**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Дисциплина:** Объектно-ориентированное программирование

**Тема:** Структуры. Типы значений и ссылочные типы

**Выполнил: студент группы** *231-338*

*Шаура Илья Максимович*

(Фамилия И.О.)

**Дата, подпись** \_\_\_*09.09.2024*\_\_\_  ***\_\_\_*** ***\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2024**

**Цель:** получить практические навыки в создании программ, содержащих добавление в программу структур. Изучить разницы в поведении типов значений и ссылочных типов. Изучить работу сборщика мусора, а также работы сборщика мусора с использованием пользовательских типов данных.

**Обоснование:** в языке C#, в работе структур и классов используются разные механизмы. При создании экземпляра структуры, экземпляр вместе со всеми его значениями помещается в стек (stack) – зону статической памяти, и переменная хранит в себе конкретные значения структуры. При создании класса же, экземпляр вместе со всеми его значениями помещается в кучу (heap) – зону динамической памяти, а переменная хранит в себе ссылку на зону памяти, в которой лежит созданный экземпляр.

Сборщик мусора не затрагивает работу стека - он работает с кучей. Если объект класса перестает использоваться, то при очистке стека ссылка на участок памяти также очищается, однако это не приводит к немедленной очистке самого участка памяти в куче. Впоследствии сборщик мусора увидит, что на данный участок памяти больше нет ссылок, и очистит его.

**Задание.**

1. Выберите любую концепцию, тему или приложение, которое вам интересно разработать в рамках этой лабораторной работы. Это может быть любое приложение, реализующее какую-либо логику (симулятор поиска данных, вычислений, обработка строк и тд). Обязательно убедитесь, что выбранное вами приложение можно логически разделить на компоненты, где одни компоненты будут использовать структуры, а другие - классы.

2. Создайте в своем проекте как минимум одну структуру и один класс, которые будут представлять компоненты вашего приложения.

3. Реализуйте функциональность для работы с этими компонентами, включая их инициализацию, изменение и вывод информации.

4. Реализуйте методы для создания копий экземпляров классов и структур:

4.1. Реализуйте методы, которые будут создавать новые экземпляры классов (копировать их) с изменёнными параметрами.

4.2. Реализуйте методы, которые будут создавать новые экземпляры структур (копировать их) с изменёнными параметрами. Для создания копий экземпляров примените выражение «with».

5. Протестируйте ваше приложение и убедитесь, что оно работает корректно и демонстрирует различия между значимыми и ссылочными типами.

**Выполнение**Файл Airport.cs:

|  |
| --- |
| namespace Shaura.OOP.Lab2;  public enum IsoCode {  **US**,  **CA**,  **RU**,  **DE** }  public struct Airport(IsoCode isoCode, string? localCode, string? name, double latitude, double longitude) {  public IsoCode IsoCode { get; set; } = isoCode;  public string? LocalCode { get; set; } = localCode;  public string? Name { get; set; } = name;  public double Latitude = latitude;  public double Longitude = longitude;   public void ChangePosition(double latitude, double longitude)  {  Latitude = latitude;  Longitude = longitude;  }   public Airport Copy(IsoCode? isoCode, string? localCode, string? name, double? latitude, double? longitude)  {  return new Airport() with  {  IsoCode = isoCode ?? IsoCode,  LocalCode = localCode ?? LocalCode,  Name = name ?? Name,  Latitude = latitude ?? Latitude,  Longitude = longitude ?? Longitude  };  } }  public class Map {  public double CurrentLatitude = 0;  public double CurrentLongitude = 0;  public List<Airport> Airports { get; set; }    public Map()  {  Airports = [  new Airport(IsoCode.**US**,  "02OI",  "Murtha Airport",  41.801998138427734, -80.56539916992188),   new Airport(IsoCode.**RU**,  "SVO",  "Sheremetyevo International Airport",  55.972599, 37.4146),   new Airport(IsoCode.**CA**,  "CHB3",  "Hope Bay Aerodrome",  68.156, -106.618),   new Airport(IsoCode.**DE**,  "DE-0029",  "Berlinchen Airfield",  53.2251775192, 12.565526962299998)  ];  }    public void ChangeCurrentPosition(double latitude, double longitude)  {  CurrentLatitude = latitude;  CurrentLongitude = longitude;  }   public Airport? SearchAirport(string searchString)  {  return Airports.Find(airport => airport.LocalCode == searchString   || airport.Name == searchString   || airport.IsoCode.ToString() == searchString);  } } |

Файл Program.cs:

|  |
| --- |
| using Shaura.OOP.Lab2;  *// Структуры Airport* var airport1 = new Airport(   IsoCode.**US**,  "02OI",  "Murtha Airport",  41.801998138427734, -80.56539916992188); var airport2 = new Airport(  IsoCode.**RU**,  "SVO",  "Sheremetyevo International Airport",  55.972599, 37.4146);  airport2.ChangePosition(30, 30); airport1 = airport2; *// присвоение значения* airport2.ChangePosition(40, 40); *// airport1.Latitude = 30 и airport1.Longitude = 30 по прежнему* Console.WriteLine("------------- Структуры -------------"); Console.WriteLine($"First Airport Location: {airport1.Latitude} lat., {airport1.Longitude} long."); *// 30, 30* Console.WriteLine($"Second Airport Location: {airport2.Latitude} lat., {airport2.Longitude} long."); *// 40, 40* var map1 = new Map(); var map2 = new Map();  map2.ChangeCurrentPosition(10, 10); map1 = map2; *// присвоение ссылки* map2.ChangeCurrentPosition(20, 20); *// теперь и map1.Latitude и map1.Longitude = 20,  // так как обе ссылки (map1 и map2)  // указывают на один объект в куче* Console.WriteLine(); Console.WriteLine("------------- Классы -------------"); Console.WriteLine($"First Map Location: {map1.CurrentLatitude} lat., {map1.CurrentLongitude} long."); *// 20, 20* Console.WriteLine($"Second Map Location: {map2.CurrentLatitude} lat., {map2.CurrentLongitude} long."); *// 20, 20* Console.WriteLine(); Console.WriteLine("------------- Payload -------------"); *// Полезная нагрузка лаб. работы* Console.WriteLine(map1.SearchAirport("SVO")?.Name); *// Поиск аэропорта на карте по названию* |

Консольный вывод:

