4/18/2017

|  |
| --- |
| ***With Knowledge and Entertainment Sessions*** |



Report to the Object Oriented Programming Project

|  |  |
| --- | --- |
| B.Vasudha R.Ruhitaj | **GAMING WITH OOP** |

**ABSTRACT** : The project is a successful attempt to make the concepts of Object Oriented Programming (OOP) - interactive, joyful and understandable using the coding language C++. The project has ‘Entertainment’ and ‘Knowledge’ session, where each round is provided with score. The code has concepts of OOP like-

* Classes and objects
* Dynamic Memory allocation
* Constructor and Destructor
* Function overloading
* Operation overloading
* Friend Function
* Virtual function
* Inline function
* Generic classes
* Type Conversion
* This pointer
* Inheritance

**REQUIREMENTS:**

* **Hardware Requirements:** The code runs successfully on any Computer having drivers suitable for compiling codes of C++ language and which can play sounds of .wav format.
* **Software Requirements:** The code runs successfully on Code::Blocks 13.12 with lwinmm linker and libbgi.a linker library installed into the code blocks, before attempting to run the code. It works well on other compiler software of C++ as well, if the header <mmsystem.h> can be successfully run, where the “Playsound” function works well. Turbo C++ is also a supporting software environment.
* **User Requirements:** The code can be used by anyone, who can understand Basic English and knows to compile and run the code.

**OPERATING THE PROJECT:**

**The project has two major sessions.**

* **Entertainment Session:**

It has three levels, played by three players belonging to the same team. The score is inherited from player 1 to player 2 and from player 2 to player 3. The score of team is the score of the player3.

Level 1- ***Guess Music*:**

The player 1 of the team has to guess music according to the music played, during the execution of the code. The player scores +1 on winning the game and -1 on losing the guess. The user has to play this level for three music sound tracks.

Level 2- ***Rock Paper Scissor:***

The player 2 of the team has to play the classic rock-paper-scissor, with the computer. The game is played intelligently by the computer. The player has to play this level for 5 trials. If the player wins, then the player scores +1 on winning the trial with computer and -1 on losing the trial with computer. The score of player 1 is inherited by the player 2 and player 2 passes on the score to player 3, to be inherited.

Level 3- ***Mine Sweeper:***

The player 3 of the team has to play the mine sweeper, where the user gets to choose the number of boxes in the mine sweeper. Player can choose to use the default number of boxes. Player is requested to enter within the range of particular boxes. Now, after he chooses, the boxes are displayed, with the scores the boxes hold. The player has to guess some boxes, where player thinks there are no bombs. If the number, the player chose has a bomb, score of the player is decremented by the same number and incremented by the same number, if there is no bomb.

The score obtained here will be the score of team. This will be end of Entertainment session.

* **Knowledge Session :**

It has simple arithmetic operations like addition, multiplication and subtraction, which uses templates enabling the use of both integers as well as float data types. The learner gets to choose both the adding numbers and he is only asked to enter the value of the correct answer. If the answer is correct, his score is incremented by +1 and if the answer is wrong, the score is decrement as -1 is added to the code. The process is similar in the way for both subtraction and multiplication as well.

**USAGE OF VARIOUS CONCEPTS:**

* ***Classes and Objects:*** The entire code and the functionality is based on the usage of the classes and objects. Classes are the pillars of the gaming, where each type of game is a class. In order to run the code objects are created from the classes of their respective types.
* ***Dynamic Memory Allocation***:The class ‘MineSweeper’ uses the dynamic memory allocation to smartly save space, when the user inputs the places where he thinks there are no bombs.
* ***Constructor and Destructor:*** All the classes used in the Gaming have a constructor, where the required functionality is created and allows user to smartly differentiate game from one to one. Destructor is used in order to delete the used data after using the values and hence saving the space.
* ***Function Overloading:*** The code has two function overloading properties. In the game of mine sweeper, the user gets to no of boxes or the user can let it be default, where the constructor is overloaded. Cout is overloaded as ”output”.
* ***Operator overloading***: Pre- increment and pre-decrement are over loaded in order to increment or decrement the score in the games- Guess music and rock-paper-scissor.
* ***Friend function:*** In overloading the “<<” operator, friend function is used to access the data members of the class.
* ***Virtual Function:*** Display is a function, which is all through the game, which is a base class as well as all the other three game derived classes. Virtual function was used in order to differentiate between the main “display” function in the base class and the display function in the virtual class.
* ***Inline Function:*** It is used for a small function, which occupy a single line to reduce the time taken to execute and run the program. It is used in the template classes and display function.
* ***Generic Classes:*** These are classes which use the key word template. These are used in the knowledge session.
* ***Type conversion:*** The code uses the basic three types of type conversion, i.e., BUILT IN-TO-OBJECT TYPE, OBJECT TYPE-TO-BUILT IN TYPE, ONE-TO-ANOTHER OBJECT TYPE. These are used in knowledge sessions.
* ***Inheritance***: In the entertainment session, all the derived classes, i.e., the three games- inherit their basic properties from the main base class game.