Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 5

Дисциплина: Низкоуровневое программирование

Тема: программирование на языке C

Вариант: 5

Выполнил студент гр. 3530901/90002 _____Е. В. Бурков
(подпись)

Принял преподаватель ___ _____ Д. С. Степанов
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург

2021

Содержание

Формулировка задачи

1.Разработать статическую библиотеку, реализующую определенный вариантом задания абстрактный тип данных.

2. Разработать демонстрационную программу – консольное приложение, обеспечивающее ввод данных из файла (файлов), их обработку и вывод в файл (файлы); имена файлов передаются в качестве параметров командной строки.

Требования к ПО

1. Язык разработки – C.

2. Реализация абстрактного типа данных должна использовать динамическое выделение памяти, при этом должна быть предусмотрена функция деинициализации, обеспечивающая освобождение всей выделенной памяти.

3. Библиотека и демонстрационная программа должны быть снабжены модульными тестами.

4. Разработанный исходный код должен компилироваться gcc без ошибок и предупреждений со следующими параметрами: -std=c11 -pedantic –Wall -Wextra.

5. Сборка библиотеки, демонстрационной программы и модульных тестов должна осуществляться утилитой make.

Вариант задания

Вариант 5: двоичная куча.

Двоичная куча или пирамида (англ. Binary heap) — такое двоичное подвешенное дерево, для которого выполнены следующие три условия:

- Значение в любой вершине не больше (если куча для минимума), чем значения её потомков.
- На i-ом слое 2^i вершин, кроме последнего. Слои нумеруются с нуля.
- Последний слой заполнен слева направо (как показано на рисунке)

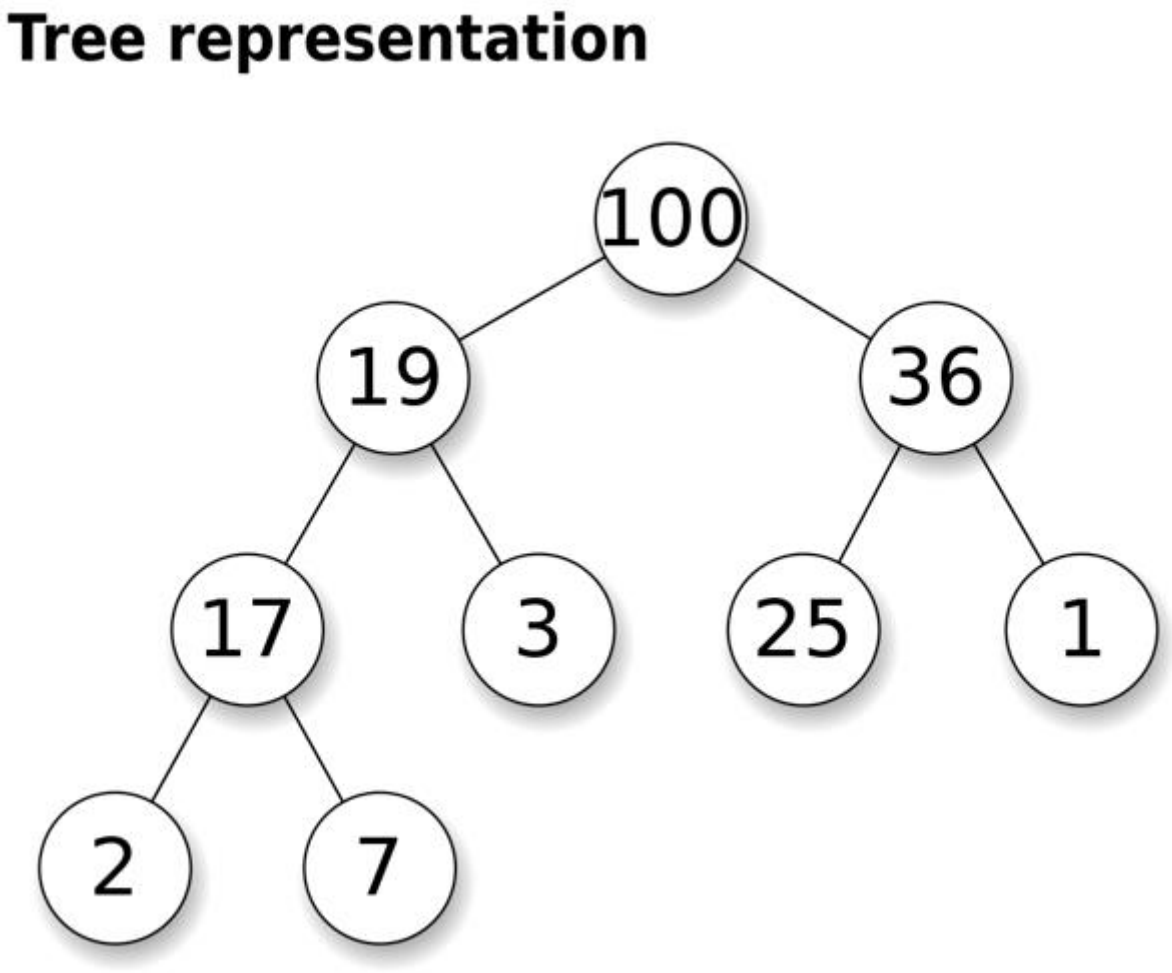

Рис. 1 Бинарная куча максимума

Описание реализованной библиотеки

Реализуемая куча представляет из себя динамическую структуру данных. Каждая куча хранит динамический массив, в котором находятся значения вершин.

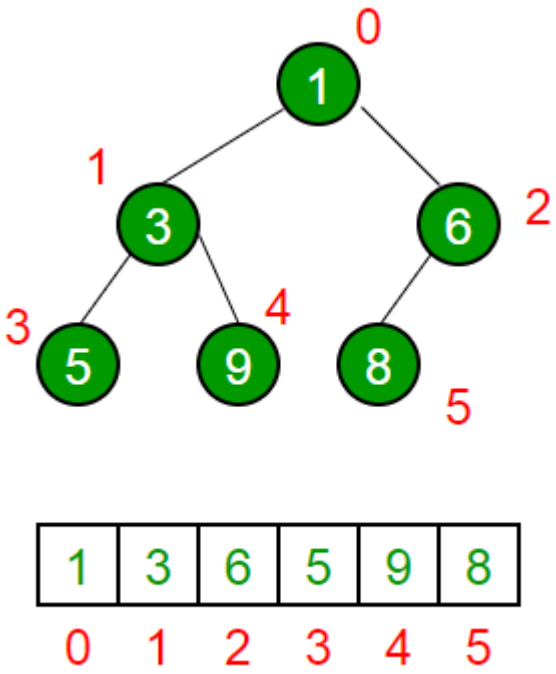

Рис. 2 Представление кучи в массиве

Функции входящие в API библиотеки помечены жёлтым.

Основные структуры
typedef int key_heap; typedef unsigned int value_heap; typedef struct { key_heap key; value_heap value; } pair_heap; typedef struct { pair_heap *array; size_t size; unsigned int data; bool max; } heap;

Была выделена структура для пары ключ + значение и структура самой кучи. В ней указать на массив, в котором находятся вершины кучи, размер кучи, размер выделенной памяти и булева, которая отвечает за вид кучи (max/min).

Создание кучи
`heap *heapInit(const unsigned int start_data)`

Выделение памяти для структуры h и массива. Для массива размер равен start_data.

Создание max/min-кучи
`heap *maxHeap(const unsigned int start_data)` `heap *minHeap(const unsigned int start_data)`

Просеивание наверх
`void heapShiftUp(heap *h, size_t i)`

Данный метод необходим для восстановления свойств кучи, когда при добавлении элемента в конец кучи его надо поднять наверх.

Просеивание вниз
`void heapShiftDown(heap *h, size_t i)`

Данный метод необходим для восстановления свойств кучи, когда происходит извлекание корня (при данной процедуре на место корня ставится последнее число и его нужно просеять вниз, чтобы восстановить свойства кучи).

Добавление элемента в кучу
`void heapAdd(heap *h, pair_heap p)`

Извлечение корня из кучи
`pair_heap heapRoot(heap *h)`

Восстановление свойств кучи
`void buildHeap(heap *h)`

Построение кучи из массива
`heap *minHeapArray(pair_heap *p, size_t size)` `heap *maxHeapArray(pair_heap *p, size_t size)`

Пирамидальная сортировка
`void heapSort(key_heap *array, size_t size)`

Отсортировать массив. В данном методе используется куча минимума.

Деинициализация кучи
`void heapRemove(heap *h)`

Печать кучи на консоль
`void printHeap(heap *h)`

Описание форматов файлов, параметров командной строки

В процессе выполнения работы была выделена программа, использующая статическую библиотеку кучи. Список доступных команд доступны при вызове программы с опцией “—help”.

```
D:\projects\lowlevelprog\lowlevelprog\lab5\heap\src>heap --help
Heap sort [-i] [input file] [-o] [output file]
Sorting array of integers
If you use input from CMD print any letter in the end
Heap (-max|-min) tree [-i] [input file] [-o] [output file]
Print heap
Heap (-max|-min) insert [-i] [input file] [-o] [output file]
Deleting max or min key in heap
```

Рис. 3 Перечень доступных команд

Для каждого действия есть одинаковый формат ввода. Опции -i и -o задают входной и выходной файл соответственно. Если при вызове программы таких опций не было, то будет использоваться консоль. При вводе с консоли необходимо заканчивать ввод любой буквой.

Heap sort

Отсортировать исходный целых чисел массив и вывести его. Используется пирамидальная сортировка. Значения в массиве могут разделяться или пробелами, или отступами.

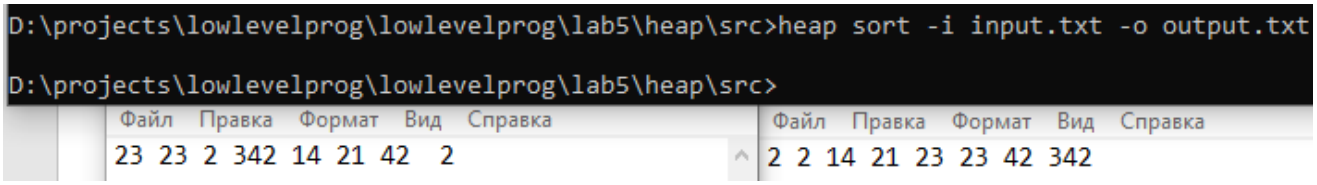

Рис. 4 Примеры работы Heap sort

Heap tree

Сформировать кучу из массива данных вида (ключ + значение) и вывести дерево данной кучи. Ключ и значение отделяются пробелами, пары отделяются отступами или пробелами.

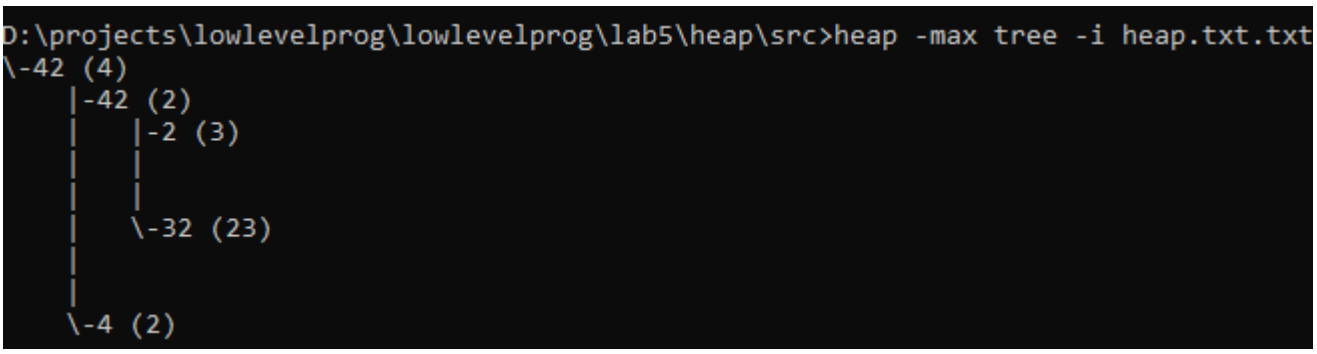

Рис. 5 Пример работы Heap tree

Heap extract

Изъять корень из кучи. Может использоваться для нахождения максимума минимума набора данных.

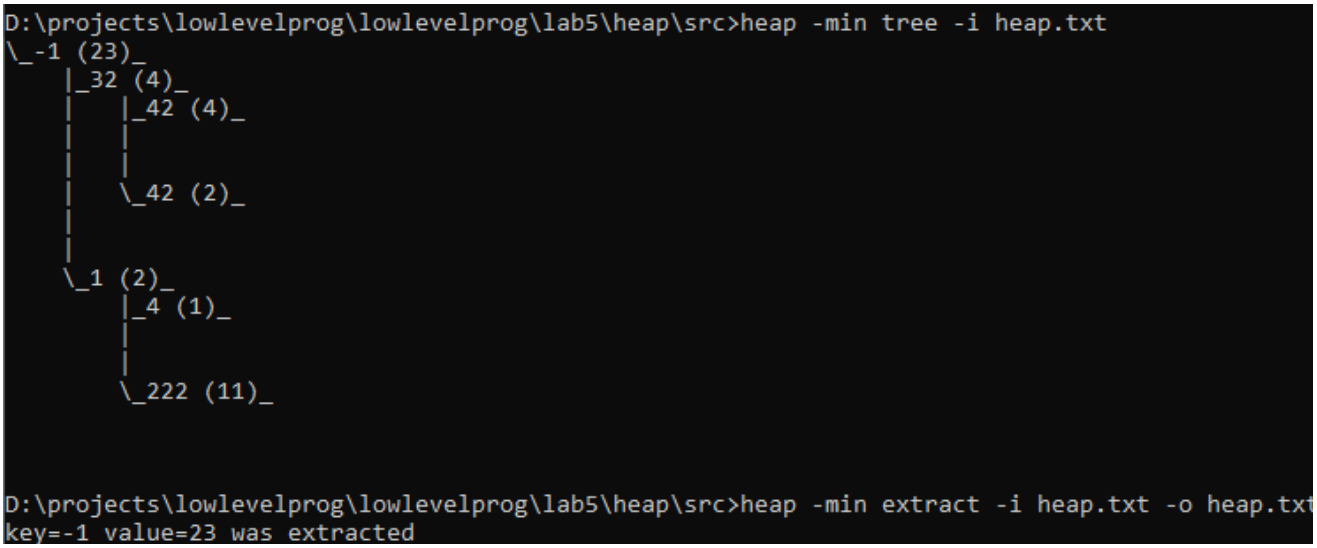

Рис. 6 Пример работы Heap extract

Руководство программиста

Для использования библиотеки и приложения необходимо клонировать git репозиторий проекта к себе на машину.

```
git clone https://github.com/wooftown/lowlevelprog.git
```

После при помощи утилиты make собираем библиотеку. Необходимо в корневом каталоге проекта в консоли выполнить:

```
make
```

(При использовании Linux заметь переменные RM, а также расширение исполняемых файлов). Для сборки приложения в корневой папке вводится:

```
make cmd
```

После этой команды запуститься мейк-файл из папки src, который проверит, собрана ли библиотека и после скомпилирует консольное приложение.

Для выполнения модульных тестов

```
make test
```

Данная команда запускает мейк-файл из папки test. Данный файл запускает модульные тесты и выводит их результаты в консоль.

Вывод

В процессе выполнения лабораторной работы была реализована статически линкуемая библиотека с функциональностью бинарной кучи. Была реализована консольная утилита, использующая нашу библиотеку. Были написаны модульные тесты. Возникли проблемы с форматом printf для типа size_t.

Список использованной литературы и источников

- Язык программирования Си 3-е издание Брайан Керниган, Деннис Ритчи
- C Interfaces and Implementations: Techniques for Creating Reusable Software by David Hanson
- https://neerc.ifmo.ru/wiki/index.php?title=Двоичная_куча
- https://github.com/ennorehling/cutest