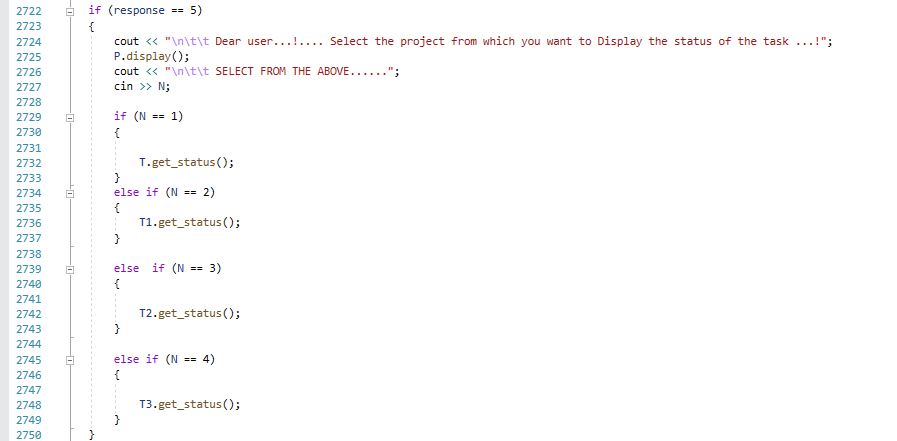
* Project
* Task
* Members
* Resources

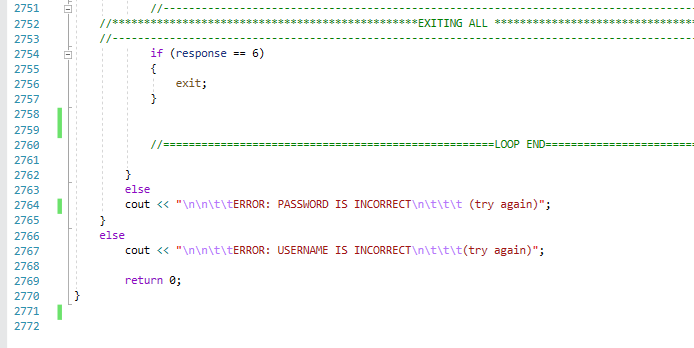
Next on the editing menu, if he selects **1** user can display projects, and if user selects **2**, a list of projects is displayed so that the user can select project to display the task in.



****If user selects **3**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to display members for it.

If user selects **4**, a list of projects is displayed so that the user can select project and then a list to task is displayed so that the user can select the task to display resources for it.



If user selects **5** from the main menu, he is allowed to get all the **delayed** tasks.

Finally, if user selects **6** from the main menu, he gets out of the program.