

Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення



Звіт
Про виконання лабораторної роботи №8
на тему:
«Структури та об’єднання»
з дисципліни «Основи програмування»

Лектор:
ст. викл. каф. ПЗ
Муха Т.О.

Виконав:
ст. гр. ПЗ-11
Ясногородський Н.В.

Прийняв:
асист. каф. ПЗ
Дивак І.В.

« __ » _____ 2021 р.

Σ = _____ .

Львів – 2021

Тема: Структури та об'єднання

Мета: Навчитися створювати нові типи даних у вигляді структур та об'єднань, а також розробляти алгоритми їх обробки засобами мови С

ЗАВДАННЯ

Завдання 1.

З текстового файлу зчитати послідовність записів, які містять дані про результати сесії студентів групи у такому форматі: <Прізвище>, <Ім'я>, <Дата народження>, <Список екзаменаційних оцінок>. Роздрукувати введені дані у вигляді таблиці, а також подати інформацію згідно варіанту.

3. Відсортувати дані за віком студентів у зростаючому порядку. Роздрукувати список студентів з рейтинговим балом нижчим від середнього балу в групі.

ТЕКСТ ПРОГРАМИ

Завдання 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_STUDENTS_COUNT 100
#define MAX_LINE_LENGTH 512
#define BIRTH_DATE_LENGTH 14

struct Students
{
    struct Student
    {
        char name[30], surname[50], birthDate[14];
        struct Grades
        {
            int entries[20];
            int length;
            double avarage;
        } grades;
    } entries[MAX_STUDENTS_COUNT];

    int length;
    double avarageGrade;
};

struct Date
```

```

{
    int day;
    int month;
    int year;
};

void getStudentsFromFile(struct Students *students)
{
    FILE *file = fopen("students.txt", "r");

    char buffer[MAX_LINE_LENGTH];
    char comaDelim[] = ",";
    char spaceDelim[] = " ";

    int studentsCount = 0, gradesCount = 0, gradesSum = 0;
    double totalGradesSum = 0;

    while (fgets(buffer, MAX_LINE_LENGTH, file))
    {
        strcpy(students->entries[studentsCount].surname, strtok(buffer, comaDelim));
        strcpy(students->entries[studentsCount].name, strtok(NULL, comaDelim));
        strcpy(students->entries[studentsCount].birthDate, strtok(NULL, comaDelim));

        // read space-separated grades
        char *entry = strtok(NULL, comaDelim);
        entry[strcspn(entry, "\n")] = 0;
        entry = strtok(entry, spaceDelim);

        while (entry)
        {
            gradesSum +=
                (students->entries[studentsCount].grades.entries[gradesCount++] =
                 strtol(entry, 0, 10));
            entry = strtok(NULL, spaceDelim);
        }
        students->entries[studentsCount].grades.length = gradesCount;
        totalGradesSum += (students->entries[studentsCount].grades.average =
                           (double)gradesSum / gradesCount);

        gradesCount = 0;
        gradesSum = 0;
        studentsCount++;
    }
    students->length = studentsCount;
    students->averageGrade = totalGradesSum / studentsCount;
    fclose(file);
}

void getParsedDate(char *str, struct Date *date)

```

```

{
    const char dotDelim[] = ".";
    const char buffer[BIRTH_DATE_LENGTH];
    strcpy(buffer, str);

    date->day = strtol(strtok(buffer, dotDelim), 0, 10);
    date->month = strtol(strtok(NULL, dotDelim), 0, 10);
    date->year = strtol(strtok(NULL, dotDelim), 0, 10);
}

int studentsAgeComparator(const void *a, const void *b)
{
    struct Student *ia = (struct Student *)a;
    struct Student *ib = (struct Student *)b;
    struct Date dateA, dateB;

    getParsedDate(ia->birthDate, &dateA);
    getParsedDate(ib->birthDate, &dateB);

    if (dateA.year != dateB.year)
        return dateA.year - dateB.year;

    if (dateA.month != dateB.month)
        return dateA.month - dateB.month;

    if (dateA.day != dateB.day)
        return dateA.day - dateB.day;

    return 0;
}

void printStudents(struct Students *students)
{
    for (int i = 0; i < students->length; i++)
    {
        printf("Student %d:\n\tSurname: %s\n\tName: %s\n\tAvarage Grade: %.3lf\n\n",
            i + 1, students->entries[i].surname, students->entries[i].name,
            students->entries[i].grades.avarage);
    }
}

int main(void)
{
    printf("Task 3:\n\n");

    struct Students students;
    getStudentsFromFile(&students);

    // create students array, where each student has less than avarage grade

```

```

struct Students filteredStudents = {.length = 0};

for (int i = 0; i < students.length; i++)
{
    if (students.entries[i].grades.avarage < students.avarageGrade)
    {
        // shallow copy
        filteredStudents.entries[filteredStudents.length++] = students.entries[i];
    }
}

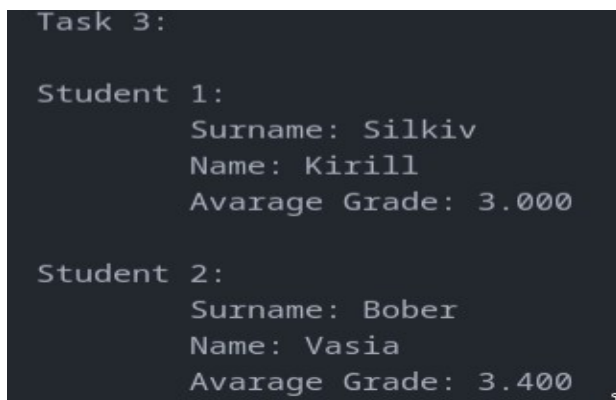
// sort by age
qsort(filteredStudents.entries, filteredStudents.length,
    sizeof(filteredStudents.entries[0]), studentsAgeComparator);

printStats(&filteredStudents);

return 0;
}

```

РЕЗУЛЬТАТИ



```

Task 3:

Student 1:
    Surname: Silkiv
    Name: Kirill
    Avarage Grade: 3.000

Student 2:
    Surname: Bober
    Name: Vasia
    Avarage Grade: 3.400

```

Рис 1. Результат виконання програми №1

ВИСНОВКИ

На даній лабораторній роботі створено програму, що зчитує послідовність записів та агрегує згідно завдання