

В.А.Кирюхин

РТУ МИРЭА, БК-252

v.0.4.2

11 февраля 2021 г.

Лабораторные работы по курсу
«Методы программирования»
II семестр

Москва, 2021

Общие сведения

Представленный список лабораторных работ является примерным содержанием практической части курса «Методы программирования» за II семестр.

По выбору студента лабораторные могут выполняться на языках C++, Java, C#, Python. Применение иных языков согласовывается в индивидуальном порядке. Наименования файлов и некоторые иные аспекты (шаблоны, деструкторы, указатели и т.д.) приведены для языка C++ и должны заменяться на аналогичные при использовании любого иного языка.

Лабораторная работа в отведённый срок загружается в соответствующий раздел СДО – в виде архива с именем `Фамилия_Имя_ЛР№n_ver1` или ссылки на свой публичный репозиторий (<https://github.com>, <https://gitlab.com> и т.д.). Вопросы по лабораторным работам можно задавать по почте bk252-mp2016@ya.ru

Поиск ошибок и опечаток в настоящем документе вознаграждается дополнительными баллами.

Список лабораторных может дополняться и корректироваться в течение семестра!

Лабораторная работа №1 – Очередь с приоритетом

Задача: реализовать контейнер-адаптер `priority_queue`. Принцип работы очереди: первым извлекается наибольший элемент. Контейнер должен обеспечивать логарифмическое время работы для добавления и удаления элемента.

Рассматриваемые вопросы:

- использование композиции;
- паттерн проектирования «Адаптер»;
- шаблоны с параметрами по умолчанию;

Указания по выполнению:

- лабораторная работа состоит из двух файлов:

priority-queue.h – описание класса `priority_queue`, прототипы методов, реализация методов;

test-queue.cpp – тестирование и проверка возможностей класса;

- класс `priority_queue` содержит два шаблонных параметра: хранимый тип данных, используемый контейнер (по умолчанию динамический массив стандартной библиотеки)

- класс `priority_queue` содержит одно поле — используемый контейнер;

- хранение элементов организовать в виде двоичной кучи (пирамиды);

- для определения приоритета элемента пользоваться оператором `<` или эквивалентным методом (`compareTo` и т.п.);

- класс должен содержать следующие методы: проверка очереди на пустоту; получение числа элементов в очереди; добавление элемента в очередь; удаление элемента из очереди; доступ к максимальному элементу очереди;

Рекомендуемые материалы:

[3] – Глава 9. Очереди по приоритетам и пирамидальная сортировка

Лабораторная работа №2 – Контейнер map

Задача: реализовать ассоциативный массив (контейнер map) на основе двоичного дерева поиска (красно-черного или AVL). Каждый узел двоичного дерева должен содержать пару ключ-значение. Пользователь должен иметь возможность: получить значение по ключу, изменить значение по ключу, добавить в контейнер новую пару.

Рассматриваемые вопросы:

- двоичное дерево поиска, красно-черное дерево, AVL-дерево;
- паттерн проектирования «Итератор»;

Указания по выполнению:

- лабораторная работа состоит из двух файлов:
map.h – описание класса `map`, прототипы методов, реализация методов;
test-map.cpp – тестирование и проверка возможностей класса;
- создать вспомогательную шаблонную структуру `node`, содержащую поле для хранения значения произвольного типа и 3 поля для хранения указателей на родителя и левого/правого потомков;
- создать шаблонный класс `comparator` – функтор, который принимает два аргумента произвольного типа, возвращает логическое значение: `true`, если первый аргумент меньше; `false` в противном случае. *Реализация данного класса может быть опущена при отсутствии языковой поддержки;*
- класс `map` содержит три шаблонных параметра: тип ключа; тип значения; тип критерия сравнения (по умолчанию `comparator`);
- каждый узел дерева в поле данных содержит пару ключ-значение;
- класс `map` содержит одно поле – указатель на узел, являющийся корнем дерева.
- класс `map` должен содержать следующие методы:
конструктор, деструктор;
конструктор копирования;
копирующий оператор присваивания;
проверка на пустоту;
удаление всех элементов;
добавление пары ключ-значение;
оператор `[]` или соотв. метод – получает ключ, возвращает ссылку на значение;

- поиск по ключу – возвращает логическое значение или итератор;
- при необходимости разделять методы на интерфейс и реализацию;

Дополнительные задания:

- реализовать поддержку итераторов;
- реализовать методы удаления элемента по итератору и по ключу;

Рекомендуемые материалы:

- [1] – 4.3.3 Сбалансированные деревья поиска
- [1] – 4.3.4. Красно-чёрные деревья
- [3] – 13.4 Красно-черные деревья, или RB-деревья
- [5] – Глава 5. Паттерны поведения. Паттерн Iterator
- [6] – Глава 13. Красно-черные деревья

Лабораторная работа №3 – Контейнер `btree_map`

Задача: реализовать ассоциативный массив (контейнер `btree_map`) на основе В-дерева. Каждый узел дерева должен содержать набор пар ключ-значение. Пользователь должен иметь возможность: получить значение по ключу, изменить значение по ключу, добавить в контейнер новую пару.

Рассматриваемые вопросы:

- деревья поиска во внешней памяти, В-дерево;

Указания по выполнению:

- лабораторная работа состоит из двух файлов:

`btree_map.h` – описание класса `btree_map`, прототипы методов, реализация методов;

`test-map.cpp` – тестирование и проверка возможностей класса;

- создать вспомогательную шаблонную структуру `node`, содержащую два поля (контейнер с парами {ключ, значение}, контейнер с указателями на дочерние структуры `node`);

- создать шаблонный класс `comparator` – функтор, который принимает два аргумента произвольного типа, возвращает логическое значение: `true`, если первый аргумент меньше; `false` в противном случае. *Реализация данного класса может быть опущена при отсутствии языковой поддержки;*

- класс `btree_map` содержит три шаблонных параметра: тип ключа; тип значения; тип критерия сравнения (по умолчанию `comparator`);

- класс `btree_map` содержит два поля (указатель на узел, являющийся корнем дерева; параметр $t \geq 2$);

- класс `btree_map` должен содержать следующие методы:

конструктор, деструктор;

конструктор копирования;

копирующий оператор присваивания;

проверка на пустоту;

удаление всех элементов;

добавление пары ключ-значение;

оператор `[]` или соотв. метод – получает ключ, возвращает ссылку на значение;

поиск по ключу – возвращает логическое значение или итератор;

- при необходимости разделять методы на интерфейс и реализацию;

- параметр t (минимальная степень В-дерева) определяется в конструкторе и не меняется в процессе работы;

- для поиска ключа в структуре **node** использовать двоичный поиск.

Дополнительные задания:

- реализовать поддержку итераторов;

- реализовать методы удаления элемента по итератору и по ключу;

Рекомендуемые материалы:

[3] – 16.3 В-деревья

[6] – Глава 18. В-деревья

Лабораторная работа №4 – Контейнер `hash_map`

Задача: реализовать ассоциативный массив (контейнер `hash_map`) на основе хеш-таблицы с разрешением коллизий по методу цепочек.

Рассматриваемые вопросы:

- хеш-функции и хеш-таблицы;
- методы разрешения коллизий;
- амортизационные оценки сложности;

Указания по выполнению:

- лабораторная работа состоит из двух файлов:

hash-map.h – описание класса `hash_map`, прототипы методов, реализация методов;

test-hash-map.cpp – тестирование и проверка возможностей класса;

- класс `hash_map` содержит три шаблонных параметра: тип ключа `K`; тип значения `V`; тип функции хеширования (по умолчанию стандартная функция используемого языка);

- каждый элемент контейнера – пара ключ-значение;

- для хранения элементов воспользоваться стандартными контейнерами динамический массив и связный список;

- класс `hash_map` содержит следующие поля:

- вектор со списками элементов;

- число элементов;

- число списков;

- вещественный коэффициент – максимальный уровень загрузки (по умолчанию равен 2.0);

- класс `hash_map` должен содержать следующие методы:

- конструктор;

- удаление всех элементов;

- добавление пары ключ-значение;

- оператор `[]` или соотв. метод – получает ключ, возвращает ссылку на значение;

- удаление элемента по ключу;

- получение числа элементов;

- получение и изменение коэффициента загрузки;

получение текущего уровня загруженности (число всех элементов к числу всех списков);

перехеширование всех элементов;

– если текущий уровень загруженности превышает максимально допустимый, выполнять перехеширование контейнера, увеличив его размер по формуле $2 \cdot size + 1$;

– при добавлении элемента e выбирать список по формуле: $hash(e) \bmod$ (число списков);

– метод перехеширования получает новое число списков, изменяет размер вектора, перераспределяет все элементы по спискам согласно формуле. Данный метод сделать закрытым. Не изменять состояние контейнера, если новое число списков меньше текущего;

Дополнительные задания:

– добавить в метод перехеширования возможность уменьшения числа списков. Если при уменьшении числа списков уровень загруженности превысит максимальный, уменьшить число списков только до максимально допустимого;

Рекомендуемые материалы:

[1] – 4.4 Хеширование

[3] – Глава 14. Хеширование

[6] – Глава 17. Амортизационный анализ

Лабораторная работа №5.а – Алгоритмы на графах

Задача: реализовать следующие функции для работы с графами:

- поиск в глубину;
- поиск в ширину;
- алгоритм Дейкстры;
- алгоритм Крускала;
- алгоритм Прима;
- алгоритм Флойда-Уоршалла;

Указания по выполнению:

- лабораторная работа состоит из трех файлов:

graph-a.h – описание функций;

graph-a.cpp – реализация функций;

test-graph-a.cpp – тестирование функций;

- формат ввода/вывода, структуры для хранения информации, интерфейсы функций выбираются студентом самостоятельно;

Дополнительные задания:

- осуществить оптимальный выбор формата ввода/вывода, структур для хранения информации, интерфейсов функций;
- выполнить лабораторную работу №5.б.

Рекомендуемые материалы:

- [4] – 18.2 Поиск в глубину
- [4] – 18.7 Поиск в ширину
- [4] – 20.3 Алгоритм Прима
- [4] – 20.4 Алгоритм Крускала
- [4] – 21.2 Алгоритм Дейкстры
- [6] – 25.2 Алгоритм Флойда-Уоршалла

Лабораторная работа №5.6 – Алгоритмы на графах

Задача: реализовать следующие функции для работы с графами:

- алгоритм Тарьяна для топологической сортировки;
- алгоритм Флёри;
- алгоритм поиска эйлерова цикла на основе объединения циклов;
- алгоритм Косарайю;

Указания по выполнению:

- лабораторная работа состоит из трех файлов:

graph-b.h – описание функций;

graph-b.cpp – реализация функций;

test-graph-b.cpp – тестирование функций;

- формат ввода/вывода, структуры для хранения информации, интерфейсы функций выбираются студентом самостоятельно;

Дополнительные задания:

- осуществить разумный выбор формата ввода/вывода, структур для хранения информации, интерфейсов функций;

Рекомендуемые материалы:

[4] – 17.7 Простые, эйлеровы и гамильтоновы пути

[4] – 19.8 Сильные компоненты в орграфах

[6] – 22.4 Топологическая сортировка

Лабораторная работа №6 – Алгоритмы поиска подстрок

Задача: реализовать функции для поиска подстрок по следующим алгоритмам:

- Бойера-Мура;
- Рабина-Карпа;
- Кнута-Морриса-Пратта;
- с помощью конечного автомата.

Указания по выполнению:

- лабораторная работа состоит из трех файлов:

substring.h – описание функций;

substring.cpp – реализация функций;

test-substring.cpp – тестирование функций;

- достаточно реализации трех алгоритмов из четырех;

– каждая функция должна получать два аргумента: ссылку на строку, в которой должен осуществляться поиск; искомую строку-образец; – каждая функция должна возвращать динамический массив с индексами всех вхождений образца;

Дополнительные задания:

- реализовать все четыре алгоритма.

Рекомендуемые материалы:

[1] – 4.6 Поиск подстрок

[6] – Глава 32. Поиск подстрок

Список литературы

- [1] Анашкина Н. В., Петухова Н. Н — Технологии и методы программирования. Курс лекций. УК №1789 — Москва, 2011
- [2] Г.Шилдт. С++: базовый курс, 3-е издание.: Пер. с англ. — М.: Издательский дом «Вильямс», 2011 — 624 с.
- [3] Р.Седжвик. Алгоритмы на С++.: Пер. с англ. — М.: ООО «И.Д.Вильямс», 2011. — 1056 с.
- [4] Р.Седжвик. Алгоритмы на С++. Алгоритмы на графах: Пер. с англ. — М.: ООО «И.Д.Вильямс», 2011. — 1056 с.
- [5] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. — Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил.
- [6] Кормен, Томас Х. и др. — Алгоритмы: построение и анализ, 3-е издание. : Пер. с англ. — М.: Издательский дом "Вильямс", 2013. — 1328 с. : ил. — Парал. тит. англ.