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 № 2 з дисципліни
«Основи програмування»

«Бінарні файли»
Варіант 35

Виконав студент ІП-11 Ющенко Андрій Вікторович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2021

Варіант 34

34. Створити файл з інформацією про студентів: ПІБ, дата народження, форма навчання, група, середній рейтинг успішності за останню сесію. На кожному потоці (групи потоку мають однакову абревіатуру) визначити студентів з мінімальним середнім балом успішності і групи, в яких вони навчаються. В новому файлі сформувати відсортований за прізвищами список студентів-четверокурсників денної форми навчання, середній бал успішності яких не менше за вказаний.

C#

Program.cs

```
namespace lab2
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Enter name of First file");
            string path1 = Console.ReadLine();
            Console.WriteLine("Enter name of Second file:");
            string path2 = Console.ReadLine();

            create_info createInfo = new create_info();
            List<info> list_of_students1 = createInfo.information();
            List<info> list_of_students2 = list_of_students1;
            Console.WriteLine("All students:");
            createInfo.output_list(list_of_students1);

            int amount_of_students = createInfo.CreateBinaryFile(list_of_students1, path1);

            List<info> lowest = createInfo.LowLevelStudents(path1, amount_of_students);
            Console.WriteLine("Lowest mark students: ");
            createInfo.output_list(lowest);

            List<info> sorted_list = createInfo.Identify(list_of_students2);
            Console.WriteLine("4-th grade students:");
            createInfo.output_list(sorted_list);
            int stud = createInfo.CreateBinaryFile(sorted_list, path2);
            Console.ReadLine();
        }
    }
}
```

functions.cs

```
namespace lab2
{
    21 usages
    public struct info
    {
        public string name,
            FormOfStudying,
            GroupName,
            DateOfBirth;
        public int GroupNumber;
        public double AverageMark;
    };
    2 usages
    public class create_info
    {
        1 usage
        public List<info> information()
        {
            Console.WriteLine("Enter amount of students:");
            string n = Console.ReadLine();
            int amount_of_students= Convert.ToInt32(n);
            List<info> list = new List<info>();
            info student_info;
            Console.Write("Enter the name of the group: ");
            string groupName = Console.ReadLine();
            for (int i = 0; i < amount_of_students; i++)
            {
                Console.Write("Student's name: ");
                student_info.name = Console.ReadLine();

                Console.Write("Date of Birth(DD.MM.YEAR): ");
                student_info.DateOfBirth = Console.ReadLine();

                Console.Write("Form of studying(Daily or Correspondence): ");
```

```
            student_info.FormOfStudying = Console.ReadLine();

            Console.Write("Group number: ");
            student_info.GroupNumber = Convert.ToInt32(Console.ReadLine());

            Console.Write("Average mark(0-100): ");
            student_info.AverageMark = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine();
            student_info.GroupName = groupName;
            list.Add(student_info);
        }
        return list;
    }
    3 usages
    public void output_list(List<info> list)
    {
        for (int i = 0; i < list.Count; i++)
        {
            Console.WriteLine($"Name: {list[i].name}, birthday: {list[i].DateOfBirth}, group: {list[i].GroupName}-{list[i].GroupNumbe
        }
    }
    2 usages
    public int CreateBinaryFile(List<info> list, string fileName)
    {
        BinaryWriter BinaryFile;
        BinaryFile = new BinaryWriter( output: new FileStream(fileName, FileMode.OpenOrCreate, FileAccess.Write));
        for (int i = 0; i < list.Count; i++)
        {
            BinaryFile.Write(list[i].name);
            BinaryFile.Write(list[i].DateOfBirth);
            BinaryFile.Write(list[i].FormOfStudying);
            BinaryFile.Write(list[i].GroupName);
            BinaryFile.Write(list[i].GroupNumber);
            BinaryFile.Write(list[i].AverageMark);
```

```
        }
        BinaryFile.Close();
        return list.Count;
    }
    1 usage
    public List<info> LowLevelStudents(string FileName, int AmountOfStudents)
    {
        List<info> MainList = new List<info>();
        List<info> InfoList = new List<info>();
        info StudentInfo;
        BinaryReader BinaryFileToRead;
        BinaryFileToRead = new BinaryReader( input: new FileStream(FileName, FileMode.OpenOrCreate, FileAccess.Read));
        for (int i = 0; i < AmountOfStudents; i++)
        {
            StudentInfo.name = BinaryFileToRead.ReadString();
            StudentInfo.DateOfBirth = BinaryFileToRead.ReadString();
            StudentInfo.FormOfStudying = BinaryFileToRead.ReadString();
            StudentInfo.GroupName = BinaryFileToRead.ReadString();
            StudentInfo.GroupNumber = BinaryFileToRead.ReadInt32();
            StudentInfo.AverageMark = BinaryFileToRead.ReadDouble();
            InfoList.Add(StudentInfo);
        }
        double LowestAverageMark;
        int GroupNum;
        info LowestStudent;
        for (int i = 0; i < InfoList.Count;)
        {
            LowestAverageMark = InfoList[i].AverageMark;
            GroupNum = InfoList[i].GroupNumber;
            LowestStudent = InfoList[i];
            for(int j = 0; j < InfoList.Count;)
            {
                if(InfoList[j].GroupNumber == GroupNum)
                {
```

```
                    if(InfoList[j].AverageMark < LowestAverageMark)
                    {
                        LowestAverageMark = InfoList[j].AverageMark;
                        LowestStudent = InfoList[j];
                    }
                    InfoList.Remove(InfoList[j]);
                }
                else j++;
            }
            MainList.Add(LowestStudent);
        }
        return MainList;
    }
    1 usage
    public List<info> Identify(List<info> list)
    {
        List <info> list2 = new List<info>();
        Console.Write("Pick the average mark: ");
        double averageGrage = Convert.ToDouble(Console.ReadLine());
        for(int i = 0; i < list.Count;)
        {
            if(!(list[i].GroupNumber > 39 && list[i].GroupNumber < 50 && list[i].FormOfStudying == "Daily" && list[i].AverageMark > averageGrage) && list.Co
                list.Remove(list[i]);
            else i++;
        }

        if (list.Count() > 1)
        {
            list2 = list.OrderBy(i:info => i.name).ToList();
        }
        return list2;
    }
}
}
```

Python

Main.py

```python
import functions


path1 = input('Enter name of First file: ')
path2 = input('Enter name of Second file: ')

list_of_students = functions.creating_list_of_students(path1)
print('All students:')
functions.output_info_student(list_of_students)

readable_list_1 = functions.create_readable(path1)

worst_students = functions.worst_students(readable_list_1)
print('Worst student of each group: ')
functions.output_info_student(worst_students)

readable_list_2 = functions.create_readable(path1)

sorted_list_students = functions.sorted_file(readable_list_2, path2)
print('Sorted 4-th grade students: ')
functions.output_info_student(sorted_list_students)

```

Functions.py

```python
import pickle


class Student:
    def __init__(self, name, birthday, group_name, group_number, form_study, mark):
        self.name = name
        self.birthday = birthday
        self.group_name = group_name
        self.group_number = group_number
        self.form_study = form_study
        self.mark = mark


def create_readable(path):
    student_list = []
    with open(path, 'rb') as file:
        list_s = pickle.load(file)

    for student in list_s:
        student_list.append(student)
    return student_list


def creating_list_of_students(path):
    n = int(input("Enter amount of students: "))
    group_name = input('Enter the name group: ')
    students_list = list()
    for i in range(n):
        print(f"Student number {i + 1}")
        name = input('Enter name of student: ')
        birthday = input('Enter the date of birth: ')
        group_number = input('Enter number of group: ')
        form_study = input('Enter form of study (full-time / distance learning): ')
        mark = input('Enter average mark of student: ')
        student = Student(name, birthday, group_name, group_number, form_study, mark)
        students_list.append(student)
    with open(path, 'ab+') as file:
        pickle.dump(students_list, file)
```

```python
    return students_list


def worst_students(students_list):
    worst_mark_students = []

    while len(students_list) > 0:
        worst_mark = students_list[0].mark
        group_number = students_list[0].group_number
        worst_student = students_list[0]
        j = 0
        while j < len(students_list):
            if students_list[j].group_number == group_number:
                if students_list[j].mark <= worst_mark:
                    worst_mark = students_list[j].mark
                    worst_student = students_list[j]
                students_list.remove(students_list[j])
            else:
                j += 1
        worst_mark_students.append(worst_student)
    return worst_mark_students


def output_info_student(Students):
    for i in range(len(Students)):
        print(f'Name: {Students[i].name}, birthday: {Students[i].birthday}, group: {Students[i].group_name}-{Students[i].group_number}, form of study: {Students[i].form_study}, average mark: {Students[i].mark}')


def sorted_file(student_list, path2):
    n = float(input('Enter average mark: '))
    i = 0
    while i < len(student_list):
        if int(student_list[i].group_number) <= 39 or int(student_list[i].group_number) >= 50 or student_list[i].form_study != "full-time" or float(student_list[i].mark) < n and len(student_list) != 0:
            student_list.pop(i)
        else:
            i += 1

    if len(student_list) > 1:
        for i in range(len(student_list)):
```

```
            for j in range(i + 1, len(student_list)):
                if student_list[i].name[0] > student_list[j].name[0]:
                    student_list[i], student_list[j] = student_list[j], student_list[i]

    with open(path2, 'ab+') as file:
        pickle.dump(student_list, file)
    return student_list

```

Вивід C#

```
Enter name of First file
main.txt
Enter name of Second file:
sorted.txt
Enter amount of students:
4
Enter the name of the group: IP
Student's name: Bob
Date of Birth(DD.MM.YEAR): 10.10.1000
Form of studying(full-time / distance learning): full-time
Group number: 11
Average mark(0-100): 75

Student's name: Tom
Date of Birth(DD.MM.YEAR): 10.10.1000
Form of studying(full-time / distance learning): full-time
Group number: 41
Average mark(0-100): 50

Student's name: Danil
Date of Birth(DD.MM.YEAR): 10.10.1000
Form of studying(full-time / distance learning): full-time
Group number: 41
Average mark(0-100): 65

Student's name: Andrry
Date of Birth(DD.MM.YEAR): 10.10.1000
Form of studying(full-time / distance learning): full-time
Group number: 42
Average mark(0-100): 70

All students:
Name: Bob, birthday: 10.10.1000, group: IP-11, form of study: full-time, average mark: 75
Name: Tom, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 50
Name: Danil, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 65
Name: Andrry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
```

```
Lowest mark students:
Name: Bob, birthday: 10.10.1000, group: IP-11, form of study: full-time, average mark: 75
Name: Tom, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 50
Name: Andrry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
Pick the average mark: 60
4-th grade students:
Name: Andrry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
Name: Danil, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 65
```

Вівід Python

```
Enter name of First file: main.txt
Enter name of Second file: sorted.txt
Enter amount of students: 4
Enter the name group: IP
Student number 1
Enter name of student: Bob
Enter the date of birth: 10.10.1000
Enter number of group: 11
Enter form of study (full-time / distance learning): full-time
Enter average mark of student: 40
Student number 2
Enter name of student: Tom
Enter the date of birth: 10.10.1000
Enter number of group: 41
Enter form of study (full-time / distance learning): full-time
Enter average mark of student: 50
Student number 3
Enter name of student: Danil
Enter the date of birth: 10.10.1000
Enter number of group: 41
Enter form of study (full-time / distance learning): full-time
Enter average mark of student: 65
Student number 4
Enter name of student: Andry
Enter the date of birth: 10.10.1000
Enter number of group: 42
Enter form of study (full-time / distance learning): full-time
Enter average mark of student: 70
All students:
Name: Bob, birthday: 10.10.1000, group: IP-11, form of study: full-time, average mark: 40
Name: Tom, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 50
Name: Danil, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 65
Name: Andry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
Worst student of each group:
Name: Bob, birthday: 10.10.1000, group: IP-11, form of study: full-time, average mark: 40
Name: Tom, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 50
Name: Andry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
```

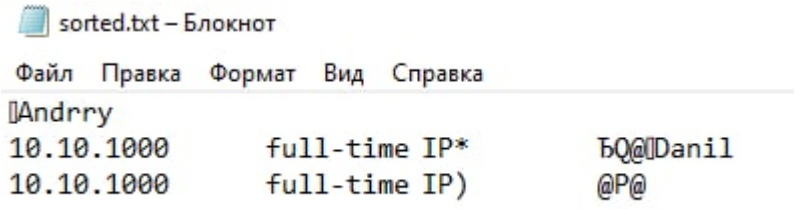
```
Enter average mark: 60
Sorted 4-th grade students:
Name: Andry, birthday: 10.10.1000, group: IP-42, form of study: full-time, average mark: 70
Name: Danil, birthday: 10.10.1000, group: IP-41, form of study: full-time, average mark: 65
```

Файли на C#

main.txt – Блокнот

Файл Правка Формат Вид Справка

```
Bob
10.10.1000      full-time IP        AR@Tom
10.10.1000      full-time IP)         I@Danil
10.10.1000      full-time IP)        @P@Andrry
10.10.1000      full-time IP*        ЬQ@
```



Файли на Python