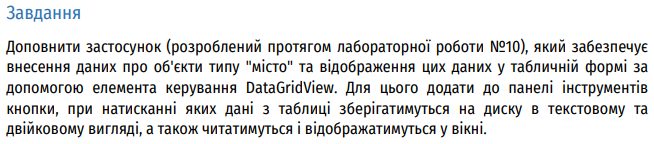
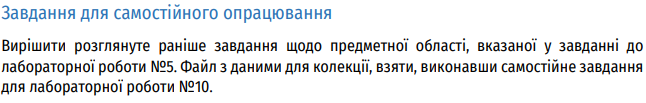
Звіт з лабораторної роботи №14





Лістинг:

1. Клас Processor:

using System;

namespace Lab14

{

// Full description of processor class:

public class Processor : IComparable

{

public string Name { get; set; }

public string Socket { get; set; }

public string Architecture { get; set; }

public int CoresCount { get; set; }

public double ClockFrequency { get; set; }

public double EnergyConsumption { get; set; }

public bool HasIntegratedVideocard { get; set; }

public bool HasUnlockedMultiplier { get; set; }

public double GetThreadsCount()

{

return CoresCount \* 2;

}

public Processor() { }

public Processor(string name, string socket, string architecture,

int coresCount, double clockFrequency, double energyConsumption,

bool hasIntegratedVideocard, bool hasUnlockedMultiplier)

{

Name = name;

Socket = socket;

Architecture = architecture;

CoresCount = coresCount;

ClockFrequency = clockFrequency;

EnergyConsumption = energyConsumption;

HasIntegratedVideocard = hasIntegratedVideocard;

HasUnlockedMultiplier = hasUnlockedMultiplier;

}

public string Info()

{

return Name + ", " + Socket + ", " + Architecture;

}

public int CompareTo(object obj)

{

Processor p = obj as Processor;

return string.Compare(this.Name, p.Name);

}

}

}

1. Клас Program:

using System;

using System.Collections.Generic;

using System.IO;

namespace Lab14

{

class Program

{

static List<Processor> processors;

static void PrintTowns()

{

foreach (Processor processor in processors)

{

Console.WriteLine(processor.Info().Replace('і', 'i'));

}

Console.WriteLine();

}

static void Main(string[] args)

{

processors = new List<Processor>();

FileStream fs = new FileStream("procesir\_binary.towns", FileMode.Open);

BinaryReader reader = new BinaryReader(fs);

try

{

Processor processor;

Console.WriteLine("Читаємо данi з файлу...\n");

while (reader.BaseStream.Position < reader.BaseStream.Length)

{

processor = new Processor();

for (int i = 1; i <= 8; i++)

{

switch (i)

{

case 1: processor.Name = reader.ReadString(); break;

case 2: processor.Socket = reader.ReadString(); break;

case 3: processor.Architecture = reader.ReadString(); break;

case 4: processor.CoresCount = reader.ReadInt32(); break;

case 5: processor.ClockFrequency = reader.ReadDouble(); break;

case 6: processor.EnergyConsumption = reader.ReadDouble(); break;

case 7: processor.HasIntegratedVideocard = reader.ReadBoolean(); break;

case 8: processor.HasUnlockedMultiplier = reader.ReadBoolean(); break;

}

}

processors.Add(processor);

}

}

catch (Exception ex)

{

Console.WriteLine("Сталась помилка: {0}", ex.Message);

}

finally

{

reader.Close();

}

Console.WriteLine("Несортований перелiк процесорiв: {0}", processors.Count);

PrintTowns();

processors.Sort();

Console.WriteLine("Cортований перелiк процесорiв: {0}", processors.Count);

PrintTowns();

Console.WriteLine("Додаємо новий запис: AMD");

Processor processorAMD = new Processor("AMD", "Socket", "Architecture", 4, 8, 20, false, true);

processors.Add(processorAMD);

processors.Sort();

Console.WriteLine(" Перелiк процесорiв: {0}", processors.Count);

PrintTowns();

Console.WriteLine(" Видаляємо останнє значення");

processors.RemoveAt(processors.Count - 1);

Console.WriteLine(" Перелiк процесорiв: {0}", processors.Count);

PrintTowns();

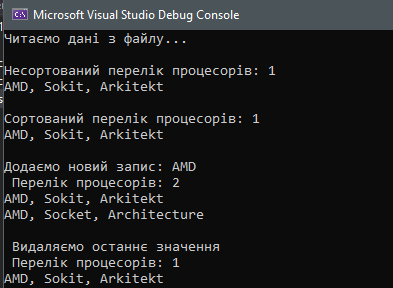
Console.ReadKey();

}

}

}

Результат:



Кінець звіту