|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №6**

**«Использование стандартных библиотек для поиска и сортировки данных»**

**ДИСЦИПЛИНА: «Логика и теория алгоритмов»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличев Г.Э. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** приобрести практические навыки использования стандартных библиотеки при решении задач сортировки и поиска записей.

**Вариант №8-7-5**

**Реализовать класс:**

*Сущность*: Товар (Stock)

*Атрибуты*: Инвентарный номер, дата поступления, стоимость, местоположение на складе (отдельный класс с атрибутами: зона, область, ячейка)

**Реализовать отношение порядка по умолчанию:**

По возрастанию уникального номера

**Реализовать несколько других возможных отношений порядка:**

*Порядок 1*: по убыванию даты, для равных дат – в лексикографическом порядке инвентарного номера

*Порядок 2*: по возрастанию цены

Разработать класс, представляющий хранилище объектов-сущностей. Данный класс должен содержать метод Add добавления объекта в хранилище, метод GetAll получения всех объектов в порядке их добавления, метод GetAllOrdered получения всех объектов в порядке сортировки по умолчанию, методы получения всех объектов в порядках сортировки, реализованных в ходе

выполнения п. 3.

Разработать демонстрационное консольное приложение, в котором жестко задан набор из 6-8 сущностей с различными значениями атрибутов. Приложение должно выводить список сущностей в исходном порядке, а также в порядках, реализованных в пп. 2 и 3.

**Реализовать отношение равенства по умолчанию:**

Для класса Местоположение: по равенству значений всех полей

В класс, представляющий хранилище объектов-сущностей, добавить методы поиска сущностей по ключевым значениям. Реализовать требуемые отношения равенства, отличные от равенства по умолчанию.

*Поиск 1*: товаров по местоположению (ключ – местоположение)

*Поиск 2*: товаров по зоне и области (ключ – местоположение)

*Поиск 3*: товара по инвентарному номеру (ключ – инвентарный номер)

Доработать демонстрационное консольное приложение, так, чтобы оно позволяло осуществлять поиск по правилам, реализованным в п. 7. Для каждого варианта поиска приложение должно запросить значение ключа, выполнить поиск и вывести список найденных сущностей.

**Листинг:**

***Stock.cs***

using System;

using System.Collections.Generic;

namespace LITA\_LW\_6

{

public class Stock

{

public Stock(uint id, string inventoryNumber, DateTimeOffset arrivalDate, uint cost, WarehouseLocation location)

{

Id = id;

InventoryNumber = inventoryNumber;

ArrivalDate = arrivalDate;

Cost = cost;

Location = location;

}

public uint Id { get; }

public string InventoryNumber { get; set; }

public DateTimeOffset ArrivalDate { get; set; }

public uint Cost { get; set; }

public WarehouseLocation Location { get; set; }

public static bool operator <(Stock first, Stock second) => first.Id < second.Id;

public static bool operator >(Stock first, Stock second) => first.Id > second.Id;

public static bool operator <=(Stock first, Stock second) => first.Id <= second.Id;

public static bool operator >=(Stock first, Stock second) => first.Id >= second.Id;

public static bool operator ==(Stock first, Stock second) => first.Id == second.Id;

public static bool operator !=(Stock first, Stock second) => first.Id != second.Id;

public static List<Stock> Sort(List<Stock> list)

{

int i = 0;

bool isUnsorted = true;

while (isUnsorted)

{

isUnsorted = false;

for (int j = 0; j < list.Count - i - 1; ++j)

if (list[j] > list[j + 1])

{

Stock temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

isUnsorted = true;

}

++i;

}

return list;

}

public static List<Stock> SortByDate(List<Stock> list)

{

int i = 0;

bool isUnsorted = true;

while (isUnsorted)

{

isUnsorted = false;

for (int j = 0; j < list.Count - i - 1; ++j)

{

if (list[j].ArrivalDate < list[j + 1].ArrivalDate)

{

Stock temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

isUnsorted = true;

}

else if (list[j].ArrivalDate == list[j + 1].ArrivalDate)

{

bool isNeededToSort = false;

for (int k = 0; k < (list[j].InventoryNumber.Length > list[j + 1].InventoryNumber.Length ?

list[j + 1].InventoryNumber.Length : list[j].InventoryNumber.Length); ++k )

{

if (list[j].InventoryNumber[k] > list[j + 1].InventoryNumber[k])

{

isNeededToSort = true;

break;

}

if (list[j].InventoryNumber[k] < list[j + 1].InventoryNumber[k])

break;

}

if (isNeededToSort)

{

Stock temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

isUnsorted = true;

}

}

}

++i;

}

return list;

}

public static List<Stock> SortByCost(List<Stock> list)

{

int i = 0;

bool isUnsorted = true;

while (isUnsorted)

{

isUnsorted = false;

for (int j = 0; j < list.Count - i - 1; ++j)

if (list[j].Cost > list[j + 1].Cost)

{

Stock temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

isUnsorted = true;

}

++i;

}

return list;

}

}

}

***WarehouseLocation.cs***

namespace LITA\_LW\_6

{

public class WarehouseLocation

{

public WarehouseLocation(string zone, string area, uint cell)

{

Zone = zone;

Area = area;

Cell = cell;

}

public string Zone { get; set; }

public string Area { get; set; }

public uint Cell { get; set; }

public static bool operator ==(WarehouseLocation first, WarehouseLocation second) =>

first.Zone == second.Zone && first.Area == second.Area && first.Cell == second.Cell;

public static bool operator !=(WarehouseLocation first, WarehouseLocation second) => !(first == second);

}

}

***StockStorage.cs***

using System.Collections.Generic;

namespace LITA\_LW\_6

{

public class StockStorage

{

public void Add(Stock stock) { \_stocks.Add(stock); }

public List<Stock> GetAll() { return \_stocks; }

public List<Stock> GetAllOrdered() { return Stock.Sort(\_stocks); }

public List<Stock> GetAllOrderedByDate() { return Stock.SortByDate(\_stocks); }

public List<Stock> GetAllOrderedByCost() { return Stock.SortByCost(\_stocks); }

public List<Stock> FindByLocation(WarehouseLocation location)

{ return \_stocks.FindAll(x => x.Location == location); }

public List<Stock> FindByZoneAndArea(string zone, string area)

{ return \_stocks.FindAll(x => x.Location.Zone == zone && x.Location.Area == area); }

public List<Stock> FindByNumber(string number)

{ return \_stocks.FindAll(x => x.InventoryNumber == number); }

List<Stock> \_stocks = new();

}

}

***Program.cs***

using System;

namespace LITA\_LW\_6

{

class Program

{

static void Main(string[] args)

{

WarehouseLocation locationA = new("A", "a", 1);

WarehouseLocation locationB = new("B", "b", 2);

WarehouseLocation locationC = new("B", "b", 3);

StockStorage storage = new();

storage.Add(new(1, "1", DateTimeOffset.Parse("01.01.2001"), 300, locationA));

storage.Add(new(8, "8", DateTimeOffset.Parse("02.02.2002"), 700, locationC));

storage.Add(new(5, "5", DateTimeOffset.Parse("03.03.2003"), 500, locationB));

storage.Add(new(4, "1", DateTimeOffset.Parse("04.04.2004"), 100, locationA));

storage.Add(new(6, "6", DateTimeOffset.Parse("01.01.2001"), 400, locationC));

storage.Add(new(3, "1", DateTimeOffset.Parse("02.02.2002"), 800, locationA));

storage.Add(new(2, "2", DateTimeOffset.Parse("03.03.2003"), 200, locationB));

storage.Add(new(7, "7", DateTimeOffset.Parse("04.04.2004"), 600, locationB));

string command = "";

while (command != "0")

{

Console.Clear();

foreach (var stock in storage.GetAll())

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.WriteLine("\n1. Get all ordered");

Console.WriteLine("2. Get all ordered by date");

Console.WriteLine("3. Get all ordered by cost");

Console.WriteLine("4. Find by location");

Console.WriteLine("5. Find by zone and area");

Console.WriteLine("6. Find by number");

Console.WriteLine("7. Add");

Console.WriteLine("0. Exit");

Console.Write(">>> ");

command = Console.ReadLine();

Console.WriteLine();

if (command == "1")

{

foreach (var stock in storage.GetAllOrdered())

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "2")

{

foreach (var stock in storage.GetAllOrderedByDate())

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "3")

{

foreach (var stock in storage.GetAllOrderedByCost())

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "4")

{

string zone;

string area;

uint cell;

Console.Write("Zone: ");

zone = Console.ReadLine();

Console.Write("Area: ");

area = Console.ReadLine();

Console.Write("Cell: ");

cell = uint.Parse(Console.ReadLine());

Console.WriteLine();

foreach (var stock in storage.FindByLocation(new(zone, area, cell)))

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "5")

{

string zone;

string area;

Console.Write("Zone: ");

zone = Console.ReadLine();

Console.Write("Area: ");

area = Console.ReadLine();

Console.WriteLine();

foreach (var stock in storage.FindByZoneAndArea(zone, area))

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "6")

{

string number;

Console.Write("Inventory number: ");

number = Console.ReadLine();

Console.WriteLine();

foreach (var stock in storage.FindByNumber(number))

Console.WriteLine("ID: {0}; Inventory number: {1}; Arrival date: {2}; Cost: {3}; Zone: {4}; Area: {5}; Cell: {6}",

stock.Id, stock.InventoryNumber, stock.ArrivalDate.DateTime.ToShortDateString(), stock.Cost,

stock.Location.Zone, stock.Location.Area, stock.Location.Cell);

Console.ReadKey();

}

else if (command == "7")

{

uint id;

string inventoryNumber;

string arrivalDate;

uint cost;

string zone;

string area;

uint cell;

Console.Write("ID: ");

id = uint.Parse(Console.ReadLine());

Console.Write("Inventory number: ");

inventoryNumber = Console.ReadLine();

Console.Write("Arrival date: ");

arrivalDate = Console.ReadLine();

Console.Write("Cost: ");

cost = uint.Parse(Console.ReadLine());

Console.Write("Zone: ");

zone = Console.ReadLine();

Console.Write("Area: ");

area = Console.ReadLine();

Console.Write("Cell: ");

cell = uint.Parse(Console.ReadLine());

Console.WriteLine();

storage.Add(new(id, inventoryNumber, DateTimeOffset.Parse(arrivalDate), cost, new(zone, area, cell)));

}

}

}

}

}

**Результат:**

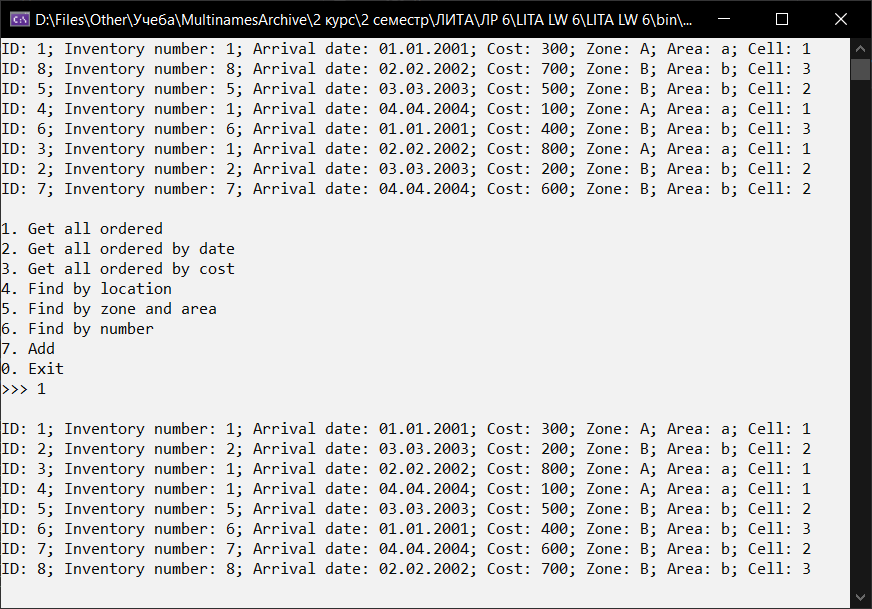


Рис. 1.1. Результат

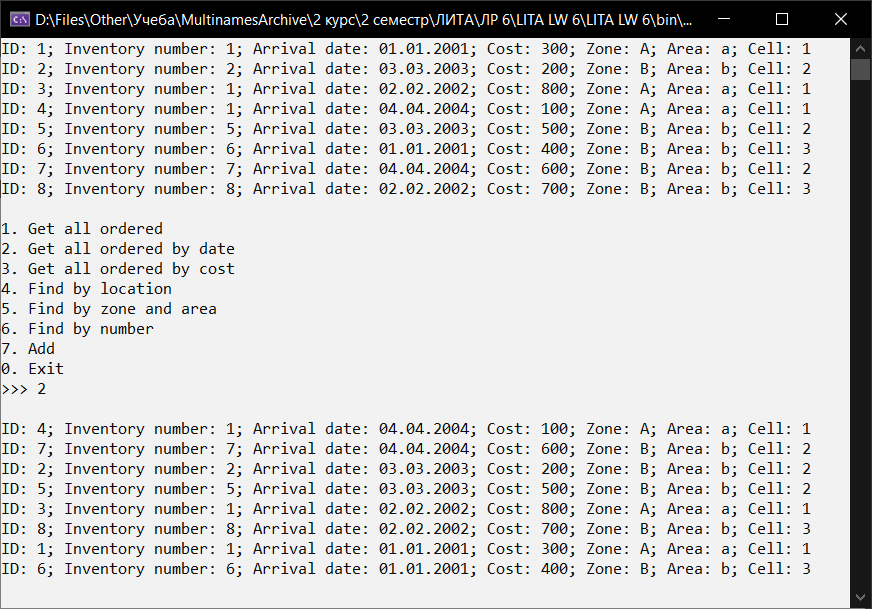


Рис. 1.2. Результат

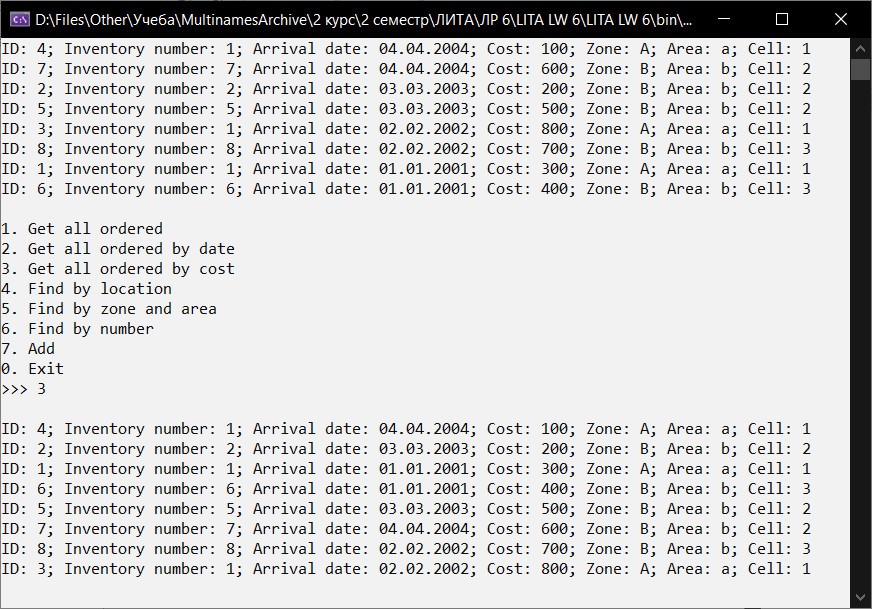


Рис. 1.3. Результат

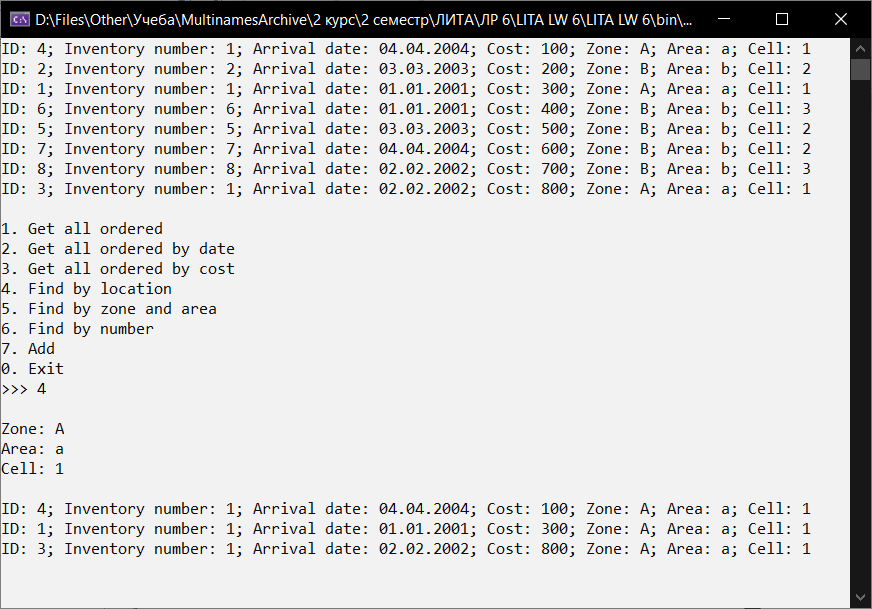


Рис. 1.4. Результат

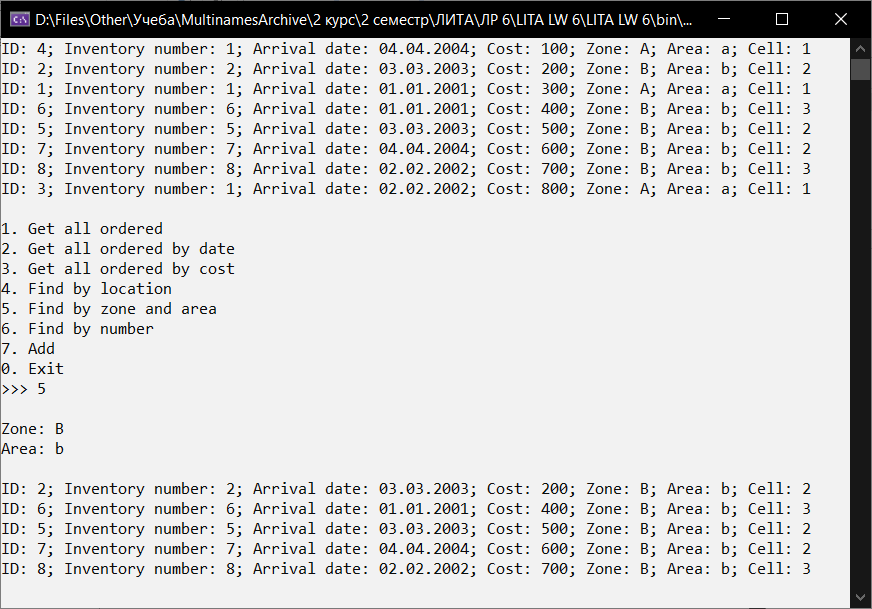


Рис. 1.5. Результат

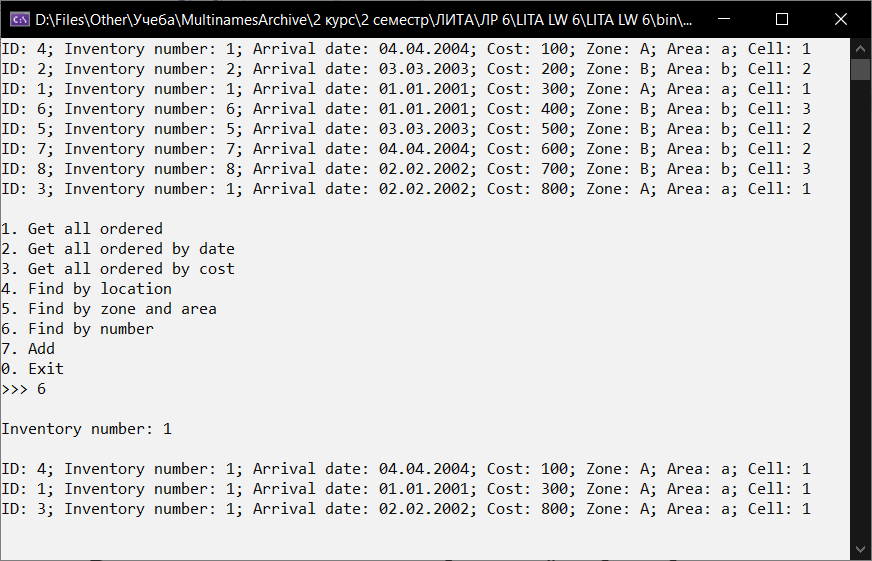


Рис. 1.6. Результат

**Вывод:** в ходе выполнения лабораторной работы были получены навыки использования стандартных библиотеки при решении задач сортировки и поиска записей.