# Московский авиационный институт (национальный исследовательский университет)

## Факультет информационных технологий и прикладной математики

## Кафедра вычислительной математики и программирования


**Лабораторная работа №3 по курсу «Дискретный анализ»**


Студент: Е. А. Айрапетова
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

**Москва, 2021**

# Лабораторная работа №3

**Задача:** Необходимо провести исследование скорости выполнения и потребления оперативной памяти в программе по лабороторной работе 2. В случае выявления ошибок или явных недочётов, требуется их исправить.

Минимальный набор используемых средств должен содержать утилиту gprof и библиотеку dmalloc, однако их можно заменять на любые другие аналогичные или более известные утилиты (например, Valgrind или Shark) или добавлять к ним новые (например, gcov).

**Вариант дерева:** Красно-чёрное.

# 1 Описание

Результатом лабораторной работы является отчёт, состоящий из:

- Дневника выполнения работы, в котором отражено что и когда делалось, какие средства использовались и какие результаты были достигнуты на каждом шаге выполнения лабораторной работы.
- Выводов о найденных недочётах.
- Сравнение работы исправленной программы с предыдущей версией.
- Общих выводов о выполнении лабораторной работы, полученном опыте.

# 2 Дневник выполнения работы

Для выполнения работы:

- Для тестирования программы создадим файл benchmark.cpp, в котором содержится сравнение скорости работы моего дерева и std::map с помощью библиотеки <chrono>.
- С помощью valgrind оценим расход памяти и утечки программы.
- Выявим и оптимизируем участки кода, вызывающие утечки.
- На основе полученной информации сделаем выводы об эффективности программы по времени работы и занимаемой ею памяти.

# 3 Скорость работы

Программа представляет из себя тестировку одних и тех же данных на моем красно-чёрном дереве и std::map и сравнение времени выполнения:

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР2$ ./benchmark <file1.txt
std::map ms = 127
rb ms = 171
```

В данном примере тестирование происходит на 100 различных узлах.

# 4 Тестирование valgrind

Протестируем мою программу на этих же данных на утечки памяти.

```
jane@Evgenia:/mnt/c/Files/ДА/ЛР2/solution$ valgrind --leak-check=full --show-leak-kinc
./solution <file1.txt >trash.txt
==398== Memcheck,a memory error detector
==398== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==398== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==398== Command: ./solution
==398==
==398== error calling PR_SET_PTRACER,vgdb might block
==398==
==398== HEAP SUMMARY:
==398==     in use at exit: 0 bytes in 0 blocks
==398==   total heap usage: 411 allocs,411 frees,143,325 bytes allocated
==398==
==398== All heap blocks were freed --no leaks are possible
==398==
==398== For counts of detected and suppressed errors,rerun with: -v
==398== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

# 5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я познакомилась с таким иснструментом поиска утечек, как valgrind. Это средство является очень полезным в самых разных областях программирования. С помощью него можно не только проверить утечки в программе, но и понять, в какой части кода их искать.