

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: Красно-черное дерево

Студент гр. 9304

Кузнецов Р.В.

Преподаватель

Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить красно-черные деревья. Реализовать красно-черное дерево на языке программирования C++.

Задание.

Вариант 28

БДП: красно-чёрное дерево; действие: 1) По заданной последовательности элементов Elem построить структуру данных определённого типа–БДП или хеш-таблицу; 2) Записать в файл элементы построенного БДП в порядке их возрастания; вывести построенное БДП на экран в наглядном виде.

Выполнение работы.

Для выполнения работы была реализована шаблонная структура RBNode, являющаяся структурной единицей красно-черного дерева. Она содержит три указателя: на левое, правое поддерево, и указатель на предка. Также имеется поле данных и цвет узла. Цвет принимает значения из перечисления RBColor: RED или BLACK.

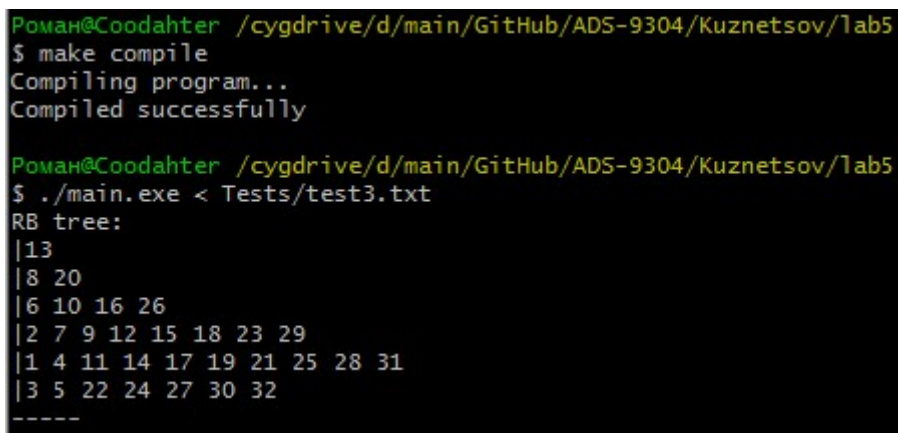
Основной функционал выполняет шаблонный класс RBTree. Он имеет указатель на корень, переменную, сохраняющую размер дерева, и несколько методов для работы с ним. Имеются публичные size(), возвращающий размер дерева, insert(), производящий вставку элемента в дерево, outputSorted() и outputLayers() для вывода элементов дерева в отсортированном порядке и слоями соответственно. Также имеются приватные методы, помогающие выполнить вставку элемента: RRight(), выполняющий правый поворот поддерева, RLeft(), выполняющий левый поворот поддерева, insert_1() и insert_2(), восстанавливающие свойства красно-черного дерева.

Алгоритм вставки заключается в том, что сначала мы, как и у обычного дерева поиска, вставляем элемент на подходящее место и красим его в красный, а потом восстанавливаем свойства красно-черного дерева:

1. Узел может быть либо красным, либо чёрным и имеет двух потомков;
2. Корень — как правило чёрный. Это правило слабо влияет на работоспособность модели, так как цвет корня всегда можно изменить с красного на чёрный;
3. Все листья, не содержащие данных — чёрные.
4. Оба потомка каждого красного узла — чёрные.
5. Любой простой путь от узла-предка до листового узла-потомка содержит одинаковое число чёрных узлов.

Также описаны вспомогательные функции получения дедушки и дяди для узла дерева.

На рисунке 1 представлен вывод дерева на экран в послойном виде.



```
Роман@Coodahter /cygdrive/d/main/GitHub/ADS-9304/Kuznetsov/lab5
$ make compile
Compiling program...
Compiled successfully

Роман@Coodahter /cygdrive/d/main/GitHub/ADS-9304/Kuznetsov/lab5
$ ./main.exe < Tests/test3.txt
RB tree:
|13
|8 20
|6 10 16 26
|2 7 9 12 15 18 23 29
|1 4 11 14 17 19 21 25 28 31
|3 5 22 24 27 30 32
-----
```

Рисунок 1 – послойный вывод полученного дерева в консоль.

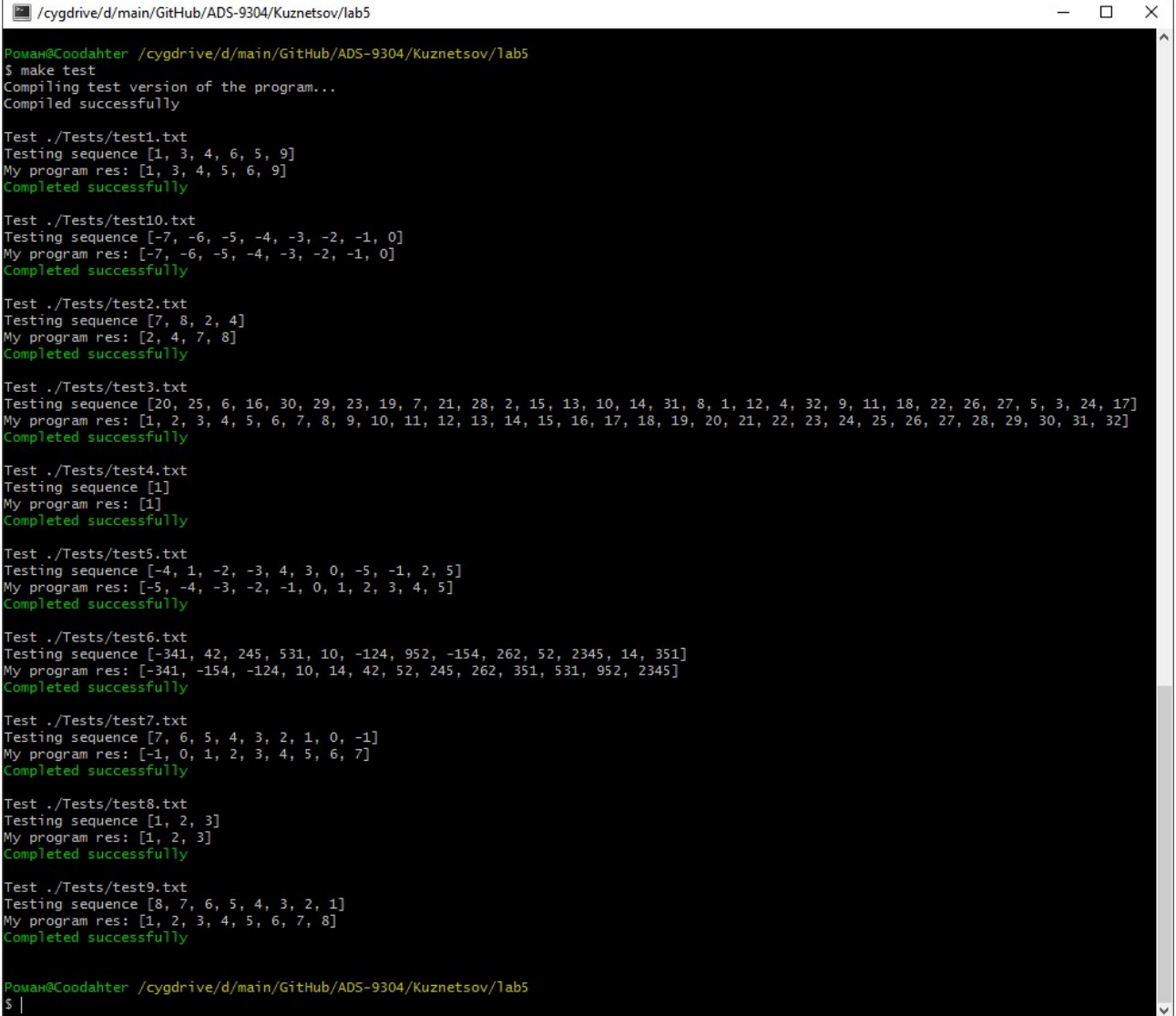
На экран данные выводятся послойно с помощью алгоритма обхода дерева в ширину (BFS), в консоль — с помощью алгоритма обхода дерева в глубину (DFS).

Тестирование

Для тестирования были написаны *python* и *bash* скрипты. *Python* скрипт принимает сначала вывод программы, затем входные данные (одним файлом, на разных строчках). После чего сортирует исходный массив сама и сравнивает с результатом работы программы. В случае соответствия выводится соответствующее сообщение, в случае несоответствия также выводится правильно отсортированный массив. *Bash* скрипт ответственен за поставку

тестовых данных из папки *Tests* и результата работы программы тестовому скрипту. Запуск тестов производится при помощи команды «make test», компиляция программы с помощью «make compile».

Пример запуска тестирующего скрипта представлен на рисунке 2.



```
/cygdrive/d/main/GitHub/ADS-9304/Kuznetsov/lab5
Povan@Coodahter /cygdrive/d/main/GitHub/ADS-9304/Kuznetsov/lab5
$ make test
Compiling test version of the program...
Compiled successfully

Test ./Tests/test1.txt
Testing sequence [1, 3, 4, 6, 5, 9]
My program res: [1, 3, 4, 5, 6, 9]
Completed successfully

Test ./Tests/test10.txt
Testing sequence [-7, -6, -5, -4, -3, -2, -1, 0]
My program res: [-7, -6, -5, -4, -3, -2, -1, 0]
Completed successfully

Test ./Tests/test2.txt
Testing sequence [7, 8, 2, 4]
My program res: [2, 4, 7, 8]
Completed successfully

Test ./Tests/test3.txt
Testing sequence [20, 25, 6, 16, 30, 29, 23, 19, 7, 21, 28, 2, 15, 13, 10, 14, 31, 8, 1, 12, 4, 32, 9, 11, 18, 22, 26, 27, 5, 3, 24, 17]
My program res: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
Completed successfully

Test ./Tests/test4.txt
Testing sequence [1]
My program res: [1]
Completed successfully

Test ./Tests/test5.txt
Testing sequence [-4, 1, -2, -3, 4, 3, 0, -5, -1, 2, 5]
My program res: [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
Completed successfully

Test ./Tests/test6.txt
Testing sequence [-341, 42, 245, 531, 10, -124, 952, -154, 262, 52, 2345, 14, 351]
My program res: [-341, -154, -124, 10, 14, 42, 52, 245, 262, 351, 531, 952, 2345]
Completed successfully

Test ./Tests/test7.txt
Testing sequence [7, 6, 5, 4, 3, 2, 1, 0, -1]
My program res: [-1, 0, 1, 2, 3, 4, 5, 6, 7]
Completed successfully

Test ./Tests/test8.txt
Testing sequence [1, 2, 3]
My program res: [1, 2, 3]
Completed successfully

Test ./Tests/test9.txt
Testing sequence [8, 7, 6, 5, 4, 3, 2, 1]
My program res: [1, 2, 3, 4, 5, 6, 7, 8]
Completed successfully

Povan@Coodahter /cygdrive/d/main/GitHub/ADS-9304/Kuznetsov/lab5
$ |
```

Рисунок 2 – Пример запуска тестирующего скрипта

Выводы.

В процессе работы ознакомились с красно-черными деревьями, реализовали программу, создающую красно-черное дерево на основе введенных элементов и выводящую полученное дерево на экран и в файл.