Індивідуальне завдання 1

з предмету алгоритми та структури даних

студента Чувайкін Данило Дмитрович

групи МФ-22

на тему «Односпрямовані списки»

Умова завдання:

Наказ про відрахування

Є набір текстових файлів – відомості з предметів. Ім'я файлу містить назву предмета, а файл містить набір рядків, у кожному з яких ПІБ студента та його оцінка з цього предмета. Потрібно створити інформацію для наказу на відрахування у вигляді списку списків; список ПІБ студентів, упорядкований за абеткою, і для кожного студента – список назв предметів, за якими у нього хвіст (менше 50 балів). Забезпечити операції додавання, видалення та розумного пошуку інформації, виведення інформації в розумному вигляді (наприклад, список всіх «хвостів» з конкретного предмету з кількістю балів).

Аналіз коду:

Загалом програма починається з підключення бібліотек:

- <stdio.h> для роботи з введенням та виведенням;
- <stdlib.h> для роботи з файловою системою та пам’яттю;
- <string.h> для кращої взаємодії з рядками;

Та зі створення директив #define:

- MAX_LINE_LEN 200 для максимальної довжина рядка;
- MAX_NAME_LEN 100 для максимальної довжини імені студента;
- MAX_SUBJ_LEN 50 для максимальної довжини назви предмета;
- MIN_SCORE 50 мінімальна оцінка для «хвосту»;
- SUBJECTS_FILE "subjects.txt" ім'я файлу для збереження списку предметів;

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

#define MAX_LINE_LEN 200
#define MAX_NAME_LEN 100
#define MAX_SUBJ_LEN 50
#define MIN_SCORE 50
#define SUBJECTS_FILE "subjects.txt"
```

Підключення бібліотек та створення директив #define

Далі створюються структури для зберігання інформації про студента та предмета:

SubjectGrade – містить назву предмета, оцінку студента та вказівник на оцінку з наступного предмета.

```c
typedef struct SubjectGrade{
    char subject[MAX_SUBJ_LEN];
    int grade;
    struct SubjectGrade* next;
} SubjectGrade;
```

Структура SubjectGrade

StudentRecord – містить ім’я студента, вказівник на список його оцінок та вказівник на наступного студента.

```c
typedef struct StudentRecord{
    char name[MAX_NAME_LEN];
    SubjectGrade* grades;
    struct StudentRecord* next;
} StudentRecord;
```

Структура StudentRecord

Далі розберемо головні функції програми:

Функція для очищення консолі

Використовує команду system з параметром “cls” для очищення консолі (працює тільки для операційної системи Windows)

```c
void clearConsole(){
    system("cls");
}
```

Тіло функції clearConsole

Функція нічого не повертає.

Функція для створення нового студенту:

```c
StudentRecord* createStudent(const char* name){
    StudentRecord* newStudent = (StudentRecord*)malloc(sizeof(StudentRecord));
    strncpy(newStudent->name, name, MAX_NAME_LEN - 1);
    newStudent->name[MAX_NAME_LEN - 1] = '\0';
    newStudent->grades = NULL;
    newStudent->next = NULL;
    return newStudent;
}
```

Тіло функції createStudent

Спочатку виділяємо пам'ять під нового студента, використовуючи функцію malloc, щоб виділити необхідний обсяг пам'яті для структури StudentRecord. Далі копіюємо ім’я студента в поле name за допомогою функції strncpy, щоб уникнути переповнення буфера. Забезпечуємо коректне завершення рядка, додавши символ закінчення рядка \0. Оцінки студента (вказівник grades) та вказівник на наступного студента (вказівник next) ініціалізуються як NULL. Функція повертає вказівник на створеного студента типу StudentRecord.

Функція повертає вказівник на створеного студента типу StudentRecord.

Функція для додавання оцінки для студента

```
void addGrade(StudentRecord* student, const char* subject, int grade){
    SubjectGrade* newGrade = (SubjectGrade*)malloc(sizeof(SubjectGrade));
    strncpy(newGrade->subject, subject, MAX_SUBJ_LEN - 1);
    newGrade->subject[MAX_SUBJ_LEN - 1] = '\0';
    newGrade->grade = grade;
    newGrade->next = student->grades;
    student->grades = newGrade;
}
```

Тіло функція addGrade

Функція додає нову оцінку для студента по певному предмету. Спочатку виділяється пам'ять під нову структуру SubjectGrade для оцінки. Потім копіюється назва предмета у відповідне поле структури. Оцінка студента зберігається у полі grade. Вказівник next структури SubjectGrade вказує на попередній список оцінок студента. В результаті оцінка додається на початок списку.

Функція для додавання студента у відсортований список

```
void addStudentSorted(StudentRecord** head, StudentRecord* student){
    if (*head == NULL || strcmp((*head)->name, student->name) > 0){
        student->next = *head;
        *head = student;
    }
    else{
        StudentRecord* current = *head;
        while (current->next != NULL && strcmp(current->next->name, student->name) < 0){
            current = current->next;
        }
        student->next = current->next;
        current->next = student;
    }
}
```

Тіло функції addStudentSorted

Функція додає студента в список студентів, при цьому список підтримує сортування за алфавітом. Якщо список порожній або ім'я студента

лексикографічно менше, ніж ім'я першого студента в списку, новий студент стає на початок списку. Якщо ім'я студента більша за поточне, проходимо список і додаємо студента після відповідного студента, зберігаючи порядок.

Функція нічого не повертає.

Функція для пошуку студента за ім'ям

```
StudentRecord* findStudent(StudentRecord* head, const char* name){
    while (head != NULL){
        if (strcmp(head->name, name) == 0){
            return head;
        }
        head = head->next;
    }
    return NULL;
}
```

Тіло функції findStudent

Функція шукає студента в списку студентів за ім'ям. Для цього вона послідовно перебирає список і порівнює ім’я студента з іменем кожного елемента в списку. Якщо студент знайдений, повертається вказівник на нього, якщо ні – повертається NULL.

Функція для отримання назви файлу без розширення (необхідне для зчитування з файлу, адже назва файлу без розширення – назва предмету, до якого відноситься файл)

```
void extractSubjectName(const char* filename, char* subject){
    strncpy(subject, filename, MAX_SUBJ_LEN - 1);
    subject[MAX_SUBJ_LEN - 1] = '\0';
    char* dot = strrchr(subject, '.');
    if (dot){
        *dot = '\0';
    }
}
```

Тіло функції extractSubjectName

Функція витягує назву предмета з імені файлу. Вона копіює назву файлу в змінну subject, потім шукає крапку в імені файлу, яка розділяє назву і розширення. Якщо така крапка знайдена, вона заміняється на символ закінчення рядка, тим самим відкидаючи розширення файлу.

Функція для зчитування файлу з оцінками для предмета

```c
void readSubjectFile(StudentRecord** head, const char* filename){
    FILE* file = fopen(filename, "r");
    if (file == NULL){
        puts("Error opening file");
        return;
    }
    char line[MAX_LINE_LEN];
    char name[MAX_NAME_LEN];
    int grade;
    char subject[MAX_SUBJ_LEN];
    extractSubjectName(filename, subject);

    while (fgets(line, sizeof(line), file)){
        line[strcspn(line, "\n")] = 0;

        char* lastSpace = strrchr(line, ' ');
        if (lastSpace == NULL){
            printf("Error in line format: %s\n", line);
            continue;
        }
        grade = atoi(lastSpace + 1);
        *lastSpace = '\0';
        strncpy(name, line, MAX_NAME_LEN);
        StudentRecord* student = findStudent(*head, name);
        if (student == NULL){
            student = createStudent(name);
            addStudentSorted(head, student);
        }
        addGrade(student, subject, grade);
    }
    fclose(file);
}
```

Тіло функції readSubjectFile

Функція читає файл, що містить дані про студентів і їхні оцінки. Спочатку вона відкриває файл для читання, після чого для кожного рядка визначає ім’я студента та оцінку по певному предмету. Для кожного студента додається предмет з оцінкою в його список. Якщо студент ще не існує, він створюється і додається в загальний список студентів.

Функція для виведення студентів з «хвостами»

```c
void printStudentsWithTales(StudentRecord* head){
    printf("List of students with failing grades:\n");
    while (head != NULL){
        int hasFails = 0;
        SubjectGrade* grade = head->grades;
        while (grade != NULL){
            if (grade->grade < MIN_SCORE){
                if (!hasFails){
                    printf("%s:\n", head->name);
                    hasFails = 1;
                }
                printf("  %s - %d\n", grade->subject, grade->grade);
            }
            grade = grade->next;
        }
        head = head->next;
    }
}
```

Тіло функції printStudentsWithTales

Функція виводить список студентів, які мають оцінки нижче мінімальної межі (50 балів). Для кожного студента перебираються всі його оцінки по предметах, і якщо оцінка нижча за мінімум, то виводиться відповідний запис.

Функція для додавання предмету до файлу з предметами

```c
void addSubjectToFile(const char* subject){
    FILE* subjectsFile = fopen(SUBJECTS_FILE, "a");
    if (subjectsFile){
        fseek(subjectsFile, 0, SEEK_END);
        long fileSize = ftell(subjectsFile);
        if (fileSize > 0){
            fprintf(subjectsFile, "\n%s", subject);
        }
        else{
            fprintf(subjectsFile, "%s", subject);
        }
        fclose(subjectsFile);
    }
}
```

Тіло функції addSubjectToFile

Функція додає новий предмет до файлу, що містить список всіх предметів, з якими працює програма. Спочатку вона перевіряє, чи існує файл з предметами. Якщо файл не існує, він буде створений. Потім функція відкриває файл для додавання нового предмета. Якщо предмет ще не був доданий, він записується в кінець файлу, при цьому кожен новий предмет додається з нового рядка.

Аналогічна функція додавання студента до файлу

```c
void addStudentToFile(StudentRecord** head){
    char name[MAX_NAME_LEN];
    char filename[MAX_SUBJ_LEN];
    int grade;
    printf("Enter the subject filename (e.g., math): ");
    fgets(filename, sizeof(filename), stdin);
    filename[strcspn(filename, "\n")] = '\0';
    char fullFilename[MAX_SUBJ_LEN + 5];
    snprintf(fullFilename, sizeof(fullFilename), "%s.txt", filename);
    FILE* file = fopen(fullFilename, "r");
    if (file == NULL){
        printf("File %s does not exist. Do you want to create it? (y/n): ", fullFilename);
        char response;
        scanf(" %c", &response);
        getchar();
        if (response == 'y' || response == 'Y'){
            file = fopen(fullFilename, "w");
            if (file == NULL){
                puts("Error creating file");
                return;
            }
            fclose(file);
            addSubjectToFile(filename);
        }
```

```
        else{
            printf("Operation cancelled.\n");
            return;
        }
    }
    else{
        fclose(file);
    }
    printf("Enter student's full name: ");
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = '\0';
    printf("Enter grade: ");
    scanf("%d", &grade);
    getchar();
    file = fopen(fullFilename, "a");
    if (file == NULL){
        puts("Error opening file");
        return;
    }
    // Пишемо з нового рядка тільки якщо файл не пустий
    fseek(file, 0, SEEK_END);
    long fileSize = ftell(file);
    if(fileSize > 0){
        fprintf(file, "\n%s %d", name, grade);
    }
    else{
        fprintf(file, "%s %d", name, grade);
    }
    fclose(file);
    StudentRecord* student = findStudent(*head, name);
```

```
    if(student == NULL){
        student = createStudent(name);
        addStudentSorted(head, student);
    }
    addGrade(student, filename, grade);
}
```

Тіло функції addStudentToFile

Функція додає нового студента до відповідного файлу предмета. Спочатку функція перевіряє, чи існує файл з оцінками по заданому предмету. Якщо такого файлу немає, користувачу пропонується створити його. Після цього програма запитує ім’я студента та його оцінку, додаючи новий запис в кінець файлу.

Функції для видалення студента з системи та файлу

```c
void deleteStudent(StudentRecord** head, const char* name, const char* filename){
    StudentRecord *current = *head, *prev = NULL;
    while (current != NULL && strcmp(current->name, name) != 0){
        prev = current;
        current = current->next;
    }
    if (current == NULL){
        printf("Student not found.\n");
        return;
    }
    if (prev == NULL){
        *head = current->next;
    }
    else{
        prev->next = current->next;
    }
    free(current);
    FILE* file = fopen(filename, "r");
    if (file == NULL){
        puts("Error opening file");
        return;
    }
    FILE* tempFile = fopen("temp.txt", "w");
    if (tempFile == NULL){
        puts("Error opening temporary file");
        fclose(file);
        return;
    }
    char line[MAX_LINE_LEN];
    while (fgets(line, sizeof(line), file)){
        if (strstr(line, name) == NULL || !strstr(line, " ")){
            fprintf(tempFile, "%s", line);
        }
    }
    fclose(file);
    fclose(tempFile);
    remove(filename);
    rename("temp.txt", filename);
    printf("Student %s deleted successfully.\n", name);
}
```

Тіло функції deleteStudent

Функція видаляє студента з загального списку та з файлу предмету. Спочатку вона шукає студента в списку, а потім видаляє його зі списку та звільняє пам'ять. Потім програма відкриває файл з оцінками по предмету і створює тимчасовий файл, у який переписуються всі записи, крім того, що містить

інформацію про видаленого студента. Після цього тимчасовий файл замінює оригінальний.

Функція для пошуку студентів з «хвостами»

```
void searchFailsBySubject(StudentRecord* head, const char* subject){
    printf("Students with failing grades in %s:\n", subject);
    int found = 0;
    while (head != NULL){
        SubjectGrade* grade = head->grades;
        while (grade != NULL){
            if (strcmp(grade->subject, subject) == 0 && grade->grade < MIN_SCORE){
                printf("  %s - %d\n", head->name, grade->grade);
                found = 1;
            }
            grade = grade->next;
        }
        head = head->next;
    }
    if (!found){
        printf("No failing students found for %s.\n", subject);
    }
}
```

Тіло функції searchFailsBySubject

Функція шукає студентів, які мають незадовільні оцінки по конкретному предмету. Програма перебирає список студентів, перевіряє кожну оцінку по заданому предмету, і якщо вона менша за мінімальну оцінку (50 балів), виводить ім’я студента та його оцінку.

Функція для зчитування предметів із файла subjects.txt

```
void loadSubjects(StudentRecord** head){
    FILE* subjectsFile = fopen(SUBJECTS_FILE, "r");
    if (subjectsFile){
        char subject[MAX_SUBJ_LEN];
        while (fgets(subject, sizeof(subject), subjectsFile)){
            subject[strcspn(subject, "\n")] = '\0';
            char filename[MAX_SUBJ_LEN + 5];
            snprintf(filename, sizeof(filename), "%s.txt", subject);
            readSubjectFile(head, filename);
        }
        fclose(subjectsFile);
    }
}
```

Тіло функції loadSubjects

Функція завантажує список всіх предметів з файлу subjects.txt, де містяться назви предметів, з якими працює програма. Програма відкриває файл для читання, зчитує кожен рядок (який містить назву предмета) і додає ці предмети в загальний список. Якщо файл не існує, функція виведе

повідомлення про помилку і припинить виконання. Функція повертає кількість предметів, що були завантажені.

Функція основного меню

```c
void menu(StudentRecord** head){
    int choice;
    char filename[MAX_SUBJ_LEN];
    char name[MAX_NAME_LEN];
    char subject[MAX_SUBJ_LEN];
    do{
        clearConsole();
        printf("\nHello! What do you want to do:\n");
        printf("1. Add a student\n");
        printf("2. Delete a student\n");
        printf("3. Show failing students\n");
        printf("4. Search fails by subject\n");
        printf("5. Exit\n");
        printf("Choose an option: ");
        scanf("%d", &choice);
        getchar();
```

```c
        switch (choice){
            case 1:
                addStudentToFile(head);
                break;
            case 2:
                printf("Enter student name to delete: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                printf("Enter subject filename (e.g., math): ");
                fgets(filename, sizeof(filename), stdin);
                filename[strcspn(filename, "\n")] = '\0';
                snprintf(subject, sizeof(subject), "%s.txt", filename);
                deleteStudent(head, name, subject);
                break;
            case 3:
                printStudentsWithTales(*head);
                break;
            case 4:
                printf("Enter subject name to search for failing students: ");
                fgets(subject, sizeof(subject), stdin);
                subject[strcspn(subject, "\n")] = '\0';
                searchFailsBySubject(*head, subject);
                break;
            case 5:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Try again.\n");
        }

        printf("\nPress Enter to continue...");
        getchar();

    } while (choice != 5);
```

Тіло функції меню (червоним виділені основні функції меню)

Функція відповідає за виведення меню програми і обробку вибору користувача. Вона дозволяє користувачеві виконувати різні дії, такі як:

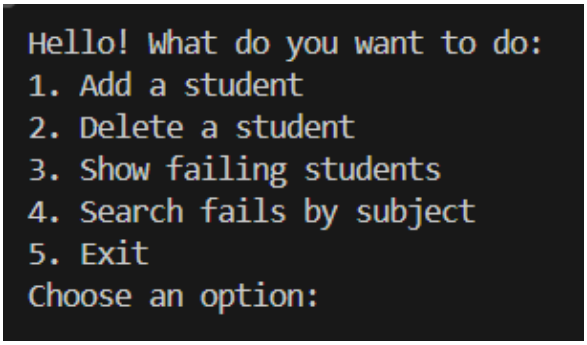
- додавання студента;

- видалення студента;
- перегляд студентів з невдалими оцінками;
- пошук студентів з невдалими оцінками за певним предметом;
- вихід з програми;

Функція викликає відповідні функції для кожної з цих операцій, а також забезпечує коректне оновлення екрану після кожної дії. Меню повторюється, поки користувач не вибере вихід.

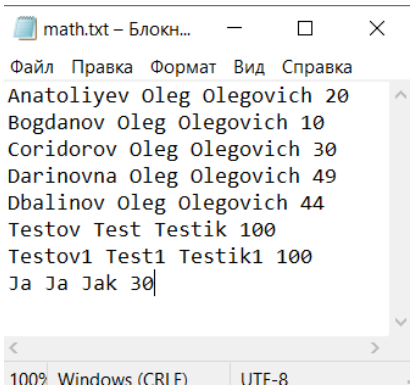
Приклад роботи програми:

При запуску програми отримуємо наступне меню:

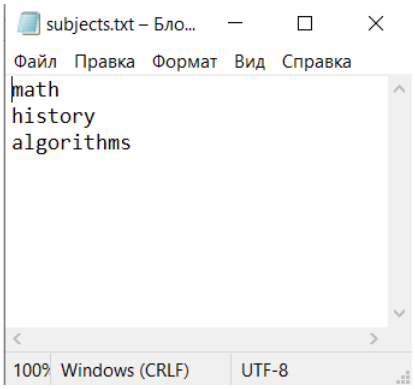


Меню при запуску програми

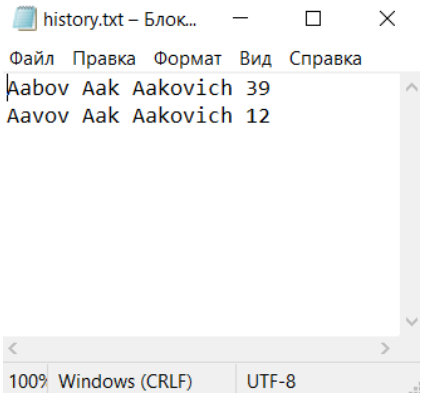
Для тестування вивдення у алфавітному порядку звіту для відрахування маємо такі файли з невідсортованими даними:



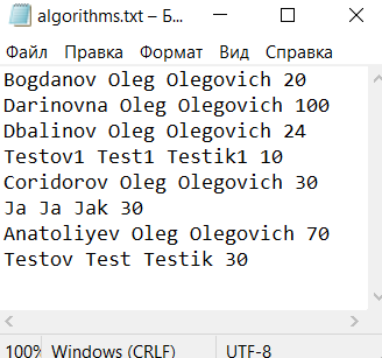
Файл math.txt



Файл subjects.txt (з предметами)



Файл history.txt



Файл algorithm.txt

Результат при пошуку студентів до відрахування:

```
Choose an option: 3
List of students with failing grades:
Aabov Aak Aakovich:
  history - 39
Aavov Aak Aakovich:
  history - 12
Anatoliyev Oleg Olegovich:
  math - 20
Bogdanov Oleg Olegovich:
  algorithms - 20
  math - 10
Coridorov Oleg Olegovich:
  algorithms - 30
  math - 30
Darinovna Oleg Olegovich:
  math - 49
Dbalinov Oleg Olegovich:
  algorithms - 24
  math - 44
Ja Ja Jak:
  algorithms - 30
  math - 30
Testov Test Testik:
  algorithms - 30
Testov1 Test1 Testik1:
  algorithms - 10

Press Enter to continue...
```

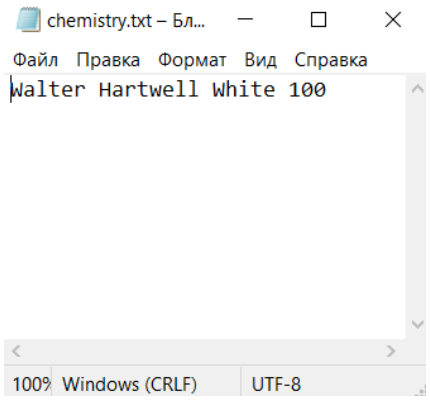
Результат виконання програми

Спробуємо додати студента та новий предмет:

```
Choose an option: 1
Enter the subject filename (e.g., math): chemistry
File chemistry.txt does not exist. Do you want to create it? (y/n): y
Enter student's full name: Walter Hartwell White
Enter grade: 100

Press Enter to continue...
```

Додавання студента зі створенням нового предмету



Новий створений файл зі студентом

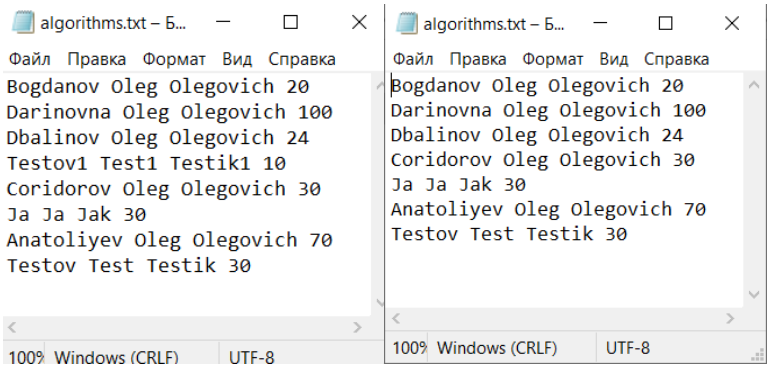
Спробуємо видалити студента:

```
Hello! What do you want to do:
1. Add a student
2. Delete a student
3. Show failing students
4. Search fails by subject
5. Exit
Choose an option: 2
Enter student name to delete: Testov1 Test1 Testik1
Enter subject filename (e.g., math): algorithms
Student Testov1 Test1 Testik1 deleted successfully.

Press Enter to continue...
```

Результат виконання програми

Як бачимо студент був видалений з файлу:



Файл до та після видалення

Пошук студентів за певним предметом до відрахування:

```
Choose an option: 4
Enter subject name to search for failing students: math
Students with failing grades in math:
  Anatoliyev Oleg Olegovich - 20
  Bogdanov Oleg Olegovich - 10
  Coridorov Oleg Olegovich - 30
  Darinovna Oleg Olegovich - 49
  Dbalinov Oleg Olegovich - 44
  Ja Ja Jak - 30
```

Результат пошуку