**Требования к оформлению:**

* Шрифт у обычного текста 14pt. Нет пробелов или отступов между текстом.
* Должен быть указан вариант и задание.
* Листинги и рисунки нумеруются и подписываются снизу с описанием что это за листинг/рисунки. Так же на листинг или рисунок должна быть ссылка в тексте по типу: «Ниже представлен листинг 1», «как показано на рисунке 1».
* Подпись внизу листинга/рисунка 12pt.
* Между текстом и листингами/рисунками ставится пустая строка 10pt.

**Пример того, как должен выглядеть отчет показан ниже.**

**Государственное образовательное учреждение**

**среднего профессионально образования «Тираспольский техникум информатики и права»**

**ОТЧЁТ**

**Практическое задание №1**

по дисциплине «Разработка программных модулей»

**тема: «Ветвление. Условный оператор»**

Работу выполнил

студент группы 215

Кучеренко Данил

Руководитель,

преподаватель

Подсекина Т.С.

Тирасполь, 2025

**Вариант 2**

**Задание:** **Клонирование объектов (поверхностное и глубокое)**

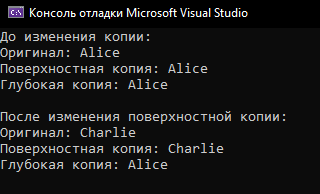
Создайте класс Company, содержащий массив Employee[].  
**Задача:** Реализуйте **поверхностное** и **глубокое** копирование компании.

* При поверхностном копировании измените сотрудника в копии.
* Докажите, что изменения повлияли на оригинал.
* Реализуйте глубокое копирование и убедитесь, что изменения теперь **не затрагивают оригинал**.

|  |
| --- |
| using System;  class Company  {  public class Employee  {  public string Name { get; set; }  public int Age { get; set; }    public Employee(string name, int age)  {  Name = name;  Age = age;  }    public Employee ShallowCopy()  {  return (Employee)this.MemberwiseClone();  }    public Employee DeepCopy()  {  return new Company.Employee(this.Name, this.Age);  }  }    public Employee[] Employees { get; set; }    public Company(Employee[] employees)  {  Employees = employees;  }    public Company ShallowCopy()  {  return (Company)this.MemberwiseClone();  }    public Company DeepCopy()  {  Employee[] copiedEmployees = new Employee[Employees.Length];  for (int i = 0; i < Employees.Length; i++)  {  copiedEmployees[i] = Employees[i].DeepCopy();  }  return new Company(copiedEmployees);  }    static void Main()  {  Company.Employee[] employees =  {  new Company.Employee("Alice", 30),  new Company.Employee("Bob", 25)  };  Company originalCompany = new Company(employees);  Company shallowCopy = originalCompany.ShallowCopy();  Company deepCopy = originalCompany.DeepCopy();  Console.WriteLine("До изменения копии:");  Console.WriteLine($"Оригинал: {originalCompany.Employees[0].Name}");  Console.WriteLine($"Поверхностная копия: {shallowCopy.Employees[0].Name}");  Console.WriteLine($"Глубокая копия: {deepCopy.Employees[0].Name}\n");  // Меняем имя в поверхностной копии  shallowCopy.Employees[0].Name = "Charlie";  Console.WriteLine("После изменения поверхностной копии:");  Console.WriteLine($"Оригинал: {originalCompany.Employees[0].Name}"); // Изменится  Console.WriteLine($"Поверхностная копия: {shallowCopy.Employees[0].Name}"); // Изменится  Console.WriteLine($"Глубокая копия: {deepCopy.Employees[0].Name}\n"); // Не изменится  }  } |

Листинг 1 – программа определения max и min числа

Ниже на рисунках 1-3 представлены тесты программы при разных входных параметрах.

****Рисунок 1 – тест 1