|  |
| --- |
| **Assignment # 11**  ***Session*: Spring 2022 *Total marks*: 100**  ***Name*** : ***\_\_NIMRA MAQBOOL\_\_ Roll number* : \_BSCE21012\_** |

***Submission:***

• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• Submission after due time will not be accepted.*

**There should be a Report explaining your code and highlighting results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSCE21001\_Assignment11.pdf.**

**Note:** From this assignment onwards, you will create default and parameterized constructor/s and destructor for every class.

**TASK 1:**

You are going to make an application for drawing different Geometrical shapes. You should break this application into objects and classes. In this Question, you are going to make one base class by the name Shape. Shape will have Surface Area (SA), Volume(V), Pi and R as attributes. Make these attributes private. This class will have two virtual member functions(public) set/get Volume (parameters will be Pi and R) and set/get Surface Area (Parameters will be Pi and R). You will have to write default constructor and destructor for this class as well.

|  |
| --- |
| **Function.h:**  class Shapes { private:  float surfaceArea;  float volume; //declaring variables(private members)  float pi;  float R; public:  Shapes() {  cout << "DEFAULT CONSTRUCTOR OF SHAPES.." << endl;  surfaceArea = 0.00; //setting values in default constructor  volume = 0.00;  pi = 0.00;  R = 0.00;  }   Shapes(float Pi, float r, float Volume, float SurfaceArea) {  cout<<"PARAMETRIZED CONSTRUCTOR OF SHAPES IS CALLED.."<<endl;  pi = Pi;  R = r;  volume = Volume; //setting values in parametrized constructor  surfaceArea = SurfaceArea;  }   virtual void setVolume(float r) {  const float Pi = 3.14; //copying values  pi = Pi;  cout << "ENTER VALUE OF R FOR VOLUME = ";  cin >> r; //taking input  R = r;  }   virtual float getVolume() {  return volume; //getting surface area  }   virtual float getSurfaceArea() {  return surfaceArea; //getting surface area  }   virtual void setSurfaceArea(float r) {  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA = "; //taking input  cin >> r;  R = r;  }   virtual void display() {  cout << "SURFACE AREA = " << surfaceArea << endl;  cout << "VOLUME = " << volume << endl; //displaying  }   ~Shapes() {  cout << "destructor of shapes is called.." << endl; //making destructor  }  }; |

**TASK 2:**

In this task, you will make a class Sphere. Use public as access specifier and inherit form Shape. This class will have its own following protected attributes Pi, R, SA and V (Why aren’t we reusing the shape class attributes?). Override the set Volume and Surface Area member functions from Shape class with same parameters but with different implementations to calculate the Volume and Surface Area. You will also implement setters and getters for this class along with constructor and destructor.

Formulas:

SA = 4 \* Pi \* R \* 22

V= 4/3 \* Pi \* R3

|  |
| --- |
| **Function.h:**  class sphere : public Shapes { protected:  float surfaceArea; //declaring protected data members  float volume;  float pi;  float R; public:  sphere() {  cout << "DEFAULT CONSTRUCTOR OF SPHERE.." << endl;  surfaceArea = 0.00; //setting values in default constructor  volume = 0.00;  pi = 0.00;  R = 0.00;  }   sphere(float Pi, float r, float Volume, float SurfaceArea) {  cout<<"PARAMETRIZED CONSTRUCTOR OF SPHERE IS CALLED.."<<endl;  pi = Pi;  R = r;  volume = Volume; //copying values in parametrized constructor  surfaceArea = SurfaceArea;  }   void setVolume(float r) {  const float Pi = 3.14; //copying  pi = Pi;  cout << "ENTER VALUE OF R FOR VOLUME = ";  cin >> r; //taking input  R = r;  float v;  v = (4 / 3 \* (Pi \* (R \* R \* R))); //calculating volume  volume = v; //copying  }   void setSurfaceArea(float r) {  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA= ";  cin >> r; //taking input  R = r;  float sA;  sA = 4 \* Pi \* R \* 22; //calculating surface area  surfaceArea = sA; //copying  }   float getVolume() {  return volume; //getting surface area  }   float getSurfaceArea() {  return surfaceArea; //getting volume  }   void display() {  cout << "SURFACE AREA = " << surfaceArea << endl; //displaying surface area and volume  cout << "VOLUME = " << volume << endl;  }   ~sphere() {  cout << "destructor of sphere is called.." << endl; //calling destructor  } }; |

**TASK 3:**

In this Question, you will make a class Cone. Use public as access specifier and inherit from Sphere instead of Shape. This class will have two private attributes s and h (Why are we reusing Sphere class’s attributes) . Now override the functions of SA and V along with setters/getters of its own. Notice that SA and V will have different parameters this time so will you be able to override them without declaring a virtual method?

**Formulas:**

**SA = Pi \* R \* S + Pi \* R2**

**V = 1/3 \* Pi \* R2 \* H**

You must call these overridden functions by the object of shape. Your code must look like given below:

**Shape \*s;**

**Sphere sp;**

**s= &sp;**

**s->set\_SA(parameters)**

Execute the above code before and after using the virtual keyword with class members and report what difference do you see after using the virtual keyword.

//we know A C++ virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword. It is used to tell the compiler to perform dynamic linkage or late binding on the function. So by using virtual keyword me can access the function with the same name in parent and derived class.

|  |
| --- |
| **Function.h:**  class cone : public sphere { private:  float s;  float h; public:  cone() {  cout<<"DEFAULT CONSTRUCTOR OF CONE IS CALLED.."<<endl;  s = 0.00;  h = 0.00; //making default constructor and setting values to zero  }   cone(float s1, float h1) {  cout<<"PARAMETRIZED CONSTRUCTOR OF CONE IS CALLED.."<<endl;  s = s1;  h = h1; //copying values in parametrized constructor  }   void setSurfaceArea(float S) {  float r;  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA = ";  cin >> r; //taking input  R = r; //copying  cout << "ENTER S FOR SURFACE AREA= ";  cin >> S; //taking input  s = S; //copying  float SA;  SA = (pi \* R \* s + pi \* R \* R); //calculating surface area  surfaceArea = SA; //copying  }   void setVolume(float H) {  float r;  cout << "ENTER H FOR VOLUME = ";  cin >> H; //taking input  h = H; //copying  const float Pi = 3.14;  pi = Pi; //copying  float v;  v = (pi \* R \* R \* H); //calculating volume  float v1;  v1 = v / 3;  volume = v1; //copying  }   float getSurfaceArea() {  return surfaceArea; //getting surface area  }   float getVolume() {  return volume; //getting volume  }   void display() {  cout << "SURFACE AREA = " << surfaceArea << endl;  cout << "VOLUME = " << volume << endl; //displaying surface area and volume  }   ~cone() {  cout << "destructor of cone is called.." << endl; //destructor is called  } };  #endif //INC\_2022\_SPRING\_CE\_OOP\_WEEK11\_ASSIGNMENT11\_BSCE21012\_FUNCTIONS\_H   * **In function.h I have made 3 classes, shape, sphere, and the cone** * **In shape I have made a default, parametrized constructor and getter, setters, and a display function.** * **In sphere class I have inherited shape class and made default, parametrized constructor getter and setter and a display function.** * **In cone class I have inherited the sphere class and then getter setters and a destructor.**   **main.cpp:**  #include <iostream> #include "Functions.h"  using namespace std;  int main() {  int opt;  do {  cout << endl;  cout << "1.CALL SHAPE CLASS." << endl;  cout << "2.CALL SPHERE CLASS." << endl; //displaying options  cout << "3.CALL CONE CLASS." << endl;  cout << "4.EXIT." << endl;  cin >> opt; //taking input  if (opt == 1) {  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  Shapes s2(3.14, 0, 0, 0); //calling parametrized constructor  cout << endl;  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  }  else if (opt == 2) {  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  Shapes \*S; //making pointer object  sphere sp; //making object  float Volume;  float SurfaceArea;  float Pi; //declaring  float r;  sphere S2(Pi, r, Volume,  SurfaceArea); //calling parametrized constructor  S = &sp; //storing the address in the object  S->setSurfaceArea(r); //calling functions  S->setVolume(r);  S->getSurfaceArea();  S->getVolume();  S->display();  cout << endl;  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  }  else if (opt == 3) {  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  sphere \*sp; //making pointer object  cone c; //making object  float s1;  float h1;  float S; //declaring object  float H;  cone c1(s1, h1);  sp = &c; //storing address  sp->setSurfaceArea(S);  sp->setVolume(H); //calling  sp->getVolume();  sp->getSurfaceArea();  sp->display();  cout << endl;  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  }  else if (opt == 4) {  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  cout << "CHOOSE TO EXIT.." << endl; //displaying  cout << endl;  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  exit(3);  }  else{  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  cout<<"YOU HAVE ENTERED INVALID ARGUMENT.."<<endl;  exit(4);  cout << endl;  cout << "-----------------------------------------------------------------------------" << endl;  cout << endl;  }  } while (opt >= 1 && opt <= 4); //condition for do while loop that if opt is greater than 4 or less than 1 then exit it  return 0; }   * **In main I have just made some objects and pointer object stored the address and then called the functions.**   **Output:**  **Text  Description automatically generated**  **Text  Description automatically generated**  **if we add the virtual key word in the sphere class still the output remains same.**  **Function.h:**  // // Created by Lenovo on 5/26/2022. //  #ifndef INC\_2022\_SPRING\_CE\_OOP\_WEEK11\_ASSIGNMENT11\_BSCE21012\_FUNCTIONS\_H #define INC\_2022\_SPRING\_CE\_OOP\_WEEK11\_ASSIGNMENT11\_BSCE21012\_FUNCTIONS\_H  #include <iostream>  using namespace std;  class Shapes { private:  float surfaceArea;  float volume; //declaring variables(private members)  float pi;  float R; public:  Shapes() {  cout << "DEFAULT CONSTRUCTOR OF SHAPES.." << endl;  surfaceArea = 0.00; //setting values in default constructor  volume = 0.00;  pi = 0.00;  R = 0.00;  }   Shapes(float Pi, float r, float Volume, float SurfaceArea) {  cout<<"PARAMETRIZED CONSTRUCTOR OF SHAPES IS CALLED.."<<endl;  pi = Pi;  R = r;  volume = Volume; //setting values in parametrized constructor  surfaceArea = SurfaceArea;  }   virtual void setVolume(float r) {  const float Pi = 3.14; //copying values  pi = Pi;  cout << "ENTER VALUE OF R FOR VOLUME = ";  cin >> r; //taking input  R = r;  }   virtual float getVolume() {  return volume; //getting surface area  }   virtual float getSurfaceArea() {  return surfaceArea; //getting surface area  }   virtual void setSurfaceArea(float r) {  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA = "; //taking input  cin >> r;  R = r;  }   virtual void display() {  cout << "SURFACE AREA = " << surfaceArea << endl;  cout << "VOLUME = " << volume << endl; //displaying  }   ~Shapes() {  cout << "destructor of shapes is called.." << endl; //making destructor  }  };  class sphere : public Shapes { protected:  float surfaceArea; //declaring protected data members  float volume;  float pi;  float R; public:  sphere() {  cout << "DEFAULT CONSTRUCTOR OF SPHERE.." << endl;  surfaceArea = 0.00; //setting values in default constructor  volume = 0.00;  pi = 0.00;  R = 0.00;  }   sphere(float Pi, float r, float Volume, float SurfaceArea) {  cout<<"PARAMETRIZED CONSTRUCTOR OF SPHERE IS CALLED.."<<endl;  pi = Pi;  R = r;  volume = Volume; //copying values in parametrized constructor  surfaceArea = SurfaceArea;  }   virtual void setVolume(float r) {  const float Pi = 3.14; //copying  pi = Pi;  cout << "ENTER VALUE OF R FOR VOLUME = ";  cin >> r; //taking input  R = r;  float v;  v = (4 / 3 \* (Pi \* (R \* R \* R))); //calculating volume  volume = v; //copying  }   virtual void setSurfaceArea(float r) {  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA= ";  cin >> r; //taking input  R = r;  float sA;  sA = 4 \* Pi \* R \* 22; //calculating surface area  surfaceArea = sA; //copying  }   virtual float getVolume() {  return volume; //getting surface area  }   virtual float getSurfaceArea() {  return surfaceArea; //getting volume  }   virtual void display() {  cout << "SURFACE AREA = " << surfaceArea << endl; //displaying surface area and volume  cout << "VOLUME = " << volume << endl;  }   ~sphere() {  cout << "destructor of sphere is called.." << endl; //calling destructor  } };  class cone : public sphere { private:  float s;  float h; public:  cone() {  cout<<"DEFAULT CONSTRUCTOR OF CONE IS CALLED.."<<endl;  s = 0.00;  h = 0.00; //making default constructor and setting values to zero  }   cone(float s1, float h1) {  cout<<"PARAMETRIZED CONSTRUCTOR OF CONE IS CALLED.."<<endl;  s = s1;  h = h1; //copying values in parametrized constructor  }   void setSurfaceArea(float S) {  float r;  const float Pi = 3.14;  pi = Pi; //copying  cout << "ENTER VALUE OF R FOR SURFACE AREA = ";  cin >> r; //taking input  R = r; //copying  cout << "ENTER S FOR SURFACE AREA= ";  cin >> S; //taking input  s = S; //copying  float SA;  SA = (pi \* R \* s + pi \* R \* R); //calculating surface area  surfaceArea = SA; //copying  }   void setVolume(float H) {  float r;  cout << "ENTER H FOR VOLUME = ";  cin >> H; //taking input  h = H; //copying  const float Pi = 3.14;  pi = Pi; //copying  float v;  v = (pi \* R \* R \* H); //calculating volume  float v1;  v1 = v / 3;  volume = v1; //copying  }   float getSurfaceArea() {  return surfaceArea; //getting surface area  }   float getVolume() {  return volume; //getting volume  }   void display() {  cout << "SURFACE AREA = " << surfaceArea << endl;  cout << "VOLUME = " << volume << endl; //displaying surface area and volume  }   ~cone() {  cout << "destructor of cone is called.." << endl; //destructor is called  } };   * **I have just added the virtual keyword in the class sphere.**   **Output:**  **Text  Description automatically generated**  **Text  Description automatically generated** |