

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: Жураев Д.С.

Группа: М8О-213Б-21

Преподаватель: Ирбитский И.С.

Оценка:

Дата:

Москва 2023

Содержание

Введение	4
Аллокаторы	6
Освобождение в рассортированном списке	6
Освобождение с дескрипторами границ	7
Система двойников	7
Деревья	8
Двоичное дерево поиска.....	8
Операции двоичного дерева поиска	8
Splay tree	9
Операции косого дерева.....	10
Splay.....	10
Merge	12
Поиск.....	13
Вставка	13
Удаление.....	13
АВЛ-дерево	13
Операции АВЛ-дерева.....	13
Balance.....	13
Поиск.....	14
Вставка	14
Удаление.....	14
Интерактивный диалог с пользователем	14
Механизм сохранения состояния системы хранения данных в файловую систему и восстановление состояния системы хранения данных из файловой системы.....	14
Хранение объектов строк с помощью паттерна Flyweight.....	15
Клиент-серверная реализация приложения	17
Кастомизация средств IPC	18
Shared memory.....	18
File mapping.....	19

Руководство пользователя.....	21
Демонстрация работы приложения.....	22
Вывод	28
Приложение	28
Список использованных источников.....	28

Введение

Приложение, реализованное в рамках курсового проекта на языке программирования C++, позволяет выполнять операции над коллекциями данных заданного типа и контекстами их хранения.

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.

Также были реализованы дополнительные задания.

Реализована возможность кастомизации (для заданного пула схем) аллокаторов для размещения объектов данных:

- освобождение в рассортированном списке;
- освобождение с дескрипторами границ;

- система двойников.

Ассоциативный контейнер в рамках реализации данного варианта представляет собой косое дерево.

Также реализована кастомизация реализаций ассоциативных контейнеров, репрезентирующих коллекции данных: косое дерево, AVL-дерево.

Также реализованы следующие дополнительные задания:

- интерактивный диалог с пользователем;
- хранение объектов строк, размещенных в объектах данных, на основе структурного паттерна проектирования «Приспособленец»;
- механизм сохранения состояния системы хранения данных в файловую систему и восстановление состояния системы хранения данных из файловой системы;
- функционал приложения в виде сервера, запросы на который поступают от клиентских приложений. При этом взаимодействие между сервером и клиентом происходит через средства межпроцессного взаимодействия (IPC);
- кастомизация средств IPC: Unix message queues, Unix shared memory + Unix semaphores, Unix file mapping.

Коллекция данных представляет собой данные о встречах в рабочем календаре, содержащие следующие поля:

- id встречи (данное поле формирует уникальный ключ объекта данных);
- вид встречи (ежедневная встреча/встреча по результатам отчётного периода/собеседование/корпоратив);
- формат проведения (очный/дистанционный);
- описание встречи;
- ссылка на встречу в дистанционном формате;
- ФИО создателя встречи (раздельные поля);
- дата встречи;
- время начала встречи;
- продолжительность встречи (в минутах);
- список приглашённых на встречу (в виде строки).

Аллокаторы

Аллокатор представляет собой объект, который управляет выделением памяти под данные, хранящиеся в коллекциях. Сам же аллокатор содержит блок памяти с сервисной информацией и блоки, которые выделяются по внешнему запросу.[1]

Реализованы 4 вида аллокаторов:

- на глобальной куче;
- с освобождением на рассортированном списке;
- с освобождением с дескрипторами границ;
- система двойников.

Также реализованы методы поиска подходящего для выделения блока:

- метод первого подходящего: выбирается первый по списку свободный участок подходящего размера;
- метод лучшего подходящего: выбирается наименьший по размеру участок из всех подходящих;
- метод худшего подходящего: выбирается наибольший по размеру участок из всех подходящих.

Освобождение в рассортированном списке

Идея данного аллокатора в том, что каждый свободный блок хранит указатель на следующий блок и свой размер, а в занятых блоках памяти хранится только размер этого блока без указателя на следующий занятый блок. Таким образом, получается рассортированный односвязный список из свободных блоков памяти.[2]

При выделении памяти с помощью этого аллокатора возникают два случая:

- Размер блока равен размеру, запрашиваемому пользователем. В таком случае указатель предыдущего блока переставляется с текущего на следующий;
- Размер блока больше размера, запрашиваемого пользователем. В таком случае свободная часть блока отделяется от занятой, размер блока уменьшается до размера выделяемой памяти и указатель с предыдущего блока переставляется с текущего на оставшийся свободный участок.

Выделенный блок помечается как занятый и результат возвращается пользователю. Также если данный блок был первым в списке, то указатель в сервисной части аллокатора на первый свободный блок переставляется на следующий за этим.

При освобождении памяти надо найти адреса блоков, между которыми стоит наш блок. Как только мы нашли блок, адрес которого больше адреса нашего участка, то значит мы нашли освобождаемый блок. Тут также может быть два случая:

- Освобождаемый участок стоит перед следующим свободным. В таком случае мы склеиваем эти два блока, то есть увеличиваем размер, прибавляя размер этого свободного блока, и переставляем указатель на следующий свободный участок.
- Между освобождаемым и найденным участками есть еще участки. В таком случае у освобождаемого блока указатель ставится на найденный

Сложность операций выделения и освобождения памяти: $O(n)$.

Освобождение с дескрипторами границ

Идея данного аллокатора заключается в образовании двусвязного списка из занятых участков памяти, благодаря чему освобождение происходит достаточно быстро.

Чтобы найти необходимый свободный блок при выделении, нужно вычислить разницу адресов занятых блоков. После того, как мы нашли блок, необходимо указатели у соседних блоков переставить на наш, а у нашего блока установить указатели на соседние занятые. После проверок можно вернуть данный блок пользователю.[3]

Освобождение в данном аллокаторе достаточно простое. Необходимо найти соседние занятые блоки для нашего и переставить их указатели друг на друга. На этом освобождение заканчивается.

Сложность операции выделения: $O(n)$.

Сложность операции освобождения: $O(1)$.

Система двойников

В данном аллокаторе размер кучи равен кратен целой степени двойки, то есть 2^N .

При поиске участка для выделения нужно найти блок минимального подходящего размера 2^k . Если блок найден, то используем его, в ином же случае находим ближайший блок большего размера и делим его до тех пор, пока не получим необходимый участок. Два блока, образовавшихся в результате деления называются двойниками.[2]

Также если известен адрес и размер участка, то известен и адрес его двойника. Этот адрес можно получить с помощью операции «исключающее или».

В каждом блоке имеется дополнительный бит занятости и два указателя, указывающие на следующий и предыдущий свободные блок.

При освобождении блока памяти также может возникнуть два случая:

- Двойник свободен. В этом случае объединяются двойники до тех пор, пока это возможно.
- Двойник занят. В этом случае ищем соседние от освобождаемого блоки, затем переставляем указатели у всех блоков.

Сложность операции выделения: $O(n)$.

Сложность операции освобождения: $O(\log n)$.

Деревья

Двоичное дерево поиска

Двоичное дерево - это иерархическая структура данных, в которой каждый узел имеет значение и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне называется корнем. Узлы, не имеющие потомков, называются листьями.

Бинарное дерево поиска отличает наличие некоторых свойств: значение левого потомка меньше значения родителя, а значение правого потомка больше значения потомка родителя. Данные в бинарном дереве поиска хранятся в отсортированном виде. При каждой операции вставки нового или удаления существующего узла отсортированный порядок дерева сохраняется. При поиске элемента сравнивается искомое значение с корнем. Если искомое больше корня, то поиск продолжается в правом потомке корня, если меньше, то в левом, если равно, то значение найдено и поиск прекращается.[4]

Операции двоичного дерева поиска

Поиск элемента.

Поиск элемента в двоичном дереве поиска достаточно прост, так как данные хранятся в отсортированном виде. Если искомое значение меньше значения узла, то идем в левое поддерево, в ином случае идем в правое поддерево, и так пока не найдем искомый узел.

Сложность операции поиска:

- В худшем случае: $O(n)$;
- В среднем: $O(\log n)$.

Вставка элемента.

Как и с поиском элемента, мы спускаемся по дереву, сравнивая значение вставляемого узла со значениями элементов дерева, и как только мы достигаем узла, у которого поддерево имеет значение NULL, вставляем туда наш узел.

Сложность операции вставки:

- В худшем случае: $O(n)$;
- В среднем: $O(\log n)$.

Удаление элемента.

При удалении элемента возникает три случая:

- Удаляемый элемент – лист. В этом случае просто удаляем лист из дерева.
- У удаляемого узла есть потомок. В этом случае меняем удаляемый узел на потомка, а затем удаляем наш узел.
- У удаляемого узла есть два потомка. Если в первом узле правого поддерева нет левого узла, то используем его, в другом случае находим самый левый элемент правого поддерева. После нахождения этого элемента заменяем удаляемый узел на найденный, затем удаляем наш узел.

Сложность операции удаления:

- В худшем случае: $O(n)$;
- В среднем: $O(\log n)$.

Splay tree

В косом дереве, как и в АВЛ и красно-черном деревьях, поддерживается свойство сбалансированности. В splay-дереве обеспечивается выполнение операций поиска, вставки и удаления за логарифмическое время. Идея косого

дерева в том, что вставка элемента, к которому недавно осуществлялся доступ, происходит в корень дерева, что позволяет получить этот элемент за $O(1)$. [5]

Операции косого дерева

Splay

Основная операция данного дерева, которая заключается в перемещении узла в корень с помощью выполнения следующих операций:

- Zig;
- Zig-Zig;
- Zig-Zag.

Обозначим вершину, которую надо поставить на место корня, за x . Родителя обозначим за p , а уже его родителя за g .

Операция Zig:

Также эта операция называется малый правый поворот.

Данная операция выполняется в случае, когда p является корнем. Выполняется поворот по ребру между вершинами x и p . Эта операция нужна только для разбора крайнего случая и выполняется один раз в конце, когда изначальная глубина x нечетна.

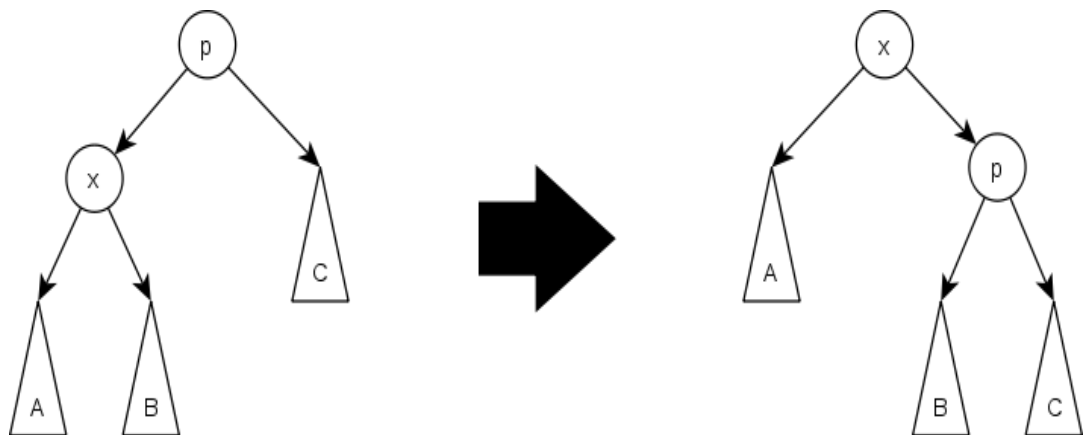


Рисунок 1. Zig (малый правый поворот)

Листинг 1. Операция Zig.

```
void right_rotation(node *&subtree_root) const
{
    node *tmp = subtree_root;
    subtree_root = subtree_root->left_subtree_address;
    tmp->left_subtree_address = subtree_root->right_subtree_address;
    subtree_root->right_subtree_address = tmp;
}
```

Операция Zig-Zig:

Данная операция выполняется в случае, когда обе вершины x и p являются левыми сыновьями. Выполняется поворот по ребру между вершинами p и g , затем по ребру между вершинами x и p .

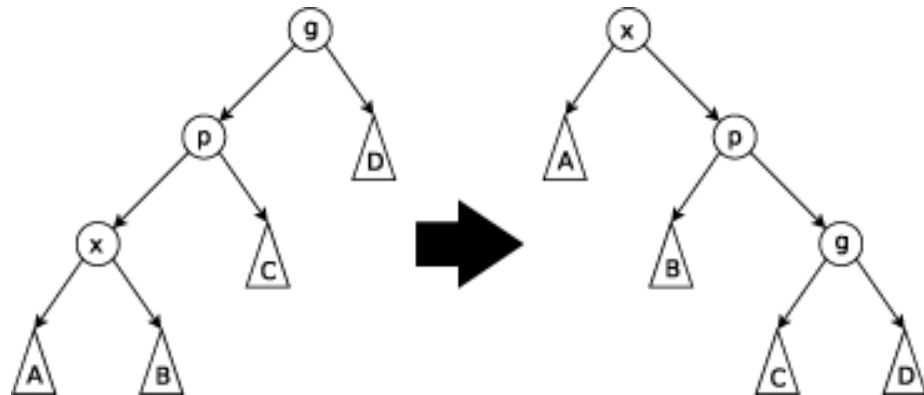


Рисунок 2. Операция Zig-Zig (большой правый поворот)

Аналогично, когда вершины x и p являются правыми сыновьями. В этом случае выполняется операция Zag-Zag, то есть большой левый поворот.

Операция Zig-Zag:

Данная операция выполняется в случае, когда вершина x является правым потомком, а вершина p – левым. Выполняется поворот по ребру между узлами x и p , затем по ребру между узлами x и g .

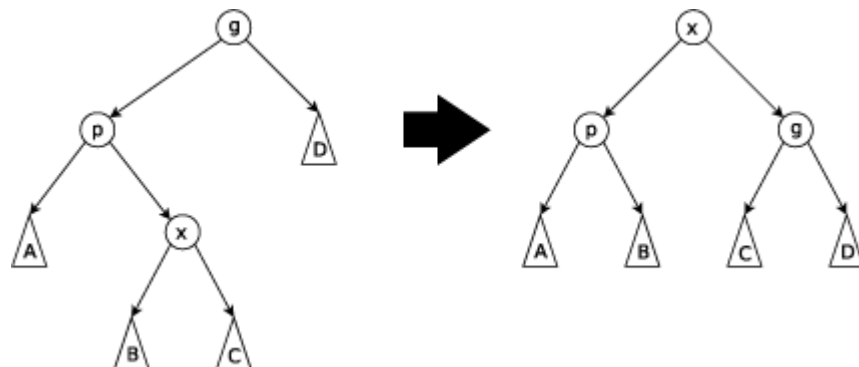


Рисунок 3. Операция Zig-Zag (малый правый поворот и малый левый поворот)

Аналогично, когда x – левый потомок, а p – правый. В этом случае выполняется операция Zag-Zig, то есть малый левый поворот и малый правый поворот.

Листинг 2. Операция Zag.

```
void left_rotation(node *&subtree_root) const
{
    node * tmp = subtree_root;
```

```

    subtree_root = subtree_root->right_subtree_address;
    tmp->right_subtree_address = subtree_root->left_subtree_address;
    subtree_root->left_subtree_address = tmp;
}

```

Merge

Данная операция вызывается, когда требуется удалить узел из дерева. Эта операция нужна для слияния двух поддеревьев удаляемого узла. Сначала находим максимальный элемент поддерева, значения узлов которого больше, чем у другого поддерева, то есть у левого. Затем вызываем операцию Splay для этого узла. После этого приклеиваем правого потомка удаляемого узла к получившемуся поддереву. Таким образом мы получаем целое дерево без нарушения сбалансированности.[5]

Листинг 3. Операция Merge.

```

node * splay_tree<tkey, tvalue, tkey_comparer>::merge(node *&left_subtree, node
    *&right_subtree)
{
    if (left_subtree == nullptr && right_subtree == nullptr)
    {
        return nullptr;
    }

    if (left_subtree == nullptr && right_subtree != nullptr)
    {
        return right_subtree;
    }
    else if (left_subtree != nullptr && right_subtree == nullptr)
    {
        return left_subtree;
    }
    else
    {
        std::stack<node **> path_to_subtree_root_exclusive;

        node * successor = left_subtree;
        node ** previous_node_address = &left_subtree;

        while (successor->right_subtree_address != nullptr)
        {
            path_to_subtree_root_exclusive.push(previous_node_address);
            previous_node_address = &(successor->right_subtree_address);
            successor = successor->right_subtree_address;
        }

        splay(successor, path_to_subtree_root_exclusive);

        left_subtree->right_subtree_address = right_subtree;

        return left_subtree;
    }
}

```

```
}  
}
```

Поиск

Поиск в косом дереве выполняется так же, как и в обычном бинарном дереве поиска, однако после завершения поиска выполняется операция Splay для найденного узла.

Вставка

Вставка в косом дереве похожа на вставку элемента в бинарное дерево поиска, как и операция поиска. Как и в поиске, после вставки элемента выполняется операция Splay для вставленного узла, вследствие чего узел становится корневым.

Удаление

При выполнении данной операции сначала находим узел, который необходимо удалить, затем вызываем операцию Splay для этого узла. После этого вызываем операцию Merge для потомков удаляемого узла.

АВЛ-дерево

АВЛ-дерево – это прежде всего двоичное дерево поиска, ключи которого удовлетворяют стандартному свойству: ключ любого узла дерева не меньше любого ключа в левом поддереве данного узла и не больше любого ключа в правом поддереве этого узла.

Особенностью АВЛ-дерева является то, что оно является сбалансированным в следующем смысле: для любого узла дерева высота его правого поддерева отличается от высоты левого поддерева не более чем на единицу. Также узел АВЛ-дерева хранит поле height – высоту поддерева с корнем в данном узле.[6]

Операции АВЛ-дерева

Balance

Данная операция выполняет балансировку дерева. Она может понадобиться, когда разность высот некоторых узлов оказывается равной 2 или -2, т.е. возникает разбалансировка поддерева. Такая ситуация возникает в процессе добавления и удаления узла.[6]

Балансировка происходит посредством поворотов, рассмотренных в разделе про Splay-дерево.

В реализации АВЛ-дерева доопределяется лишь один шаг — это операция, которая выполняется после удаления и вставки. Операция поиска не требует никаких модификаций.

Поиск

В реализации АВЛ-дерева операция поиска никак не модифицируется, то есть она аналогична операции поиска в обычном бинарном дереве поиска.

Вставка

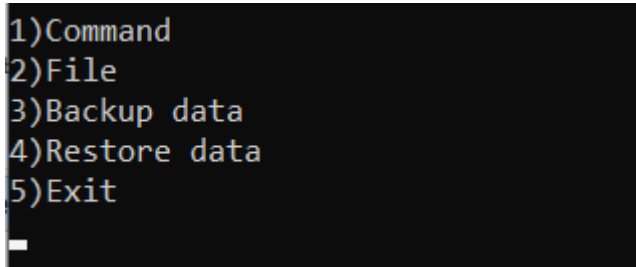
После вставки для каждого узла, который находится в стеке, содержащем путь к вставленному узлу, выполняется функция балансировки.

Удаление

В операции удалении, как и в операции вставки для АВЛ-дерева, вызывается операция балансировки для каждого узла, находящегося в стеке пути.

Интерактивный диалог с пользователем

Диалог дает возможность пользователю выбирать формат ввода команд, то есть можно вводить команды из консоли и можно передать путь к файлу, из которого считываются команды. Также есть возможность сохранить состояние базы данных в файл и восстановить базу данных из файла.



```
1)Command
2)File
3)Backup data
4)Restore data
5)Exit
_
```

Рисунок 4. Меню выбора операции над базой данных

Механизм сохранения состояния системы хранения данных в файловую систему и восстановление состояния системы хранения данных из файловой системы

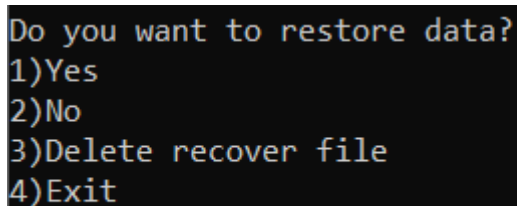
Данный механизм реализован в виде отдельного класса, в котором реализованы методы backup (сохранение) и restore (восстановление). Также в нем реализованы вспомогательные методы, которые вызываются в методах backup и restore. Данное задание реализовано по принципу журналирования.

При вызове метода backup нынешнее состояние базы данных сохраняется в файл. При этом файл можно создать, либо сохранить в существующий. Существует служебный файл broker.txt, в который записываются все команды, которые передавал пользователь. Когда пользователь выбирает backup, команды из служебного файла записываются в файл для сохранения.

При вызове метода restore база данных восстанавливается из файла. Файл выбирается из предоставляемого списка существующих файлов. При этом если ни одного файла не существует, то выводится информация о том, что файлов нет, и операция restore не выполняется. Данный метод возвращает вектор строк, в котором лежат команды, которые отправляются на сервер для выполнения.

Также существует служебный файл existing_recover_files.txt, в котором хранятся пути к файлам, в которых сохранены некоторые состояния базы данных, сохраненные пользователем.

Также при старте работы программы есть опция удаления файла из системы. При этом после удаления файла стартовое меню не пропадает, и пользователь может дальше удалять файлы или восстановить базу данных.



```
Do you want to restore data?  
1)Yes  
2)No  
3>Delete recover file  
4)Exit  
_
```

Рисунок 5. Стартовое меню программы

Хранение объектов строк с помощью паттерна Flyweight

Flyweight (Легковес, Приспособленец) – это структурный паттерн проектирования, который позволяет вместить большее количество объектов в отведённую оперативную память. Легковес экономит память, разделяя общее состояние объектов между собой, вместо хранения одинаковых данных в каждом объекте.

Приспособленец – это совместно используемый объект, который можно задействовать одновременно в нескольких контекстах. В каждом контексте он выглядит как независимый объект, то есть неотличим от экземпляра, который не используется совместно. Приспособленцы не могут делать предположений о контексте, в котором работают. Ключевая идея здесь – различие между

внутренним и внешним состояниями. Внутреннее состояние хранится в самом приспособленце и состоит из информации, не зависящей от его контекста. Именно поэтому он может использоваться совместно. Внешнее состояние зависит от контекста и изменяется вместе с ним, поэтому совместно не используется. Объекты-клиенты отвечают за передачу внешнего состояния приспособленцу, когда в этом возникает необходимость.[7]

Приспособленцами управляет класс `flyweight_factory`, содержащий пул легковесов.

Листинг 4. Класс `flyweight_string`

```
class flyweight_string
{
private:

    std::string _value;

public:

    const std::string &get_value() const;

    void set_value(const std::string &value);

};
```

Чтобы избежать хранения в памяти одинаковых строк, создается приспособленец и управляющий им объект. Значению поля строки присваивается указатель на легковес, который находится в пуле. Если такой строки нет, то создается новый легковес. Таким образом, избегается хранение одинаковых строк, что оптимизирует программу по памяти.

Листинг 5. Класс `flyweight_factory`

```
class flyweight_factory
{
private:

    std::unordered_map<std::string, std::shared_ptr<flyweight_string>>
    _flyweight_string;

public:

    static flyweight_factory &get_instance();

    std::shared_ptr<flyweight_string> get_flyweight_string(const std::string &value);

};
```


Клиент-серверная реализация приложения

Данное задание реализовано при помощи использования средств межпроцессного взаимодействия (IPC).

На сервере хранится база данных, которая заполняется по запросу клиента.

Клиент выбирает один из трех видов межпроцессного взаимодействия, с помощью которого в дальнейшем отправляются команды для выполнения на сервер.

При запуске сервера и клиента создается очередь сообщений Unix, которая связывает клиента с сервером. Через эту очередь передается сообщение клиента о выборе типа межпроцессного взаимодействия, после чего, в зависимости от выбора, клиент и сервер связываются либо очередью сообщений, либо разделяемой памятью, либо с помощью отображения файла в памяти. При этом все средства межпроцессного взаимодействия реализуются на платформе Unix.

Листинг 6. Создание очереди сообщений в клиенте для начальной связи

```
if((my_qid = msgget(IPC_PRIVATE, 0660)) == -1)
{
    perror("msgget: main, my_qid error");
    exit(1);
}
if((server_queue_key = ftok(SERVER_KEY_PATHNAME, PROJECT_ID)) == -1)
{
    perror("ftok: main, server_qid error");
    exit(1);
}
if((server_qid = msgget(server_queue_key, 0)) == -1)
{
    perror("msgget: main server_qid error");
    exit(1);
}
```

Листинг 7. Создание очереди сообщений на сервере для начальной связи

```
if((msg_queue_key = ftok(SERVER_KEY_PATHNAME, PROJECTD_ID)) == -1)
{
    perror("ftok error");
    exit(1);
}
if((qid = msgget(msg_queue_key, IPC_CREAT | 0660)) == -1)
{
    perror("msgget error");
    exit(1);
}
```

Функция `ftok` использует файл с именем `SERVER_KEY_PATHNAME`, которое должно указывать на существующий файл к которому есть доступ, и младшие 8 бит `PROJECT_ID`, который должен быть отличен от нуля, для создания ключа с типом `key_t`, используемого в System V IPC для работы с `msgget()`, `semget()`, и `shmget()`.

Функция `msgget(key_t key, int msgflag)` возвращает идентификатор очереди сообщений, связанный со значением параметра `key`. Она также создает новую очередь сообщений, если `key` равен `IPC_PRIVATE`; в случае если `key` не равен `IPC_PRIVATE`, то с параметром `key` не существует ни одной очереди сообщений и в поле `msgflag` включен флаг `IPC_CREAT`. [8]

Кастомизация средств IPC

Данное задание реализовано с помощью очереди сообщений, то есть сначала сервер и клиент связываются очередью сообщений Unix, затем клиент выбирает тип межпроцессного взаимодействия, и сообщение отправляется на сервер. На сервере уже, в зависимости от выбора клиента, создается связь с помощью того метода IPC, которое было выбрано клиентом.

Shared memory

Разделяемая память (Shared memory) является самым быстрым средством обмена данными между процессами.

В других средствах межпроцессового взаимодействия (IPC) обмен информацией между процессами проходит через ядро, что приводит к переключению контекста между процессом и ядром, т.е. к потерям производительности.

Листинг 7. Создание разделяемой памяти и семафора в клиенте

```
int shm_id = shmget(SHARED_MEMORY_KEY, SHARED_MEMORY_SIZE, IPC_CREAT | 0666);
if(shm_id == -1)
{
    perror("shmget error");
    exit(1);
}

auto *shared_data = (SharedData*) shmat(shm_id, nullptr, 0);
if(shared_data == (SharedData*)(-1))
{
    perror("shmat error");
    exit(1);
}

int sem_id = semget(SEMAPHORE_KEY, SEMAPHORE_COUNT, IPC_CREAT | 0666);
if(sem_id == -1)
```

```
{
    perror("semget error");
    exit(1);
}
```

Листинг 8. Создание разделяемой памяти и семафора на сервере

```
int shm_id = shmget(SHARED_MEMORY_KEY, SHARED_MEMORY_SIZE, 0666);
if(shm_id == -1)
{
    perror("shmget error");
    exit(1);
}

auto *shared_data = (SharedData*) shmat(shm_id, nullptr, 0);
if(shared_data == (SharedData*)(-1))
{
    perror("shmat error");
    exit(1);
}

int sem_id = semget(SEMAPHORE_KEY, SEMAPHORE_COUNT, 0666);
if(sem_id == -1)
{
    perror("semget error");
    exit(1);
}
```

Функция `shmget(key_t key, int size, int shmflag)` возвращает идентификатор разделяемому сегменту памяти, соответствующий значению аргумента `key`. Создается новый разделяемый сегмент памяти с размером `size` (округленным до размера, кратного `PAGE_SIZE`), если значение `key` равно `IPC_PRIVATE` или если значение `key` не равно `IPC_PRIVATE` и нет идентификатора, соответствующего `key`; причем, выражение `shmflag & IPC_CREAT` истинно.[9]

Функция `shmat` подстыковывает сегмент разделяемой памяти к адресному пространству вызывающего процесса. При ошибке функция возвращает `-1`, а переменной `errno` присваивается номер ошибки. При удачном выполнении `shmat` возвращает адрес подстыкованного сегмента памяти.

File mapping

Отображение файла в память (File mapping) – это способ работы с файлами в некоторых операционных системах, при котором всему файлу или некоторой непрерывной его части ставится в соответствие определённый участок памяти.

Файл, отображаемый в памяти, содержит содержимое файла в виртуальной памяти. Такое сопоставление файла и пространства памяти позволяет приложению, включая несколько процессов, изменять файл путем чтения и записи непосредственно в память.

Листинг 9. Создание файла, отображенного в памяти, и семафора в клиенте

```
int fd = open(FILE_MAPPING_PATHNAME, O_CREAT | O_RDWR);
if(fd == -1)
{
    perror("file error");
    exit(1);
}

char *addr = (char*) mmap(NULL, file_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
0);
if(addr == MAP_FAILED)
{
    perror("file mapping error");
    close(fd);
    exit(1);
}
close(fd);

int sem_id = semget(SEMAPHORE_KEY, SEMAPHORE_COUNT, IPC_CREAT | 0666);
if(sem_id == -1)
{
    perror("semget error");
    exit(1);
}
```

Листинг 10. Создание файла, отображенного в памяти, и семафора на сервере

```
int fd = open(FILE_MAPPING_PATHNAME, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
if(fd == -1)
{
    perror("file error");
    exit(1);
}

lseek(fd, file_size - 1, SEEK_SET);
write(fd, "", 1);

char *addr = (char*) mmap(NULL, file_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
0);
if(addr == MAP_FAILED)
{
    perror("file mapping error");
    close(fd);
    exit(1);
}
close(fd);
```

```
int sem_id = semget(SEMAPHORE_KEY, SEMAPHORE_COUNT, 0666);
if(sem_id == -1)
{
    perror("semget error");
    exit(1);
}
```

Чтобы отобразить файл в память, можно воспользоваться функцией `mmap(void *addr, size_t len, int prot, int flag, int filedes, off_t off)`. Она возвращает адрес начала участка отображаемой памяти или `MAP_FAILED` в случае неудачи. Первый аргумент – желаемый адрес начала участка отображенной памяти; `len` – количество байт, которое нужно отобразить в память; `prot` – число, определяющее степень защищённости отображенного участка памяти; `flag` – описывает атрибуты области; `fildes` – дескриптор файла, который нужно отобразить; `off` – смещение отображенного участка от начала файла.

В этом методе межпроцессного взаимодействия, как и в разделяемой памяти, используются семафоры.

Семафор – примитив синхронизации работы процессов и потоков, в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующей. Служит для построения более сложных механизмов синхронизации и используется для синхронизации параллельно работающих задач, для защиты передачи данных через разделяемую память, для защиты критических секций, а также для управления доступом к аппаратному обеспечению.

Системный вызов `semget(key_t key, int nsems, int semflg)` возвращает идентификатор набора семафоров, связанный с аргументом `key`. Создается новый набор из семафоров `nsems`, если значение `key` равно `IPC_PRIVATE` или с ключом `key` не связано ни одного существующего набора семафора, а выражение `semflg & IPC_CREAT` истинно.

При удачном завершении возвращаемое значение будет представлять собой идентификатор набора семафоров (целое неотрицательное значение), иначе возвращается -1, а переменной `errno` присваивается номер ошибки.

Руководство пользователя

При запуске приложения в консоли выводится меню для работы с ним.

Для работы с базой данных представлены 11 команд. Формат команд следующий

1. ADD_POOL <pool_name> <allocator_type> <memory_size>
<allocate_mod> <tree_type>
2. ADD_SCHEME <pool_name> <scheme_name> <tree_type>
3. ADD_COLLECTION <pool_name> <scheme_name>
<collection_name> <tree_type>
4. ADD_DATA <pool_name> <scheme_name> <collection_name>
<id_meeting> <meeting_type> <meeting_format> <description>
<link> <creator_surname> <creator_name> <creator_patronymic>
<date> <start_time> <minimal_duration> <invited_people>
5. REMOVE_POOL <pool_name>
6. REMOVE_SCHEME <pool_name> <scheme_name>
7. REMOVE_COLLECTION <pool_name> <scheme_name>
<collection_name>
8. REMOVE_DATA <pool_name> <scheme_name> <collection_name>
<id_meeting>
9. GET_DATA <pool_name> <scheme_name> <collection_name>
<id_meeting>
10. GET_RANGE <pool_name> <scheme_name> <collection_name>
<id_meeting_1> <id_meeting_2>
11. UPDATE_DATA <pool_name> <scheme_name> <collection_name>
<id_meeting> <meeting_type> <meeting_format> <description>
<link> <creator_surname> <creator_name> <creator_patronymic>
<date> <start_time> <minimal_duration> <invited_people>

Названия пулов, схем и коллекций могут быть любыми, кроме пустых строк, id встречи должно быть целым неотрицательным числом. Вид встречи имеет три разновидности: daily, period, interview, corporative. Формат встречи может online и offline. Далее идет описание встречи, ссылка на нее, если она проходит в формате online, ФИО создателя встречи, дата встречи в формате дд.мм.гггг или дд/мм/гггг, время ее начала в формате чч:мм, минимальная продолжительность в минутах и приглашенные люди, которые перечисляются через запятую. Также при добавлении пула в базу данных указывается вид аллокатора, размер и метод выделения (first_match, the_best_match, the_worst_match). Однако при выборе аллокатора global_heap указывать размер и метод выделения не нужно.

Демонстрация работы приложения

Создадим файл с командами.

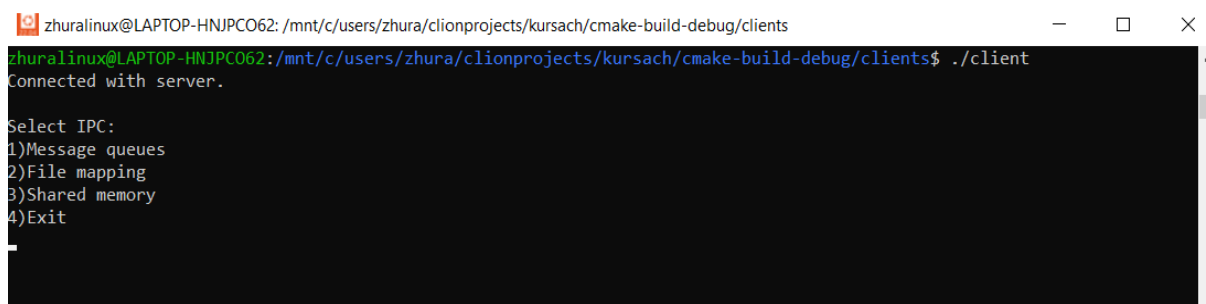
```

1 ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
2 ADD_SCHEME pool_1 scheme_1 splay_tree
3 ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
4 ADD_DATA pool_1 scheme_1 collection_1 1 daily offline bugdet_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
5 GET_DATA pool_1 scheme_1 collection_1 1
6 REMOVE_POOL pool_2
7 REMOVE_DATA pool_1 scheme_1 collection_1 1
8 REMOVE_COLLECTION pool_1 scheme_1 collection_1
9 REMOVE_SCHEME pool_1 scheme_1
10 REMOVE_POOL pool_1

```

Рисунок 6. Содержимое файла для чтения

Запустим сначала сервер, а затем клиента.



```

zhuralinux@LAPTOP-HNJPC062: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/clients
zhuralinux@LAPTOP-HNJPC062: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/clients$ ./client
Connected with server.

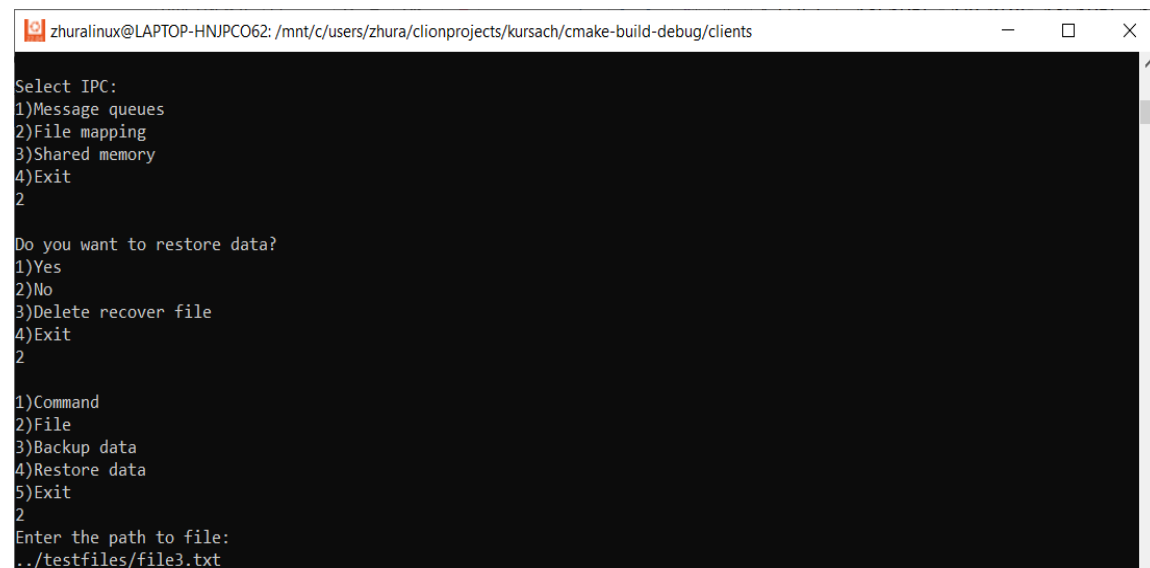
Select IPC:
1)Message queues
2)File mapping
3)Shared memory
4)Exit

```

Рисунок 7. Запуск работы клиента

Далее выбираем тип IPC и появляется стартовое меню, в котором можно восстановить сохраненное состояние базы данных.

Выбираем File mapping и передаем путь к файлу, команды из которого нужно считать.



```

zhuralinux@LAPTOP-HNJPC062: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/clients
Select IPC:
1)Message queues
2)File mapping
3)Shared memory
4)Exit
2

Do you want to restore data?
1)Yes
2)No
3)Delete recover file
4)Exit
2

1)Command
2)File
3)Backup data
4)Restore data
5)Exit
2
Enter the path to file:
../testfiles/file3.txt

```

Рисунок 8. Передача файла в клиенте

```

zhuralinux@LAPTOP-HNJPC062:/mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server$ ./server
Connected with client.
ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
[DATA BASE] pool with name:pool_1 added to data base

ADD_SCHEME pool_1 scheme_1 splay_tree
[DATA BASE] scheme with name:scheme_1 added to pool_1

ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
[DATA BASE] collection with name:collection_1 added to pool_1 scheme_1

ADD_DATA pool_1 scheme_1 collection_1 1 daily offline bugdet_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_1 scheme_1 collection_1

GET_DATA pool_1 scheme_1 collection_1 1
[DATA BASE] received value from pool_1 scheme_1 collection_1

      Data
meeting type: daily
meeting format: offline
description: bugdet_meeting
link: -
creator's surname: ivanov
creator's name: ivan
creator's patronymic: ivanovich
date: 16.09.2023
start time: 15:34
minimal duration: 60 min
invited people: ivan,maksim,danila,ilya

REMOVE_POOL pool_2
[DATA BASE] pool with name:pool_2 not found

REMOVE_DATA pool_1 scheme_1 collection_1 1
[DATA BASE] data removed from pool_1 scheme_1 collection_1

REMOVE_COLLECTION pool_1 scheme_1 collection_1
[DATA BASE] collection with name:collection_1 removed from pool_1 scheme_1

REMOVE_SCHEME pool_1 scheme_1
[DATA BASE] scheme with name:scheme_1 removed from pool_1

REMOVE_POOL pool_1
[DATA BASE] pool with name:pool_1 removed from data base

```

Рисунок 9. Вывод результата работы на сервере

Попробуем передать какие-то невалидные команды.

```

1 ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
2 ADD_SCHEME pool_1 scheme_1 avl_tree
3 ADD_SCHEME pool_1 scheme_2 splay
4 ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
5 ADD_DATA pool_1 scheme_1 collection_1 1 daily offline bugdet_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
6 GET_DATA pool_1 scheme_1 collection_1 1
7 REMOVE pool_1 scheme_1 collection_1

```

Рисунок 10. Содержимое файла с невалидными командами


```
zhuralinux@LAPTOP-HNJPCO62: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server

ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
[DATA BASE] pool with name:pool_1 added to data base

ADD_SCHEME pool_1 scheme_1 avl_tree
[DATA BASE] scheme with name:scheme_1 added to pool_1

ADD_SCHEME pool_1 scheme_2 splay
[DATA BASE] command can't be executed

ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
[DATA BASE] collection with name:collection_1 added to pool_1 scheme_1

ADD_DATA pool_1 scheme_1 collection_1 1 daily offline bugdet_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_1 scheme_1 collection_1

GET_DATA pool_1 scheme_1 collection_1 1
[DATA BASE] received value from pool_1 scheme_1 collection_1

Data
meeting type: daily
meeting format: offline
description: bugdet_meeting
link: -
creator's surname: ivanov
creator's name: ivan
creator's patronymic: ivanovich
date: 16.09.2023
start time: 15:34
minimal duration: 60 min
invited people: ivan,maksim,danila,ilya

REMOVE pool_1 scheme_1 collection_1
[DATA BASE] command can't be executed
```

Рисунок 11. Вывод результата работы невалидных команд

Как видно, программа выдала сообщение о том, что команда не может быть выполнена.

Теперь создадим несколько пулов с разными видами аллокаторов, добавим в них схемы, коллекции и данные.

```
1 ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
2 ADD_POOL pool_2 sorted_list 100000 the_best_match avl_tree
3 ADD_POOL pool_3 border_descriptors 70000 the_worst_match splay_tree
4 ADD_SCHEME pool_1 scheme_1 avl_tree
5 ADD_SCHEME pool_2 scheme_2 splay_tree
6 ADD_SCHEME pool_3 scheme_3 avl_tree
7 ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
8 ADD_COLLECTION pool_2 scheme_2 collection_2 splay_tree
9 ADD_COLLECTION pool_3 scheme_3 collection_3 splay_tree
10 ADD_DATA pool_1 scheme_1 collection_1 1 daily offline bugdet_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
11 ADD_DATA pool_2 scheme_2 collection_2 1 period online bugdet_meeting - khasanov ivan ivanovich 21.09.2023 16:30 60 ivan,maksim,danila,ilya
12 ADD_DATA pool_3 scheme_3 collection_3 1 period online lesson zoom.com/meeting1 zhuraev donish salohitdinovich 19.04.2023 12:00 240 daniil,mihail,rudolf,ilya,denis
13 GET_DATA pool_1 scheme_1 collection_1 1
14 GET_DATA pool_2 scheme_2 collection_2 1
15 GET_DATA pool_3 scheme_3 collection_3 1
```

Рисунок 12. Содержимое файла для выполнения

```

zhuralinux@LAPTOP-HNJPC062: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server

ADD_POOL pool_1 buddies_system 50000 first_match splay_tree
[DATA BASE] pool with name:pool_1 added to data base

ADD_POOL pool_2 sorted_list 100000 the_best_match avl_tree
[DATA BASE] pool with name:pool_2 added to data base

ADD_POOL pool_3 border_descriptors 70000 the_worst_match splay_tree
[DATA BASE] pool with name:pool_3 added to data base

ADD_SCHEME pool_1 scheme_1 avl_tree
[DATA BASE] scheme with name:scheme_1 added to pool_1

ADD_SCHEME pool_2 scheme_2 splay_tree
[DATA BASE] scheme with name:scheme_2 added to pool_2

ADD_SCHEME pool_3 scheme_3 avl_tree
[DATA BASE] scheme with name:scheme_3 added to pool_3

ADD_COLLECTION pool_1 scheme_1 collection_1 avl_tree
[DATA BASE] collection with name:collection_1 added to pool_1 scheme_1

ADD_COLLECTION pool_2 scheme_2 collection_2 splay_tree
[DATA BASE] collection with name:collection_2 added to pool_2 scheme_2

ADD_COLLECTION pool_3 scheme_3 collection_3 splay_tree
[DATA BASE] collection with name:collection_3 added to pool_3 scheme_3

ADD_DATA pool_1 scheme_1 collection_1 1 daily offline budget_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_1 scheme_1 collection_1

ADD_DATA pool_2 scheme_2 collection_2 1 period online budget_meeting - khasanov ivan ivanovich 21.09.2023 16:30 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_2 scheme_2 collection_2

ADD_DATA pool_3 scheme_3 collection_3 1 period online lesson zoom.com/meeting1 zhuraev donish salohitdinovich 19.04.2023 12:00 240 daniil,mihail,rudolf,ilya,denis
[DATA BASE] data added to pool_3 scheme_3 collection_3

```

Рисунок 13. Вывод результата работы программы

```

zhuralinux@LAPTOP-HNJPC062: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server

GET_DATA pool_1 scheme_1 collection_1 1
[DATA BASE] received value from pool_1 scheme_1 collection_1

Data
meeting type: daily
meeting format: offline
description: budget_meeting
link: -
creator's surname: ivanov
creator's name: ivan
creator's patronymic: ivanovich
date: 16.09.2023
start time: 15:34
minimal duration: 60 min
invited people: ivan,maksim,danila,ilya

GET_DATA pool_2 scheme_2 collection_2 1
[DATA BASE] received value from pool_2 scheme_2 collection_2

Data
meeting type: period
meeting format: online
description: budget_meeting
link: -
creator's surname: khasanov
creator's name: ivan
creator's patronymic: ivanovich
date: 21.09.2023
start time: 16:30
minimal duration: 60 min
invited people: ivan,maksim,danila,ilya

GET_DATA pool_3 scheme_3 collection_3 1
[DATA BASE] received value from pool_3 scheme_3 collection_3

Data
meeting type: period
meeting format: online
description: lesson
link: zoom.com/meeting1
creator's surname: zhuraev
creator's name: donish
creator's patronymic: salohitdinovich
date: 19.04.2023
start time: 12:00
minimal duration: 240 min
invited people: daniil,mihail,rudolf,ilya,denis

```

Рисунок 14. Вывод результата работы программы

Теперь проверим работу команды GET_RANGE.

```
1 ADD_POOL pool_1 global heap splay_tree
2 ADD_SCHEME pool_1 scheme_1 avl_tree
3 ADD_COLLECTION pool_1 scheme_1 collection_1 splay_tree
4 ADD_DATA pool_1 scheme_1 collection_1 1 daily offline budget_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
5 ADD_DATA pool_1 scheme_1 collection_1 2 period offline budget_meeting - khasanov ivan ivanovich 21.09.2023 16:30 60 ivan,maksim,danila,ilya
6 ADD_DATA pool_1 scheme_1 collection_1 3 period online lesson zoom.com/meeting1 zhuraev donish salohitdinovich 19.04.2023 12:00 240 daniil,mihail,rudolf,ilya,denis
7 GET_RANGE pool_1 scheme_1 collection_1 1 3
```

Рисунок 15. Содержимое файла для выполнения

```
zhuralinux@LAPTOP-HNJPCO62: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server

ADD_POOL pool_1 global_heap splay_tree
[DATA BASE] pool with name:pool_1 added to data base

ADD_SCHEME pool_1 scheme_1 avl_tree
[DATA BASE] scheme with name:scheme_1 added to pool_1

ADD_COLLECTION pool_1 scheme_1 collection_1 splay_tree
[DATA BASE] collection with name:collection_1 added to pool_1 scheme_1

ADD_DATA pool_1 scheme_1 collection_1 1 daily offline budget_meeting - ivanov ivan ivanovich 16.09.2023 15:34 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_1 scheme_1 collection_1

ADD_DATA pool_1 scheme_1 collection_1 2 period offline budget_meeting - khasanov ivan ivanovich 21.09.2023 16:30 60 ivan,maksim,danila,ilya
[DATA BASE] data added to pool_1 scheme_1 collection_1

ADD_DATA pool_1 scheme_1 collection_1 3 period online lesson zoom.com/meeting1 zhuraev donish salohitdinovich 19.04.2023 12:00 240 daniil,mihail,rudolf,ilya,denis
[DATA BASE] data added to pool_1 scheme_1 collection_1
```

Рисунок 16. Вывод результата работы программы

```
zhuralinux@LAPTOP-HNJPCO62: /mnt/c/users/zhura/clionprojects/kursach/cmake-build-debug/server

GET_RANGE pool_1 scheme_1 collection_1 1 3
[DATA BASE] received values from pool_1 scheme_1 collection_1

      Data:1
meeting type: daily
meeting format: offline
description: budget_meeting
link: -
creator's surname: ivanov
creator's name: ivan
creator's patronymic: ivanovich
date: 16.09.2023
start time: 15:34
minimal duartion: 60
invited people: ivan,maksim,danila,ilya

      Data:2
meeting type: period
meeting format: offline
description: budget_meeting
link: -
creator's surname: khasanov
creator's name: ivan
creator's patronymic: ivanovich
date: 21.09.2023
start time: 16:30
minimal duartion: 60
invited people: ivan,maksim,danila,ilya

      Data:3
meeting type: period
meeting format: online
description: lesson
link: zoom.com/meeting1
creator's surname: zhuraev
creator's name: donish
creator's patronymic: salohitdinovich
date: 19.04.2023
start time: 12:00
minimal duartion: 240
invited people: daniil,mihail,rudolf,ilya,denis
```

Рисунок 17. Результат работы команды GET_RANGE

Вывод

В данном курсовом проекте реализовано приложение, которое управляет хранилищем, имеющим несколько уровней хранения данных: пул, схема, коллекция. Программа способна выполнять команды, которые подаются пользователем через консоль или файл.

В курсовом проекте используются описанные аллокаторы и деревья. Также используются следующие паттерны проектирования: Flyweight, Singleton, Chain of responsibilities, Command.

Также реализация сохранения и восстановления состояния базы данных улучшила функционал приложения, так как перед выходом из приложения можно сохранять данные.

Помимо всего, реализован функционал клиент-серверного приложения с помощью средств IPC платформы Unix.

Реализованное приложение имеет широкий функционал, обеспечивающий гибкость и эффективность управления данными.

Приложение

Весь код реализованного приложения можно увидеть, перейдя по ссылке на Github: <https://github.com/Donish/kursach>

Список использованных источников

1. [Электронный ресурс] <https://habr.com/ru/articles/707032/> (22.10.2023)
2. Д. Кнут. Искусство программирования том 1.
3. [Электронный ресурс] <https://habr.com/ru/articles/473294/> (22.10.2023)
4. [Электронный ресурс] <https://codechick.io/tutorials/dsa/dsa-binary-search-tree> (22.10.2023)
5. [Электронный ресурс] <https://habr.com/ru/companies/JetBrains-education/articles/210296/> (22.10.2023)
6. [Электронный ресурс] <https://habr.com/ru/articles/150732/> (22.10.2023)
7. Э.Гамма, Р.Хелм, Р.Джонсон, Дж. Влиссидес. Паттерны объектно-ориентированного программирования.
8. [Электронный ресурс] <https://www.opennet.ru/man.shtml?topic=msgget&category=2&russian=0> (22.10.2023)

9. [Электронный ресурс]
<https://www.opennet.ru/man.shtml?topic=shmget&category=2&russian=0>
(22.10.2023)