САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Кафедра вычислительной техники

Отчёт по лабораторной работе № 1

по дисциплине «Тестирование программного обеспечения»

Вариант №776

Студенты:
Куклина М.
Кириллова А.

Преподаватель:
Клименков С.В.

Санкт-Петербург
2017 г.

Задание

1. Для указанной функции провести модульное тестирование разложения функции в степенной ряд. Выбрать достаточное тестовое покрытие.
   Функция sec().

2. Провести модульное тестирование указанного алгоритма. Для этого выбрать характерные точки внутри алгоритма, и для предложенных самостоятельно наборов исходных данных записать последовательность попадания в характерные точки. Сравнить последовательность попадания с эталонной.
   Программный модуль для работы с Фибоначчиевой кучей

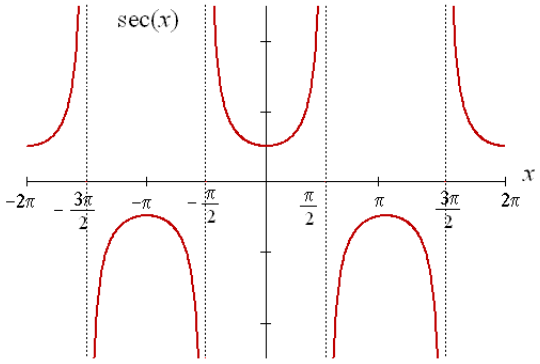
3. Сформировать доменную модель для заданного текста. Разработать тестовое покрытие для данной доменной модели
   В первый момент показалось, что ничего не произошло, затем что-то засветилось на краю огромного экрана. По нему ползла красная звезда величиной с тарелку, а следом за ней еще одна: бинарная звездная система. Затем в углу картинки возник большой полумесяц – красный свет, переходящий в черноту – ночная сторона планеты.

Тестирование функции sec()

Для создания тестового покрытия были выделены классы эквивалентности, то есть промежутки, где функция меняется одинаково:

1. $(\frac{-3\pi}{2} + 2\pi n; \frac{-\pi}{2} + 2\pi n), n \in Z$
2. $(\frac{-\pi}{2} + 2\pi n; \frac{\pi}{2} + 2\pi n), n \in Z$
3. Точки, в которых функция неопределена: $\frac{\pi}{2} + \pi n, n \in Z$.

Рис. 1. График функции



Тестирование модуля Fibonacci Heap

Функции модуля

Вставка

```
1: function Fib_Heap_Insert(H, x)
2:     degree[x] ← 0
3:     p[x] ← NIL
4:     child[x] ← NIL
5:     left[x] ← x
6:     right[x] ← x
7:     mark[x] ← falsa
8:     Merge roots list of x and H
9:     if min[H] = Nil or key[x] < key[min[H]] then
10:        min[H] ← x
11:    end if
12:    n[H] ← n[H] + 1
13: end function
```

Рис. 2. Вставка элемента 1

Рис. 3. Вставка элемента 2

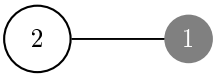


Рис. 4. Вставка элемента 3

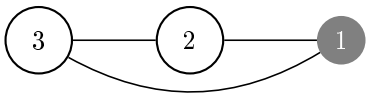
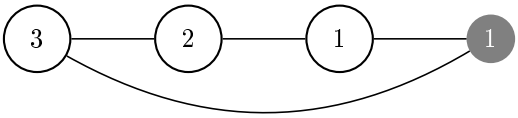


Рис. 5. Вставка элемента 0



Шаг	Ключ	min	Рисунок
0	x	null	Initialize heap.
1	1	1	Рис. 2
2	2	1	Рис. 3
3	3	1	Рис. 4
4	0	0	Рис. 5

Таблица 1. Эталонная таблица вставок.

Минимальный узел

```
1: function Fin_Minimum()
2:     return min
3: end function
```

Объединение двух куч

```
1: function Fib_Heap_Union(H_1, H_2)
2:     H ← Make_Fib_Heap()
3:     min[H] ← min[H_1]
4:     Add roots of H_2 to H.
5:     if min[H_1] = NIL or min[H_2] ≠ NIL and key[min[H_2]] < key[min[H_1]] then
6:         min[H] ← min[H_2]
7:     end if
8:     n[H] ← n[H_1] + n[H_2]
9:     return H
10: end function
```

Таблица 2. Последовательность объединеная пустой кучи и кучи

Шаг	Линия	Рисунок	Комментарий
1	x	Рис. 6	Даём функции два дерева; допустим, H_1 пустое.
2	4	Рис. 7	Добавляем корни H_2 в H.
3	5-9	Рис. 8	Так как $min[H_1] = NIL$, обновляем $min[H]$; возвращаем кучу.

Рис. 6. Этап 1: Куча.

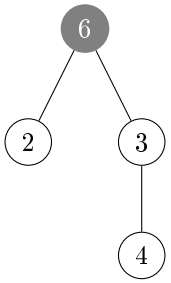


Рис. 7. Этап 2: Добавление корней в список H.

Рис. 8. Этап 3: Итоговая куча.

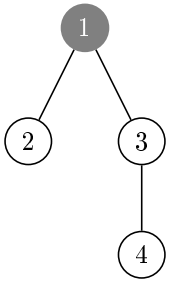


Рис. 9. Этап 1: Две кучи

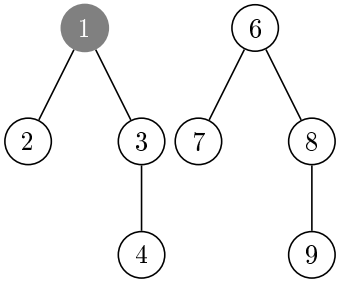
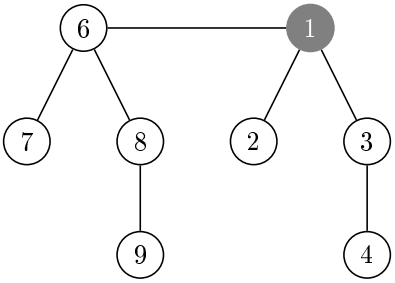


Рис. 10. Этап 2: Добавление корней H_2 в H

Рис. 11. Этап 3: Результатирующая куча



Извлечение минимального узла

1: **function** $Fib\_Heap\_Extract\_Min()$

Таблица 3. Последовательность объединения двух куч

Шаг	Линия	Рисунок	Комментарий
1	x	Рис. 9	Даём функции два дерева.
2	4	Рис. 10	Добавляем корни H_2 в H.
3	5-9	Рис. 11	Возвращаем кучу.

```
2:     z ← min[H]
3:     if z ≠ NIL then
4:         for Each child x of z do
5:             Add x to roots list.
6:             p[x] ← NIL
7:         end for
8:         Remove z from roots list.
9:         if z = right[z] then
10:            min[H] ← NIL
11:        else
12:            min[H] ← right[z]
13:            Consolidate(H)
14:        end if
15:    end if
16:    n[H] ← n[H] − 1
17: end function
1: function Consolidate()
2:     for i ← 0 to D(n[H]) do
3:         A[i] ← NIL
4:     end for
5:     for Each w in roots list of H do
6:         x ← w
7:         d ← degree[x]
8:         while A[d] ≠ NIL do
9:             y ← A[d]
10:            if key[x] > key[y] then
11:                swap(x, y)
12:            end if
13:            Fib_Heap_Link(H, y, x)
14:            A[d] ← NIL
15:            d ← d + 1
16:        end while
17:        A[d] ← x
18:    end for
19:    min[H] ← NIL
20:    for  doi ← 0 to D(n[H])
21:        if A[i] ≠ NIL then
22:            Add A[i] to roots list of H
23:            if min[H] = NIL or key[A[i]] < key[min[H]] then
24:                min[H] ← A[i]
25:            end if
26:        end if
27:    end for
28: end function
1: function Fib_Heap_Link(H, y, x)
2:     Remove y from roots list of H
3:     Merge y and child[x] list
4:     degree[x] ← degree[x] + 1
5:     mark[y] ← false
6: end function
```

Рис. 12. Начальная куча

Таблица 4. Последовательность извлечение минимального узла

Шаг	Линия	Рисунок	Комментарий
1	2-3	Рис. 12	Минимальное значение не равно NIL
2	4		поэтому просматриваем детей, одно их нет.
3	8		Затем удаляем элемент из списка корней.
4	9		Так как элемент был один, то его правый указатель указывает на него же.
5	10		Поэтому минимальный элемент обращается в NIL.

Рис. 13. Начальная куча

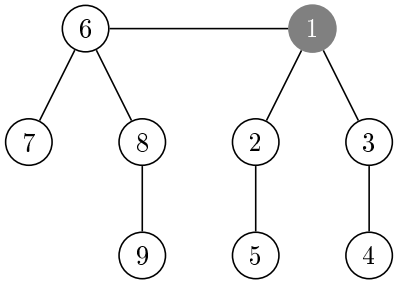


Рис. 14. Записываем детей в список корней.

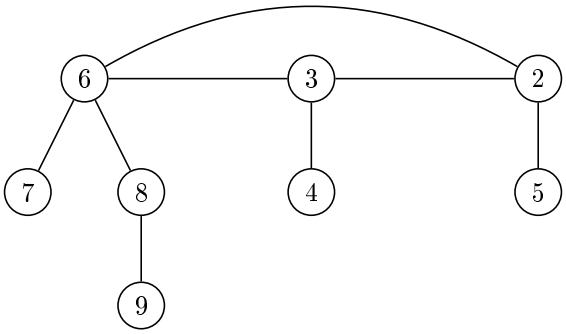
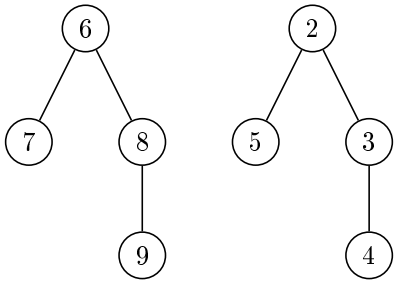


Рис. 15. Добавляем ноду 3 в список детей ноды 2



Уменьшение ключа

```
1: function Finc_Heap_Decrease_Key(H, x, k)
2:     if k > key[x] then
3:         error New key is bigger than old one.
4:     end if
5:     key[x] ← k
6:     y ← p[x]
```

Рис. 16. Добавляем ноду 6 в список детей ноды 2

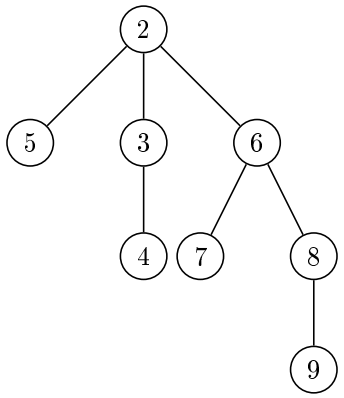


Таблица 5. Последовательность извлечение минимального узла

Шаг	Линия	Рисунок	Комментарий
1	2-3	Рис. 13	Минимальное значение не равно NIL
2	4-7	Рис. 14	Записываем детей минимальной ноды в список корней H
3	9-14		Правый указатель указывает не на себя же, поэтому меняем min и запускаем функцию $Consolidate(H)$
4	5-18	Рис. 15	Для нод с одинаковым уровнем делаем большую ноду ребёнком меньшей.
5	5-18	Рис. 16	Для нод с одинаковым уровнем делаем большую ноду ребёнком меньшей.
6	20-27	Рис. 16	Так как нода одна не происходит объединения.

```
7:     if y ≠ NIL and key[x] < key[y] then
8:         Cut(H, x, y)
9:         Cascading_Cut(H, y)
10:    end if
11:    if key[x] < key[min[H]] then
12:        min[H] ← x
13:    end if
14: end function
```

```
1: function Cut(H, x, y)
2:     Delete x from child list of y
3:     degree[y] ← degree[y] − 1
4:     Add x to roots list H
5:     p[x] ← NIL
6:     mark[x] ← false
7: end function
```

```
1: function Cascading_Cut(H, y)
2:     z ← p[y]
3:     if z ≠ NIL then
4:         if mark[y] = true then
5:             mark[y] = false
6:         else
7:             Cut(H, y, z)
8:             Cascading_Cut(H, z)
9:         end if
10:    end if
11: end function
```

Удаление элемента

```
1: function Fib_Heap_Delete()
2:     Fib_Heap_Decrease_Key(H, x, −∞)
3:     Fib_Heap_Extract_Min(H, x)
4: end function
```

Рис. 17. Начальная куча

Рис. 18. В функции $Cut()$

Рис. 19. Итоговая куча

Таблица 6. Последовательность изменения минимального узла

Шаг	Линия	Рисунок	Комментарий
1	5	Рис. 17	Уменьшим ноду 2 до 0.
2	5-6		Именяем ключ; записываем в y ноду 1.
3	7		Заходим в тело **if**.
4	8	Рис. 18	Заходим в функцию $Cut()$
5	9		Выходим из функции $Cascading\_Cut()$ без изменений.
6	12	Рис. 19	Так как $0 < 1$ меняем минимальный ключ.

Таблица 7. Последовательность удаления узла

Шаг	Линия	Рисунок	Комментарий
1	2		Уменьшаем ключ выбранной ноды на минимальное значение
2	3		Удаляем минимальный

Тестирование доменной модели для заданной области

Текст

В первый момент показалось, что ничего не произошло, затем что-то засветилось на краю огромного экрана. По нему ползла красная звезда величиной с тарелку, а следом за ней еще одна: бинарная звездная система. Затем в углу картинки возник большой полумесяц – красный свет, переходящий в черноту – ночная сторона планеты.

UML диаграмма

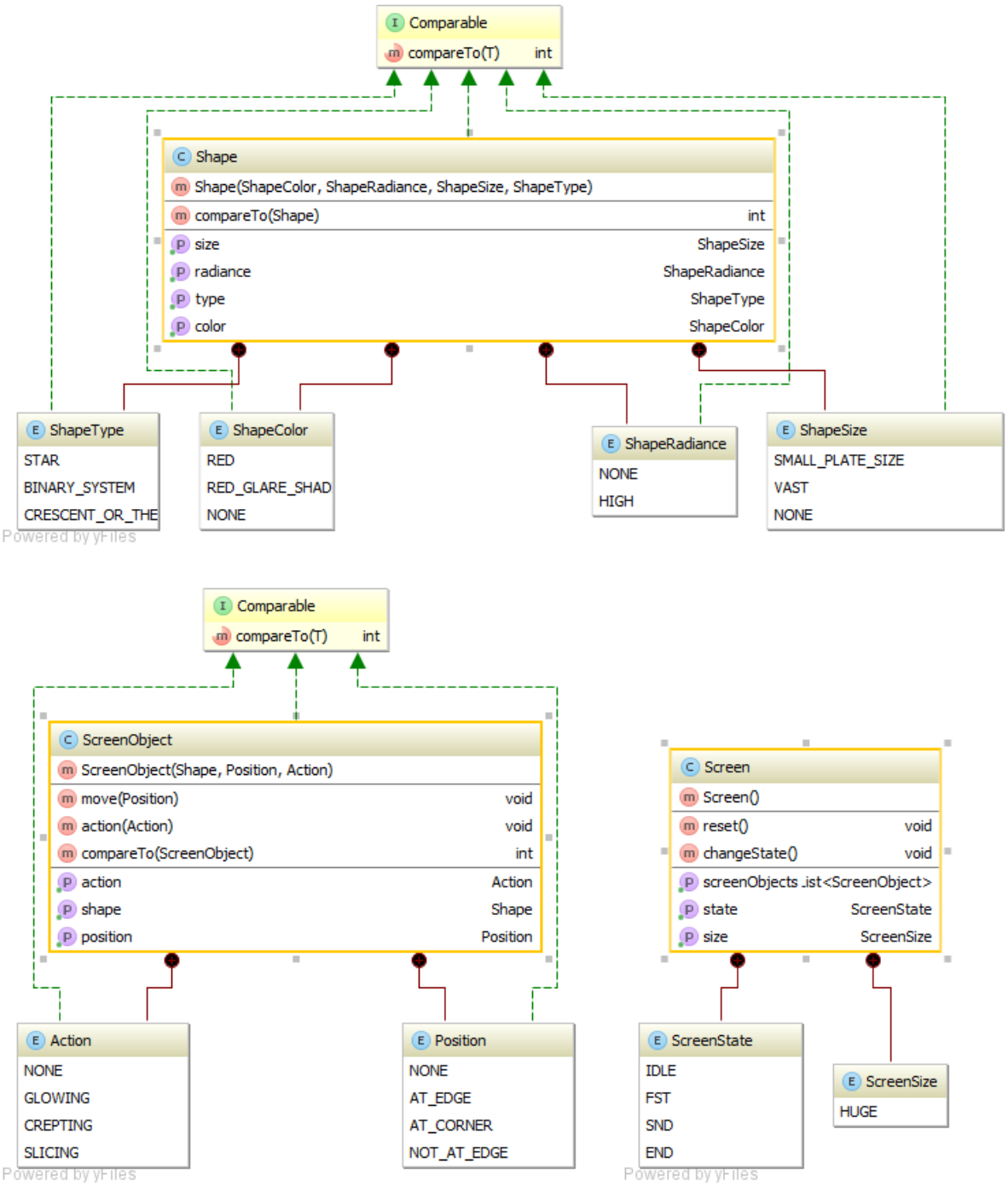


Рис. 20. UML диаграмма доменной модели

Вывод

В ходе выполнения лабораторных работ было проведено тестирование разработанных программных модулей. При выполнении работы использовались библиотеки JUnit4 и JUnit5. Явных отличий этих библиотек в данной работе отмеченно не было, разве что JUnit5 имеет иную иерахию классов и модульность, что делает его более гибким в сравнении с JUnit4, в котором все модули включены в платформу.