Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное   
учреждение высшего профессионального образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

**Отчет по учебной практике**

**Лабораторная работа № 4**

**Имитационное моделирование системы обслуживания потока заданий на ЭВМ (очереди)**

**Выполнил** Ковальчук Александр 3822Б1ПР4

**Проверил:** Лебедев Илья Генадьевич

**Содержание**

[Введение 3](#__RefHeading__3039_1807997502)

[Постановка задачи 4](#__RefHeading__3041_1807997502)

[Руководство пользователя 5](#__RefHeading__3043_1807997502)

[Руководство программиста 6](#__RefHeading__3045_1807997502)

[Описание структуры программы 7](#__RefHeading__3047_1807997502)

[Описание алгоритмов 8](#__RefHeading__3049_1807997502)

[Заключение 9](#__RefHeading__3051_1807997502)

[Литература 10](#__RefHeading__3053_1807997502)

[Приложение 11](#__RefHeading__3055_1807997502)

# **Введение**

Динамические структуры занимают важное место в современной теории программирования. Динамическая структура есть математическая структура, которой соответствует частично-упорядоченное (по включению) базовое множество ***М,*** элементы которого являются структурами данных. При этом отношения включения формируются операциями преобразования структуры данных.

Очередь — динамическая структура

S=(Mi, P1, P2)

в которой:

* каждый элемент — структура,
* в любой момент существует только один конкретный элемент из  ***М***,
* элементы частично упорядочены по включению.

P1 – отношение следования, порождаемое операцией вставки,

P2 – отношение следования, порождаемое операцией исключение.

P1, P2 организованы по правилу FIFO или «первый вошел — первый вышел».

Лабораторная работа направлена на практическое освоение динамической структуры данных Очередь. С этой целью в лабораторной работе изучаются различные варианты структуры хранения очереди и разрабатываются методы и программы решения задач с использованием очередей.

В качестве области приложений выбрана тема эффективной организации выполнения потока заданий на вычислительных системах. Очередь характеризуется таким порядком обработки значений, при котором вставка новых элементов производится в конец очереди, а извлечение – из начала. Подобная организация данных широко встречается в различных приложениях. В качестве примера использования очереди предлагается задача разработки системы имитации однопроцессорной ЭВМ. Рассматриваемая в рамках лабораторной работы схема имитации является одной из наиболее простых моделей обслуживания заданий в вычислительной системе и обеспечивает тем самым лишь начальное ознакомление с проблемами моделирования и анализа эффективности функционирования реальных вычислительных систем.

# **Постановка задачи**

Для вычислительной системы (ВС) с одним процессором и однопрограммным последовательным режимом выполнения поступающих заданий требуется разработать программную систему для имитации процесса обслуживания заданий в ВС. При построении модели функционирования ВС должны учитываться следующие основные моменты обслуживания заданий:

* генерация нового задания;
* постановка задания в очередь для ожидания момента освобождения процессора;
* выборка задания из очереди при освобождении процессора после обслуживания очередного задания.

По результатам проводимых вычислительных экспериментов система имитации должна выводить информацию об условиях проведения эксперимента (интенсивность потока заданий, размер очереди заданий, производительность процессора, число тактов имитации) и полученные в результате имитации показатели функционирования вычислительной системы, в т.ч.

* количество поступивших в ВС заданий;
* количество отказов в обслуживании заданий из-за переполнения очереди;
* среднее количество тактов выполнения заданий;
* количество тактов простоя процессора из-за отсутствия в очереди заданий дляобслуживания.

# **Руководство пользователя**

Данная программа написана в среде разработки Microsoft Visual Studio 2019 на языке С++ и является консольным приложением. При запуске приложения, будет предложено инициализировать размер очереди, число тактов имитации, интенсивность потока задач и производительность процессора.

# **Руководство программиста**

## **Описание структуры программы**

Программа содержит один основной проект – Laba5.

Проект Laba5 содержит в себе модули TQueue.h, TClasster.h Модули содержат в себе объявление классов TQueue, соответственно. Реализация методов классов содержатся в TClasster.cpp

## **Описание алгоритмов**

**Алгоритмы класса TQueue**

1. TQueue(int \_size) – Конструктор-инициализатор
2. bool IsEmpty()– Метод проверки пустоты очереди.
3. bool IsFull() – Метод проверки полноты очереди.
4. void Push(T elem) – Функция кладет в начало очереди элемент и увеличивает top
5. T GetTop() – Возвращает последний элемент очереди
6. T Pop(); - Вдаляет последний элемент очереди
7. ~TQueue() - Деструктор
8. int GetLenght() – Возвращение длинны очереди

**Алгоритмы класса TClasster**

1. TClasster::TClasster() – Конструктор-инициализатор
2. void TClasster::StartNewJob(int \_size, int \_Proc, int \_tact,double \_ProcIntensity) – Этот метод начинает новую работу, инициализирует параметры задачи и обрабатывает её.
3. void TClasster::ProcessJob(TQueue<Task> &queue) – Метод обрабатывает задачи в очереди, управляет их выполнением и загрузкой процессоров
4. Task TClasster::TaskGeneration() – Метод генерирует новые задачи в соответствии с заданной интенсивностью.
5. int TClasster::GetGenerationTasks() – Счётчик общего кол-ва заданий
6. int TClasster::GetCountCompleteTask() – Счётчик кол-ва выполненных заданий
7. int TClasster::GetCountNotCompleteTask() - Счётчик кол-ва невыполненных заданий
8. int TClasster::GetCountTaskError() - Счётчик кол-ва заданий с ошибками
9. int TClasster::GetAverageLoadProc() – Средний процесс загрузки

# **Заключение**

Данная лабораторная работа познакомила нас с методом хранения информации «очередь», алгоритмами создания очереди, помещение и извлечение элементов на примере имитации работы процессора.

# **Литература**

1. http://old.itmm.unn.ru/most/obuchenie/uchebnye-materialy/

# **Приложение**