|  |  |
| --- | --- |
| **Name** | ADITI RAO |
| **UID no.** | 202220003 |
| **Experiment No.** | 6 |

|  |  |
| --- | --- |
| **AIM:** | **Program on Inheritance: Implement a Program to demonstrate multiple Inheritance** |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Define a class called Home with two data members used to store the number of rooms and the size in square meters. Supply default values for your constructor definition to create a default constructor. In addition to accessor methods, also define the method display(), which outputs the data members of an apartment.  Define another class called Car with data members used to store car number, seating capacity and model name. Add appropriate constructors and member functions. |
| **ALGORITHM:** | CLASS Home  PRIVATE:  num\_rooms: int  size\_sqm: double  PUBLIC:  Home(num\_rooms: int = 0, size\_sqm: double = 0.0)  getNumRooms() const: int  setNumRooms(num\_rooms: int)  getSizeSqM() const: double  setSizeSqM(size\_sqm: double)  display() const  CLASS Car  PRIVATE:  car\_number: string  seating\_capacity: int  model\_name: string  PUBLIC:  Car()  Car(car\_number: string, seating\_capacity: int, model\_name: string)  getCarNumber() const: string  setCarNumber(car\_number: string)  getSeatingCapacity() const: int  setSeatingCapacity(seating\_capacity: int)  getModelName() const: string  setModelName(model\_name: string)  display() const  FUNCTION main()  choice: int  myMotorHome: MotorHome  numRooms: int  size: double  carNumber: string  seatingCapacity: int  modelName: string  category: int  DO  OUTPUT "Hello, welcome to the Motor Home Configuration System!"  OUTPUT "Please choose from the following options:"  OUTPUT "1. Set number of rooms"  OUTPUT "2. Set size in square meters"  OUTPUT "3. Set car number"  OUTPUT "4. Set seating capacity"  OUTPUT "5. Set model name"  OUTPUT "6. Set category"  OUTPUT "7. Display details"  OUTPUT "8. Quit"  OUTPUT "Enter your choice: "  INPUT choice  SWITCH choice  CASE 1:  OUTPUT "Enter number of rooms: "  INPUT numRooms  myMotorHome.setNumRooms(numRooms)  BREAK  CASE 2:  OUTPUT "Enter size in square meters: "  INPUT size  myMotorHome.setSizeSqM(size)  BREAK  CASE 3:  OUTPUT "Enter car number: "  INPUT carNumber  myMotorHome.setCarNumber(carNumber)  BREAK  CASE 4:  OUTPUT "Enter seating capacity: "  INPUT seatingCapacity  myMotorHome.setSeatingCapacity(seatingCapacity)  BREAK  CASE 5:  OUTPUT "Enter model name: "  INPUT modelName  myMotorHome.setModelName(modelName)  BREAK  CASE 6:  OUTPUT "Choose category:"  OUTPUT "1. Luxury"  OUTPUT "2. First Class"  OUTPUT "3. Middle Class"  OUTPUT "4. Economy"  INPUT category  SWITCH category  CASE 1:  myMotorHome.setCategory(MotorHome::LUXURY)  BREAK  CASE 2:  myMotorHome.setCategory(MotorHome::FIRST\_CLASS)  BREAK  CASE 3:  myMotorHome.setCategory(MotorHome::MIDDLE\_CLASS)  BREAK  CASE 4:  myMotorHome.setCategory(MotorHome::ECONOMY)  BREAK  DEFAULT:  OUTPUT "Invalid choice"  BREAK  CASE 7:  myMotorHome.display()  BREAK  CASE 8:  OUTPUT "Exiting program..."  BREAK  DEFAULT:  OUTPUT "Invalid choice"  END SWITCH  WHILE choice != 8  OUTPUT "Thank you for using the Motor Home Configuration System!"  END FUNCTION |
| **PROGRAM:** | #include <iostream>  #include <string>  using namespace std;  class Home  {  private:  int num\_rooms;  double size\_sqm;  public:  Home(int num\_rooms = 0, double size\_sqm = 0.0)  {  this->num\_rooms = num\_rooms;  this->size\_sqm = size\_sqm;  }  int getNumRooms() const  {  return num\_rooms;  }  void setNumRooms(int num\_rooms)  {  this->num\_rooms = num\_rooms;  }  double getSizeSqM() const  {  return size\_sqm;  }  void setSizeSqM(double size\_sqm)  {  this->size\_sqm = size\_sqm;  }  void display() const  {  cout << "Number of rooms: " << num\_rooms << endl;  cout << "Size in square meters: " << size\_sqm << endl;  }  };  class Car  {  private:  string car\_number;  int seating\_capacity;  string model\_name;  public:  Car()  {  car\_number = "";  seating\_capacity = 0;  model\_name = "";  }  Car(string car\_number, int seating\_capacity, string model\_name)  {  this->car\_number = car\_number;  this->seating\_capacity = seating\_capacity;  this->model\_name = model\_name;  }  string getCarNumber() const  {  return car\_number;  }  void setCarNumber(string car\_number)  {  this->car\_number = car\_number;  }  int getSeatingCapacity() const  {  return seating\_capacity;  }  void setSeatingCapacity(int seating\_capacity)  {  this->seating\_capacity = seating\_capacity;  }  string getModelName() const  {  return model\_name;  }  void setModelName(string model\_name)  {  this->model\_name = model\_name;  }  void display() const  {  cout << "Car number: " << car\_number << endl;  cout << "Seating capacity: " << seating\_capacity << endl;  cout << "Model name: " << model\_name << endl;  }  };  class MotorHome : public Car, public Home  {  public:  enum CATEGORY {LUXURY, FIRST\_CLASS, MIDDLE\_CLASS, ECONOMY};  CATEGORY category;  public:  MotorHome(int num\_rooms = 0, double size\_sqm = 0.0, string car\_number = "", int seating\_capacity = 0, string model\_name = "", CATEGORY category = ECONOMY)  : Car(car\_number, seating\_capacity, model\_name), Home(num\_rooms, size\_sqm)  {  this->category = category;  }  void setCategory(CATEGORY category)  {  this->category = category;  }  string getCategory() const  {  switch(category)  {  case LUXURY:  return "Luxury";  case FIRST\_CLASS:  return "First Class";  case MIDDLE\_CLASS:  return "Middle Class";  case ECONOMY:  return "Economy";  default:  return "";  }  }  void display() const  {  Car::display();  Home::display();  cout << "Category: " << getCategory() << endl;  }  };  int main()  {  int choice;  MotorHome myMotorHome;  int numRooms;  double size;  string carNumber;  int seatingCapacity;  string modelName;  int category;  do  {  cout << "Hello, welcome to the Motor Home Configuration System!" << endl;  cout << "Please choose from the following options:" << endl;  cout << "1. Set number of rooms" << endl;  cout << "2. Set size in square meters" << endl;  cout << "3. Set car number" << endl;  cout << "4. Set seating capacity" << endl;  cout << "5. Set model name" << endl;  cout << "6. Set category" << endl;  cout << "7. Display details" << endl;  cout << "8. Quit" << endl;  cout << "Enter your choice: ";  cin >> choice;  switch(choice)  {  case 1:  cout << "Enter number of rooms: ";  cin >> numRooms;  myMotorHome.setNumRooms(numRooms);  break;  case 2:  cout << "Enter size in square meters: ";  cin >> size;  myMotorHome.setSizeSqM(size);  break;  case 3:  cout << "Enter car number: ";  cin >> carNumber;  myMotorHome.setCarNumber(carNumber);  break;  case 4:  cout << "Enter seating capacity: ";  cin >> seatingCapacity;  myMotorHome.setSeatingCapacity(seatingCapacity);  break;  case 5:  cout << "Enter model name: ";  cin >> modelName;  myMotorHome.setModelName(modelName);  break;  case 6:  cout << "Choose category:" << endl;  cout << "1. Luxury" << endl;  cout << "2. First Class" << endl;  cout << "3. Middle Class" << endl;  cout << "4. Economy" << endl;  cin >> category;  switch(category)  {  case 1:  myMotorHome.setCategory(MotorHome::LUXURY);  break;  case 2:  myMotorHome.setCategory(MotorHome::FIRST\_CLASS);  break;  case 3:  myMotorHome.setCategory(MotorHome::MIDDLE\_CLASS);  break;  case 4:  myMotorHome.setCategory(MotorHome::ECONOMY);  break;  default:  cout << "Invalid choice" << endl;  }  break;  case 7:  myMotorHome.display();  break;  case 8:  cout << "Exiting program..." << endl;  break;  default:  cout << "Invalid choice" << endl;  }  }  while(choice != 8);  cout << "Thank you for using the Motor Home Configuration System!" << endl;  return 0;  } |
| **RESULT:** |  |
| **Program 2** | |
| **PROBLEM STATEMENT:** | Imagine a publishing company that markets both book and audiocassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. For this class consider two base classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book , tape and publication classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata(). |
| **ALGORITHM:** | CLASS Publication  PROTECTED:  title: string  price: float  PUBLIC:  FUNCTION getData()  OUTPUT "Kindly enter the title: "  INPUT title  OUTPUT "Kindly enter the price: "  INPUT price  FUNCTION putData()  OUTPUT "Title: " + title  OUTPUT "Price: " + price  END CLASS  CLASS Book INHERITS Publication  PRIVATE:  pageCount: int  PUBLIC:  FUNCTION getData()  CALL Publication.getData()  OUTPUT "Kindly enter the playing time (in minutes): "  INPUT playingTime  FUNCTION putData()  CALL Publication.putData()  OUTPUT "Playing Time: " + playingTime + " minutes"  END CLASS  FUNCTION main()  DECLARE book: Book  DECLARE tape: Tape  OUTPUT "Enter book details:"  CALL book.getData()  OUTPUT "Enter tape details:"  CALL tape.getData()  OUTPUT "Book details:"  CALL book.putData()  OUTPUT "Tape details:"  CALL tape.putData()  RETURN 0  ENDFUNCTION |
| **PROGRAM:** | #include <iostream>  #include <string>  using namespace std;  class Publication  {  protected:  string title;  float price;  public:  void getData()  {  cout << "Kindly enter the title: ";  getline(cin, title);  cout << "Kindly enter the price: ";  cin >> price;  cin.ignore();  }  void putData()  {  cout << "Title: " << title << endl;  cout << "Price: " << price << endl;  }  };  class Book : public Publication  {  private:  int pageCount;  public:  void getData()  {  Publication::getData();  cout << "Kindly enter the page count: ";  cin >> pageCount;  cin.ignore(); // Ignore the newline character  }  void putData()  {  Publication::putData(); // Call base class function  cout << "Page Count: " << pageCount << endl;  }  };  class Tape : public Publication  {  private:  float playingTime;  public:  void getData()  {  Publication::getData(); // Call base class function  cout << "Kindly enter the playing time (in minutes): ";  cin >> playingTime;  cin.ignore(); // Ignore the newline character  }  void putData()  {  Publication::putData(); // Call base class function  cout << "Playing Time: " << playingTime << " minutes" << endl;  }  };  int main()  {  Book book;  Tape tape;  cout << "Enter book details:\n";  book.getData();  cout << "\nEnter tape details:\n";  tape.getData();  cout << "\nBook details:\n";  book.putData();  cout << "\nTape details:\n";  tape.putData();  return 0;  } |
| **RESULT:** |  |
|  |  |
| **PROBLEM STATEMENT:** | WAP to find log of numbers with log10, loge, log2.  Create an abstract class Logarithm with a method log(number).  Create classes Log10, LogE, Log2 which inherits from Logarithm and overrides method log(number) and return the log of that number. |
| **ALGORITHM** | CLASS Logarithm  ABSTRACT FUNCTION calcLog(number: double): double  CLASS Log10 EXTENDS Logarithm  FUNCTION calcLog(number: double): double  RETURN log10(number)  CLASS LogE EXTENDS Logarithm  FUNCTION calcLog(number: double): double  RETURN log(number)  CLASS Log2 EXTENDS Logarithm  FUNCTION calcLog(number: double): double  RETURN log2(number)  FUNCTION main()  number: double  logObj: pointer to Logarithm  result1, result2, result3: double  number <- 100.0  logObj <- new Log10()  result1 <- logObj->calcLog(number)  OUTPUT "Log10(" + number + ") = " + result1  logObj <- new LogE()  result2 <- logObj->calcLog(number)  OUTPUT "LogE(" + number + ") = " + result2  logObj <- new Log2()  result3 <- logObj->calcLog(number)  OUTPUT "Log2(" + number + ") = " + result3  RETURN 0 |
| **PROGRAM:** | #include <iostream>  #include <math.h>  using namespace std;  class Logarithm  {  public:  virtual double calcLog(double number) = 0;  };  class Log10 : public Logarithm  {  public:  double calcLog(double number)  {  return log10(number);  }  };  class LogE : public Logarithm  {  public:  double calcLog(double number)  {  return log(number);  }  };  class Log2 : public Logarithm  {  public:  double calcLog(double number)  {  return log2(number);  }  };  int main()  {  double number = 100.0;  Logarithm\* logObj;  logObj = new Log10();  double result1 = logObj->calcLog(number);  cout << "Log10(" << number << ") = " << result1 << endl;  logObj = new LogE();  double result2 = logObj->calcLog(number);  cout << "LogE(" << number << ") = " << result2 << endl;  logObj = new Log2();  double result3 = logObj->calcLog(number);  cout << "Log2(" << number << ") = " << result3 << endl;  return 0;  } |
| **RESULT:** |  |
|  |  |