НИТУ «МИСиС»
Кафедра Инженерной кибернетики

Параллельные вычисления

Лабораторная работа №6. «Многопоточное программирование.
Клиент-серверная архитектура приложений,
front- и back-потоки исполнения»

Сенченко Р.В.

8 мая 2020 г.

I. Базовая часть. Необходимо разработать GUI приложение (от англ.: graphical user interface), реализующее аналитические функции работы с текстами.

Предусмотреть в приложении следующие возможности:

1. Открытие множества заданных текстовых файлов в формате .txt для их последующего анализа; открытие предусматривает вызов соответствующих диалоговых окон.

2. Решение аналитических задач: *поиск n самых часто-встречающихся слов, n задаётся пользователем посредством механизмов GUI*. При поиске необходимо учитывать разлчиные словоформы, падежи, числа и пр., для чего рекомендуется воспользоваться утилитой **MyStem** (Яндекс) для нормализации текста. Бинарный файл утилиты, а также документация доступны по ссылке *yandex.mystem*.

   Замечание. *Вызов утилиты **MyStem** должен производиться программным способо из основного кода приложения. Для удобства и упрощения работы рекомендуется изучить параметры утилиты, позволяющие привести её вывод в более приемлемый и удобный для последующей обработки вид. (Например, -l -d и др.; см. подробнее документацию.)*

3. Вывод результатов текстового анализа на GUI приложения. Структуру и дизайн пользовательского интерфейса продумать самостоятельно. Интерфейс должен выглядеть стройным и достойным звания простой текстовой утилиты.

Выполнение GUI приложения должно происходить в основном потоке исполнения, исполнение аналитических функций должно происходить в программно-создаваемых подчинённых потоках по запросу пользователя (т.е. по нажатию на соответствующую кнопку) в асинхронном режиме. Запуск исполнения аналитических функций не должен приводить к блокировке и “зависанию” GUI; соответствующие реакции элементов GUI должны быть грамотно прописаны (блокировки кнопок, информационные сообщения и пр.).

Реализованный алгоритм должен быть хорошо документирован и снабжён исчерпывающими комментариями. Программный язык разработки: c++, построение GUI предполагается и использованием фреймворка Qt.

II. Дополнительная часть.

1. **Progress bar, 3⋆.** Реализовать между основным и подчинённым потоками исполнения обмен информацией, позволяющей корректным образом отображать текущий статус исполнения аналитической функции (процент выполнения). Соответствующий статус отображать на элементе типа *Progress Bar*.

2. **Дерево N-грамм, 6-8⋆.** Расширить список аналитических задач Базовой части, добавив задачу построения дерева N-грамм; распространить на новую аналитическую задачу все требования базовой части: дерево должно строиться в подчинённом потоке исполнения, а также далее отображаться на GUI приложения.

   Замечание. Задача оценивается в дополнительные **2⋆** в том случае, если предусмотрена возможность отслеживания процента выполнения и его отображения на элементе типа *Progress Bar*.

3. **Система со-подчинённых потоков, 4⋆.** Организовать выполнение аналитических функций Базовой части лабораторной работы на основе системы со-подчинённых потоков параллельного исполнения: в единственном *управляющем потоке* и множестве *исполнительных потоков*.

   - *Управляющий поток* создаёт множество исполнительных потоков, отслеживает их статус и организует общую работу с потоками, ожидает до тех пор, пока все подчинённые исполнительные потоки не завершат свою работу.
   - *Исполнительные потоки* выполняют непосредственную обработку файлов; каждый поток обрабатывает один входной файл. Следует обратить внимание на то обстоятельство, что все исполнительные потоки должны записывать результат своей работы в общий (разделяемый) объект. Указанный объект необходимо спроектировать и разработать самостоятельно, а также защитить его от коллизий множественного параллельного доступа с помощью механизмов мьютексов.