Experiment No: 01

Experiment Name: How to improve code structure and readability.

Theory: Code readability is a universal subject in the world of computer programming. It's one of the first things we learn as developers. Making our code easier to read will also help us debug our own programs, making it easier on ourself. This experiment will detail the most important best practices when writing readable code.

Some methods to improve code structure and readability

1.Commenting and documentation

commenting on code make it easier to understand than ever.

```
#include <iostream>
#include <fstream>
using namespace std;

// Class to define the properties
class Contestant {
public:
    string Name;
    int Age, Ratings;

    int input();

    // Function declaration of output_highest_rated() to
    // extract info from file Data Base
    int output_highest_rated();
};
```

2.Consistent Indentation

Consistent indents are vital for writing code that's readable for humans.

```
    while (!file_obj.eof()) {
        // Assigning max ratings
        if (obj.Ratings > max) {
            max = obj.Ratings;
            Highest_rated = obj.Name;
        }

        // Checking further
        file_obj.read((char*)&obj, sizeof(obj));
    }
```

3.Avoid Obvious Comments

Commenting on code is fantastic. However, it can be overdone or just be plain redundant.When the text is that obvious, it's really not productive to repeat it within comments.

```
#include <iostream>
#include <fstream>
using namespace std;

// Class to define the properties
class Contestant {
public:
    //input name
    string Name;
    //input age and ratings
    int Age, Ratings;

    //call input function
    int input();

    // Function declaration of output_highest_rated() to
    // extract info from file Data Base
    int output_highest_rated();
};
```

4.Consistent Naming Scheme

The names should have word boundaries. There are two popular options:

- **camelCase:** First letter of each word is capitalized, except the first word.
- **underscores:** Underscores between words, like: mysql_real_escape_string().

```
int Contestant::output_highest_rated()
{
    ifstream file_obj;
    file_obj.open("Input.txt", ios::in);
    Contestant obj;
    file_obj.read((char*)&obj, sizeof(obj));
    int max = 0;
    string Highest_rated;
    while (!file_obj.eof()) {
        // Assigning max ratings
        if (obj.Ratings > max) {
            max = obj.Ratings;
            Highest_rated = obj.Name;
        }
        file_obj.read((char*)&obj, sizeof(obj));
    }
```

```
    cout << Highest_rated;
    return 0;
}
```

5.Avoid Deep Nesting

Too many levels of nesting can make code harder to read.

```
int Contestant::output_highest_rated()
{
    ifstream file_obj;
    file_obj.open("Input.txt", ios::in);
    Contestant obj;
    file_obj.read((char*)&obj, sizeof(obj));
    int max = 0;
    string Highest_rated;
    while (!file_obj.eof()) {
        // Assigning max ratings
        if (obj.Ratings > max) {
            max = obj.Ratings;
            Highest_rated = obj.Name;
        }
        file_obj.read((char*)&obj, sizeof(obj));
    }
    cout << Highest_rated;
    return 0;
}
```

6.Code Grouping

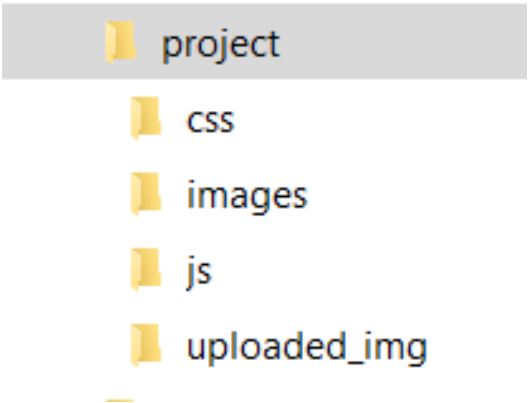
More often than not, certain tasks require a few lines of code. It is a good idea to keep these tasks within separate blocks of code, with some spaces between them.

```
int Contestant::input()
{
    ofstream file_obj;

    file_obj.open("Input.txt", ios::app);

    Contestant obj;

    string str = "Michael";
    int age = 18, ratings = 2500;

    obj.Name = str;
    obj.Age = age;
    obj.Ratings = ratings;

    file_obj.write((char*)&obj, sizeof(obj));
```

```
    str = "Terry";
    age = 21;
    ratings = 3200;

    obj.Name = str;
    obj.Age = age;
    obj.Ratings = ratings;

    file_obj.write((char*)&obj, sizeof(obj));

    return 0;
}
```

7.File and Folder Organization:

Technically, we could write an entire application's code within a single file. But that would prove to be a nightmare to read and maintain. File management is organizing and keeping track of files and folders, helping you stay organized, so information is easily located.



Conclusion:

In conclusion in this experiment we learn how to made code more readable.Though code structure and readability is a huge topic we tried to over some of them.