# 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: | Морозов Даниил Павлович | (\_\_\_\_\_\_\_\_\_) |
| группа: | М8О-213Б-21 |  |
| преподаватель: | Ирбитский Илья Сергеевич | (\_\_\_\_\_\_\_\_\_) |
| Оценка: | \_\_\_\_\_\_\_\_\_ |  |
| дата: | \_\_/\_\_/\_\_\_\_ |  |

# 

# Введение

* **Задание курсовой работы:**

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 ее 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 этой курсовой работе:**

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

2. Реализуйте механизм, позволяющий выполнять запросы к данным в рамках коллекции данных на заданный момент времени (дата и время, для которых нужно вернуть актуальную версию данных, передаётся как параметр). Для реализации используйте поведенческие паттерны проектирования “Команда” и “Цепочка обязанностей”.

4. Обеспечьте хранение объектов строк, размещенных в объектах данных, на основе структурного паттерна проектирования “Приспособленец”. Дублирования объектов строк для разных объектов (независимо от контекста хранения) при этом запрещены. Доступ к строковому пулу обеспечьте на основе порождающего паттерна проектирования “Одиночка”.

6. Реализуйте возможность кастомизации (при создании) реализаций ассоциативных контейнеров, репрезентирующих коллекции данных: АВЛ-дерево, красно-чёрное дерево, косое дерево.

7. Реализуйте возможность кастомизации (для заданного пула схем) аллокаторов для размещения объектов данных: первый + лучший + худший подходящий + освобождение в рассортированном списке, первый + лучший + худший подходящий + освобождение с дескрипторами границ, система двойников.

* **Вариант курсовой работы 44:**

Тип данных: 6, контейнер: косое дерево, IPC: Windows shared memory + Windows semaphores

# Описание реализованного приложения

Реализованное в ходе курсовой работы приложение представляет собой многокомпонентную программу, предназначенную для формирования, хранения, добавления, удаления и просмотра данных о прохождении контеста соискателем. Программа применяется для структурирования и упорядочивания данных, а также быстрого поиска необходимых записей.

Поток команд, выполняемых в начале приложения, подаётся из файла, путь к которому подаётся в качестве аргумента командной строки. После выполнения изложенных в файле команд приложение предлагает возможность интерактивного диалога с пользователем. При удовлетворительном ответе в стандартный поток вывода печатается справка о программе и выполняемом функционале приложения, предлагается ввод пользователем команд. Пользователь вводит конкретные команды, которые в ходе выполнения программой синтаксического разбора введенных лексем проверяются на корректность, и, в случае успеха, выполняются с обращением к объекту класса . Класс реализует основную задачу курсовой работы – выполнение операций над коллекциями данных заданных типов и контекстами их хранения (коллекциями данных).

# Описание внешних библиотек

|  |  |  |
| --- | --- | --- |
| Таблица 1. Описание внешних библиотек | | |
| Внешняя библиотека | Описание | В программе |
| <iostream> | Часть стандартной библиотеки C++. Представляет собой объектно-ориентированную иерархию классов, где используется и множественное, и виртуальное наследование. В библиотеке реализована поддержка для файлового ввода/вывода данных встроенных типов. | Чтение пользовательского ввода/вывода, ведение интерактивного диалога с пользователем, сообщение программой об ошибках/неточностях пользовательских запросов, сообщение о возникших исключительных ситуациях. |
| <tuple> | Создает последовательность элементов фиксированной длины. Шаблон класса описывает объект, в котором хранятся объектов типов ,..., соответственно. . | Возвращение функциями, отвечающими за синтаксический разбор пользовательского ввода, отдельных лексем, предназначенных для работы отдельных методов и функций. |
| <vector> | Эмулирует работу стандартного массива C, а также некоторые дополнительные возможности, вроде автоматического изменения размера вектора при вставке или удалении элементов.  Все элементы вектора должны принадлежать одному типу. | Является типом возвращаемого значения метода класса , который находит в заданной коллекции записи с ключом , . |
| <fstream> | Класс потока ввода/вывода для работы с файлами.  Объекты этого класса поддерживают объект filebuf в качестве своего внутреннего буфера потока, который выполняет операции ввода/вывода в файле, с которым они связаны. | Чтение команд из файла, подаваемого как аргумент командной строки программы. |
| <stack> | Контейнерный адаптер, который дает программисту функциональность стека, в частности, структуру данных LIFO (последним пришел — первым вышел). Шаблон класса выступает в качестве оболочки базового контейнера — предоставляется только определенный набор функций. | Используется при поиске, вставке и удалении элементов в деревьях. |
| <cmath> | Объявляет набор функций для вычисления общих математических операций и преобразований. | Вычисление размера блоков памяти для аллокатора типа системы двойников, представленного классом . |

# Спецификация программы

Ниже представлены заголовочные файлы программы и их краткое описание.

|  |  |
| --- | --- |
| Таблица 2. Спецификация программы | |
| main.cpp | Производит открытие и чтение файла, путь к которому передан как аргумент командной строки, создает объекты классов и , выполняет команды из файла, консольные команды. |
| data\_base.h | Содержит класс , наследник классов и . |
| db\_user\_communication.h | Содержит идентификаторы возможных команд, исключение , функции, предназначенные для синтаксического разбора пользовательского ввода, функции, предназначенные для обращения к объекту класса с целью добавления, обновления, чтения, удаления записи по ключу. |
| key.h | Содержит классы и . |
| db\_value.h | Содержит идентификаторы полей значения записи, класс . |
| db\_value\_builder.h | Содержит класс . |
| string\_holder.h | Содержит класс . |
| command.h | Содержит базовый виртуальный класс и его классы-наследники: , , . |
| handler.h | Содержит базовый виртуальный класс и его классы-наследники: , , . |
| logger.h | Содержит базовый виртуальный класс и его класс-наследник . |
| logger\_builder.h | Содержит базовый виртуальный класс и его класс-наследник . |
| logger\_holder.h | Содержит класс . |
| memory\_base\_class.h | Содержит класс , наследник . |
| memory\_holder.h | Содержит класс . |
| memory\_from\_global\_heap.h | Содержит класс , наследник класса . |
| memory\_with\_sorted\_list\_deallocation.h | Содержит класс , наследника класса . |
| memory\_with\_boundary\_tags.h | Содержит класс , наследника класса . |
| memory\_with\_buddy\_system.h | Содержит класс , наследника класса . |
| associative\_container.h | Содержит базовый виртуальный класс . |
| bs\_tree.h | Содержит класс , наследника классов , , . |
| avl\_tree.h | Содержит класс , наследника класса . |
| splay\_tree.h | Содержит класс , наследника класса  . |
| rb\_tree.h | Содержит класс , наследника класса  . |

# Описание классов и структур

Класс . Класс-наследник классов и . Содержит следующие приватные поля: объект класса , объект класса , словарь () аллокаторов, сформированных в процессе работы программы при создании кастомных объектов деревьев структуры, – структура, представляющая собой дерево пулов, каждый из которых – дерево схем, каждая из которых – дерево коллекций. В коллекциях хранятся пары объекта класса и .

Также класс содержит классы исключений , , и идентификаторы аллокаторов и деревьев, доступных для кастомизации при создании пула, схемы или коллекции. Ниже приведены методы, реализованные в классе .

**Вспомогательные методы:**

|  |
| --- |
| Листинг 1. Вспомогательные методы класса |
| [[nodiscard]] associative\_container<std::string, associative\_container<std::string, associative\_container<key, db\_value \*> \*> \*> \*  find\_data\_pool  (std::string const & pool\_name) const; |
| [[nodiscard]] associative\_container<std::string, associative\_container<key, db\_value \*> \*> \*  find\_data\_scheme  (std::string const & pool\_name, std::string const & scheme\_name) const; |
| [[nodiscard]] associative\_container<key, db\_value \*> \*  find\_data\_collection  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name) const; |

: находит в структуре пул по ключу . Генерирует исключение в случае неудачи.

: находит пул по ключу , вызывая метод , находит в этом пуле схему по ключу .

: находит схему по ключу , вызывая метод , в этой схеме находит коллекцию данных по ключу .

**Методы, относящиеся к работе непосредственно со значениями:**

|  |
| --- |
| Листинг 2. Методы, относящиеся к работе непосредственно со значениями класса |
| void add\_to\_collection  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  const key& \_key, db\_value \* value) const; |
| void update\_in\_collection  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  const key& \_key, std::map<db\_value\_fields, unsigned char \*> upd\_dict) const; |
| [[nodiscard]] db\_value \* find\_among\_collection  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  const key& \_key) const; |
| [[nodiscard]] db\_value \* find\_with\_time  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  const key& \_key, uint64\_t time\_parameter) const; |
| [[nodiscard]] std::vector<db\_value \*> find\_in\_range  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  key min\_key, key max\_key) const; |
| void delete\_from\_collection  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  const key& key) const; |

: находит коллекцию по ключу , вызывая метод . Находит значение типа по ключу из этой коллекции, вызывая метод . Если этого ключа нет в коллекции, добавляет новую запись по ключу , иначе проверяет цепочку . Если последняя операция с этим значением – удаление, добавляет в новый – объект класса . Если последняя операция – не удаление, генерирует исключение.

: находит коллекцию по ключу , вызывая метод . Находит значение типа по ключу из этой коллекции, вызывая метод . Если ключ не найден, генерирует исключение. Иначе проверяет цепочку . Если последняя операция с этим значением – не удаление, добавляет в новый – объект класса , иначе генерирует исключение.

: находит коллекцию по ключу , вызывая метод . Находит значение типа по ключу из этой коллекции, вызывая метод . Если ключ не найден, генерирует исключение. Возвращает найденное методом значение .

: находит коллекцию по ключу , вызывая метод . Находит значение типа по ключу из этой коллекции, вызывая метод . Если ключ не найден, генерирует исключение. Создает копию найденного значения , вызывает метод первого в найденного значения. Возвращает .

: находит коллекцию по ключу , вызывая метод . Создает . С помощью инфиксного обхода дерева коллекции, находит значения, ключ которых удовлетворяет условиям . Возвращает .

: находит коллекцию по ключу , вызывая метод . Находит значение типа по ключу из этой коллекции, вызывая метод . Если ключ не найден, генерирует исключение. Иначе проверяет цепочку . Если последняя операция с этим значением – не удаление, добавляет в новый – объект класса , иначе генерирует исключение.

**Методы, относящиеся к структуре :**

|  |
| --- |
| Листинг 3. Методы, относящиеся к структуре класса |
| memory \*  get\_new\_allocator\_for\_inner\_trees  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  allocator\_types\_ allocator\_type, size\_t allocator\_pool\_size); |
| void add\_to\_structure  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name,  trees\_types\_ tree\_type, allocator\_types\_ allocator\_type, size\_t allocator\_pool\_size); |
| void delete\_collection  (const std::string & full\_path, const std::string & collection\_name, associative\_container<std::string,  associative\_container<key, db\_value \*> \*> \* parent\_scheme); |
| void delete\_scheme  (const std::string & full\_path, const std::string & scheme\_name, associative\_container<std::string,  associative\_container<std::string, associative\_container<key, db\_value \*> \*> \*> \* parent\_pool); |
| void delete\_pool(const std::string & pool\_name); |
| void delete\_from\_structure  (std::string const & pool\_name, std::string const & scheme\_name, std::string const & collection\_name); |

: создает объект класса . В соответствии с параметром и создает аллокатор требуемого типа с размером доступной памяти . Добавляет сгенерированный аллоктор в словарь объекта класса . Возвращает сгенерированный аллокатор.

: получает объект класса . При создании нового пула, схемы, коллекции создает соответствующее пулу, схеме, коллекции дерево в соответствии с параметром . Вставляет новый пул, схему или коллекцию соответственно в структуру , родительский для добавляемой схемы пул или родительскую для добавляемой коллекции схему. Если пул, схема, коллекция с ключом , или уже есть в соответствующих родительских структурах данных, генерирует исключение.

: удаляет коллекцию из родительской схемы.

: удаляет схему из родительского пула, удаляет схемы.

: удаляет пул из структуры , удаляет пул.

: находит пул, схему или коллекцию, которую нужно удалить. Если записи с ключом , или нет в в соответствующих родительских структурах данных, генерирует исключение. Иначе вызывает метод , или для этого пула, схемы или коллекции.

**Правило пяти:**

|  |
| --- |
| Листинг 4. Правило пяти класса |
| explicit data\_base(logger \* this\_db\_logger = nullptr, memory \* this\_db\_allocator = nullptr) |
| ~data\_base() override; |
| data\_base(data\_base const &obj) = delete; |
| data\_base(data\_base &&obj) noexcept = delete; |
| data\_base &operator=(data\_base const &obj) = delete; |
| data\_base &operator=(data\_base &&obj) noexcept = delete; |

конструктор: с помощью списков инициализации инициализирует поля и значениями и . Создает структуру , определенную номером варианта (splay tree).

деструктор: если структура инициализирована, с помощью постфиксного обхода дерева структуры удаляет все пулы структуры, вызывая метод . Удаляет объект структуры.

**Контракт и :**

|  |
| --- |
| Листинг 5. Контракт и класса |
| [[nodiscard]] logger \*get\_logger() const noexcept override |
| [[nodiscard]] memory \*get\_memory() const noexcept override |

: возвращает поле .

: возвращает поле .

Класс . Содержит приватное поле *\_meeting\_id* и класс . В классе реализованы: перегрузка оператора << и правило пяти.

|  |
| --- |
| Листинг 6. Методы класса |
| friend std::ostream &operator<<(std::ostream &out, const key &\_key\_); |
| key(std::ifstream\* input\_stream, bool is\_cin); |
| key() = default; |
| key(int *meeting\_id*); |
| key(key const &obj); |
| key(key &&obj) noexcept; |
| key &operator=(key const &obj); |
| key &operator=(key &&obj) noexcept; |

Класс . Дружественный класс класса . Содержит перегруженный оператор () – функтор. Служит для сравнения ключей (объектов класса ).

|  |
| --- |
| Листинг 7. Метод класса |
| int operator()(key x, key y); |

Класс . Содержит приватные поля , , , , , , , , , репрезентирующие поля данных, определенные вариантом, – дата в миллисекундах, когда был создан этот объект, – объект класса , репрезентирующий цепочку обязанностей, – объект класса , последний объект в цепочке обязанностей этого значения. Объекты строк, размещенные в хранятся в объекте класса , не допуская дублирования. В классе реализованы:

: создает и возвращает копию объекта .

: вызывает метод класса , возвращает указатель на уникальный объект строки.

: вызывает метод класса .

конструктор: инициализирует не строковые поля переданными параметрами. Строковые поля запрашивает из объекта класса с помощью метода . Вычисляет время создания этого объекта класса .

деструктор: Удаляет строковые поля из объекта класса с помощью метода . Вызывает удаление цепочки обязанностей этого объекта.

|  |
| --- |
| Листинг 8. Методы класса |
| friend std::ostream &operator<<(std::ostream &out, const db\_value &value); |
| db\_value \* make\_a\_copy(); |
| static std::string \* get\_ptr\_from\_string\_holder(std::string const & s); |
| static void remove\_string\_from\_string\_holder(std::string const & s) |
| db\_value(std::string const & surname, std::string const & name, std::string const & patronymic,  std::string const & birthday, std::string const & link\_to\_resume, unsigned hr\_id,  std::string const & prog\_lang, unsigned task\_count, unsigned solved\_task\_count, bool copying); |
| ~db\_value(); |

Класс . Реализация паттерна “Строитель”. Дружественный класс класса . Реализует методы:

with\_meeting\_type: Инициализирует тип встречи, указывая ее характеристики.

with\_meeting\_format: Инициализирует формат встречи, определяя ее структуру или вид.

with\_meeting\_description: Устанавливает описание встречи, содержащее информацию о ее целях и содержании.

with\_link\_to\_meeting: Устанавливает ссылку на встречу, что может включать URL или другой способ доступа к ней.

with\_name: Инициализирует имя участника встречи.

with\_surname: Инициализирует фамилию участника встречи.

with\_patronymic: Инициализирует отчество участника встречи.

with\_meeting\_date: Устанавливает дату встречи.

with\_meeting\_time: Устанавливает время начала встречи.

with\_duration: Устанавливает продолжительность встречи.

with\_list\_of\_invitees: Создает список участников встречи, включая их имена, фамилии и другие данные.

: проверяет параметр s на пустоту.

: возвращает объект класса по собранным данным.

: принимает, проверяет на корректность пользовательский ввод значения полей объекта класса . Возвращает объект класса по собранным данным.

Класс . Реализация паттерна “Приспособленец” и “Одиночка”. Содержит приватные поля и , класс – компаратор строк. Реализует методы:

: возвращает указатель на статический объект класса .

: ищет в структуре значение по переданной строке. В случае удачи инкрементирует значение поля найденного значения, репрезентирующего количество ссылок на объект строки. Иначе добавляет новое значение строки в структуру .

: ищет в структуре значение по переданной строке. Если количество ссылок на объект этой строки – 1, удаляет запись из структуры .

|  |
| --- |
| Листинг 10. Методы класса |
| static string\_holder \*get\_instance(); |
| std::string \*get\_string(std::string const &key); |
| void remove\_string(std::string const &key); |

Класс . Виртуальный (базовый, описывающий контракт взаимодействия) класс. Реализация паттерна “Команда”. Перечисляет метод выполнения команды – .

|  |
| --- |
| Листинг 11. Методы класса |
| virtual db\_value \*\* execute(db\_value \*\*) = 0; |

Класс . Наследник класса . Реализует метод выполнения команды удаления – : вызывает деструктор объекта класса , возвращает указатель на указатель на этот объект.

Класс . Дружественный класс класса . Наследник класса . Содержит приватное поле – словарь <название поля><новое значение>. Реализует метод выполнения команды обновления значения – : удаляет старые значения полей объекта класса , заменяет их на новые, переданные пользователем.

Класс . Дружественный класс класса . Наследник класса Содержит приватное поле – словарь <название поля><значение>. Реализует метод выполнения команды добавления значения – : с помощью объекта класса создает новый объект класса на основе .

Класс . Реализация паттерна “Цепочка обязанностей”. Содержит тип перечислений , перечисляющий типы классов-наследников , класс-исключение . Также содержит приватные поля – следующий за этим обработчик в цепочке обязанностей, – команда, которую выполняет этот обработчик, - время в миллисекундах, когда был создан этот обработчик, – тип этого обработчика. Реализует методы:

: возвращает значение поля .

: инициализирует значения поля параметром .

: в соответствии с параметром и положением в цепочке обязанностей, вызывает метод объекта и метод объекта , либо возвращает текущее состояние параметра .

: покомпонентно удаляет обработчики цепочки обязанностей.

|  |
| --- |
| Листинг 12. Методы класса |
| [[nodiscard]] handler\_types get\_handler\_type() const; |
| handler \*set\_next(handler \*handler); |
| db\_value\* handle(db\_value \*\* request, uint64\_t time\_parameter); |
| void delete\_chain\_of\_responsibility(); |

Класс . Наследник класса . Реализует конструктор , который инициализирует приватные поля командой и как .

Класс . Наследник класса . Реализует конструктор , который инициализирует приватные поля командой и как .

Класс . Дружественный класс класса . Наследник класса . Реализует конструктор класса , который принимает на вход новое значение. Собирает из этого значения словарь – параметр для команды , которой инициализируется приватное поле объекта класса. принимает значение .

Класс . Интерфейсный класс. Содержит тип перечислений , класс-исключение . Перечисленные методы отображены в листинге 13:

|  |
| --- |
| Листинг 13. Методы класса |
| static std::string severity\_to\_string\_logger(severity s); |
| static logger::severity from\_string\_to\_severity\_parse(const std::string &s); |
| virtual logger const \*log(std::string const &target, severity level) const = 0; |

Класс . Наследник класса . Содержит приватные поля – словарь потоков этого логгера и – словарь потоков всех логгеров. Переопределяет метод : для каждого из потоков проверяется условие достаточно строгого , генерируется запись вида [timestamp][severity]{сообщение }.

Класс . Реализация паттерна “Строитель”. Базовый класс. Перечисленные методы отображены в листинге 14.

|  |
| --- |
| Листинг 14. Методы класса |
| virtual logger\_builder \*with\_stream(std::string const &, logger::severity) = 0; |
| [[nodiscard]] virtual logger \*build() const = 0; |
| virtual logger \*config\_from\_json(const std::string &) = 0; |

Класс . Дружественный класс класса . Содержит приватное поле представляющее собой словарь название файла – . Переопределяет методы класса :

: добавляет в словарь запись <путь к файлу><>.

: возвращает объект класса , созданного на основе .

: создает и возвращает объект класса , созданного после парсинга .json файла с помощью методов библиотеки .

Класс . Представляет собой интерфейс-оболочку для использования объекта класса . Реализует методы логирования с заданной : , , , , , , вызывающие метод с соответствующими , который получает объект класса и, при его наличии, вызывает его метод . Перечисляет виртуальный метод .

Класс . Наследник класса . Базовый класс. Содержит тип перечислений , класс-исключение , приватное поле , репрезентирующее начало блока доверенной аллокатору памяти. Реализует методы:

– логгирует неслужебное содержимое освобождаемого участка перед освобождением в виде коллекции значений байт.

: возвращает указатель на размер доверенной аллокатору области памяти.

: возвращает указатель на указатель на объект класса этого аллокатора.

: возвращает указатель на родительский аллокатор этого аллокатора (объект класса ).

: возвращает указатель на стратегию аллокации этого аллокатора (объект типа перечислений ).

: возвращает указатель на первый пул аллокатора, где размещаются данные пользователя.

: возвращает адрес указателя в виде строки в шестнадцатеричной записи.

: вызывает метод объекта .

: вызывает метод объекта .

Перечисленные методы класса отображены в листинге 15.

|  |
| --- |
| Листинг 15. Методы класса memory |
| [[nodiscard]] virtual size\_t get\_allocator\_service\_block\_size() const; |
| [[nodiscard]] virtual size\_t\* get\_ptr\_size\_of\_allocator\_pool() const; |
| [[nodiscard]] virtual void \*get\_first\_available\_block\_address() const; |
| [[nodiscard]] virtual void \*\* get\_first\_available\_block\_address\_address() const; |
| [[nodiscard]] virtual size\_t get\_available\_block\_service\_block\_size() const; |
| virtual size\_t get\_available\_block\_size(void \* memory\_block) const; |
| virtual void \* get\_next\_available\_block\_address(void \* memory\_block) const; |
| [[nodiscard]] virtual void \*\* get\_first\_occupied\_block\_address\_address() const; |
| [[nodiscard]] virtual size\_t get\_occupied\_block\_service\_block\_size() const; |
| virtual size\_t get\_occupied\_block\_size(void \* memory\_block) const; |
| virtual size\_t get\_size\_of\_occupied\_block\_pool(void \* const occupied\_block) const = 0; |
| virtual void \* get\_next\_occupied\_block\_address(void \* memory\_block) const; |
| virtual void \* get\_previous\_occupied\_block\_address(void \* memory\_block) const; |
| virtual void \*allocate(size\_t target\_size) const = 0; |
| virtual void deallocate(void const \* const target\_to\_dealloc) const = 0; |

Класс . Представляет собой интерфейс для использования объекта класса . Содержит методы для обращения к памяти и , которые получают объект класса , вызывая виртуальный метод , и, при наличии объекта, вызывают его метод , иначе возвращают указатель на выделенный из глобальной кучи участок памяти.

Класс . Наследник класса . Представляет собой прослойку между пользователем и операторами ::new и ::delete, таким образом, что взаимодействие с объектом класса является подобным взаимодействию с ::allocate и ::deallocate. Его указывает только на объект класса этого аллокатора. Реализует и переопределяет методы:

: переопределяет метод наследуемого класса , возвращает указатель на указатель на объект класса .

: возвращает указатель на блок памяти, выделенный из глобальной кучи с помощью оператора , сохраняет размер блока в в начале этого блока, как метаданные блока.

: вызывает метод класса , возвращает блок памяти в глобальную кучу с помощью метода .

: переопределяет метод наследуемого класса , возвращает указатель на размер этого занятого блока.

– контракт взаимодействия с наследуемым классом .

Класс . Наследник класса . Представляет собой аллокатор с освобождением блоков в рассортированном списке. Структура, на которую указывает состоит из размера доверенной аллокатору области памяти, указателя на объект класса этого аллокатора, указателя на объект класса — родительского для этого аллокатора аллокатора, – одного из значений , указателя на первый свободный блок в доверенной области. Свободные блоки этого аллокатора представляют собой метаданные (размер свободного блока, включая метаданные, указатель на следующий свободный блок) и незанятую область, занятые блоки представляют собой метаданные (размер занятого блока, включая метаданные) и занятую область. Все свободные блоки аллокатора связаны последовательно в виде односвязного списка. Класс реализует и переопределяет методы:

: возвращает размер метаданных аллокатора.

: указывает на начало области памяти аллокатора, доступной для пользовательских манипуляций памятью.

: возвращает указатель на первый свободный блок в доверенной аллоктору области памяти.

: возвращает указатель на указатель на первый свободный блок в доверенной аллоктору области памяти.

: возвращает размер метаданных свободного блока.

: возвращает указатель на размер свободного блока.

: возвращает указатель на следующий свободный блок этого свободного блока.

: возвращает размер метаданных занятого блока.

: возвращает размер занятого блока без учета метаданных.

: проходит по списку свободных блоков аллокатора. В соответствии со стратегией выделения (best, worst или first), находит подходящий участок памяти. Добавляет остаток свободного блока к аллоцируемуму блоку, если его размер слишком мал, чтобы вместить служебные данные свободного блока. Корректирует односвязный список свободных блоков.

: вызывает метод класса , возвращает деаллоцируемый блок в список свободных блоков этого аллокатора.

– контракт взаимодействия с наследуемым классом .

Класс . Наследник класса . Представляет собой аллокатор с освобождением блоков с дескрипторами границ. Структура, на которую указывает состоит из размера доверенной аллокатору области памяти, указателя на объект класса этого аллокатора, указателя на объект класса — родительского для этого аллокатора аллокатора, – одного из значений , указателя на первый занятый блок в доверенной области. Свободные блоки этого аллокатора не хранят никаких метаданных; занятые представляют собой метаданные (размер блока с учётом метаданных и указатели на предыдущий и следующий занятые блоки) и занятый участок памяти. Все занятые блоки аллокатора последовательно связаны в виде двусвязного списка. Класс реализует и переопределяет методы:

: возвращает размер метаданных аллокатора.

: указывает на начало области памяти аллокатора, доступной для пользовательских манипуляций памятью.

: возвращает указатель на указатель на первый занятый блок этого аллокатора.

: возвращает размер метаданных занятого блока.

: возвращает размер занятого блока.

: возвращает размер занятого блока без учета метаданных.

: возвращает указатель на следующий занятый блок.

: возвращает указатель на предыдущий для этого занятый блок.

: находит блок достаточного размера в соответствии со стратегией выделения (best, worst или first. Добавляет остаток свободного блока к аллоцируемуму блоку, если его размер слишком мал, чтобы вместить служебные данные занятого блока. Корректирует двусвязный список занятых блоков этого аллокатора.

: вызывает метод класса , удаляет деаллоцируемый блок из списка занятых блоков.

– контракт взаимодействия с наследуемым классом .

Класс . Наследник класса . Представляет собой аллокатор с выделением и освобождением памяти при помощи алгоритма систему двойников. Согласно этому алгоритму, все блоки имеют размер , таким образом найти “двойника” – блок с тем же размером легко, достаточно использовать оператор побитового исключения. Структура, на которую указывает состоит из степени доверенной области аллокатора, указателя на объект класса этого аллокатора, указателя на объект класса — родительского для этого аллокатора аллокатора, указателя на первый свободный блок. Свободный блок представляет собой метаданные (занят ли участок, степень этого блока, указатель на следующий свободный блок) и свободный участок памяти. Занятый блок представляет собой метаданные (занят ли участок, степень этого блока) и занятый участок памяти. Класс реализует и переопределяет методы:

: возвращает указатель на переменную типа , отображающую, занят ли блок.

: возвращает указатель на степень этого блока.

: возвращает указатель на следующий свободный блок в этом аллокаторе.

: возвращает указатель на блок-двойник этого блока.

: возвращает размер метаданных аллокатора.

: возвращает указатель на степень доверенной области памяти аллокатора.

: возвращает указатель на указатель на объект класса .

: возвращает указатель на указатель на объект класса , родительский аллокатор этого аллокатора.

: возвращает указатель на первый свободный блок этого аллокатора.

: возвращает указатель на указатель на начало области памяти, доступной для пользовательских манипуляций памятью.

: возвращает указатель на указатель на объект класса .

: возвращает размер метаданных занятого блока.

: возвращает размер занятого блока без учета метаданных.

: возвращает указатель на первый свободный блок этого аллокатора.

: возвращает размер метаданных свободного блока.

: возвращает указатель на следующий свободный блок этого свободного блока.

: возвращает .

: возвращает – степень блока.

: находит свободный блок, минимальный по размеру. Удаляет этот блок из списка свободных блоков. Делит найденный блок надвое, пока размер блока удовлетворяет искомому, каждый двойник делимого блока добавляется в список свободных блоков.

: вызывает метод класса , добавляет блок в список свободных блоков, если это возможно, объединяет новый свободный блок с его двойником.

– контракт взаимодействия с наследуемым классом .

Класс . Родовой интерфейс (кастомизируемые параметры родового интерфейса: тип ключа tkey, тип значения tvalue, тип компаратора на ключах tkey\_comparer), предоставляющий функционал для работы с ассоциативным контейнером:

* – вставка пары <ключ><значение> в ассоциативный контейнер.
* – получает значение из ассоциативного контейнера по ключу.
* – удаляет запись из ассоциативного контейнера по ключу.

|  |
| --- |
| Листинг 16. Методы класса |
| virtual void insert(tkey const &key, tvalue &&value) = 0; |
| virtual tvalue const &get(tkey const &key) = 0; |
| virtual tvalue remove(tkey const &key) = 0; |

Класс . Наследник класса , , . Шаблонный класс с кастомизируемыми параметрами: тип ключа tkey, тип значения tvalue, тип компаратора на ключах tkey\_comparer. Распределение вложенных в объект дерева данных организовано через аллокатор, подаваемый объекту через конструктор. Реализованы классы префиксного (), инфиксного () и постфиксного () итераторов для обхода (префиксного, инфиксного, постфиксного) дерева с возвратом из итератора ключа, значения, глубины (относительно корня; глубина корня дерева равна нулю) обходимого узла; для каждого узла дерева, в порядке, определяемом правилом обхода. Также реализованы защищенные методы малого левого и малого правого поворотов. Класс содержит структуру , представляющую собой узел двоичного дерева и содержащую поля , , указатели на левое и правое поддерево. В классе определены классы-исключения , , , . Операции CRD реализованы на основе поведенческого паттерна проектирования “шаблонный метод” с помощью определения классов , , , , предусмотрены хуки для выполнения в подклассах BST дополнительных операций до/после рекурсивного вызова относительно текущего узла дерева, реализовано правило пяти. Приватное поле класса – – указатель на корень BST дерева. Реализует контракт класса

* – вызывает метод класса .
* – вызывает метод класса .
* – вызывает метод класса .

Класс . Дружественный класс класса , наследник класса . Содержит приватное поле – указатель на дерево, для которого будут реализованы вставка, поиск и удаление. Включает в себя методы:

: возвращает указатель на корень дерева .

: возвращает пару <стек><узел>, где стек – путь от корня до этого узла.

: возвращает родителя этого узла — элемент на вершине стека.

: возвращает родителя родителя этого узла (деда) с помощью стека.

: малый поворот налево.

: малый поворот направо.

: контракт взаимодействия с наследуемым классом .

Класс , наследник и . Реализует методы:

: находит место для вставки узла с помощью метода класса , выделяет память под новый узел и инициализирует его параметрами. Вызывает виртуальный метод .

: виртуальный метод, возвращает размер структуры .

: виртуальный метод, инициализирует выделенный участок памяти как объект структуры .

: контракт взаимодействия с наследуемым классом .

|  |
| --- |
| Листинг 17. Методы класса |
| void insert(tkey const &key, tvalue &&value); |
| [[nodiscard]] virtual size\_t get\_node\_size() const; |
| virtual void initialize\_memory\_with\_node(node \*target\_ptr) const; |
| virtual void after\_insert\_inner(std::stack<node \*\*> &path, node \*\*target\_ptr); |
| [[nodiscard]] memory \*get\_memory() const noexcept override; |

Класс , наследник . Реализует метод:

: находит узел с помощью метода класса , вызывает виртуальный метод , возвращает значение узла.

|  |
| --- |
| Листинг 18. Методы класса |
| tvalue const &find(tkey const &key); |
| virtual void after\_find\_inner(std::stack<node \*\*> &path, node \*\*&target\_ptr); |

Класс , наследник и . Реализует методы:

: находит узел с помощью метода класса , удаляет элемент из дерева в соответствии с количеством дочерних узлов и корректирует указатели объектов. Вызывает метод , и, если удаляемый элемент не был последним в дереве, вызывает виртуальный метод . Возвращает значение удаленного узла.

: меняет указатели местами.

: меняет местами узлы с учетом их возможного близкого “родства”.

: вызывает деструктор объекта , деаллоцирует выделенный под него участок памяти.

: контракт взаимодействия с наследуемым классом .

|  |
| --- |
| Листинг 19. Методы класса |
| virtual tvalue remove(tkey const &key); |
| template<typename T>  void swap(T \*\*left, T \*\*right); |
| node\*\* swap\_nodes(node \*\*one\_node, node \*\*another\_node); |
| void cleanup\_node(node \*\*node\_address); |
| virtual void swap\_additional\_data(node \*one\_node, node \*another\_node); |
| virtual void after\_remove(std::stack<node \*\*> &path) const; |
| memory \*get\_memory() const noexcept override; |

Класс . Наследник класса . Представляет собой AVL-дерево, содержит структуру – наследник структуры с приватным полем – высотой узла. Баланс узла – разность значений его левого и правого поддеревьев. Класс включает в себя классы , наследника , , наследника , , наследника , реализует правило пяти.

Класс . Реализует метод : считает баланс текущего узла, балансы левых и правых поддеревьев этого узла. При балансе узлов : , производит операции поворота в зависимости от баланса узлов:

* если баланс текущего узла
  + баланс левого поддерева , производит малый правый поворот
  + баланс левого поддерева , производит большой правый поворот.
* если баланс текущего узла
  + баланс правого поддерева , производит малый левый поворот
  + баланс правого поддерева , производит большой левый поворот.

Корректирует значения узлов и продолжает балансировку до корня дерева.

Класс . Реализует и переопределяет методы:

: возвращает размер .

: инициализирует выделенный участок памяти как объект структуры .

: обновляет значения родительского и прародительского (деда) узлов. Если есть дед, вызывает метод от узла деда.

Класс . Переопределяет методы:

: обновляет значение родительского узла. Вызывает метод от родительского узла.

: обменивает значения двух узлов.

Класс . Наследник класса . Представляет собой splay-дерево (косое). Класс включает в себя классы , наследника , , наследника , наследника класса , , наследника , реализует правило пяти.

Класс . Реализует метод : находит родителя текущего узла. Если родитель – корень дерева, производит операцию : выполняет малый левый или малый правый поворот в зависимости от положения текущего узла относительно родительского. Иначе находит деда текущего узла. Выполняет операции поворота в зависимости от положения узлов:

* если и родитель, и текущий узел – левые поддеревья, производит операцию : малый правый поворот относительно родителя и деда и малый правый поворот относительно текущего и родительского узлов. Случай зеркален, если и родитель, и текущий узел – правые поддеревья.
* если родитель – левое поддерево, текущий узел – правое, производит операцию : большой левый поворот. Случай зеркален, если родитель – правое поддерево, текущий узел – левое.

Операция продолжается, пока текущий узел не станет корневым узлом дерева.

Класс . Переопределяет метод : вызывает метод от добавленного узла.

Класс . Переопределяет метод : вызывает метод от найденного узла.

Класс . Переопределяет метод : вызывает метод от родителя удаленного узла.

Класс . Наследник класса . Представляет собой красно-черное дерево, свойства которого – одинаковая черная высота (количество черных узлов от корня до любого листа) и отсутствие у красного родителя красных дочерних узлов. Класс содержит структуру – наследник структуры с приватным полем – цветом узла. поддерживает операции чтения, изменения цвета узла. Класс включает в себя классы , наследника , , наследника , реализует правило пяти.

Класс . Реализует и переопределяет методы:

: возвращает размер .

: инициализирует выделенный участок памяти как объект структуры .

: возвращает указатель на дядю узла – поддерево деда, отличное от родительского текущего узла.

: производит балансировку дерева после вставки. Если текущий узел – корень дерева, меняет цвет этого узла на черный. Иначе находит родителя, деда и дядю этого узла. Балансировка требуется, только если родитель текущего узла – красный.

* дядя – красный: меняет цвет родителя и дяди на черный, деда – на красный. Вызывает метод от деда.
* дядя – черный, родитель – левое поддерево:
  + текущий узел – правое поддерево: производит малый левый поворот относительно родительского и текущего узлов. Меняет текущий узел с родительским, после чего переходит к следующему пункту (текущий узел – левое поддерево).
  + текущий узел – левое поддерево: производит малый правый поворот относительно родителя и деда, окрашивает родителя в черный, а деда – в красный.
* дядя – черный, родитель – правое поддерево:
  + текущий узел – левое поддерево: производит малый правый поворот относительно родительского и текущего узлов. Меняет текущий узел с родительским, после чего переходит к следующему пункту (текущий узел – правое поддерево).
  + текущий узел – правое поддерево: производит малый левый поворот относительно родителя и деда, окрашивает родителя в черный, а деда – в красный.

: меняет цвет добавленного узла на красный. Вызывает метод .

Класс . Реализует и переопределяет методы:

: обменивает значения двух узлов

: метод выполняет удаление из дерева, в зависимости от количества детей удаляемого узла. При наличии двух потомков – находим максимально правый элемент в левом поддереве, обмениваем его с удаляемым вплоть до цвета узлов, переходим к случаю с одним и отсутствием потомков. Согласно свойствам красно-черного дерева, не может быть красного узла с одним потомком, поэтому при его наличии вызывается метод для удаляемого узла и корректируется ссылка из родительского для удаляемого узла на узел-потомок, последний окрашивается в черный. Если потомков у удаляемого нет, вызывается метод для него, и, если удаляемый узел – черный, вызывается метод .

: находит родителя и деда узла. Пусть удаляемый узел был левым поддеревом. Случай, когда удаляемый узел – правое поддерево зеркален.

* Если брат – красный, то производит малый левый поворот относительно брата и родителя, перекрашивает родителя в красный, брата – в черный. Переходит к следующему шагу.
* Если брат текущей вершины черный, то получает три случая:
  + Оба ребёнка у брата чёрные. Перекрашивает брата в красный цвет, отца в черный.
  + Если у брата левое поддерево – красное, производит малый правый поворот относительно брата и его левого поддерева, перекрашивает брата в черный и его правого сына в красный. Переходит к следующему шагу.
  + Если у брата правое поддерево – красное, выполняет малый левый поворот относительно брата и отца, перекрашивает брата в цвет отца, правое поддерево брата и отца – в черный.

# Описание примененных паттернов

“Строитель” – порождающий паттерн, порождающий объекты. Отделяет конструирование сложного объекта от его представления, так что в результате одного и того же процесса конструирования могут получаться разные представления. При его использовании алгоритм создания сложного объекта не зависит от того, из каких частей состоит объект и как они стыкуются между собой; процесс конструирования обеспечивает разные представления конструируемого объекта. Паттерн позволяет изменять внутреннее представление продукта, изолирует код, реализующий конструирование и представление, дает более тонкий контроль над процессом конструирования.

“Приспособленец” – структурный паттерн. Использует разделение для эффективной поддержки множества мелких объектов. Используется, когда в приложении используется большое количество объектов, из-за чего возникают высокие накладные расходы на их хранение, большую часть состояния объектов можно вынести вовне, многие группы объектов можно заменить относительно небольшим количеством разделяемых объектов и приложение не зависит от идентичности объекта. Таким образом, применив паттерн “Приспособленец”, получим значительную экономию памяти.

“Одиночка” – порождающий паттерн. Гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа. “Одиночка” предоставляет контролируемый доступ к единственному экземпляру, уменьшает числа имен, допускает уточнение операций и представления, переменное число экземпляров и дает большую гибкость, чем у операций класса.

“Команда” – поведенческий паттерн. Инкапсулирует запрос как объект, позволяя тем самым задавать параметры клиентов для обработки соответствующих запросов, ставить запросы в очередь или протоколировать их, а также поддерживать отмену операций. “Команда” разрывает связь между объектом, инициирующим операцию, и объектом, имеющим информацию о том, как ее выполнить. Из простых команд можно собирать более составные, добавлять новые команды легко, поскольку никакие существующие классы изменять не нужно.

“Цепочка обязанностей” – поведенческий паттерн. Позволяет избежать привязки отправителя запроса к его получателю, давая шанс обработать запрос нескольким объектам. Связывает объекты-получатели в цепочку и передает запрос вдоль этой цепочки, пока его не обработают. “Цепочка обязанностей” позволяет ослабить связанности, дает дополнительную гибкость при распределении обязанностей между объектами.

“Шаблонный метод” – поведенческий паттерн, определяющий основу алгоритма и позволяющий подклассам переопределить некоторые шаги алгоритма, не изменяя его структуру в целом.

“Итератор” – поведенческий паттерн. Предоставляет способ последовательного доступа ко всем элементам составного класса, не раскрывая его внутреннего представления. “Итератор” позволяет поддерживать различные виды обхода агрегата, упрощает интерфейс агрегата.

# Руководство пользователя

После выполнения команд из файла, путь к которому подан на вход программе как аргумент командной строки, пользователю будет предложен интерактивный диалог с приложением, изложена справочная информация по программе.

|  |
| --- |
| Листинг 20. Справочная информация о программе |
| Hello! Would you like to get some closer interaction with my program? Print y for yes an n for no  >>y  ----- Course work help -----  Collection commands list:  Full path to collection: <pool\_name>/<scheme\_name>/<collection\_name>  - add  - find  - find dataset  - find dataset DD/MM/YYYY hh/mm/ss  - find DD/MM/YYYY hh/mm/ss  - update  - delete  Structural and customization commands list:  Supported trees: BST (binary tree), AVL, SPLAY, RB (red-black tree)  Supported allocators: global, sorted\_list best||worst||first, descriptors best||worst||first, buddy\_system  - add <tree type> <allocator type> <size of allocator> <full path>  one can note tree/allocator type only  if one wants to note an allocator, it's needed to also node desired length in bytes (not for global one)  - delete <full path>  DB commands list:  - help  - delete DB  - exit |

Данные о встречах в рабочем календаре (id встречи, 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))

Для всего пользовательского ввода одинаковы:

* формат полного пути к пулу/схеме/коллекции: <pool\_name>/<scheme\_name>/<collection\_name>. При добавлении/удалении пула полный путь – <pool\_name>, схемы – <pool\_name>/<scheme\_name>
* формат записи ключа: <applicant id>, <contest id>
* формат записи нового значения:
  + 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)

# Вывод

В результате выполнения курсового проекта было реализовано приложение, позволяющее выполнять операции над коллекциями данных заданных типов и контекстами их хранения. Реализован класс , репрезентирующий многоуровневую структуру, состоящую из пулов, схем, коллекций. Выполнено основное задание курсовой работы 0 и дополнительные задания 1, 2, 4, 6 (без реализации и деревьев), 7. На языке С++ в курсовой работе мною были реализованы такие структуры данных, как:

* аллокаторы: аллокатор с освобождением в рассортированном списке, аллокатор с освобождением с дескрипторами границ, система двойников.
* деревья: бинарное дерево поиска, AVL-дерево, косое дерево, красно-черное дерево.

Использованы паттерны строитель, приспособленец, одиночка, команда, цепочка обязанностей, шаблонный метод, итератор. В процессе выполнения курсовой работы были изучены и применены парадигмы ООП, разработаны алгоритмы системы хранения и управления данными на основе динамических структур данных.

# 

# Список использованных источников

1. Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му. Язык программирования С++. Базовый курс. Пятое издание.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. — СПб.: Питер, 2020. — 448 с.: ил. — (Серия «Библиотека программиста»).
3. B-tree [Электронный-ресурс]. Режим доступа: <https://www.programiz.com/dsa/b-tree> . (Дата обращения: 25.08.2023).
4. Паттерн проектирования “Цепочка обязанностей” [Электронный ресурс]. Режим доступа: <https://radioprog.ru/post/1491> . (Дата обращения: 31.08.2023).
5. Паттерн проектирования “Команда” [Электронный ресурс]. Режим доступа: <http://cpp-reference.ru/patterns/behavioral-patterns/command/> . (Дата обращения: 01.09.2023).

# Приложение:

Листинг кода находится в репозитории на интернет-ресурсе github.com по ссылке:  
 [moroooozov/Course\_project\_Morozov (github.com)](https://github.com/moroooozov/Course_project_Morozov) [актуально но момент 01.09.2023]