**Chapter Three: Review of Related Literatures**

This game is based in middle-level language general purpose language i.e. C++ programming. In this project we use important parts of C++ programming which are class, objects, interface, inheritance, pointer, data file, functions, control statement, looping.

**C++ programming language:**

C++ is a programming language developed by Bjarne Stroustrup in 1979 at Bell Labs. C++ is regarded as a middle-level language, as it comprises a combination of both high-level and low-level language features. It is a superset of C, and that virtually any legal C program is a legal C++ program. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

**Class and Objects:**

C++ is an object-oriented programming language.

Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

Attributes and methods are basically variables and functions that belongs to the class. These are often referred to as "class members".

A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

**Creating Class:**

class MyClass {       // The class  
  public:             // Access specifier  
    int myNum;        // Attribute (int variable)  
    string myString;  // Attribute (string variable)  
};

* The class keyword is used to create a class called MyClass.
* The public keyword is an access specifier, which specifies that members (attributes and methods) of the class are accessible from outside the class. You will learn more about access specifiers later.
* Inside the class, there is an integer variable myNum and a string variable myString. When variables are declared within a class, they are called attributes.
* At last, end the class definition with a semicolon ;.

**Creating object:**

In C++, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.

To create an object of MyClass, specify the class name, followed by the object name.

To access the class attributes (myNum and myString), use the dot syntax (.) on the object:

class MyClass {       // The class  
  public:             // Access specifier  
    int myNum;        // Attribute (int variable)  
    string myString;  // Attribute (string variable)  
};  
  
int main() {  
  MyClass **myObj**;  // Create an object of MyClass  
  
  // Access attributes and set values  
  **myObj.**myNum = 15;   
  **myObj.**myString = "Some text";  
  
  // Print attribute values  
  cout << myObj.myNum << "\n";  
  cout << myObj.myString;  
  return 0;  
}

**Interface:**

An interface describes the behavior or capabilities of a C++ class without committing to a particular implementation of that class.

The C++ interfaces are implemented using abstract classes and these abstract classes should not be confused with data abstraction which is a concept of keeping implementation details separate from associated data.

A class is made abstract by declaring at least one of its functions as pure virtual function. A pure virtual function is specified by placing "= 0" in its declaration as follows –

class Box {

public:

// pure virtual function

virtual double getVolume() = 0;

private:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

**Inheritance:**

In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

**Base and derived class:**

* An existing class that is "parent" of a new class is called a base class. New class that inherits properties of the base class is called a child class or derived class.
* Inheritance is a technique of code reuse. It also provides possibility to extend existing classes by creating derived classes.

#include <iostream>

using namespace std;

class Account {

public:

float salary = 60000;

};

class Programmer: public Account {

public:

float bonus = 5000;

};

int main(void) {

Programmer p1;

cout<<"Salary: "<<p1.salary<<endl;

cout<<"Bonus: "<<p1.bonus<<endl;

return 0;

}

**Output:**

Salary: 60000

Bonus: 5000

**Pointer:**

Pointer is a variable in C++ that holds the address of another variable. They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and a character type pointer can hold the address of char variable.

**Syntax:**

data\_type \*pointer\_name;

**Example:**

#include <iostream>

using namespace std;

int main(){

//Pointer declaration

int \*p, var=101;

//Assignment

p = &var;

return 0;

}

**Data File:**

Most programs need to save data to disk files and read it back in. Working with disk files requires another set of classes: ifstream for input, fstream for both input and output, and ofstream for output. Objects of these classes can be associated with disk files, and we can use their member functions to read and write to the files.

**Opening and closing files:**

* A file must be opened before you can read from it or write to it.
* Either ofstream or fstream object may be used to open a file for writing. The ifstream object is used to open a file for reading purpose only.
* Following is the standard prototype for open() function, which is a member of fstream, ifstream, and ofstream objects.

*void open(const char \*filename, ios::openmode mode);*

* Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.
* In order to close an opened file, the corresponding ofstream/ifstream/fstream used to open the file should invoke the close() function.

**Example:**

#include<iostream>

#include<fstream>

#include<cstdlib>

using namespace std;

int main()

{

char ch;

ifstream in;

in.open("student.txt",ios::in);

if(in.fail()) //Error handling while opening a file

{

cout<<"Error opening file,exiting..."<<endl;

exit(1);

}

cout<<"Contents of the file:"<<endl;

while(in.get(ch))

{

cout<<ch;

}

in.close();

return 0;

}

**Functions:**

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

**Syntax:**

void myFunction() {  
  // code to be executed  
}

* myFunction() is the name of the function
* void means that the function does not have a return value.
* inside the function (the body), code that defines what the function should do is added

**Calling a function:**

Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are called. To call a function, the function's name is written followed by two parentheses () and a semicolon ;.

**Example:**

// Create a function  
void myFunction() {  
  cout << "I just got executed!";  
}  
  
int main() {  
  **myFunction();** // call the function  
  return 0;  
}

**Output:**

I just got executed!

**Control statements:**

A control statement is used in a programming language to control the flow of the program. They are nothing but a keyword or statements that are used in a program to transfer the flow of control to another statement based on the conditions. Based on the given condition, it evaluates the result and executes the corresponding statements. Control statements are the statement that controls the flow of the program in order to execute the piece of the code using various controls statement like if statement, if-else statement, break statement, continue statement, for loop, while loop, do while loop.

**if-else statement:**

if(*condition*) {

*//code if condition true*

} else {

*//code if condition false*

}

**switch statement:**

switch(*expression*){

case *constant1:*

*//code to be executed if expression is equal to the constant1*

break;

case constant2:

*//code to be executed if expression is equal to the constant2*

break;

default;

*//code to be executed if the expression is not equal to any constant*

}

**Looping:**

Looping is used when a certain block of code needs to run multiple times. There are different types of looping. They are for loop, while loop and do..while loop.

**for loop:**

for(*statement1, statement2, statement3*){

//code to run multiple times

}

**while loop:**

while(*condition*){

*//code run till the condition is false*

}

**do..while loop:**

do{

*//code to run*

} while (*condition);*

The code of do..while loop runs at least once.