*Yerkinbek Rinat*

*Neptun code:DSGVVB*

*Lab group ID: EE16A*

*Lab instructor: AL-Magsoosi Husam Kareem Farhan*

*Submission date: 19.05.2024*

*Gym membership management system*

Basics of Programming 2

Final Project

# Introduction

A Gym Membership Management system fulfills the need for a talented and organized system for checking members in the gym office. In today's fitness-conscious society, workout centers are becoming more common, necessitating simplified registration forms. This projects goal to get jobs of gym admins easier by automating the job process.

**Problem Definition**

* Overseeing the enrollment of students at a training center can be a challenging task, especially as the number of people increases. Problems regularly encountered include:
* Data organization: Different individuals can join and leave gym in different times, which is hard to keep track of, leading to errors and irregularities.
* Membership Tracking: Keeping track of membership status, counting start and end dates can be challenging as it occurs on registration sheets or covers.
* Membership types. There are different membership options in gyms. Effective control of these different types is the basis for providing personalized control to individuals.
* Efficient processes: Manual enrollment administration strategies are time-consuming and error prone. Automation of these processes can increase accuracy and efficiency.

**Solution overview**

A gym membership management system offers a comprehensive approach to solve these problems. It provides gym admins with an intuitive and user-friendly interface that allows them to:

* Create members. Easily add unused people to your structure by capturing key points of interest such as title, age and contact information.
* Membership Management: Assign separate registration types to individuals, including Gym Only, Muay Thai or Boxing memberships. Thus, the system tracks the start and end dates of participation, ensuring timely replenishment of the balance and anticipating failures.
* Track membership status. Directors can quickly see each member's membership status by counting dynamic, expired, or upcoming renewals.
* Generate reports: Generate reports to analyze enrollment patterns, revenue forecasts, and other key metrics to help make decisions and prioritize clues.

Gym membership management system, gyms can regulate its work operations and enhance their membership control skills be.

# Program Interface

Below are the steps to start and end the program:

**Executing the program:**

* Activation: to run the program, all necessary dependencies must be installed. This includes C++ compiler and any other required libraries for specific development environment
* Execution: open the IDE (Integrated Development Environment) and go to directory containing the program files.

Compile and Execute: compile the program using IDE or command line. Following command can be used: g++ main.cpp member.cpp gym.cpp muay\_thai.cpp boxing.cpp -o main

**Terminating the program:**

To terminate program simply use stop button on IDE or “Ctrl + C” command in terminal.

By this, the program terminates.

# Program Execution

A gym membership management system offers gym administrators a user-friendly interface to effectively manage memberships and member data.

Main menu:

When program is launched, following navigation menu is shown (Fig.1)

A screenshot of a computer

Description automatically generated

Figure 1 Main menu

To choose the from menu, type the required number into the terminal. All the data is stored into ‘data.rtf’ file inside the source folder. In the start, the data will be uploaded from data file and after execution, updated data will be saved

1. Create Member. When this menu item is chosen program will ask for data of new member, write the required data into a terminal as shown below (Fig.2)

A screenshot of a computer

Description automatically generated

Figure 2 Create member

1. Edit member data. This is for editing data of existing member. Program will ask for ID of member, enter it and new data for the member like shown below (Fig.3)

A screenshot of a computer code

Description automatically generated

Figure 3 Edit member data

1. Sign Up Member for Gym. In this system, one membership can be signed for every member. Program will ask for member ID, just enter it and member will be signed. (Fig.4)

A black background with white text

Description automatically generated

Figure 4 Sign up member

The next 2 items in menu are analogical.

6. Search Member. This option is for searching member by ID and displaying his data. Like before, just enter member’s id like shown below. (Fig.5)



Figure 5 Search member

7. Display All Members. This option displays data of all members to the screen, like shown in the figure below. (Fig.6)

A screenshot of a computer

Description automatically generated

Figure 6 Display all data

In all previous cases the existing members ID must be entered, it member with such ID does not exist, program will show message and return to main menu.

9. Exit. This option stops the program. All data will be saved to data.rtf file.

# Input and Output

**Input**

The input is accepted by console. User interacts with program by choosing required option from menu and entering required data after.

**Output**

The output is also shown in the console: menu options, status messages and members’ information.

Also, program writes the data into a data.rtf file after termination.

**File format (Input/Output)**

* The data.rtf file must already exist in source folder of file. File format:
* Each line represents data of one member.
* Data in lines separated by spaces.
* Format of each line is: “member\_id name age number membershipType startDate endDate”
* For example: “1 Rinat 19 +36206271350 MuayThaiMembership 5-18-2024 11-15-2024”

After termination, program writes new data to same file with same format.

Example input:

A screenshot of a computer

Description automatically generated

Figure 7 Example input

Example output:

A screenshot of a computer

Description automatically generated

Figure 8 Example output

“data.rtf” file:

A screenshot of a computer

Description automatically generated

Figure 9 “data.rtf” file

# Program Structure

## Main **program**:

* Responsible for user interaction, menu options and overall control
* Reads and writes the data to file: readDataFromFile and writeDataToFile functions are used for this. [1]

**readDataFromFile**

Reads member data from a file specified by the filename parameter.

Parses each line of the file to extract member details (ID, name, age, phone number, membership type, start date, end date).

Creates Member objects and adds them to the static vector Member::members.

Updates the next\_id parameter to ensure unique member IDs.

**writeDataToFile(const std::string& filename)**

Writes member data stored in the static vector Member::members to a file specified by the filename parameter.

Formats each member's data (ID, name, age, phone number, membership type, start date, end date) into a line in the file.

* Calculates the date. [2][3]

**getCurrentDate** function returns current time in "YYYY-MM-DD" format.

* Contains main functions:

**createMember(int next\_id)\***

Prompts the user to enter details (name, age, phone number) for creating a new member.

Validates user input to ensure data integrity by using the **exception handling** try and catch.

Creates a new Member object with the provided details and adds it to the Member::members vector.

Increments the next\_id parameter to ensure unique member IDs.

**editMemberData()**

Prompts the user to enter the ID of the member whose data they want to edit.

Searches for the member by ID using Member::findMemberByID().

If the member with such ID not found, it returns to menu

Calls Member::editMember() to edit members data, by creating new object of member class and changing the previous by new one

**searchMember()**

Prompts the user to enter the ID of the member they want to search for.

Searches for the member by ID using Member::findMemberByID().

If the member is found, displays their details including membership information if available.

**displayAllMembers()**

Calls Member::displayAllMembers() to display details of all members stored in the Member::members vector.

**signUpForMemberGym()**

Prompts the user to enter the ID of the member they want to sign up for gym membership.

Searches for the member by ID using Member::findMemberByID().

If there is no member with such ID, it goes to main menu.

Sets the member's membership to the newly created GymMembership object.

**signUpForMemberMuayThai()**

Similar to signUpForMemberGym(), but for signing up members for Muay Thai membership.

**signUpForMemberBoxing()**

Similar to signUpForMemberGym(), but for signing up members for Boxing membership.

* Included libraries

<iostream>: For input and output operations

<fstream>: For handling file input and output operations.

<sstream>: For work with string streams

<stdexcept>: This header provides standard exception classes.

"member.hpp": For having the access to the classes and methods in the member.hpp header file.

"gym.hpp", "muay\_thai.hpp", "boxing.hpp": For having the access to classes and methods defined in those header files.

<chrono>, <ctime>: These headers are included to work with date and time functionality.[2]

The using namespace std; directive is included to avoid having to prepend standard library elements (such as cin, cout, etc.)

Member.hpp and member.cpp

Constructor and Destructor:

Definitions for the constructor and destructor are provided, including initialization of member variables and memory deallocation.[4]

Getter and Setter Implementation: Definitions for getter and setter methods are implemented to retrieve and modify member data.

* getMemberID(): Returns the member ID.
* getName(): Returns the member's name.
* getAge(): Returns the member's age.
* getNumber(): Returns the member's phone number.
* getMembership(): Returns the pointer to the membership object.
* setMembership(): Sets the membership pointer to the provided membership object.

Static Member Functions

* createMember():

Creates a new Member object with the provided parameters (ID, name, age, phone number).

Adds the this created member to the static vector members [4].

Returns a pointer to the this created member object.

* editMember():

Edits the data of an existing member (name, age, phone number).

Accepts a pointer to the member and the new values for name, age, and phone number.

* findMemberByID():

Searches for a member with the specified ID in the static vector members [4].

Returns a pointer to the member if found, otherwise returns nullptr.

Vector members works on base of dynamic memory management

* displayAllMembers():

Iterates through the static vector members.

Displays information about each member: ID, name, age, phone number and membership

* cleanUpMemory():

Deallocates memory for all members stored in the static vector members.

Clears the vector after deleting all member objects.

member.hpp declares the member class and its interface, while member.cpp provides the implementation details for the class methods.

## membership.hpp and membership.cpp

Membership Class:

Membership class is a base class for GymMembership, MuayThaiMembership, BoxingMembership

Constructor: Initializes a Membership object with provided start and end dates.

Virtual Destructor: Ensures proper cleanup of derived class objects.

Virtual Method: displayDetails(): Provides a default implementation to display membership details, to be overridden by derived classes.

Inherited GymMembership, MuayThaiMembership, BoxingMembership classes

These classes are similar, they all have same methods and members.

Constructor: Initializes a object with start and end dates.

displayDetails(): Overrides the base class method to display membership details, including start and end dates.

The getters:

* getStartDate(): Returns the start date of the gym membership.
* getEndDate(): Returns the end date of the gym membership.
* getType(): Returns the type of membership.

Program Interface

**Setting Up**

These tools needs to be installed:

C++ Compiler

Standard Libraries: <iostream>, <fstream>, <sstream>, <stdexcept>, and <chrono>.

**Compilation**

In order to compile program, choose compilation option in IDE or write this code into terminal:

g++ -o main main.cpp member.cpp gym.cpp muay\_thai.cpp boxing.cpp membership.cpp

**Execution**

Use following command in terminal: ./main

**Terminating the Program**

Select the "9. Exit" option from the main menu.

The program will write data to data.rtf and perform cleanup before terminating.

Testing and Verification

To test program, lets execute and try every function in the menu with different inputs.

1. Create a member.

If we try to create member empty name, negative number for age and empty phone number, following errors occur. (Fig.10)

A screenshot of a computer

Description automatically generated

Figure 10 Create a member

1. Edit Member Data. If we try to enter ID of non-existing member (Fig.11)

A black background with white text

Description automatically generated

Figure 11 Edit Member Data

Same error occur during 3, 4, 5th options in menu- trying to sign non-existing member to membership. (Fig.12)

A black background with white text

Description automatically generated

Figure 12

7. Display all members, just show all data (Fig.13)

A screenshot of a computer

Description automatically generated

Figure 13

9. Exit. This will save the updated data to data.rtf file (Fig.14)

A screen shot of a computer

Description automatically generated

Figure 14

Improvements and Extensions

* Input handling, more accurate validation of data can be implemented for input data: phone number, name, and age.
* User interface: more user-friendly GUI – graphical user interface can be implemented, additional options can be created: delete member, extend membership, calculate the price.
* File handling: for suitable format for data storage can be implemented using additional libraries.
* Membership management: more methods and functionalities can be implemented to customize memberships, calculate the prices or addition of more membership classes.

Difficulties Encountered

Calculation the time was a problem for me, but I found a solution in internet portal for this. Using the error handling was also an obstacle, checking the data for its format and displaying error. In addition, inheritance: creating virtual functions so they can be accessible from main program, just visualizing the data, passing it by pointer, how it should be approached was not easy. Working with members data was complicated, so I learned how about vectors, how to use them to work with data [4]. Connecting the object of member class to vector principle was difficult. In general, there were some difficulties, but they helped me to understand the C++ programming language better.

Conclusion

The design of a comprehensive membership management system for a gyms effectively illustrates the application of object-oriented programming standards in C++.

The project created a strong basis for the control system, displaying basic coding skills and understanding of Object Oriented principles. This meeting and the lessons learned from this endeavor will no doubt be useful for future program development efforts.

The dynamic memory management, exception handling and file management were implemented as well as inheritance during the development of the program.

References

Websites:

1. Cplusplus.com, "Input/output with files - C++ Tutorials," [Online]. Available: <https://www.cplusplus.com/doc/tutorial/files/> . [Accessed: May 12, 2024].
2. GeeksforGeeks, "C++ Date and Time," [Online]. Available: <https://www.geeksforgeeks.org/c-date-and-time/> . [Accessed: May 14, 2024].
3. Stack Overflow, "How to calculate the next month in C++?", [Online]. Available: <https://stackoverflow.com/questions/24952668/how-to-calculate-the-next-month-in-c> . [Accessed: May 18, 2024].
4. Stack Overflow, "How can I create objects while adding them into a vector?," Stack Overflow, Apr. 4, 2013. [Online]. Available: <https://stackoverflow.com/questions/15802006/how-can-i-create-objects-while-adding-them-into-a-vector> . [Accessed: May 19, 2024].