МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

КУРСОВА РОБОТА

з дисципліни «Програмування»

на тему:

Ведення обліку студентів

Виконав:
студент І курсу групи КМ-32,
спеціальність 113
Прикладна математика
Латко Назар Андрійович

Оцінка: _____

Кількість балів: ____

Київ - 2024

ВСТУП

У сучасних умовах навчальний процес постійно вдосконалюється, що вимагає відповідного підходу до організації обліку студентів. Викладачі та адміністрація навчальних закладів все частіше стикаються з необхідністю автоматизації процесів, пов'язаних із зберіганням, обробкою та аналізом інформації про студентів. Це завдання стає особливо актуальним у зв'язку зі зростанням кількості студентів і збільшенням обсягу інформації, яку потрібно обробляти.

Однією з основних задач в цьому процесі є створення ефективної системи ведення обліку студентів. Вона повинна забезпечувати швидкий доступ до даних, можливість їх зберігання, перезапису і видалення, а також надавати зручний інтерфейс для пошуку та управління інформацією про студентів. Важливим аспектом є забезпечення цілісності та захищеності даних, що зберігаються, а також підтримка актуальності інформації.

Однією з найбільш ефективних структур даних для вирішення подібних задач є 2-3 дерево. Ця структура дозволяє зберігати дані в збалансованому вигляді, що забезпечує високу швидкість виконання основних операцій (додавання, видалення, пошук). Використання 2-3 дерева забезпечує ефективну організацію даних та спрощує процеси управління ними.

Важливою метою даної курсової роботи є розробка програми для ведення обліку студентів з використанням 2-3 дерева як основної структури даних. Програма повинна забезпечувати можливість додавання нових записів, редагування та видалення існуючих, пошуку інформації за різними параметрами, а також надавати візуалізацію поточного стану дерева даних. Така програма значно полегшить роботу викладачів та адміністрації, забезпечить ефективний і зручний інструмент для управління інформацією про студентів.

Отже, метою даної курсової роботи є створення програми, яка дозволить автоматизувати процес ведення обліку студентів, забезпечуючи при цьому високу швидкість та ефективність роботи з даними.

Це завдання є актуальним у сучасних умовах і має значний потенціал для подальшого розвитку і вдосконалення

1 ПОСТАНОВКА ЗАДАЧІ

Мета роботи:

Створити програму для управління та аналізу даних про студентів навчального закладу.

Програма повинна виконувати такі поставлені задачі:

- Додавання студента. Записування його даних.
- Пошук студента.
- Видалення студента.
- Вивід дерева.
- Збереження даних.

Опис формату вхідних та вихідних даних:

Формат вхідних даних:

1) Ім'я - рядок. (char [])

2) Прізвище - рядок. (char [])

3) По-батькові - рядок. (char [])

4) ID - ціле число. (int)

5) Дата народження: рік, місяць, день - цілі числа. (int)

6) Середній бал - ціле число. (int)

7) Група - рядок. (char [])

8) Стать - символ. (char)

Формат вихідних даних:

1) Дані про студента. (Текст)

2) Список студентів з їх інформацією. (Текст)

3) Вигляд 2-3 дерева. (Схема вигляду)

Шлях введення вхідних даних: введення всіх даних здійснюється через консоль.

Шлях виведення вихідних даних: всі результати виводяться в консол

2 ВИБІР МЕТОДУ РОЗВ’ЯЗАННЯ

Чому я вибрав 2-3 дерево як структуру даних для вирішення моєї задачі?

Для розробки програми буде використано структуру даних 2-3 дерева. Ця структура забезпечує балансування дерева та швидкий доступ до даних. Ключовими особливостями 2-3 дерева є:

1. Кожен вузол містить один або два ключі.
2. Всі листки розташовані на одному рівні.
3. Дерево залишається збалансованим після кожної операції вставки або видалення.

Вершини нашого дерева буду представляти у вигляді 4-вершин (це коли вершина може мати 3 ключі та 4 сини). Дане рішення було обрано з кількох причин: по-перше, так легко зробити реалізацію, а по-друге, код і так сильно обрамлений іf'ами і це рішення дозволило зменшити кількість перевірок і спростити код.

Чому я вибрав бінарний файл для збереження?

Вибір бінарного формату для збереження даних програми обліку студентів був здійснений з кількох вагомих причин:

Ефективність зберігання та читання даних

Бінарні файли, на відміну від текстових, дозволяють зберігати дані в компактному форматі, що суттєво зменшує обсяг збереженої інформації. Це досягається за рахунок відсутності необхідності конвертувати дані у текстовий формат та назад при читанні.

Швидкість доступу

Зберігання даних у бінарному форматі значно підвищує швидкість їх читання та запису. Це особливо важливо для програм, що працюють з великим обсягом даних. У бінарних файлах відсутні проблеми, пов'язані з обробкою текстових рядків, таких як пропуски, форматування та конвертація числових значень.

Точність зберігання даних

Бінарні файли дозволяють зберігати дані в їх початковому форматі без втрат точності. Наприклад, числові значення з плаваючою точкою можуть втратити точність при конвертації в текстовий формат та назад. Використання бінарних файлів виключає такі проблеми.

Захист даних

Бінарні файли складніші для прочитання та модифікації без відповідного програмного забезпечення, що забезпечує додатковий рівень захисту даних. Це може бути корисним для збереження конфіденційної інформації, наприклад, даних про студентів.

Складні структури даних

Бінарні файли дозволяють зберігати складні структури даних, такі як об'єкти або структури, без необхідності конвертації в текстовий формат. Це значно спрощує процес збереження та відновлення даних у програмах, які використовують такі структури.

Я вважаю, що поєднання 2-3 дерев з бінарними файлами дозволяє досягти високої продуктивності та надійності програми.

Чому я вибрав (ID) як критерій для пошуку видалення і додавання?

Я вибрав використання ідентифікаційних номерів (ID) як спосіб відрізняти студентів з кількох важливих причин.

1) ID є унікальним для кожного студента. Це гарантує, що кожен студент має свій власний, неповторний номер, який відрізняє його від інших, навіть якщо інші характеристики, такі як ім'я чи прізвище, можуть співпадати. Це дуже важливо для уникнення плутанини та помилок при зберіганні та обробці даних.

2) Використання ID полегшує процес пошуку і сортування інформації. Працюючи з великою базою даних студентів, швидке та точне визначення потрібного запису за допомогою унікального номера є значно ефективнішим, ніж пошук за іншими параметрами, такими як ім'я чи група.

3) ID забезпечує стандартизацію та узгодженість в системі управління студентськими записами. Це дозволяє легко інтегруватися з іншими системами, які також використовують унікальні ідентифікатори для обробки даних.

Використання ID є безпечною практикою для захисту конфіденційності студентів. Замість використання особистої інформації, такої як ім'я чи дата народження, для ідентифікації студентів, ID дозволяє зберігати та обробляти дані більш анонімно, що є важливим аспектом при роботі з персональними даними.

З усіх цих причин, я вважаю, що використання ідентифікаційних номерів є найкращим способом для відрізнення студентів у системі.

3 ОПИС ПРОГРАМИ

3.1 Опис алгоритму

Алгоритм обраного методу подано на рис 1.

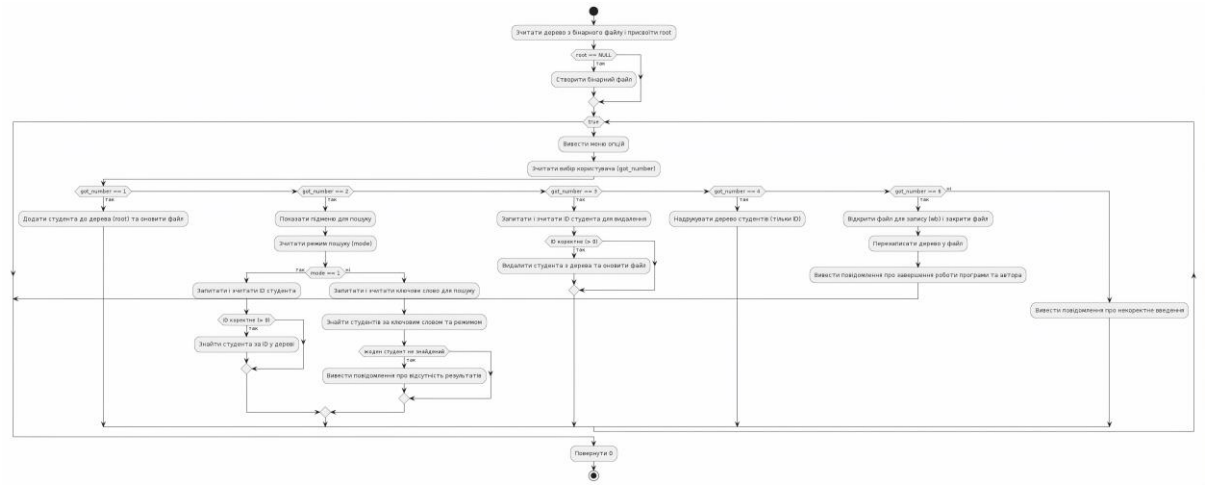


Рисунок 1 – Алгоритм обраного методу

1) Програма зчитує бінарний файл та передає дані у дерево
2) Якщо дерево порожнє: програма створює бінарний файл.
3) Програма виводить меню опцій
4) Програма зчитує вибір користувача (НОМЕР). (Користувач вибирає потрібну опцію)

4.1) Якщо НОМЕР = 1:

Користувач додає студента до дерева та програма оновлює бінарний файл

4.2) Якщо НОМЕР = 2:

4.2.1) Програма показує підменю для пошуку

4.2.2) Користувач вводить режим пошуку

4.2.3) Програма зчитує (Режим) та переходить до цього режиму пошуку:

Якщо (Режим)== 1:

Пошук по ідентифікатор (ID)

Інакше:

Пошук по ключовому слову (в одній з вибраних категорій)

4.3) Інакше якщо НОМЕР == 3:

4.3.1) Програма запитує і зчитує ID студента для видалення

4.3.2) Якщо студент існує - програма видаляє його. Якщо ні то виводить відповідне повідомлення

4.4) Інакше якщо НОМЕР == 4:

Програма друкує дерево студентів (тільки ID)

4.5) Інакше якщо НОМЕР == 6:

4.5.1) Програма перезаписує дерево у файл

4.5.2) Програма виводить повідомлення про завершення роботи програми та автора

4.5.3) Програма завершує роботу

3.2 Структура програми

Таблиця: Опис функцій

Назва функції	**Параметри**	**Призначення (результат роботи)**
split	- (Node *item) Вузол який треба розділити	- Повертає посилання на розділений вузол (той що має нащадків)
insert	- (Node *p) Посилання на вузол. (Найчастіше передається посилання на корінь дерева) - (student_info *k) Посилання на структуру з інформацією про студента	- Повертає корінь бо він може змінюватися
search	- (Node *p) Посилання на вузол. Найчастіше передається корінь. - (int k) число для пошуку	- Повертає знайдений вузол
search_min	- (Node *p) Посилання на вузол	- Повертає мінімальний елемент в правому піддереві
merge	- (Node *leaf) Посилання на листовий вузол	- Повертає батьківський вузол
redistribute	- (Node *leaf) Посилання на листовий вузол	- Робить перерозподіл ключів. Повертає батьківський вузол.
fix	- (Node *leaf) Посилання на листову вершину з якої потрбіно почати виправляти дерево	- Виправляє властивості дерева
delete_student	- `(Node *root) Посилання на корінь дерева` - `int id. Ключ за яким ми буемо шукати та видаляти студента`	- Видаляє студента.
remove_el	- (Node *p) Посилання на вузол - (student_info *k) Посилання на структуру з даними про студента	- Видаляє студента
delete_func	- (Node *node) Посилання на вузол.	- Видаляє студента

	Найчастіше передається корінь - (Int id) Ідентифікатор за яким шукаємо і видаляємо	
create_node	(student_info *k) Посилання на структуру з даними про студента.	- Створює вузол за даними.
bool find	- (Node *node) Вузол в якому шукаємо індекс - Int k - ігдекс за яким шукаємо	- Повертає так якщо знайдено і ні якщо не знайдено
swap	- student_info *x посилання на першу структуру - student_info *y Посилання на другу структуру	- Міняє місцями дві структури
sort2	- student_info *x посилання на першу структуру - student_info *y Посилання на другу структуру	- Сортує ключі у вузлі з двома ючами
sort3	- student_info *x - student_info *y - student_info *z Аналогічно до sort2 тільки додається 3 ключ	- Сортує ключі у вузлі з двома ючами
sort	- (Node *node) Посилання на вузол який треба відсортувати	- Сортує ключі у вузлі за допомогою sort2 і sort3
insert_to_node	- (Node *node) Вузол в який вставляємо ключ - (student_info *x) Посилання на структуру з інформацією про студента	- Вставляє ключ у вершину
remove_from_node	- (Node *node) Вузол в якому видаляємо ключ - (student_info *x) - ключ	- Видаляє ключ з вершини (не з дерева!!!)
become_node2	- (Node *node) - (student_info *x) - (Node *first) Посилання на лівого нащадка - (Node *second) Посилання на правого нащадка	- Повертає вузол
is_leaf	- (Node *node) - вузол	- Перевіряє чи є вузол листовим
recursivePrint	- (Node *node) вузол - Int depth - висота дерева	- Друкує дерево

getLevelCount	- (Node *node) - вузол (Найчастіше корінь!)	- Знаходить висоту дерева
Index_det	- (Node *node) - вузол - Int id - айді студента	- Знаходить і повертає індекс елемента за знайденим ключем
is_int_type	- (char *input) - рядок який перевіряємо чи є він числом	- Повертає число якщо вхідні дані були коректні
char_input	- (char **name) - рядок	- Функція для вводу рядка
int_input	немає	- Функція для вводу цілого числа. Повертає ціле число
check_id_list	- (int id) - айді	- Функція перевіряє чи зайняте айді
float_input	немає	- Функція для вводу числа з плаваючою комою. Повертає це саме число
data_input	немає	- Функція для вводу даних про користувача. - Повертає структуру з даними.
find_place	- (Node *node) - вузол - (int id) - айді	- Повертає індекс
write_to_binary	- (student_info *info) - Стурктура яку треба записати у файл - (char filename[]) - назва файлу	- Записує в бінарний файл
add_student	- (Node *node) - вузол. Найчастіше корінь. - (char filename[]) - назва файлу	- Додає студента. Додає ключ в дерево. Записує у файл
find_student	- (Node *node) - вузол - Int id - айді	- Шукає студента за його айді. Виводить його дані
read_from_binary_file	- (Node *root) - посилання на корінь - (char filename[]) - назва файлу	- Зчитує з файлу всі дані та передає їх в структури. Додає в дерево
Recursive_re_write	- (Node *node) - вузол. Найчастіше корінь - (int depth) - висота дерева - (char filename[]) - назва файлу	- Перезаписує всі дані
PrintAllFound	- (Node *node) - вузол. Найчатсіше корнь. - (int mode) - режим - char *keyword - ключове слово - int depth* - висота дерева - int *found - кількість знайдених елементів	- Знаходить і виводить всіх знайдених студентів за вказаним критерієм

Таблиця: опис основних змінних.

Назва об’єкта	**Опис**	**Призначення**
Id_list[1000]	- Int type.	Використовується для зберігання списка зайнятих ключів
List_len	- Int type	Кількість зайнятих айді
Struct date_of_birh	- Int year - Int month - Int day	Структура даних для збереження дат
Struct gender	- gender type - Male - Female	Структура для збереження статі
Struct node	- Node type: Int size Student info key[3] Struct node *first Struct node *second Struct node *third Struct node *fourth Struct node *parent	Структура вузла. Містить посилання на нащадків і посилання на батьківський вущол. Містить ключі з даними про студентів.
last_id	- Int type	Останнє використане айді
File_name[]	- Char	Назва файлу
Task_number	- char	Номер завдання записаний як рядок
id_num	- Char type	ID записаний як рядок
Got_number	- Int type	Номер завдання

4 ВІДЛАГОДЖЕННЯ

4.1 Перевірка виконання програми на витік пам’яті

```
Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 6
Thanks for using my program!
Author: Latko Nazar
==1349==
==1349== HEAP SUMMARY:
==1349==     in use at exit: 64 bytes in 1 blocks
==1349==   total heap usage: 31 allocs, 30 frees, 22,778 bytes allocated
==1349==
==1349== 64 bytes in 1 blocks are definitely lost in loss record 1 of 1
==1349==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==1349==    by 0x10BB14: data_input (main.c:679)
==1349==    by 0x10C10E: add_student (main.c:797)
==1349==    by 0x10CE7D: main (main.c:994)
==1349==
==1349== LEAK SUMMARY:
==1349==    definitely lost: 64 bytes in 1 blocks
==1349==    indirectly lost: 0 bytes in 0 blocks
==1349==      possibly lost: 0 bytes in 0 blocks
==1349==    still reachable: 0 bytes in 0 blocks
==1349==         suppressed: 0 bytes in 0 blocks
==1349==
==1349== For lists of detected and suppressed errors, rerun with: -s
==1349== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

В програмі я майже всюди використовував динамічне виділення пам’яті, що забезпечувало гнучкість у роботі з даними. Проте це ускладнило відстеження всіх виділених блоків пам'яті, особливо в місцях, де алгоритми вимагали частого створення і видалення тимчасових структур. Під час розробки я зосереджувався на функціональності та коректності алгоритмів, іноді нехтуючи ретельним відслідковуванням пам’яті. У поєднанні з обмеженим часом на тестування це призвело до виникнення витоків пам'яті.

5 ВИКОРИСТАННЯ ПРОГРАМИ

5.1 Інструкція для користувача

Запуск програми:

Після запуску користувач бачить головне меню де є 5 опцій:

```
Student record keeping program
Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number:
```

- 1) Користувач може додати студента

```
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 1
Please enter student's first name: Nazar
Please enter student's patronimic: Andriyovich
Please enter student's sursname: Latko
Please enter student's date of birth:
Id: 2006
Year: 2006
Month: 02
Day: 14
Please enter student's group: km-32
Enter student's rating score: 83.2
Please enter student's gender (M/m) or (F/f): m
```

- Користувач може шукати студента за певними критеріями:

  Пошук по ID:

```
1) Search by id - 1
2) Search by name - 2
3) Search by surname - 3
4) Search by group - 4
Enter the option number: 1
Enter the student's id: 2006

Student is found:
Latko Nazar Andriyovich
Student's id: 2006
Date of birth: 14 2 2006
Group: km-32
Rating score: 83.20
Gender: Male
```

  Пошук по імені:

```
Enter the option number: 2
1) Search by id - 1
2) Search by name - 2
3) Search by surname - 3
4) Search by group - 4
Enter the option number: 2
Enter keyword: Nazar
Student is found:
Latko Nazar Andriyovich
Student's id: 2006
Date of birth: 14 2 2006
Group: km-32
Rating score: 83.20
Gender: Male
```

  Пошук по прізвищу:

```
Enter the option number: 2
1) Search by id - 1
2) Search by name - 2
3) Search by surname - 3
4) Search by group - 4
Enter the option number: 3
Enter keyword: Latko
Student is found:
Latko Nazar Andriyovich
Student's id: 2006
Date of birth: 14 2 2006
Group: km-32
Rating score: 83.20
Gender: Male
```

Пошук за групою:

```
Enter the option number: 2
1) Search by id - 1
2) Search by name - 2
3) Search by surname - 3
4) Search by group - 4
Enter the option number: 4
Enter keyword: km-32
Student is found:
Latko Nazar Andriyovich
Student's id: 2006
Date of birth: 14 2 2006
Group: km-32
Rating score: 83.20
Gender: Male
```

- Користувач може видаляти студента за його ID:

  Спочатку подивимось на вигляд дерева:

```
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 4
    1 2006
```

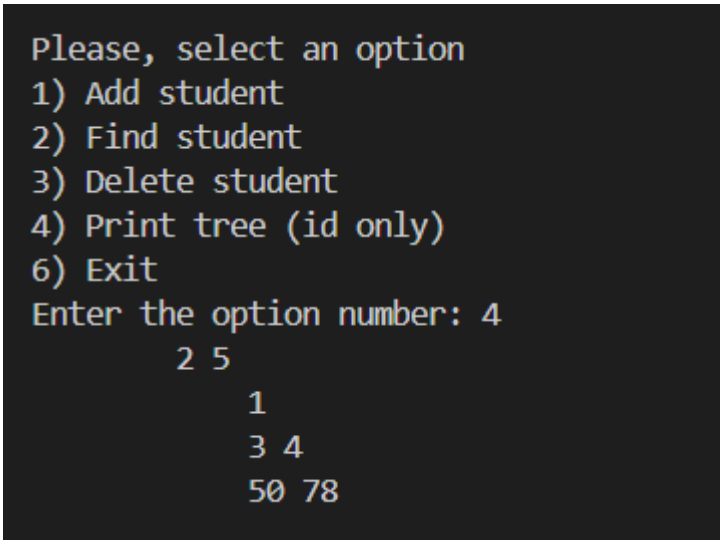
Тепер видалимо студента з ID 2006:

```
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 3
Enter the student's id: 2006
Found node. Removing...

Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 4
    1
```

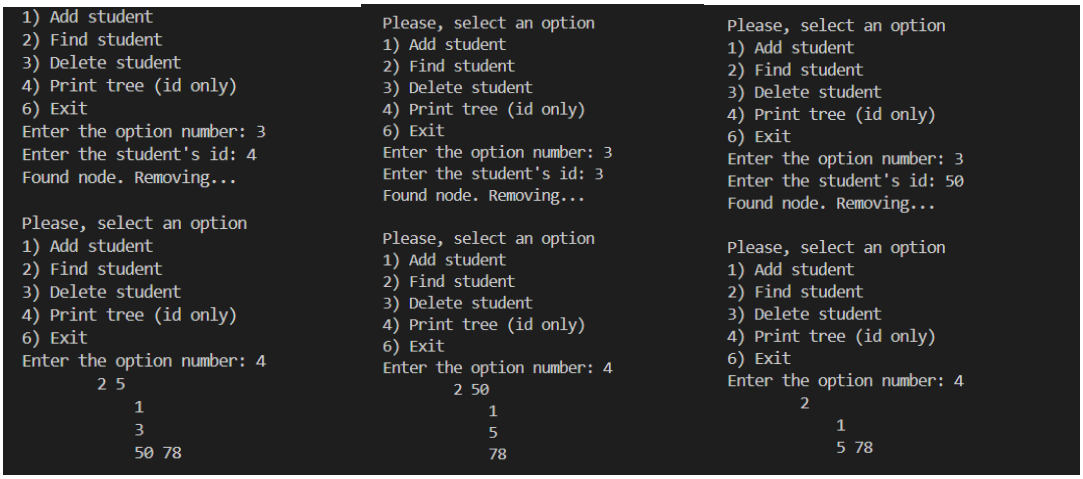
Бачимо що студент був видалений.

- Користувач може бачити вигляд дерева (для зручності виводяться тільки ключі). Я додав декілька студентів щоб показати що дерево балансується:



5.2 Перевірка що алгоритм працює правильно.

Видалимо по черзі декілька ключів:



Додаймо декілька ключів:

```
Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 1
Please enter student's first name: 44
Please enter student's patronimic: 44
Please enter student's sursname: 44
Please enter student's date of birth:
Id: 44
Year: 44
Month: 44
Day: 44
Please enter student's group: 44
Enter student's rating score: 44
Please enter student's gender (M/m) or (F/f): f

Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 4
        2 44
            1
            5
            78
```

```
Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 1
Please enter student's first name: Artem
Please enter student's patronimic: Andriyovich
Please enter student's sursname: Latko
Please enter student's date of birth:
Id: 3
This id is not available now! Choose another one
Id: 4
This id is not available now! Choose another one
Id: 8
Year: 2006
Month: 02
Day: 14
Please enter student's group: km-32
Enter student's rating score: 82
Please enter student's gender (M/m) or (F/f): f

Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 4
        2 44
            1
            5 8
            78
```

!!! ID оновлюються при запуску програми тому їх не можна використати до того як програма знову запуститься !!!

Тепер покажемо, що пошук всіх студентів за критерієм працює правильно (Я додав студента прізвищем Latko знову, бо раніше студент з таким прізвищем був видалений):

```
Please, select an option
1) Add student
2) Find student
3) Delete student
4) Print tree (id only)
6) Exit
Enter the option number: 2
1) Search by id - 1
2) Search by name - 2
3) Search by surname - 3
4) Search by group - 4
Enter the option number: 3
Enter keyword: Latko
Student is found:
Latko Artem Andriyovich
Student's id: 8
Date of birth: 14 2 2006
Group: km-32
Rating score: 82.00
Gender: Female

Student is found:
Latko Nazar Andriyovich
Student's id: 48
Date of birth: 14 2 2006
Group: km-32
Rating score: 82.00
Gender: Female
```

Висновки

Поставлену мету було досягнено, оскільки було реалізовано мету курсової роботи - реалізація програми про ведення обліку студентів. Результати тестового прикладу свідчать про те, що програма працює коректно.
Програма може виконувати такі поставлені задачі як:

- Додавання студента.
- Пошук студентів за критеріями.
- Видалення студента.
- Показ як виглядає дерево під час роботи програми.
- Збереження даних.

Переваги програми:

- Гнучкість: Використання динамічного виділення пам'яті дозволяє гнучко керувати обсягом і структурою даних, що полегшує роботу з різноманітними вхідними даними та виконанням різноманітних завдань.
- Ефективність: Динамічне виділення пам'яті може сприяти ефективному використанню ресурсів системи, забезпечуючи виділення пам'яті лише тоді, коли це необхідно, і звільнення її після закінчення використання.
- Розширюваність: Завдяки можливості динамічного виділення пам'яті програма може легко розширюватися та адаптуватися до нових вимог і сценаріїв використання без значних змін у вихідному коді. Є можливість додати купу нових функцій в майбутньому, які покращать дану програму і зроблять її більш зручною для навчальних закладів.

Недоліки програми:

- Витоки пам'яті: Використання динамічного виділення пам'яті може призвести до витоків пам'яті, особливо якщо некоректно керувати цим процесом. Недостатній контроль за виділенням і звільненням пам'яті може призвести до витрати ресурсів та непередбачуваної поведінки програми.
- Складність відладки: Пошук і виправлення помилок, пов'язаних з динамічним виділенням пам'яті, може бути складним завданням через складність відстеження життєвого циклу виділених об'єктів і виявлення місць, де відбувається витік пам'яті або некоректне звільнення ресурсів. Власне я сам витрачав купу часу щоб виправити прості помилки.

Отже програма може бути використана навчальними закладами, для полегшення ведення обліку студентів.

Перелік посилань

1) https://www.geeksforgeeks.org/2-3-trees-search-and-insert/
2) https://www.eecs.yorku.ca/~kamalis/Teaching/Winter23/2c-b-trees.pdf
3) https://www.cse.iitd.ac.in/~mausam/courses/col106/autumn2017/lectures/10-234trees.pdf
4) https://www.youtube.com/live/JHitk1lq-NM?si=647NVS8YXCSxlgdt

Додаток А. Тестування функціоналу для інших студентів

Я тестував код Піскуна Михайла Володимировича, тема його курсової роботи «Складання розкладу занять»

Я виявив такі проблеми в роботі програми:

1)

Problem with managing the number of subjects

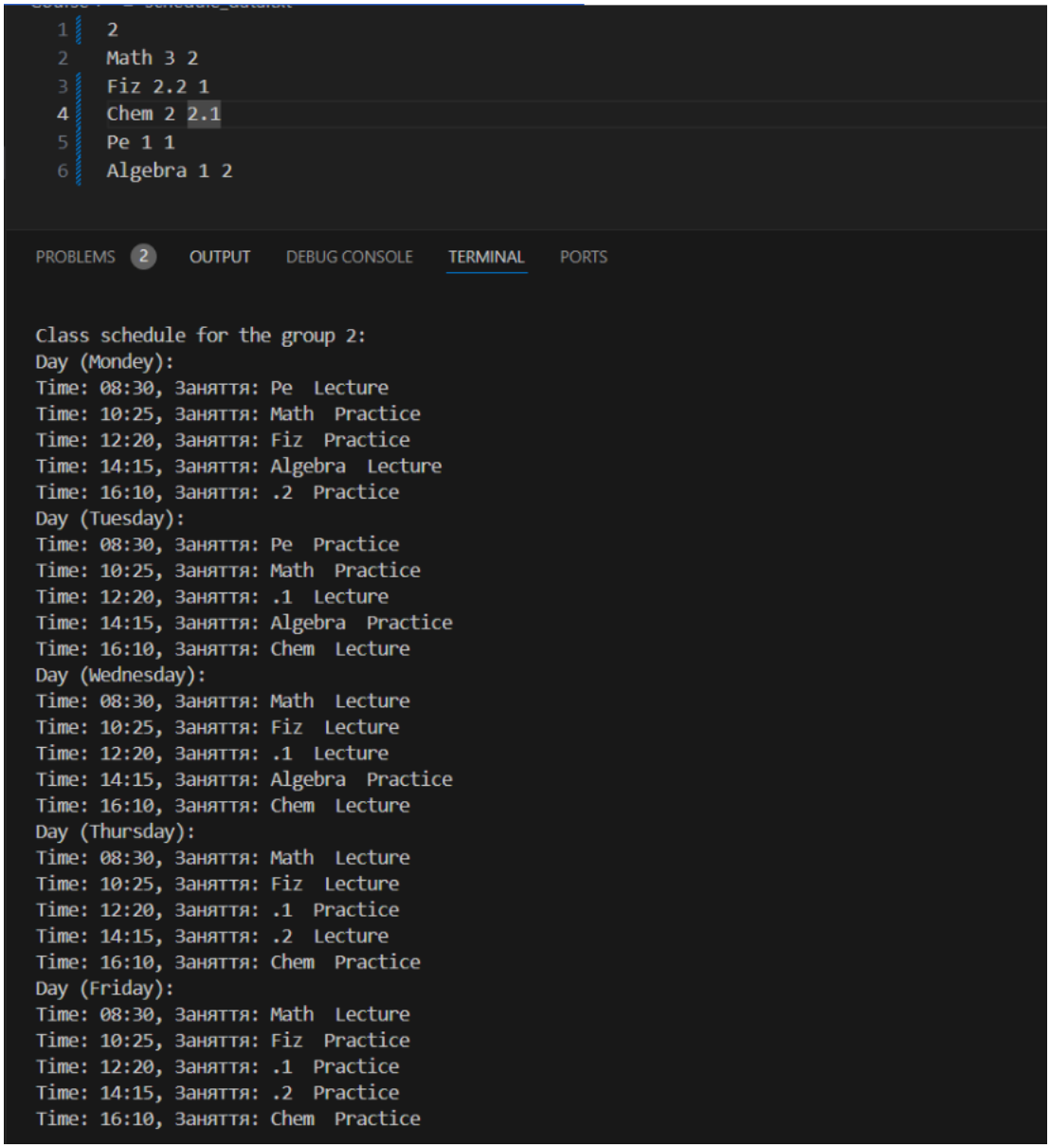
Your program can managing only 35 couples per week (from 8:30 to 20:00 = 7. 7*5 = 35). If there's more than 35 subjets, program works but no correctly as it should work. Example: There's 3 algebra per week but 2 is available in schedule.

```
2
Math 3 2
Fiz 3 1
Chem 2 20
Pe 1 1
Algebra 1 2
```

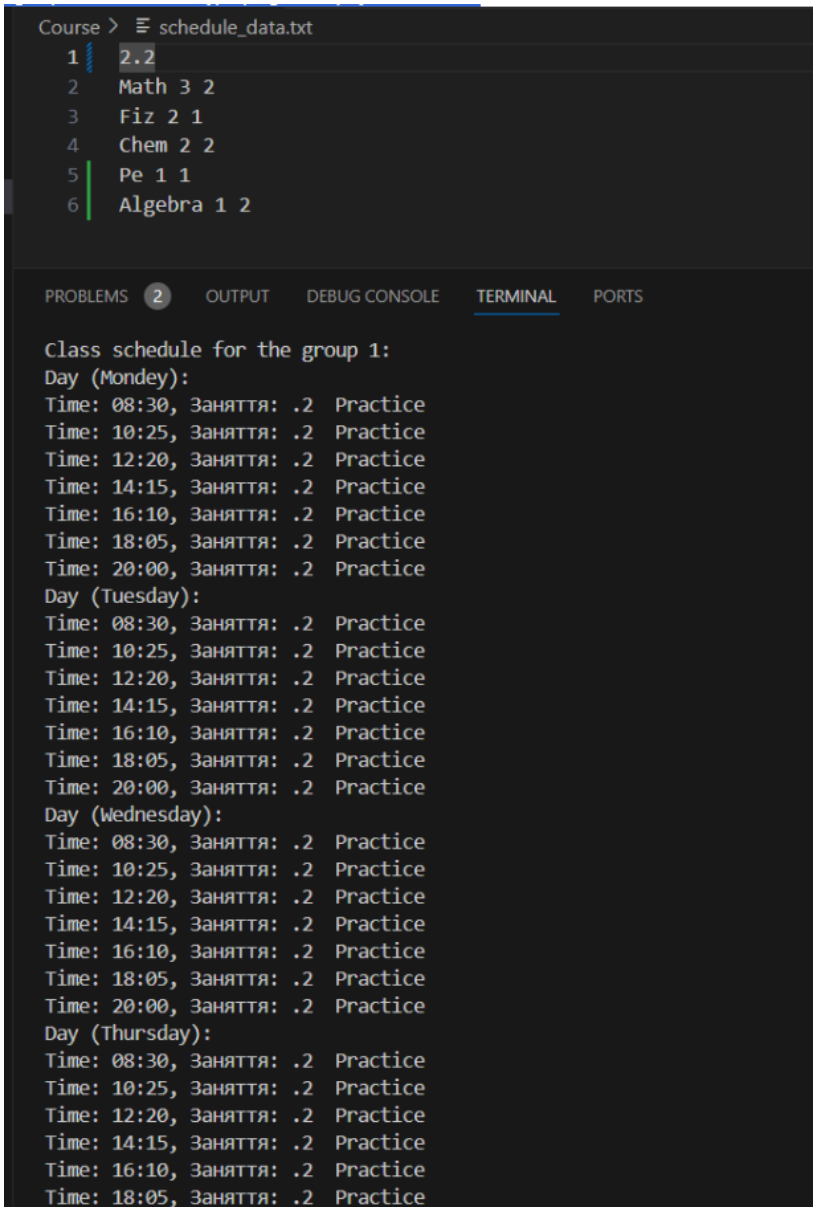
```
Class schedule for the group 1:
Day (Mondey):
Time: 08:30, Заняття: Math  Lecture
Time: 10:25, Заняття: Pe  Lecture
Time: 12:20, Заняття: Fiz  Practice
Time: 14:15, Заняття: Chem  Practice
Time: 16:10, Заняття: Chem  Practice
Time: 18:05, Заняття: Chem  Practice
Time: 20:00, Заняття: Chem  Practice
Day (Tuesday):
Time: 08:30, Заняття: Math  Lecture
Time: 10:25, Заняття: Pe  Practice
Time: 12:20, Заняття: Chem  Lecture
Time: 14:15, Заняття: Chem  Practice
Time: 16:10, Заняття: Chem  Practice
Time: 18:05, Заняття: Chem  Practice
Time: 20:00, Заняття: Chem  Practice
Day (Wednesday):
Time: 08:30, Заняття: Math  Lecture
Time: 10:25, Заняття: Fiz  Lecture
Time: 12:20, Заняття: Chem  Lecture
Time: 14:15, Заняття: Chem  Practice
Time: 16:10, Заняття: Chem  Practice
Time: 18:05, Заняття: Chem  Practice
Time: 20:00, Заняття: Chem  Practice
Day (Thursday):
Time: 08:30, Заняття: Math  Practice
Time: 10:25, Заняття: Fiz  Lecture
Time: 12:20, Заняття: Chem  Practice
Time: 14:15, Заняття: Chem  Practice
Time: 16:10, Заняття: Chem  Practice
Time: 18:05, Заняття: Chem  Practice
Time: 20:00, Заняття: Algebra  Lecture
Day (Friday):
```

Програма неправильно поділяла предмети на тиждень якщо загальна кількість предметів була більшою за час який виділявся на тиждень.

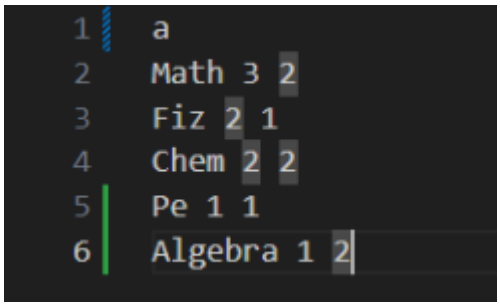
2) Проблема з обробкою кількості лекцій та практик: якщо кількість лекцій або практики є число з плаваючою комою, програма вважає це назвою предмета.

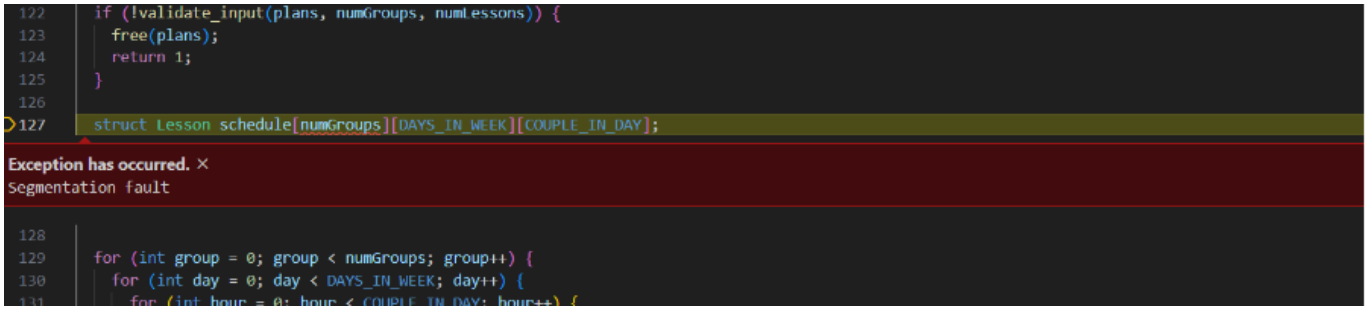


3) Неправильний вивід із номером групи з плаваючою комою: якщо номер групи має тип float, програма відображає не зрозуміло що.



4) Немає перевірки номера групи: Якщо номер групи є буквою або від'ємним числом, програма покаже помилку та припинить роботу.





Додаток Б. Код-рев’ю для інших студентів

Тестуючи код, я виділив декілька моментів, які б на мою думку полегшили роботу з програмою:

1) Якщо файлу не існує, я б додав можливість ввести дані вручну та зберегти їх у файлі .txt. Думаю, програмою буде зручніше користуватися.

2) int lessonIndices[numLessons]" вираз повинен мати постійне значення. Роботі програми це не заважає, але я вважаю що при розширенні кода це може призвести до виходу за межі допустимої області.