Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни

«Основи програмування-2. Методології програмування»

«Файли даних. Бінарні файли»

Варіант 06

Виконав студент ІП-11 Головатюк Владислав Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №1.2**

**Файли даних. Бінарні файли**

**Мета**

***Мета роботи:*** вивчити особливості створення і обробки бінарних файлів

***Завдання:*** Створити файл з інформацією про товари меблевого магазину: назва товару, його вид, колір, ціна та наявна кількість. В новому бінарному файлі сформувати список усіх наявних у магазині стільців вказаного виду, іх кулькість та вартість. Видалити з нового файлу інформацію про стільці вартістю від 300 до 500 грн.

***Постановка задачі:*** Заповнюємо список даними від користувача, записуємо дані в перший файл. Після цього читаємо дані з першого файла та формуємо список на основі першого, залишаючи лише ті стільці, які відповідають введеному виду, і кількість яких в магазині більше нуля. Записуємо отриманий список у інший файл. Після отримуємо дані із файла і видаляємо із його мебель, вартість якої від 300 та 500 і перезаписуємо в той самий файл.

1. **С#**
   1. **Код:**
      1. **Program.cs**

using System;

using System.Collections.Generic;

namespace Lab2

{

internal class Program

{

static void Main(string[] args)

{

string firstPath = "first.dat",

outputPath = "output.dat";

var allFurniture = FileWorker.ReadFromFile(firstPath);

ConsoleWorker.OutputItems("All furniture list: ".PadLeft(32), allFurniture);

var availableChairs = ContentWorker.GetAvailableTypes(allFurniture, "chair", ConsoleWorker.Input("Kind to select: "));

FileWorker.WriteObjectsToFile(outputPath, availableChairs);

var chairs = FileWorker.ReadFromFile(outputPath);

ConsoleWorker.OutputItems("Available chairs of kind:".PadLeft(32), chairs);

var chairsInRange = ContentWorker.GetItemsOutOfRange(chairs, 300, 500);

FileWorker.WriteObjectsToFile(outputPath, chairsInRange);

var result = FileWorker.ReadFromFile(outputPath);

ConsoleWorker.OutputItems("Chairs with price not in [300-500]:".PadLeft(32), result);

Console.ReadLine();

}

}

}

* + 1. **FileWorker.cs**

using System;

using System.Collections.Generic;

using System.IO;

namespace Lab2

{

internal static class FileWorker

{

public static void WriteObjectsToFile(string path, params Furniture[] objs)

{

WriteInMode(path, objs, FileMode.Truncate);

}

public static void AppendObjectsToFile(string path, params Furniture[] objs)

{

WriteInMode(path, objs, FileMode.Append);

}

public static Furniture[] ReadFromFile(string path)

{

List<Furniture> list = new List<Furniture>();

if (File.Exists(path) == false)

return new Furniture[0];

using (BinaryReader reader = new BinaryReader(File.Open(path, FileMode.Open)))

{

while (reader.PeekChar() != -1)

{

list.Add(new Furniture(

reader.ReadString(),

reader.ReadString(),

reader.ReadInt32(),

reader.ReadInt32(),

ColorFromString(reader.ReadString())

));

}

}

return list.ToArray();

}

private static Color ColorFromString(string str)

{

return (Color)Enum.Parse(typeof(Color), str, true);

}

private static void WriteInMode(string path, Furniture[] objs, FileMode fileMode)

{

if (File.Exists(path) == false) File.Create(path).Close();

using (BinaryWriter writer = new BinaryWriter(File.Open(path, fileMode)))

{

foreach (var item in objs)

{

writer.Write(item.Name);

writer.Write(item.Kind);

writer.Write(item.Amount);

writer.Write(item.Price);

writer.Write(item.Color.ToString());

}

}

}

}

enum OpenMode

{

Append,

Write

}

}

* + 1. **ConsoleWorker.cs**

using System;

using System.Linq;

namespace Lab2

{

internal static class ConsoleWorker

{

public static Furniture GetFurnitureFromInput()

{

Console.WriteLine("===================");

return new Furniture(

Input("Enter furniture type".PadRight(32)),

Input("Enter purpose of item".PadRight(32)),

Convert.ToInt32(Input("Enter amount of item in storage".PadRight(32), str => uint.TryParse(str, out uint \_))),

Convert.ToInt32(Input("Enter price per one item".PadRight(32), str => uint.TryParse(str, out uint \_))),

(Color)Enum.Parse(typeof(Color), Input("Enter color of item".PadRight(32), str => Enum.GetNames(typeof(Color)).Contains(str)))

);

}

public static void OutputItems(string header, Furniture[] items)

{

Console.WriteLine(header);

foreach (var item in items)

{

Console.WriteLine(item.ToString());

Console.WriteLine("==================");

}

}

public static string Input(string content, Func<string, bool> func)

{

Console.Write(content + ": ");

string input = Console.ReadLine();

while (func(input) == false)

{

Console.Write("Invalid input. Try again: ");

input = Console.ReadLine();

}

return input;

}

public static string Input(string content)

{

Console.Write(content + ": ");

return Console.ReadLine();

}

}

}

* + 1. **ContentWorker.cs**

using System.Linq;

namespace Lab2

{

internal static class ContentWorker

{

public static Furniture[] GetAvailableTypes(Furniture[] furnitures, string category, string kind)

{

return furnitures.Any(f => f.Name == category)

? furnitures.Where(f => f.Name == category && f.Kind == kind && f.Amount > 0).ToArray()

: new Furniture[0];

}

public static Furniture[] GetItemsOutOfRange(Furniture[] furnitures, int minValue, int maxValue)

{

return furnitures.Where(f => f.Price <= minValue || f.Price >= maxValue).ToArray();

}

}

}

* + 1. **Furniture.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

internal class Furniture

{

public Furniture(string name, string type, int amount, int price, Color color)

{

Name = name;

Kind = type;

Amount = amount;

Price = price;

Color = color;

}

public string Name { get; private set; }

public string Kind { get; private set; }

public Color { get; private set; }

public int Price { get; private set; }

public int Amount { get; private set; }

public override string ToString()

{

return $"Title : {Name} {Kind}\n" +

$"Amount: {Amount}\n" +

$"Price : {Price}\n" +

$"Color : {Color}";

}

}

enum Color

{

Red,

Green,

Blue,

Yellow,

Orange,

Purple,

Brown,

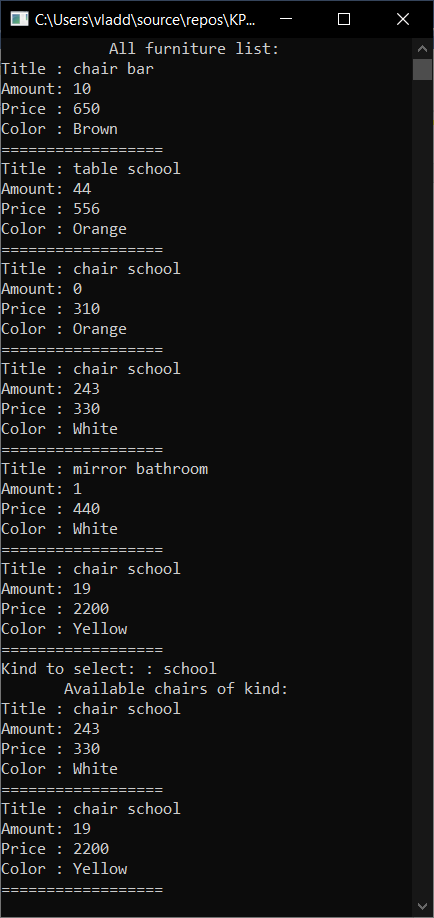
White,

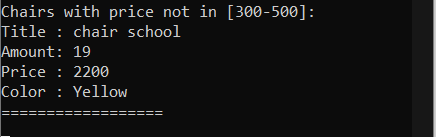
Black

}

}

* 1. **Запуск програми**
     1. **Перший**



****

1. **Python**
   1. **Код**
      1. **main.py**

**from** os **import** system  
  
**import** console\_worker **as** console  
**import** content\_worker **as** content  
**import** file\_worker **as** file  
**from** furniture **import** Furniture  
  
  
**def** main():  
 furniture = [  
 Furniture(**'chair'**, **'bar'**, 10, 650, **'Brown'**),  
 Furniture(**'table'**, **'school'**, 44, 556, **'Orange'**),  
 Furniture(**'chair'**, **'school'**, 0, 310, **'Orange'**),  
 Furniture(**'chair'**, **'school'**, 243, 330, **'White'**),  
 Furniture(**'mirror'**, **'bathroom'**, 1, 440, **'White'**),  
 Furniture(**'chair'**, **'school'**, 19, 2200, **'Grey'**)  
 ]  
 console.print\_items(**'All furniture list: '**, furniture)file.write\_to\_file(**'first.dat'**, furniture)  
  
 from\_file = file.read\_from\_file(**'first.dat'**)  
 available\_chairs = content.select\_available\_types(from\_file, **'chair'**, input(**'Print kind of item to select: '**))  
 console.print\_items(**'Available chairs with selected kind: '**, available\_chairs)  
 file.write\_to\_file(**'output.dat'**, available\_chairs)  
  
 from\_file = file.read\_from\_file(**'output.dat'**)  
 removed = content.without\_range(from\_file, 300, 500)  
 console.print\_items(**'Removed items with price in [300; 500]: '**, removed)  
 file.write\_to\_file(**'output.dat'**, removed)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()  
 system(**'pause'**)

* + 1. **content\_worker.py**

**from** furniture **import** Furniture  
  
  
**def** select\_available\_types(furniture\_list: list[Furniture], category: str, kind: str) -> list[Furniture]:  
 **return** [furn **for** furn **in** furniture\_list **if** furn.name == category **and** furn.kind == kind **and** furn.amount > 0]  
  
  
**def** without\_range(furniture\_list: list[Furniture], min\_value: int, max\_value: int) -> list[Furniture]:  
 **return** [furn **for** furn **in** furniture\_list **if** min\_value < furn.price > max\_value ]

* + 1. **furniture.py**

**class** Furniture:  
 **def** \_\_init\_\_(self, name: str, kind: str, amount: int, price: int, color: str) -> object:  
 self.name = name  
 self.kind = kind  
 self.amount = amount  
 self.price = price  
 self.color = color  
  
 **def** print(self):  
 print(**f'Title : {**self.name**} {**self.kind**}\n'  
 f'Amount in storage: {**self.amount**}\n'  
 f'Price per one : {**self.price**}\n'  
 f'Color : {**self.color**}'**)

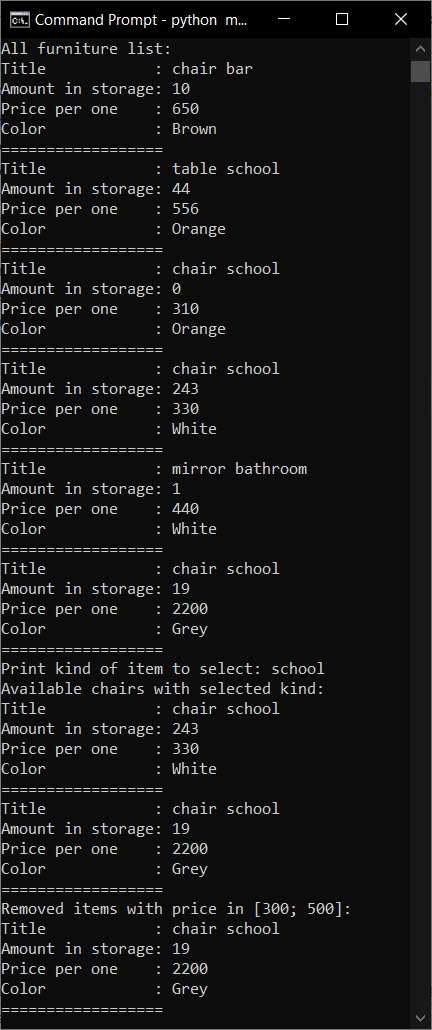
* + 1. **file\_worker.py**

**import** pickle  
  
**from** furniture **import** Furniture  
  
  
**def** write\_to\_file(path: str, furniture\_list: list[Furniture]):  
 **with** open(path, **'wb'**) **as** file:  
 **for** item **in** furniture\_list:  
 pickle.dump(item, file)  
  
  
**def** append\_to\_file(path: str, furniture\_list: list[Furniture]):  
 **with** open(path, **'ab'**) **as** file:  
 **for** item **in** furniture\_list:  
 pickle.dump(item, file)  
  
  
**def** read\_from\_file(path: str) -> list[Furniture]:  
 items = []  
 end\_file = **False  
 with** open(path, **'rb'**) **as** file:  
 **while not** end\_file:  
 **try**:  
 items.append(pickle.load(file))  
 **except** EOFError:  
 end\_file = **True  
 return** items

* + 1. **console\_worker.py**

**from** furniture **import** Furniture  
  
  
**def** get\_items\_from\_user() -> list[Furniture]:  
 cont = **False** items = [Furniture]  
 **while not** cont:  
 item = Furniture(  
 input(**'Enter category: '**),  
 input(**'Enter purpose : '**),  
 int(input(**'Enter amount : '**)),  
 int(input(**'Enter price : '**)),  
 input(**'Enter color : '**)  
 )  
 items.append(item)  
 cont = input(**"Continue? [y/n]"**) == **'y'  
 return** items  
  
**def** print\_items(header: str, items: list[Furniture]):  
 print(header)  
 **for** i **in** items:  
 i.print()  
 print(**'=================='**)

* 1. **Запуск програми**
     1. **Перший**



***Висновок:*** В процесі виконання лабораторної роботи я освоїв та опрацював навички в роботі із бінарними файлами на інтерпритованій МП Python та компільовій C#.