|  |  |
| --- | --- |
| **Name** | ADITI RAO |
| **UID no.** | 202220003 |
| **Experiment No.** | 3 |

|  |  |
| --- | --- |
| **AIM:** | **Program on Polymorphism: Implement a Program to demonstrate constructor overloading** |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Write a menu-driven program to recruit an employee (depending on his performance in various rounds) in some software company using constructor overloading.  Selection Criteria for each post is given below:  i) Programmer (Minimum total of 80 marks):-  Rounds:-  (1) Course Work  (2) Aptitude Test  (3) Technical Test  (4) Interview  ii) Team Leader (Minimum total of 85 marks ):-  Rounds:-  (1) Technical Test  (2) Interview  iii) Project Manager (Minimum score 90 marks)  Rounds:-  (1) Interview  Create a class Posting and write 3 constructors to initialize the object and set the parameters  and display the employee post according to selection criteria.  Data members:  ● int courseWork;  ● int AptTest;  ● int TechTest;  ● int interview;  Methods:  ● Posting (int courseWork, int AptTest, int TechTest,int interview)  ● Posting (int TechTest,int interview)  ● Posting (int interview)  Make use of ‘this’ keyword. |
| **ALGORITHM:** | 1. START 2. DEFINE a class "Posting" with private members "courseWork", "AptTest", "TechTest", and "interview". 3. DECLARE a constructor with no parameters for initializing the data members to 0. 4. DECLARE a parameterized constructor for a Programmer that initializes the data members with input values. 5. DECLARE a parameterized constructor for a Team Leader that initializes the "TechTest" and "interview" data members with input values. 6. DECLARE an explicit parameterized constructor for a Project Manager that initializes the "interview" data member with input values. 7. DEFINE a member function "displayPost" that takes "choice" as input, calculates the total marks, and checks if the employee is recruited or not. 8. DEFINE the main function. 9. PRINT a welcome message and declare an object "emp" of class "Posting". 10. DECLARE variables "choice", "courseWork", "AptTest", "TechTest", and "interview". 11. PRINT a message to select the post for recruitment. 12. READ the "choice" variable from the user. 13. START a loop until "choice" is 0. 14. SWITCH "choice". 15. CASE 1: 16. a. PRINT a message to input marks for the programmer. 17. b. READ "courseWork", "AptTest", "TechTest", and "interview" values. 18. c. CREATE an object "emp" of class "Posting" using the Programmer parameterized constructor. 19. d. CALL the "displayPost" function of the object "emp". 20. CASE 2: 21. a. PRINT a message to input marks for the team leader. 22. b. READ "TechTest" and "interview" values. 23. c. CREATE an object "emp" of class "Posting" using the Team Leader parameterized constructor. 24. d. CALL the "displayPost" function of the object "emp". 25. CASE 3: 26. a. PRINT a message to input marks for the project manager. 27. b. READ "interview" value. 28. c. CREATE an object "emp" of class "Posting" using the Project Manager parameterized constructor. 29. d. CALL the "displayPost" function of the object "emp". 30. DEFAULT: 31. a. PRINT an error message for an incorrect input. 32. PRINT a message to exit the loop. 33. READ the "choice" variable from the user. 34. END LOOP. 35. END. |
| **PROGRAM:** | #include <iostream>  using namespace std;  class Posting  {  private:  int courseWork;  int AptTest;  int TechTest;  int interview;  public:  Posting() : courseWork(0), AptTest(0), TechTest(0), interview(0){}  Posting(int courseWork, int AptTest, int TechTest, int interview) //Parameterized constructor for a Programmer  {  this->courseWork = courseWork;  this->AptTest = AptTest;  this->TechTest = TechTest;  this->interview = interview;  }  Posting(int TechTest,int interview) //Parameterized constructor for a Team Leader  {  this->courseWork = 0;  this->AptTest = 0;  this->TechTest = TechTest;  this->interview = interview;  }  explicit Posting(int interview) //Parameterized constructor for a Project Manager  {  this->courseWork = 0;  this->AptTest = 0;  this->TechTest = 0;  this->interview = interview;  }  void displayPost(int choice)  {  int totalMarks = courseWork + AptTest + TechTest + interview;  if (totalMarks >= 80 && courseWork >= 0 && AptTest >= 0 && TechTest >= 0 && interview >= 0)  {  if (totalMarks >= 90 && choice == 3)  cout << "\nCongratulations! The Employee selected for the post of Project Manager" << endl;  else if (totalMarks >= 85 && choice == 2)  cout << "\nCongratulations! The Employee selected for the post of Team Leader" << endl;  else if (choice == 1)  cout << "\nCongratulations, the Employee selected for the post of Programmer" << endl;  else  cout << "\nSorry, but the employee has not been recruited to their desired post. "<< endl;  }  else  cout << "\nSorry, but the employee has not been recruited to their desired post." << endl;  }  };  int main()  {  cout << "This program was brought to you by Aditi Rao." << endl;  Posting emp;  int choice, courseWork, AptTest, TechTest, interview;  cout << "This is a program that allows for employee recruitment. \nBased on the post to which the employee is to be recruited to, please make your choice. \n1. Programmer.\n2. Team Leader\n3. Project Manager.\n" << endl;  cin >> choice;  int i = 0;  while (choice != 0)  {  switch(choice)  {  case 1:  cout << "\nTo recruit a programmer, you must input the marks of 4 tests." << endl;  cout << "Kindly enter marks obtained in Course Work: ";  cin >> courseWork;  cout << "Kindly enter marks obtained in Aptitude Test: ";  cin >> AptTest;  cout << "Kindly enter marks obtained in Technical Test: ";  cin >> TechTest;  cout << "Kindly enter marks obtained in Interview: ";  cin >> interview;  {  Posting emp(courseWork, AptTest, TechTest, interview);  emp.displayPost(choice);  }  break;  case 2:  cout << "\nTo recruit a team leader, you must input the marks of 2 tests." << endl;  cout << "Enter marks obtained in Technical Test: ";  cin >> TechTest;  cout << "Enter marks obtained in Interview: ";  cin >> interview;  {  Posting emp(TechTest, interview);  emp.displayPost(choice);  }  break;  case 3:  cout << "\nTo recruit a project manager, you must input the marks of 1 interview." << endl;  cout << "Enter marks obtained in the Interview." << endl;  cin >> interview;  {  Posting emp(interview);  emp.displayPost(choice);  }  break;  default:  cout << "\nYou've given incorrect input. Please, try again." << endl;  }  cout << "\nKindly enter 0 if you wish to exit the loop.\n " << endl;  cin >> choice;  };  } |
| **RESULT:** |  |
| **Program 2** | |
| **PROBLEM STATEMENT:** | Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two argument is used to initialized real and imag to two different values. |
| **ALGORITHM:** | 1. BEGIN 2. CLASS ComplexNumber 3. DECLARE private integer real, imag 4. FUNCTION ComplexNumber() 5. real <- 0 6. imag <- 0 7. FUNCTION ComplexNumber(int value) 8. real <- value 9. imag <- 0 10. FUNCTION ComplexNumber(int realValue, int imagValue) 11. real <- realValue 12. imag <- imagValue 13. FUNCTION operator+(const ComplexNumber& other) 14. CREATE ComplexNumber sum 15. sum.real <- real + other.real 16. sum.imag <- imag + other.imag 17. RETURN sum 18. FUNCTION display() 19. OUTPUT real + imagi 20. END CLASS 21. DECLARE integer value, real\_value, imaginary\_value 22. CREATE ComplexNumber num1, num2, num3, num4 23. OUTPUT "This program was brought to you by Aditi Rao." 24. // First Constructor - Takes no argument 25. OUTPUT "Complex Number 1:" 26. num1.display() 27. // Second Constructor - Takes one argument 28. OUTPUT "Enter a value for Complex Number 2: " 29. INPUT value 30. CREATE ComplexNumber num2(value) 31. OUTPUT "Complex Number 2:" 32. num2.display() 33. // Third Constructor - Takes two arguments 34. OUTPUT "Enter real part for Complex Number 3: " 35. INPUT real\_value 36. OUTPUT "Enter imaginary part for Complex Number 3: " 37. INPUT imaginary\_value 38. CREATE ComplexNumber num3(real\_value, imaginary\_value) 39. OUTPUT "Complex Number 3:" 40. num3.display() 41. // Addition of two complex numbers 42. CREATE ComplexNumber num4 43. num4 <- num2 + num3 44. OUTPUT "Sum of Complex Number 2 and 3:" 45. num4.display() 46. END |
| **PROGRAM:** | #include <iostream>  using namespace std;  class ComplexNumber  {  private:  int real;  int imag;  public:  // First Constructor - Takes no argument  ComplexNumber()  {  real = 0;  imag = 0;  }  // Second Constructor - Takes one argument  ComplexNumber(int value)  {  real = value;  imag = 0;  }  // Third Constructor - Takes two arguments  ComplexNumber(int realValue, int imagValue)  {  real = realValue;  imag = imagValue;  }  /\*What follows is an overloaded addition operator function for the ComplexNumber class.  operator+ is the name of the overloaded addition operator.  It specifies that this function will be called when the + operator is used on objects of the ComplexNumber class.\*/  ComplexNumber operator+(const ComplexNumber& other)  {  ComplexNumber sum;  sum.real = real + other.real;  sum.imag = imag + other.imag;  return sum;  }  // Display the complex number  void display()  {  cout << real << " + " << imag << "i" <<endl;  }  };  int main()  {  cout << "This program was brought to you by Aditi Rao. " << endl;  int value, real\_value, imaginary\_value;  // Constructor Overloading  // First Constructor - Takes no argument  ComplexNumber num1;  cout << "Complex Number 1: " << endl;  num1.display();  // Second Constructor - Takes one argument  cout << "Enter a value for Complex Number 2: ";  cin >> value;  ComplexNumber num2(value);  cout << "Complex Number 2: " << endl;  num2.display();  // Third Constructor - Takes two arguments  cout << "Enter real part for Complex Number 3: ";  cin >> real\_value;  cout << "Enter imaginary part for Complex Number 3: ";  cin >> imaginary\_value;  cout << "Complex Number 3: " << endl;  ComplexNumber num3(real\_value, imaginary\_value);  num3.display();  // Addition of two complex numbers  ComplexNumber num4 = num2 + num3;  cout << "Sum of Complex Number 2 and 3:" << endl;  num4.display();  return 0;  } |
| **RESULT:** |  |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Write a program for finding area of different geometric shapes (circle, rectangle, cuboid). Use constructor overloading with type, order, sequence of arguments to find the area of shapes. |
| **ALGORITHM:** | 1. CLASS Geometry 2. PRIVATE:   radius  length  breadth  height   1. PUBLIC:   FUNCTION Geometry() // First Constructor - Takes no arguments  SET radius TO 0  SET length TO 0  SET breadth TO 0  SET height TO 0  END FUNCTION   1. FUNCTION Geometry(r) // Second Constructor - Takes radius as argument   SET radius TO r  SET length TO 0  SET breadth TO 0  SET height TO 0  END FUNCTION     1. FUNCTION Geometry(l, b) // Third Constructor - Takes length and breadth as arguments   SET radius TO 0  SET length TO l  SET breadth TO b  SET height TO 0  END FUNCTION     1. FUNCTION Geometry(l, b, h) // Fourth Constructor - Takes length, breadth and height as arguments   SET radius TO 0  SET length TO l  SET breadth TO b  SET height TO h  END FUNCTION     1. FUNCTION area()   SET area TO 0  IF radius IS NOT 0  SET area TO PI \* radius \* radius  ELSE IF height IS EQUAL TO 0  SET area TO length \* breadth  ELSE IF height IS NOT EQUAL TO 0  SET area TO 2 \* ((length \* breadth)+(height \* breadth)+(length \* height))  END IF  RETURN area  END FUNCTION     1. FUNCTION display()   DISPLAY "The area of the given object is: " CONCATENATED WITH area()  END FUNCTION  END CLASS  FUNCTION main()   1. DISPLAY "This program was brought to you by Aditi Rao, 2022200003." 2. DECLARE choice AS INTEGER 3. DISPLAY "Press the listed numbers to get areas of: \n1. Circle\n2. Rectangle\n3. Cuboid\n" 4. GET choice FROM USER 5. DECLARE r, l, b, h AS FLOAT 6. SWITCH choice 7. CASE 1: 8. DISPLAY "Kindly enter the radius of the circle. " 9. GET r FROM USER 10. Geometry circle(r) 11. DISPLAY "Area of a circle is: " 12. CALL circle.display() 13. BREAK GOTO STEP 39 14. CASE 2: 15. DISPLAY "Kindly enter the length and breadth of the rectangle. " 16. GET l, b FROM USER 17. Geometry rectangle(l, b) 18. DISPLAY "Area of the rectangle is: " 19. CALL rectangle.display() 20. BREAK GOTO STEP 39 21. CASE 3: 22. DISPLAY "Kindly enter the length, breadth and height of the cuboid. " 23. GET l, b, h FROM USER 24. Geometry cuboid(l, b, h) 25. DISPLAY "Area of the cuboid is: " 26. CALL cuboid.display() 27. BREAK GOTO STEP 39 28. DEFAULT: 29. DISPLAY "Incorrect Input. " 30. END SWITCH 31. DISPLAY "To exit the loop, press 0." 32. GET choice FROM USER 33. WHILE choice NOT EQUAL TO 0, GOTO step 14 34. END |
| **PROGRAM:** | #include <iostream>  #include <cmath>  using namespace std;  class Geometry  {  private:  float radius;  float length;  float breadth;  float height;  public:  //First Constructor - Takes no arguments  Geometry()  {  radius = 0;  length = 0;  breadth = 0;  height = 0;  }  //Second Constructor - Takes radius as argument  Geometry (float r)  {  radius = r;  length = 0;  breadth = 0;  height = 0;  }  //Third Constructor - Takes length and breadth as arguments  Geometry(float l, float b)  {  radius = 0;  length = l;  breadth = b;  height = 0;  }  //Fourth Constructor - Takes length, breadth and height as arguments  Geometry(float l, float b, float h)  {  radius = 0;  length = l;  breadth = b;  height = h;  }  //calculate area  float area()  {  float area;  if (radius != 0)  area = M\_PI \* radius \* radius;  else if (height == 0)  area = length \* breadth;  else if (height != 0)  area = 2 \* ((length \* breadth)+(height \* breadth)+(length \* height));  return area;  }  //Display Area of the given shape  void display()  {  cout << "The area of the given object is: " << area() << endl;  }  };  int main()  {  cout << "This program was brought to you by Aditi Rao, 2022200003." << endl;  int choice;  cout << "Press the listed numbers to get areas of: \n1. Circle\n2. Rectangle\n3. Cuboid\n" << endl;  cin >> choice;  float r, l, b, h;  do  {  switch (choice)  {  case 1:  {  cout << "Kindly enter the radius of the circle. " << endl;  cin >> r;  Geometry circle(r);  cout << "Area of a circle is: ";  circle.display();  break;  }  case 2:  {  cout << "Kindly enter the length and breadth of the rectangle. " << endl;  cin >> l >> b;  Geometry rectangle(l, b);  cout << "Area of the rectangle is: ";  rectangle.display();  break;  }  case 3:  {  cout << "Kindly enter the length, breadth and height of the cuboid. " << endl;  cin >> l >> b >> h;  Geometry cuboid(l, b, h);  cout << "Area of the cuboid is: ";  cuboid.display();  break;  }  default:  cout << "Incorrect Input. " << endl;  }  cout << "To exit the loop, press 0." << endl;  cin >> choice;  }  while (choice != 0);  return 0;  } |
| **RESULT:** |  |
| **Conclusion** | In conclusion, the programs implemented to demonstrate constructor overloading in C++ highlight the versatility and flexibility of polymorphism. By utilizing constructors with different parameter lists in a class, the programs showcase how objects of the same class can be created and initialized in multiple ways. This allows for flexibility in object creation, enabling the use of different constructor signatures based on the requirements of the program. The demonstration of constructor overloading emphasizes the power of polymorphism in enabling code reusability and extensibility, allowing for efficient and flexible object-oriented programming in C++. |