**Выполнил:** Волков Евгений Дмитриевич, ИСТбп-18-1

# Лабораторное задание №1

## Задача на If

### Постановка задачи

**Задача №12.** Общая точка. Два отрезка на плоскости заданы координатами своих концов. Определить, имеют ли эти отрезки общие точки. Замечание: Необходимо рассмотреть различные случаи взаимной ориентации отрезков: на одной прямой, на параллельных или пересекающихся прямых.

### Словесное описание алгоритма

1. Создаем класс Point для работы с координатами x и y
2. Проверяем взаимное положение отрезков
   1. Если данные точки введены неверно, то есть сначала введена правая точка, затем левая, то восстановим порядок перечисления точек слева направо
   2. Если конец первого отрезка расположен левее, чем начало правого, то они не пересекаются. Выведем на экран "Отрезки не пересекаются" и завершим выполнение функции
   3. Если оба отрезки вертикальные
      1. Если высшая точка нижнего отрезка находится ниже, чем нижняя точка верхнего отрезки, то выведем на экран «Отрезки не пересекаются» и завершим выполнение функции
      2. Иначе
         1. Если точки совпадают, то выведем "Отрезки на одной прямой и совпадают"
         2. Иначе выведем "Отрезки на одной прямой и пересекаются"
         3. Завершим выполнение функции
   4. Если один из отрезков вертикальный – разница абсцисс крайних точек одного отрезка равна нулю
      1. Посчитаем значения переменных A1, B1, A2, B2, которые необходимы в формуле A\*x-B=y. Она в свою очередь нужна, чтобы высчитать точку пересечения отрезков
         1. Если вертикальный первый отрезок, то посчитаем значение переменных A2, B2
            1. A2 = (точка 2.y - точка 1.y) / (точка 2.x - точка 1.x);
            2. B2 = A2 \* точка 3.x - точка 3.y;
         2. Если вертикальный второй отрезок, то посчитаем значение переменных A1, B1
            1. A1 = (точка 2.y - точка 1.y) / (точка 2.x - точка 1.x);
            2. B1 = A1 \* точка 2.x - точка 2.y;
      2. Если полученные значения находятся в допустимых диапазонах, заданных крайними точками отрезков, то выведем значение точки пересечения, иначе выведем, что отрезки не пересекаются
      3. Иначе выведем, что отрезки не пересекаются
      4. Завершим выполнение функции
   5. Посчитаем значения переменных A1, B1, A2, B2, которые необходимы в формуле A\*x-B=y. Она в свою очередь нужна, чтобы высчитать точку пересечения отрезков
      1. A1 = (точка 2.y - точка 1.y) / (точка 2.x - точка 1.x);
      2. A2 = (точка 2.y - точка 1.y) / (точка 2.x - точка 1.x);
      3. B1 = A1 \* точка 2.x - точка 2.y;
      4. B2 = A2 \* точка 3.x - точка 3.y;
   6. Если A1 = B2
      1. Выведем, что отрезки параллельны
      2. Завершим выполнение функции
   7. Если полученные значения находятся в допустимых диапазонах, заданных крайними точками отрезков
      1. Выведем значение точки пересечения,
      2. Иначе выведем, что отрезки не пересекаются
      3. Завершим выполнение функции

### Таблица спецификаций

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| x | Значение абсциссы | Вещественное число |
| y | Значение ординаты | Вещественное число |
| Выходные величины | | |
| res | Взаимное расположение отрезков | Текст |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Результат работы при непересекающихся отрезках | {-3, 4}, {-1, 2},  {1, 1}, {3, 2} | Отрезки не пересекаются |
| 2 | Результат работы при расположенных на одной прямой отрезках | {1, -2}, {1, 1}, {1, 2}, {1, 4} | Отрезки не пересекаются |
| 3 | Результат работы при расположенных на одной прямой и совпадающих отрезках | {1, -2}, {1, 1}, {1, -2}, {1, 1} | Отрезки на одной прямой и совпадают |
| 4 | Результат работы при первом вертикальном отрезке | {1, 0}, {1, 5}, {-3, 4}, {3, 1} | Отрезки пересекаются в точке (1, 2) |
| 5 | Результат работы при втором вертикальном отрезке | {-3, 4}, {3, 1},  {1, 0}, {1, 5} | Отрезки пересекаются в точке (1, 2) |
| 6 | Результат работы при параллельных отрезках | {-3, 4}, {3, 1},  {-3, 2}, {3, -1} | Отрезки параллельны |
| 7 | Результат работы при общем случае | {-3, 4}, {3, 1}, {-1, 0}, {2, 3} | Отрезки пересекаются в точке (1, 2) |

### Код программы

|  |
| --- |
| using System;  class Lab1  {  public class Point  {  public double x, y;  public Point(double abc, double ord)  {  x = abc;  y = ord;  }  public Point()  {  }  }  //Проверка на отношение отрезков  private static string CheckLines(Point p1, Point p2, Point p3, Point p4)  {  //Разположение точек слева направо  if (p1.x > p2.x)  {  Point temp = p1;  p1 = p2;  p2 = temp;  Console.WriteLine("X");  }  if (p3.x > p4.x)  {  Point temp = p3;  p3 = p4;  p4 = temp;  Console.WriteLine("Y");  }  //Если конец первого отрезка расположен левее, чем начало правого, то они не пересекаются  if (p2.x < p3.x)  {  return "Отрезки не пересекаются";  }  //Если оба отрезки вертикальные  if ( (p2.x - p1.x == 0) && (p4.x - p3.x == 0) )  {  if ( Math.Max(p1.y, p2.y) < Math.Min(p3.y, p4.y) || Math.Min(p1.y, p2.y) > Math.Max(p3.y, p4.y))  {  return "Отрезки не пересекаются";  }  else  {  if (p1.x == p3.x &&  p1.y == p3.y &&  p2.x == p4.x &&  p2.y == p4.y)  {  return "Отрезки на одной прямой и совпадают";  }  else  {  return "Отрезки на одной прямой и пересекаются";  }  }  }  // A\*x + b = y  if (p2.x - p1.x == 0)  {  double a2 = (p4.y - p3.y) / (p4.x - p3.x);  double b2 = p3.y - a2 \* p3.x;  double PointIntersectionX = p1.x;  double PointIntersectionY = a2 \* PointIntersectionX + b2;  if (p3.x <= PointIntersectionX &&  p4.x >= PointIntersectionX &&  Math.Min(p1.y, p2.y) <= PointIntersectionY &&  Math.Max(p1.y, p2.y) >= PointIntersectionY)  {  return "Отрезки пересекаются в точке (" + PointIntersectionX + ", " + PointIntersectionY + ")";  } else  {  return "Отрезки не пересекаются";  }  }  else if (p4.x - p3.x == 0)  {  double a1 = (p2.y - p1.y) / (p2.x - p1.x);  double b1 = p2.y - a1 \* p2.x;  double PointIntersectionX = p3.x;  double PointIntersectionY = a1 \* PointIntersectionX + b1;  if (p1.x <= PointIntersectionX &&  p2.x >= PointIntersectionX &&  Math.Min(p3.y, p4.y) <= PointIntersectionY &&  Math.Max(p3.y, p4.y) >= PointIntersectionY)  {  return "Отрезки пересекаются в точке (" + PointIntersectionX + ", " + PointIntersectionY + ")";  }  else  {  return "Отрезки не пересекаются";  }  }  double A1 = (p2.y - p1.y) / (p2.x - p1.x);  double B1 = A1 \* p2.x - p2.y;  double A2 = (p4.y - p3.y) / (p4.x - p3.x);  double B2 = A2 \* p3.x - p3.y;  if (A1 == A2)  {  return "Отрезки параллельны";  }  double pointIntersectionX = (B1 - B2) / (A1 - A2);  double pointIntersectionY = A1 \* pointIntersectionX - B1;  if ((pointIntersectionX < Math.Max(p1.x, p3.x)) || (pointIntersectionX > Math.Min(p2.x, p4.x)))  {  //точка X находится вне пересечения проекций отрезков на ось X  return "Отрезки не пересекаются";  }  return "Отрезки пересекаются в точке (" + pointIntersectionX + ", " + pointIntersectionY + ")";  }  public static void Main(string[] args)  {  string res;  string[] names = new string[7]  {  "Не пересекаются",  "Расположены на одной прямой",  "Расположены на одной прямой и совпадают",  "Первый отрезок вертикальный",  "Второй отрезок вертикальный",  "Отрезки параллельны",  "Общий случай"  };  double[, ,] points = new double[7, 4, 2] {  {  {-3, 4},  {-1, 2},  {1, 1},  {3, 2}  },  {  {1, -2},  {1, 1},  {1, 2},  {1, 4}  },  {  {1, -2},  {1, 1},  {1, -2},  {1, 1}  },  {  {1, 0},  {1, 5},  {-3, 4},  {3, 1}  },  {  {-3, 4},  {3, 1},  {1, 0},  {1, 5}  },  {  {-3, 4},  {3, 1},  {-3, 2},  {3, -1}  },  {  {-3, 4},  {3, 1},  {-1, 0},  {2, 3}  }  };  Point Point1 = new Point();  Point Point2 = new Point();  Point Point3 = new Point();  Point Point4 = new Point();  for (int i = 0; i < names.Length; i++)  {  Console.WriteLine("\r\n" + names[i]);  Point1.x = points[i, 0, 0];  Point1.y = points[i, 0, 1];  Point2.x = points[i, 1, 0];  Point2.y = points[i, 1, 1];  Point3.x = points[i, 2, 0];  Point3.y = points[i, 2, 1];  Point4.x = points[i, 3, 0];  Point4.y = points[i, 3, 1];  res = CheckLines(Point1, Point2, Point3, Point4);  Console.WriteLine(res);  }  Console.ReadKey();  }  } |

## Задача на switch

### Постановка задачи

**Задача №12.** В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В каждом подцикле годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года определить его название, если 1984 год — начало цикла: «год зеленой крысы».

### Словесное описание алгоритма

1. С помощью функции getYearName проверяем значение переменной year на соответствие условиям
   1. Получаем разницу между 1984 (год начала цикла) и полученным годом, записываем в переменную deference
   2. Создаем переменную res, запишем в нее данные в следующем виде: \*Введенный год\* - год, чтобы в дальнейшем добавить название года и вывести, как результат работы функции
   3. С помощью функции convertToNormal, получим значение и запишем в переменную deference. Эта функции преобразует разницу между введенным годом и началом цикла таким образом, чтобы введенное значение было неотрицательным и не более 60, для дальнейшей работы с 60-летним циклом
      1. Введенное значение в переменную notNormalYear преобразуем таким образом, чтобы получить деление от остатка этой переменной на 60
      2. Если это значение меньше 60, то прибавим 60, чтобы получить допустимое значение.
      3. Вернем значение переменной notNormalYear
   4. Получим значение для расчета первого цикла – результат деления на 12 без остатка, запишем в firstSubCycle
   5. Получит значение для расчета второго цикла – остаток от деления deference на 12, запишем в secondSubCycle
   6. С помощью конструкции switch сравним условия и значения переменных и добавим соответствующее название года в переменную res
      1. С помощью switch получим название из 12-летного цикла и добавим его в переменную res
      2. С помощью switch получим название из 5-летного цикла и добавим его в переменную res
   7. Выведем res, как результат работы функции

### Таблица спецификаций

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| year | Год | Целое число |
| Выходные величины | | |
| res | Название года | Текст |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Название года | 1921 | 1921 - год черной курицы |
| 2 | 1954 | 1954 - год желтой лошади |
| 3 | 1984 | 1984 - год зеленой крысы |
| 4 | 2007 | 2007 - год красной свиньи |
| 5 | Название года | 2045 | 2045 - год зеленой коровы |

### Код программы

|  |
| --- |
| using System;  namespace ConsoleApplication3  {  class Program  {  public static string getYearName(int year)  {  int startCycle = 1984;  int deference = year - startCycle;  string res = Convert.ToString(year) + " - год ";  deference = convertToNormal(deference);  int firstSubCycle = deference / 12;  int secondSubCycle = deference % 12;  switch (firstSubCycle)  {  case 0:  res += "зеленой ";  break;  case 1:  res += "красной ";  break;  case 2:  res += "желтой ";  break;  case 3:  res += "белой ";  break;  case 4:  res += "черной ";  break;  default:  break;  }  switch (secondSubCycle)  {  case 0:  res += "крысы";  break;  case 1:  res += "коровы";  break;  case 2:  res += "тигра";  break;  case 3:  res += "зайца";  break;  case 4:  res += "дракона";  break;  case 5:  res += "змеи";  break;  case 6:  res += "лошади";  break;  case 7:  res += "овцы";  break;  case 8:  res += "обезьяны";  break;  case 9:  res += "курицы";  break;  case 10:  res += "собаки";  break;  case 11:  res += "свиньи";  break;  default:  break;  }  return res;  }  public static int convertToNormal(int notNormalYear)  {  notNormalYear %= 60;  if (notNormalYear < 0)  {  notNormalYear += 60;  }  return notNormalYear;    }  static void Main(string[] args)  {  int[] inputData = new int[] {  1921,  1954,  1984,  2007,  2045  };  foreach (var item in inputData)  {  Console.WriteLine(getYearName(item));  }  Console.ReadKey();  }  }  } |