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

Звіт
Про виконання лабораторної роботи №10
на тему:
«Динамічні структури даних»
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.

∑ = _____ _____ .

Львів – 2021

Тема: Динамічні структури даних

Мета: Оволодіти практичними прийомами створення та опрацювання динамічних списків.

ЗАВДАННЯ

Завдання 1.

Виконати завдання з лабораторної роботи № 11, організувавши послідовність структур в однозв'язний список. Реалізувати операцію вставки нового елемента у відсортований список і операцію вилучення зі списку даних, які відповідають одній з наступних умов:

3) про студентів, які не мають оцінки 5;

ТЕКСТ ПРОГРАМИ

Завдання 1

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

```
#include "./1.h"

int main(void)
{
   student *head = NULL;
   addStudentsFromFile(&head);

   printf("All students\n");
   printStudentsList(head);

   deleteAllStudentsMatching(&head, studentsGradesMatcher);

   printf("Students which have at least one 5 mark\n");
   printStudentsList(head);

   return 0;
}

student *createStudent()
{
   student *new_node = malloc(sizeof(student));
```

```
    new_node->next = NULL;
    return new_node;
}

void deleteStudent(student **head, student *studentToDelete)
{
    if (!*head || !studentToDelete)
        return;

    student *temp = NULL;

    if (*head == studentToDelete)
    {
        temp = *head;
        *head = (*head)->next;
        free(temp);
        return;
    }

    student *current = *head;
    while (current)
```

```
    {
        if (current->next == studentToDelete)
        {
            temp = current->next;
            current->next = current->next->next;
            free(temp);
            return;
        }
        current = current->next;
    }
}

void deleteAllStudentsMatching(student **head, int (*matcher)(student *current))
{
    student *iterator = *head;
    while (iterator)
    {
        if (matcher(iterator))
        {
            deleteStudent(head, iterator);
        }
```

```
            iterator = iterator->next;
        }
    }

    void insertStudentSorted(student **head, student *newStudent, int (*comparator)(student *a, student
*b))
    {
        // in case head is null or newStudent is "smaller" than head
        if (!*head || comparator(*head, newStudent) > 0)
        {
            newStudent->next = *head;
            *head = newStudent;
            return;
        }

        student *current = *head;
        while (current->next && comparator(current->next, newStudent) < 0)
        {
            current = current->next;
        }
        newStudent->next = current->next;
        current->next = newStudent;
```

```
}

void addStudentsFromFile(student **head)
{
   FILE *file = fopen("students.txt", "r");

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

   while (fgets(buffer, MAX_LINE_LENGTH, file))
   {
      student *newStudent = createStudent();
      strcpy(newStudent->surname, strtok(buffer, comaDelim));
      strcpy(newStudent->name, strtok(NULL, comaDelim));
      strcpy(newStudent->birthDate, strtok(NULL, comaDelim));

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

```
        int gradesCount = 0;
        int gradesSum = 0;
        while (entry)
        {
           gradesSum += (newStudent->grades.entries[gradesCount++] =
                        strtol(entry, 0, 10));
           entry = strtok(NULL, spaceDelim);
        }
        newStudent->grades.length = gradesCount;
        newStudent->grades.avarage = gradesCount ? gradesSum / gradesCount : 0;

        // insert
        insertStudentSorted(head, newStudent, studentsAgeComparator);
     }
     fclose(file);
}

void printStudentsList(student *head)
{
     student *iterator = head;
```

```
    int count = 0;

    printf("----------------------------------------\n");
    while (iterator)
    {
        printf("Student %d:\n\tSurname: %s\n\tName: %s\n\tBirth: %s\n\tAvarage Grade: %.3lf\n\n",
            ++count, iterator->surname, iterator->name, iterator->birthDate,
            iterator->grades.avarage);

        iterator = iterator->next;
    }
    printf("----------------------------------------\n");
}

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(student *a, student *b)
{
    struct Date dateA, dateB;

    getParsedDate(a->birthDate, &dateA);
    getParsedDate(b->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;
```

```
}

int studentsGradesMatcher(student *current)
{
   for (int i = 0; i < current->grades.length; i++)
   {
      if (current->grades.entries[i] == 5)
         return 0;
   }
   return 1;
}
```

File 1.h:

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

typedef struct Student
{
    char name[30], surname[50], birthDate[BIRTH_DATE_LENGTH];
    struct Grades
    {
        int entries[20];
        int length;
        double avarage;
    } grades;
    struct Student *next;
} student;

struct Date
{
    int day;
    int month;
    int year;
};

student *createStudent();
void insertStudentSorted(student **head, student *newStudent, int
(*comparator)(student *a, student *b));
void getParsedDate(char *str, struct Date *date);
int studentsAgeComparator(student *a, student *b);
void addStudentsFromFile(student **head);
void printStudentsList(student *head);
void deleteStudent(student **head, student *studentToDelete);
int studentsGradesMatcher(student *current);
void deleteAllStudentsMatching(student **head, int (*matcher)(student
*current));
```

РЕЗУЛЬТАТИ

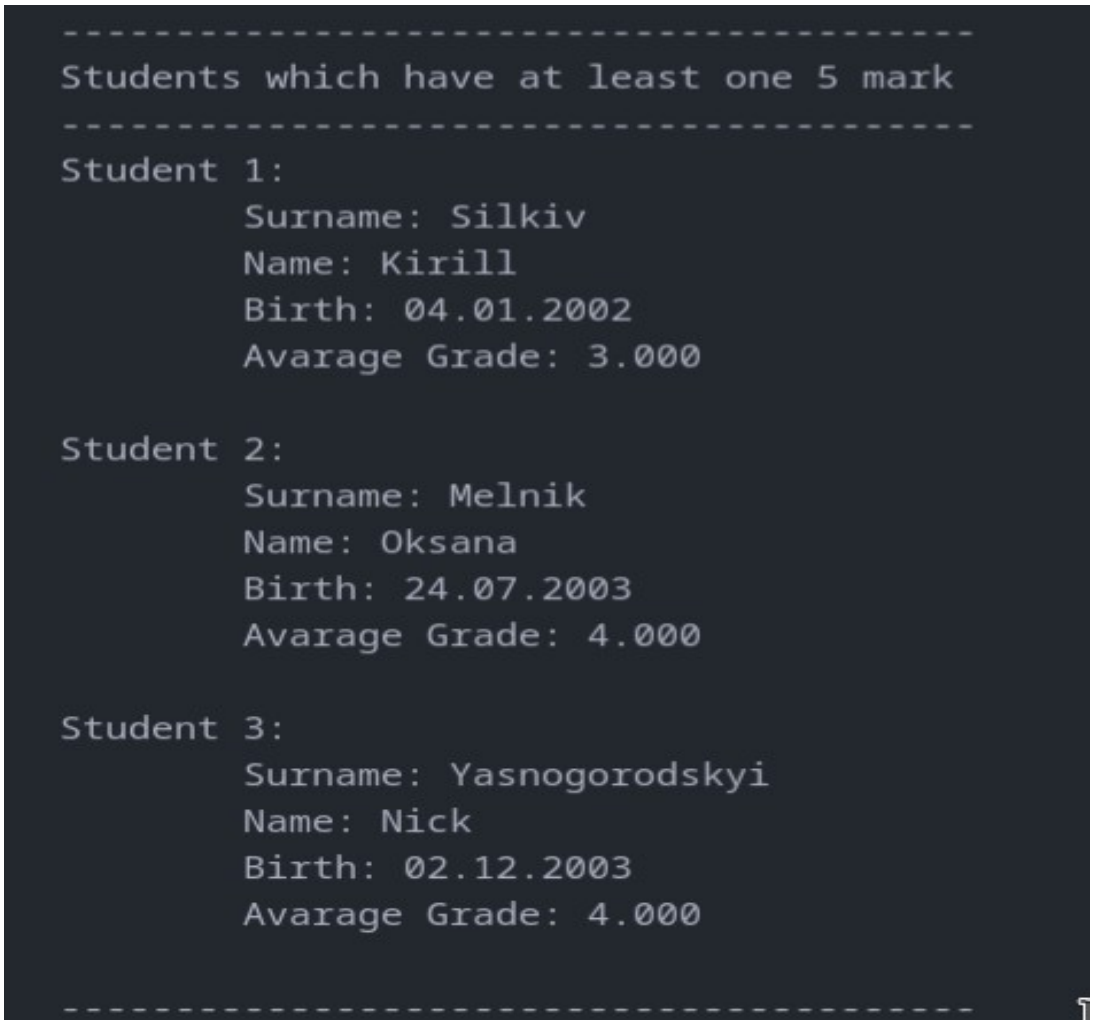

Рис 1. Результат виконання програми №1

ВИСНОВКИ

На даній лабораторній роботі здобуто практичні прийоми створення та опрацювання динамічних списків; створено програму, що считує дані з текстового файлу, формуючи однозв'язний список, друкує дані у вигляді таблиці