|  |
| --- |
| *Варіант 4.* |
| 1. Описати клас, який містять вказані поля і методи.   |  |  | | --- | --- | | **Клас** “Трикутник ” – TTriangle | | | **поля** | * для зберігання довжин сторін; | | **методи** | * конструктор без параметрів, конструктор з параметрами, конструктор копіювання; * введення/виведення даних; * визначення площі; * визначення периметру; * порівняння з іншим трикутником; * перевантаження операторів + (додавання довжин сторін), – (віднімання довжин відповідних сторін), \* (множення сторін на деяке число). | |
| 2. Створити клас-нащадок TTrianglePrizm (пряма призма, в основі якої трикутник) на основі класу TTriangle. Додати метод знаходження об’єму призми та перевизначити відповідні методи. |
| 3. Створити програму-клієнт для тестування. |
| using System;  namespace task1  {  class Program  {  static void Main(string[] args)  {  try  {  TTriangle tr1 = new TTriangle(3, 4, 5);  TTriangle tr2 = new TTriangle(6, 8, 10);  TTrianglePrizm a = new TTrianglePrizm(18, 3, 10, 4);  TTrianglePrizm k = new TTrianglePrizm(11, 13, 1, 2);  Console.Write("Введіть число: ");  double num = Convert.ToDouble(Console.ReadLine());  Console.WriteLine($"Площа(S) = {tr1.Area()}");  Console.WriteLine($"Периметр(P) = {tr1.Perimeter()}");  if (tr1 == tr2)  Console.WriteLine("Рівний");  else  Console.WriteLine("Не рівний");  Console.WriteLine($"Додавання {num}: {tr1 + num}");  Console.WriteLine($"Віднімання{num}: {tr1 - num}");  Console.WriteLine($"Множення {num}: {tr1 \* num}");  Console.WriteLine($"Tr1 = Tr2: { tr1 == tr2}");  TTriangle copiedTriangle = new TTriangle(tr2);  Console.WriteLine(copiedTriangle);  Console.Write("Введіть число: ");  double num2 = Convert.ToDouble(Console.ReadLine());  Console.WriteLine($"Площа(S) = {a.Area()}");  Console.WriteLine($"Обєм(V) = {a.Volume()}");  if (tr1 == tr2)  Console.WriteLine("Рівний");  else  Console.WriteLine("Не рівний");  Console.WriteLine($"Додавання {num2}: {a + num2}");  Console.WriteLine($"Віднімання{num2}: {a - num2}");  Console.WriteLine($"Множення {num2}: {a \* num2}");  Console.WriteLine($"A = B: { a==k}");  TTrianglePrizm copiedTTrianglePrizm = new TTrianglePrizm(k);  Console.WriteLine(copiedTTrianglePrizm);  }  catch  {  Console.WriteLine("Сторона має додатнє число");  }        }  }  } |
| using System;  using System.Collections.Generic;  using System.Text;  namespace task1  {  class TTriangle  {  private const double Epsilon = 0.0001;  private double a;  private double b;  private double c;  public double A  {  get { return a; }  set  {  if (value < 0)  throw new Exception("Сторона має додатнє число");  a = value;  }  }  public double B  {  get { return b; }  set  {  if (value < 0)  throw new Exception("Сторона має додатнє число");  b = value;  }  }  public double C  {  get { return c; }  set  {  if (value < 0)  throw new Exception("Сторона має додатнє число");  c = value;  }  }  public TTriangle(double userA, double userB, double userC)  {  A = userA;  B = userB;  C = userC;  }  public override string ToString()  {  return $"a = {a}, b = {b}, c = {c}";  }  // Конструктор копіювання  public TTriangle(TTriangle previousTriangle)  {    A = previousTriangle.a;  B = previousTriangle.b;  C = previousTriangle.c;  }  // Площа  public double Area()  {  double p = (a + b + c) / 2;  return Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));  }  // периметр  public double Perimeter()  {  return a + b + c;  }  //Порівняння трикутників    public static bool operator ==(TTriangle tr1, TTriangle tr2)  {  return (Math.Abs(tr1.a - tr2.a) < Epsilon && Math.Abs(tr1.b - tr2.b) < Epsilon && Math.Abs(tr1.c - tr2.c) < Epsilon);  }  public static bool operator !=(TTriangle tr1, TTriangle tr2)  {  return !(tr1 == tr2);  }  // Перевантаження операції +  public static TTriangle operator +(TTriangle triangle, double num)  {  return new TTriangle(triangle.a + num, triangle.b + num, triangle.c + num);  }  // Перевантаження операції-  public static TTriangle operator -(TTriangle triangle, double num)  {  return new TTriangle(triangle.a - num, triangle.b - num, triangle.c - num);  }  // Перевантаження операції \*  public static TTriangle operator \*(TTriangle triangle, double num)  {  return new TTriangle(triangle.a \* num, triangle.b \* num, triangle.c \* num);  }  }  } |
| using System;  using System.Collections.Generic;  using System.Text;  namespace task1  {  class TTrianglePrizm : TTriangle  {  private double h;  public double H  {  get { return h;  }  set  {  if (value < 0)  throw new Exception("Сторона має додатнє число");  h = value;  }  }  public TTrianglePrizm(double a, double b, double c, double h) : base(a, b, c)  {  H = h;  }  public double Volume()  {  return base.Area() \* h;  }  new double Area()  {  return base.Perimeter() \* h + 2 \* base.Area();  }  public TTrianglePrizm(TTrianglePrizm previousTTrianglePrizm)  {  A = previousTTrianglePrizm.A;  B = previousTTrianglePrizm.B;  C = previousTTrianglePrizm.C;  H = previousTTrianglePrizm.h;  }  // Перевантаження операції +  public static TTrianglePrizm operator +(TTrianglePrizm triangle, double num)  {  return new TTrianglePrizm (triangle.A + num, triangle.B + num, triangle.C + num, triangle.h + num);  }  // Перевантаження операції-  public static TTrianglePrizm operator -(TTrianglePrizm triangle, double num)  {  return new TTrianglePrizm (triangle.A - num, triangle.B - num, triangle.C - num, triangle.h - num);  }  // Перевантаження операції \*  public static TTrianglePrizm operator\* (TTrianglePrizm triangle, double num)  {  return new TTrianglePrizm (triangle.A \* num, triangle.B \* num, triangle.C \* num, triangle.h \* num);  }  }  } |