МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный исследовательский университет)

Институт №8

«Компьютерные науки и прикладная математика»

Кафедра 806

«Вычислительная математика и программирование»

Курсовой проект по дисциплине «Фундаментальные алгоритмы»

Тема: «Разработка алгоритмов системы хранения и управления данными на основе динамических структур данных»

Студент: Мазурец Кирилл Александрович

Группа: М8О-211Б-21

Преподаватель: Ирбитский И.С.

Оценка:

Дата:

Москва, 2023

Введение

Задание курсового проекта:

На языке программирования C++ (стандарт C++14 и выше) реализуйте приложение, позволяющее выполнять операции над коллекциями данных заданных типов (типы обслуживаемых объектов данных определяются вариантом) и контекстами их хранения (коллекциями данных). Коллекция данных описывается набором строковых параметров (набор параметров однозначно идентифицирует коллекцию данных):

- название пула схем данных, хранящего схемы данных;
- название схемы данных, хранящей коллекции данных;
- название коллекции данных.

Коллекция данных представляет собой ассоциативный контейнер (конкретная реализация определяется вариантом), в котором каждый объект данных соответствует некоторому уникальному ключу. Для ассоциативного контейнера необходимо вынести интерфейсную часть (в виде абстрактного класса C++) и реализовать этот интерфейс. Взаимодействие с коллекцией объектов происходит посредством выполнения одной из операций над ней:

- добавление новой записи по ключу;
- чтение записи по её ключу;
- чтение набора записей с ключами из диапазона [minbound... maxbound];
- обновление данных для записи по ключу;
- удаление существующей записи по ключу.

Во время работы приложения возможно выполнение также следующих операций:

- добавление/удаление пулов данных;
- добавление/удаление схем данных для заданного пула данных;
- добавление/удаление коллекций данных для заданной схемы данных заданного пула данных.

Поток команд, выполняемых в рамках работы приложения, поступает из файла, путь к которому подаётся в качестве аргумента командной строки. Формат команд в файле определите самостоятельно.

Дополнительные задания, реализованные в курсовом проекте:

- Реализуйте интерактивный диалог с пользователем. Пользователь при этом может вводить конкретные команды (формат ввода определите самостоятельно) и подавать на вход файлы с потоком команд;

- Реализуйте механизмы сохранения состояния системы хранения данных в файловую систему и восстановления состояния системы хранения данных из файловой системы.

Вариант курсового проекта 3:

Тип данных:

Информация о прохождении контеста соискателем (id соискателя, ФИО соискателя (раздельные поля), дата рождения соискателя, ссылка на резюме соискателя, id закреплённого за соискателем HR-менеджера, id контеста, ЯП на котором реализуются задачи контеста, кол-во задач в контесте, кол-во решённых задач в контесте, было ли обнаружено списывание с микронаушником).

Контейнер: косое дерево

IPC: Unix file mapping

Описание реализованного приложения

Реализованное в ходе курсового проекта приложение позволяет создавать, добавлять, хранить, искать и удалять записи данных о прохождении контеста соискателем.

Приложение включает в себя несколько классов, которые, взаимодействуя между собой, обеспечивают полноценную работу программы, хранение данных и обработку запросов. Разберём их более подробно.

1. AbstractCollection:

Это абстрактный базовый класс, предоставляющий интерфейс для работы с коллекцией данных. Он параметризован двумя типами данных, k для ключей и v для значений. Этот класс определяет чисто виртуальные методы, которые должны быть реализованы в классах-наследниках:

- addRecord: Добавляет запись в коллекцию с указанным ключом и значением.

- getRecord: Возвращает указатель на запись в коллекции по указанному ключу. Если запись не найдена, возвращается nullptr.

- getRecordsInRange: Возвращает вектор записей, ключи которых находятся в указанном диапазоне.

- updateRecord: Обновляет запись в коллекции с указанным ключом новым значением.

- deleteRecord: Удаляет запись из коллекции по указанному ключу.

2. SplayNode (Узел косого дерева):

Это структура данных, представляющая узел косого дерева. Каждый узел содержит ключ k, значение v, указатели на родителя, левого и правого потомка. Эта структура используется для построения косого дерева.

3. SplayTree (Косое дерево):

Этот класс наследуется от AbstractCollection и реализует ассоциативный контейнер с использованием косого дерева. Он содержит приватное поле root, представляющее корень дерева, и реализует методы из интерфейса AbstractCollection. Опишем его методы:

- rotateLeft: Поворачивает указанный узел влево (одна из операций для поддержания баланса косого дерева).

- rotateRight: Поворачивает указанный узел вправо (одна из операций для поддержания баланса косого дерева).

- splay: Выполняет операцию "сплей" для указанного узла, перемещая его ближе к корню дерева.

- find: Рекурсивно ищет узел с указанным ключом и выполняет операцию "сплей" для найденного узла.

- insert: Вставляет новый узел с указанным ключом и значением в дерево.

- deleteNode: Удаляет узел с указанным ключом из дерева.

- getRecordsInRange: Рекурсивно собирает записи в диапазоне ключей и добавляет их в вектор result.

- inOrderPrintKeys: Рекурсивно обходит дерево в порядке возрастания ключей и выводит их на экран.

- clearTree: Рекурсивно очищает память, освобождая узлы дерева.

Кроме выше перечисленных методов, класс SplayTree также реализует методы из интерфейса AbstractCollection.

Косое дерево это это самобалансирующееся бинарное дерево поиска. Дереву не нужно хранить никакой дополнительной информации, что делает его эффективным по памяти. После каждого обращения, косое дерево меняет свою структуру. Подъем реализуется через повороты вершин. За один поворот, можно поменять местами родителя с потомком. Но просто поворачивать вершину, пока она не станет корнем, недостаточно. При продвижении вершины вверх, расстояние до корня сокращается не только для поднимаемой вершины, но и для всех ее потомков в текущих поддеревьях. Для этого используется техника “zig-zig” и “zig-zag” поворотов. Основная идея “zig-zig” и “zig-zag” поворотов, рассмотреть путь от дедушки к ребенку. Если путь идет только по левым детям или только по правым, то такая ситуация называется “zig-zig”. Для её обработки необходимо сначала повернуть родителя, потом ребенка. В противном случае, следует сначала поменять ребенка с текущим родителем, потом с новым - такая ситуация называется “zig-zag”. Если у вершины дедушки нет, производится обычный поворот.

Процедура поиска в косом дереве заключается в сравнении искомого ключа с ключом рассматриваемого узла. Если искомый ключ меньше рассматриваемого, переходим к рассмотрению левого поддерева, иначе - правого поддерева. После того, как вершина найдена, она поднимается вверх и становится корнем через процедуру ”сплей”.

Процедура вставки в косое дерево схожа с процедурой поиска и заключается в переходе по поддеревьям, чтобы найти оптимальное место для вставки нового узла. После нахождения ближайшего родителя, новый узел вставляется в свободное место слева, или справа от найденного узла в зависимости от того, меньше ли вставляемый узел, или больше. Для вставленного узла выполняется процедура “сплей”.

Удаление из косого дерева имеет три различных сценария. В каждом из них первый этап - нахождение удаляемого узла. Если удаляемый узел не имеет правого и левого потомков, он удаляется. Если удаляемый узел имеет только одного потомка, узел удаляется, а на его место встает его потомок. Если удаляемый узел имеет обоих потомков, он удаляется, а на его место встает

крайний левый лист его правого поддерева. После первых двух случаев, операция “сплей” выполняется для родителя удаляемого узла. После третьего случая, операция “сплей” выполняется для родителя крайнего левого листа правого поддерева удаляемого узла.

Класс DataManager представляет собой менеджер данных, который использует иерархию коллекций на основе класса SplayTree. Разберем методы и функциональность этого класса:

1. private поле pools:

- Это экземпляр класса SplayTree с четырьмя уровнями вложенности. Он используется для хранения данных в иерархии "пулы" -> "схемы" -> "коллекции" -> "записи".

2. addDataPool:

- Добавляет новый "пул данных" с заданным именем. "Пул данных" - это верхний уровень иерархии данных.

3. removeDataPool:

- Удаляет "пул данных" с заданным именем и все связанные с ним схемы и коллекции.

4. addDataScheme:

- Добавляет новую "схему данных" в указанный "пул данных".

5. removeDataScheme:

- Удаляет "схему данных" из указанного "пула данных" и все связанные с ней коллекции.

6. addDataCollection:

- Добавляет новую "коллекцию данных" в указанную "схему данных".

7. removeDataCollection:

- Удаляет "коллекцию данных" из указанной "схемы данных".

8. addRecord:

- Добавляет запись в указанную "коллекцию данных".

9. removeRecord:

- Удаляет запись из указанной "коллекции данных" по applicantId.

10. getRecord:

- Возвращает указатель на запись в указанной "коллекции данных" по applicantId. Если запись не найдена, возвращается nullptr.

11. getRecordsInRange:

- Возвращает вектор записей из указанной "коллекции данных" в заданном диапазоне minApplicantId до maxApplicantId.

12. updateRecord:

- Обновляет запись в указанной "коллекции данных".

13. printPools:

- Выводит на экран список всех "пулов данных".

14. printSchemes:

- Выводит на экран список всех "схем данных" в указанном "пуле данных".

15. printCollections:

- Выводит на экран список всех "коллекций данных" в указанной "схеме данных".

16. clear:

- Очищает все данные, удаляя все "пулы данных" и связанные с ними "схемы данных", "коллекции данных" и записи.

17. saveState:

- Сохраняет текущее состояние данных в текстовый файл. Состояние сохраняется в формате, который можно будет восстановить с помощью метода loadState.

18. loadState:

- Загружает состояние данных из текстового файла и восстанавливает его. Состояние должно быть в формате, совместимом с методом saveState.

Этот класс предоставляет управление иерархией данных и позволяет добавлять, удалять, обновлять и получать записи внутри пулов, схем и коллекций данных, а также сохранять и загружать состояние данных, используя текстовый файл.

Функция processCommands выполняет обработку команд из текстового файла filename и взаимодействие с экземпляром класса DataManager на основе этих команд.

Функция processDialog создает текстовый интерфейс для взаимодействия с системой управления данными. Она предоставляет пользователю меню с различными опциями для выполнения операций над данными. Вот краткое описание каждой опции.

Руководство пользователя

После запуска приложения, пользователю доступен интерактивный режим, в котором он может выбирать команды для управления коллекцией данных. Были реализованы следующие команды:

- file: Позволяет загрузить и выполнить команды из текстового файла. При выполнении этой команды пользователю будет предложено ввести имя файла.

- ADD_POOL: Добавляет новый "пул данных" в систему. Пользователь должен ввести имя нового пула данных.

- REMOVE_POOL: Удаляет существующий "пул данных" из системы. Пользователь должен ввести имя пула данных, который необходимо удалить.

- ADD_SCHEME: Добавляет новую "схему данных" в указанный "пул данных". Пользователь должен ввести имя пула данных и имя новой схемы данных.

- REMOVE_SCHEME: Удаляет существующую "схему данных" из указанного "пула данных". Пользователь должен ввести имя пула данных и имя схемы данных.

- ADD_COLLECTION: Добавляет новую "коллекцию данных" в указанную "схему данных". Пользователь должен ввести имя пула данных, имя схемы данных и имя новой коллекции данных.

- REMOVE_COLLECTION: Удаляет существующую "коллекцию данных" из указанной "схемы данных". Пользователь должен ввести имя пула данных, имя схемы данных и имя коллекции данных.

- ADD_RECORD: Добавляет новую запись в указанную "коллекцию данных". Пользователь должен ввести информацию о записи, включая данные об соискателе, такие как имя, фамилия, дата рождения и другие параметры.

- REMOVE_RECORD: Удаляет запись из указанной "коллекции данных". Пользователь должен ввести идентификатор соискателя и информацию о местонахождении записи.

- GET_RECORD: Получает информацию о записи из указанной "коллекции данных" по идентификатору соискателя.

- GET_RECORDS_RANGE: Получает все записи из указанной "коллекции данных", у которых идентификатор соискателя находится в указанном диапазоне.

- UPDATE_RECORD: Обновляет информацию о существующей записи в указанной "коллекции данных". Пользователь должен ввести идентификатор соискателя и новые данные для обновления.

- PRINT_POOLS: Выводит список всех "пулов данных" в системе.

- PRINT_SCHEMES: Выводит список всех "схем данных" в указанном "пуле данных".

- PRINT_COLLECTIONS: Выводит список всех "коллекций данных" в указанной "схеме данных".

- CLEAR: Очищает все данные в системе, удаляя все "пулы данных", "схемы данных" и "коллекции данных".

- SAVE_STATE: Сохраняет текущее состояние данных в файл. Пользователь должен ввести имя файла, в который будет сохранено состояние.

- LOAD_STATE: Загружает состояние данных из файла. Пользователь должен ввести имя файла, из которого будет загружено состояние.

- Пустая строка: Если пользователь вводит пустую строку, программа продолжает выполнение без выполнения каких-либо действий.

- Неизвестная команда: Если введенная команда не распознается, программа выведет сообщение об ошибке.

Пользователь может завершить работу программы, введя команду "exit", после чего программа завершит свою работу. Рассмотрим работу интерактивного диалога, в котором пользователь отправляет запросы к данным. Отдельно рассмотрим работу команды file.

mmands.txt

```
ADD_POOL pool1
ADD_POOL pool2

ADD_SCHEME pool1 scheme1
ADD_SCHEME pool1 scheme2
ADD_SCHEME pool2 scheme3

ADD_COLLECTION pool1 scheme1 collection1
ADD_COLLECTION pool1 scheme1 collection2
ADD_COLLECTION pool1 scheme2 collection3
ADD_COLLECTION pool2 scheme3 collection4

ADD_RECORD pool1 scheme1 collection1 1 "John Alexander Doe" "1995-01-15" "resume1.pdf" 101 201 "C++" 5 3 false
ADD_RECORD pool1 scheme1 collection1 2 "Jane Elizabeth Smith" "1998-07-20" "resume2.pdf" 102 202 "Python" 7 7 false
ADD_RECORD pool1 scheme1 collection2 3 "Bob William Johnson" "1993-03-10" "resume3.pdf" 103 203 "Java" 10 9 false
ADD_RECORD pool1 scheme2 collection3 4 "Alice Mary Brown" "1996-11-03" "resume4.pdf" 104 204 "C#" 8 8 true
ADD_RECORD pool2 scheme3 collection4 5 "Michael James Williams" "1997-05-25" "resume5.pdf" 105 205 "JavaScript" 6 4 false

GET_RECORDS_RANGE pool1 scheme1 collection1 1 2

GET_RECORD pool1 scheme1 collection2 3
GET_RECORD pool1 scheme2 collection3 4
GET_RECORD pool2 scheme3 collection4 5

SAVE_STATE state.txt
```

Рисунок 1. Входной файл с потоком команд

```
dasoj@HVY-WXX9-69bd0fa9:~/Документы/МАИ/FundAlg/Course$ ./a.out
Welcome to the Data Management System!

Enter a command (or 'exit' to quit): file
Enter the filename: commands.txt

Records in range:
Applicant ID: 1
Last name: "John
First name: Alexander
Middle name: Doe"
Date of birth: "1995-01-15"
Resume link: "resume1.pdf"
HR manager ID: 101
Contest ID: 201
Programming language: "C++"
Total tasks: 5
Solved tasks: 3
Cheating detected: false

Applicant ID: 2
Last name: "Jane
First name: Elizabeth
Middle name: Smith"
Date of birth: "1998-07-20"
Resume link: "resume2.pdf"
HR manager ID: 102
Contest ID: 202
Programming language: "Python"
Total tasks: 7
Solved tasks: 7
Cheating detected: false


Applicant ID: 3
Last name: "Bob
First name: William
Middle name: Johnson"
Date of birth: "1993-03-10"
Resume link: "resume3.pdf"
HR manager ID: 103
Contest ID: 203
Programming language: "Java"
```

Рисунок 2. Полученный вывод

```
Applicant ID: 3
Last name: "Bob
First name: William
Middle name: Johnson"
Date of birth: "1993-03-10"
Resume link: "resume3.pdf"
HR manager ID: 103
Contest ID: 203
Programming language: "Java"
Total tasks: 10
Solved tasks: 9
Cheating detected: false


Applicant ID: 4
Last name: "Alice
First name: Mary
Middle name: Brown"
Date of birth: "1996-11-03"
Resume link: "resume4.pdf"
HR manager ID: 104
Contest ID: 204
Programming language: "C#"
Total tasks: 8
Solved tasks: 8
Cheating detected: false


Applicant ID: 5
Last name: "Michael
First name: James
Middle name: Williams"
Date of birth: "1997-05-25"
Resume link: "resume5.pdf"
HR manager ID: 105
Contest ID: 205
Programming language: "JavaScript"
Total tasks: 6
Solved tasks: 4
Cheating detected: false


Enter a command (or 'exit' to quit): 
```

Рисунок 3. Полученный вывод

Листинг 1. Работа интерактивного режима

```
dasoj@HVY-WXX9-69bd0fa9:~/Документы/МАИ/FundAlg/Course$ ./a.out
Welcome to the Data Management System!

Enter a command (or 'exit' to quit): LOAD_STATE
Enter the filename: state.txt

Enter a command (or 'exit' to quit): GET_RECORDS_RANGE
Enter the pool name: pool4
Enter the scheme name: scheme111
Enter the collection name: col
Enter the min applicant ID: 0
Enter the max applicant ID: 15
Pool with key pool4 not found

Records in range:

Enter a command (or 'exit' to quit): PRINT_POOLS
pool1 pool2

Enter a command (or 'exit' to quit): PRINT_SCHEMES
Enter the pool name: pool1
scheme1 scheme2

Enter a command (or 'exit' to quit): PRINT_COLLECTIONS
Enter the pool name: scheme1
Enter the scheme name: ...

Enter a command (or 'exit' to quit): PRINT_COLLECTIONS
Enter the pool name: pool1
Enter the scheme name: scheme1
collection1 collection2

Enter a command (or 'exit' to quit): GET_RECORDS_RANGE
Enter the pool name: pool1
Enter the scheme name: scheme1
Enter the collection name: collection1
Enter the min applicant ID: 1
Enter the max applicant ID: 3

Records in range:
Applicant ID: 1
Last name: "John
First name: Alexander
Middle name: Doe"
Date of birth: "1995-01-15"
Resume link: "resume1.pdf"
HR manager ID: 101
Contest ID: 201
Programming language: "C++"
Total tasks: 5
Solved tasks: 3
Cheating detected: false

Applicant ID: 2
Last name: "Jane
First name: Elizabeth
Middle name: Smith"
Date of birth: "1998-07-20"
Resume link: "resume2.pdf"
HR manager ID: 102
Contest ID: 202
Programming language: "Python"
```

```
Total tasks: 7
Solved tasks: 7
Cheating detected: false


Enter a command (or 'exit' to quit): UPDATE_RECORD
Enter the pool name: pool1
Enter the scheme name: scheme1
Enter the collection name: collection1
Enter the applicant ID: 2
Enter the last name: Jane
Enter the first name: Elizabeth
Enter the middle name: Smith
Enter the date of birth: 1998-07-20
Enter the resume link: new_resume.pdf
Enter the HR manager ID: 222
Enter the contest ID: 333
Enter the programming language: Scala
Enter the total tasks: 5
Enter the solved tasks: 3
Enter the cheating detected (true/false): 0

Enter a command (or 'exit' to quit): GET_RECORD
Enter the pool name: pool1
Enter the scheme name: scheme1
Enter the collection name: collection1
Enter the applicant ID: 2

Applicant ID: 2
Last name: Jane
First name: Elizabeth
Middle name: Smith
Date of birth: 1998-07-20
Resume link: new_resume.pdf
HR manager ID: 222
Contest ID: 333
Programming language: Scala
Total tasks: 5
Solved tasks: 3
Cheating detected: false


Enter a command (or 'exit' to quit): GET_RECORD
Enter the pool name: pool1
Enter the scheme name: scheme2
Enter the collection name: collection3
Enter the applicant ID: 4

Applicant ID: 4
Last name: "Alice
First name: Mary
Middle name: Brown"
Date of birth: "1996-11-03"
Resume link: "resume4.pdf"
HR manager ID: 104
Contest ID: 204
Programming language: "C#"
Total tasks: 8
Solved tasks: 8
Cheating detected: false


Enter a command (or 'exit' to quit): REMOVE_SCHEME
```

```
Enter the pool name: pool1
Enter the scheme name: scheme2

Enter a command (or 'exit' to quit): PRINT_SCHEMES
Enter the pool name: pool1
scheme1

Enter a command (or 'exit' to quit): REMOVE_RECORD
Enter the pool name: pool1
Enter the scheme name: scheme1
Enter the collection name:
collection1
Enter the applicant ID: 1

Enter a command (or 'exit' to quit): GET_RECORDS_RANGE
Enter the pool name: pool1
Enter the scheme name: scheme1
Enter the collection name: collection1
Enter the min applicant ID: 1
Enter the max applicant ID: 3

Records in range:
Applicant ID: 2
Last name: Jane
First name: Elizabeth
Middle name: Smith
Date of birth: 1998-07-20
Resume link: new_resume.pdf
HR manager ID: 222
Contest ID: 333
Programming language: Scala
Total tasks: 5
Solved tasks: 3
Cheating detected: false


Enter a command (or 'exit' to quit): CLEAR

Enter a command (or 'exit' to quit): PRINT_POOLS


Enter a command (or 'exit' to quit): ADD_POOL
Enter the pool name: new_pool

Enter a command (or 'exit' to quit): PRINT_POOLS
new_pool

Enter a command (or 'exit' to quit): PRINT
Unknown command: PRINT

Enter a command (or 'exit' to quit): PRINT_COLLECTIONS
Enter the pool name: pool3
Enter the scheme name: scheme1

Enter a command (or 'exit' to quit): exit
Exiting the program.
```

Здесь мы видим использование всех основных команд для взаимодействия с коллекцией данных.

Вывод

В рамках выполнения курсового проекта было разработано приложение, представляющее собой систему управления данными. Это приложение позволяет пользователям выполнять различные операции с данными, такие как добавление, удаление, обновление и получение информации из коллекций данных различных типов и контекстов их хранения.

Программа обеспечивает удобный интерфейс взаимодействия пользователя с системой. Пользователь может вводить команды в интерактивном режиме или загружать команды из текстовых файлов для массовой обработки данных.

Приложение было разработано с использованием эффективных механизмов управления памятью, что позволяет оптимизировать использование ресурсов компьютера и обеспечить стабильную работу приложения.

Эта программа является основным результатом курсового проектирования.

Приложение

https://github.com/Dasojj/FundAlg_MAI/tree/main/Course

Для получения доступа к проекту писать на почту: kirillm28@yandex.ru